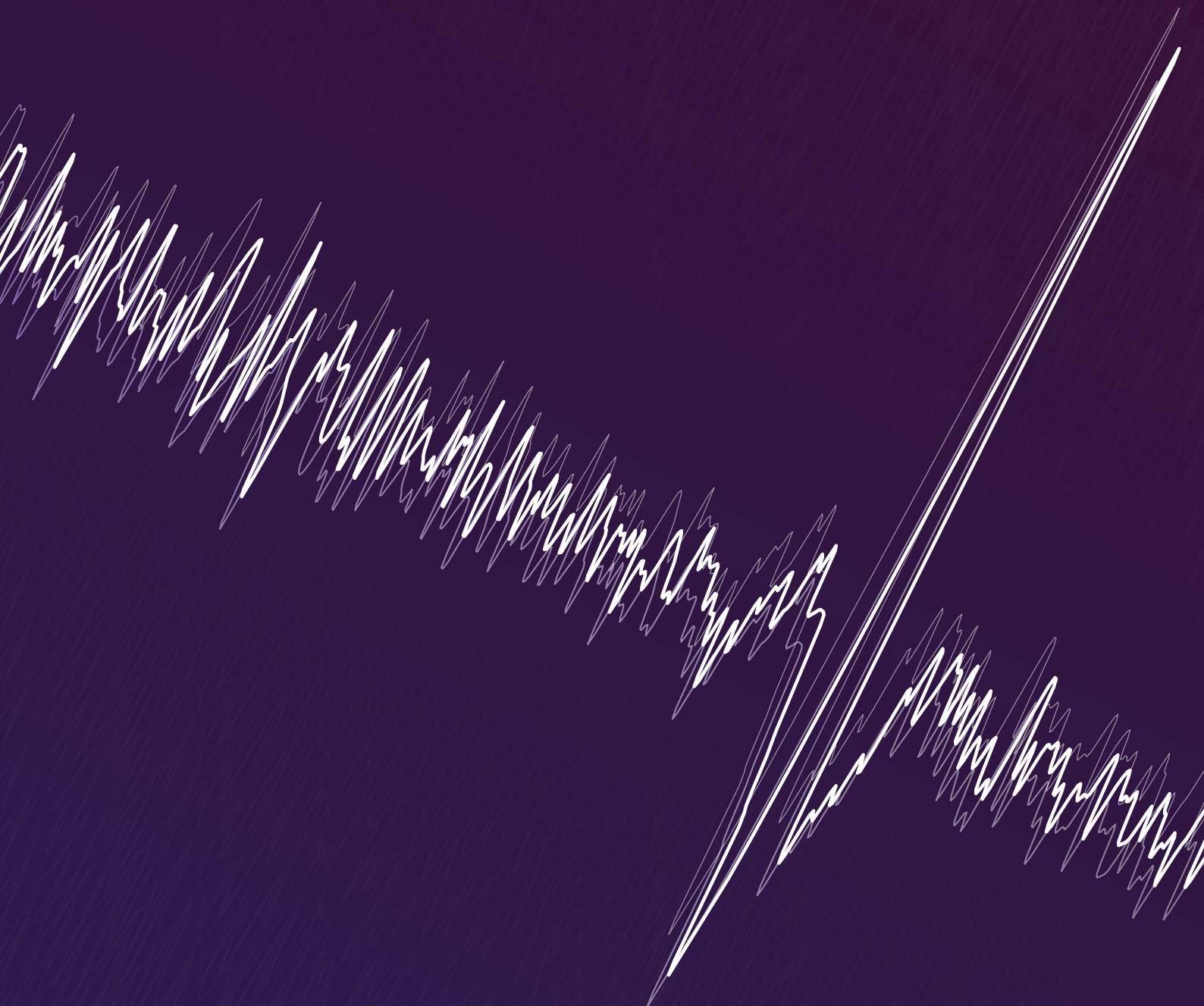


PROJEKT ZALICZENIOWY
KOMUNIKACJA CZŁOWIEK-KOMPUTER

WARIANT „SPELLER”

JAKUB MAŃCZAK, MICHAŁ KAMIENIAK



SPIS TREŚCI

Informacje porządkowe	2
Program do wyświetlania i rejestrowania bodźców	3
Zbieranie danych	4
Analiza główna	4
Błąd analizy	5
Poprawa błędu	5
Podsumowanie	6
Refleksja nad rozwiązaniem błędu	6
Kod źródłowy	7
Program do wyświetlania liter (wyswietlacz_liter.py)	7
Skrypt do analizy głównej (składowa projekt.ipynb)	9

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 `asegg.py`.

Do pracy nad projektem użyto menedżera modułów `uv`.

PROGRAM DO WYŚWIETLANIA I REJESTROWANIA BODŹCÓW

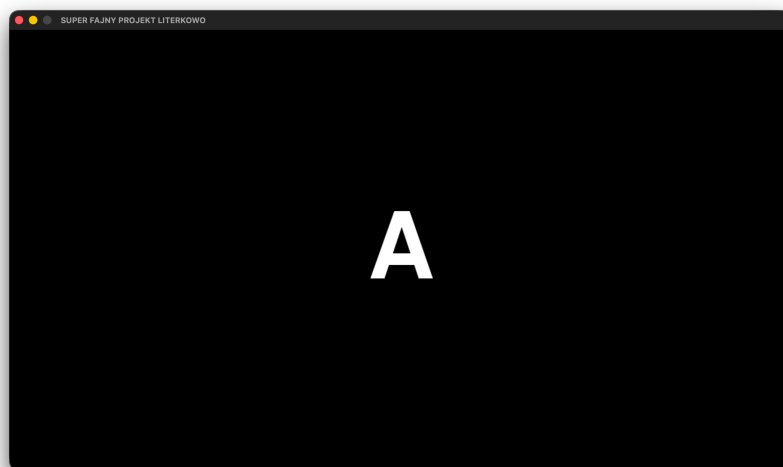
Wykorzystując bibliotekę pygame utworzono prosty program do wyświetlania bodźców i rejestrowania o nich podstawowych informacji (wyświetlaną literę, czas początku i końca jej wyświetlania).

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 upłygnięciu nie wyświetla niczego przez następną sekundę.

Dane zapisane w wyniku działania programu znajdują 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 sekundy 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 całkowitego wypierania się odruchu mrugania za wyjątkiem podczas wyświetlania pożądaných 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`)

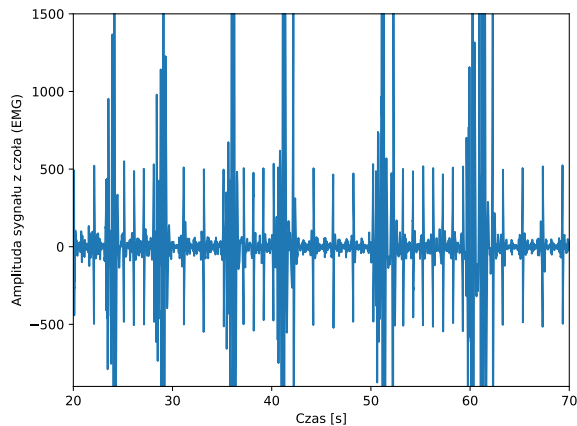
ZBIERANIE DANYCH

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

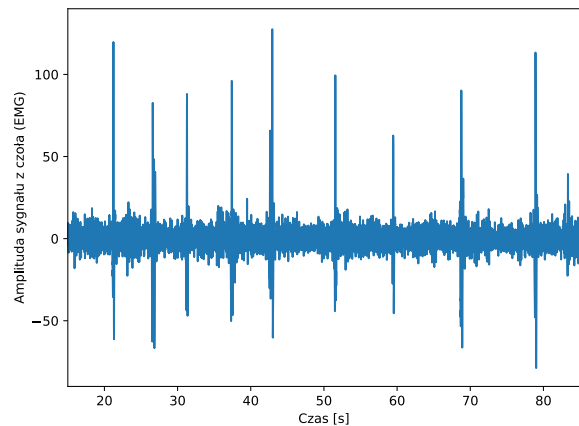
Użyto trzech elektrod - jedną przylegającą do czoła w celu odczytu aktywności mięśni czoła (ruch powiek), oraz dwie przylegające do uszu, służące eliminacji szumu.

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:

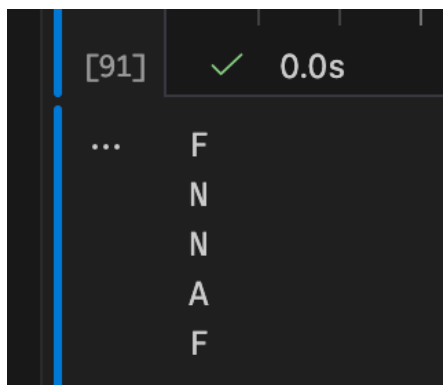
1. Zebranie czasów pojedynczych mrugnięć do tablicy.

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.

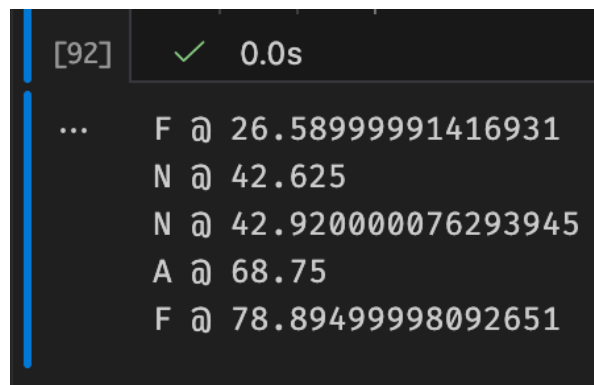
2. Porównanie z wartościami początków i końców wyświetlania liter.

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

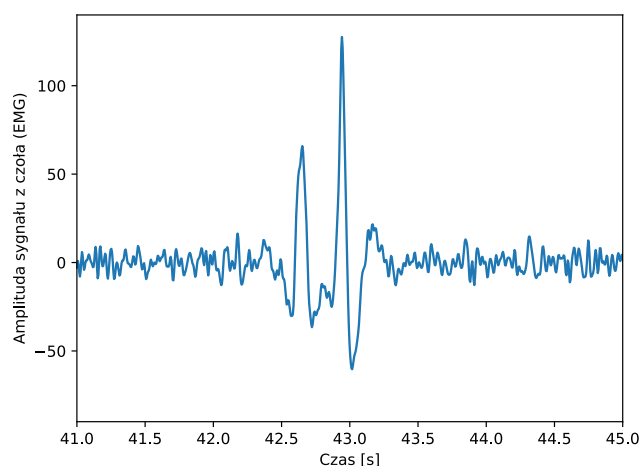


Rysunek 4: Fragment zrzutu ekranu z wynikiem działania wyżej opisanego skryptu.



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.

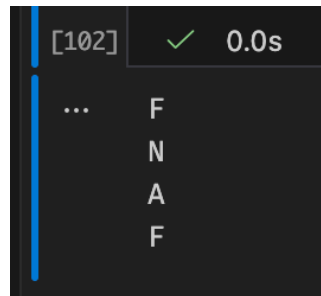
POPRAWA BŁĘDU

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)

```
1  #!/usr/bin/env python3
2  import time
3
4  import pygame
5
6  SCREEN_WIDTH = 1280
7  SCREEN_HEIGHT = 720
8  FONT_SIZE = 220
9  LETTER_INTERVAL = 1.0
10 BREAK_DURATION = 1.0
11 LOG_FILENAME = "letter_log.txt"
12 BG_COLOR = (0, 0, 0)
13 FG_COLOR = (255, 255, 255)
14
15 # fmt: off
16 letters = [
17     "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K",
18     "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V",
19     "W", "X", "Y", "Z"
20 ]
21
22 pygame.init()
23 screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
24 pygame.display.set_caption("SUPER FAJNY PROJEKT LITERKOWO")
25 clock = pygame.time.Clock()
26 FONT = pygame.font.Font(None, FONT_SIZE)
27
28 log_file = open(LOG_FILENAME, "a")
29
30 state = "show_letter"
31 next_switch = time.perf_counter()
32
33 index = 0
34 pending_letter = None
35 pending_start_unix = None
36
37 running = True
38 while running:
39     for event in pygame.event.get():
40         if event.type == pygame.QUIT:
41             running = False
42         elif event.type == pygame.KEYDOWN:
43             if event.key == pygame.K_ESCAPE:
```

```
44         running = False
45
46     now_perf = time.perf_counter()
47     if now_perf >= next_switch:
48         if state == "show_letter":
49             start_unix = time.time()
50             letter = letters[index]
51
52             screen.fill(BG_COLOR)
53             surf = FONT.render(letter, True, FG_COLOR)
54             rect = surf.get_rect(center=(SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2))
55             screen.blit(surf, rect)
56             pygame.display.flip()
57
58             pending_letter = letter
59             pending_start_unix = start_unix
60
61             next_switch = now_perf + LETTER_INTERVAL
62             state = "breaktime"
63
64             index = (index + 1) % len(letters)
65
66         elif state == "breaktime":
67             end_unix = time.time()
68             if pending_letter is not None:
69                 log_file.write(
70                     f"{pending_letter}\t{pending_start_unix:.9f}\t{end_unix:.9f}\n"
71                 )
72                 log_file.flush()
73
74             pending_letter = None
75             pending_start_unix = None
76
77             screen.fill(BG_COLOR)
78             pygame.display.flip()
79
80             next_switch = now_perf + BREAK_DURATION
81             state = "show_letter"
82
83     clock.tick(60)
84
85     if pending_letter is not None:
86         exit_unix = time.time()
87
88         log_file.write(f"{pending_letter}\t{pending_start_unix:.9f}\t{exit_unix:.9f}\n")
```



```
88     log_file.flush()
89
90 log_file.close()
91 pygame.quit()
```

SKRYPT DO ANALIZY GŁÓWNEJ (SKŁADOWA projekt.ipynb)

```
1  # SKRYPT DEKODUJĄCO-PORÓWNAWCZY WERSJA 2 Python
2
3  import pandas as pd
4  import numpy as np
5  import matplotlib.pyplot as plt
6  import aseegg as ag
7
8  df = pd.read_csv("proba2/ganglion_4ch_2025-12-05_10-58-15.csv")
9  przef = ag.gornoprzepustowy(df['ch1'], 200, 1)
10 przef = ag.pasmowozaporowy(przef, 200, 48, 52)
11 przef = ag.pasmowoprzepustowy(przef, 200, 3, 40)
12
13 mrugniecia = []
14 flaga = False
15 granica = 50
16 odstep = 0.3
17 for indeks, wartosc in enumerate(przef):
18     if wartosc > 50 and flaga == False:
19         flaga = True
20         if len(mrugniecia) != 0:
21             if df['time'][indeks] - mrugniecia[-1] >= odstep:
22                 mrugniecia.append(df['time'][indeks])
23         else:
24             mrugniecia.append(df['time'][indeks])
25     if wartosc < 50:
26         flaga = False
27
28 literki = pd.read_csv("proba2/letter_log.txt", names=['litera', 'czas_start',
29 'czas_stop'], sep="\t")
30
31 for mrug in mrugniecia:
32     for kol, rzad in literki.iterrows():
33         if mrug > rzad['czas_start'] and mrug < rzad['czas_stop']:
34             print(rzad['litera'])
```