A Novel Low-Cost Inner Speech Recognition Brain-Computer Interface & Integrated Machine Learning Model to Assist Differently Abled and Neuromuscular Disorder Patients in Home Automation

Anish Kalra https://orcid.org/0000-0001-9232-7489

Trishay Naman https://orcid.org/0000-0003-3837-3091

Coppell High School

185 W Parkway Blvd, Coppell, TX 75019

ABSTRACT

More than 750 million people suffer from conditions preventing mobility, such as cerebral palsy, several neurodegenerative disorders, and other strenuous health conditions associated with age and cardiovascular health. Current solutions require patients to invest in expensive treatments like nursing homes, major lifestyle adjustments, and invasive technologies. A home appliance user-interface that accommodates for these patients' needs can be incorporated in daily lives through Brain-Computer Interface (BCI) technology. However, current technology requires prolonged attention and laborious concentration from the user to be interpreted accurately by algorithms. Additionally, machine learning algorithms have relatively low accuracy in detecting raw inner speech due to neuroplastic differences in thought patterns between patients. This project seeks to go beyond current research in brain signal interpretation by directly interpreting event related potential and its differentiation for inner speech. This combines four phases to create an end-to-end model for home automation: a hardware device, machine learning model, remote control technology, and statistical significance testing were constructed. First, the hardware phase was done to create an affordable autonomous system that unifies all phases for convenient calibration. Next, the machine learning algorithm used data from the IMAL Research Institute in Argentina. A linear discriminant analysis model (LDA), was tested amongst others (CNN, Random Forest, and SVM) and exhibited the highest accuracy of 0.87. Based on brain signals corresponding to the linguistic internal thoughts "Up", "Down", "Left", "Right" (lights on, lights off, temperature down, temperature up; respectively). Phase 3 converts output from the machine learning algorithm into output signals for the Raspberry Pi GPIO pins to home automation commands (Lights On/Off, Temp Up/Down). Finally, statistical analysis was performed, which showed significance at the p=0.01 significance level. Future improvements will utilize available wireless technology to make the device more convenient for patient use on the market and will integrate a wider model of diverse patient cases to account for neuroplasticity and increase accuracy.

KEYWORDS: beta wave, Brain-Computer Interface, Linear Discriminant Analysis Model (LDA), Electroencephalogram (EEG), Event Related Potential (ERP).

BACKGROUND INFORMATION

The leading causes of paralysis are stroke, spinal cord injury and multiple sclerosis [1]. Reports suggest that 12,000 to 15,000 people in the United States have amyotrophic lateral sclerosis (ALS) and approximately 17 million people have cerebral palsy, globally. The emergence of Brain Computer Interface (BCI) technology presents a compelling avenue to improve the lives of those that suffer from these debilitating conditions. Electroencephalogram devices can monitor brain signals from a user, extract relevant features, and translate them into a desired action. It has shown tremendous potential towards helping those with neuromuscular disorders or injuries that cause serious motor disabilities and hindering their normal communication with the outside world.

BCI is a direct communication tool between the human brain and any external device. Human BCI systems use brain activity and communicate with a computer to perform an action that is compatible with the human intention. BCI is recognized as a potential approach to establishing interactions through the use of human thinking. It is also used for intelligent assistive systems in medical fields and also in entertainment areas. In the medical field, BCI has been utilized to help disabled people control a wheelchair or robot with eye blinks. To study the activities of the brain, electrical activities of neurons are monitored through electroencephalogram (EEG). It can be measured either by invasive or noninvasive methods. One of the techniques to study the brain activity is to stimulate it by presenting a paradigm. The beta waves event related potential (ERP) embedded within the EEG signals can be used to analyze, improve and communicate human thinking. The beta waves signal is seen in the EEG signals as a rapid single potential change in response to a sensory, cognitive, or motor event. The signal's peak appears at an average of 300 milliseconds after the stimulus. The detection of the beta waves signals require the subject to correctly recognize the stimulus event to generate a strong and observable beta waves [2]. BCI allows a user to willingly send messages or control commands bypassing the brain’s natural output paths. There are different approaches for BCI implementation and beta waves BCI is a safe and noninvasive system. The user is required to wear a small headset carrying a set of electrodes to detect beta waves ERP. The system measures EEG signals from the human scalp and processes them in real time to detect beta waves ERP that reflect the user’s intent. EEG-based BCI can have three stages to process signals: preprocessing, feature extraction, and detection and classification of beta waves [2].

However, beta waves signals have been a barrier in streamlining current research to wide usability. beta waves signals require extremely intense and intentional concentration to be detected. Despite current research claiming to solve this problem, most solutions require a screen or consistent stimulus presented in order to emulate a consistent ERP signal.

PURPOSE

750 million individuals worldwide suffer from some paralysis-related disorder. The elderly are another population group that are at risk of motor dysfunction. Extraneous tasks and movement can worsen their health and exacerbate current conditions.

Expensive and invasive treatment and lifestyle procedures, therapy homes, and other status quo solutions often bar many across the world from seeking a solution, which can have devastating outcomes.

Current technology in this industry requires prolonged visual cues to activate stimuli, oftentimes requiring a great deal of mental energy for simple tasks.

Current technology is also unreliable and has a relatively low accuracy, typically exhibiting an accuracy of 60.0±15.0%

First, the machine learning algorithm must be compatible with the EEG nodes needed to match the hardware device that we have made. The EEG device created as a part of this project requires only 2 nodes, while the machine learning algorithm collected data for 8 nodes. Some accuracy might be lost transferring data from the hardware device to the model, since the algorithm is not trained on data directly from the hardware device. However, EEG data is generally standardized with error accounted for different conditions.

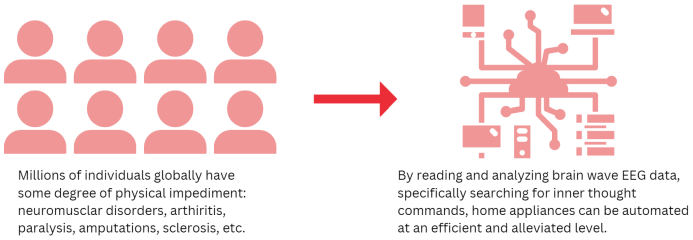
Second, significance testing should be conducted through an online library of EEG data (n = 2500) for a p-value below 0.01. The algorithm phase should explore multiple alternatives, including LDA, KNN, decision-tree, and different CNN architectures in binary classification to optimize the model.

Constraints: Budget of $300.00 and limited access to large datasets of EEGs.

ENGINEERING PROBLEM/SOLUTION

Engineering Problem: Over 750 million people worldwide suffer from paralysis, which often requires dire solutions that denigrate quality of life for those affected.

Engineering Objective: Develop an accessible solution for neuromuscular disorder patients to access appliance systems in their home through only brain signal output.



RESEARCH METHODOLOGY

This project consists of 4 phases that coheres the entire retinal screening process for broad spectrum diagnosis. The materials for each phase in the process are listed as below.

PHASE 1: Hardware

The communication between the client (EEG Pill) and server (Raspberry Pi) is implemented using User Datagram Protocol (UDP) over wireless LAN. In this communication protocol, the client can send messages (datagram) to the server on an Internet Protocol network.

Client Process - The client begins by creating a socket and naming it to server IP address. Then it attempts to connect to the server. If the connection request succeeds, the client proceeds to transfer data as necessary [11]. The flow chart of the Client process running on the BCI system is given in Figure 9 (a).

Server Process - Server process starts with a socket creation, bind it to a name. Client uses this to access the service [11]. The flow chart of the server process running on Raspberry Pi is given in Figure 9 (b).

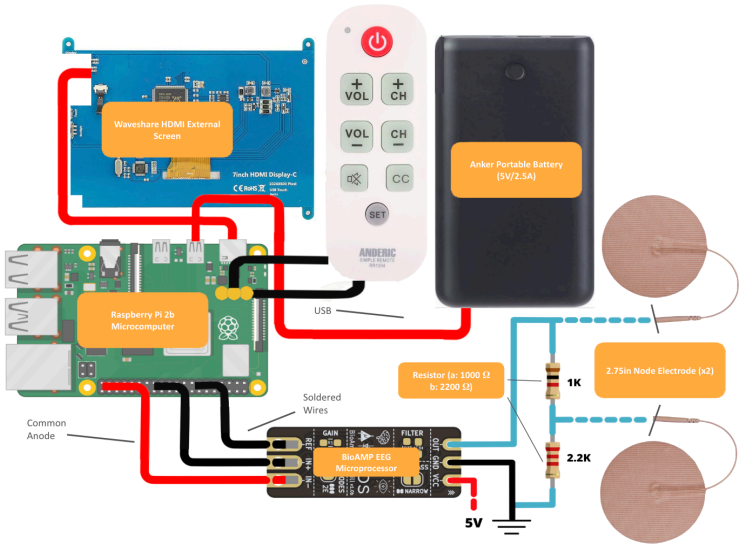
After establishing the UDP connection using the client-server protocol as described above, the client sends the identified command to the server. On the other end of the socket, a server is running on Raspberry Pi listening to a dedicated port. The server is waiting for the request, and once the request is received, the connection is established. The server functions are integrated in a single Python program on the Raspberry Pi which includes listening to the client (EPOC+ BCI system), receives the commands in string format, decode the string command, and process the commands to activate the respective interface. The Raspberry Pi interprets the received command and activate the relevant resources like GPIO, wireless power control, audio player apps, TV remote control. The Pi executes and completes the client request. The actions taken on various device control commands are described below.

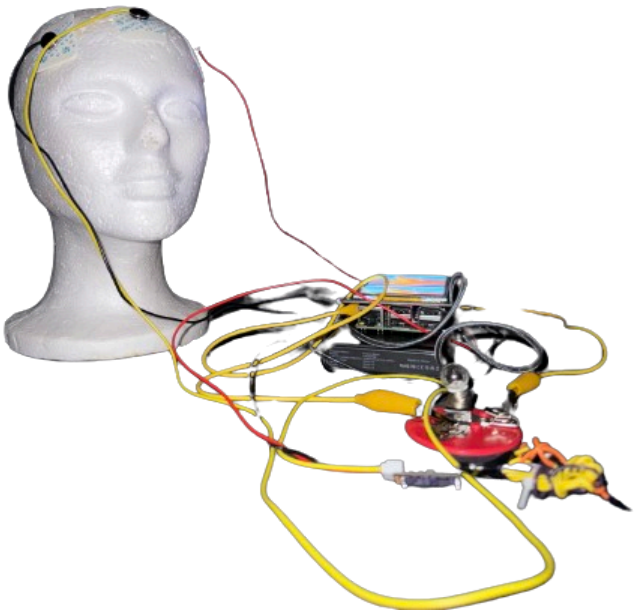
Light ON: The power outlet remote controller has independent ON and OFF switches. The ON switch is controlled by the GPIO signal and sends the ON signal to the power outlet to turn on the light.

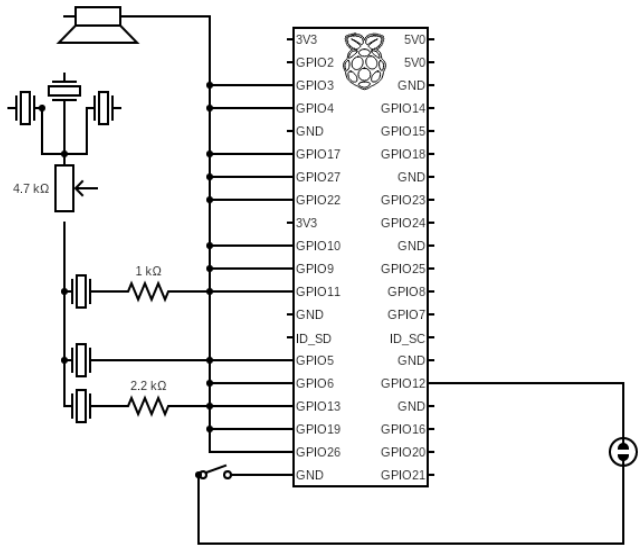
Light OFF: The OFF switch is controlled by the GPIO signal. When GPIO signal is activated, the OFF signal is sent to the power outlet and switch off the light.

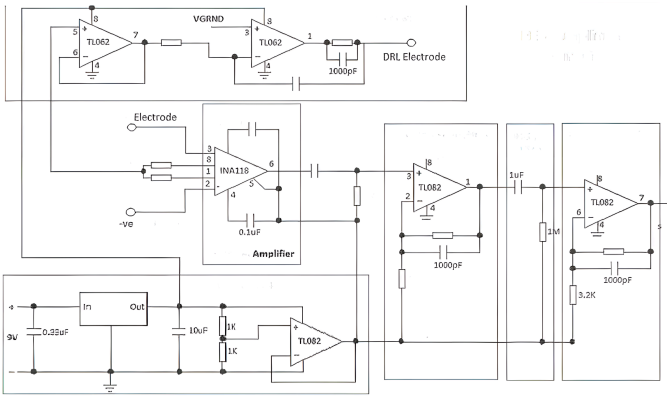
AC Up: AC Up-Down is controlled by RF remote controller. This remote has independent ON and OFF switches. When AC Up command is received, Raspberry Pi sends a pulse through the GPIO to toggle the Up switch which increases temperature by 1 degree.

AC Down: The Down switch is controlled by the GPIO signal. When GPIO signal is activated, the down signal is sent to the power outlet and turns down the temperature by 1 degree.









Although there are some existing solutions that aim to automate home appliances with brain wave data, such solutions are problematic:

- Physically Taxing: Most solutions require users to focus on specific commands for large amounts of time, thereby eliminating its applicability in everyday life.
- Invasive: Companies like Neuralink place a microchip inside users' brains to automate home appliances. Such a procedure has only worked on monkeys as of January 30th, 2024.

PHASE 2: Algorithms

Data Acquisition

The EEG data for this study was collected using a BDF (BioSemi Data Format) system from the IMAL Research Institute in Argentina (open source). The dataset comprises EEG recordings from multiple subjects, specifically during tasks involving inner speech. Files named eeg_data_02.bdf to eeg_data_07.bdf were downloaded and used for analysis. Each file corresponds to a different subject's EEG data, allowing for a comprehensive study across diverse brain patterns.

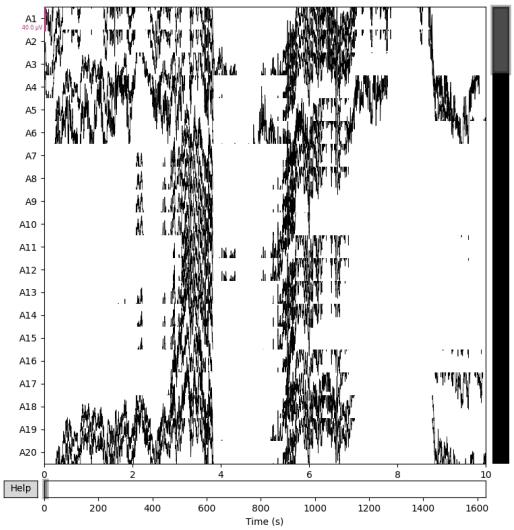
Preprocessing

Data Loading and Visualization

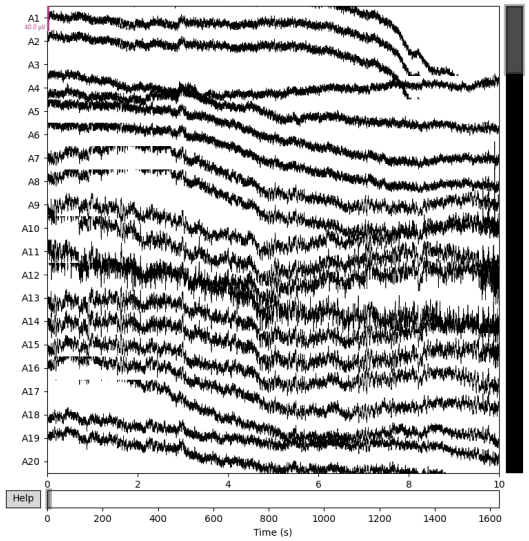
For each subject's data, the EEG recording was first loaded using the mne.io.read_raw_bdf function from the MNE library. This step was crucial for initial data visualization and to check for any apparent anomalies in the EEG signals. A plot for each dataset was generated to visually inspect the EEG signals. The graphs below represent the raw EEG data for all six individuals (subject 1 was skipped as data file was corrupted). The code that generated these plots is also provided.

```
import mne

file_paths = ['eeg_data_02.bdf', 'eeg_data_03.bdf', 'eeg_data_04.bdf
              'eeg_data_05.bdf', 'eeg_data_06.bdf', 'eeg_data_07.bdf

for i in range(len(file_paths)):
  # Read the .bdf file
  raw = mne.io.read_raw_bdf(file_paths[i], preload=True)
  # Define a title for the plot
  plot_title = f"EEG Data - {file_paths[i]}"
  # Plot the data with the title
  raw.plot(title=plot_title)
```

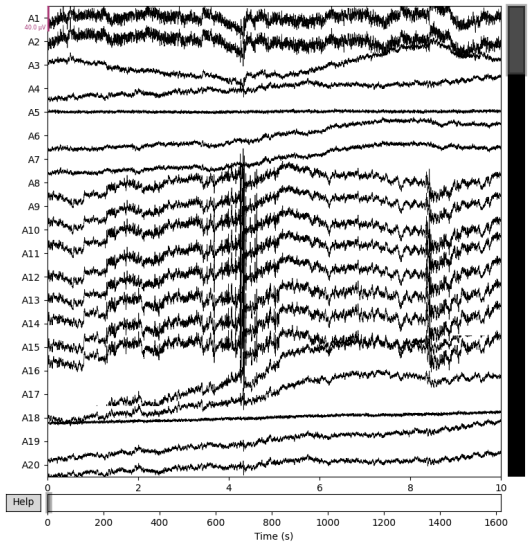
eeg_data-02.bdf → subject 2



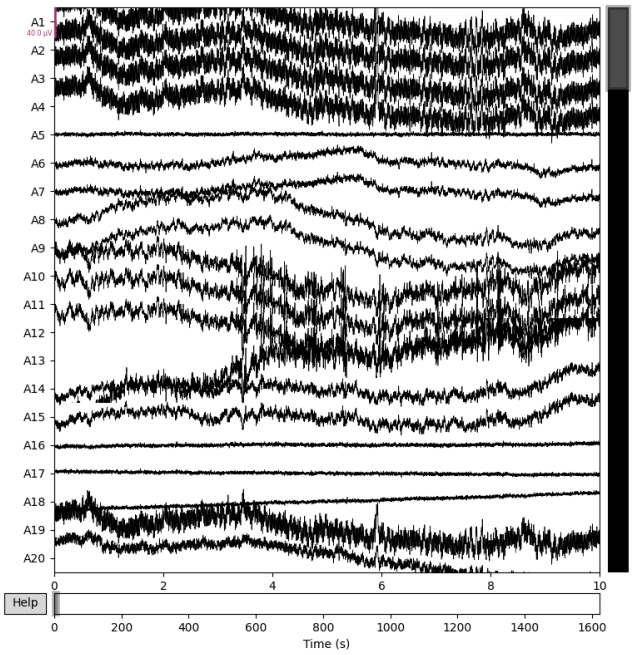
eeg_data-03.bdf → subject 3



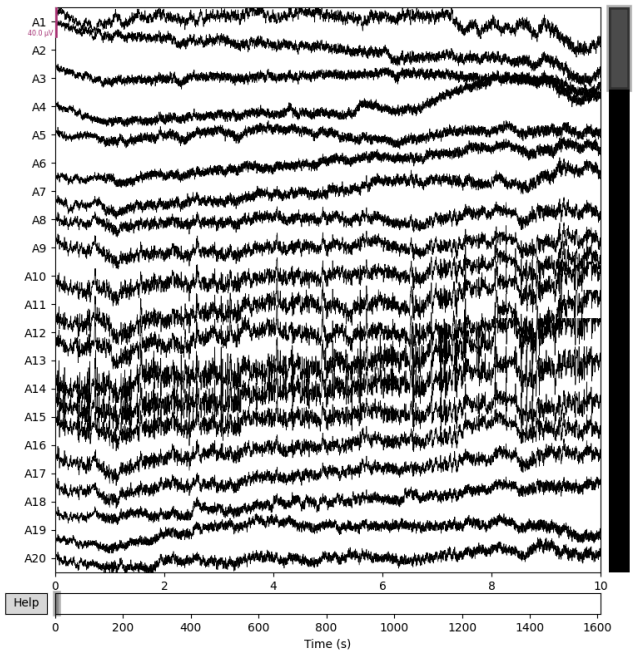
eeg_data-04.bdf → subject 4



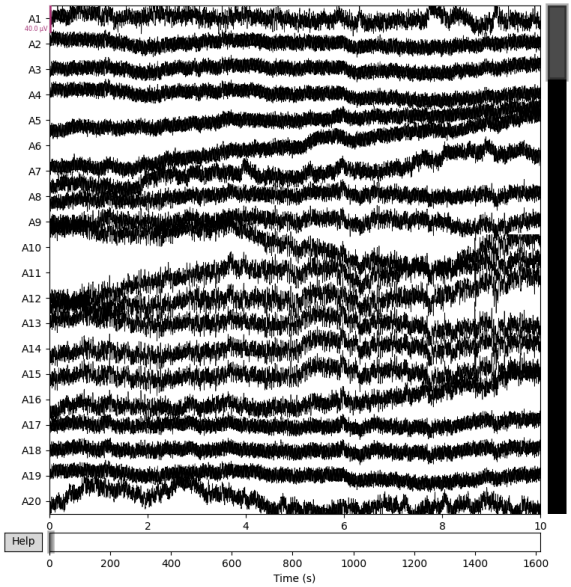
eeg_data-05.bdf → subject 5



eeg_data-06.bdf → subject 6



eeg_data-07..bdf → subject 7



Signal Processing

1. Re-referencing: The EEG data was re-referenced using two external electrodes (EXG1 and EXG2) to reduce external noise and improve the signal-to-noise ratio.
2. Bandpass and Notch Filtering: A bandpass filter (8 to 30 Hz) was applied to isolate the frequency range of interest. Additionally, a notch filter at 50 Hz was used to remove power line noise.
3. Resampling: The data was decimated to reduce the sampling rate to around 254 Hz, which is sufficient for capturing EEG dynamics while reducing computational load.
4. Epoching: The continuous EEG data was segmented into epochs based on specific event markers (e.g., Up, Down, Left, Right), relevant to the inner speech tasks. This step was crucial for isolating the brain signals associated with specific mental activities.

```
import mne

# Function to process each dataset
def process_dataset(file_path):
    # Read BDF File
    raw = mne.io.read_raw_bdf(file_path, preload=True)
    # Re-reference; NODE CHANGES HERE: 'EXG1', 'EXG2',
    # EXG3', 'EXG4', 'EXG5', 'EXG6', 'EXG7', 'EXG8'
    raw.set_eeg_reference(['EXG1', 'EXG2'])
    # Apply bandpass filter (8 to 30 Hz) and Notch
    # filter at 50 Hz
    raw.filter(l_freq=8, h_freq=30, fir_design='firwin')
    raw = mne.io.Raw.notch_filter(raw, freqs=50)
    # Decimate the data to reduce sampling rate to around 254 Hz
    raw.resample(254)
    # Epoching: Define events and epochs based on experimental design
    events = mne.find_events(raw, initial_event=True,
                             consecutive=True, min_duration=0.002)
    event_id = dict(Down=32, Left=33)
    picks_vir = mne.pick_types(raw.info, eeg=True, include=['EXG1',
                                                            'EXG2'], stim=False)
    epochs = mne.Epochs(raw, events, event_id=event_id, tmin=-0.5,
                        tmax=4, picks=picks_vir, preload=True, decim = 3, baseline=None)
    print("*************************************************************************")
    return epochs

# Download and process each dataset
epochs_02 = process_dataset('eeg_data_02.bdf')
epochs_03 = process_dataset('eeg_data_03.bdf')
epochs_04 = process_dataset('eeg_data_04.bdf')
epochs_05 = process_dataset('eeg_data_05.bdf')
epochs_06 = process_dataset('eeg_data_06.bdf')
epochs_07 = process_dataset('eeg_data_07.bdf')

# Combine epochs from all datasets
all_epochs = mne.concatenate_epochs([epochs_02, epochs_03, epochs_04,
                                     epochs_05, epochs_06, epochs_07])
```

Feature Extraction

Power Spectral Density (PSD) was computed for each epoch using the Welch method. This step transformed the time-domain EEG signals into the frequency domain, facilitating the identification of frequency-specific patterns associated with inner speech. Mean PSD values across frequencies for each channel were computed to serve as features for the subsequent machine learning models.

```
# Feature Extraction

import numpy as np

# Compute PSD using the compute_psd method
spectrum = all_epochs.compute_psd(method='welch')
data, freqs = spectrum.get_data(return_freqs=True)

# Skip conversion to dB, use the power values directly
psds_power = data

# Mean PSD values across frequencies for each channel
features = psds_power.mean(axis=1)
```

Data Normalization

The features were standardized using a StandardScaler from the sklearn library. This step ensured that all features contributed equally to the models, preventing features with larger magnitudes from dominating the learning process.

```
# Data Normalization

from sklearn.preprocessing import StandardScaler

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)
```

Dimensionality Reduction

Principal Component Analysis (PCA) was applied to reduce the dimensionality of the feature set while retaining 95% of the variance. This reduction was essential to simplify the models and to focus on the most informative aspects of the EEG signals.

```
# Dimensionality Reduction

from sklearn.decomposition import PCA

# Apply PCA for dimensionality reduction
pca = PCA(n_components=0.95)  # Keep 95% of variance
X_pca = pca.fit_transform(X_scaled)
```

Model Building

Two machine learning models were employed: Linear Discriminant Analysis (LDA) and Random Forest (RF). Each model aimed to classify the epochs into categories corresponding to different inner speech tasks.

Linear Discriminant Analysis (LDA)

1. The dataset was split into training and test sets.
2. The LDA model was trained on the training set with class priors adjusted to handle potential class imbalances.
3. The model's performance was evaluated on the test set using metrics like accuracy, confusion matrix, and classification report.

```
# LDA Model Building
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import train_test_split

# Preparing labels for classification
y = all_epochs.events[:, -1]  # Assuming last column of events array contains the labels

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.3, random_state=42)

# Class Splits
class_priors = [0.5, 0.5, 0.5, 0.5];

# Create LDA model
lda = LinearDiscriminantAnalysis(solver='lsqr', shrinkage='auto', priors=class_priors)
#try solver='eigen'
lda.fit(X_train, y_train)
```

Random Forest (RF)

1. The Random Forest model was built and trained similarly to the LDA model.
2. A grid search with cross-validation was performed to fine-tune the hyperparameters of the Random Forest model (e.g., number of estimators, max depth, and min samples split).
3. The best-performing model was then evaluated on the test set.

```
# Random Forest Model Building
from sklearn.ensemble import RandomForestClassifier

# Preparing labels for classification
# Assuming last column of events array contains the labels
y = all_epochs.events[:, -1]

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_pca, y,
                                                    test_size=0.3,
                                                    random_state=42)

# Create RF Model
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

Model Validation and Evaluation

Both models (LDA and RF) were rigorously evaluated on the test set. The evaluation metrics included accuracy, confusion matrix, and a detailed classification report. These metrics provided insights into the models' performance, specifically their ability to accurately classify different types of inner speech-related brain signals.

```
# Model Validation and Evaluation

from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, classification_report,

# Predict on test data
y_pred_lda = lda.predict(X_test)

# Evaluate the LDA model
accuracy_lda = accuracy_score(y_test, y_pred_lda)
conf_matrix_lda = confusion_matrix(y_test, y_pred_lda)
class_report_lda = classification_report(y_test, y_pred_lda)
```

Model Building: Rationale for Selection of Linear Discriminant Analysis (LDA) and Random Forest (RF) for EEG Data Analysis

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis is a widely used technique for dimensionality reduction and classification, particularly suited for EEG data due to several reasons:

1. Class Separability: LDA aims to find a linear combination of features that best separates two or more classes. This is particularly beneficial for EEG data, which often involves distinct brain signal patterns corresponding to different cognitive states or tasks.

2. Simplicity and Efficiency: LDA is computationally less intensive compared to other complex models. This simplicity makes it faster to train and more straightforward to interpret, an important consideration given the high-dimensional nature of EEG data.

3. Handling Multiclass Problems: EEG-based tasks often involve multiple categories (e.g., different types of inner speech). LDA can effectively handle multiclass classification problems, making it a suitable choice for such scenarios.

4. Performance with Small Sample Sizes: In situations where the amount of data is limited, LDA can still perform well, assuming the number of samples is greater than the number of features. This is often the case in EEG datasets after dimensionality reduction.

Random Forest (RF)

Random Forest is a robust ensemble learning method that is highly effective for EEG data analysis due to several characteristics:

1. Handling Non-Linearity: Unlike LDA, Random Forest can capture nonlinear relationships between features, which is crucial in EEG data where the relationship between brain signals and cognitive states may not be linear.

2. Feature Importance: RF provides insights into feature importance, allowing researchers to identify which EEG channels or frequency bands are most predictive of the inner speech tasks. This is particularly useful in neuroscientific studies for understanding brain function.

3. Robustness to Overfitting: RF, being an ensemble of decision trees, is less prone to overfitting, especially with a large number of trees. This robustness is important in EEG analysis where the model needs to generalize well to new data.

4. Handling High Dimensionality and Complex Interactions: RF can handle high-dimensional data and is effective in capturing complex interactions between features, which are common in EEG data.

5. Flexibility and Customization: The ability to tune various hyperparameters (like the number of trees, depth of trees, etc.) in RF allows for customization according to the specificities of the EEG dataset, enhancing model performance.

The combination of LDA and RF in the analysis provides a comprehensive approach - LDA for its efficiency and simplicity in linear separability, and RF for its robustness and ability to capture complex, non-linear patterns in EEG data. This dual-model approach ensures a more thorough analysis, catering to both linear and non-linear aspects inherent in EEG signals associated with inner speech.

In-Depth Analysis of Parameters in LDA and RF Models for EEG Data

Linear Discriminant Analysis (LDA) Parameters

1. Solver: 'lsqr' and 'eigen'
   - 'lsqr': This solver is used for its robustness and efficiency, especially in high-dimensional spaces. It's particularly suitable for datasets where the number of features exceeds the number of samples.
   - 'eigen': This solver is effective when dealing with covariance matrices that are not full rank, which can be a case in EEG data due to multicollinearity among EEG channels.

2. Shrinkage: 'auto'
   - Shrinkage: This parameter is used to regularize the covariance matrix, especially important in scenarios where the number of features is high compared to the number of samples. In EEG data, where signals might be highly correlated, shrinkage helps to improve the model's performance by reducing overfitting.

3. Class Priors: [0.5, 0.5, 0.5, 0.5]

- These priors are set assuming equal probability for each class. In the context of EEG data, where class distributions might be uneven (e.g., different amounts of data for different types of inner speech), setting equal priors can help balance the classifier's sensitivity towards each class.

Random Forest (RF) Parameters

1. Number of Estimators: 100

- This represents the number of trees in the forest. A higher number of trees increases the model's robustness and accuracy, but also computational cost. In EEG data analysis, a moderate number like 100 strikes a balance between accuracy and efficiency.

2. Random State: 42

- Setting a random state ensures reproducibility of the results. Consistency is crucial in scientific experiments, including EEG analyses, to ensure that results are not due to random chance.

3. Grid Search Parameters:

- n_estimators: [50, 100, 200]

- Varying the number of trees helps to identify the optimal complexity of the model. More trees can lead to better performance but increase computation time.

- max_depth: [None, 10, 20, 30]

- Controls the maximum depth of each tree. Deeper trees can capture more complex patterns but risk overfitting, especially in EEG data where irrelevant or noisy features might be learned.

- min_samples_split: [2, 5, 10]

- This parameter determines the minimum number of samples required to split an internal node. Adjusting this can prevent trees from becoming too deep and overfitting, especially important in EEG data which can be noisy.

4. Cross-Validation: 5-fold

- Using cross-validation helps in assessing the model's generalization ability. In EEG data analysis, where the goal is to create models that generalize well across different subjects and sessions, cross-validation is vital.

The selection of these parameters for both LDA and RF models is a result of a balance between model complexity, computational efficiency, and the unique characteristics of EEG data, such as high dimensionality, potential non-linearity, and the need for generalization across different brain patterns. Fine-tuning these parameters ensures the models are well-suited for capturing the intricate patterns associated with inner speech in EEG data.

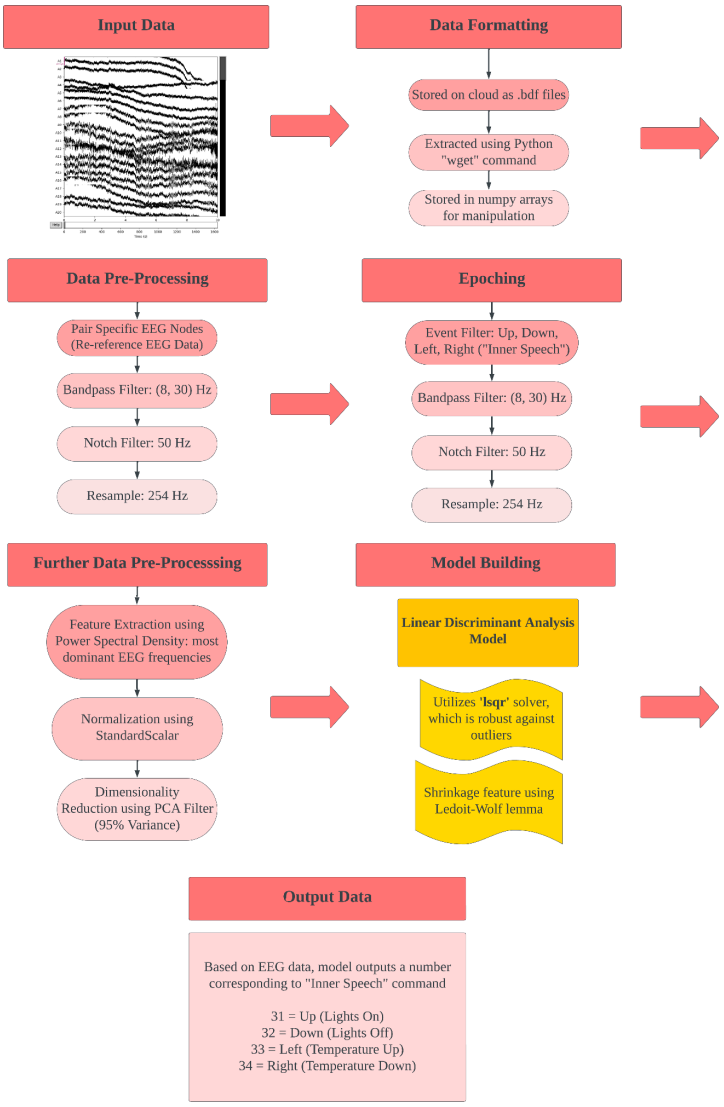
```
from sklearn.model_selection import GridSearchCV

# Parameters to tune
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10]
}

# Grid Search with cross-validation
rf_grid = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5)
rf_grid.fit(X_train, y_train)

# Use the best estimator for predictions
y_pred_rf = rf_grid.best_estimator_.predict(X_test)
```

Summarized Algorithm Pipeline:

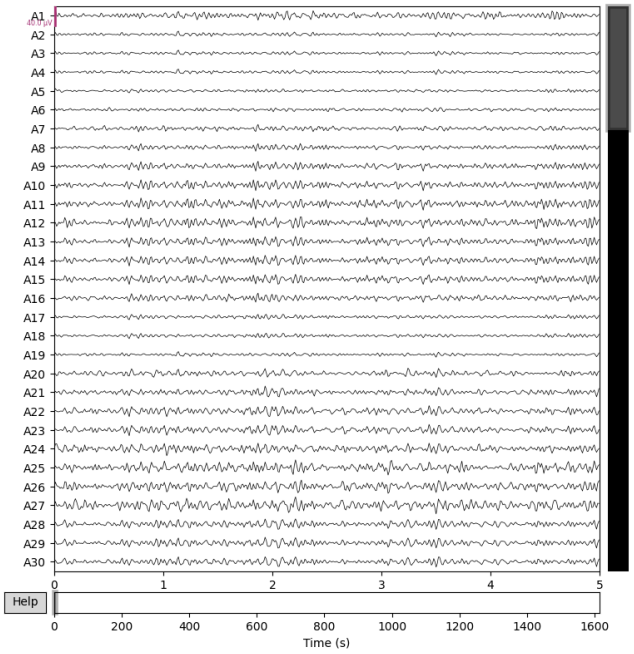
Dataset obtained from the IMAL Research Institute in Argentia collected patient EEG data for various brain commands/tasks.

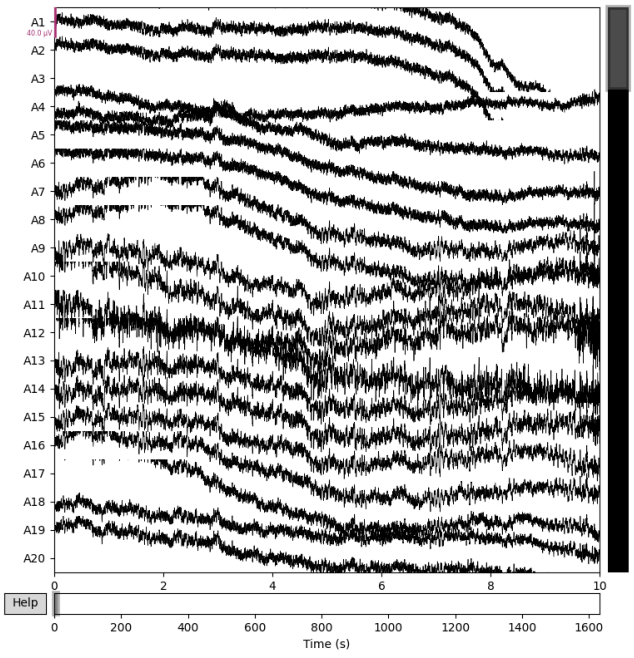
This algorithm creates filters through data and creates a machine learning model that can detect brain waves associated with internal thoughts (inner speech).
Many forms of data filtration, feature extraction, and reduction were employed to convert the data into its most readable and efficient form.

Linear Discriminant Analysis was selected because it specializes in feature reduction and data extrapolation, key components in working with EEG data. Other models, such as CNNs, RFs, and SVMs were tested, although they yielded poor results.

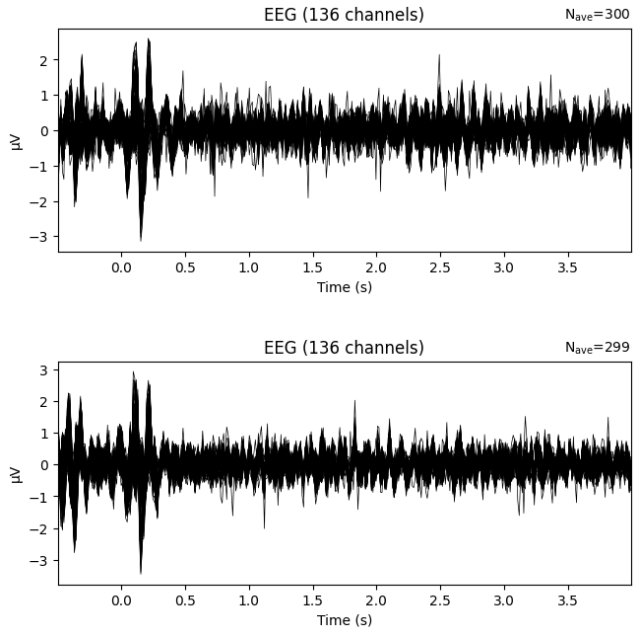


PHASE 2 Continued: Algorithm Analysis

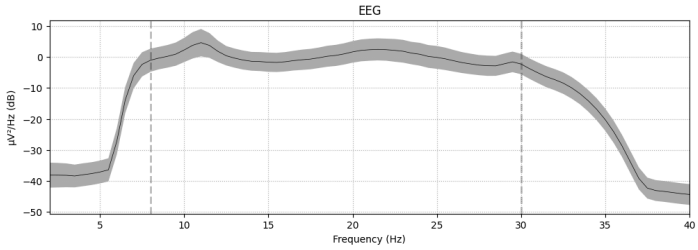




Filtered Data (top) vs. Unfiltered Data (bottom): Data filtration removes a large degree of noise from incoming brain wave data. Cleaner waves permit more accurate analysis of wave crests and troughs. This, in turn, allows us to compare the first and second order derivatives of brain waves to a much larger extent.



Average brain wave for “lights on” command (top) vs. Average brain wave for “lights off” command (bottom): Although the difference in each command’s brain wave is quite small, there is enough variance for the LDA model to capture. Moreover, it would be expected that such commands are largely similar, as the commands themself are also similar.



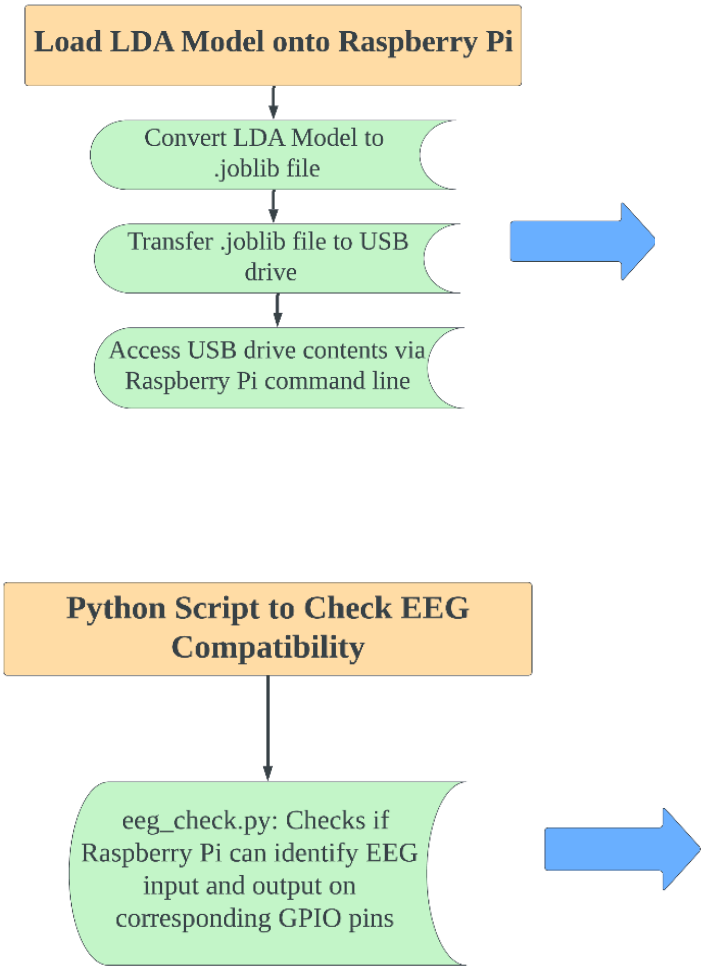
Power Spectral Density uses first and second order derivatives of the frequency of the brain waves to construct a range (depicted in shaded grey) for which subsects of waves may fall in to be accurately predicted.

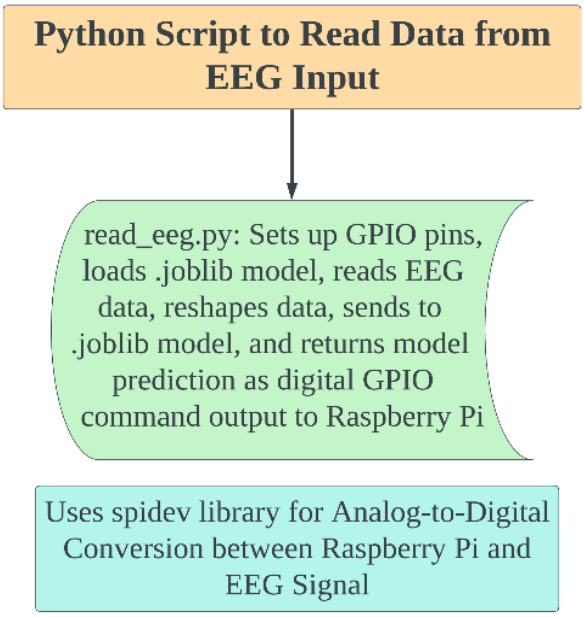
PHASE 3: Output & Utilization

By integrating the machine learning model from Phase 2 into the Raspberry Pi computer (Raspberry Pi GPIO pins will be used for export), binary classification can be sent directly to connected remotes via Male/Female wires to control lights and air conditioning. The commands the user must think are “Up”, “Down”, “Right”, and “Left” (Up=Lights on, Down=Lights off, Right=Temperature +1, Left=Temperature -1). The brain-computer interface is coded to discern between intentional and unintentional thoughts by differentiating EEG graph gradients via Linear Discriminant Analysis and derivation.

The Event-Related Potential for intentional thoughts has a shorter period, resulting in larger derivative and smaller second derivative values approaching the peak.

Phase 3 focuses on combining the elements from phase 1 and phase 2, in order to create a functioning pipeline that can read incoming brain wave data and filter through such data to accurately turn on/off a light (and other home appliances) when the subject thinks of the appropriate command using internal thoughts.





The above flow chart demonstrates how the LDA model was linked to the Raspberry Pi and various hardware components.

The eeg_check.py file was written to make sure the EEG has been detected by the Raspberry Pi.

The output from the read_eeg.py file returns either the number 31 or 32. The entirety of the 'predict' function is stored within a command that calls to the GPIO pins of the Raspberry Pi. If the function returns 31, GPIO pin 31 is signaled, and the light turns on. If the function returns 32, GPIO pin 32 is signaled, and the light turns off.

Video Demonstration:

https://drive.google.com/file/d/1fQFq9-apAhN5-mCExt-DYWQ3ujR6EaNY/view?usp=drive_link

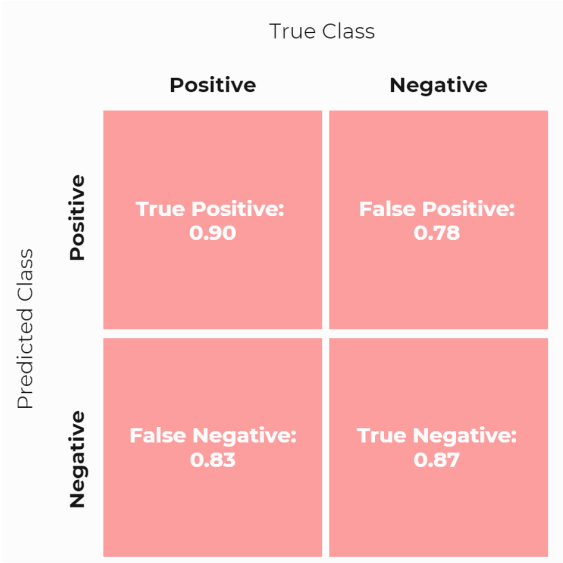
PHASE 4: Output & Utilization

INNER SPEECH COMMANDS	LINEAR DISCRIMINANT ANALYSIS MODEL
LIGHTS ON VS. LIGHTS OFF	0.87

The table above indicates that the LDA model is able to identify the correct command between "lights on" and "lights off" 87% of the time.

While this accuracy is fairly high, transporting the LDA model as a .joblib file to the Raspberry Pi reduces its accuracy by roughly 10%.

Still, this indicates that the LDA model can accurately filter through inner thought commands every 3 out of 4 times.



The confusion matrix above reveals that the current model has lower false positive and false negative rates compared to true cases. This is likely due to rare edge cases that affect the overshooting of the LDA model. Exploration of how to improve these rates is explored in the "Future Improvements."

McNemar's Test for Linear Discriminant Analysis Algorithm

This Study's Model

Current Models		Correct Output	Incorrect Output	Total
	Correct Output	55	4	5
	Incorrect Output	37	4	4
	Total	92	8	10

p = 0.004677735 | χ2 = 8

Cochran's Q test

n = 2	df = 1
Test Statistic Q = 403.61	p-value = 3.22E-5

Statistics

VAR	Sum	Proportions: 0	Proportions: 1
α	12	66.66%	33.33
β	10	25.00%	75.00
γ	8	20.00%	80.00
δ	18	16.66%	83.33

$$\text{mid-p-value} = 2\left(\sum_{i=b}^{n}\binom{n}{i}0.5^{i}(1-0.5)^{n-i} - 0.5\binom{n}{b}0.5^{b}(1-0.5)^{n-b}\right)$$

h0: No difference between the validation accuracy of current models and this study's model.

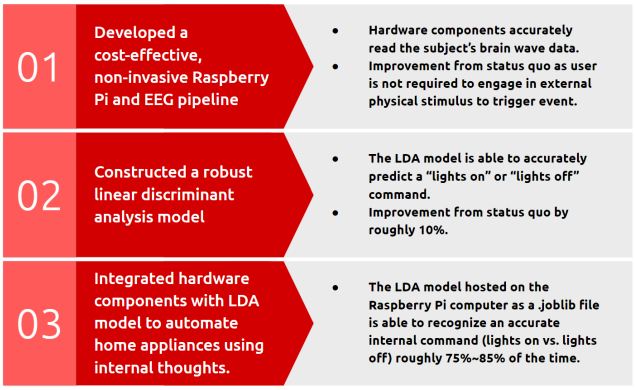
ha: Some difference between the validation accuracy of current models and this study's model.

P-values for the the models in phase 2 are shown above in the McNemar's test and Cochran's Q test.

Chi-squared critical values are large, indicating variation was statistically significant.

CONCLUSIONS

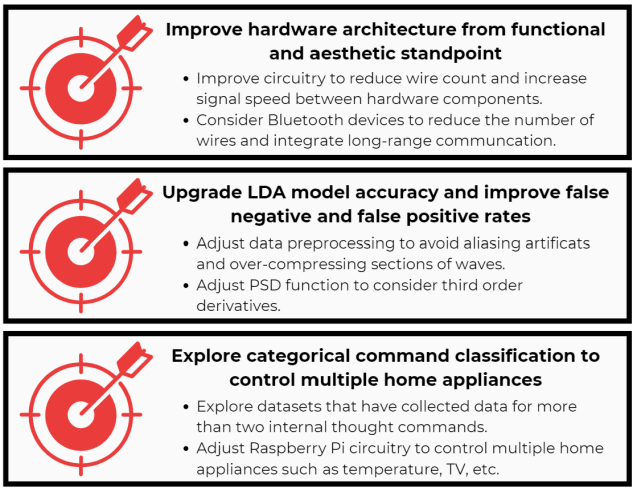
The participants performed real-time sessions to select commands related to home-automation tasks such as Light ON or Music ON simply by looking at a target command word from a display of 12 words on the user interface. The trained classifier identifies the user's intended command. The system then provides visual and auditory feedback of the identified command to the user. The selected command is then wirelessly sent to a Raspberry Pi device using network communication. The Raspberry Pi translates the received command to appropriate response and controls the respective home device. The system prototype is evaluated with various home device control commands in two different scenarios: providing random control commands and by participants' choice commands. The system is able to control home devices like light, music, television, and air-conditioner. The system was able to predict 97.92% of the commands correctly and execute the device control successfully with the first participant and over 95% with the second and third participants. People suffering from neuromuscular disorders have limited or no communication capacity. The proposed system would be very helpful and easy to use for the patients suffering from debilitating neurological disorders and patients with paralysis, allowing them to execute daily tasks with ease and improving their quality of life.



FUTURE RESEARCH

The following are the areas of interest in the future research:

- Longer-term, this system will be able to account for Bluetooth and hands-free signaling, thereby eliminating the need for fully connected circuits. However, this implementation involves a larger focus on hardware-specific initiatives, deviating from our engineering goal, which focuses on using an LDA model to interpret EEG brain waves. This allows for a more wide ranging grasp of appliance control. Since the purpose of this research is to interpret EEG data, the appliance endpoint serves more of a proof of concept rather than an end goal for this research.
- Another future step is to include a larger variety of commands, as well as commands that go beyond binary classification. The difficulty of introducing commands beyond binary classification is the minute difference in brain wave output on a linear scale, since there could be very minor activity differences between numbers of approximately similar values.
- Another future step is to go beyond beta waves signals, which are conscious thought processes, to more intricate brain signals (i.e. P200, P150). This allows sleep states and subconscious states of minds to be tapped into, allowing for more nuanced readings of subconscious desires as well as optimizing preventative approaches in healthcare.
- Inclusion of additional characteristics of the brain EEG data other than beta waves ERP such as eye movement activity and facial emotions to improve the accuracy.
- Add more home appliances to the system to control through the brain-computer interface.



RESULTS AND DISCUSSION

The engineering goal of the project was to develop a home automation system which is controlled only by brain activity to support people with neuromuscular disorders. A low cost EEG device, Emotiv EPOC+ headset with 14 channels is used to record EEG data while the participants performed real-time sessions to select home device control commands simply by

looking at a target command word from a display of 12 words on the user interface. OpenViBE software is used to design the application scenarios for real-time and off-line analysis of EEG data. A Python interface is used to stream the raw EEG data from the Emotiv EPOC+ headset to OpenViBE. The EEG data is preprocessed and the features extracted for the target and non-target events are used to train the classifier, Linear Discriminant Analysis (LDA). The performance is calculated according to the number of samples correctly classified into the target (beta waves) and non-target (non-beta waves) commands with the classifier. The classifier was able to predict more than 95% of the commands correctly.

The participants performed real-time sessions to select commands related to home-automation tasks such as Light ON or Music ON simply by looking at a target command word from a display of 12 words on the user interface. The trained classifier identifies the user's intended command. The system then provides visual and auditory feedback of the identified command to the user. The selected command is then wirelessly sent to a Raspberry Pi device using network communication. The Raspberry Pi translates the received command to appropriate response and controls the respective home device. The system prototype is evaluated with various home device control commands in two different scenarios: providing random control commands and by participants' choice commands. The system is able to control home devices like light, music, television, and air-conditioner. The system was able to predict 97.92% of the commands correctly and execute the device control successfully with the first participant and over 95% with the second and third participants.

Error Analysis:

Systemic:

- Limited dataset, and number of patients for testing.
- Possible incorrect circuitry that may be over- or under-amplifying the signal transmitted between the Raspberry Pi, EEG, and light bulb.

Random:

- Random variances in provided dataset (aliasing artifacts, incorrect baselines, etc.).
- Wave distributions vary from patient to patient (some calibration is necessary to overcome).

Confounding Variables:

- Possible data discrepancies between trained LDA model and actual data collected by BioAmp EEG pill: LDA model trained on data using more advanced EEGs.

REFERENCES

1. Paralysis statistics. Reeve Foundation. Retrieved from https://www.christopherreeve.org/living-with-paralysis/stats-about-paralysis.

2. Haider, A., & Fazel-Rezai, R. (2017, November 29). Application of beta waves Event-Related Potential in Brain-Computer Interface. Retrieved from https://www.intechopen.com/books/event-related-potentials-and-evoked-potentials/application-of-beta waves-event-related-potential-in-brain-computer-interface

3. Wilson, M. J., Harkrider, A. W., & King, K. A. (2012). Effects of complexity of visual distracters on attention and information processing speed reflected in auditory beta waves. Ear & Hearing, 33(4), 480–488. https://doi.org/10.1097/aud.0b013e3182446a42

4. Hagen, G., Gatherwright, J., Lopez, B., & Polich, J. (2006). P3A from visual stimuli: Task difficulty effects. International Journal of Psychophysiology, 59(1), 8–14. https://doi.org/10.1016/j.ijpsycho.2005.08.003

5. Helal, S., & Chen, C. (2009). The Gator Tech Smart House. Proceedings of the 3rd International Convention on Rehabilitation Engineering & Assistive Technology - ICREATE '09. https://doi.org/10.1145/1592700.1592715

6. Alrajhi, W., Hosny, M., Al-Wabil, A., & Alabdulkarim, A. (2014). Human factors in the design of BCI-controlled wheelchairs. Human-Computer Interaction. Advanced Interaction Modalities and Techniques, 513–522. https://doi.org/10.1007/978-3-319-07230-2_49

7. “EPOC+ User Manual.” EMOTIV Applications, EMOTIV Inc., 2019, emotiv.gitbook.io/epoc-user-manual/.

8. Renard, Yann. OpenViBE: An Open-Source Software Platform to Design, Test, and Use Brain–Computer Interfaces in Real and Virtual Environments. www.cs.ucf.edu/courses/cap6121/spr16/readings/OpenViBEPaper.pdf.

9. Rivet, B., Souloumiac, A., Attina, V., & Gibert, G. (2009). XDAWN Algorithm to Enhance Evoked Potentials: Application to Brain–Computer Interface. IEEE Transactions on Biomedical Engineering, 56(8), 2035-2043. doi:10.1109/tbme.2009.2012869

10. Raspberry pi documentation. Teach, Learn, and Make with Raspberry Pi. https://www.raspberrypi.org/documentation/. Raspberry Pi reference

11. Abbott, D. (2012). Linux for embedded and real-time applications. Elsevier Science

12. Brownlee, J. (2019, August 8). Confidence Intervals for Machine Learning. Retrieved from https://machinelearningmastery.com/confidence-intervals-for-machine-learning