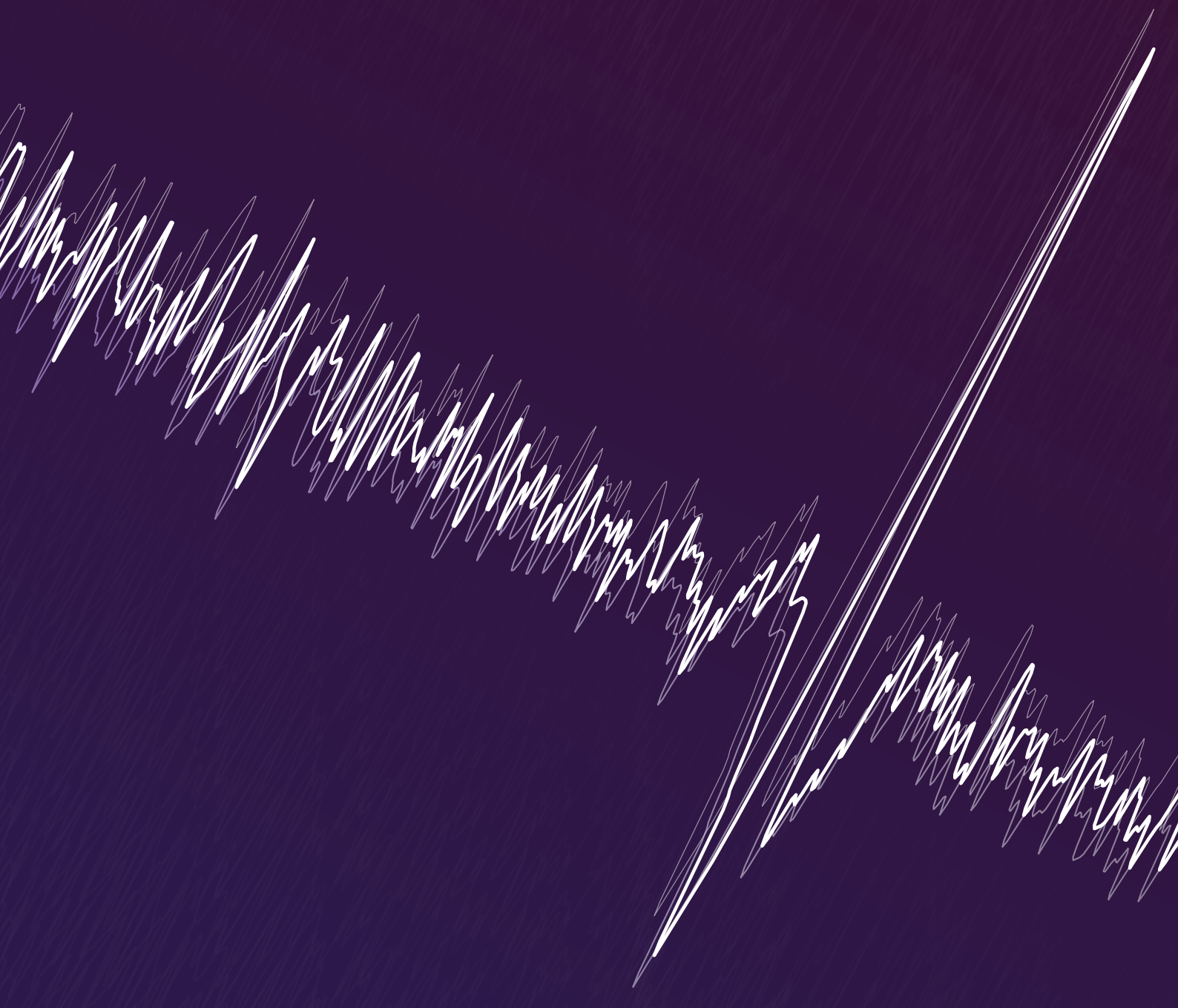
projekt zaliczeniowy

# Komunikacja Człowiek-Komputer

## Wariant „Speller”

Jakub Mańczak, Michał Kamieniak



# Spis treści



# Informacje porządkowe

Projekt został wykonany wspólnie przez Michała Kamieniaka i Jakuba Mańczaka, jako forma składowa zaliczenia laboratoriów z przedmiotu Komunikacja Człowiek-Komputer w roku akademickim 2025/2026.

Całość projektu jest dostępna na GitHubie: https://github.com/jakubmanczak/literkowo.

Do poprawnej rekreacji i wykonania kodu potrzebne będzie kilka modułów zależnych, wymienionych w pliku `pyproject.toml`, a także moduł `aseegg.py`.

Do pracy nad projektem użyto menedżera modułów `uv`.

## Program do wyświetlania i rejestrowania bodźców

Wykorzystując bibliotekę `pygame` utworzono prosty program do wyświetlania bodźców i rejestrowania o nich podstawowych informacji (wyświetlaną literę, czas początku i końca jej wyświetlania).

Owy program działa w pętli; mając w tle otwarty plik z zapisem bodźców, iteruje się przez alfabet łaciński, wyświetlając każdą literę przez jedną sekundę, po której upłynięciu nie wyświetla niczego przez następną sekundę.

Dane zapisane w wyniku działania programu znajdą się w pliku `letter_log.txt`. Każda linijka w pliku zawiera dane dotyczące jednego bodźca (litery), którego dane to kolejno: wyświetlana litera, moment rozpoczęcia wyświetlania litery, oraz moment zakończenia wyświetlania litery - dwie ostatnie w formacie UNIX Timestamp z dokładnością do 9 miejsc po przecinku. Dane są oddzielone znakiem tabu (`\t`).

Odstęp sekundowy został wybrany ze względu na komfort osoby badanej - tej kodującej wiadomość za pomocą mrugnięć. W ten sposób możliwe jest ignorowanie mrugnięć występujących podczas przerw, zdejmując z badanego wymóg całkowego wypierania się odruchu mrugania za wyjątkiem podczas wyświetlania pożądanych liter.

Program działa do momentu przerwania jego wykonywania za pomocą sygnału `SIGINT` (np. wciskując równocześnie Ctrl + C na klawiaturze) lub w inny sposób.


Rysunek 1: Program wyświetlający litery (`wyswietlacz_liter.py`)

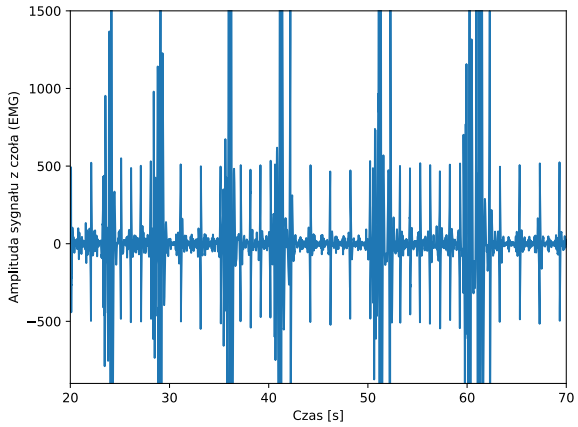
## Zbieranie danych

Dane zebrano 5 grudnia 2025 w sali 67 w budynku na Kampusie Ogrody.

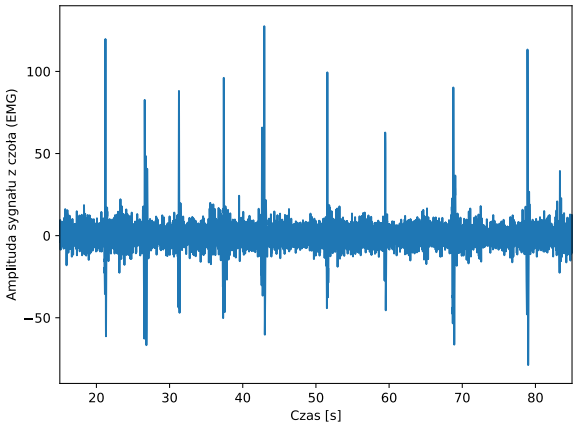
Użyto trzech elektrod - jedną przylegającą do czoła w celu odczytu aktywności mięśni czoła (ruch powiek), oraz dwie przylegające do uszu, służące eliminacji szumu.

Badany wybierając słowo zdecydował się na skrótowiec znanej serii Five Night's at Freddy's - - „FNAF", której film miał premierę tego samego dnia.

Podjęto dwie próby; jednak zgodnie z zaleceniem prowadzącego wybrano skupić się na tej drugiej, gdyż po wstępnej wizualizacji danych wyglądała bardziej obiecująco.




Rysunek 2: Wizualizacja próby nr. 1.



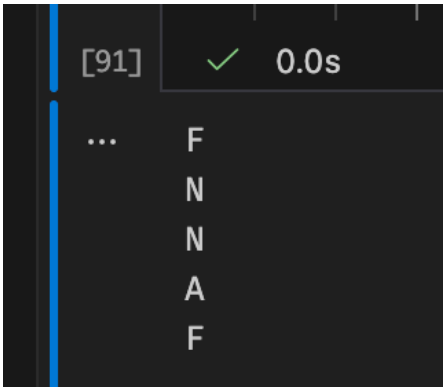

Rysunek 3: Wizualizacja próby nr. 2.

## Analiza główna

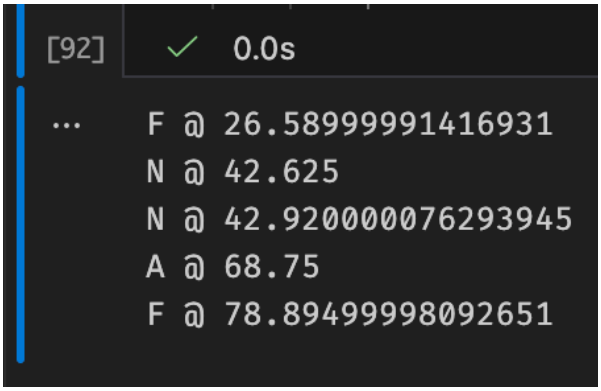
Do analizy mającej na celu odczyt przeliterowanego mrugnięciami słowa zdecydowano się zastosować poniższą strategię:

1. **Zebranie czasów pojedynczych mrugnięć do tablicy.**
   Ustalono wartość progową amplitudy sygnału EMG równą 50. Następnie sprawdzając każdy z zapisów amplitudy, jeżeli wartość przekroczyła próg dodano czas jej wystąpienia do tablicy. Aby uzyskać pojedynczą wartość na każde mrugnięcie zastosowano zmienną logiczną: jeżeli wartość przekroczyła próg, nie zbierano nowych wartości dopóki krzywa nie zeszła z powrotem poniżej progu.
2. **Porównanie z wartościami początków i końców wyświetlania liter.**
   Każdy z zapisów mrugnięć został porównany z całą tablicą wyświetlonych liter, oraz ich czasów granicznych. Jeżeli mrugnięcie wystąpiło podczas wyświetlania danej litery, litera ta została wyświetlona przez skrypt analizujący.

## Błąd analizy

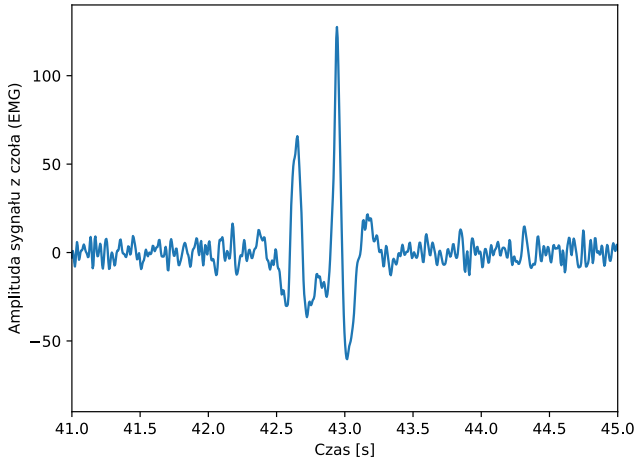



Rysunek 4: Fragment zrzutu ekranu z wynikiem działania wyżej opisanego skryptu.




Rysunek 5: Fragment zrzutu ekranu z wynikiem dalszej analizy błędu.

Mimo pojedynczego mrugnięcia osoby badanej skrypt wykrył dwa mrugnięcia dla litery „N”.
Po dodaniu do wyświetlanych liter ich czasów wystąpienia, poddano analizie fragment powodujący podwójne wystąpienie litery.




Rysunek 6: Wykres ilustrujący powód błędu w skrypcie: podwójne przekroczenie wyznaczonego progu.

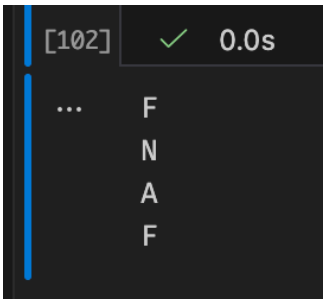
## Poprawa błędu

Aby zaradzić powyższemu błędowi oraz błędom mu podobnym, do skryptu analizującego wprowadzono odstęp czasowy między akceptacją czasów mrugnięć; jeżeli następne wykryte mrugnięcie ma miejsce mniej niż 0.3 sekundy po poprzednim, nie jest akceptowane.

Dzięki takiej zmianie wynik analizy pokrywał się z zamiarem osoby badanej.

## Podsumowanie

Projekt zakończył się sukcesem - w dwie próby udało się uzyskać wynik zgodny z założeniami i odczytać słowo przeliterowane przez osobę badaną.




Rysunek 7: Finalny, poprawny i zgodny
z zamiarem osoby badanej wynik.

### Refleksja nad rozwiązaniem błędu

Zastosowane w tym przypadku rozwiązanie - ignorowanie kolejnych przekroczeń progu przez krótki czas od poprzedniego przekroczenia - sprawdzi się dobrze w sytuacji, w której wiemy, że mrugnięcia będą sporadyczne i kontrolowane, a odstępy między bodźcami zniewlują możliwość pomyłki czy reakcji na zły bodziec. Nie sprawdzi się ono jednak w sytuacjach, gdy dwa bodźce występują jeden po drugim, a mrugnięcie znajdzie się na ich granicy; nie będzie bowiem wiadomo, które z wykrytych mrugnięć jest tym prawidłowym.

# Kod źródłowy

## Program do wyświetlania liter (wyswietlacz_liter.py)

Python

```python
#!/usr/bin/env python3
import time

import pygame

SCREEN_WIDTH = 1280
SCREEN_HEIGHT = 720
FONT_SIZE = 220
LETTER_INTERVAL = 1.0
BREAK_DURATION = 1.0
LOG_FILENAME = "letter_log.txt"
BG_COLOR = (0, 0, 0)
FG_COLOR = (255, 255, 255)

# fmt: off
letters = [
    "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K",
    "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V",
    "W", "X", "Y", "Z"
]

pygame.init()
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("SUPER FAJNY PROJEKT LITERKOWO")
clock = pygame.time.Clock()
FONT = pygame.font.Font(None, FONT_SIZE)

log_file = open(LOG_FILENAME, "a")

state = "show_letter"
next_switch = time.perf_counter()

index = 0
pending_letter = None
pending_start_unix = None

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
```

```
                running = False

    now_perf = time.perf_counter()
    if now_perf >= next_switch:
        if state == "show_letter":
            start_unix = time.time()
            letter = letters[index]

            screen.fill(BG_COLOR)
            surf = FONT.render(letter, True, FG_COLOR)
            rect = surf.get_rect(center=(SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2))
            screen.blit(surf, rect)
            pygame.display.flip()

            pending_letter = letter
            pending_start_unix = start_unix

            next_switch = now_perf + LETTER_INTERVAL
            state = "breaktime"

            index = (index + 1) % len(letters)

        elif state == "breaktime":
            end_unix = time.time()
            if pending_letter is not None:
                log_file.write(
                    f"{pending_letter}\t{pending_start_unix:.9f}\t{end_unix:.9f}\
                )
                log_file.flush()

            pending_letter = None
            pending_start_unix = None

            screen.fill(BG_COLOR)
            pygame.display.flip()

            next_switch = now_perf + BREAK_DURATION
            state = "show_letter"

    clock.tick(60)

if pending_letter is not None:
    exit_unix = time.time()
    log_file.write(f"{pending_letter}\t{pending_start_unix:.9f}\t{exit_unix:.9f}\
```

```
    log_file.flush()

log_file.close()
pygame.quit()
```

### Skrypt do analizy głównej (składowa projekt.ipynb)

Python

```
# SKRYPT DEKODUJĄCO-PORÓWNAWCZY WERSJA 2

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import aseegg as ag

df = pd.read_csv("proba2/ganglion_4ch_2025-12-05_10-58-15.csv")
przef = ag.gornoprzepustowy(df['ch1'], 200, 1)
przef = ag.pasmowozaporowy(przef, 200, 48, 52)
przef = ag.pasmowoprzepustowy(przef, 200, 3, 40)

mrugniecia = []
flaga = False
granica = 50
odstep = 0.3
for indeks, wartosc in enumerate(przef):
    if wartosc > 50 and flaga == False:
        flaga = True
        if len(mrugniecia) != 0:
            if df['time'][indeks] - mrugniecia[-1] >= odstep:
                mrugniecia.append(df['time'][indeks])
        else:
            mrugniecia.append(df['time'][indeks])
    if wartosc < 50:
        flaga = False

literki = pd.read_csv("proba2/letter_log.txt", names=['litera', 'czas_start', 'czas_stop'], sep="\t")

for mrug in mrugniecia:
    for kol, rzad in literki.iterrows():
        if mrug > rzad['czas_start'] and mrug < rzad['czas_stop']:
            print(rzad['litera'])
```