Raport

Projekt na zajęcia *Komunikacja Człowiek-Komputer*
Jakub Markil, Kaja Ratajczak

Opis zadania:
Projekt polega na stworzeniu prostego programu, opartego na mruganiu. Założenie polega na przekazaniu przez osobę wiadomości (w przypadku naszego projektu - słowa) komputerowi. Opiera się to na mrugającym literowaniu przekazu. Algorytm wyświetla litery, a uczestnik mrugając wybiera poszczególne z nich, aby budować słowa. Następnie za pomocą odczytu czasu mrugnięć i znajomości czasu wyświetlania poszczególnych liter, możemy odkodować wybierane litery i zrekonstruować przekaz.

Wybrane metody:
Przy przefiltrowywaniu danych używamy filtra górnoprzepustowego by odciąć początek sygnału i wyeliminować zakłócenia, następnie filtra pasmowozaporowego - aby pozbyć się zakłóceń w okolicy 50 Hz (zakłócenia z gniazdek) oraz filtra pasmowoprzepustowego, aby wyodrębnić wybrany fragment.

```
filtr_a: list = ag.gornoprzepustowy(pomiary[0]["uV"], freq, 3)
filtr_a = ag.pasmowozaporowy(filtr_a, freq, 48, 52)
filtr_a = ag.pasmowoprzepustowy(filtr_a, freq, 1, 40)
```

```
filtr_b: list = ag.gornoprzepustowy(pomiary[1]["uV"], freq, 3)
filtr_b = ag.pasmowozaporowy(filtr_b, freq, 48, 52)
filtr_b = ag.pasmowoprzepustowy(filtr_b, freq, 1, 40)
```

Do analizy przefiltrowanego sygnału korzystamy z różnic w wartościach sygnału przy mrugnięciach (więcej na ten temat napisaliśmy w następnej sekcji). W celu uniknięcia powtarzania się liter w wynikowym napisie dodaliśmy specjalny warunek sprawdzający czy poprzednia litera nie jest taka sama jak obecnie sprawdzana. Podobnie zrobiliśmy z wartościami “pustymi”, czyli miejscem na swobodne mruganie. Wyniki tych mrugnięć nie są dodawane do ostatecznego napisu. Mogliśmy pozwolić sobie na takie rozwiązania, ponieważ nasze słowa nie mają podwójnych liter i wprowadzamy pojedyncze słowa. Dodatkowo zdecydowaliśmy się na przesunięcie czasowe, które jest swoistym zabezpieczeniem przed latencją spowodowaną czasem reakcji osoby mrugającej.

Opis analizy i uzyskanych wyników:

Za pomocą pętli wykrywane i zapisywane są sygnały powyżej 24050/23800 uV, które poprzedza wartość nie większa niż 22000. Następnym krokiem jest porównanie tego co zapisane, z czasem wyświetlania poszczególnych liter (z uwzględnieniem występujących przerw między wyświetlaniem segmentów liter oraz niewielkiego przesunięcia czasowego w postaci 0,2 sekundy). Jako wynik tych działań otrzymujemy odczyt wymruganych liter, składający się na przekazywane słowo.

```
id_a: list[int] = []

for i in range(1, len(filtr_a)):
    if filtr_a[i] > 24_050 and filtr_a[i-1] <= 22_000:
        id_a.append(pomiary[0]["time"][i])
```

```
wynik_a = '_'
for idx in id_a:
    for i in range(len(litery["time"])-1):
        if idx >= litery["time"][i]+0.2 and idx < litery["time"][i+1]+0.2 and litery["char"][i] != " " and wynik_a[-1] != litery["char"][i]:
            wynik_a += litery["char"][i]

wynik_a = wynik_a[1:]
wynik_a
```

```
'ADEPTB'
```

```
id_b: list[int] = []

for i in range(1, len(filtr_b)):
    if filtr_b[i] > 23_800 and filtr_b[i-1] <= 22_000:
        id_b.append(pomiary[1]["time"][i])
```

```
wynik_b = '_'
for idx in id_b:
    for i in range(len(litery["time"])-1):
        if idx >= litery["time"][i]+0.2 and idx < litery["time"][i+1]+0.2 and litery["char"][i] != " " and wynik_b[-1] != litery["char"][i]:
            wynik_b += litery["char"][i]

wynik_b = wynik_b[1:]
wynik_b
```

```
'BEKON'
```

Opis błędów które się pojawiły i możliwych ulepszeń:

Największym błędem, który został zauważony podczas pisania kodu było to, że sygnał w przypadku drugiego słowa był słabszy od pierwszego (stąd różnica w wartości szukanej amplitudy przy wykrywaniu mrugnięć w obu sygnałach). Niestety nie wiemy co jest przyczyną tego problemu. Drugim błędem jest pojawienie się na końcu litery “B” przy wypisywaniu pierwszego słowa. Tutaj przyczyną jest mrugnięcie na krótko przed zamknięciem programu wyświetlającego litery, co zaowocowało odczytem ostatniej wyświetlonej litery. W tym przypadku można by

próbować “wyrzucić” tę literę poprzez przesunięcie czasowe w odczytach, natomiast mieliśmy obawy czy wartość większa niż obecna (tj. 0,2 sekundy) nie powodowałaby zbyt dużej desynchronizacji między odczytami a wyświetlanymi literami.
Jeżeli chodzi o ulepszenia, na pewno można by trochę zoptymalizować czas działania programu (np. poprzez wstawienie komendy ‘break’ po dodaniu litery do wyniku). Co prawda w naszych przypadkach słowa są krótkie, a odczytów mrugnięć nie ma wiele, lecz w przypadku dłuższych słów bądź zdań wyświetlenie całego wpisu będzie trwało widocznie dłużej. Dodatkowo można by zastąpić pętle ‘for’ pętlami ‘while’, to również przyspieszyłoby działanie programu, co wynika z działania interpretera języka Python. Na potrzeby naszego projektu postanowiliśmy skorzystać z prostszych rozwiązań, które dają nam dokładnie taki sam wynik.