Institute for Cognitive Systems
Technische Universität München
Prof. Gordon Cheng

Biosignal Processing and Modeling
Tutorial – Setting Up the Software and Hardware

Course Instructors: Alireza Malekmohammadi `alireza.malekmohammadi@tum.de`,
Nicolas Berberich, `n.berberich@tum.de`
Teaching Assistant: Karahan Yilmazer, `karahan.yilmazer@tum.de`

Summer Semester 2023

Introduction

This tutorial is aimed to help you set up every component you need for the tutorials. Even though it can be a cumbersome process, follow through the whole tutorial and set everything up, so that you do not have to do it again.

In the following, you will learn how to set up

- a Python virtual environment
- Visual Studio Code
- Unicorn Hybrid Black EEG headset
- Smarting 24 EEG headset
- Lab Streaming Layer (LSL) and LabRecorder

After you finish this tutorial, you can follow the other tutorials in the BCI Course Tutorials repository. Each tutorial is a sub-project on its own and will give you the intuition how to combine them all in your own project. Without further ado, let's get started!

Institute for Cognitive Systems
Technische Universität München
Prof. Gordon Cheng

1 Installing Python and Setting up a Virtual Environment

First off, you will have to install Python if you have not done so already. If you do not have a version preference, simply install the latest Python version (3.11.3).

For Windows, you can download the executable directly from the Python website. For Linux, you can refer to this short tutorial for different installation methods.

Once you have installed Python, you can set up a virtual environment for the course. This way all the packages you need will live in one virtual space without affecting your other Python projects. For that, open a terminal (PowerShell for Windows, bash for Linux) and follow the code snippet for your system below.

Setting Up A Virtual Environment (Windows)

```
PS C:\Users\yilma> py --version # Make sure Python is installed
Python 3.11.3
PS C:\Users\yilma> cd ..\..\Programming\Python\ # Navigate to the desired folder
PS C:\Programming\Python> mkdir venvs # Create a folder for virtual environments
PS C:\Programming\Python> cd .\venvs\ # Switch to the venvs folder
PS C:\Programming\Python\venvs> py -m venv neuro # Create a virtual environment
PS C:\Programming\Python\venvs> .\neuro\Scripts\activate # Activate the virtual environment
(neuro) PS C:\Programming\Python\venvs> deactivate # Deactivate the virtual environment
PS C:\Programming\Python\venvs>
```

Info:
If you receive the error "File C:\Programming\Python\venvs\neuro\Scripts\Activate.ps1 cannot be loaded because running scripts is disabled on this system.", you should change the execution policy of your system.

You can check your current execution policy using `Get-ExecutionPolicy`. If it is `Restricted`, you can change it to a different policy where you can run scripts on your system. You can do so by typing **`Set-ExecutionPolicy AllSigned`** in a PowerShell with Administrator privileges. You can read more about execution policies here.

Info:
In Windows, by using `py` instead of `python`, you automatically use the latest Python installation on your computer. If you want to use a different Python installation, you can do so by replacing `py` with `py -3.X`.

Setting Up A Virtual Environment (Linux)

```
karahan@yilma:~$ python3.10 --version # Make sure Python is installed
Python 3.11.3
karahan@yilma:~$ cd Python/ # Navigate to the desired folder
karahan@yilma:~/Python$ mkdir venvs # Create a folder for virtual environments
karahan@yilma:~/Python$ cd venvs # Switch to the venvs folder
karahan@yilma:~/Python/venvs$ python3.11 -m venv neuro # Create a virtual environment
karahan@yilma:~/Python/venvs$ source neuro/bin/activate # Activate the virtual environment
(neuro) karahan@yilma:~/Python/venvs$ deactivate # Deactivate the virtual environment
karahan@yilma:~/Python/venvs$
```

2 Installing Packages With Pip

Assuming that you have successfully set up your virtual environment, you can activate it once again to install the necessary packages for the tutorial. You can either use the `requirements.txt` file to automatically install all the necessary packages or by install all packages one by one.

You can find the `requirements.txt` file in the GitLab repository for the course under the folder `misc`. This is a file that contains a list of all the necessary packages and it can be easily managed by pip. Just make sure that you are in the folder that contains this file when you are running the corresponding line.

Installing the Necessary Packages Using requirements.txt

```
karahan@yilma:~/Python$ source venvs/neuro/bin/activate # Activate the virtual environment
    (Linux)
(neuro) karahan@yilma:~$ cd Python/bci-course-tutorials/misc # Navigate to the project
    folder
(neuro) karahan@yilma:~/Python/bci-course-tutorials/misc$ ls # List the contents of the
    folder
lsl_inlet.py
lsl_outlet.py
requirements.txt
(neuro) karahan@yilma:~/Python/bci-course-tutorials/misc$ pip install --upgrade pip
    setuptools wheel # Update the elementary modules
(neuro) karahan@yilma:~/Python/bci-course-tutorials/misc$ pip install -r requirements.txt #
    Install the necessary packages
```

Installing the Necessary Packages Manually

```
karahan@yilma:~/Python$ source venvs/neuro/bin/activate # Activate the virtual environment
    (Linux)
(neuro) karahan@yilma:~/Python$ pip install --upgrade pip setuptools wheel # Update the
    elementary modules (you can use py instead of python for Windows)
(neuro) karahan@yilma:~/Python$ pip install numpy matplotlib sklearn scipy screeninfo mne
    pyxdf pylsl pyqt5 pygame ... # Install the necessary packages
```

3 Setting up Visual Studio Code

Visual Studio Code is a powerful code editor, that can also be used as an IDE (integrated development environment). This means you can edit, run and debug your Python code using only VS Code. It has a marketplace for extensions, which further advances its usability.

To be able to run Python scripts in VS Code, you will have to install the **official Python extension** from the VS Code extensions marketplace. For that, click on the icon with four squares on the left side bar or press Ctrl+Shift+X to open the marketplace. Search for "python" and install the extension provided by Microsoft. You can find a screenshot of this procedure in Figure 1.

Next up, you will have to connect your virtual environment with VS Code. Click on **`File > Preferences > Settings`**. Then type "venvs" in the search bar. If nothing shows up you may have to restart VS Code. Copy and paste the path to your `venvs` folder to **`Python: Venv Path`**. You can refer to Figure 2 for this procedure.

With this, you will be able to choose your virtual environment in the bottom status bar. This way, you can quickly switch between different virtual environments. By switching to a virtual environment, you will be able to use all the packages installed in that virtual environment in your Python scripts.

The Python scripts you will find in the course repository will have special comments **`# %%`**. These special comments divide the code into cells which can be run separately in VS Code, just like a Jupyter Notebook. To be able to use this feature, you have to install the **official Jupyter extension** provided by Microsoft from the marketplace as well. You can simply type "jupyter" in the search bar to install it just like the Python extension.

Now, when you click on "Run Cell" or press Shift+Enter in a cell, a pop up might appear. It will read: "Running cells with 'Python 3.11.3' requires ipykernel package.". Simply click 'Install'. This will open up a terminal to install all the necessary pip packages. As a side note, you can use this built-in terminal, for example, to install different packages, without leaving VS Code.

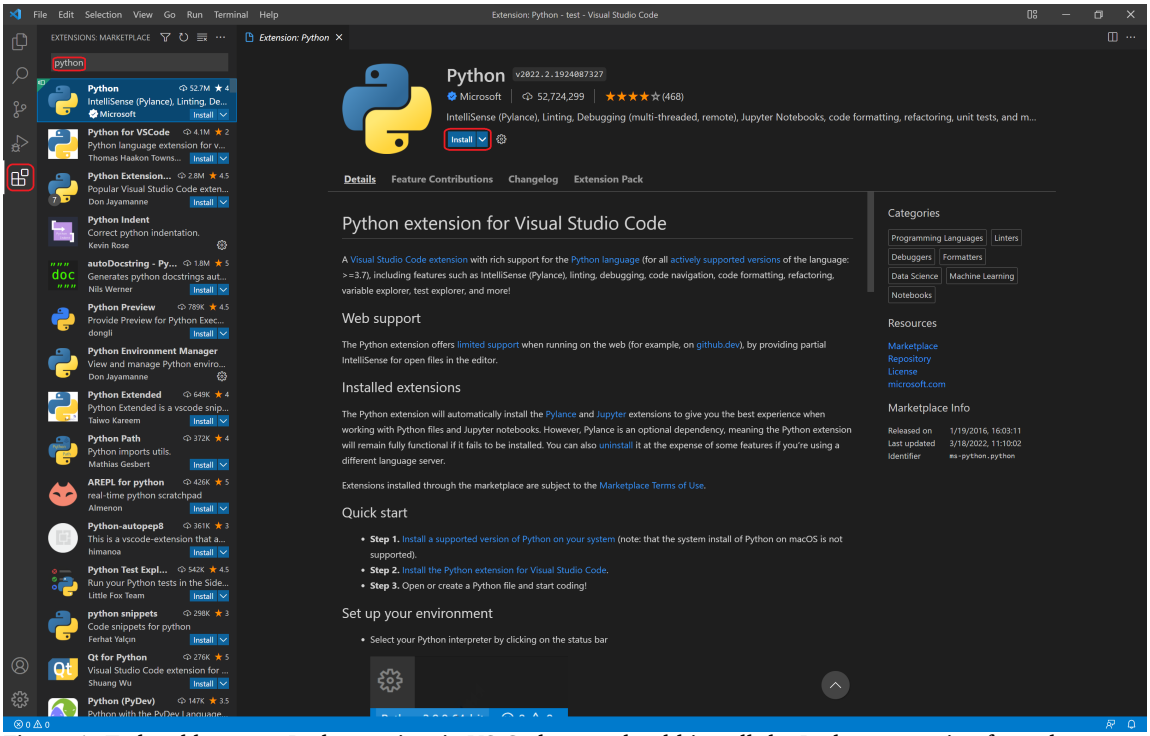

Figure 1: To be able to run Python scripts in VS Code, you should install the Python extension from the extensions marketplace. The red box on the left side bar shows the icon for the marketplace. By typing in "python" in the search bar, you can find and install the official Python extension.

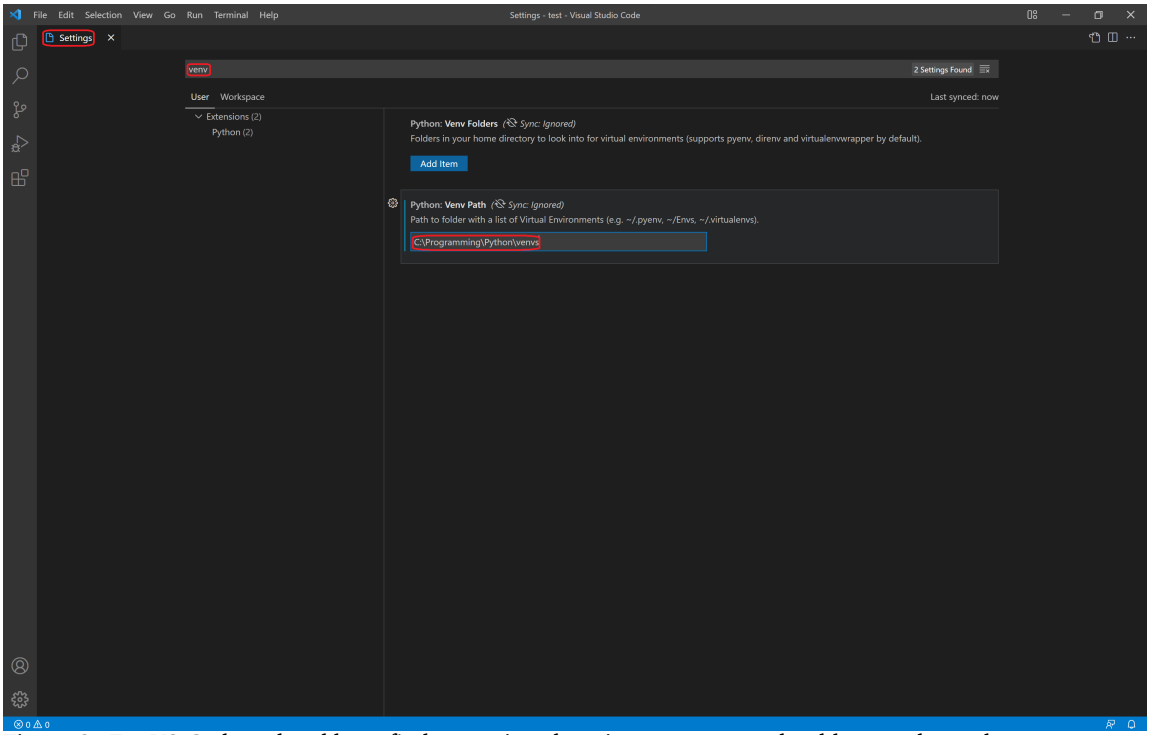

Figure 2: For VS Code to be able to find your virtual environments, you should paste the path to your venvs folder to `Python: Venv Path` in Settings.

4 Unicorn Hybrid Black

Before going on with how to access EEG data using Python, it makes sense to get familiar with the EEG system used for the course.

Unicorn Hybrid Black (UHB) is a wireless EEG system manufactured by g.tec medical engineering GmbH. The wireless communication is done through a Bluetooth 2.1 interface. It also has a 3-axis (x, y, z) accelerometer and a gyroscope to detect head movements.

UHB consists of eight conductive rubber electrodes (Fz, C3, Cz, C4, Pz, PO7, Oz, PO8) with 24 bits resolution, a sampling rate of 250 Hz per channel, and an input sensitivity of ± 750 mV. The electrodes are passive (the signal is not amplified on the electrode itself) and can be used either dry or wet.

A picture of the system and its sensor coverage can be seen in Figure 3.

Info:
For wet electrode recordings, an electrolyte gel is applied to each electrode. This lowers the impedance values of the electrodes, assuring that the signal-to-noise ratio (SNR) increases. Even though the impedance values of the electrode contacts are used as the main source of information for the signal quality in EEG studies, UHB does not come with built-in impedance measurements. Thus unfortunately, the user is left with a rough guess of the signal quality computed from two measures: the band power mean difference (BPMD) and the standard deviation (SD).

SD criterion: If the SD of the last 2 seconds of the signal of a channel is within 7-50 µV, the signal quality is assumed to be Good. If it is outside this range, it is assumed to be Poor.

BPMD criterion: The signal is notch filtered at 50 Hz and 60 Hz and band-pass filtered with cutoff frequencies of 0.1 Hz and 30 Hz. Then the mean signal amplitude of the 50 Hz and 60 Hz signal components are calculated with a band of ±2 Hz. Then the same is done for the 0.1-30 Hz frequency component. If the difference between the two mean values (i.e., band-pass minus notch) is greater than 0.1 µV, the signal quality is assumed to be Good. Otherwise, it is assumed to be Poor.

These criteria are then visualized in the Unicorn Suite Hybrid Black Software for each electrode to give the user a sense of signal quality. However, in this course, we will not be using this software but rather our own implementations of signal visualization.

As a side note, you will be seeing how impedance measurements are used to check the signal quality when you get introduced to the research-grade actiCHamp system.

Later on in the tutorials, when you are streaming data from the UHB, you will notice that there are 17 channels in the data stream. The first 8 channels correspond to EEG data from each electrode. The next 6 channels are the accelerometer (x, y, z) and gyroscope (x, y, z) data. Then you have 1 channel for the battery level, 1 channel for the sample counter (incremented with each new sample) and a final channel for validation indicator (if samples are lost during the data acquisition).

As you might guess, we will mainly be using only the first 8 channels. However, you might want to use the next 6 channels depending on your project.

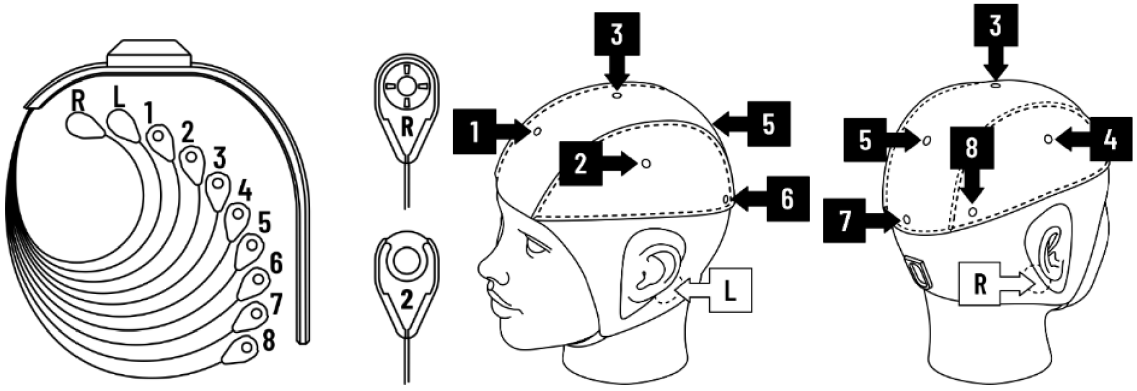

Figure 3: (left) EEG system without the cap attached. There are eight recording electrodes (1-8) and two mastoid reference electrodes (L and R). (middle and right) Diagram showing where each electrode belongs to on the elastic cap. If the recording electrodes are not already attached to circular rubber electrodes, they have to be slid in. The mastoid reference electrodes have to be clipped to the sticky electrodes placed on both mastoids.

Info:
EEG recordings, in general, are quite sensitive to several disturbances such as movements of the participant, the cables and electrodes, or coupling of electromagnetic and electrostatic fields in the recording environment. These disturbances (artifacts) are even more pronounced while doing dry electrode recordings with the UHB.

5 Lab Streaming Layer (LSL) for Unicorn Hybrid Black

With that out of the way, we can focus on how we can interface with EEG data. There are various ways to achieve this but for this course, we will be using a combination of Python and Lab Streaming Layer (LSL) with the help of the pylsl library.

LSL lets you access and record different data and marker streams on the same network as your computer. This is particularly useful when we are recording EEG data and time markers at the same time. But more on that later!

Unfortunately, streaming data from UHB through LSL is not very straightforward. However, it is a simpler task on Linux. So, if you are not using Linux, consider a dual-boot. In this case, you can simply skip to the "Linux" subsection.

5.1 Windows

First, you have to install the Unicorn Suite Hybrid Black. Even though we won't be using the program, the Unicorn LSL plugin needs this installation.

Find the **`Unicorn Software`** repository under **Shared Projects** in the course's GitLab group. Download the repository and navigate to the folder called **`unicorn-suite-windows`**. Then, run

setup.exe and simply follow the installer. Once you are done, go to your start menu, search for **Unicorn Suite Hybrid Black** and check if the program is installed correctly.

Now, you have to install the plugin that lets you stream data from Unicorn through LSL. If you prefer a video tutorial, you can follow this YouTube video.

Navigate back one folder and then to the folder called **`unicorn-lsl-windows`**. In this folder, you will find another folder called **`UnicornLSL`**. Copy this folder to **`Documents > gtec > Unicorn Suite > Hybrid Black > Unicorn DotNet > Examples`**.

At this point, you will have to install Visual Studio on your computer (not Visual Studio Code, but Visual Studio), in case you do not have it already. Open the file called **`UnicornLSL.sln`** in the `Unicorn LSL` folder that you newly copied into the **`Examples`** folder.

Then click on **`Build > Build Solution`**. Ideally, this will create an executable called **`UnicornLSL.exe`** in the **`Documents > gtec > Unicorn Suite > Hybrid Black > Unicorn DotNet > Examples > UnicornLSL > UnicornLSL > bin > x64 > Debug`**.

Info:
Depending on your security settings, a pop up might ask you for access while building the Visual Studio solution. Simply allow this for an error-free compilation.

For convenience, create a shortcut of this executable and move it to your desktop or to a folder that is easily accessible.

Now, disable Bluetooth and open **Device Manager**. In Device Manager you should be seeing a single component in the dropdown list for Bluetooth. Click on it and then on the "Disable Device" button (Figure 4). Note that the Bluetooth component's name does not have to match the one in the screenshot.

Now, plug in the Unicorn Bluetooth adapter. After doing this, you should be seeing different Bluetooth components popping up. At this point, you can close the Device Manager.

Go to your Bluetooth settings and click on "Add device". You should be able to see the UHB headset listed under the Bluetooth devices. Once you connect to the device, you will not have to go to Settings again to reconnect in future recording sessions.

At this point, you can open the Unicorn LSL application that you created a shortcut for. Choose your current headset from the dropdown menu. If you want, you can define a stream name, otherwise you can go with the default one. Then click on **Open**, which would create an LSL outlet. By clicking on **Start** you will start streaming data through LSL.

Congratulations, you are finally ready to stream EEG data from UHB and hopefully you will not need to go over this set up procedure again!

5.2 Linux

To stream EEG data from UHB through LSL using Linux, you must use the C API. All the source code, header files and libraries that you need are included under the **unicorn-c-linux** folder in the **unicorn-software** repository. You can find the repository under Shared projects in the GitLab group for the course.

In a terminal window navigate to `<root>/bci-course-tutorials/unicorn-c-linux`. From here, you need to build an executable from the source code. A sample Makefile is included which compiles the source code, links the necessary libraries, and finally generates an executable program. (Note

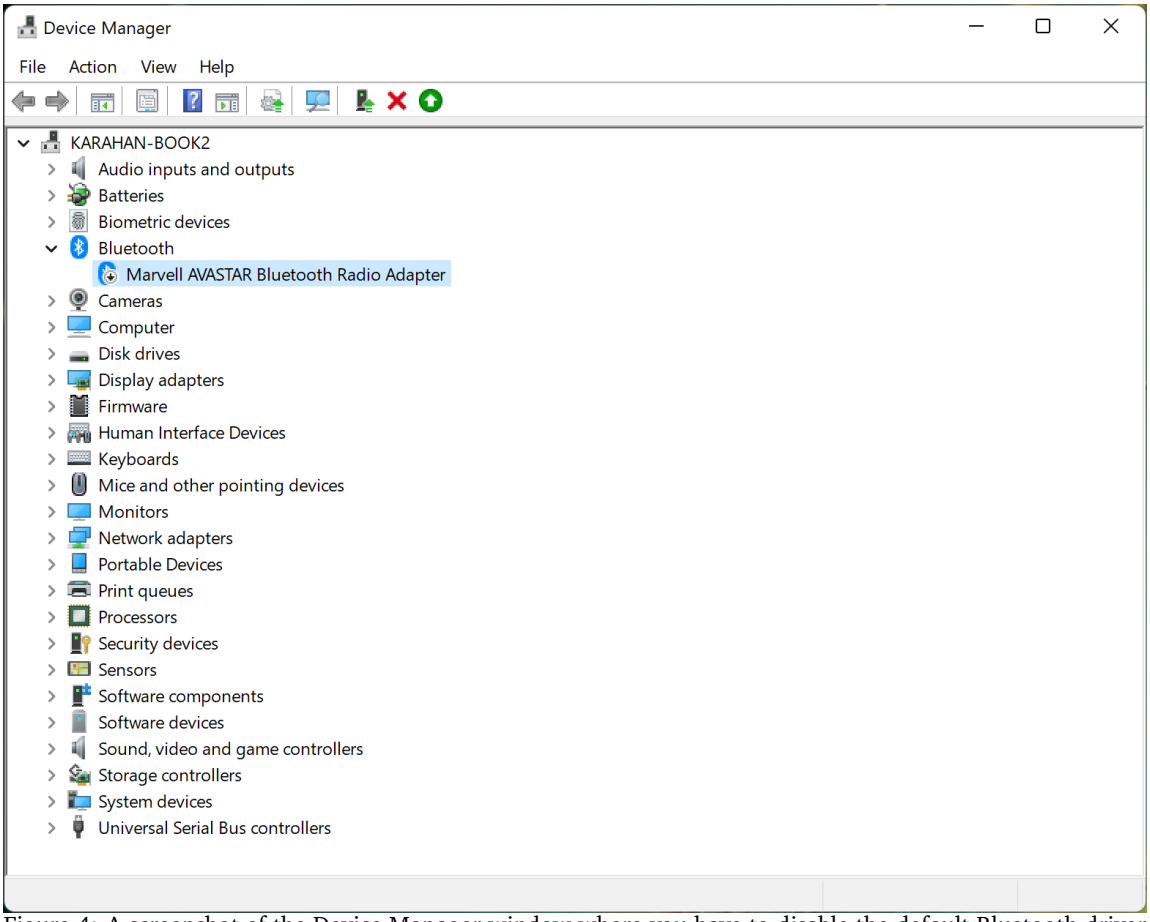

Figure 4: A screenshot of the Device Manager window where you have to disable the default Bluetooth driver to be able to use the Unicorn's Bluetooth adapter.

that this Makefile has been tested on a standard Ubuntu distribution but may give you problems depending on your C++ compiler & linker. To compile the the executable simply run:

```
karahan@yilma:~/Python/bci-course-tutorials/unicorn-software/unicorn-c-linux$ make
```

If no errors are encountered you should now see an executable **`scan`** in the `unicorn-c-linux` folder. Before you run the program, make sure that your Bluetooth is enabled, the Unicorn headset is switched on, the headset is in range and not connected to another device (a light should be flashing on the back). At this point, you can run the program with:

Connect to and Stream Data from Unicorn

```
karahan@yilma:~/Python/bci-course-tutorials/unicorn-software/unicorn-c-linux$ ./scan
```

If everything works correctly, this program should find your Unicorn, connect to it, and start streaming data through LSL.

6 Smarting 24

As the name suggests, Smarting 24 (S24) is another EEG channel with 24 channels. Whereas UHB had the so called "hybrid" channels which can work both dry and wet, S24 channels can only be used wet. This means you will have to apply conducting gel to all the electrodes you want to use (and clean them afterwards ;)).

S24 has a sampling rate of 250 Hz and 500 Hz, left for the user to choose during recording. Some features of the Smarting software are not available for the 500 Hz option and 250 Hz will be enough for our current setting.

Just like UHB, S24 also has a Bluetooth interface. However there is a catch. You will have to install the BlueSoleil software to connect to the device. This software might mess up the connection between your laptop and your Bluetooth devices (like your mouse or headphones) until you uninstall the software again. That is why we will be providing 1-2 computers with pre-installed BlueSoleil and Smarting software for you to utilize if you do not want to install these on your laptop.

Figures 5 and 6 show the connection interface to S24. Note down the COM port number for the Smarting Streamer app.

After connecting the device, open Smarting Streamer 3.0. The installation files are provided in the GitLab repository. Click on "CONNECT" on the top left corner and select the COM port that you noted down earlier. This window can be seen in Figure 7.

As mentioned before, first select the 250 Hz mode. Then click on "Measure impedance on ref" (Figure 8). This will give you a impedance measurement of the reference channel. You should reduce the impedance of this channel as much as you can (it should light up green) because all the other channels will be based off of this channel's signal. Failing to reduce this channel's impedance will severely impact your recording quality.

Now stop streaming and switch to "Measure impedance" (Figure 9). Apply gel to all the channels you want to record from and get their impedances as low as possible (all of the channels should be green).

You should now be able to record. First stop streaming signals, then select "No impedance measurement" and start streaming signals again. When you open LabRecorder (covered in section 7) you should be able to see your EEG stream. In the meantime, you can click on "show signals" to check the real-time signal.

Further instructions can be found in the Smarting 24 manual in the repository.

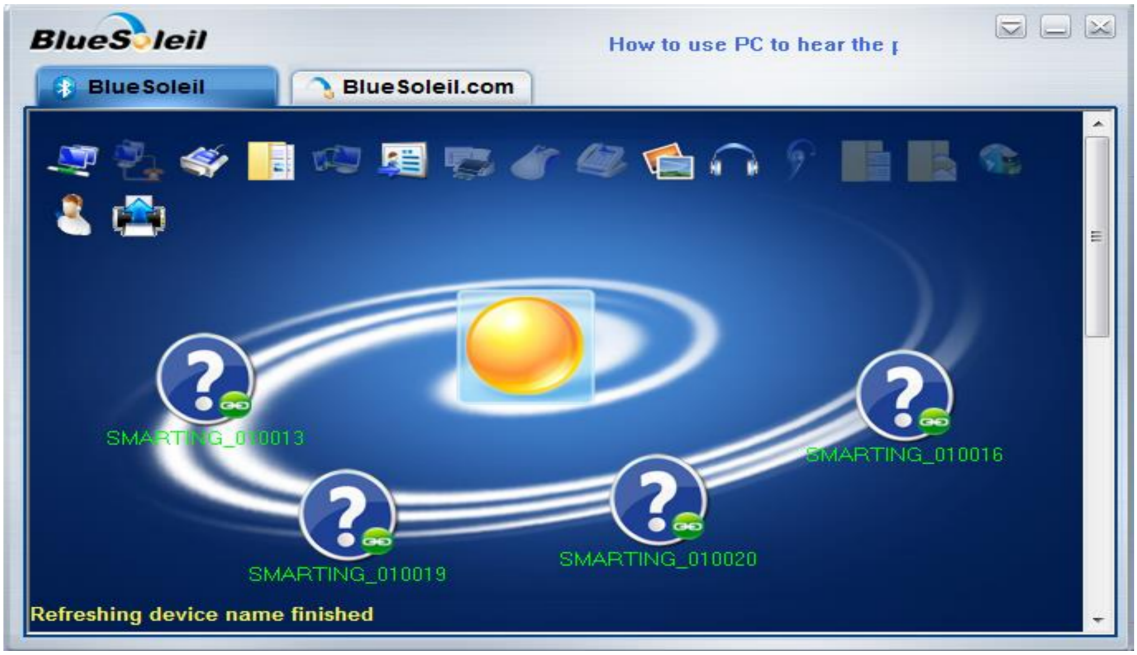

Figure 5: The interface of BlueSoleil for connecting to the Smarting 24 device. In this example there are four devices available but in our case there will be only one.

7 LabRecorder

At this point, you might have a feeling that there are a lot of LSL streams flying around and you might be disoriented. Fortunately, LabRecorder comes to the rescue. This app can time-synchronize and save different LSL streams available in your network into a single XDF file. For the installation, please refer to the installation guide in their GitHub repository. You can find a screenshot of this app in Figure 10

Info:
For Windows, navigate to the Releases page and install the latest version with the suffix "amd64". This will install a zip folder which you can extract anywhere. In the extracted folder you will find the executable to the LabRecorder app called "LabRecorder.exe". For convenience, create a shortcut to this executable and place it somewhere easy to navigate to.

When you open the program, you will find a list of all available LSL streams in your network. Choose the ones coming from your EEG and the LSL marker stream (which you will set up in one of the upcoming tutorials). Then on the right panel, specify the metadata for your recording and choose a location for the recording data to be saved in. By clicking on Start on the top left corner, you will start the recording.

Congratulations! With this you have finished the boring setup part of your project. Now you can focus on the more entertaining science part ;)

Figure 6: The interface of BlueSoleil for connecting to the Smarting 24 device. "Connect Bluetooth Serial Port (COM24)" is to be selected. Note down the COM port number for upcoming steps. If the connection is successful, the question mark should turn green.

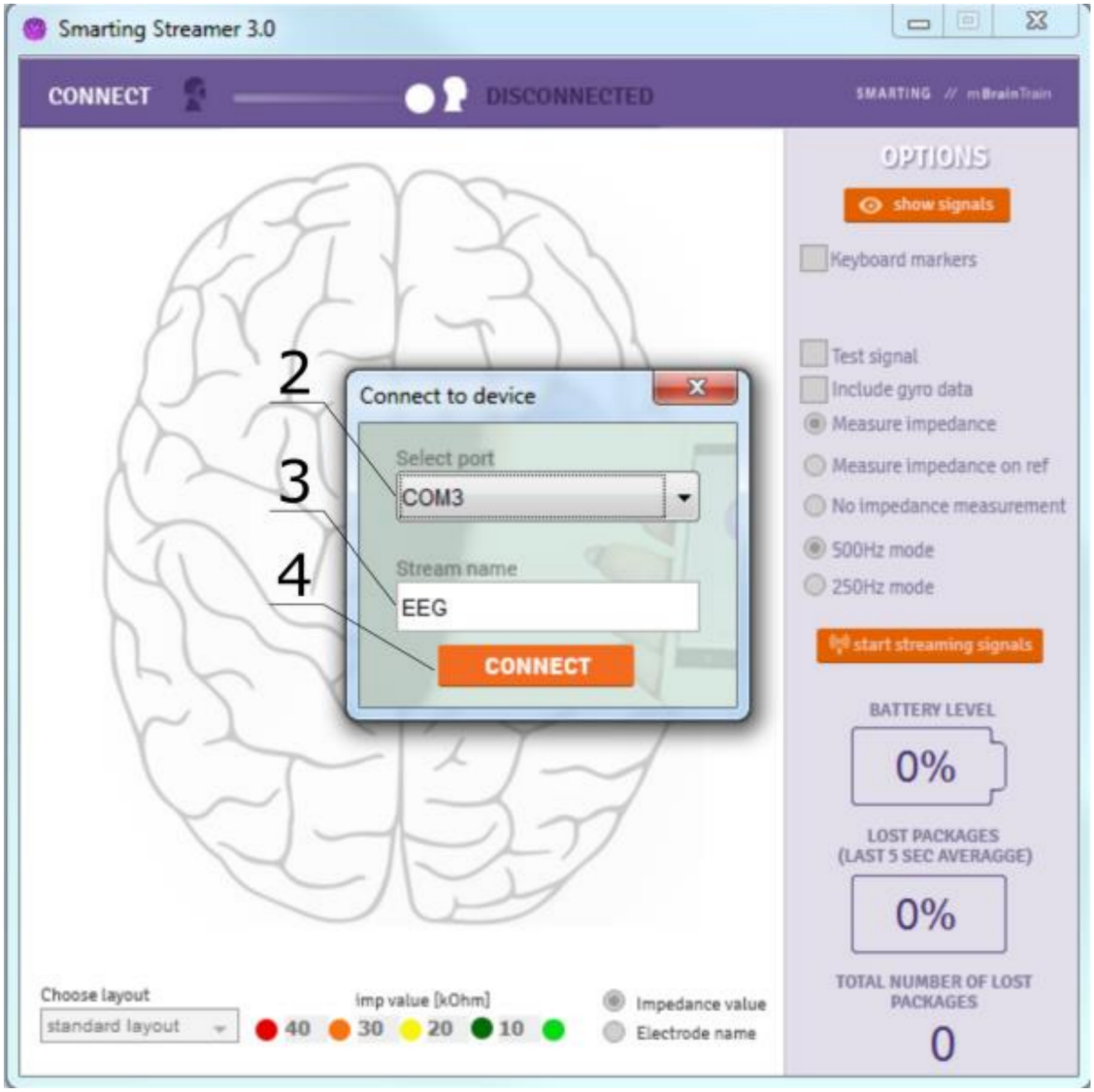

Figure 7: The graphic interface to connect to S24. Choose the COM port that you noted down earlier and click on "CONNECT".

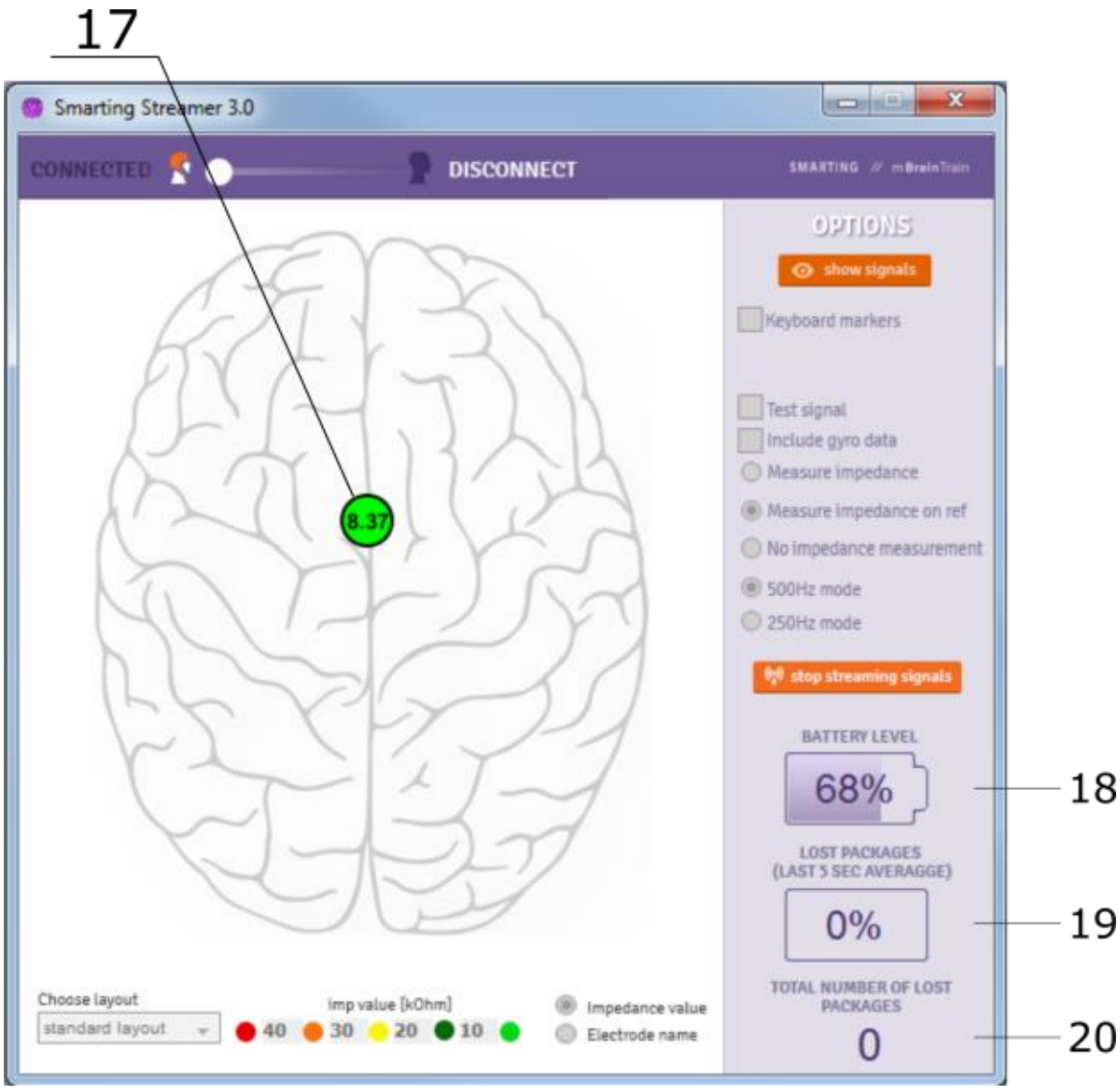

Figure 8: The graphic interface to see the impedance on the reference electrode. It is very important to get this as low as possible.

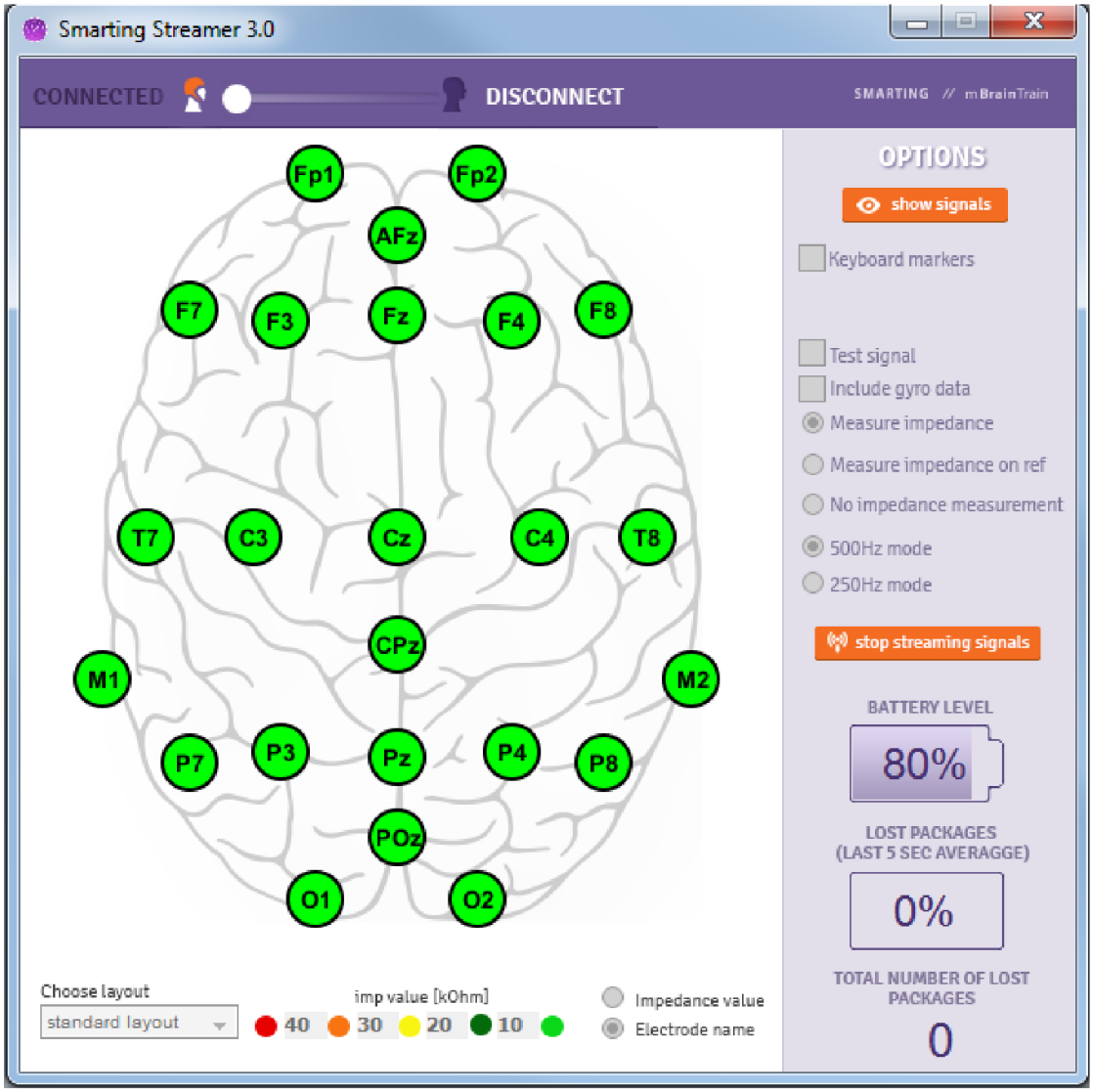

Figure 9: The graphic interface to see the impedance on each electrode. Apply gel to the electrodes you want to record from.

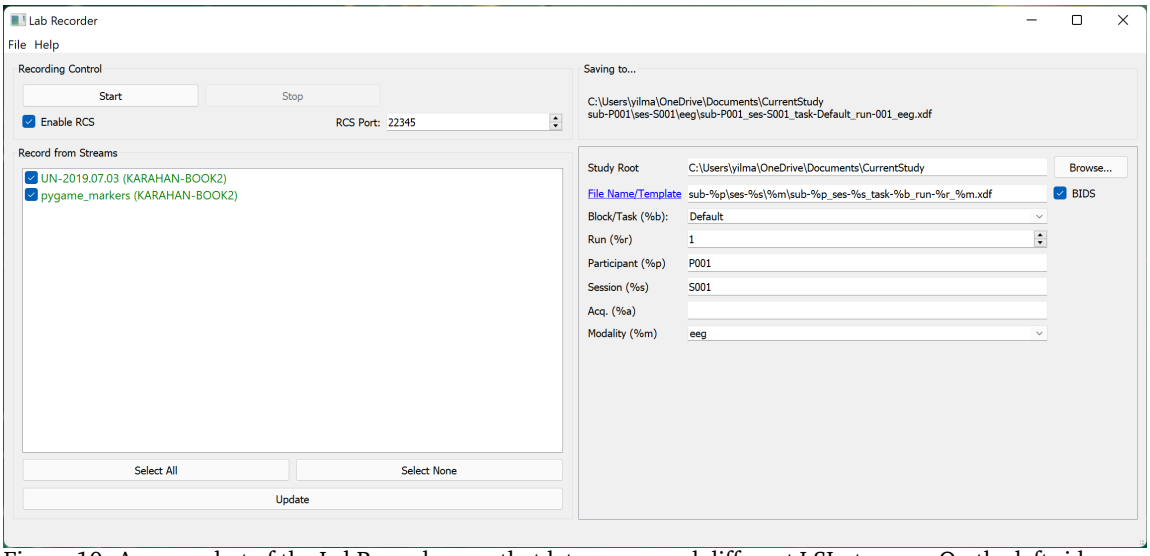

Figure 10: A screenshot of the LabRecorder app that lets you record different LSL streams. On the left side, you have to select the streams that you want to record. On the right side you have to type in information about the current recording, which is very important, otherwise all the recordings get very messy!