**Visualisation** – ulazna točka: stream tab, na njega se spajaju data sources

Base Stream Widget – svaki strean widget je zaduzen za vizualizaciju corresponding stream type-a

four types of visualization formats in PhysioLabXR

1. Line Chart - cesto koristen za neurosignale, eyetracking i ostale psihološke signale za koje je potrebno promatrati promjene kroz period, parametri: nominal sampling rate (scale x-axis), Seconds of Data to Display (number of seconds of data to display on the plot, scale x-axis), ako ima previše channela jedni prek drugih su i onda se može staviti offset konstanta
2. Image – za neki dana koji je obtained iz medical devices
3. Bar Chart – plots one frame of dana at a time, super je za vizualizaciju predictiona napravljenih iz modela strojnog učenja, svaki probability je prikazan zasebnnim barom. Parametri: plot range min, plot range max
4. Spectrogram – vizualizacija spectralnog contenta signala, commonly employed in the analysis of EGG and audio dana, dobar za frenkvencijske komponente i njihove promjene kroz vrijeme

Channels sa istog streama se mogu organizirati u grupe i biti plottane u zasebne plots. To je useful kad channeli mjere različite phenomena sa različitom vertikalnom skalom.

PhysioLabXR is designed to cater to experiments that involve multiple data sources.

**Recording Data –** recording interface da da se snimi multi-stream dana sa sinkroniziranim vremenskim isječcima iz različitih izvora.

Supported formati: .dats, .p (pickle): Python’s data format( ovaj format je optimiziran za quick serialization and deserialization, recommended za downstream analysis), .m (MATLAB), .cvs, .xdv

Physiolab sprema podatke za svaki frame i vrijeme kad je uzeo taj frame. Kad se podaci učitaju npr u Phythonu returnaju se ko a dictionary of arrays and timestamps. Keys of the dictionary are the stream names. tuple of (data, timestamps).

Tutorial za recordanje data <https://physiolabxrdocs.readthedocs.io/en/latest/Recording.html> – dummy stream (kod), za importanje snimljenog isječka koda: pip install PhysioLabXR + from physiolabxr.utils.user\_utils import stream\_in + data = stream\_in('path/to/dats/file')

**Replaying –** za replayanje data iz prijašnjih eksperimenta, ako je više streamova snimljeno u jednoj snimci sinkronizirat će im se timestamps. Tutorijal za replayanje: <https://physiolabxrdocs.readthedocs.io/en/latest/Replay.html>

Kad se neki replay ubaci sadrži: Stream name, Channels + Time points (The number of channels and time points in the stream), Averaged sampling rate, data type, stream interface (npr. LSL)

**Scripting –** mogu se runnat Python scripts za stvaranje i deployanje custom data processing pipelines, npr. može se napraviti skripta za: streamanje podataka sa custom hardware devicea

Components: Script Name( path of your script file on the system ), Run Frequency( max frequency na kojoj ce se skripta runnati ), Input Buffer Duration ( the maximum data you can access to in during a loop of your script), Inputs Pane (gdje se mogu streams dodati skripti kao inputi, dodani su pomocu self.inputs[my\_stream\_name – data matrix((f nominal ∗ T bufferduration, N channels) i timestamp vector ), output pane – za kreiranje output streamova – ime i broj channela ( self.outputs[my\_stream\_name] ), parameters pane – inicijalizirat varijable koje user želi promjeniti tijekom runtimea( name, type, values - self.params[my\_parameter\_name]), console log, run/stop/kill, simulate input check box ( kad želiš randomizirane podatke za input streams ), loop per second ( koliko puta se skripta loopala u sekundi), average loop call runtime

Tutorial za laganu skriptu, cini se dost korisno <https://physiolabxrdocs.readthedocs.io/en/latest/Scripting.html>

# Start will be called once when the run button is hit. ( defining variables, loading pre-trained parameters for an ML model from the file system, and connecting with servers and cloud platforms)

def init(self): pass

# loop is called <Run Frequency> times per second - continuously process input data and communicate with other processes via the outputs

def loop(self):

# cleanup is called when the stop button is hit -  may free occupied memory, disconnect hardware ports and close any opened file system resources

def cleanup(self):

**Real-time Digital Signal Processing –** Physiolab ima selekciju build-in digital signal processing modules. Npr. low-pass filters, high-pass filters, band-pass filters, root mean square calculations, and clutter removal techniques

Filter Types:

1. NotchFilter: to remove a specific frequency from the signal.
2. ButterworthLowpassFilter: to remove high frequency components from the signal
3. ButterworthHighpassFilter: to remove low frequency components from the signal
4. ButterworthBandpassFilter: to remove low and high frequency components from the signal
5. RootMeanSquare: to calculate the root mean square of the signal over time
6. ClutterRemoval: statistic method to remove the clutter from the signal

drop-down menu allows you to select the DSP module you want to add to the selected stream group

Mogu se koristiti i DSP modeli u kodu: from physiolabxr.utils.dsp\_utils.dsp\_modules import ButterworthLowpassFilter