The data for both the MoCap sensors and the fNIRS are obtained using a simple graphical user interface (GUI) that allow for a start-stop feature when collecting data. Although both the MoCap and fNIRS systems already have their own native softwares, MOCAP Studio and NIRStar, respectively, building a custom GUI was necessary for more control over the synchronization of data collection for both systems. The GUI will not be the focus here, as it was only made for practicality, with no emphasis on esthetics whatsoever. Rather, this section will explain how the APIs (application programming interfaces) are used to make the combined utilizations of the respective systems fast and easy to support.

We will begin with the MoCap sensors and their corresponding API, named threespace (all the files needed to get started can be found on GitHub under https://github.com/Knio/threespace). The API is an open-source collection of classes, functions, structure definitions, and static variables designed to facilitate writing applications to interface with YEI 3-space motion-capture devices. It is made available as collection of Python source files and a Windows binary application.

The main file named threespace\_api.py, contains all the classes that are instanced by the user when connecting to 3-Space Sensor devices. This file can easily be modified at the user’s convenience (note: it is important to always remember to generate the compiled python file –.pyc file –for the changes to be effective. To do so, run the file named \_\_init\_\_.py). For our purposes, only minor changes were made, such as adding timestamps to the data collected.

The other files provided, namely threespace\_utils.py, win32\_threespace\_utils.py, and others inside the try\_port folder are used by the API to interface with the ports and the devices. For more details, visit https://github.com/Knio/threespace/blob/master/threespace/README.txt, for the README.txt file.

This API thus provides a wide set of commands that can be called to interact with 3-space devices. Calling these commands with the GUI, we can connect to the Bluetooth dongles through USB, that in turn automatically connect to the wireless motion-capture sensors to collect the desired data. That is, in our case, the raw accelerometer, gyroscope, and magnetometer readings, each with 3 degrees of freedom (X, Y, and Z), as well as the state of the built-in button. Given the significant number of sensors (up to 17), collecting the data sequentially wouldn’t be the best option; for the following sample of every sensor would only be collected after all other sensors’respective samples are collected. This is overcome by the API through the option of batch data collection. This batch data collection records data simultaneously from all sensors involved. The data can then easily be written to a text file with native python functions.

Moving on to the fNIRS wrapper, this one comes as a software add-on to the NIRStar15 that allows streaming data through an API called Lab Streaming Layer (LSL). This API is available in several different languages, including Python. It is also available on GitHub under https://github.com/sccn/labstreaminglayer. This API detects the available fNIRS channels, before collecting the readings from all of them. So, most of the setup is done from the native software (NIRStar), and the API simply collects the data and writes it to a text file.