IoT-Enabled EEG System for the Detection of Real-Time Psychological and Neurological State and Data Imputation using Machine Learning

Anshuman Sahoo
Roll Number: 122CS0102

Department of Computer Science and Engineering
National Institute of Technology Rourkela

November, 2025

Supervisor: Prof. Pabitra Mohan Khilar

Abstract

This study details the design and creation of a complete IoT framework for monitoring mental states in real-time with EEG. We introduce a new data acquisition module that employs an ESP32 to convert analog signals from an affordable 3-electrode EEG device. A Python-based processing pipeline, using bandpass filtering and ICA, extracts features from the standard brainwave bands (α, β, γ, δ, θ) to categorize conditions like High Stress, Focus, and Relaxation. The system's architecture is designed for eventual integration with an Android application, and future work will incorporate advanced data imputation to address signal loss.

Background and Motivation

Current challenges motivating this project:

- **Subjectivity:** Heavy reliance on self-reporting and clinician interpretation.
- **Late Diagnosis:** Conditions identified after significant impairment.
- **Accessibility:** Limited availability of specialists in remote areas.
- **Stigma:** Social barriers reduce help-seeking.

The project aims to provide an objective, portable monitoring framework to help mitigate these issues.

EEG Brainwaves

Principal EEG frequency bands used as features:

Band	Frequency (Hz)	Associated mental states
Delta (δ)	0.5–4	Deep sleep; abnormal wakefulness may indicate injury
Theta (θ)	4–8	Drowsiness, deep relaxation, memory consolidation
Alpha (α)	8–13	Quiet wakeful relaxation; calm alertness
Beta (β)	13–30	Active thinking, focus; excessive beta $\rightarrow$ anxiety
Gamma (γ)	$>$30	High-level information processing

Project Objectives

- **Data Acquisition:** 3-electrode EEG headset with the ESP32's Analog-to-Digital Converter for wireless data transmission over BLE.
- **Signal Processing & Feature Engineering:** Bandpass filtering and ICA to eliminate artifacts before extracting key spectral and statistical features from the cleaned signal.
- **ML Model Development:** Train and compare classifiers to detect mental states.
- **System Design:** Android app for real-time feedback and on-device inference.

Scope & Limitations

- As a proof-of-concept, this work is limited to classifying discrete states (e.g., Focused, Relaxed, Stressed) and is not intended for clinical diagnosis.
- The dataset's collection in a structured laboratory setting means it may not fully capture the dynamics of everyday, mobile contexts.
- The collected dataset is not demographically diverse, requiring further validation on a broader population.
- The current version of the system does not include the android app for users, which is scheduled for future implementation.

Literature Review — EEG Processing Pipeline

Standard workflow informing the project:

1. **Preprocessing:** Bandpass filtering (0.5–45 Hz), remove power-line noise; ICA to remove EOG/EMG artifacts.
2. **Feature Extraction:** PSD in Delta/Theta/Alpha/Beta/Gamma; time-domain stats (mean, variance, skewness, kurtosis); entropy; alpha/beta ratio; physiological HR.
3. **Classification models:** SVM, LDA, Decision Trees, KNN, Naïve Bayes; deep models (CNN/RNN) possible but heavier.

Research Gap & Contribution

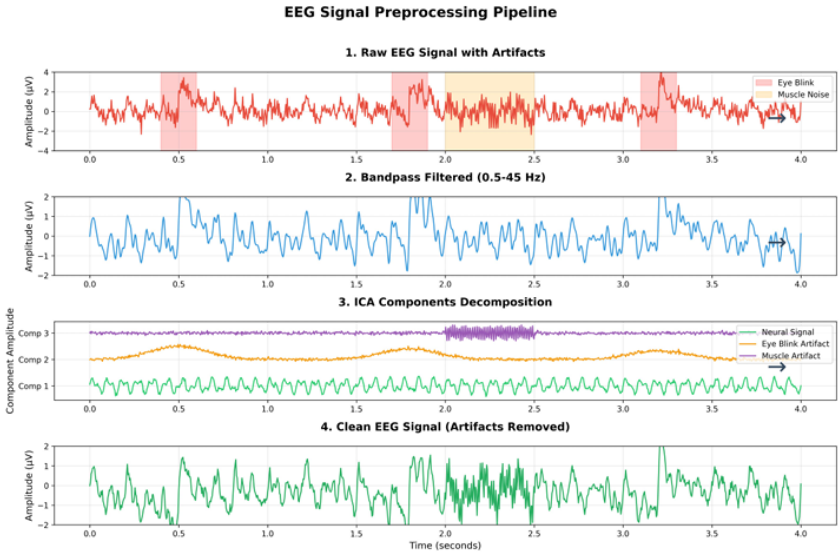
Identified gaps:

- Many high-performance models are too computationally demanding for real-time use on mobile applications.
- There is a shortage of complete, low-cost implementations that efficiently connect hardware to mobile platforms.

Contributions of this work:

- Novel low-cost ESP32-based digitizer for a 3-electrode headset.
- Efficient Python-based preprocessing and feature-engineering pipeline.
- Architecture designed for mobile deployment with on-device inference.

Signal Processing Pipeline



The Imputation Framework: FA-KNN

- **The Base Method: K-Nearest Neighbors (KNN) Imputer**
  - Estimates a missing value by finding the 'k' most similar, complete data points (neighbors) in the feature space.
  - The missing value is replaced by the mean of its 'k' neighbors.
- **How to find the optimal number for 'k'?**
  - A small 'k' (e.g., 1) can be noisy and overfit.
  - A large 'k' can oversmooth the data, washing out important details.
- **Solution is the Firefly Algorithm**
  - We use a metaheuristic, nature-inspired Firefly Algorithm (FA) to automatically optimize the 'k' hyperparameter.

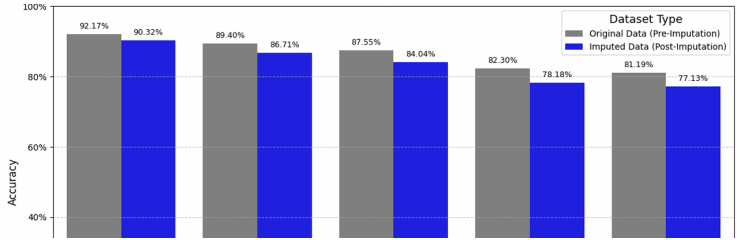
How the Firefly Algorithm Optimizes 'k'

- **Define Fireflies :** A population of "fireflies" is created. Each firefly represents a potential solution, in this case, a single integer value for 'k' (e.g., within the search space of 1 to 15).
- **Fitness Function (Brightness) :** A firefly's "brightness" is determined by the quality of the imputation it produces. A stable, well-chosen 'k' restores the dataset's information content, which is reflected in its variance. A poor 'k' will result in a lower-variance (oversmoothed) or unstable dataset.
- **Convergence :**The algorithm iteratively "moves" fireflies toward the brightest ones (the best 'k' values). The Firefly Algorithm converged on an optimal value of k=7. This imputer was then used to repair the full dataset for training.

1D-CNN Framework

- **The Data Pipeline**
  - **Input:** Raw 4-class EEG signals (Healthy, Mild, Moderate, Severe).
  - **Epoching:** The continuous signal is segmented into 2-second (500-sample) epochs.
  - **Overlapping:** A 50% overlap (250-sample step) is used to create a sliding window, ensuring no temporal information is lost at the edges.
  - **Final Input Shape:** (Batch Size, 500, 3)
- **Normalization :**Each of the 3 channels is standardized independently using a StandardScaler (fit only on training data) to preserve its unique characteristics.

Performance Metrics of the Models




```
--- Final Model Performance on Held-Out Test Set (15%) ---
                     Accuracy  Precision (Macro)  Recall (Macro)  F1
Model
Random Forest (Tuned)    0.9217             0.9188          0.9210
Logistic Regression      0.8940             0.8875          0.8940
KNN (k=5)                0.8755             0.8701          0.8750
Decision Tree            0.8230             0.8199          0.8230
Naive Bayes              0.8119             0.8090          0.8110

--- SCRIPT COMPLETE ---
```

Fig: Imputation Accuracies and Model Performance

Conclusion & Future Work

Conclusion

- Established a high-fidelity data acquisition chain: 3-electrode headset → PAMTRON → oscilloscope → ESP32 ADC.
- Robust pipeline for preprocessing and feature engineering was implemented in Python.
- This project developed high-accuracy (92.17%) Random Forest and 1D-CNN models for Alzheimer's classification, and created a novel Firefly Algorithm-optimized KNN imputer to robustly handle missing data.

Future Work

- To further improve upon the accuracies, more sophisticated deep learning models can be explored.
- An Android application will be developed to serve as the user interface. This app will be designed to connect to IoT-based EEG hardware via Bluetooth (BLE).

References

1. Lotte, F., Bougrain, L., Cichocki, A., Clerc, M., Congedo, M., Rakotomamonjy, A., & Yger, F. (2018). A review of classification algorithms for EEG-based brain-computer interfaces. Journal of Neural Engineering.
2. Insel, T. R. (2017). Digital phenotyping: technology for a new science of behavior. JAMA.
3. Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research.
4. Horton, J. (2019). Android Programming with Kotlin for Dummies. Wiley.
5. Subasi, A., Gursoy, M. I. (2010). EEG signal classification using PCA, ICA, LDA and support vector machines. Expert Systems with Applications
6. Kumar, V., Kumar, D. A Systematic Review on Firefly Algorithm: Past, Present, and Future. Arch Computat Methods Eng 28, 3269–3291 (2021). https://doi.org/10.1007/s11831-020-09498-y

Thank You