COVER PAGE

Table of Contents

Introduction

The software is designed to facilitate experiment logging and data recording for research purposes. It offers a user-friendly interface that allows researchers to log timestamps of various events during an experiment, while also capturing data from basic sensors such as the microphone, mouse, and keyboard. The software ensures accurate timestamp logging by converting the GMT timestamp to the Finnish time zone.

To begin the logging process, the user provides essential information including the experiment name, participant ID, and the folder path where the logged timestamps will be saved. Once the necessary details are entered, the software provides buttons to log events such as "experiment start" and "experiment end," "task start" and "task end," interval start and end, as well as individual event starts.

Each time a button is pressed, the software records the GMT timestamp and converts it to the Finnish time zone before logging it. This ensures consistency in the timestamp recordings regardless of the researcher's location.

When the experiment is completed, the software allows the researcher to save all the logged information as a tab-delimited text file. The file is saved in the selected folder and is named according to the participant's name, making it easy to identify and organize the data.

Simultaneously, the software utilizes the Lab Streaming Layer (LSL) to capture data from the basic sensors available on the researcher's computer, namely the microphone, mouse, and keyboard. Once the experiment starts, the LSL initiates the recording of streams from these sensors. When the experiment ends, the LSL saves the recorded streams to the same folder where the timestamp logs are stored, ensuring the data remains organized and easily accessible.

With this software, researchers can streamline the process of experiment logging, ensuring accurate and consistent timestamp records while simultaneously capturing data from essential sensors. By providing an intuitive interface and automated data recording, the software simplifies the research workflow, enabling researchers to focus on their study objectives and data analysis.

Key Features

Experiment Logging: The Experiment Logger provides a convenient interface for logging experiment-related information, such as experiment name, participant name, and folder path. This ensures proper organization and categorization of data.

Timestamp Logging: The tool offers the ability to log timestamps for various events during the experiment, such as experiment start and end, task start and end, interval start and end, and custom events. This allows for accurate tracking and synchronization of events with the recorded data.

LSL Stream Recording: The Experiment Logger supports recording data streams using the Lab Streaming Layer (LSL) protocol. It enables the user to select and record audio, keyboard, and mouse data streams, capturing essential information during the experiment.

User-Friendly GUI: The graphical user interface (GUI) of the Experiment Logger is designed to be intuitive and user-friendly. It provides clear labels, input fields, and buttons for easy interaction and navigation.

Data Preview: The tool offers a real-time data preview feature, allowing users to view and monitor the recorded data as it is being logged. This helps researchers ensure the quality and accuracy of the recorded data during the experiment.

Audio Recording: The Experiment Logger enables audio recording, capturing audio signals during the experiment. It utilizes the PyAudio and Wave libraries to record and process audio data.

Keyboard and Mouse Monitoring: The tool provides the capability to monitor keyboard and mouse activities during the experiment using the pynput library. It records key presses, mouse clicks, and movements, capturing user interactions.

Experiment Control: The Experiment Logger includes buttons for controlling the experiment flow, such as starting and ending the experiment, starting and ending tasks or intervals, and triggering custom events. These controls provide flexibility and enable researchers to precisely manage the experiment timeline.

Data Saving: The tool allows users to save the logged experiment data in a specified folder. It ensures that the recorded data is securely stored for further analysis and future reference.

Dependency Management: The Experiment Logger handles dependency management by providing a list of required libraries and their versions. This simplifies the setup process and ensures that all necessary dependencies are installed.

Development Procedure

1. **Importing Required Libraries:**

- The code begins by importing the necessary libraries for GUI (Graphical User Interface) interface, mouse and keyboard control, audio streaming, and data recording.

2. **Creating GUI Interface:**

- The code creates a GUI interface using the Tkinter library.
- It creates a window and sets its title and background color.
- The window is divided into two frames: left_frame and right_frame.
- Labels, entry fields, and buttons are placed in the left_frame to capture user inputs and control the experiment logging process.
- The right_frame contains a Text widget to display the logged information.

3. **Timestamp Logging:**

- The code defines functions to handle different events in the experiment logging process, such as experiment start/end, task start/end, interval start/end, and event start.
- Each function updates the timestamp and logs the event in the Text widget.
- The functions also update counters for different events to keep track of their occurrences.

4. **Folder Creation:**

- The "Create Folder" button allows the user to select a folder where the experiment data will be stored.

- The function associated with the button creates a folder with a specified name (experiment name + participant name) in the selected directory.
- It also updates the GUI to display the selected folder path.

5. **Experiment Control:**

- Buttons for experiment start/end, task start/end, and interval start/end are provided to control the experiment process.
- These buttons call the corresponding functions to update the timestamp and log the events.
- The buttons are disabled/enabled based on the experiment flow to ensure proper sequencing of events.

6. **Saving Logged Information**:

- The "Save" button allows the user to save the logged information to a text file.
- The function associated with the button writes the experiment details and timestamp information to the file.
- It also clears the input fields and enables them for new inputs.

7. **Data Recording**:

- The code includes functions for recording audio, mouse, and keyboard data.
- The functions use various libraries such as PyAudio, wave, pynput, and pylsl for audio recording and mouse/keyboard event capture.
- The recorded data is saved in files with appropriate timestamps.
- Threads are used for concurrent recording of audio and mouse/keyboard data.

8. **GUI Configuration and Execution**:

- The code sets the initial states of buttons and input fields.
- It configures the GUI layout and starts the main event loop for the window.

Project Usage

4.1 **Running the Application**

- To start the Experiment Logger, execute the Python script on a compatible system.
- The Experiment Logger window will appear, providing an intuitive interface for experiment management and data recording.

4.2. **Experiment Information**

- The Experiment Logger requires the user to provide essential information before proceeding with data logging.
- Enter the experiment name, participant name, and folder path in the respective fields.
- Experiment Name: Reflects the purpose or name of the experiment.
- Participant Name: Identifies the participant for better organization and identification of data.
- Folder Path: Specifies the directory where the experiment data will be stored.

4.3. **Folder Creation**

- To ensure proper organization of the recorded data, the Experiment Logger allows users to create a dedicated folder for each experiment.
- Click the "Create Folder" button to create a folder in the specified directory.
- The folder will be named based on the experiment name.
- All recorded data will be saved within this dedicated folder.

4.4. **Timestamp Logging**

- The Experiment Logger enables users to log timestamps for different phases of the experiment, such as experiment start and end, tasks, intervals, and events.
- Use the provided buttons to mark the beginning and end of each phase:
- Experiment Start: Marks the start of the experiment.

- Experiment End: Marks the end of the experiment.
- Task Start and Task End: Define specific tasks within the experiment.
- Interval Start and Interval End: Indicate time intervals within a task.
- Event: Allows logging of any other relevant events during the experiment.
- The Experiment Logger records and logs the timestamps for each action, enabling easy tracking and analysis of experiment progression.

4.5. **Data Recording**

- The Experiment Logger supports the recording of audio, keyboard, and mouse data during the experiment.
- Check the corresponding checkboxes to select the desired data types to record:
- Audio: Records audio input from the connected microphone.
- Keyboard: Captures keyboard events, including key presses and releases.
- Mouse: Captures mouse events, such as clicks and movements.
- Click the "Start Record" button to begin recording the selected data types.
- The recorded data will be saved as separate files within the experiment folder, ensuring data integrity and ease of access.

4.6. **Stopping the Recording**

- To stop the recording process, click the "Stop Record" button.
- The Experiment Logger will cease recording the selected data types and save the recorded files in the experiment folder.

4.7. **Saving the Logged Information**

- After the completion of an experiment or a significant phase, users can save the logged information for future reference and analysis.
- Click the "Save" button to save the logged information, including timestamps and recorded data.

- The Experiment Logger saves the information as a text file within the experiment folder.
- This file provides a comprehensive log of all the recorded events and associated timestamps.

Dependencies

List all the libraries and packages that your project depends on. You can mention them as follows:

1) cffi==1.15.1
2) keyboard==0.13.5
3) mouse==0.7.1
4) numpy==1.24.3
5) pandas==2.0.2
6) PyAudio==0.2.13
7) pycparser==2.21
8) pylsl==1.16.1
9) pynput==1.7.6
10) PyQt5==5.15.9
11) PyQt5-Qt5==5.15.2
12) PyQt5-sip==12.12.1
13) python-dateutil==2.8.2
14) pytz==2023.3
15) pywin32==306
16) pyxdf==1.16.4
17) six==1.16.0
18) sounddevice==0.4.6
19) tzdata==2023.3
20) Wave==0.0.2
21) windows-filedialogs==0.0.6

A single-line explanation for each dependency

- tkinter: Used to create the GUI interface for the Experiment Logger.
- pynput: Enables monitoring and control of input devices such as the keyboard and mouse.
- pyaudio: Facilitates audio recording and processing.
- wave: Provides functionality for working with WAV audio files.
- cffi: Required by some dependencies for CFFI-based bindings.
- keyboard: Allows capturing and controlling keyboard events.
- mouse: Allows capturing and controlling mouse events.
- numpy: Provides support for numerical operations and data manipulation.
- pandas: Offers data analysis and manipulation tools.
- PyAudio: Provides audio input/output functionality.
- pycparser: Required by some dependencies for parsing C code.
- pylsl: Integrates with Lab Streaming Layer (LSL) for stream recording.
- pynput: Enables monitoring and control of input devices such as the keyboard and mouse.
- PyQt5: Used for creating the GUI interface and interacting with Qt framework.
- PyQt5-Qt5: Provides Qt5 library integration for PyQt5.
- PyQt5-sip: A binding generator used by PyQt5.
- python-dateutil: Offers powerful date and time parsing and manipulation capabilities.
- pytz: Provides timezone support and conversion functions.
- pywin32: Required for accessing Win32 API functions and resources.
- pyxdf: Allows reading and writing of data in the XDF format.
- six: A compatibility library that provides Python 2 and 3 compatibility utilities.
- sounddevice: Provides high-level audio I/O in Python.
- tzdata: Offers timezone database and conversion functionality.
- Wave: A small library for working with WAV audio files.
- windows-filedialogs: Provides Windows-specific file dialog functionality.

System Requirements

- Operating System: Windows
- Python Version: 3.9
- Processor: Any modern processor
- RAM: Minimum 4GB
- Disk Space: Minimum 100MB

Limitations

Platform Compatibility: The Experiment Logger code has been developed and tested on a specific platform (e.g., Windows). It may not be fully compatible with all operating systems, and some functionalities or features may behave differently or not work as expected on certain platforms.

External Dependencies: The Experiment Logger relies on external libraries such as tkinter, pynput, pyaudio, wave, and pytz. These dependencies need to be installed separately to ensure the proper functioning of the code. Any issues or limitations associated with these libraries may also affect the Experiment Logger.

User Interface Limitations: The graphical user interface (GUI) of the Experiment Logger has a specific design and layout. While efforts have been made to provide a user-friendly interface, it may not fully accommodate all user preferences or accessibility requirements. The GUI may also not scale well on all screen sizes or resolutions.

Hardware Compatibility: The Experiment Logger code assumes the availability and compatibility of input devices such as a keyboard, mouse, and audio input on the system where it is run. Any limitations or issues related to the hardware setup or configuration may impact the recording and logging functionality.

Limited Error Handling: Although efforts have been made to handle errors and exceptions gracefully, the Experiment Logger may encounter unexpected errors or exceptions in certain scenarios. It is important to be aware of potential error conditions and implement appropriate error handling mechanisms for specific use cases.

Performance and Scalability: The Experiment Logger has been designed to handle recording and logging tasks for a single user. It may not be optimized for large-scale experiments or scenarios involving multiple concurrent users. Performance issues may arise when recording extensive audio data or logging a high volume of events over an extended period.

Lack of Advanced Features: The Experiment Logger provides a basic set of functionalities for experiment logging and data recording. Advanced features such as data analysis, synchronization with external devices, or real-time data visualization are not included in the current implementation. These features may require additional development or integration with other tools.

Future Scopes:

Enhanced Data Analysis: Currently, the Experiment Logger focuses on data recording and logging. A future enhancement could involve incorporating advanced data analysis capabilities. This could include statistical analysis, data visualization, and integration with popular data analysis libraries such as NumPy and pandas.

Real-Time Data Visualization: Adding real-time data visualization capabilities would provide users with instant feedback and insights during the experiment. This could include live graphs, charts, or visual representations of logged data, allowing researchers to monitor and analyze the data as it is being recorded.

Integration with External Devices: To expand the functionality and flexibility of the Experiment Logger, future enhancements could involve integrating with external devices commonly used in experiments. This could include synchronizing with eye-tracking devices, biometric sensors, or other data acquisition systems to capture additional information during the experiment.

Customizable User Interface: Providing users with the ability to customize the user interface (UI) would enhance the user experience. This could include options to adjust the layout, theme, and preferences of the Experiment Logger's UI, allowing researchers to tailor the interface to their specific needs and preferences.

Multi-User Support: If the need arises to conduct experiments involving multiple participants simultaneously, future enhancements could include adding multi-user support. This would involve implementing mechanisms to handle data recording and logging for multiple users concurrently while ensuring data separation and organization.

Error Handling and Logging: Improving error handling and logging mechanisms would assist users in identifying and troubleshooting issues that may arise during the experiment. Enhancements could involve implementing comprehensive error handling, error messages, and logging features that provide detailed information about any errors or exceptions encountered.

Exporting and Data Sharing: Enabling the export and sharing of experiment data in various formats would enhance collaboration and facilitate data analysis. Future enhancements could include options to export logged data in commonly used file formats such as CSV or Excel, as well as integration with cloud storage or collaboration platforms for easy data sharing.

Cross-Platform Compatibility: While the Experiment Logger currently supports a specific platform, future enhancements could focus on achieving cross-platform compatibility. This would involve ensuring that the code works seamlessly on multiple operating systems, such as Windows expanding the reach and usability of the software.

Conclusion

The Experiment Logger is a user-friendly tool designed to streamline the process of experiment management, data recording, and information logging. By providing an intuitive GUI and comprehensive features, it empowers researchers to efficiently conduct experiments and gather valuable data for analysis and further research.

Installation Guide

Follow these steps to install and set up the Experiment Logger:

Step-1: Clone the Repository: Start by cloning the Experiment Logger repository from GitHub using the following command on bash:

git clone https://github.com/your-username/experiment-logger.git

Step-2: Install Dependencies: Navigate to the project directory and install the required dependencies by running the following command:

pip install -r requirements.txt

This will install all the necessary libraries, including tkinter, pynput, pyaudio, wave, pytz, and threading.

Step-3: Verify Installation: To verify that the installation was successful, run a test script provided in the repository. Execute the following command:

python test_script.py

If the script runs without any errors and the GUI window opens, the installation was successful.

Step-4: Run the Experiment Logger: You can now run the Experiment Logger by executing the main script:

python experiment_logger.py

This will launch the graphical user interface (GUI) for the Experiment Logger.