The International Journal of Technology, Knowledge and Society
ISSN: 1832-3669 (Print), ISSN: 2835-0391 (Online)
Volume 20, Issue x, 2024
https://doi.org/10.18848/0000-0000/CGP/v00i00/0-0

Original Research

Free Software for Institutional Review Board Compliance

Brayden Theisen, St Cloud State University
Dingfang Kan, St Cloud State University
Adriano Cavalcanti, St Cloud State University
Wei-Ying Hsaio, University of Alaska-Anchorage

Received: 07/12/2024; ***Accepted:*** xx/xx/2024; ***Published:*** xx/xx/2024

Abstract: Brain-computer interface is a new bleeding-edge technology integrating human-computer interaction for machines and systems. Our article explores developing and implementing an open-source automation software designed for Brain-Computer Interface (BCI) systems to ensure compliance with Institutional Review Board (IRB) standards for human subject data. The software implements protocols that help obtain ethical approvals for human subject research by automating documentation and protecting volunteers' personal information, thus enhancing security, efficiency, and transparency. The article delves into the features of BCI open-source automation software. It discusses its potential impact on advancing a neuroscientific framework for practical research while upholding ethical standards mandated by IRBs. As a study case for demonstration, we collected over 1800 brain readings to create an Avatar platform for flying drones with mind control that provides haptics and a real-time immersive reality. The innovative presented open source software is a valuable tool for automating confidentiality and improving human subject big data management. The system, which is an open-source code available on GitHub, can be used under the MIT License guidelines. The MIT License is known for its brevity and clarity. It grants permission to use, modify, and distribute the software with the condition that the original copyright notice and the license text are retained in the redistributed software.

Keywords: Automation, Brain-Computer Interface, Institutional Review Board, Spark, TensorFlow, Avatar.

1. Introduction

In recent years, advancements in Brain-Computer Interface (BCI) technology have paved the way for innovative applications in healthcare, research, and beyond. As researchers delve into the possibilities of BCI, ethical considerations and compliance with Institutional Review Board (IRB) regulations become paramount. To address these concerns, the development of open-source automation software for BCI and other Human Subject applications not only fosters collaboration but also ensures adherence to ethical standards.

Recent technological breakthroughs have significantly expanded the frontiers of human-machine interaction, paving the way for revolutionary applications across diverse industries. A notable example of this progress is the pioneering integration of Brain-Computer Interface (BCI) technology with Unmanned Aerial Vehicles (UAVs), commonly referred to as drones. This synergy of neuroscience and drone technology unlocks vast potential, offering unprecedented control and capabilities for drone operators. Furthermore, the BCI platform enables the exploration of novel haptic interfaces, heralding a new era in Human-Computer Interaction and user interface design.

2. Understanding BCI and IRB Compliance

BCI technology enables direct communication between the brain and external devices, offering groundbreaking opportunities for neuroscientific research [1], medical interventions, and assistive technologies (Figure 1). However, the ethical implications of working with human subjects in BCI studies necessitate stringent oversight and adherence to IRB guidelines.

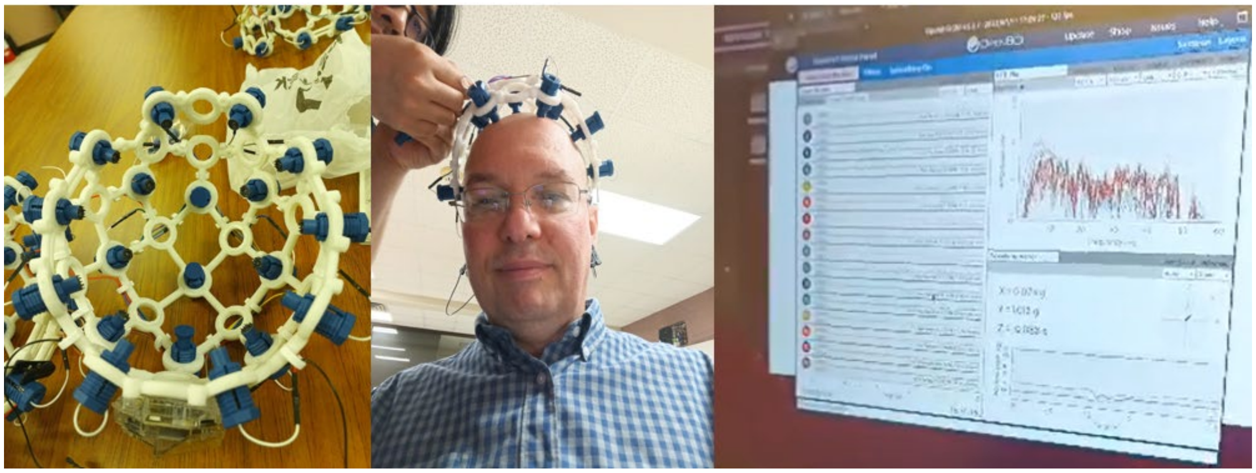


Figure 1: The process of calibrating and adjusting the EEG headset to capture live brain wave readings, which are then uploaded to our cloud-based big data server for analysis and storage.

The Institutional Review Board (IRB), a fundamental pillar of ethical research, ensures the safeguarding of human participants by rigorously evaluating the risks and benefits of proposed studies. Adherence to IRB regulations is essential to guarantee the ethical treatment of research subjects and the validity of study outcomes. Research involving human subjects is a crucial driver of scientific and medical progress, providing invaluable insights across various disciplines. However, to ensure the ethical and responsible conduct of such studies, obtaining proper authorization from human subjects is imperative. This authorization not only protects the well-being and rights of individual participants but also upholds the integrity of the research process. The primary rationale for seeking human subject authorization is to safeguard the rights and well-being of individuals participating in research studies. Human subjects have the inherent right to be treated with respect, dignity, and fairness throughout the research process.

Informed consent is typically obtained through a detailed and transparent process. It involves providing participants with comprehensive information about the study, allowing them to make informed decisions regarding their involvement. Informed consent ensures that individuals understand the nature of the research, the potential risks and benefits, the voluntary nature of their participation, and the measures in place to protect their confidentiality. Human subject authorization is closely tied to ethical considerations in research. Researchers must adhere to ethical guidelines to maintain the trust of participants and the broader community. Obtaining proper authorization helps researchers navigate potential ethical dilemmas, such as ensuring the confidentiality of participant information, minimizing harm, and promoting fairness in participant selection. Ethical research practices contribute to the credibility and reliability of research outcomes.

Many countries and institutions have specific regulations and laws governing research involving human subjects. Obtaining proper authorization ensures compliance with these legal requirements, protecting both researchers and participants. Failure to adhere to these regulations can lead to serious consequences, including legal action, reputational damage, and potential suspension of research activities. Researchers must establish clear protocols for data collection, storage, and dissemination to protect sensitive information. Obtaining proper authorization allows participants to understand how their data will be handled and ensures that researchers adhere to stringent privacy measures.

Building and maintaining trust between researchers and human subjects are essential for the success of any research endeavor. Proper authorization processes contribute to transparent communication, fostering trust and collaboration. When participants are confident that their rights are respected and their well-being is a top priority, they are more likely to engage in research activities, leading to more robust and reliable results.

3. Open Source Automation Software

Open-source software has gained popularity for fostering collaboration, transparency, and innovation in various fields [2]. In the realm of BCI research, open-source automation software plays a pivotal role in facilitating compliance with IRB regulations.

Our open-source BCI automation software provides transparency in the development and implementation of experimental protocols. Researchers and developers can collaborate to refine the software, enhancing its functionality and addressing any ethical concerns that may arise during the development process. IRB compliance often requires customization of experimental protocols to meet the specific needs of diverse research studies. Open-source BCI automation software allows researchers to tailor protocols according to their study objectives while ensuring adherence to ethical guidelines.

The open-source nature of the software encourages community input, fostering a collective approach to ethical oversight. Peer review and collaboration within the scientific community help identify and address potential ethical challenges, ultimately contributing to more robust and ethically sound research practices.

Open-source software is freely accessible, making it a cost-effective solution for researchers with limited resources. This accessibility promotes inclusivity in BCI research, allowing a broader range of researchers to engage in ethical and compliant studies.

4. Methods

Automation software can include features for real-time monitoring of BCI experiments, ensuring that researchers can promptly respond to any unforeseen ethical issues. Additionally, robust data security measures can be implemented to safeguard sensitive information, addressing another critical aspect of IRB compliance [3].

Bash programming, also known as Bash scripting, is the process of writing scripts in the Bash shell programming language. Bash, which stands for "Bourne-Again SHell," is a Unix shell and command-line interpreter written by Brian Fox for the GNU Project. Bash programming involves writing scripts that use Bash commands, syntax, and features to automate tasks, manipulate data, and interact with the operating system. Bash scripts typically have a .sh extension and are executed by the Bash shell.

Bash programming is commonly used for:

- System administration: Automating system tasks, backups, and maintenance.
- Data processing: Manipulating text files, data analysis, and reporting.
- Software development: Building tools, automating builds, and testing.
- System integration: Integrating different systems, services, and applications.

Bash programming features include: Variables; Conditionals (if/then/else); Loops (for/while/do); Functions; Arrays; File operations (read/write/copy); Command substitution; Regular expressions. Bash scripts can be used for simple tasks, like automating backups, or complex tasks, like building software packages. Bash programming is a powerful tool for automating tasks and simplifying system administration.

Bash can invoke a diverse range of programs, which are part of a comprehensive toolset designed to automate tasks [4]. These tools are typically developed using programming languages such as C, Bash, Rust, or Assembly, and are integrated into the system to provide efficient automation capabilities.

4.1 Crontab and Cron

Crontab, the cron daemon, was not implemented using Bash programming [5]. The original cron daemon was written in C programming language. Crontab, the cron daemon and the cron table, was created by Brian Kernighan in 1975, while he was working at Bell Labs. At the time, Kernighan was part of a team developing the Unix operating system. The concept of cron was inspired by the clockwork mechanisms of the ancient Greeks, which used gears and levers to automate tasks. The name "cron" comes from the Greek word "chronos," meaning time.

Kernighan's implementation of cron was designed to automate system maintenance tasks, such as backups and disk cleanups, by scheduling them to run at specific times or intervals. The crontab format, with its five fields (minute, hour, day, month, and day of the week), was also introduced by Kernighan. Since then, cron has become a standard tool in Unix-like operating systems, including Linux and macOS, and is widely used for automating tasks and workflows.

- **Crontab Configuration:** A user creates a crontab file using the crontab -e command, which opens a text editor to configure the schedule. The file contains a list of tasks, each specified by five fields: minute, hour, day of the month, month, and day of the week, followed by the command to be executed.
- **Cron Daemon:** The cron daemon, a background process, continuously runs on the system, scanning the crontab files for tasks to execute. It checks the schedules and compares them to the current system time.
- **Task Execution:** When the cron daemon finds a task with a matching schedule, it executes the specified command. The command is run as the user who owns the crontab file, with the same environment and permissions.

- **Logging:** After executing the task, the cron daemon logs the output and any errors to the system log files, typically /var/log/cron or /var/log/syslog. This allows users to monitor and troubleshoot their scheduled tasks.
- **Differences:** cron refers to the cron daemon (crond), a system process that runs in the background and executes scheduled tasks. It is the engine that runs the scheduled jobs. crontab, on the other hand, refers to the table or file that contains the scheduled tasks. It is the configuration file that defines the jobs to be executed by the cron daemon. In other words: cron is the program that runs the scheduled tasks; crontab is the file that contains the schedule and commands to be executed.

4.2 Systemctl

Systemctl is a system management tool and command-line utility that is part of the systemd software suite. It is used to manage and control system services, daemons, and units on Linux-based systems. Systemctl was implemented by Lennart Poettering and Kay Sievers as part of the systemd project, which was started in 2009. The first version of systemd was released in 2010, and systemctl was introduced as a replacement for the traditional init system and service management tools like sysvinit and init scripts. Systemctl is written in C and uses the systemd libraries to interact with the system. It provides a unified interface for managing system resources, including: Services (daemons); Sockets; Devices; Mount points; Timers.

Systemctl allows users to start, stop, restart, enable, and disable system services, as well as monitor their status and configure their behavior [6]. Lennart Poettering, a German software engineer, is the primary author of systemd and systemctl. He is credited with designing and implementing the systemd architecture, which has become a widely adopted standard in the Linux world. Kay Sievers, also a German software engineer, was a key contributor to the systemd project and helped develop systemctl. Systemctl has become an essential tool for Linux system administration and is widely used in many Linux distributions, including Ubuntu, Debian, Fedora, AlmaLinux, and Red Hat Enterprise Linux.

4.3 Count files

The find command can be used in combination with the wc (word count) command to automatically count the number of files in a directory and its subdirectories [7]. Here's an example:

```
find . -type f | wc -l
```

This command works as follows:

- find searches for files in the current directory (.) and its subdirectories.

- -type f specifies that only files (not directories) should be counted.

- The output of find is piped (|) to wc -l, which counts the number of lines (in this case, the number of files found).

The result is the total count of files in the directory and its subdirectories.

If you want to count files in a specific directory only (not including subdirectories), use:

```
find . -maxdepth 1 -type f | wc -l
```

The -maxdepth 1 option limits the search to the current directory only.

When utilized correctly and efficiently, the find command proves to be a straightforward yet potent tool for automatically determining the number of files requiring processing.

4.4 Rename and mv

The mv and rename commands in Linux were created by the GNU Project, a free software foundation established by Richard Stallman in 1983.

mv (short for "move") was written by Mike Parker, David MacKenzie, and Jim Meyering. It is used to move or rename files and directories.

rename (also known as perl-rename) was written by Larry Wall, the creator of Perl. It is used to rename files using Perl regular expressions.

Both commands can be used for automation in various ways:

- **Batch renaming:** Use rename to rename multiple files at once based on patterns or criteria.
- **File organization:** Use mv to move files to different directories based on their names, extensions, or other attributes.
- **Scripting:** Incorporate mv and rename into shell scripts to automate file management tasks.
- **Data processing:** Use mv and rename in conjunction with other commands to process and transform data.

Examples:

- mv *.txt ~/Documents (move all text files to the Documents directory)

- rename 's/old/new/' *.txt (rename all text files, replacing "old" with "new")

These commands are essential tools for automating file management tasks in Linux [8], and are often used in combination with other commands and scripting languages to streamline workflows.

4.5 The birth of a file

The commands ctime, atime, mtime, crtime, and stat are implemented in C programming language, as part of the GNU Coreutils package.

Here's a brief overview of each command and their usefulness:

- ctime: Displays the last change time of a file's metadata (permissions, ownership, etc.).

- atime: Displays the last access time of a file (when it was read or executed).

- mtime: Displays the last modification time of a file (when its contents were changed).

- crtime: Not a standard command, but some systems provide it to display the file creation time (also known as "birth time").

- stat: Displays detailed information about a file, including its metadata, birth time, access times, and modification times.

To check when a file was created (its "birth time"), you can use:

- stat -c %W FILENAME (on some systems, like Linux)

- stat -f %B FILENAME (on macOS)

- Get-File FILENAME | Select-Object -ExpandProperty CreationTime (on PowerShell)

Note that the availability of the crtime command and the stat options may vary depending on your operating system and file system.

These commands are useful for various purposes, such as:

- Tracking file modifications and access

- Monitoring system activity

- Debugging issues related to file changes

- Data forensics and auditing

- Backup and version control systems

Keep in mind that the crtime command is not widely supported, and the stat command may have different options on different systems.

The stat command can be used to verify the birth or creation time of a file, but it depends on the file system and operating system you're using.

On Linux, stat uses the %W format specifier to display the file's birth time (also known as creation time or crtime). For example:

```
stat -c %W FILENAME
```

On macOS, stat uses the %B format specifier to display the file's birth time:

```
stat -f %B FILENAME
```

However, not all file systems support storing birth times. For example:

- ext4 and XFS file systems on Linux do support birth times.

- HFS+ and APFS file systems on macOS do support birth times.

- NTFS and FAT file systems on Windows do not support birth times.

If the file system doesn't support birth times, stat will not be able to display the creation time, and you may see an error message or a default value like "0" or "1970-01-01".

In summary, stat can be used to verify the birth of a file, but it depends on the file system and operating system you're using, and not all file systems support storing creation times.

4.6 Network Time Protocol

The Linux system time can be precisely controlled by executing these commands in a specific sequence [9]:

- **set-ntp false:** This command disables Network Time Protocol (NTP) synchronization, allowing you to manually set the system time without interference from NTP.
- **set-time "specific date":** This command sets the system time to a specific date and time. Replace "specific date" with the desired date and time in the format YYYY-MM-DD HH:MM:SS.
- **set-ntp true:** This command re-enables NTP synchronization, allowing the system to sync its time with NTP servers.
- **systemctl restart chrony:** This command restarts the Chrony service, which is responsible for NTP synchronization. This ensures that the changes made to the system time are applied and synchronized with NTP servers.

This process allows you to temporarily set a specific system time for testing or development purposes, while also ensuring that the system time is synchronized with NTP servers afterwards.

4.7 Changing the birth of a file

The Linux touch command is used to create a new file or update the timestamp of an existing file. However, it has some limitations:

- **Cannot change the birth time:** touch can only update the last modification (mtime) and last access (atime) times, but not the creation time (birth time) of a file.
- **Cannot set arbitrary timestamps:** touch can only set the timestamp to the current system time or a specific timestamp in the past (using the -d option). It cannot set a future timestamp.
- **Dependent on system clock:** touch relies on the system clock to set the timestamp, so if the system clock is incorrect or outdated, the timestamp will also be incorrect.

To overcome these limitations and set a new birth time for a file, you may need to change the operating system time. Here's why:

- **File system limitations:** Some file systems, like ext4, store the creation time in the file's inode. Changing the system time allows you to set a new creation time for the file.
- **Timestamp precision:** Changing the system time allows you to set a specific timestamp with precision, which may not be possible with touch alone.
- **Future timestamps:** By changing the system time, you can set a future timestamp for a file, which is not possible with touch.

To change the operating system time, you can use the date command (e.g., sudo date --set="2022-07-11 12:00:00"). However, be cautious when changing the system time, as it can affect other system processes and applications that rely on the accurate system clock.

It's important to note that altering the system time is an unconventional practice [10], reserved for specific circumstances such as testing or development scenarios. In our particular case, modifying the file's creation date serves a singular purpose: to guarantee the absolute anonymity of brain wave data generously provided by volunteers. By adjusting the file's creation date, we ensure that the data remains untraceable to the individual's identity, thereby maintaining their privacy and confidentiality.

5. Implementation

BCI is a technology that establishes a direct communication pathway between the brain and an external device or system, allowing users to control devices or interact with the environment using their thoughts. Electroencephalography (EEG) is a common method employed in BCI, where electrodes placed on the scalp detect electrical signals produced by the brain. OpenBCI (Open Brain-Computer Interface) is a platform for BCI research and development. The typical data file format for brain wave readings from OpenBCI is a text-based format called "OpenBCI Data File" or "ODF".

An ODF file contains the following content:

- **Header:** Metadata about the recording, such as:

  - Device information (e.g., OpenBCI board version)

  - Sampling rate (e.g., 250 Hz)

  - Channel count (e.g., 8 channels)

  - Start and end timestamps

- **Data:** Brain wave data, represented as:

  - Time series data (one column per channel)

  - Values are in microvolts (μV)

  - Sampling rate is uniform (e.g., 250 Hz)

- **Footer:** Additional metadata, such as:

  - Data quality metrics (e.g., signal quality, noise levels)

  - Event markers (e.g., stimuli onset, subject responses)

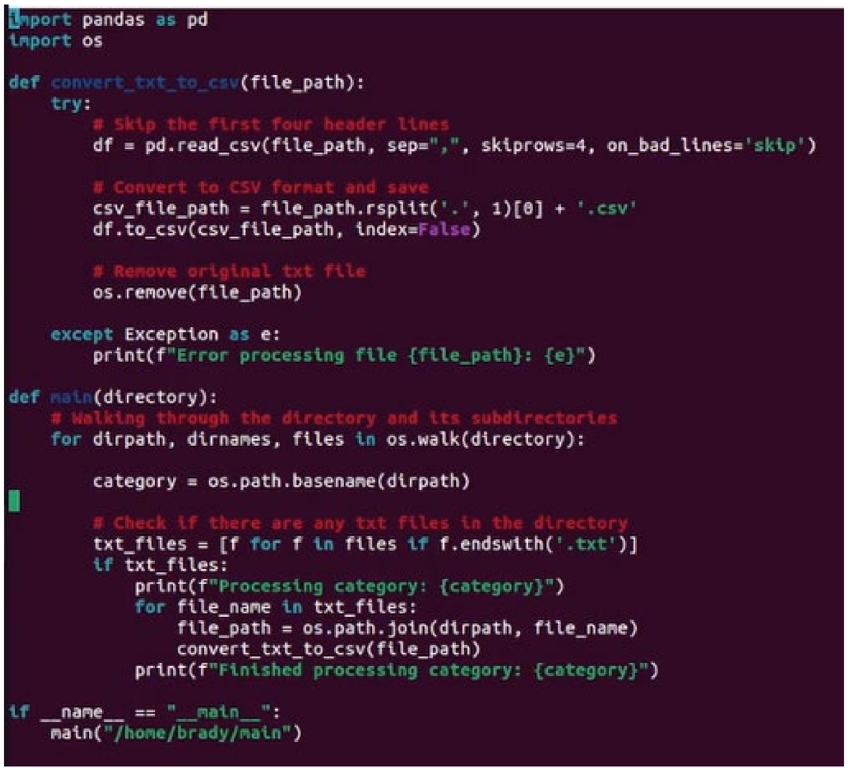
The data is typically stored in a plain text file with a .txt or .odf extension. Each line represents a single sample, with columns separated by spaces or tabs. Here's an example of an ODF file snippet:

```
# OpenBCI Data File
# Device: OpenBCI Ganglion
# Sampling Rate: 250 Hz
# Channels: 8
# Start Time: 1643723456
# End Time: 1643723510
0.000000 1.234567 2.345678 3.456789 4.567890 5.678901 6.789012 7.890123
0.004000 1.234567 2.345678 3.456789 4.567890 5.678901 6.789012 7.890123
0.008000 1.234567 2.345678 3.456789 4.567890 5.678901 6.789012 7.890123
...
```

This format allows for easy import and analysis of brain wave data in various software tools, such as Python libraries like Pandas and NumPy, or specialized BCI software like OpenBCI's own GUI tools.

5.1 Pandas

BCI brain wave readings can be computationally effective, depending on the specific application and analysis requirements. By converting the txt file to csv, you can take advantage of pandas' powerful data analysis and manipulation tools, making it a computationally effective way to work with BCI brain wave data (Figure 2).



```
import pandas as pd
import os

def convert_txt_to_csv(file_path):
    try:
        # Skip the first four header lines
        df = pd.read_csv(file_path, sep=",", skiprows=4, on_bad_lines='skip')

        # Convert to CSV format and save
        csv_file_path = file_path.rsplit('.', 1)[0] + '.csv'
        df.to_csv(csv_file_path, index=False)

        # Remove original txt file
        os.remove(file_path)

    except Exception as e:
        print(f"Error processing file {file_path}: {e}")

def main(directory):
    # Walking through the directory and its subdirectories
    for dirpath, dirnames, files in os.walk(directory):

        category = os.path.basename(dirpath)

        # Check if there are any txt files in the directory
        txt_files = [f for f in files if f.endswith('.txt')]
        if txt_files:
            print(f"Processing category: {category}")
            for file_name in txt_files:
                file_path = os.path.join(dirpath, file_name)
                convert_txt_to_csv(file_path)
            print(f"Finished processing category: {category}")

if __name__ == "__main__":
    main("/home/brady/main")
```

Figure 2: A component of the system showcasing the implementation of our program that utilizes Pandas to efficiently convert txt files to CSV format.

To convert a txt file into csv using Python pandas, you can follow these steps:

- Import the pandas library:

  import pandas as pd

- Load the txt file into a pandas DataFrame:

  df = pd.read_csv('data.txt', delim_whitespace=True)

The delim_whitespace=True parameter specifies that the columns are separated by whitespace (spaces or tabs).

- Save the DataFrame to a csv file:

  df.to_csv('data.csv', index=False)

The index=False parameter omits the row index from the csv file.

Here's an example of conversion:

```
import pandas as pd
df = pd.read_csv('data.txt', delim_whitespace=True)
df.to_csv('data.csv', index=False)
```

This code assumes that the txt file has a simple structure with columns separated by whitespace. If your file has a more complex structure, you may need to adjust the read_csv parameters accordingly. Some benefits of using pandas for this conversion include: efficient handling of large datasets; automatic data type detection and conversion; easy data manipulation and analysis capabilities (Figure 2).

5.2 Applying BCI to Drone Control

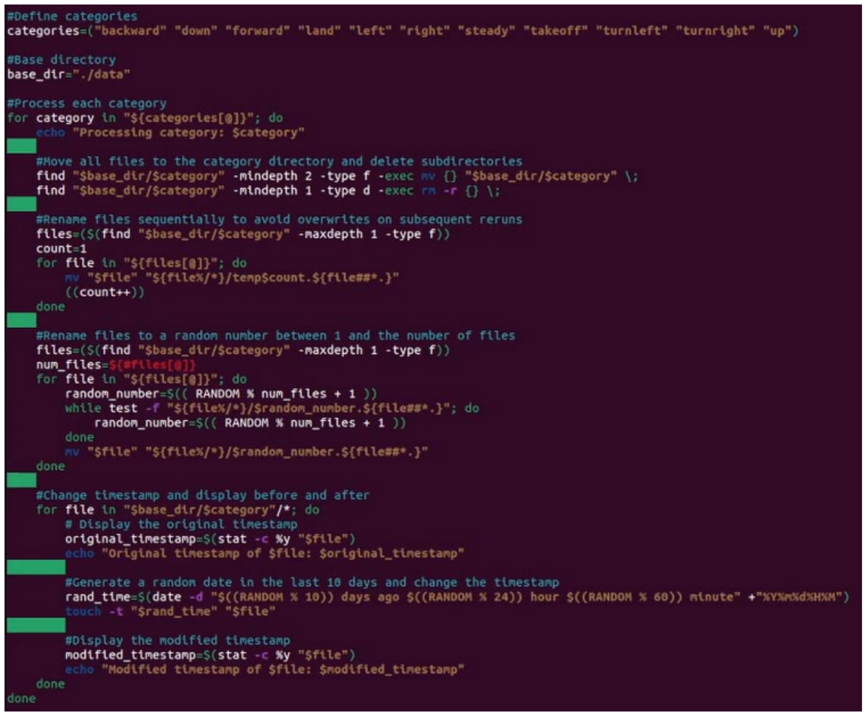
The application of BCI to drone control involves translating brain signals into actionable commands for the UAV. This opens up a realm of possibilities, providing a more intuitive and direct way for operators to guide drones. With BCI, users can potentially control flight movements, navigate through obstacles, and even capture images or videos by merely thinking about the desired actions.

- **Advantages of BCI-enabled Drone Control:** BCI offers a more natural and intuitive means of drone control, eliminating the need for complex remote controls or manual joystick inputs. Operators can guide drones effortlessly, streamlining the learning curve for beginners. Individuals with physical disabilities may find BCI-enabled drone control to be a liberating experience. The technology opens up opportunities for those who may face challenges operating traditional remote controls.
- **Real-time Adaptability:** BCI allows for real-time adjustments based on the operator's cognitive state. Drones can respond dynamically to changes in focus, attention, or intention, offering a more adaptable and responsive flying experience. With BCI, operators can control drones without the need for hands-on devices. This hands-free operation is particularly advantageous in situations where manual control is impractical or unsafe.
- **Pattern Recognition and Classification:** By leveraging machine learning algorithms, we can train systems to identify patterns in brain activity linked to specific thoughts or intentions, such as "move left" or "take off." Our approach utilizes TensorFlow to analyze EEG data and classify brain signals into distinct categories, enabling accurate identification of user intentions. To ensure optimal system performance with minimal latency, we employ Spark for scalable big data pipeline processing. Furthermore, integrating our solution with real-time drone camera feeds provides a comprehensive avatar haptics visualization, offering a birds-eye view of the environment.

6. Results

Human subject authorization is not merely a procedural formality; it is a fundamental aspect of ethical and responsible research. By prioritizing the protection of human rights, ensuring informed consent, adhering to ethical guidelines, and complying with legal requirements, researchers can conduct studies that contribute meaningfully to knowledge while respecting the dignity and well-being of those who participate. As the scientific community continues to advance, the emphasis on human subject authorization remains critical to maintaining the integrity and credibility of research practices.

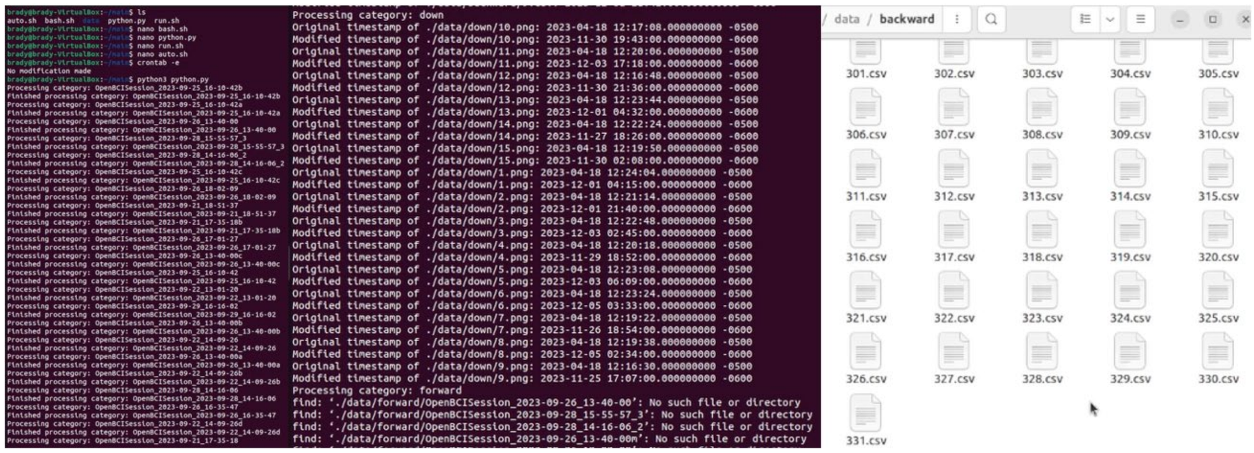
Anonymization is a cornerstone of human subject privacy, and its implementation is a crucial aspect of our work, as illustrated in Figure 3.



```
#Define categories
categories=("backward" "down" "forward" "land" "left" "right" "steady" "takeoff" "turnleft" "turnright" "up")

#Base directory
base_dir="./data"

#Process each category
for category in "${categories[@]}"; do
    echo "Processing category: $category"

    #Move all files to the category directory and delete subdirectories
    find "$base_dir/$category" -mindepth 2 -type f -exec mv {} "$base_dir/$category" \;
    find "$base_dir/$category" -mindepth 1 -type d -exec rm -r {} \;

    #Rename files sequentially to avoid overwrites on subsequent reruns
    files=($(find "$base_dir/$category" -maxdepth 1 -type f))
    count=1
    for file in "${files[@]}"; do
        mv "$file" "${file%/*}/temp$count.${file##*.}"
        ((count++))
    done

    #Rename files to a random number between 1 and the number of files
    files=($(find "$base_dir/$category" -maxdepth 1 -type f))
    num_files=${#files[@]}
    for file in "${files[@]}"; do
        random_number=$(( RANDOM % num_files + 1 ))
        while test -f "${file%/*}/$random_number.${file##*.}"; do
            random_number=$(( RANDOM % num_files + 1 ))
        done
        mv "$file" "${file%/*}/$random_number.${file##*.}"
    done

    #Change timestamp and display before and after
    for file in "$base_dir/$category"/*; do
        # Display the original timestamp
        original_timestamp=$(stat -c %y "$file")
        echo "Original timestamp of $file: $original_timestamp"

        #Generate a random date in the last 10 days and change the timestamp
        rand_time=$(date -d "$((RANDOM % 10)) days ago $((RANDOM % 24)) hour $((RANDOM % 60)) minute" +"%Y%m%d%H%M")
        touch -t "$rand_time" "$file"

        #Display the modified timestamp
        modified_timestamp=$(stat -c %y "$file")
        echo "Modified timestamp of $file: $modified_timestamp"
    done
done
```

Figure 3: A component of the system that implements file shuffling to ensure the anonymization of brain wave data, thereby safeguarding participant privacy.

In an era dominated by digital information, the protection of sensitive data has become paramount. Organizations handling vast amounts of data often seek innovative solutions to safeguard the privacy of their users and clients. Server automation, specifically in the realm of file anonymization, has emerged as a powerful tool in this endeavor. Our system addresses the crucial intricacies of server automation and its role in anonymizing file data, examining the benefits, challenges, and best practices involved in implementing such systems (Figure 4).



```
brady@brady-VirtualBox:~/main$ ls
auto.sh  bash.sh  data  python.py  run.sh
brady@brady-VirtualBox:~/main$ nano bash.sh
brady@brady-VirtualBox:~/main$ nano python.py
brady@brady-VirtualBox:~/main$ nano run.sh
brady@brady-VirtualBox:~/main$ nano auto.sh
brady@brady-VirtualBox:~/main$ crontab -e
No modification made
brady@brady-VirtualBox:~/main$ python3 python.py
Processing category: OpenBCISession_2023-09-25_16-10-42b
Finished processing category: OpenBCISession_2023-09-25_16-10-42b
Processing category: OpenBCISession_2023-09-25_16-10-42a
Finished processing category: OpenBCISession_2023-09-25_16-10-42a
Processing category: OpenBCISession_2023-09-26_13-40-00
Finished processing category: OpenBCISession_2023-09-26_13-40-00
Processing category: OpenBCISession_2023-09-28_15-55-57_3
Finished processing category: OpenBCISession_2023-09-28_15-55-57_3
Processing category: OpenBCISession_2023-09-28_14-16-06_2
Finished processing category: OpenBCISession_2023-09-28_14-16-06_2
Processing category: OpenBCISession_2023-09-25_16-10-42c
Finished processing category: OpenBCISession_2023-09-25_16-10-42c
Processing category: OpenBCISession_2023-09-26_18-02-09
Finished processing category: OpenBCISession_2023-09-26_18-02-09
Processing category: OpenBCISession_2023-09-21_18-51-37
Finished processing category: OpenBCISession_2023-09-21_18-51-37
Processing category: OpenBCISession_2023-09-21_17-35-18b
Finished processing category: OpenBCISession_2023-09-21_17-35-18b
Processing category: OpenBCISession_2023-09-26_17-01-27
Finished processing category: OpenBCISession_2023-09-26_17-01-27
Processing category: OpenBCISession_2023-09-26_13-40-00c
Finished processing category: OpenBCISession_2023-09-26_13-40-00c
Processing category: OpenBCISession_2023-09-25_16-10-42
Finished processing category: OpenBCISession_2023-09-25_16-10-42
Processing category: OpenBCISession_2023-09-22_13-01-20
Finished processing category: OpenBCISession_2023-09-22_13-01-20
Processing category: OpenBCISession_2023-09-29_16-16-02
Finished processing category: OpenBCISession_2023-09-29_16-16-02
Processing category: OpenBCISession_2023-09-26_13-40-00b
Finished processing category: OpenBCISession_2023-09-26_13-40-00b
Processing category: OpenBCISession_2023-09-22_14-09-26
Finished processing category: OpenBCISession_2023-09-22_14-09-26
Processing category: OpenBCISession_2023-09-26_13-40-00a
Finished processing category: OpenBCISession_2023-09-26_13-40-00a
Processing category: OpenBCISession_2023-09-22_14-09-26b
Finished processing category: OpenBCISession_2023-09-22_14-09-26b
Processing category: OpenBCISession_2023-09-28_14-16-06
Finished processing category: OpenBCISession_2023-09-28_14-16-06
Processing category: OpenBCISession_2023-09-26_16-35-47
Finished processing category: OpenBCISession_2023-09-26_16-35-47
Processing category: OpenBCISession_2023-09-22_14-09-26d
Finished processing category: OpenBCISession_2023-09-22_14-09-26d
Processing category: OpenBCISession_2023-09-21_17-35-18
```

```
Processing category: down
Original timestamp of ./data/down/10.png: 2023-04-18 12:17:08.000000000 -0500
Modified timestamp of ./data/down/10.png: 2023-11-30 19:43:00.000000000 -0600
Original timestamp of ./data/down/11.png: 2023-04-18 12:20:06.000000000 -0500
Modified timestamp of ./data/down/11.png: 2023-12-03 17:18:00.000000000 -0600
Original timestamp of ./data/down/12.png: 2023-04-18 12:16:48.000000000 -0500
Modified timestamp of ./data/down/12.png: 2023-11-30 21:36:00.000000000 -0600
Original timestamp of ./data/down/13.png: 2023-04-18 12:23:44.000000000 -0500
Modified timestamp of ./data/down/13.png: 2023-12-01 04:32:00.000000000 -0600
Original timestamp of ./data/down/14.png: 2023-04-18 12:22:24.000000000 -0500
Modified timestamp of ./data/down/14.png: 2023-11-27 18:26:00.000000000 -0600
Original timestamp of ./data/down/15.png: 2023-04-18 12:19:50.000000000 -0500
Modified timestamp of ./data/down/15.png: 2023-11-30 02:08:00.000000000 -0600
Original timestamp of ./data/down/1.png: 2023-04-18 12:24:04.000000000 -0500
Modified timestamp of ./data/down/1.png: 2023-12-01 04:15:00.000000000 -0600
Original timestamp of ./data/down/2.png: 2023-04-18 12:21:14.000000000 -0500
Modified timestamp of ./data/down/2.png: 2023-12-01 21:40:00.000000000 -0600
Original timestamp of ./data/down/3.png: 2023-04-18 12:22:48.000000000 -0500
Modified timestamp of ./data/down/3.png: 2023-12-03 02:45:00.000000000 -0600
Original timestamp of ./data/down/4.png: 2023-04-18 12:20:18.000000000 -0500
Modified timestamp of ./data/down/4.png: 2023-11-29 18:52:00.000000000 -0600
Original timestamp of ./data/down/5.png: 2023-04-18 12:23:08.000000000 -0500
Modified timestamp of ./data/down/5.png: 2023-12-03 06:09:00.000000000 -0600
Original timestamp of ./data/down/6.png: 2023-04-18 12:23:24.000000000 -0500
Modified timestamp of ./data/down/6.png: 2023-12-05 03:33:00.000000000 -0600
Original timestamp of ./data/down/7.png: 2023-04-18 12:19:22.000000000 -0500
Modified timestamp of ./data/down/7.png: 2023-11-26 18:54:00.000000000 -0600
Original timestamp of ./data/down/8.png: 2023-04-18 12:19:38.000000000 -0500
Modified timestamp of ./data/down/8.png: 2023-12-05 02:34:00.000000000 -0600
Original timestamp of ./data/down/9.png: 2023-04-18 12:16:30.000000000 -0500
Modified timestamp of ./data/down/9.png: 2023-11-25 17:07:00.000000000 -0600
Processing category: forward
find: './data/forward/OpenBCISession_2023-09-26_13-40-00': No such file or directory
find: './data/forward/OpenBCISession_2023-09-28_15-55-57_3': No such file or directory
find: './data/forward/OpenBCISession_2023-09-28_14-16-06_2': No such file or directory
find: './data/forward/OpenBCISession_2023-09-26_13-40-00m': No such file or directory
```

Figure 4: System workflow illustrating the sequential processing steps, with the conversion of files to CSV format, followed by file renaming, and finally, updating the file creation dates according to their respective categories.

The proliferation of digital data has led to an increased focus on data privacy and security. As organizations collect and store large volumes of information, the need to protect sensitive data from unauthorized access and breaches becomes more critical. Server automation, with its ability to streamline processes and reduce human intervention, has proven to be an effective means of fortifying data privacy. Our implementation aims to shed light on the specific application of server automation in making file data anonymous, emphasizing its role in mitigating privacy risks.

File anonymization is a key aspect of data privacy, particularly when dealing with personally identifiable information (PII). Anonymizing files involves stripping them of any identifying information, rendering them devoid of any connection to specific individuals. This process is crucial for compliance with data protection regulations and building trust among users. Server automation offers a systematic and efficient approach to achieve file anonymization at scale.

Our system has showcased its proficiency in delivering robust server automation, significantly enhancing data privacy through the anonymization of file data. A detailed demonstration video is accessible at https://cgscholar.com/cg_event/events/T24en/proposal/72084, and the accompanying source code is freely available for public use on GitHub at https://github.com/3C-SCSU/Avatar. As organizations engage with the imperative to safeguard sensitive information and adhere to stringent privacy regulations, the implementation of automated file anonymization processes becomes indispensable. By grasping the principles of server automation, investigating diverse anonymization techniques, addressing potential challenges, and embracing best practices, organizations can strengthen their data privacy fortifications and establish a foundation of trust with their stakeholders.

7. Conclusion

The development and utilization of open-source automation software for BCI research mark a significant step towards ensuring IRB compliance and ethical research practices. By promoting transparency, collaboration, and community input, this software empowers researchers to conduct studies that not only advance our understanding of the human brain but also prioritize the well-being and ethical treatment of research participants. As BCI technology continues to evolve, the integration of open-source automation software becomes essential for creating a responsible and ethically grounded research environment.

The marriage of brain-computer interface technology with drone control represents a paradigm shift in the way we interact with unmanned aerial vehicles, systems and devices. The potential applications are vast, ranging from recreational drone flying to professional cinematography and surveillance. As research and development in this field progress, we can anticipate a future where the power of our thoughts propels drones to new heights, unlocking unprecedented levels of control and functionality. The convergence of neuroscience and drone technology is not only a testament to human ingenuity but also a signal of a more interconnected and advanced era in unmanned aerial systems.

While the integration of BCI with drones presents exciting possibilities, several challenges need to be addressed. Signal accuracy, reliability, and the potential for interference are critical considerations.

Ongoing research aims to improve the precision and robustness of BCI systems, ensuring consistent and accurate drone control. Future developments may involve the incorporation of artificial intelligence (AI) algorithms to enhance the interpretability of brain signals, enabling more sophisticated and nuanced control. Additionally, advancements in wearable BCI technology could make the integration process more seamless and user-friendly.

REFERENCES

[1] Knudson, John, Adriano Cavalcanti, and Maninder Singh. 2024. “Neuro-Haptic BCI Spark for TensorFlow Flying Avatars”, *24th IEEE EIT International Conference on Electro Information Technology*, Eau Claire, Wisconsin, USA, May.

[2] Fortunto, Laura, and Mark Galassi. 2021. “The case for free and open source software in research and scholarship”, *Phylosophical Transactions of the Royal Society A*, Vol 379, no. 2197, May.

[3] Ferretti, Agata, Marcello Ienca, Samia Hurst, and Effy Vayena. 2020. “Big data, biomedical research, and ethics review: new challenges for IRBs”, *Ethics & Human Research*, Vol 42, no. 5, pp 17-28, Sept.

[4] DeWitt, Peter E., Margaret A. Rebull, and Tellen D. Bennett. 2024. “Open source and reproducible and inexpensive infrastructure for data challenges and education”, *Scientific Data*, Vol. 11, Article number: 8.

[5] Zhu, Yongbo, Ee Haihong, and Meina Song. 2020. “A Scheduling System for Big Data Hybrid Computing Workflow”, *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China, Nov.

[6] Novotny, Georg, Simon Schwaiger, Lucas Muster, and Mohamed Aburaia. 2023. “On the Applicability of Docker Containers and systemd Services for Search and Rescue Applications”, *Proceedings of the Austrian Robotics Workshop*, Linz Austria.

[7] Anderson, Robyn. 2022. “Basics of Bash”. In: Edwards, D. (eds) *Plant Bioinformatics*. Methods in Molecular Biology, vol 2443. Humana, New York, NY.

[8] Campesato, Oswald. 2022. “Bash for Data Scientists”, ISBN-10 168392973X, *Mercury Learning and Information*, Dec.

[9] Dinar, Amina Elbatoul, Samir Ghouali, and Merabet Boualem. 2021. “NTP Security by Delay-based Detection in Intelligent Defense Systems”, Journal of Telecommunication, Electronic and Computer Engineering, Vol 13, No. 1, Jan.

[10] Thierry, Aurélien, and Tilo Müller. 2022. “A systematic approach to understanding MACB timestamps on Unix-like systems”, *Forensic Science International: Digital Investigation*, Supplement, Volume 40, 301338, April.