Read\_Me TestScripts Nathan

In this document, the pipeline for MVC and training measurements will be explained for MU decomposition.

**Example\_MU\_Training.py:**

Enables all the right channels and configurations for the measurement. The force channel and grid type must be changed according to the measurement setup.

After, it initializes and runs a Setup\_Experiment instance in which the right kind of measurement will be set up.

**Setup\_Experiment.py:**

Asks what kind of measurement you want to do: an calibration, MVC or training

Asks what kind of file type you want to use for the measurement.

Initializes the forcefeedbackplotterhelper, GUI\_experiment and right file writer instances.

Launches the app.

**Gui\_Experiment.py:**

Creates the windows and sets the force channel and force profile to be displayed.

Also initiates the execution of a measurement when the main window is closed.

**ForceFeedbackPlotterHelper.py: (child class of the FilteredSignalPlotterHelper)**

Initialize():

Initializes channel settings, where the force\_profile channel is added.

Also extracts the names, unit names, and channel types of the channels.

Initializes an extra file writer to write away the data and uses CHANGED file writers.

initialize\_file\_writer():

Initializes a ForceFileWriter, which is an instance of the right file writer.

The poly5 and xdf file writers are changed to be able to handle the added force profile.

Start(): Waits for user input in the app before starting a calibration, MVC or training.

Includes the preview button to preview the force profile.

Start\_MVC(): initializes the force profile for MVC.

Initializes the Consumer, ConsumerThread, and Monitor instances and starts the measurement and timer.

Start\_training():

Initializes the force profile for a training measurement based on user input.

Runs the MVC\_Extraction.py which opens the file director and

extracts the offset and mvc value from a self-chosen file.

Initializes the Consumer, ConsumerThread, and Monitor instances and starts the measurement.

determine\_offset(): (in Construction)

Function that runs a measurement in the background without showing the signals.

It tells the user what to do. It will create a calibration measurement that can be

loaded when doing a MVC measurement to get the ranges right.

Callback():

Gets the data from the Monitor instance and plots it in 2 different windows.

Main window contains the force feedback plotter and secondary window has the normal signal plotter

In MVC:

Load cell data gets plotted normalized, however gets saved unnormalized.

In Training:

Load cell data is plotted and saved normalized.

put\_sample\_data():

Reshapes the new send data from the device in the right shape and puts it in the SampleData instance.

Puts it in the Queue of the ForceFileWriter

Some extra added functions that are self-explanatory.

**MVC\_Extraction.py:**

Loads a .poly5 or .xdf file and calculates the offset and maximum value of the MVC measurement

(Must still be changed to get a more objective measure for the mvc (feedback Utku)).

ForceFeedbackPlotter.py: (copy of SignalPlotter)

Changes made in initialize\_channels\_components(), update\_chart()

initialize\_channels\_components():

Any channel that is no real channel of the device (such as the force profile)

will get unit = 'a.u.' and ChannelType.Unknown. This is necessary for the XDF file writer.

update\_chart():

#if self.\_compute\_autoscale: deleted, so the y range gets updated each iteration.

The standard scale value is 10 for all channels.

The offset is set to the middle value of the force profile.

In this way, it follows the force profile with a range of +/- 1.5\*self.scale.

**ForceFeedbackChart.py:**

Changes made in setup\_signals() and update\_y\_ticks().

Changes are discussed in the code

**\_experiment\_plotter.py:**

Includes the **Ui\_Experiment\_Plotter** for the GUI. The MVC and Training GroupBoxes are added in setupUi(): with all their buttons for the settings of the measurements.

**experiment\_plotter.py:**

child class **of Ui\_Experiment\_Plotter**

contains a configuration() function that enables the right layout for different types of measurements

**poly5\_force\_file\_writer.py:**

Receives channel names and units from the ForceFeedbackPlotterHelper as arguments in open().

The rest is not changed

**xdf\_force\_file\_writer.py:**

Receives channel names, units and types from the ForceFeedbackPlotterHelper as arguments in open().

\_write\_stream\_header\_chunk():

All information that was usually extracted from device.channel instances are swapped for

their equivalent element in self.channel\_names, self.channel\_units, self.channel\_types

**At the end:**

The pipeline creates 1 file for each recording

For MVC and training:

- one file with the original data and the force profile appended to it (name: {type of measurement}\_Measurement+ForceProfile\_{date}\_{time}.{file type})

For calibration:

- one file with original data. (name: Calibration\_measurement\_{date}\_{time}.{file type})