Bsense GUI User Documentation

Overview

Bsense GUI, is a Python-based graphical user interface for conducting experiments with the Bsense device. Version 1.0 of this software allows users to connect to devices, select and customize experiments, and manage experiment logs and settings. The GUI is designed to be user-friendly and intuitive, allowing for easy setup and execution of experiments.

Features

Connection Settings

- **Device Connection:** Allows users to specify the device connection port (e.g., COM3) and establish a connection to the device. (Note: The device must be connected to the computer via USB. On Windows, you can find the device port in the Device Manager under "Ports (COM & LPT)".)
- **User Validation:** Users have to enter a subject name/ID and validate it before starting an experiment. The log file will be saved with the subject name as the filename.

Experiment Configuration

- Predefined experiments: Includes a set of predefined experiments which can be selected via a dropdown menu.
- Custom experiments: Users can load custom experiments by selecting a JSON file that defines the experiment parameters. You have to select the file using the "Browse" button. (see Writing Your Own Experiment Configuration File for more details on the JSON file format)

Experiment Controls

- **Start (GO):** Begins the execution of the selected experiment.
- **Pause:** Temporarily halts the experiment.
- **Stop:** Terminates the experiment.

Log Management

- **Experiment Log:** Displays a real-time log of the experiment events and actions. Users can also add custom log entries via the "Add Log" button.

Closing the Application

- Ensures proper shutdown by closing the connection to the device and saving the log file if it was opened.

Usage

1. **Start the Application:** Launch the GUI application.
2. **Connect to Device:** Enter the device port in the connection section and click "Connect".
3. **Validate User:** Enter a subject name and click the validate button (checkmark icon).

4. **Select Experiment:** Choose an experiment from the dropdown menu or load a custom experiment using the "Browse" button.
5. **Start Experiment:** Click the "GO" button to start the experiment. Use "PAUSE" and "STOP" buttons to control the experiment flow.
6. **Add Log Entries:** Optionally, add custom log entries during the experiment.
7. **Close Application:** Close the application window or use the "STOP" button to ensure a proper shutdown.

Writing Your Own Experiment Configuration File

Overview

The experiment configuration file is a JSON document that defines the structure and behavior of an experiment. It allows for the specification of sequences, stimuli, delays, and dropout sequences to create complex, repeatable experimental conditions.

Types

Sequence

A sequence is a collection of stimuli and delays that are executed in order. It can contain other sequences, allowing for nested structures.

- **Type**: Must be `"Sequence"`.
- **Repeat**: How many times the sequence repeats.
- **Content**: An array of `stimulus` and `Delay` objects.

Stimulus

Defines a stimulus event, such as a vibration or a buzzer sound.

- **Type**: Must be `"stimulus"`.
- **Content**: An array of objects detailing the stimulus types (`Vib1`, `Vib2`, `Buzzer`) and their properties.

Properties

- **Amplitude**: The intensity of the stimulus.
- **Deviation**: Allows for random variation in the amplitude.
- **Type**: The type of stimulus (`Vib1`, `Vib2`, `Buzzer`).
- **Tone** (optional for `Buzzer`): The tone frequency.
- **Duration** (optional for `Buzzer`): How long the buzzer sounds.

Delay

Specifies a pause or delay between stimuli within a sequence.

- **Type**: Must be `"Delay"`.
- **Duration**: The length of the delay.
- **Deviation**: Allows for random variation in the duration.

Dropout_sequence

Similar to a sequence but with the ability to randomly omit certain repetitions, replacing them with alternative content.

- **Type**: Must be `"Dropout_sequence"`.
- **Repeat**: The total number of times the sequence should be attempted.
- **Number_drop**: How many of those repetitions to replace.
- **Content**: The normal sequence content.
- **Dropout_content**: The content to use when a repetition is dropped.

Example Configuration

```
{
  "Type": "Sequence",
  "Repeat": 1,
  "Content": [
    {
      "Type": "Sequence",
      "Repeat": 15,
      "Content": [
        {
          "Type": "stimulus",
          "Content": [
            {
              "Type": "Vib2",
              "Amplitude": 1,
              "Deviation": 0
            },
            {
              "Type": "Buzzer",
              "Amplitude": 1,
              "Deviation": 0
            }
          ]
        },
        {
          "Type": "Delay",
          "Duration": 5,
          "Deviation": 0
        }
      ]
    },
    {
      "Type": "Dropout_sequence",
      "Repeat": 5,
      "Number_drop": 1,
      "Content": [
        {
          "Type": "stimulus",
          "Content": [
            {
```

```
            "Type": "Vib2",
            "Amplitude": 1,
            "Deviation": 0
          }
        ]
      },
      {
        "Type": "Delay",
        "Duration": 5,
        "Deviation": 0
      }
    ],
    "Dropout_content": [
      {
        "Type": "Delay",
        "Duration": 5,
        "Deviation": 0
      }
    ]
  }
 ]
}
```

Tips for Writing Configuration Files

- Repeat values allow for repeating sequences or stimuli, creating complex experimental patterns without duplicating definitions.
- Utilize Deviation to introduce variability into your experiment, making it less predictable and more realistic.
- Dropout_sequence allows for the creation of dynamic experiments where certain sequences can be randomly omitted, introducing unpredictability and variability.