WOJSKOWA AKADEMIA TECHNICZNA
im. Jarosława Dąbrowskiego
WYDZIAŁ ELEKTRONIKI

PRACA DOKTORSKA

mgr inż. Urszula Jagodzińska-Szymańska

METODA KLASYFIKACJI ŹRÓDEŁ SYGNAŁÓW ELEKTRYCZNYCH EMITOWANYCH PRZEZ KORĘ MÓZGOWĄ CZŁOWIEKA – STEROWANIE

Promotor: prof. dr hab. Inż. Edward Sędek

WARSZAWA 2019

Podziękowanie

Autorka serdecznie dziękuje Panu Profesorowi Edwardowi Sędkowi za wyrażenie zgody na objęcie opieką promotorską niniejszej rozprawy doktorskiej i Panu Profesorowi Wiesławowi Sasinowi za konsultacje matematyczne podczas pisania artykułów.

Niniejszą rozprawę doktorską dedykuję mojemu Mężowi i moim Rodzicom, którym dziękuję za cierpliwość i pomoc podczas wykonywania pomiarów.

Streszczenie

Celem rozprawy było opracowanie algorytmów klasyfikacji źródeł sygnałów elektrycznych związanych z rozpoznawaniem intencji ruchu lewą i prawą ręką. Do zlokalizowania źródeł tych sygnałów zostało wykorzystane rozwiązanie zagadnienia odwrotnego. Algorytmy klasyfikacji sygnałów wykorzystują zjawiska desynchronizacji i synchronizacji (ERD/ERS) oraz teorię grafów. W przestawionych w rozprawie algorytmach klasyfikacji klasyfikatorem jest drzewo rozpinające. Wykazano, że sygnały EEG odczytane z 14 elektrod headsetu Emotiv pozwalają na poprawne działanie algorytmu klasyfikacji opracowanego przez autorkę. W celu praktycznego wykorzystania uzyskanych wyników autorka zaprojektowała urządzenie zewnętrzne sterowane poprzez Wi-Fi lub Bluetooth za pomocą uzyskanego wyniku klasyfikacji.

Abstract

The purpose of the dissertation was to develop algorithms for classifying the sources of electrical signals related to recognizing intentions of left-and-right-hand movements. The sources of these signals were located by using the inverse solution. Signal classification algorithms use the desynchronization and synchronization (ERD / ERS) phenomena as well as the graph theory. In the classification algorithms presented in the dissertation the spanning tree is the classifier. The classifying of the algorithm that was developed was proved correct in the tests carried out. It has also been shown that the signals read from the 14 EEG electrodes from the headset Emotiv allow the correct operation of the classification algorithm developed by the author. In order to use the results obtained in practice, the author designed an external device that can be controlled through Wi-Fi or Bluetooth with the use of the classification result.

Spis treści

Wykaz ważniejszych skrótów

AP	- potencjał czynnościowy (Action Potential)
BCI	- Interfejs Mózg-Komputer (Brain Computer Interface)
Bluetooth, WiFi	- standardy bezprzewodowej komunikacji
C++	- język programowania
CUN	- Centralny Układ Nerwowy
Data Set V	- dane z bazy sygnałów Idiap do konkursu BCI Competition III
EEG	- technika do pomiarów sygnałów bioelektrycznych mózgu
ERD	- spadek aktywności
ERS	- wzrost aktywności
HC-05	- układ Bluetooth master/slave
NodeMCU	- układ do komunikacji Wi-fi
Idiap	- Idiap Research Institute w Szwajcarii
K2	- intencja wykonania ruchu lewą ręką
K3	- intencja wykonania ruchu prawą ręką
PC	- komputer
PSD	- widmowa gęstość mocy
SDK (software development kit)	- biblioteki i narzędzia do tworzenia oprogramowania
SPS (samples per second)	- liczba próbek na sekundę
Wiring	- platforma programistyczna użyta przez Autorkę do zaprogramowania układów Arduino Uno R3 oraz NodeMCU

Robocze tłumaczenie ważniejszych terminów angielskich

Primary motor cortex	- pierwszorzędowa kora ruchowa
Supplementary motor cortex	- dodatkowa kora ruchowa
Premotor cortex	- kora przedruchowa
Primary somatic sensory cortex	- pierwszorzędowa kora czuciowa
Prefrontal cortex	- kora przedczołowa
Posterior parietal cortex	- kora ciemieniowa górna
Action Potential	- potencjał czynnościowy
Suprathreshold synaptic potential	- potencjał synaptyczny nad progiem
Subthreshold synaptic potential	- potencjał synaptyczny pod progiem
Progressively increasing stimulus strength	- stopniowo rosnąca siła bodźca

1. Wstęp

Interfejs mózg-komputer (BCI) można definiować jako system, który przekłada wzorce aktywności mózgu użytkownika na komendy. Aktywność mózgu użytkownika BCI jest zwykle mierzona za pomocą elektroencefalografii (EEG). Sygnały EEG są zazwyczaj wstępnie przetwarzane, a cechy są wydobywane z sygnałów tak, aby było możliwe przedstawienie ich w zwartej formie, np. w postaci wektorów cech. W ostatnim dziesięcioleciu można znaleźć przykłady, gdzie zamiast wykorzystywać wektory cech, można reprezentować sygnały EEG przez macierze, np. macierze kowariancji lub przez tensory.

W niniejszej rozprawie wykorzystano macierz t-statystyk liczonych między różnymi obszarami kory mózgowej. Macierz ta jest macierzą pewnego grafu ważonego. Taka technika umożliwia stosowanie algorytmów z teorii grafów. Jest to dziedzina matematyki rozwijająca się bardzo szybko w ostatnich latach.

Wykorzystano drzewo rozpinające, dla grafu zdefiniowanego macierzą t-statystyk. Drzewo rozpinające jest klasyfikatorem. Taka metoda klasyfikacji wykorzystuje jedynie najbardziej istotne cechy. Uniknięto w ten sposób tzw. „przekleństwa wymiarów".

Zazwyczaj cechy są wydobywane za pomocą elektrod rozmieszczonych na powierzchni głowy. Autorka wykorzystywała headset Emotiv Epoc, który ma ograniczoną liczbę elektrod i nie ma elektrod w obszarze kory ruchowej. Wykorzystany w rozprawie klasyfikator oparty jest o wynik rekonstrukcji źródeł sygnałów EEG, co wymagało rozwiązania zagadnienia odwrotnego. Metodą najmniejszych kwadratów uzyskano pewne przybliżenie tych źródeł. Wykonując testy autorka potwierdziła (w ślad za [14]), że klasyfikacja do BCI oparta na EEG musi brać pod uwagę użytkownika. Testując różnych użytkowników można zauważyć, że w zależności od użytkownika, najlepsze wyniki klasyfikacji uzyskuje się w różnych częstotliwościach, a także w różnych przedziałach czasowych. Dlatego uzyskanie takich informacji podczas nauki klasyfikatora ma wpływ na poprawność wyników.

Prawidłowa interpretacja wyobrażania sobie ruchu prawą i lewą ręką daje możliwość sterowania komputerowego bez jakiejkolwiek aktywności fizycznej, co otwiera wiele możliwości zastosowania.

1.1. Motywacja do podjęcia pracy nad klasyfikacją w BCI

Podjęcie pracy nad zagadnieniami związanymi z klasyfikacją źródeł sygnałów elektrycznych emitowanych przez korę mózgową człowieka wynikało z kilku przesłanek. Jedna z nich to chęć poznania oraz poszerzenie wiedzy na temat sygnałów elektrycznych płynących z mózgu człowieka oraz możliwość ich wykorzystania do sterowania urządzeniem zewnętrznym. Tematyka poruszana w rozprawie jest przedmiotem intensywnych badań w wielu ośrodkach badawczych na świecie. Pomimo istnienia bardzo bogatej literatury dotyczącej tej tematyki, istnieje wiele problemów, które nie zostały do tej pory rozwiązane. Jak zostało stwierdzone w [69 str. 22] przyszłe prace nad BCI powinny m.in. koncentrować się na opracowaniu wydajnych algorytmów klasyfikacji opartej na EEG, które mogą zarówno pracować *online*, jak również z małymi próbkami szkoleniowymi. Autorka rozprawy podjęła taką pracę.

Kolejną przesłanką do podjęcia tej tematyki było napisanie przez autorkę niniejszej rozprawy kilkunastu artykułów na temat BCI, z których niektóre przytoczone zostały w bibliografii [39 ÷ 50]. Artykuły te były ukierunkowane na opracowanie takiego klasyfikatora, który umożliwiłby sterowanie urządzeniem w praktyce.

Następną przesłanką było wykorzystanie możliwości zastosowania teorii grafów do klasyfikacji sygnałów emitowanych przez korę mózgową, jako techniki do klasyfikacji sygnałów w BCI. Klasyfikatory wykorzystujące teorię grafów w BCI oparte na zrekonstruowanych źródłach sygnałów w EEG mogą być prostą i skuteczną metodą do prawidłowej klasyfikacji niewymagającej długiej nauki klasyfikatora.

1.2. Cel i teza rozprawy doktorskiej

Celem rozprawy było opracowanie algorytmów rekonstrukcji źródeł sygnałów EEG (rozwiązanie zagadnienia odwrotnego), wyodrębniania cech i ich selekcji oraz klasyfikacji; klasyfikacja dotyczy powiązania dwóch obiektów, intencji ruchu lewą ręką – obiekt K2 i prawą ręką – obiekt K3, z odpowiadającymi im podzbiorami zbioru cech. Do wykonywania testów wykorzystywane było urządzenie mobilne. Elektrody tego urządzenia nie pokrywały w pełni obszaru kory ruchowej. Do klasyfikacji K2 i K3 wykorzystano teorię grafów. Opisane w rozprawie testy rozpoznawania intencji ruchu (K2, K3) z uwzględnieniem

indywidualnego doboru częstotliwości i przedziałów czasowych dla osoby testowanej dowodzą następującej tezy:

Wykorzystanie drzewa rozpinającego grafu (teoria grafów) do klasyfikacji intencji ruchu w oparciu o informacje o źródłach sygnałów (rozwiązanie zagadnienia odwrotnego), pomimo ograniczonej liczby elektrod niepokrywających w pełni obszarów kory ruchowej, pozwala na prawidłowe rozpoznawanie intencji ruchu prawą i lewą ręką.

1.3. Wkład własny autorki

W ramach prowadzonych badań związanych z rozpoznawaniem intencji ruchu lewą i prawą ręką autorka zastosowała drzewo rozpinające grafu do klasyfikacji sygnałów EEG do BCI. Innowacyjne osiągnięcia autorki to:

- opracowanie algorytmów lokalizacji źródeł sygnałów EEG na powierzchni kory mózgowej z wykorzystaniem teorii grafów oraz wyznaczanie gęstości prądu tych źródeł,
- opracowanie algorytmów wykorzystania drzewa rozpinającego grafu jako klasyfikatora rozpoznawania intencji ruchu lewą i prawą ręką,
- implementacja algorytmów klasyfikacji intencji ruchu w środowisku Matlab,

Dodatkowo autorka:

- opracowała bibliotekę dzieloną (eegloglib.dll) w języku C++ pobierającą dane surowe (*raw*) z elektrod headsetu Emotiv i zapisującą pliki *raw* w podkatalogach odpowiadającym kolejnym pomiarom na podstawie SDK Emotiv Epoc,
- napisała programy sterujące do układu NodeMCU (komunikacja WiFi) oraz płytki Arduino Uno R3 (komunikacja Bluetooth)
- opracowała program analizujący wyniki pomiarów w VBA w Excel.

Autorka oprogramowała w środowiskach Matlab, C++ oraz Wiring (Arduino Uno R3, NodeMCU) cały system BCI od rejestracji danych EEG, poprzez ich selekcję i klasyfikację aż do przekazania komendy sterującej do urządzenia zewnętrznego realizującego tę komendę. Wszystkie testy były wykonywane przy pomocy tego systemu BCI.

Autorka zastosowała klasyfikator minimalno-odległościowy [96]. Rozpoznawane są obiekty K2 oraz K3. W rozprawie zastosowano podział danych na uczące i testujące.

Klasyfikacja jest nadzorowana. Jak podaje źródło [96] pojęcie obiekt jest bardzo szerokie. Każdy obiekt reprezentowany jest przez podzbiory zbioru (klasy), które charakteryzują obiekty. W klasyfikacji nadzorowanej ustalane są klasy cech reprezentujące dane obiekty. Po wytypowaniu cech reprezentujących obiekty, uzyskane na tym etapie klasy cech są podstawą poprawnej klasyfikacji.

W celu praktycznego wykorzystania wyniku klasyfikacji autorka zaprojektowała proste urządzenie, którym można sterować i jako impuls sterujący wykorzystała wynik klasyfikacji. Wynik klasyfikacji jest bezprzewodowo przekazywany do urządzenia z użyciem komunikacji poprzez Wi-fi lub Bluetooth.

1.4. Układ rozprawy

Rozprawa podzielona jest na 8 rozdziałów. W rozdziale drugim znajdują się informacje o aktualnym stanie wiedzy w zakresie selekcji i klasyfikacji sygnałów EEG w BCI szczególnie w ostatnich dziesięciu latach i wkład autorki na tym tle.

W rozdziale trzecim przedstawiono opis stosowanych algorytmów w zaproponowanym systemie BCI opartym o rozwiązanie zagadnienia odwrotnego.

W rozdziale czwartym zaprezentowane zostały wyniki uzyskane w testach, które autorka przeprowadziła dla danych własnych uzyskanych za pomocą headsetu Emotiv oraz przedstawiono opis wyników uzyskanych przy testowaniu algorytmów dla danych Idiap[1].

W rozdziale piątym przedstawiono realizację sprzętową systemu BCI prezentowanego w rozprawie. Autorka przedstawiła skonstruowane przez nią proste urządzenie, którym można sterować przy pomocy aktywności myślowych K2 i K3, z opcjami sterowania bezprzewodowego przez Wi-fi i Bluetooth.

W rozdziale szóstym przedstawiono realizację programową służącą do klasyfikacji wyników pomiarów aktywności myślowych i sterowania urządzeniem zewnętrznym. Przedstawiono schematy programu opracowanego przez autorkę w języku Matlab oraz przykładowe fragmenty kodu (pełny kod programu klasyfikującego znajduje się w zał. 4).

[1] Idiap Research Institute (IDIAP) – więcej o danych Idiap: http://www.bbci.de/competition/iii/desc_V.html (stan na dzień 02.10.2019 r.). Autorka, do testowania, wykorzystała pakiet Data Set V - http://www.bbci.de/competition/iii/#data_set_v (stan na dzień 02.10.2019 r.). Więcej na temat danych testowych Idiap w rozdziale 6.

W rozdziale siódmym znajduje się podsumowanie uzyskanych wyników, a także zaproponowane kierunki dalszych prac badawczych związanych z zaprezentowaną metodą klasyfikacji.

Załącznik 1. zawiera dowód wzoru (24) przedstawiającego problem prosty w postaci układu równań liniowych.

W załączniku 2. podano ogólną charakterystykę rejestrowanych sygnałów EEG oraz informacje o źródłach tych sygnałów zlokalizowanych na powierzchni kory mózgowej.

Załącznik 3. zawiera ostateczne wyniki BCI Competition III – Data Set V.

Załącznik 4. prezentuje pliki składowe programu klasyfikującego napisane w środowisku Matlab.

2. Aktualny stan wiedzy nt. algorytmów klasyfikacji

Przegląd algorytmów klasyfikacji używanych od dawna [69]:

1. Klasyfikatory liniowe – wykorzystują liniowe granice decyzyjne między wektorami cech każdej klasy. Do tej grupy należą: maszyna wektorów wspierających (SVM), liniowa analiza dyskryminacyjna (LDA) [71].

a. Metoda SVM polega na separacji zbioru próbek różnych klas przy pomocy pewnej hiperpłaszczyzny. Metoda ta działa przy dowolnym rozkładzie. Jeżeli założy się separowalność liniową, to równanie hiperpłaszczyzny jest postaci:

$$u^T v + b = 0 \qquad (1)$$

gdzie $u^T = [u_1, \dots, u_N]$, $v = [v_1, \dots, v_N]^{\mathrm{T}}$, $b \in \boldsymbol{R}$, wektor u – opisuje hiperpłaszczyznę. Optymalna hiperpłaszczyzna to taka, która maksymalizuje szerokość marginesu zdefiniowanego przez hiperpłaszczyzny $u^T v = -1$ oraz $u^T v = 1$. Wewnątrz tego pasa nie znajdują się żadne próbki. Wartość współrzędnych wektora u ustala się przez naukę klasyfikatora. Dwie klasy próbek znajdują się w półprzestrzeniach $u^T v > 1$ oraz $u^T v < -1$. Jeżeli przy klasyfikacji sygnałów jest więcej klas niż dwie klasy, to wprowadza się modyfikację metody SVM. Najczęściej używane są strategie: 1^{o} jedna klasa przeciw wszystkim, 2^{o} jedna klasa przeciw jednej oraz jedna przeciw pozostałym [84]. W artykule [57] do klasyfikacji sygnałów EEG wykorzystano maszynę wektorów wspierających dla trzech klas zdarzeń biorąc funkcję jądra o postaci

$$K(u, v) = \varphi^T(u)\varphi(v) \qquad (2)$$

innej niż liniowa (w liniowym SVM $K(u, v) = u^T v + b$). Wówczas przy wielomianowej funkcji jądra $K(u, v) = (u^T v + b)^p$, gdzie p = 2, 3, … (stopień wielomianu), przy radialnej funkcji jądra $K(u, v) = exp\ (-b \cdot \|u - v\|^2)$. W przeprowadzonych testach najlepsze wyniki klasyfikacji sygnałów EEG autorzy artykułu uzyskali dla funkcji jądra, która jest wielomianem stopnia drugiego (p = 2).

b. LDA (Linear Discriminant Analysis) jest techniką przetwarzania sygnałów wykorzystywaną w uczeniu maszynowym, której zadaniem jest znalezienie najlepszej kombinacji sygnałów, rozdzielających dwie lub więcej klas sygnałów. Wynik tego działania może być wykorzystany jako liniowy klasyfikator lub jako technika zmniejszania wymiaru przestrzeni sygnałów przed uruchomieniem procesu klasyfikacji. Metoda rozwiązania polega na wyznaczeniu minimum warunkowego

$$\begin{cases} min_w \left(-\frac{1}{2} W^T \cdot S_B \cdot W\right) \\ takie, że\ W^T \cdot S_W \cdot W = 1 \end{cases} \tag{3}$$

gdzie

$$S_B = \sum_c N_c \, (\mu_c - \overline{x}) \cdot (\mu_c - \overline{x})^T \tag{4}$$

$$S_W = \sum\nolimits_c \sum_{i \in c} (x_i - \mu_c) \cdot (x_i - \mu_c)^T \tag{5}$$

$$\mu_c = \frac{1}{N_c} \sum_{i \in c} x_i \ , \overline{x} = \frac{1}{N} \sum_i x_i = \frac{1}{N} \sum_c N_c \cdot \mu_c \tag{6}$$

N_c – liczba próbek w klasie c.

Są to najbardziej popularne klasyfikatory, stosowane od wielu lat, dla BCI opartych na EEG, szczególnie pracujących *online* i w czasie rzeczywistym.

2. Sieci neuronowe (NN) – zespoły sztucznych neuronów ułożone w warstwach, które mogą być wykorzystane do przybliżenia dowolnej nieliniowej granicy decyzyjnej [36] Najczęściej używanym typem NN w BCI był wielowarstwowy perceptron (MLP). Inne typy NN, takie jak klasyfikator Gaussa NN, czy kwantyzacja uczenia się wektora wykorzystywane były rzadko. Obecnie wykorzystywane są nowsze typy klasyfikatorów, np. głębokie sieci neuronowe (DNN), które zostały zbadane dla wszystkich głównych typów systemów BCI opartych na EEG. Jednak wszystkie te badania przeprowadzono w trybie *offline*.

3. Nieliniowe klasyfikatory bayesowskie, to klasyfikatory modelujące rozkłady prawdopodobieństwa każdej klasy. Klasyfikatory te wykorzystują reguły Bayesa, aby wybrać klasę do przypisania do bieżącego wektora cech.

Do tej grupy klasyfikatorów należą m.in.: kwadratowe klasyfikatory Bayesa, ukryte modele Markowa (HMM).

4. Klasyfikatory najbliższych sąsiadów. Klasyfikatory te przypisują klasę do bieżącego wektora cech zgodnie z najbliższymi sąsiadami. Takimi sąsiadami mogą być treningi wektorów cech lub prototypy klas.

Do tej rodziny klasyfikatorów należą: algorytm k najbliższych sąsiadów (kNN) oraz klasyfikatory minimalno-odległościowe Mahalanobisa, Czebyszewa, Manhattan (w zależności od wykorzystywanej metryki). Klasyfikatory te charakteryzują się dużą

złożonością pamięciową i obliczeniową. Zaletą jest prostota i duża efektywność klasyfikatora.

5. Łączenie klasyfikatorów (kombinacje klasyfikatorów), to algorytmy łączące wiele klasyfikatorów.

Wykorzystywane tu metody to np.: Bootstraping, Bagging, Bootsting, Zespoły klasyfikatorów (podejmowanie decyzji za pomocą głosowania), Lasy losowe (metoda stosowana do drzew decyzyjnych; najpierw losuje się n prób bootstrapingowych, a następnie do każdej takiej próby buduje się drzewo decyzyjne). Zaletą jest duża efektywność ale głównie w klasyfikacji *offline*.

W ostatniej dekadzie głównymi wyzwaniami stojącymi przed metodami klasyfikacji sygnałów EEG to:

- niski stosunek sygnału do szumu [78], [105]
- niestacjonarność w czasie dla jednego użytkownika lub kilku użytkowników, tzn. sygnały zmieniają się w ramach jednej serii pomiarów [24], [34], [95]
- ograniczona liczba ogólnie dostępnych danych treningowych do kalibracji klasyfikatorów [58], [66]
- niska niezawodność i wydajność użytkowanych BCI [59], [67], [70], [104].

Najnowsze metody klasyfikacji EEG uwzględniają te postulaty [103]. Jedne z najnowszych rozwiązań to:

6. Klasyfikatory adaptacyjne - są to klasyfikatory, których parametry są stopniowo aktualizowane w czasie, gdy nowe dane EEG stają się dostępne [94], [95]. Klasyfikatory te mogą wykorzystywać adaptację nadzorowaną i nienadzorowaną. Przy adaptacji nadzorowanej znane są prawdziwe etykiety klas przychodzących sygnałów EEG, a klasyfikator jest przekwalifikowany (aktualizowany) na dostępne dane treningowe. W adaptacji bez nadzoru etykiety przychodzących danych nie są znane. Możliwe jest wtedy oszacowanie etykiet klas danych w celu przekwalifikowania klasyfikatora. Badane były też klasyfikatory częściowo adaptacyjne. Zbadano adaptacyjne klasyfikatory w trybie *offline* i *online*, takie jak LDA, QDA (kwadratowa analiza dyskryminacyjna) [100], [101], [38].

Autorkę szczególnie interesowały EEG w zakresie sygnałów sensomotorycznych, gdzie oprócz wyżej wymienionych wykorzystano adaptacyjny klasyfikator Gaussa [65], adaptacyjny probabilistyczny NN. W [76] zostało pokazane, że ciągłe przekwalifikowanie

parametrów klasyfikatora w przypadku rytmów sensomotorycznych poprawiło jego wydajność.

Klasyfikatory adaptacyjne mogą być wykorzystane w przypadku małej liczby danych treningowych, przez uczenie się *online*. Jednak metody adaptacyjne, które wymagają pełnego przekwalifikowania klasyfikatora mają dużą złożoność obliczeniową, co na ogół uniemożliwia ich wykorzystanie w trybie *online*.

7. Klasyfikacja EEG macierzy i tensorów:

a. Klasyfikacja oparta na geometrii Riemanna (RGC). Metoda polega na odwzorowaniu danych na przestrzeń geometryczną wyposażoną w odpowiednią metrykę. W tej przestrzeni można wykonywać uśrednienie, aproksymację, interpolację oraz klasyfikację. Dla danych EEG odwzorowanie wymaga obliczenia jakiejś formy macierzy kowariancji. Zakłada się, że moc i przestrzenny rozkład źródeł EEG można uznać za stałe dla danego stanu psychicznego. W geometrii Riemanna badane są przestrzenie nazywane rozmaitością oraz przestrzenie styczne w punkcie (jest to lokalne liniowe przybliżenie rozmaitości). Wśród najczęściej wykorzystywanych w BCI rozmaitości macierzowych są macierze hermitowskie lub symetryczne dodatnio określonych (SPD). W rozmaitości Riemanna przestrzeń styczna wyposażona jest w iloczyn wewnętrzny (metryczny), który definiuje odległość np. kwadrat odległości między macierzami C_1, C_2 może być określony wzorem

$$\sigma^2(C_1, C_2) \approx \sum_n log^2\, \lambda_n(C_1^{-1}C_2) \tag{7}$$

gdzie $\lambda_n(M)$ oznacza n-tą wartość własną macierzy M.

Używany klasyfikator RMDM (Riemann Minimum Distance to Mean [6], [8]) oblicza średnią geometryczną dla każdej klasy wykorzystując dane treningowe, a następnie przypisuje próbę nieoznakowaną do klasy odpowiadającej najbliższym sąsiadom.

Inna grupa klasyfikatorów opartych na geometrii Riemanna wykorzystuje przestrzeń styczną, gdzie używane są standardowe klasyfikatory LDA, SVM i inne [8], [7]. Procedury wykorzystujące geometrię Riemanna, jak RMDM, są prostsze niż procedury klasyczne, natomiast liczenie odległości Riemanna między dwiema macierzami może powodować problemy numeryczne (gdy np. najmniejsza wartość własna $(C_1^{-1}C_2)$ jest bliska zeru), a gdy wzrasta liczba elektrod operacja logarytmiczna jest źle uwarunkowana i numerycznie niestabilna. Zastosowanie klasyfikatorów Riemanna ma wysoką złożoność obliczeniową.

b. Wyodrębnianie cech i klasyfikacja za pomocą tensorów. Tensory zapewniają naturalną reprezentację z danych EEG. Prawie wszystkie algorytmy uczenia maszynowego

zostały rozszerzone lub uogólnione na tensory, np. SVM został uogólniony na TSM, standardowa metoda LDA została uogólniona do analizy dyskryminacyjnej Fishera (TFDA) lub analizy dyskryminacyjnej wyższego rzędu (HODA) [19], [89]. Tensory dostarczają narzędzi do analizy BCI i fuzji ogranych zbiorów danych, ale złożoność obliczeniowa metod tensorowych jest znacznie wyższa niż w przypadku metod standardowych. Metody te pojawiły się niedawno, jako narzędzie do ekstrakcji cech i klasyfikacji i nie są jeszcze dobrze dopracowane.

Inne klasyfikatory stosowane w minionych latach, to:

8. Głęboka nauka. Najbardziej popularne metody głębokiego uczenia się dla BCI to: splotowe sieci neuronowe (CNN) [31], [62], ograniczone maszyny Bolzmanna (RBM) [63] głębokie sieci neuronowe (DNN) [15]. Metody te nie są jeszcze do końca zbadane, ale np. DNN wiąże się z dużą liczbą parametrów, co wymaga dużej liczby przykładów szkoleń w celu ich aktualizacji.

9. Klasyfikatory, które do nauki nie potrzebują wielu danych. Podstawowe typy to: klasyfikator LDA skurczu (sLDA the Shrinkage LDA Classifier), losowy klasyfikator lasu (RF) – jest to zestawienie kilku klasyfikatorów drzew decyzyjnych, klasyfikator Riemanna (RMDM) [12], [21], [66], [68], [106]. Klasyfikatory te są łatwe w użyciu i zapewniają dobre wyniki. Złożoność obliczeniowa klasyfikatora RF jest większa niż pozostałych. Autorka w swojej pracy wykorzystała zaprojektowany przez siebie klasyfikator wykorzystujący teorię grafów (drzewo rozpinające), który do nauki nie potrzebuje wielu danych.

10. Klasyfikatory wielowarstwowe. W celu sklasyfikowania większej liczby zadań umysłowych niż dwa zadania można zastosować dwa podejścia:

- pierwsze podejście polega na bezpośrednim oszacowaniu klasy przy użyciu technik wieloklasowych, takich jak: drzewa decyzyjne, perceptory wielowarstwowe, naiwne klasyfikatory Bayesa lub k-najbliższych sąsiadów,
- drugie podejście polega na rozłożeniu problemu na kilka problemów klasyfikacji binarnej [2].

Taka dekompozycja może być realizowana na różne sposoby przy użyciu:

- klasyfikatorów parami jeden na jeden [11], [36]
- klasyfikatorów jeden na reszta [11], [36]
- hierarchicznych klasyfikatorów podobnych do binarnego drzewa decyzyjnego
- klasyfikatorów wieloznakowych [72], [99]

W tym podejściu odrębny podzbiór etykiet jest powiązany z każdą klasą. Przewidywana klasa jest identyfikowana zgodnie z najbliższą odległością między przewidywanymi etykietami oraz każdym podzbiorem etykiet definiujących klasę [26]. Zaletą tego podejścia jest znaczne zwiększenie liczby różnych wyobrażeń dotyczących podzbiorów w zbiorze wszystkich części ciała, które użytkownik jest w stanie sobie jednocześnie wyobrazić w porównaniu z licznością zbioru wszystkich części ciała. [64], [107], [108].

11. Inną grupę klasyfikatorów EEG stanowią klasyfikatory wykorzystujące rekonstrukcję źródeł sygnałów EEG (rozwiązanie zagadnienia odwrotnego) [87], [88], [29]. Autorka w swojej rozprawie zastosowała klasyfikator oparty na wyznaczonych danych dotyczących położenia i gęstości prądu na powierzchni kory mózgowej.

W artykule [80] wykazano, że klasyfikator źródeł sygnałów oparty na rekonstrukcji źródeł sygnałów daje wyniki klasyfikacji podobne jak klasyfikatory prawidłowo klasyfikujące źródła sygnałów EEG. W artykule zaproponowano, jako klasyfikator liniowy SVM oraz pSVM (proximal), który jest szybszy. Tego typu metodę klasyfikacji opartą na rozwiązaniu zagadnienia odwrotnego przedstawiono w [83]. W artykule tym miarą oceny różnic między poszczególnymi klasami zadań myślowych był współczynnik F_{x-y}

$$F_{x-y} = log\left[\left(N_y \cdot \sum_{i}^{N_x} J_{i,x}\right) / \left(N_x \cdot \sum_{i}^{N_y} J_{i,y}\right)\right] \tag{8}$$

gdzie J_x, J_y to porównywane zbiory danych.

W metodzie klasyfikacji opartej o rozwiązanie zagadnienia odwrotnego autorka w swojej pracy do porównania wykorzystała t-statystykę rekomendowaną w artykule [56]. W związku z rekomendacją autorów artykułu popartą analizą wyników klasyfikacji autorka zdecydowała się na wykorzystanie t-statystyki.

Wyznaczanie przybliżonego rozwiązania zagadnienia odwrotnego (lokalizacji źródeł sygnałów EEG i gęstości w tych źródłach) polega na rozwiązaniu układu równań

$$KJ = \emptyset + \eta \tag{9}$$

gdzie K – macierz o wymiarze $n_e \times n_v$, J – macierz o wymiarze $n_v \times 1$, $\emptyset$ – macierz o wymiarze $n_e \times 1$ gdzie n_e to liczba elektrod, n_v to liczba dipoli (wokseli) na powierzchni kory mózgowej, przy czym $n_e \ll n_v$, η – szum (o rozkładzie z zerową wartością średnią i macierzą kowariancji C_η).

Wyznaczone rozwiązanie $\hat{J}$ jest obarczone błędem, który można zminimalizować (rozdział 3.3.3.). W metodzie klasyfikacji sygnałów EEG w oparciu o wyznaczanie źródeł tych sygnałów jest to jeden z podstawowych problemów.

Do wyznaczania przybliżonego rozwiązania układu równań (9) można zastosować regularyzację Tichonowa [85] wykorzystując wzór

$$\hat{J} = min_J(\|\emptyset - KJ\|^2_{C_n} + \lambda^2 \cdot \|H \cdot J\|^2) \tag{10}$$

gdzie norma

$$\|\Phi - KJ\|^2_{C_n} = (\emptyset - KJ)^T \cdot C_\eta^{-1} \cdot (\Phi - KJ) \tag{11}$$

gdzie λ – parametr regulacyjny, H – macierz zawierająca informację *a priori* o rozwiązaniu układu równań.

Ostatecznie otrzymuje się

$$\hat{J} = T \cdot \emptyset \tag{12}$$

gdzie T jest macierzą pseudoodwrotną Moora-Penrosa.

Najczęściej używane cechy do reprezentowania sygnału to cechy mocy pasma, częstotliwości i cechy punktu czasowego. Autorka niniejszej rozprawy wykorzystywała cechy mocy i gęstości natężenia prądu na powierzchni kory mózgowej (rozdziały 3.2, 3.4).

Cechy mocy pasma, które reprezentują moc sygnałów, mogą być obliczane różnymi metodami [13], [37]. Funkcje punktu czasowego są konkatenacją próbek EEG z wszystkich kanałów. Takie cechy są zwykle wyodrębniane po wstępnym przetwarzaniu, jak np. filtrowanie pasmowe lub dolnoprzepustowe. Wykorzystywane typy cech zyskują dzięki ekstrakcji po filtrowaniu przestrzennym [90]. Filtrowanie przestrzenne polega na łączeniu oryginalnych sygnałów z czujników, co pozwala uzyskać sygnał o wyższym stosunku sygnału do szumu. Filtry przestrzenne to: filtr Laplace'a, odwrotne filtry przestrzenne [9], nienadzorowane filtry przestrzenne, takie jak analiza głównych składowych (PCA) lub niezależna analiza komponentów (ICA) [53]. Nadzorowane filtry przestrzenne to wspólne wzory przestrzenne (CSP) [90]. Rozszerzeniem tych podejść jest metoda banku filtrów CSP [4]. Zbadano również inne typy cech, przykładowo [4]. Takie typy cech mierzą korelację lub synchronizację między sygnałami z różnych czujników oraz różnych pasm częstotliwości [4]. Do mierzenia tych zależności służą różne funkcje, jak np. spójność widmowa czy ukierunkowane funkcje przenoszenia [33], [60].

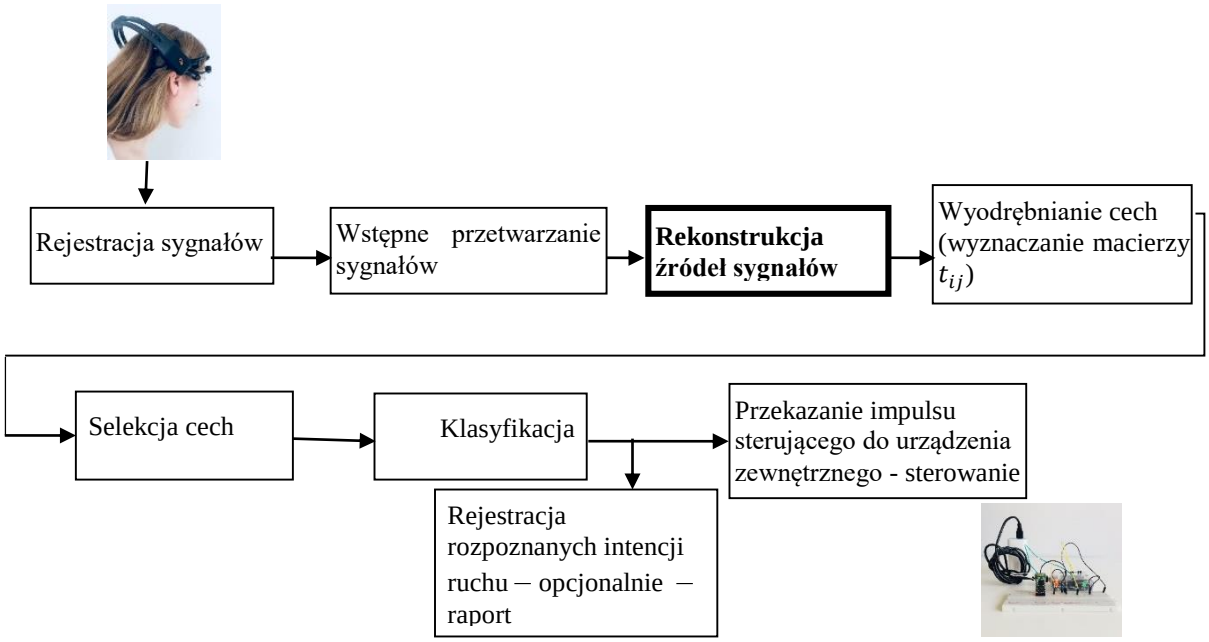
W innych pracach zamiast wykorzystywać wektory cech reprezentowano sygnały EEG przez macierz kowariancji lub tensory o wymiarze 2 lub wyższym [33], [102]. Istotna jest wtedy metoda jak klasyfikować bezpośrednio te macierze i tensory [18], [21].

W przypadku BCI problemem jest wykorzystywanie wiedzy zdobytej podczas uczenia się danego zadania do rozwiązania innego zadania (w przypadku, gdy dane pozyskiwane są dla różnych osób i w różnych sesjach czasowych). Uczenie transferowe (*transfer learning*)

ma na celu radzenie sobie z tym problemem. Prace w tym zakresie opisano w [17], [51]. Badania te są niezbędne, aby w przyszłości uzyskać tryb pracy BCI bez kalibracji, co poprawiłoby akceptację BCI. Otrzymywanie informacji zwrotnych umożliwia aktualizację parametrów klasyfikacji (co zastosowała autorka w swojej pracy). Zatem uczenie transferowe w powiązaniu z klasyfikatorami adaptacyjnymi może pozwolić na wyeliminowanie kalibracji [20], [75]. Jest to jedno z zagadnień, które w ostatnich latach stanowi temat badań w BCI. Uczenie się transferu ma szczególne znaczenie w sytuacjach, gdy istnieje wiele oznakowanych danych dla jednego zadania, oznaczonych jako domena źródłowa, podczas gdy pozyskiwanie danych dla drugiego zadania jest rzadkie lub kosztowne, oznaczone jako domena docelowa. Wówczas przekazywanie wiedzy z domeny źródłowej do domeny docelowej działa jak stabilizator do rozwiązania zadania docelowego. W artykule [86] domena jest definiowana przez przestrzeń cech i rozkład prawdopodobieństwa typu dyskretnego. Przestrzeń cech jest powiązana z przestrzenią etykiety przez wspólny rozkład prawdopodobieństwa. Stosowana metoda adaptacji domeny polega na dopasowaniu dystrybuant domeny źródłowej i docelowej przez zastosowanie odwzorowania domen.

3. Opis zastosowanych algorytmów w systemie BCI opartym o rozwiązanie zagadnienia odwrotnego

Algorytmy zastosowane w pracy oraz kolejność ich wykorzystania mogą być najogólniej opisane przy pomocy schematu - rys 1:



Rysunek 1 Problem odwrotny w BCI[2]

3.1. Rejestracja sygnałów – wykorzystywane urządzenie

Autorka wykorzystała do rejestracji sygnałów headset Epoc firmy Emotiv. Jest to bezprzewodowy, przenośny system umożliwiający rejestrację surowych sygnałów EEG (*raw data*) za pomocą 14 aktywnych elektrod, które rejestrują odczyty aktywności elektrycznej neuronów w mózgu.

Podstawowe parametry headsetu Emotiv:

14 aktywnych elektrod: AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4 oraz dwie elektrody referencyjne P3, P4

- Monitorowanie stanu użytkownika w czasie rzeczywistym
- Częstotliwość próbkowania: f_s = 128 Hz (SPS)
- Rozdzielczość przetwornika analogowo-cyfrowego: 14 bit
- Szerokość pasma: 0.2-43 Hz, filtry cyfrowe 50 i 60 Hz

[2] Rysunek i opis schematu - opracowanie własne na podstawie [79], [42]

- Komunikacja z PC: bezprzewodowa (Bluetooth)

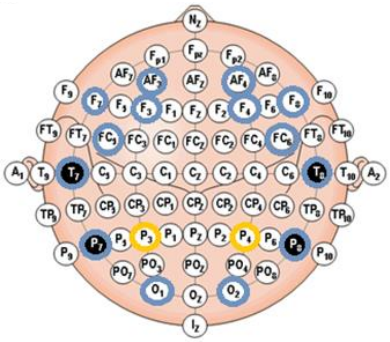
Headset Emotiv posiada aplikacje i biblioteki programistyczne, w tym interfejs graficzny pozwalający na monitorowanie stanów emocjonalnych użytkownika w czasie rzeczywistym. Umożliwia obserwowanie poziomu zaangażowania, entuzjazmu i ożywienia, zmęczenia, niezadowolenia, skupienia. Pomiar tych stanów autorka wykorzystała do wstępnego określenia formy użytkownika, przed rozpoznawaniem obiektów K2 lub K3.

Za pośrednictwem biblioteki C++ (eegloglib.dll) utworzonej przez autorkę na bazie SDK Emotiv Epoc (biblioteki i narzędzia dostarczone razem z headsetem Emitive Epoc), surowe sygnały (*raw data*) odczytane z elektrod, zostały zapisane w pliku i przekazane do programu klasyfikującego [49].

Rys. 2 przestawia wykorzystywany przez autorkę headset Emotiv Epoc.

Rysunek 2 Headset Emotiv Epoc[3]

Aktywne elektrody headsetu Emotiv Epoc zostały zaznaczone na rys. 3 kolorem niebieskim, natomiast kolorem żółtym oznaczono elektrody referencyjne. Aktywność rejonów ruchowych mózgu zbierana jest przez 14 elektrod Emotiv Epoc, co jak wykazano w rozprawie, jest wystarczające do rozwiązania zagadnienia odwrotnego.

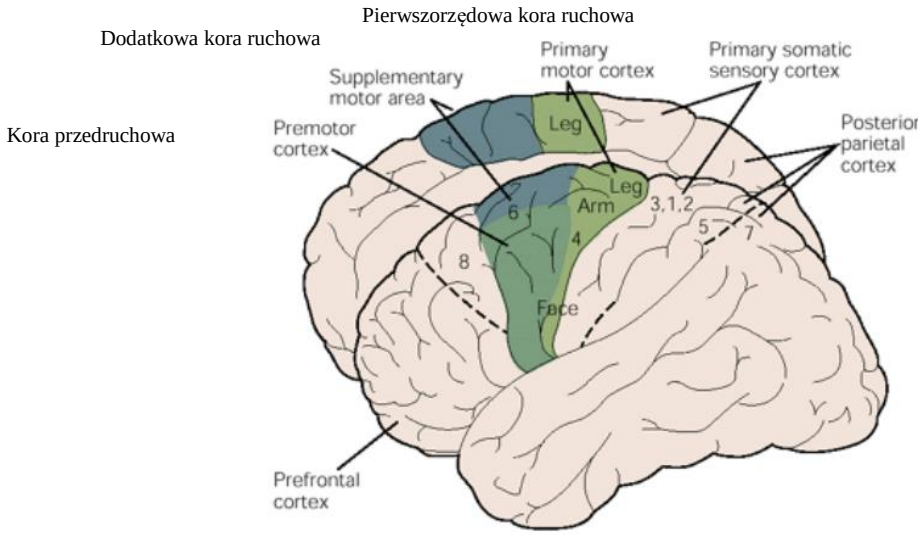


Rysunek 3 Rozmieszczenie elektrod headsetu Epoc na powierzchni głowy[4]

[3] Producent headsetu Emotiv Epoc: https://www.emotiv.com/ (stan na dzień 04-10-2019).

[4] Rysunek na podstawie http://www.bem.fi/book/13/13.htm - stan na dzień 04.02.2017 r. Cztery elektrody mają zmienione nazwy w stosunku do systemu 10-20 – zaznaczone na rysunku kolorem czarnym [1].

Można zauważyć, że elektrody headsetu Emotiv nie znajdują się bezpośrednio nad pierwszorzędową i drugorzędową korą ruchową, tzn. polami Brodmanna 4 i 6.



Rysunek 4 Pola Brodmanna 4 i 6[5]

3.2. Wstępne przetwarzanie sygnałów

Następnym etapem postępowania po rejestracji sygnałów było wstępne ich przetwarzanie (*preprocessing*), które polegało przede wszystkim na policzeniu Widmowej Gęstości Mocy (*Power Spectral Density* – PSD) w zakresie częstotliwości 8 – 30 Hz. Autorka zrezygnowała z liczenia przestrzennej filtracji Laplace'a, gdyż wobec niewielkiej liczby elektrod nie wpływała na poprawę wyników. Filtrację w zakresie częstotliwości wykonuje headset. Dla częstotliwości próbkowania headsetu, f_s =128 Hz dla próbki $N=64$ elementowej rozdzielczość z jaką liczona jest widmowa gęstość mocy wynosi $\frac{f_s}{N}$ = 2 Hz. Widmo sygnału wyznaczono stosując dyskretną transformatę Fouriera (DFT). Po zastosowaniu tej transformaty N-elementowy ciąg próbek *x[1], …, x[N]* zostaje przekształcony w ciąg prążków $X_1, \dots, X_N$ według wzoru[6]

$$X_k = \sum_{n=1}^{N} x[n] \cdot exp\left(\frac{-i \cdot 2\pi kn}{N}\right), \qquad (13)$$

gdzie k jest numerem prążka. W wyniku stosowania dyskretnej transformacji Fouriera uzyskuje się wartości amplitudy sygnału EEG (wybierane są prążki od 1 Hz do 40 Hz). Aby

[5] Rysunek na podstawie [24].

[6] Wzór na dyskretną transformatę Fouriera i odwrotną do niej dyskretną transformatę Fouriera znajduje się w pozycji [25, str. 106]. Współczynnik $\frac{1}{N}$ występuje przy odwrotnej transformacie Fouriera.

wyznaczyć energię sygnału w poszczególnych pasmach częstotliwości wygodnie jest obliczyć widmową gęstość mocy (*power spectral density*). Do obliczenia widmowej gęstości mocy wykorzystany został wzór

$$S_{xx}[k] = F\{r_{xx}[m]\} = |X_k|, \tag{14}$$

gdzie F jest dyskretną transformatą Fouriera

$$r_{xx}[m] = E(x[n] \cdot x[n+m]) = \frac{1}{N}\sum_{n=1}^{N} x[n] \cdot x[n+m], \tag{15}$$

$r_{xx}[m]$ jest funkcją autokorelacji (dla stacjonarnego procesu losowego), $E(x[n] \cdot x[n+m])$ jest wartością oczekiwaną, N licznością próby. $S_{xx}[k]$ dla $k = 1,\ldots,N$ jest mocą sygnału o częstotliwości $k \cdot \frac{f_s}{N}$, gdzie f_s jest częstotliwością próbkowania.

3.3. Rekonstrukcja źródeł sygnałów EEG (problem odwrotny)

W tkance nerwowej, w odróżnieniu od obwodów elektrycznych, gdzie prąd jest rozłożony równomiernie w przekrojach poprzecznych elementu tego obwodu, stosowane jest pojęcie gęstości prądu $\vec{J}$, który pochodzi z wielu różnych jonów (jednostka $\frac{C}{(m^2 s)}$ lub $\frac{A}{m^2}$). W skali makroskopowej[7] tkanka nerwowa spełnia prawo Ohma $\vec{J} = \sigma\vec{E}$, co oznacza liniową zależność gęstości prądu $\vec{J}$ i natężenia pola elektrycznego (σ jest przewodnością elektryczną ośrodka – jednostka $\frac{1}{\Omega m}$). Natomiast na poziomie mikroskopowym błony są przewodnikiem nieliniowym (nie zachodzi prawo Ohma). Gdy pole elektryczne wewnątrz komórki wzrośnie powyżej wartości krytycznej, błony komórkowe są nieliniowym przewodnikiem. Zjawisko to umożliwia przenoszenie impulsów nerwowych. Źródła prądowe w błonach neuronalnych są generatorami sygnałów EEG[8].

3.3.1. Model głowy

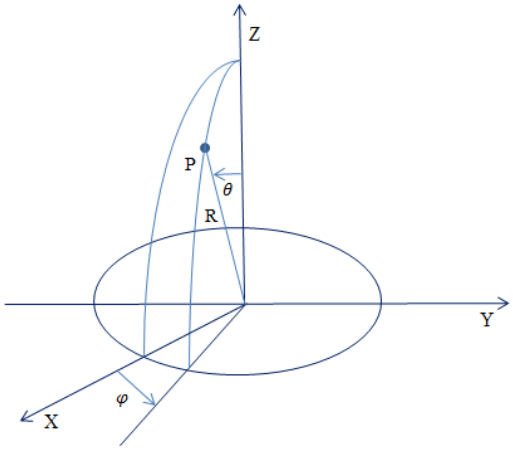
Do obliczeń przyjęto sferyczny model głowy o promieniu R=80 mm, kora mózgowa jest rozpięta na sferze o promieniu R=60mm (w realizowanych zadaniach myślowych promienie te można zmieniać w zależności od osoby badanej). Założono jednorodność i izotropowość ośrodka. Jak podaje źródło [10] wyniki obrazowania źródeł elektrycznych w modelach

[7] Źródło [82] str. 9 podaje, że jest to próbka materii, zawierająca co najmniej 10^{20} ładunków w ruchu.
[8] Na podstawie [82].

izotropowych i anizotropowych różnią się w sposób mało istotny, korzyści z różnicowania danego modelu głowy są zbyt małe w porównaniu do zakłócającego wpływu takich czynników jak: niedokładności segmentacji, niedokładności przewodności elektrycznej w różnych segmentach.

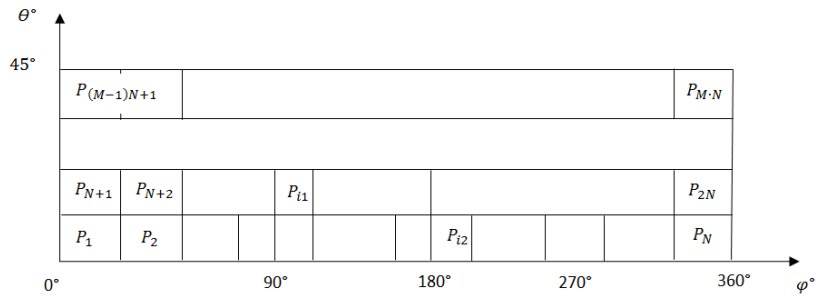
Pomiędzy powierzchnią głowy, a korą mózgową znajduje się skóra, tkanka miękka, czaszka, opona twarda. Powierzchnia kory mózgowej została podzielona na hipotetyczne elementy przestrzenne zwane wokselami. W każdym wokselu przyjmuje się wartość i kierunek gęstości prądu.

Liczba wszystkich wokseli jest zbyt duża do przeprowadzenia szybkiej klasyfikacji. Z tego powodu podzielono korę mózgową na obszary $P_1, \ldots, P_n$ .W każdym z tych obszarów zawarta jest pewna liczba wokseli. W niniejszej rozprawie n=36. W związku z przyjętym sferycznym modelem głowy w opisie wykorzystano współrzędne sferyczne (rys. 5).



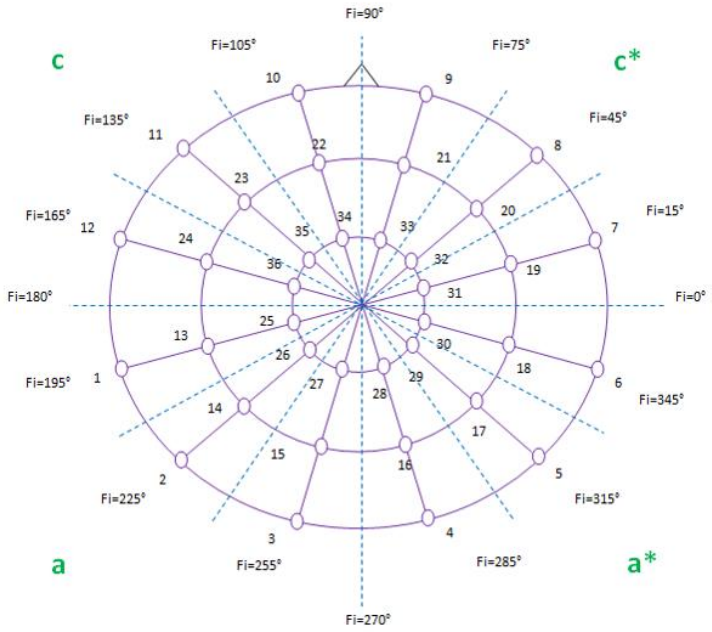
Rysunek 5 Schemat sferycznego modelu głowy

Dla n=36 w każdym z obszarów P_k , k=1,…,36 znajduje się 9 wokseli. Sposób numerowania obszarów przedstawia rys. 6. Dla N=12, M=3 szczegółowe rozmieszczenie środków obszarów znajduje się na rys. 7. Środki tych obszarów zostały ponumerowane $i_1, i_2 = 1, \ldots, n$, co zostało wykorzystane do policzenia wartości t-statystyki (wzór 58).



Rysunek 6 Numerowanie obszarów P_M

Gęstości natężenia prądu dla obszarów $P_1, \ldots, P_{M \cdot N}$ są średnimi gęstościami natężeń prądu wokseli należących do każdego z obszarów. W związku z zadaniem badawczym dotyczącym K2 i K3 dokonano podziału tych obszarów na te, które leżą w lewej półkuli mózgu i w prawej półkuli mózgu [47].



Rysunek 7 Rozmieszczenie środków obszarów $P_j, j = 1..36$ [9]

Na rys. 7 zaznaczone są współrzędne sferyczne $\varphi(Fi)$, $\theta(Theta)$, dla R= $const$.

Dla środków obszarów o numerach 1, …,12 $\theta = 37{,}5°$

Dla środków obszarów o numerach 13, …,24 $\theta = 22{,}5°$

Dla środków obszarów o numerach 25, …,36 $\theta = 7{,}5°$

Poszczególne oktanty[10] oznaczono:

(a) i (c) dla strony lewej

(a^*) i (c^*) dla strony prawej

$(a^*) \cup (c^*) = b$

3.3.2. Problem prosty

Zjawiska elektryczne i zjawiska magnetyczne w mózgu podlegają prawom Maxwella w wersji makroskopowej. W EEG wykorzystana jest na ogół wersja makroskopowa wzorów Maxwella, gdyż wersja mikroskopowa nie może być stosowana ze względu na

[9] Rysunek własny

[10] Oktanty – obszary w R^3 o stałych znakach współrzędnych x, y, z.

nieznajomość położenia ładunków elektrycznych w tkance nerwowej [82 str. 128]. Wykorzystana jest zatem wersja wzorów Maxwella:

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{16}$$

$$\nabla \times \vec{H} = \hat{J} + \frac{\partial \vec{D}}{\partial t} \tag{17}$$

$$\nabla \cdot \vec{D} = \rho \tag{18}$$

$$\nabla \cdot \vec{B} = 0 \tag{19}$$

gdzie $\vec{E}$ – natężenie pola elektrycznego, $\vec{D}$ – indukcja elektryczna, $\vec{H}$ – natężenie pola magnetycznego, $\vec{B}$ – indukcja magnetyczna, $\vec{J}$ – gęstość prądu, ρ – gęstość ładunku.

W tkance nerwowej

$$\vec{D} = \varepsilon \vec{E} \tag{20}$$

$$\vec{B} = \mu \vec{H} \tag{21}$$

W zał. 1 wyprowadzono równanie Poissona

$$\nabla \cdot \big(\sigma(r) \nabla \emptyset(r,t)\big) = -s(r,t) \tag{22}$$

Na podstawie [82 str. 168] można znaleźć przybliżone rozwiązanie dla każdej elektrody

$$\emptyset(r,t) = \frac{1}{4\pi\sigma} \sum_{n=1}^{N} \frac{I_n(t)}{R_n} \tag{23}$$

gdzie $I_n(t)$ – prąd $[\mu A]$ wypływający z N źródeł prądowych do objętościowego ośrodka (n=1, 2, …,N) o przewodności $\sigma \left[\frac{1}{\Omega cm}\right]$, $R_n = |r - r_n|$ – odległość [cm] między punktem pomiarowym a punktem źródła prądu.

Wówczas problem prosty można przedstawić w postaci układu równań liniowych (zał. 1)

$$KJ = \emptyset + \varepsilon \mathbf{1} \tag{24}$$

gdzie $\mathbf{1}$ jest macierzą jedynek o wymiarze $n_e \times 1$, ε – stała ($\varepsilon \epsilon \boldsymbol{R}$) wynikająca z fizycznej natury potencjału, który jest wyznaczony z dokładnością do stałej.

Przed przystąpieniem do rozwiązywania zagadnienia odwrotnego należy rozwiązać tzw. problem elektrody referencyjnej, polegający na wyznaczeniu stałej ε w równaniu (24), tak aby stała ε spełniała warunek:

$$\min_{\varepsilon} \|KJ - \emptyset - \varepsilon \cdot 1\| \tag{25}$$

W ślad za źródłem [88] stała ε spełniająca ten warunek jest postaci

$$\varepsilon = \frac{\mathbf{1}^T}{\mathbf{1}^T\mathbf{1}}(\emptyset - KJ) \tag{26}$$

Po wstawieniu ε do równania (24) i po przekształceniach

$$\emptyset = KJ + \frac{\mathbf{1} \cdot \mathbf{1}^T}{\mathbf{1}^T \cdot \mathbf{1}}(\emptyset - KJ) \tag{27}$$

a stąd

$$H\emptyset = HKJ \tag{28}$$

gdzie $H \coloneqq \left(I - \frac{\mathbf{1}\cdot\mathbf{1}^T}{\mathbf{1}^T\cdot\mathbf{1}}\right)$ jest macierzą o wymiarze $n_e \times n_e$, I – macierz jednostkowa ($n_e \times n_e$), $\mathbf{1}^T \cdot \mathbf{1} = [n_e]_{1\times 1}$. Macierz $\boldsymbol{H}$ jest tzw. macierzą centrującą (*average reference operator*). Następnie macierz $\emptyset$ i macierz K jest poddawana transformacji

$$\emptyset \coloneqq \boldsymbol{H}\emptyset\,, K \coloneqq \boldsymbol{H} \cdot K \tag{29}$$

Wówczas problem prosty można zapisać w postaci:

$$KJ = \emptyset \tag{30}$$

gdzie $\emptyset$ – jest macierzą o wymiarze $n_e \times 1$, n_e – jest liczbą elektrod rozmieszczonych na powierzchni głowy, $\emptyset = (\varphi_1, \dots, \varphi_m, \varphi_{n_e})^T$. Potencjały mierzone przez elektrody w stosunku do elektrody referencyjnej są rejestrowane w macierzy $\emptyset$. J jest macierzą o wymiarze $3n_v \times 1, n_v$ jest liczbą wokseli, $J = (j_1, \dots, j_\beta, \dots, j_{n_v})^T, j_\beta = (j_{\beta_x}, j_{\beta_y}, j_{\beta_z})^T, \beta = 1, \dots, n_v$, (x, y, z) – kartezjańskie współrzędne woksela, j_β – jest wektorem gęstości prądu w wokselu β, K – jest macierzą przejścia o wymiarze $n_e \times 3n_v$.

W problemie prostym $n_v \gg n_e$. Stąd wynikają trudności z rozwiązaniem równania (30), gdyż macierz K jest macierzą prostokątną.

3.3.3. Problem odwrotny

Problem odwrotny polega na wyznaczeniu przybliżenia $\hat{J}$ wektora J

$$\hat{J} = T\emptyset \tag{31}$$

gdzie $\hat{J}$ to macierz $(3_{n_v} \times 1)$. Postać operatora T zależy od stosowanej metody rozwiązywania równania $KJ = \emptyset$.

W przeprowadzonych badaniach przyjęto postać macierzy K [87] – i-ty wiersz macierzy K (i=1, …, n_e) jest postaci ($k_{i_1}, k_{i_2}, \dots, k_{i_{n_v}}$), gdzie $k_{i_j} = \left(k_{x_{i_j}}, k_{y_{i_j}}, k_{z_{i_j}}\right), j = 1,2, \dots, n_v$,

$K = [k_{ij}]$, gdzie $k_{ij} = \frac{1}{4\pi\sigma}\left(\frac{(e_i - v_j)}{\|e_i - v_j\|^3} - \frac{(e_R - v_j)}{\|e_R - v_j\|^3}\right)$, gdzie σ – przewodność, e_i – współrzędne kartezjańskie i – tej elektrody, e_R – elektroda referencyjna, v_j – współrzędne kartezjańskie j – tego woksela, $\|\bullet\|$ - norma euklidesowa.

Ze względu na to, że w układzie równań (30) macierz K jest prostokątna, przy czym liczba wierszy tej macierzy jest znacznie mniejsza niż liczba kolumn, to układ ten nie jest sprzeczny, ale jego rozwiązanie jest niejednoznaczne. Ponadto układ ten na ogół nie ma rozwiązania gdy wektor $\emptyset$ zastąpimy wektorem zaburzonym $\emptyset^{\delta}$, gdyż wartości $\emptyset$ pochodzą z odczytu z elektrod [91 str. 21]. Układ ten jest źle postawiony w sensie Hadamarda[11]. Najbardziej popularną metodą rozwiązywania układów liniowych równań algebraicznych źle postawionych jest poszukiwanie w przypadku układu równań (30) różnicy pomiędzy KJ i $\emptyset$. Jest to metoda najmniejszych kwadratów (L.S.).

Poszukuje się

$$\hat{J} = \operatorname{argmin}\{\|\emptyset - KJ\|^2, \quad J \in R^{3n_v}\} \tag{32}$$

gdzie

$$\|\emptyset - KJ\|^2 = \sum_{i=1}^{n_e}(\sum_{j=1}^{3n_v} K_{ij}J_j - \emptyset_i)^2 \tag{33}$$

Rozwiązanie $\hat{J}$ układu równań (30) w sensie najmniejszych kwadratów gdy rank $K \leq n_e \leq 3n_v$ nie jest jednoznaczne [91], ale spełnia warunek:

$$\left\|K\hat{J} - \emptyset\right\| \leq \|\emptyset - KJ\| \tag{34}$$

$\forall J \in R$. Wektor J^+ nazywany jest rozwiązaniem uogólnionym układu $KJ = \emptyset$, gdy $\|J^+\| \leq \left\|\hat{J}\right\|$, gdzie $\hat{J}$ jest dowolnym rozwiązaniem w sensie L.S.

Jeżeli oznaczy się

$$F(J) = \|KJ - \emptyset\| \tag{35}$$

to po zróżniczkowaniu $(F(J))^2$ po zmiennych $J_1, \ldots, J_{3n_v}$ i przyrównaniu pochodnych cząstkowych do zera (warunek konieczny ekstremum) uzyskuje się tzw. normalny układ równań

$$(K^T K)J = K^T\emptyset \tag{36}$$

Oznaczając przez J^+ rozwiązanie normalnego układu równań można wykazać, że jest wyznaczone jednoznacznie i spełnia nierówność [39]

[11] Układ dobrze postawiony w sensie Hadamarda - istnieje rozwiązanie. Rozwiązanie jest jedyne i jest stabilne.

$$(F(J))^2 \geq F(J^+)^2 \tag{37}$$

Oznaczając K^+ macierz pseudoodwrotną Moora-Penrosa (macierz K^+ ma właściwości $KK^+K = K$ oraz $K^+KK^+ = K^+$). Rozwiązanie układu normalnego równań

$$J^+ = (K^TK)^+K^T\emptyset \tag{38}$$

Na podstawie [61str. 40] $(K^TK)^+K^T = K^+$.

Zatem

$$T = K^+ \tag{39}$$

oraz

$$J^+ = K^+\emptyset \tag{40}$$

Korzystając z algebry liniowej:

1. Jeżeli założymy, że K jest macierzą rzeczywistą o wymiarze $n_e\ x\ 3n_v$, to K^TK jest macierzą symetryczną, nieujemnie określoną, której wartości własne są rzeczywiste, nieujemne
2. $\lambda_1 \geq \lambda_2, \ldots \geq \lambda_r > 0$ (λ_i – niezerowe wartości własne K^TK), to $r \leq \min(n_e, 3n_v)$
3. Jeżeli u_i i v_i są ortonormalnymi układami wektorów własnych odpowiednio K^TK i KK^T dla tej samej wartości własnej λ_i, to $\sigma_i = \sqrt{\lambda_i}$, dla $\lambda_i \neq 0$ (σ_i – są wartościami singularnymi (szczególnymi))
4. Macierz K ma rozkład singularny

$$K = VDU^T \tag{41}$$

gdzie U macierz o wymiarze $3n_v\ x\ 3n_v$, której kolumnami są wektory u_i , V macierz o wymiarze $n_e\ x\ n_e$, której kolumnami są wektory v_i , D jest macierzą o wymiarze $n_e\ x\ 3n_v$ postaci

$$D = \left[\begin{array}{cc|c} \sigma_1 & 0 & \\ 0 & \sigma_r & 0 \\ \hline & 0 & 0 \end{array}\right] \tag{42}$$

$(n_e - r)$ wierszy , $(3n_v - r)$ kolumn

Wynika stąd efektywna metoda wyznaczania macierzy pseudoodwrotnej Moorea-Penrosa, wykorzystująca rozkład singularny macierzy K – metoda SVD (*Singular Value Decomposition*).

Jeżeli skorzystamy z rozkładu singularnego macierzy K (41), to macierz pseudoodwrotna K^+ jest postaci

$$K^+ = UD^+V^T, \tag{43}$$

gdzie

$$D^{+} = \left[\begin{array}{cc|c} \sigma_1^{-1} & 0 & \\ 0 & \sigma_r^{-1} & 0 \\ \hline & 0 & 0 \end{array}\right] \tag{44}$$

$(3n_v - r)$wierszy , $(n_e - r)$kolumn

Rozwiązanie uogólnione zagadnienia odwrotnego jest postaci

$$J^{+} = K^{+}\emptyset \tag{45}$$

Istotne jest, że zadanie wyznaczenia rozwiązania uogólnionego J^{+} układu równań $KJ = \emptyset$ jest zadaniem dobrze postawionym w sensie Hadamarda. Autorka wyznaczyła rozwiązanie uogólnione w środowisku Matlab korzystając z komendy

$$J^{+} = pinv(K)\emptyset \tag{46}$$

Obliczony też został współczynnik uwarunkowania macierzy K. Skorzystano z wzoru

$$\kappa(K) = \|K^{+}\|\|K\| = \frac{max_{i=1,..,r}|\sigma_i|}{min_{i=1,..,r}|\sigma_i|} \tag{47}$$

Dla zaproponowanego przez autorkę sferycznego modelu głowy i sposobu wyznaczania elementów macierzy K uzyskano współczynnik uwarunkowania $\kappa(K) = 22$. W większości problemów technicznych współczynnik uwarunkowania jest mniejszy niż 1000 (na podstawie [85]).

Analizując problem wyznaczania źródeł sygnałów EEG autorka stosowała też algorytm LORETA (*Low-resolution electromagnetic tomography algorithm*) w wersji takiej jak przedstawiono w publikacjach [43], [44].

$\hat{J}$ jest wyznaczone w postaci ekstremum warunkowego [5] $\min_J(J^T W J)$, przy warunku $\emptyset = KJ$. W rozwiązaniu zagadnienia odwrotnego $\hat{J} = T\emptyset$ (wzór (31))

$$T = W^{-1}K^T[KW^{-1}K^T]^{+} \tag{48}$$

$$W = (\Omega \otimes I_3)B^T B(\Omega \otimes I_3), \tag{49}$$

gdzie $[KW^{-1}K^T]^{+}$ jest pseudoodwrotną macierzą Moora-Penrosa macierzy $[KW^{-1}K^T]$, W - jest dodatnio określoną macierzą wag,

$\otimes$ - oznacza iloczyn Kroneckera, zdefiniowany poniżej:

jeżeli macierz *A* ma wymiar *m* x *n* , a macierz *B* ma wymiar *p* x *q* , to macierz blokowa

$$A \otimes B = \begin{bmatrix} a_{11} \cdot B, \dots, & a_{1n} \cdot B \\ a_{m1} \cdot B, \dots, & a_{mn} \cdot B \end{bmatrix} \tag{50}$$

ma wymiar $(m \cdot p)x(n \cdot q)$, Ω - macierz diagonalna o wymiarze $n_v \times n_v$, $\Omega = [\omega_{jj}]$, j =1, …, n_v, $\omega_{jj} = \sqrt{\sum_{i=1}^{n_e} K_{ij}^T} \cdot K_{ij}$, $[K_{ij}] = K$, K_j – j-ta kolumna macierzy K , i = 1,…,n_e , I_3 - macierz jednostkowa o wymiarze (3×3), B - dyskretny przestrzenny operator Laplace'a

$$B = \frac{6}{d^2}(A - I_{3n_v}) \tag{51}$$

gdzie d – minimalna odległość między elektrodami, $A = A_0 \otimes I_3$, I_3 - macierz jednostkowa o wymiarze 3x3

$$A_0 = \frac{1}{2}(I_{n_v} + [diag\,(A_1 \cdot \mathbf{1}_{n_v})]^{-1} \cdot A_1, \qquad A_1 = [a_{\alpha\beta}] \tag{52}$$

to macierz o wymiarze $n_v\, x\, n_v$

$$a_{\alpha\beta} = \begin{cases} \frac{1}{6}\,, gdy\, \|v_\alpha - v_\beta\| = d \\ 0, w\, przeciwnym\, przypadku \end{cases} \tag{53}$$

I_{n_v} - macierz jednostkowa o wymiarze $n_v\, x\, n_v$

$$diag\,(A_1 \cdot \mathbf{1}_{n_v}) \tag{54}$$

to macierz diagonalna o wymiarze $n_v\, x\, n_v$, która ma na diagonali kolejne elementy macierzy $A_1 \cdot \mathbf{1}_{n_v}$, $\mathbf{1}_{n_v}$ - macierz jedynek o wymiarze $n_v x$ 1.

Czas obliczeń metodą Loreta był znacznie dłuższy w porównaniu z wcześniej opisaną metodą.

W celu znalezienia najlepszej metody wyznaczania rozwiązania zagadnienia odwrotnego autorka zastosowała różne metody do wyznaczenia przybliżonego wektora $\hat{J}$.

Do porównywania wykorzystano macierz H_0. Macierz T jest uogólnioną macierzą odwrotną macierzy przejścia K. Jeżeli oznaczy się iloczyn $KT = H_0$ i pomnoży równanie (31) stronami lewostronnie przez macierz K, to

$$K\hat{J} = H_0\emptyset \tag{55}$$

Dla dokładnego rozwiązania $\hat{J} = J$ macierz H_0 jest macierzą jednostkową o wymiarze $n_e \times n_e$. Z (55) wynika, że rozwiązanie zagadnienia odwrotnego $\hat{J}$ dane wzorem (31) tym lepiej spełnia równanie (30), im macierz H_0 mniej różni się od macierzy jednostkowej. Stosując różne metody wyznaczania zagadnienia odwrotnego (polegają one na wyznaczeniu różnych postaci macierzy odwrotnej T) można przez wyznaczenie macierzy H_0 porównywać te metody.

Do analizowania i selekcji różnych metod rozwiązywania zagadnienia odwrotnego (co dało różne postaci operatora T) wykorzystano również macierz R Backusa i Gilberta [5]. Macierz ta jest postaci

$$R = T \cdot K \tag{56}$$

gdzie macierz R ma wymiar $(3n_v) \times (3n_v)$. Po wstawieniu $\emptyset = KJ$ do równania (31) uzyska się $TKJ = \hat{J}$, a zatem

$$RJ = \hat{J} \tag{57}$$

Dla idealnego rozwiązania $\hat{J}$ macierz R jest macierzą jednostkową. Porównując macierz R dla różnych algorytmów szukania rozwiązania zagadnienia odwrotnego można wybrać taką metodę, aby R możliwie najmniej różniła się od macierzy jednostkowej.

Stwierdzono, że najlepszą metodą jest metoda najmniejszych kwadratów z dekompozycją SVD.

3.4. Wyodrębnianie cech i ich selekcja

Cechami które będą potrzebne do klasyfikacji obiektów K2 i K3 są wartości t-statystyki zapisane w macierzy $[t_{i_1 i_2}]$. Wyodrębniane cechy wyznaczane są z wzoru:

$$t_{i_1 i_2} = \frac{m_{i_1} - m_{i_2}}{\sqrt{\frac{\sigma_{i_1}^2}{N_{i_1}^2} + \frac{\sigma_{i_2}^2}{N_{i_2}^2}}} \tag{58}$$

gdzie m_{i_1}, m_{i_2} są to średnie, $\sigma_{i_1}^2, \sigma_{i_2}^2$ to wariacje gęstości natężenia prądu liczone dla wokseli w obszarach P_{i_1}, P_{i_2} zdefiniowanych w rozdziale 3.3.1., gdzie $i_1, i_2 - 1{,}2, \dots, n$ (w rozprawie przyjęto $n = 36$). Macierz cech $[t_{i_1 i_2}]$ jest antysymetryczna. Selekcja cech polega na wyborze pewnego podzbioru cech reprezentujących obiekty. Ten podzbiór wykorzystany był w procesie uczenia klasyfikatora, a jego korekta powinna zapewnić jak najlepsze wyniki klasyfikacji.

Najsilniejsze zmiany sygnału EEG związane z ruchem występują w korze ruchowej (pola Brodmanna 4 i 6). Do rozpoznawania intencji ruchu K2 i K3 wykorzystano zjawisko ERD/ERS [30]. Zjawisko to polega na tym, że największy spadek aktywności odpowiednich rytmów pojawia się na półkuli przeciwległej do strony tej części ciała, która jest związana z wyobrażeniem sobie ruchu. Ten spadek aktywności to ERD (*event-related desynchronization*). Po wykonaniu ruchu następuje wzrost aktywności nazywany EDS (*event-related synchronization*).

W związku z tym ze zbioru cech wybrane zostały te elementy, które reprezentują różne półkule mózgu (rys 7).

		Pomiar własny Emotiv Epoc12 Hz right																																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
		a	a	a	b	b	b	b	b	b	c	c	c	a	a	a	b	b	b	b	b	b	c	c	c	a	a	a	b	b	b	b	b	b	c	c	c
1	a	0	0,0702	-0,7593	-1,1196	-1,1181	-2,1208	-3,2174	-3,542	-3,2601	-2,3117	-3,56	-3,1239	-0,2423	0,483	-0,0969	-0,8736	-1,313	-2,3525	-2,4409	-2,5272	-3,2126	-2,8343	-5,2851	-2,9233	-0,2168	0,2389	-0,0183	-0,7285	-1,5134	-2,0425	-2,1232	-2,1423	-2,1677	-1,9751	-1,5782	-0,9844
2	a	-0,0702	0	-0,8058	-1,1427	-1,1326	-2,1368	-3,2365	-3,5546	-3,2685	-2,3258	-3,5829	-3,1765	-0,3164	0,4117	-0,152	-0,9144	-1,3456	-2,3858	-2,4654	-2,5453	-3,2332	-2,8622	-5,3259	-2,985	-0,3054	0,1387	-0,1145	-0,8148	-1,587	-2,1018	-2,1695	-2,182	-2,2062	-2,0155	-1,6266	-1,0504
3	a	0,7593	0,8058	0	-0,7035	-0,8649	-1,8032	-2,7638	-3,2682	-3,0979	-2,0325	-2,9548	-1,412	0,626	1,0874	0,594	-0,1732	-0,6878	-1,5538	-1,8843	-2,1421	-2,7121	-2,1359	-2,9871	-0,9717	0,716	0,9995	0,8519	0,4315	-0,0962	-0,6402	-1,013	-1,2043	-1,259	-1,0581	-0,5788	0,0965
4	b	1,1196	1,1427	0,7035	0	-0,3629	-1,1128	-1,737	-2,5766	-2,6966	-1,3997	-1,6668	0,0483	1,0522	1,2828	1,0422	0,5778	0,1814	-0,3494	-0,8271	-1,2993	-1,6134	-0,8795	-0,751	0,3136	1,0883	1,2113	1,1465	0,9597	0,7174	0,4334	0,1591	-0,0259	-0,0766	0,0644	0,4001	0,7955
5	b	1,1181	1,1326	0,8649	0,3629	0	-0,5774	-0,9798	-1,9024	-2,26	-0,8625	-0,8412	0,4506	1,0742	1,2206	1,0777	0,785	0,5174	0,163	-0,222	-0,6753	-0,8513	-0,2262	-0,0498	0,6168	1,0927	1,1666	1,1272	1,0133	0,8654	0,6907	0,5154	0,3904	0,3554	0,4496	0,6694	0,9159
6	b	2,1208	2,1368	1,8032	1,1128	0,5774	0	-0,3799	-1,4606	-1,9266	-0,3386	-0,1879	1,3669	2,0757	2,235	2,0564	1,6975	1,3654	0,9831	0,4981	-0,0698	-0,2233	0,5225	0,8009	1,5646	2,105	2,1915	2,1464	2,0164	1,846	1,6389	1,4203	1,2616	1,2183	1,3288	1,594	1,8939
7	b	3,2174	3,2365	2,7638	1,737	0,9798	0,3799	0	-1,2468	-1,7711	-0,0189	0,2559	2,2831	3,1707	3,3558	3,1005	2,6062	2,1506	1,712	1,059	0,3331	0,1925	1,124	1,5805	2,5473	3,2246	3,3382	3,2812	3,1166	2,8979	2,6217	2,313	2,0876	2,0278	2,1706	2,527	2,939
8	b	3,542	3,5546	3,2682	2,5766	1,9024	1,4606	1,2468	0	-0,8049	1,0803	1,502	2,9393	3,5083	3,6325	3,4806	3,1715	2,8726	2,5705	2,0917	1,4733	1,4198	2,1547	2,4853	3,1045	3,5359	3,606	3,57	3,466	3,3287	3,1574	2,9662	2,8223	2,7831	2,8775	3,1066	3,3615
9	b	3,2601	3,2685	3,0979	2,6966	2,26	1,9266	1,7711	0,8049	0	1,6411	1,947	2,8683	3,2359	3,3198	3,2299	3,0424	2,8601	2,6528	2,3515	1,9335	1,8925	2,3846	2,5735	2,971	3,2494	3,2934	3,2703	3,2035	3,1162	3,0097	2,895	2,8089	2,785	2,845	2,9867	3,1421
10	c	2,3117	2,3258	2,0325	1,3997	0,8625	0,3386	0,0189	-1,0803	-1,6411	0	0,2212	1,6482	2,2718	2,4123	2,256	1,9385	1,64	1,2998	0,8474	0,2933	0,1739	0,8793	1,1497	1,8226	2,297	2,3726	2,3331	2,2191	2,0696	1,8877	1,6939	1,5516	1,5125	1,6114	1,8475	2,112
11	c	3,56	3,5829	2,9548	1,6668	0,8412	0,1879	-0,2559	-1,502	-1,947	-0,2212	0	2,4104	3,5112	3,7274	3,3833	2,7457	2,1771	1,6706	0,9056	0,1209	-0,0549	0,9789	1,5461	2,7509	3,595	3,7421	3,6703	3,4634	3,1859	2,8282	2,4226	2,1327	2,0577	2,2328	2,6826	3,2199
12	c	3,1239	3,1765	1,412	-0,0483	-0,4506	-1,3669	-2,2831	-2,9393	-2,8683	-1,6482	-2,4104	0	3,1225	3,5367	2,378	1,0401	0,2355	-0,6898	-1,2478	-1,6654	-2,1991	-1,4439	-2,2325	0,9118	3,789	4,5317	4,2319	3,4094	2,3256	1,1264	0,2593	-0,1418	-0,2366	0,0397	0,819	2,2144
13	a	0,2423	0,3164	-0,626	-1,0522	-1,0742	-2,0757	-3,1707	-3,5083	-3,2359	-2,2718	-3,5112	-3,1225	0	0,7557	0,0836	-0,7574	-1,2239	-2,2792	-2,3802	-2,4792	-3,1637	-2,7736	-5,4272	-2,9762	0,0846	0,6489	0,3409	-0,4949	-1,3936	-1,9608	-2,0367	-2,0589	-2,086	-1,8866	-1,4684	-0,8173
14	a	-0,483	-0,4117	-1,0874	-1,2828	-1,2206	-2,235	-3,3558	-3,6325	-3,3198	-2,4123	-3,7274	-3,5367	-0,7557	0	-0,4788	-1,1627	-1,5457	-2,5952	-2,6179	-2,6566	-3,3622	-3,0372	-5,646	-3,4163	-0,8261	-0,4328	-0,6721	-1,3367	-2,0536	-2,4846	-2,464	-2,4323	-2,4482	-2,2687	-1,9283	-1,4551
15	a	0,0969	0,152	-0,594	-1,0422	-1,0777	-2,0564	-3,1005	-3,4806	-3,2299	-2,256	-3,3833	-2,378	-0,0836	0,4788	0	-0,7221	-1,1702	-2,1174	-2,2976	-2,4396	-3,0792	-2,631	-4,144	-2,0089	-0,0403	0,2771	0,103	-0,3898	-0,9819	-1,5081	-1,75	-1,8515	-1,8895	-1,6985	-1,2769	-0,671
16	b	0,8736	0,9144	0,1732	-0,5778	-0,785	-1,6975	-2,6062	-3,1715	-3,0424	-1,9385	-2,7457	-1,0401	0,7574	1,1627	0,7221	0	-0,5115	-1,3157	-1,7031	-2,0119	-2,5393	-1,9106	-2,4406	-0,6192	0,8363	1,0777	0,9521	0,5931	0,1365	-0,3572	-0,738	-0,9487	-1,0069	-0,812	-0,3426	0,2917
17	b	1,313	1,3456	0,6878	-0,1814	-0,5174	-1,3654	-2,1506	-2,8726	-2,8601	-1,64	-2,1771	-0,2355	1,2239	1,5457	1,1702	0,5115	0	-0,6911	-1,1896	-1,6187	-2,0486	-1,3085	-1,382	0,1444	1,2949	1,4853	1,3873	1,1059	0,7415	0,3205	-0,061	-0,2973	-0,3605	-0,1769	0,273	0,8455
18	b	2,3525	2,3858	1,5538	0,3494	-0,163	-0,9831	-1,712	-2,5705	-2,6528	-1,2998	-1,6706	0,6898	2,2792	2,5952	2,1174	1,3157	0,6911	0	-0,6431	-1,1969	-1,5811	-0,7087	-0,5449	1,1463	2,401	2,6229	2,5144	2,2034	1,7922	1,2905	0,7951	0,4828	0,4031	0,6123	1,1519	1,8573
19	b	2,4409	2,4654	1,8843	0,8271	0,222	-0,4981	-1,059	-2,0917	-2,3515	-0,8474	-0,9056	1,2478	2,3802	2,6179	2,2976	1,7031	1,1896	0,6431	0	-0,6347	-0,8973	0,0052	0,3547	1,5749	2,4484	2,5955	2,5215	2,3085	2,027	1,6792	1,3119	1,0581	0,9913	1,1592	1,5794	2,0844
20	b	2,5272	2,5453	2,1421	1,2993	0,6753	0,0698	-0,3331	-1,4733	-1,9335	-0,2933	-0,1209	1,6654	2,4792	2,6566	2,4396	2,0119	1,6187	1,1969	0,6347	0	-0,162	0,6711	1,0181	1,8981	2,5194	2,6208	2,5688	2,4186	2,2208	1,9769	1,7139	1,5233	1,4719	1,5997	1,9116	2,2689
21	b	3,2126	3,2332	2,7121	1,6134	0,8513	0,2233	-0,1925	-1,4198	-1,8925	-0,1739	0,0549	2,1991	3,1637	3,3622	3,0792	2,5393	2,0486	1,5811	0,8973	0,162	0	0,9594	1,4371	2,4881	3,2257	3,3507	3,2883	3,1087	2,8696	2,5665	2,2281	1,9835	1,9191	2,0727	2,4592	2,911
22	c	2,8343	2,8622	2,1359	0,8795	0,2262	-0,5225	-1,124	-2,1547	-2,3846	-0,8793	-0,9789	1,4439	2,7736	3,0372	2,631	1,9106	1,3085	0,7087	-0,0052	-0,6711	-0,9594	0	0,4043	1,8415	2,8737	3,0543	2,9659	2,7118	2,3733	1,9478	1,4944	1,1883	1,1098	1,3033	1,8016	2,4216
23	c	5,2851	5,3259	2,9871	0,751	0,0498	-0,8009	-1,5805	-2,4853	-2,5735	-1,1497	-1,5461	2,2325	5,4272	5,646	4,144	2,4406	1,382	0,5449	-0,3547	-1,0181	-1,4371	-0,4043	0	3,5598	6,4292	7,287	6,9793	6,1341	4,9281	3,3622	1,9928	1,3303	1,1873	1,5202	2,5552	4,4428
24	c	2,9233	2,985	0,9717	-0,3136	-0,6168	-1,5646	-2,5473	-3,1045	-2,971	-1,8226	-2,7509	-0,9118	2,9762	3,4163	2,0089	0,6192	-0,1444	-1,1463	-1,5749	-1,8981	-2,4881	-1,8415	-3,5598	0	4,2723	5,9176	5,3144	3,8035	2,0066	0,4544	-0,349	-0,6712	-0,7514	-0,4802	0,2708	1,8007
25	a	0,2168	0,3054	-0,716	-1,0883	-1,0927	-2,105	-3,2246	-3,5359	-3,2494	-2,297	-3,595	-3,789	-0,0846	0,8261	0,0403	-0,8363	-1,2949	-2,401	-2,4484	-2,5194	-3,2257	-2,8737	-6,4292	-4,2723	0	1,0261	0,4419	-0,9506	-2,0863	-2,4323	-2,2723	-2,2221	-2,2393	-2,0419	-1,6553	-1,0672
26	a	-0,2389	-0,1387	-0,9995	-1,2113	-1,1666	-2,1915	-3,3382	-3,606	-3,2934	-2,3726	-3,7421	-4,5317	-0,6489	0,4328	-0,2771	-1,0777	-1,4853	-2,6229	-2,5955	-2,6208	-3,3507	-3,0543	-7,287	-5,9176	-1,0261	0	-0,9584	-2,6448	-3,347	-3,1058	-2,6448	-2,5053	-2,5089	-2,3245	-2,0124	-1,6502
27	a	0,0183	0,1145	-0,8519	-1,1465	-1,1272	-2,1464	-3,2812	-3,57	-3,2703	-2,3331	-3,6703	-4,2319	-0,3409	0,6721	-0,103	-0,9521	-1,3873	-2,5144	-2,5215	-2,5688	-3,2883	-2,9659	-6,9793	-5,3144	-0,4419	0,9584	0	-1,8105	-2,8056	-2,8102	-2,4698	-2,3686	-2,3783	-2,187	-1,8385	-1,3642
28	b	0,7285	0,8148	-0,4315	-0,9597	-1,0133	-2,0164	-3,1166	-3,466	-3,2035	-2,2191	-3,4634	-3,4094	0,4949	1,3367	0,3898	-0,5931	-1,1059	-2,2034	-2,3085	-2,4186	-3,1087	-2,7118	-6,1341	-3,8035	0,9506	2,6448	1,8105	0	-1,4472	-2,0004	-1,9747	-1,9788	-2,0055	-1,7949	-1,3463	-0,58
29	b	1,5134	1,587	0,0962	-0,7174	-0,8654	-1,846	-2,8979	-3,3287	-3,1162	-2,0696	-3,1859	-2,3256	1,3936	2,0536	0,9819	-0,1365	-0,7415	-1,7922	-2,027	-2,2208	-2,8696	-2,3733	-4,9281	-2,0066	0,2863	3,347	2,8056	1,4472	0	-0,9891	-1,3295	-1,4669	-1,5154	-1,2824	-0,7169	0,338
30	b	2,0425	2,1018	0,6402	-0,4334	-0,6907	-1,6389	-2,6217	-3,1574	-3,0097	-1,8877	-2,8282	-1,1264	1,9608	2,4846	1,5081	0,3572	-0,3205	-1,2905	-1,6792	-1,9769	-2,5665	-1,9478	-3,3622	-0,4544	2,4323	3,1058	2,8102	2,0004	0,9891	0	-0,5964	-0,8624	-0,9326	-0,6851	-0,0373	1,0939
31	b	2,1232	2,1695	1,013	-0,1591	-0,5154	-1,4203	-2,313	-2,9662	-2,895	-1,6939	-2,4226	-0,2593	2,0367	2,464	1,75	0,738	0,061	-0,7951	-1,3119	-1,7139	-2,2281	-1,4944	-1,9928	0,349	2,2723	2,6448	2,4698	1,9747	1,3295	0,5964	0	-0,3168	-0,3967	-0,1582	0,4685	1,4129
32	b	2,1423	2,182	1,2043	0,0259	-0,3904	-1,2616	-2,0876	-2,8223	-2,8089	-1,5516	-2,1327	0,1418	2,0589	2,4323	1,8515	0,9487	0,2973	-0,4828	-1,0581	-1,5233	-1,9835	-1,1883	-1,3303	0,6712	2,2221	2,5053	2,3686	1,9788	1,4669	0,8624	0,3168	0	-0,0802	0,1447	0,7296	1,5457
33	b	2,1677	2,2062	1,259	0,0766	-0,3554	-1,2183	-2,0278	-2,7831	-2,785	-1,5125	-2,0577	0,2366	2,086	2,4482	1,8895	1,0069	0,3605	-0,4031	-0,9913	-1,4719	-1,9191	-1,1098	-1,1873	0,7514	2,2393	2,5089	2,3783	2,0055	1,5154	0,9326	0,3967	0,0802	0	0,2221	0,7985	1,5919
34	c	1,9751	2,0155	1,0581	-0,0644	-0,4496	-1,3288	-2,1706	-2,8775	-2,845	-1,6114	-2,2328	-0,0397	1,8866	2,2687	1,6985	0,812	0,1769	-0,6123	-1,1592	-1,5997	-2,0727	-1,3033	-1,5202	0,4802	2,0419	2,3245	2,187	1,7949	1,2824	0,6851	0,1582	-0,1447	-0,2221	0	0,5727	1,3727
35	c	1,5782	1,6266	0,5788	-0,4001	-0,6694	-1,594	-2,527	-3,1066	-2,9867	-1,8475	-2,6826	-0,819	1,4684	1,9283	1,2769	0,3426	-0,273	-1,1519	-1,5794	-1,9116	-2,4592	-1,8016	-2,5552	-0,2708	1,6553	2,0124	1,8385	1,3463	0,7169	0,0373	-0,4685	-0,7296	-0,7985	-0,5727	0	0,8561
36	c	0,9844	1,0504	-0,0965	-0,7955	-0,9159	-1,8939	-2,939	-3,3615	-3,1421	-2,112	-3,2199	-2,2144	0,8173	1,4551	0,671	-0,2917	-0,8455	-1,8573	-2,0844	-2,2689	-2,911	-2,4216	-4,4428	-1,8007	1,0672	1,6502	1,3642	0,58	-0,338	-1,0939	-1,4129	-1,5457	-1,5919	-1,3727	-0,8561	0

Rysunek 8 Macierz wartości t-statystyk

Na rys. 8 przedstawiono macierz cech $[t_{i_1 i_2}]$, gdzie zaznaczono kolorami (niebieskim i zielonym) wynik selekcji cech istotnych z punktu widzenia klasyfikacji, tj. takich, które łączą lewą i prawą półkulę mózgu. Macierz ta pokazuje obszary dla których liczone są wartości t-statystyki. Podział na obszary a, b, c przedstawiono na rys. 9. Połączeniu obszaru a (lewa półkula) i b (prawa półkula) odpowiada kolor niebieski w macierzy wartości t-statystyk, a połaczeniu obszaru c (lewa półkula) i b (prawa półkula) odpowiada kolor zielony w macierzy $[t_{i_1 i_2}]$. Metodę stosowaną do dalszej selekcji cech zaliczyć można do tzw. metod opakowanych, tzn. ocena podzbioru cech jest dokonywana przy użyciu konkretnego modelu klasyfikacji [16]. Miarą jakości podzbioru cech jest efektywność klasyfikatora.

Na etapie nauki klasyfikatora cechy zaznaczone w macierzy $[t_{i_1 i_2}]$, jako te które są wyselekcjonowane, można modyfikować. Jest to przykład metody opakowanej selekcji cech BDS (*BiDirectional Search*). Metoda BDS polega na równoległym stosowaniu algorytmów SFS (*Sequential Forward Selection*) i SBS (*Sequential Backward Selection*). Algorytm SFS to algorytm sekwencyjnego przeszukiwania w przód. Algorytm SBS, to algorytm sekwencyjnego przeszukiwania w tył. Metoda BDS wymaga wielokrotnego uruchamiania klasyfikatora (celem jest poprawa wyników klasyfikacji). Algorytm BDS opisany jest w [16, str. 51].

3.5. Klasyfikacja

Do rozpoznawania obiektów K2 i K3 służy klasyfikator tzn. algorytm, który na podstawie wyselekcjonowanego zbioru cech ma za zadanie rozpoznać obiekt [28].

Wyróżnia się, w zależności od tego czy wykorzystano do klasyfikacji zbiór uczący, klasyfikację nadzorowaną i nienadzorowaną. W przypadku klasyfikacji nadzorowanej, do klasyfikacji wykorzystuje się zbiór uczący. W przypadku klasyfikacji nienadzorowanej nie dysponuje się zbiorem uczącym. W rozprawie stosowana jest klasyfikacja nadzorowana do nielicznego zbioru uczącego. W tym przypadku klasyfikacja polega na tym, że aby rozdzielić obiekty wykorzystano maksymalne podobieństwo cech (maksymalna miara podobieństwa).

Ze względu na to, że do klasyfikacji wykorzystano zjawisko ERD/ERS (nie jest więc potrzebne szukanie granicy między podzbiorami cech), to autorka nie wykorzystywała klasyfikatorów SVM, LDA ani innych powszechnie używanych klasyfikatorów.

Algorytm prezentowany w niniejszej pracy oparty jest na teorii grafów [46]. Na początku skonstruowano pewien graf ważony. Oto kolejne etapy konstrukcji:

1. w związku ze zjawiskiem desynchronizacji w przypadku wyobrażania sobie ruchu elementy macierzy $[t_{i_1 i_2}]$ mają następujące własności:

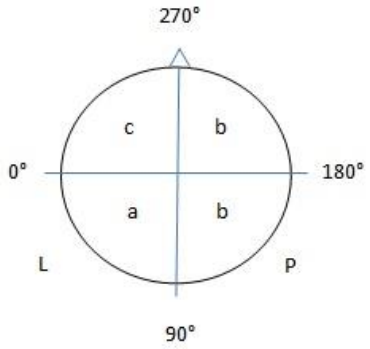
– dla obiektu K3

$$t_{i_1 i_2} < 0, gdy \begin{cases} i_1 \in a \\ i_2 \in b \end{cases} oraz, gdy \begin{cases} i_1 \in c \\ i_2 \in b \end{cases} \quad (59)$$

– dla obiektu K2

$$t_{i_1 i_2} > 0, gdy \begin{cases} i_1 \in a \\ i_2 \in b \end{cases} oraz, gdy \begin{cases} i_1 \in c \\ i_2 \in b \end{cases} \quad (60)$$

gdzie a, b, c, (podział dokładniejszy na obszary na rys. 7) to obszary kory mózgowej w prawej i lewej półkuli mózgu (rys. 9).



Rysunek 9 Obszary leżące w lewej i prawej półkuli mózgu opisane przy pomocy kąta φ (rysunek własny)

Uwzględnić należy tylko te wartości $[t_{i_1 i_2}]$, gdzie wartości średnie m_{i_1}, m_{i_2} w liczniku wzoru (58) są istotnie zróżnicowane, tzn. $t_{i_1 i_2}$ należą do zbioru krytycznego T_α dla przyjętego poziomu istotności $\alpha = 0{,}05$.

2. Zbudowano macierz podobieństwa typu obiekt-obiekt $[m_{i_1 i_2}]$.

W przypadku obiektu K3

$$m_{i_1 i_2} := \begin{cases} -t_{i_1 i_2}, & gdy\ t_{i_1 i_2} \in T_\alpha \wedge (i_1, i_2) \in H \\ 0, & p.p. \end{cases} \quad (61)$$

W przypadku obiektu K2

$$m_{i_1 i_2} := \begin{cases} t_{i_1 i_2}, & gdy\ t_{i_1 i_2} \in T_\alpha \wedge (i_1, i_2) \in H \\ 0, & p.p. \end{cases} \quad (62)$$

3. Macierz $[m_{i_1 i_2}]$ jest macierzą sąsiedztwa pewnego grafu ważonego $G(X, R)$, X jest zbiorem wierzchołków tego grafu, $X = \{P_1, P_2, \dots, P_n\}$, gdzie P_i – są obszarami na korze mózgowej, w niniejszej pracy $n = 36$. R jest zbiorem ważonych krawędzi grafu,

$(P_{i_1}, P_{i_2}) \in R \Leftrightarrow m_{i_1 i_2} \neq 0$ i waga tej krawędzi jest $t_{i_1 i_2}$ [45]. Miarą bliskości (podobieństwa) obszarów P_{i_1} , P_{i_2} jest $m_{i_1 i_2}^{-1}$, gdy $m_{i_1 i_2} \neq 0$.

4. Aby uzyskać szybką klasyfikację wyznaczono drzewo rozpinające $G(X, R')$ grafu $G(X, R)$, gdzie R' z definicji – krawędzie tego drzewa, które są takie, że nie istnieje inne drzewo rozpinające grafu G , którego suma modułów wag krawędzi jest większa niż suma modułów wag krawędzi grafu $G(X, R')$.

5. Jako algorytm służący do budowy drzewa rozpinającego $G(X, R')$ wykorzystano algorytm Gowera.

Opis algorytmu Gowera:

Dany jest graf G(X,R), gdzie X – zbiór wierzchołków grafu G, R – zbiór krawędzi grafu G, oraz funkcja wag krawędzi w: R→<0, +∞). Zakładamy, że graf G jest spójny.

Przy wyznaczaniu drzewa rozpinającego G(X, R') stosujemy tzw. strategię zachłanną, która polega na tym, że w każdym kroku, gdy dokonujemy jednego z możliwych wyborów, algorytm dokonuje wyboru najlepszego w danej chwili. Wprowadzono oznaczenia: E – pewien zbiór krawędzi, V – pewien zbiór wierzchołków.

Algorytm Gower (G, w)

1. Wybierz $x_0 \in X$, V $\leftarrow \{x_0\}$, E $\leftarrow \emptyset$
2. **while** V≠X
3. **do** Znajdź krawędź e = (x,y) o końcach x∈V, y∈ $X \setminus V$, tak aby po dołączeniu tej krawędzi do E uzyskać drzewo o największej sumie wag krawędzi
4. E ← E ∪ $\{e\}$
5. V← V ∪ $\{y\}$
6. **return** G(V, E)

Algorytm został opracowany na podstawie schematu blokowego w [23]. Jeżeli w grafie jest więcej niż jedna spójna składowa, to algorytm należy powtórzyć dla każdej składowej.

Rozpoznanie intencji ruchu: Jeżeli suma wag krawędzi drzewa rozpinającego jest dodatnia, to algorytm klasyfikacji rozpoznaje intencje ruchu lewą ręką (obiekt K2). Jeżeli suma wag krawędzi drzewa rozpinającego jest ujemna, to algorytm klasyfikacji rozpoznaje intencję ruchu prawą ręką (obiekt K3). Drzewo rozpinające jest klasyfikatorem. Algorytm wyznaczania drzewa rozpinającego nie jest jednoznaczny, ale jak udowodniono w [23] niejednoznaczność ta nie wpływa na wynik klasyfikacji.

Analizując drzewo rozpinające można wyznaczyć tylko te krawędzie tego drzewa, które zapewnią większą bliskość (podobieństwo) wybieranych wierzchołków grafu [27]. Uzyskać

to można wybierając w drzewie rozpinającym tylko takie krawędzie, których wagi są większe od przyjętego progu d_i. Dobierane były takie krawędzie, że $d_i = 0{,}9$, $d_i = 0{,}8$, $d_i = 0{,}7$ tzn. takie krawędzie, których moduł wagi jest większy od iloczynu: $d_i * (maksymalny\ moduł\ wagi - minimalny\ moduł\ wagi)$.

3.6. Przekazanie wyniku klasyfikacji do urządzenia zewnętrznego

Wynik poprawnej klasyfikacji jest wyświetlany w oknie dialogowym programu sterującego opracowanego przez autorkę. Może również zostać przekazany do urządzenia zewnętrznego w postaci impulsu sterującego, co autorka wykazała konstruując proste urządzenie zewnętrzne. Opis urządzenia zewnętrznego prezentującego możliwości sterowania za pomocą aktywności myślowych został zawarty w rozdziale 5. Po poprawnej klasyfikacji można zarejestrować wyniki klasyfikacji w postaci raportu (rozdział 5 – F1).

4. Opis uzyskanych wyników

Do opracowania metody klasyfikacji źródeł sygnałów elektrycznych emitowanych przez korę mózgową człowieka w celu jej zastosowania do sterowania wykorzystane zostały dane własne z headsetu Emotiv Epoc a do testowania poprawności działania opracowanego algorytmu autorka wykorzystała dane Idiap z pakietu Data Set V. Dane Data Set V zostały pobrane z udostępnionej przez Idiap bazy sygnałów przygotowanej do konkursu BCI Competition III. Ze zbiorów danych Idiap autorka wybrała dane typu: zagadnienia różnego typu, klasyfikacja ciągłego EEG bez wersji próbnej (*multi-class problems, classification of continuous EEG without trial structure*).

4.1. Dane z bazy Idiap

Dane EEG z bazy Idiap pochodzące z pakietu Data Set V[12] o następujących parametrach: 3 rodzaje aktywności, 32 kanały EEG (DC-256Hz) rozmieszczone zgodnie ze standardem 10-20, częstotliwość próbkowania 512Hz. Pakiet danych Idiap zawiera ciągły sygnał EEG (dane surowe *Raw*) oraz dane przetworzone (*preprocessed*). Dane zostały podzielone na treningowe i testowe. Dane treningowe zawierają znaczniki/etykiety.

Dane Idiap zawierały odczyty sygnałów EEG dotyczące trzech testowanych osób (sub1, sub2, sub3), realizujących co ok. 15 sek. kolejne zadania myślowe zlecane przez operatora. Osoby te wyobrażały sobie ruch K2 i K3 oraz wypowiadały w myślach słowa zaczynające się na literę wybraną losowo przez operatora – zadanie K7. Osoby te nie wykonywały fizycznych ruchów. Na potrzeby swojej pracy autorka wykorzystała dwie aktywności myślowe: K2 i K3.

Przetwarzanie danych Idiap polegało na przeprowadzeniu przestrzennej filtracji Laplace'a, a następnie na policzeniu Widmowej Gęstości Mocy (*Power Spectral Density - PSD*) w zakresie częstotliwości 8-30Hz dla wybranych 8 elektrod. Dane surowe i przetworzone zawierały po trzy zestawy danych treningowych (dla każdej testowanej osoby), w których do wartości sygnału była przyporządkowana odpowiadająca jej aktywność ruchowa.

[12] Więcej informacji na http://www.bbci.de/competition/iii/desc_V.html Przykład użycia danych Idiap: [55].

4.2. Dane z headsetu Emotiv Epoc

Do klasyfikacji wykorzystane zostały zestawy danych uzyskane podczas testów przeprowadzonych przez autorkę o następujących parametrach: dwa rodzaje aktywności (K3, K2), tj. osoba uczestnicząca w badaniu wyobrażała sobie ruch prawą i lewą ręką. Osoba badana koncentrowała się na myśleniu o wykonaniu ruchu ręką ale nie wykonywała fizycznie tego ruchu. Zbieranie danych rozpoczynało się po usłyszeniu sygnału dźwiękowego i kończyło po usłyszeniu kolejnego sygnału kończącego pomiar. Czas trwania pomiaru określał liczbę danych potrzebnych do zebrania 12 przedziałów czasowych.

4.3. Opis wyników badawczych dla danych Idiap

Próbki z bazy Idiap zawierały dane w pewnych przedziałach czasowych dla każdego rodzaju aktywności. Testowanie własnego algorytmu na danych Idiap polegało na porównaniu uzyskanych wyników dotyczących klasyfikacji sygnałów ze znacznikami danych zawartymi w bazie Idiap dla danych treninowych, gdzie wiadomo jakiej intencji ruchu one dotyczyły.

Podczas testowania na danych Idiap opracowanego przez siebie algorytmu autorka uzyskała w około 70%[13] testów poprawną klasyfikację.

Dla Idiap autorka wyznaczyła źródła sygnałów elektrycznych na korze mózgowej, a do wskazania odpowiednich części mózgu z których pochodziły sygnały wykorzystany został atlas mózgu[14] w celu wybierania obszarów ruchowych mózgu. Ten sam proces został wykonany dla danych Epoc.

4.4. Opis wyników dla danych Epoc

Po przetestowaniu poprawności działania algorytmu klasyfikującego na danych Idiap, autorka dokonała klasyfikacji intencji użytkownika na danych własnych zarejestrowanych z użyciem headsetu Emotive Epoc.

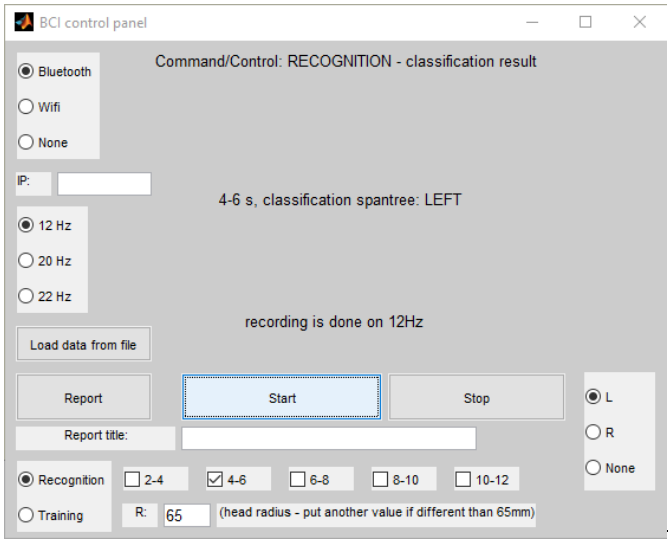
[13] Wyniki BCI Competition – http://www.bbci.de/competition/iii/results/ (stan na dzień 02.10.2019). Wynik na poziomie 70% jest zbieżny z najlepszymi wynikami przedstawionymi przez uczestników konkursu Competition III (zał. 3)

[14] Atlas mózgu Talairach - http://www.talairach.org/ (stan na dzień 14.12.2019)

Rejestrowanie danych uczących i danych testujących realizowane było z perspektywy pierwszej osoby[15] i następowało po usłyszeniu bodźca dźwiękowego. Zmiany sygnałów wysyłanych przez mózg rozpatrywane były w przedziałach czasowych 2-4, 4-6, 6-8., 8-10, 10-12. [50]. Po rejestracji danych uczących wybrano przedział czasowy w którym występuje największa poprawność wyników klasyfikacji.

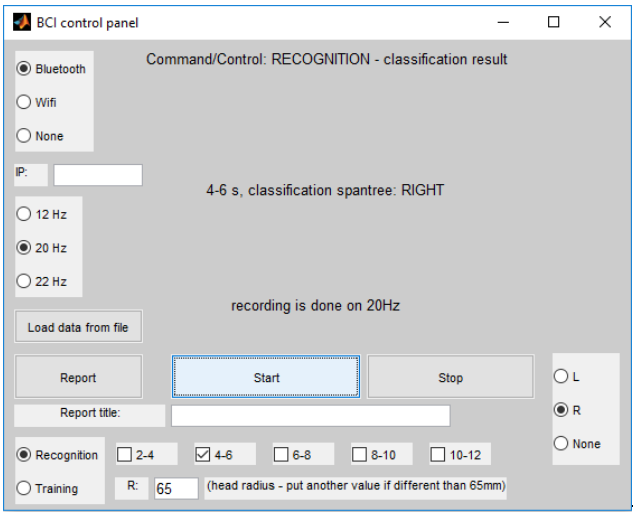
Wykonano pomiary dla częstotliwości 12 Hz, 20 Hz oraz 22 Hz. Przed przeprowadzeniem testów dokonano nauki klasyfikatora oraz wykonano testy wstępne dla osoby badanej. Dla danej osoby należało sprawdzić na której częstotliwości i dla którego przedziału czasowego uzyskiwane są najlepsze wyniki klasyfikacji. Wartości częstotliwości i przedziały czasowe mogą być różne w zależności od tego czy jest to K2, czy K3.

Wyniki klasyfikacji sygnału były wyświetlane w oknie programu w czasie wykonywania zadanych aktywności myślowych K2 i K3. Dzięki temu można było śledzić uzyskane wyniki w czasie rzeczywistym. Wyniki klasyfikacji wyświetlane są w oknie programu w sposób ciągły, do momentu zatrzymania programu. Rys. 10 i 11 przedstawiają przykładowe prawidłowe wyniki klasyfikacji sygnału dla wyobrażenia ruchu K2 przy częstotliwości 12 Hz i K3 przy częstotliwości 20 Hz [49].



Rysunek 10 Okno programu wyświetlające przykładowy wynik klasyfikacji dla lewej ręki

[15] W tym sposobie realizacji wyobrażenia ruchu osoba testowana wzbudza w sobie chęć wykonania ruchu ręką, ale nie wykonuje fizycznego ruchu.



Rysunek 11 Okno programu wyświetlające przykładowy wynik klasyfikacji dla prawej ręki

Przeprowadzono klasyfikację intencji ruchu dla danych uczących i testujących. Poniższe tabele prezentują wyniki dla danych uczących i testujących. Dla danych uczących przedstawiono wyniki pomiarów dla czterech przedziałów czasowych, dla częstotliwości 12Hz i 20Hz. Dla każdego przedziału czasowego i częstotliwości przeprowadzono łącznie 160 pomiarów. Przeprowadzono też pomiary dla 22Hz i stwierdzono, że wyniki niewiele różniły się od wyników dla 20Hz. Dla danych testujących przeprowadzono łącznie 490 pomiarów, dla czterech przedziałów czasowych dla częstotliwości 12Hz i 20Hz. Wyniki klasyfikacji uzyskano dla częstotliwości 12 Hz oraz 20 Hz i 22 Hz. Podobnie jak dla danych uczących, dla danych testujących przedstawiono wyniki klasyfikacji w tabelach dla 12 Hz i 20 Hz. Nie przedstawiono wyników dla 22 Hz, gdyż również mało różniły się od wyników uzyskanych dla 20 Hz. W każdym przypadku (dla danych uczących i danych testujących) autorka przebadała wpływ przedziałów czasowych na wynik klasyfikacji. Z przedstawionych macierzy pomyłek można odczytać w ilu przypadkach algorytm poprawnie zaklasyfikował dane, a w ilu się pomylił, tzn. np. algorytm wybrał intencję ruchu K3 zamiast K2 lub odwrotnie. Dodatkowo w każdym przypadku policzono błąd całkowity. Testowano trzy osoby a uzyskane wyniki uśredniono (średnia arytmetyczna ważona).

Tabele od 1 do 4 zawierają macierze pomyłek dla danych uczących w trybie nauki klasyfikatora dla przedziałów czasowych 2-4, 4-6, 6-8, 8-10 - dla częstotliwości 12 Hz.

Tabela 1 Wyniki klasyfikatora dla danych uczących (przedział czasowy 2-4 częstotliwość 12 Hz)

Całkowity błąd dla danych uczących (przedział czasowy 2-4 częstotliwość 12 Hz) 0,2875

2 - 4	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	42	18
K3	28	72

Tabela 2 Wyniki klasyfikatora dla danych uczących (przedział czasowy 4-6 częstotliwość 12 Hz)

Całkowity błąd dla danych uczących (przedział czasowy 4-6 częstotliwość 12 Hz) 0,1875

4 - 6	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	48	12
K3	18	82

Tabela 3 Wyniki klasyfikatora dla danych uczących (przedział czasowy 6-8 częstotliwość 12 Hz)

Całkowity błąd dla danych uczących (przedział czasowy 6-8 częstotliwość 12 Hz) 0,3175

6 - 8	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	40	20
K3	30	70

Tabela 4 Wyniki klasyfikatora dla danych uczących (przedział czasowy 8-10 częstotliwość 12 Hz)

Całkowity błąd dla danych uczących (przedział czasowy 8-10 częstotliwość 12 Hz) 0,2438

8 - 10	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	45	15
K3	24	76

Tabele od 5 do 8 zawierają macierze pomyłek klasyfikatora dla przedziałów czasowych 2-4, 4-6, 6-8, 8-10 - dla częstotliwości 12 Hz.

Tabela 5 Wyniki klasyfikatora dla danych testujących (przedział czasowy 2-4 częstotliwość 12 Hz)

Całkowity błąd dla danych testujących (przedział czasowy 2-4 częstotliwość 12 Hz) 0,2082

2 - 4	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	184	46
K3	56	204

Tabela 6 Wyniki klasyfikatora dla danych testujących (przedział czasowy 4-6 częstotliwość 12 Hz)

Całkowity błąd dla danych testujących (przedział czasowy 4-6 częstotliwość 12 Hz) 0,1429

4 - 6	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	194	36
K3	34	226

Tabela 7 Wyniki klasyfikatora dla danych testujących (przedział czasowy 6-8 częstotliwość 12 Hz)

Całkowity błąd dla danych testujących (przedział czasowy 6-8 częstotliwość 12 Hz) 0,2408

6 - 8	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	182	48
K3	70	190

Tabela 8 Wyniki klasyfikatora dla danych testujących (przedział czasowy 8-10 częstotliwość 12 Hz)

Całkowity błąd dla danych testujących (przedział czasowy 8-10 częstotliwość 12 Hz) 0,1959

8 - 10	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	184	46
K3	50	210

Tabele od 9 do 12 zawierają macierze pomyłek dla danych uczących w trybie nauki klasyfikatora dla przedziałów czasowych 2-4, 4-6, 6-8, 8-10 dla częstotliwości 20 Hz.

Tabela 9 Wyniki klasyfikatora dla danych uczących (przedział czasowy 2-4 częstotliwość 20 Hz)

Całkowity błąd dla danych uczących (przedział czasowy 2-4 częstotliwość 20 Hz) 0,1813

2 - 4	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	59	21
K3	18	62

Tabela 10 Wyniki klasyfikatora dla danych uczących (przedział czasowy 4-6 częstotliwość 20 Hz)

Całkowity błąd dla danych uczących (przedział czasowy 4-6 częstotliwość 20 Hz) 0,2437

4 - 6	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	45	15
K3	24	76

Tabela 11 Wyniki klasyfikatora dla danych uczących (przedział czasowy 6-8 częstotliwość 20 Hz)

Całkowity błąd dla danych uczących (przedział czasowy 6-8 częstotliwość 20 Hz) 0,2938

6 - 8	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	62	26
K3	21	49

Tabela 12 Wyniki klasyfikatora dla danych uczących (przedział czasowy 8-10 częstotliwość 20 Hz)

Całkowity błąd dla danych uczących (przedział czasowy 8-10 częstotliwość 20 Hz) 0,1250

8 - 10	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	61	9
K3	11	79

Tabele od 13 do 16 zawierają macierze pomyłek klasyfikatora dla przedziałów czasowych 2-4, 4-6, 6-8, 8-10 dla częstotliwości 20 Hz.

Tabela 13 Wyniki klasyfikatora dla danych testujących (przedział czasowy 2-4 częstotliwość 20 Hz)

Całkowity błąd dla danych testujących (przedział czasowy 2-4 częstotliwość 20Hz) 0,2388

2 - 4	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	178	52
K3	65	195

Tabela 14 Wyniki klasyfikatora dla danych testujących (przedział czasowy 4-6 częstotliwość 20 Hz)

Całkowity błąd dla danych testujących (przedział czasowy 4-6 częstotliwość 20Hz) 0,1816

4 - 6	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	187	43
K3	46	214

Tabela 15 Wyniki klasyfikatora dla danych testujących (przedział czasowy 6-8 częstotliwość 20 Hz)

Całkowity błąd dla danych testujących (przedział czasowy 6-8 częstotliwość 20Hz) 0,2142

6 - 8	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	177	53
K3	52	208

Tabela 16 Wyniki klasyfikatora dla danych testujących (przedział czasowy 8-10 częstotliwość 20 Hz)

Całkowity błąd dla danych testujących (przedział czasowy 8-10 częstotliwość 20Hz) 0,1286

8 - 10	Obiekt należy do klasy	
Klasyfikator przydzielił obiekt do klasy	K2	K3
K2	198	32
K3	31	229

Analizując wyniki klasyfikacji stwierdzono, że najlepszy wynik klasyfikacji dla danej częstotliwości uzyskuje się w wybranym przedziale czasowym, który jest charakterystyczny dla osoby testowanej. Ten najlepszy przedział czasowy może być zauważony już na etapie nauki klasyfikatora.

5. Prezentacja praktycznych możliwości sterowania za pomocą aktywności myślowych za pomocą Wi-fi i Bluetooth.

Wyniki klasyfikacji sygnału związanego z aktywnością myślową K2, K3 są wyświetlane na monitorze komputera w oknie dialogowym programu klasyfikującego jako pola tekstowe: „LEFT" dla K2 i „RIGHT" dla K3. Wyniki klasyfikacji wyświetlane są w oknie programu do momentu zatrzymania programu. Pozwala to na obserwację w sposób ciągły wyników klasyfikacji i ocenę w czasie rzeczywistym intencji użytkownika związanych z wyobrażaniem tych aktywności ruchowych.

Klasyfikacja wymienionych stanów umożliwia wykonanie sterowania. Algorytm klasyfikujący przelicza sygnały zebrane z headsetu, a wynik klasyfikacji wyświetlany na ekranie komputera wysyła do urządzenia zewnętrznego [32].

Autorka zaprojektowała proste urządzenie zewnętrzne pozwalające na sterowanie dwiema diodami LED za pomocą aktywności myślowej K2 i K3. Dioda zielona zapala się i gaśnie, kiedy wynik klasyfikacji wskazuje aktywność K3 („RIGHT"). Dioda czerwona zapala się i gaśnie, jeżeli aktywność zostanie zaklasyfikowana jako K2 („LEFT"). Urządzenie jest sterowane bezprzewodowo z poziomu komputera zawierającego program analizujący i klasyfikujący sygnał. Po wykonaniu klasyfikacji impuls sterujący jest przekazywany do urządzenia zewnętrznego. Przy projektowaniu sterowania wykorzystana została technologia Bluetooth – bezprzewodowa komunikacja krótkiego zasięgu pomiędzy urządzeniami oraz technologia Wi-fi.

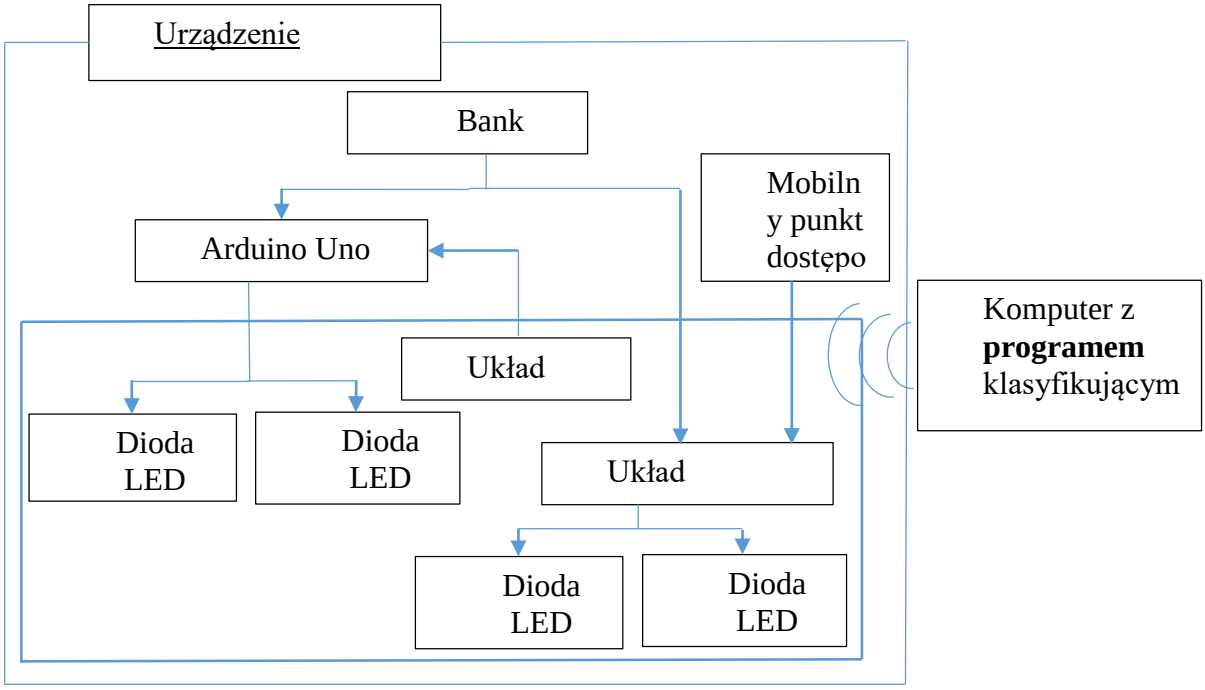
5.1. Konstrukcja urządzenia

Urządzenie zewnętrzne wyświetlające wyniki klasyfikacji obejmuje następujące elementy składowe:

- komputer z działającym oprogramowaniem do klasyfikacji sygnału aktywności myślowej, na którym wyświetla się okno dialogowe programu klasyfikującego
- układ NodeMCU, pozwalający na komunikację poprzez Wi-fi
- układ HC-05 - moduł Bluetooth z adapterem do płytki stykowej
- płytka mikrokontrolera Arduino Uno R3
- dwie diody LED (czerwona i zielona) – dwa komplety
- dwa rezystory 220 Ohm

– mobilny punkt dostępowy
– prototypowa płytka stykowa
– bank energii zasilający NodeMCU oraz Arduino Uno

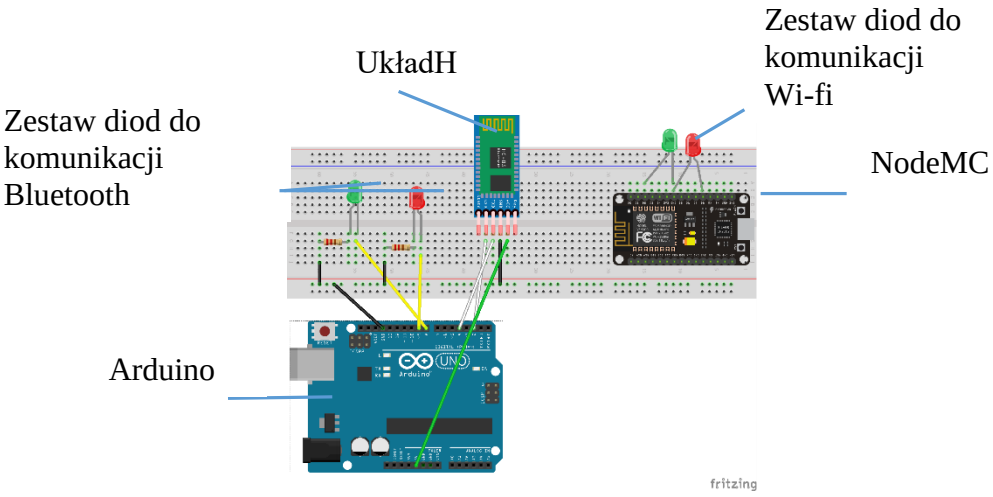
Rys. 12 przedstawia schemat blokowy urządzenia.



Rysunek 12 Schemat blokowy – urządzenie zewnętrzne i komputer z programem klasyfikującym[16]

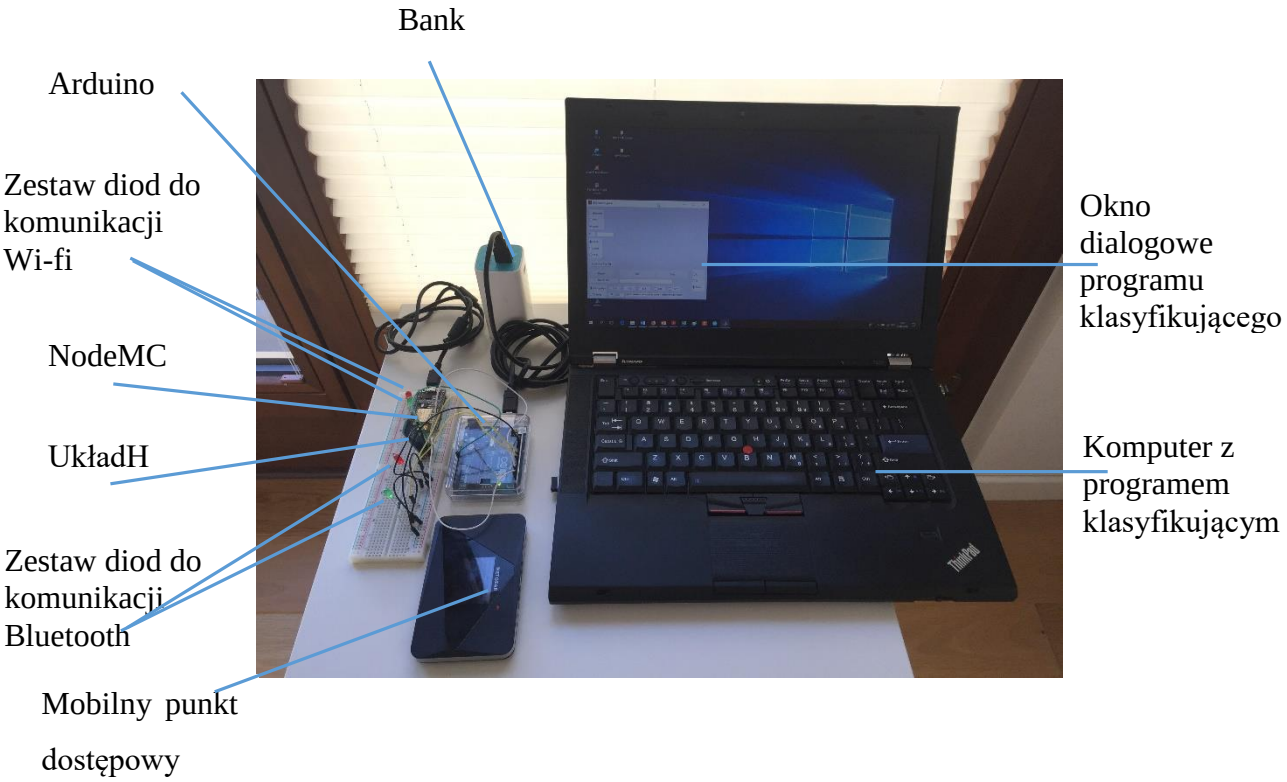
Program klasyfikujący składa się z oprogramowania napisanego w języku Matlab (wersja Matlab R2011b - 32-bit), korzystającego z biblioteki dzielonej (eegloglib.dll) napisanej przez autrokę w jezyku C++, która pobiera surowe (*raw*) dane z elektrod headsetu Emotiv i zapisuje pliki *raw* w podkatalogach odpowiadających kolejnym pomiarom. Biblioteka C++ została napisana na podstawie dostarczonego z headsetem SDK Emotiv Epoc 32-bit. Diody, rezystory oraz układ HC-05 i układ NodeMCU umieszczone są na stykowej płytce prototypowej. Dwa typy komunikacji: Wi-fi i Bluetooth, używają osobnych zestawów diod. Schemat elektryczny urządzenia przedstawia rys. 13. Schemat został wykonany w programie „Fritzing".

[16] Rysunek własny.



Rysunek 13 Schemat elektryczny urządzenia przy komunikacji Bluetooth

Rzeczywisty układ wykorzystywany przy pomiarach i sterowaniu przedstawia rys. 14.



Rysunek 14 Układ wykorzystywany przy pomiarach i sterowaniu – urządzenie zewnętrzne i komputer z programem klasyfikującym

Autorka zaprojektowała graficzny interfejs użytkownika w postaci okna dialogowego programu klasyfikującego, tak żeby możliwe było ustawianie żądanych parametrów klasyfikacji i współpracy programu klasyfikującego z urządzeniem. Okno dialogowe pozwala na wybranie komunikacji poprzez Wi-fi lub Bluetooth, czyli przekazywanie bezprzewodowo wyników klasyfikacji do urządzenia zewnętrznego. W przypadku braku wyboru którejś z tych opcji wyniki klasyfikacji wyświetlane są tylko w oknie dialogowym programu klasyfikującego. Za pomocą przycisków okna dialogowego w programie Matlab wybierane są również częstotliwości analizy sygnału zebranego z elektrod, a także przedziały czasowe sygnału, w których wyznaczany jest wynik klasyfikacji.

Okno dialogowe umożliwia wybór trybu pracy – tj. trening („Training") lub normalna praca („Recognition"). Tryb pracy „Training" przydatny jest przy ocenie najlepszych parametrów pomiaru dla użytkownika/osoby testowanej, np. przedział czasowy, w którym obserwowana jest poprawność wyników klasyfikacji lub częstotliwość która najlepiej oddaje intencje użytkownika (czy K2, czy K3).

Można również w oknie dialogowym wybrać zadania myślowe dotyczące prawej, czy też lewej ręki (jest to ułatwienie w tworzeniu raportów szczególnie przy trenowaniu lub testowaniu użytkownika, kiedy aktywność myślowa jest zadawana). Można nie dokonywać wyboru ręki („NONE"), jeżeli użytkownik steruje urządzeniem w trybie normalnej pracy „Recognition" bez konieczności analizowania wyników klasyfikacji, lub wcześniejszego określania aktywności, która ma zostać wykonana.

W oknie dialogowym można wybrać przyciski:

- *„Start"* - przycisk umożliwiający rozpoczęcie sterowania/pomiarów. Wywołuje podstawowe pliki i funkcje programu pozwalające na zainicjowanie, przeprowadzenie i zakończenie procesu sterowania/pomiaru
- *„Stop"* - przycisk umożliwiający zatrzymanie sesji pomiarowej
- *„Report"* - przycisk umożliwiający wygenerowanie raportu z danej sesji/próby sterowania/pomiarowej
- *„Load data from file"* - przycisk umożliwiający odtworzenie wyników klasyfikacji z zarejestrowanych danych surowych (*raw*) i wygenerowanie raportu na podstawie tych danych.

Nazwę raportu można wprowadzić w polu *„Report title"*. Można też ustawić wartość promienia głowy *R*.

5.2. Transmisja impulsów sterujących

Autorka przetestowała transmisję wyniku klasyfikacji z użyciem trybu bezprzewodowej komunikacji Wi-fi oraz Bluetooth. Komunikacja z użyciem Wi-fi umożliwiła przekazanie wyniku klasyfikacji na większą odległość niż w przypadku standardu Bluetooth. Komunikacja poprzez Wi-fi pozwoliła także na przekazanie wyniku klasyfikacji do urządzenia umieszczonego w innym pomieszczeniu niż komputer z programem klasyfikującym wysyłający impuls sterujący.

5.2.1. Komunikacja poprzez Wi-fi

Do komunikacji Wi-fi autorka wykorzystała układ NodeMCU oparty na układzie ESP8266-12E. Przy komunikacji Wi-Fi komputer z programem klasyfikującym wysyła bezprzewodowo, poprzez mobilny punkt dostępowy, impuls sterujący do układu NodeMCU V2 zawierającego wbudowane Wi-fi oraz mikrokontroler. Następnie układ NodeMCU V2 przekazuje impuls sterujący do odpowiedniej diody. Dla wysyłanego z programu Matlab impulsu "1" (K3), zapala się zielona dioda, dla impulsu "0" (K2) zapala się czerwona dioda.

Parametry modułu NodeMCU V2:

- Moduł zbudowany w oparciu o układ ESP8266-12E z anteną PCB
- Łączność Wi-fi w standardzie 802.11 b/g/n
- Działa w trybach AP (Access Point), STA (Standalone), AP+STA
- Obsługuje TKIP, WEP, CRC, CCMP, WPA/WPA2, WPS
- Zasilanie: 3.3V (lub 5V przez port USB)
- Procesor RISC 80MHz
- 10 portów GPIO - PWM / I2C / SPI / 1-Wire
- Maksymalne natężenie na pinach I/O: 12mA
- Konwerter USB-UART - CP2102
- Konwerter ADC - 10-bitowy
- 30 pinów w rastrze 2,54mm - pasuje do płytek stykowych
- Złącze micro USB
- Wymiary: 49 x 25mm
- Dwa przyciski: FLASH, służący do programowania, oraz RST (RESET)

- Niebieska dioda LED podłączona do GPIO16

Układ NodeMCU V2 posiada wbudowany konwerter USB-UART, dzięki któremu można zaprogramować płytkę przez USB z użyciem Arduino IDE[17] może być zasilany zewnętrznie za pomocą banku energii.

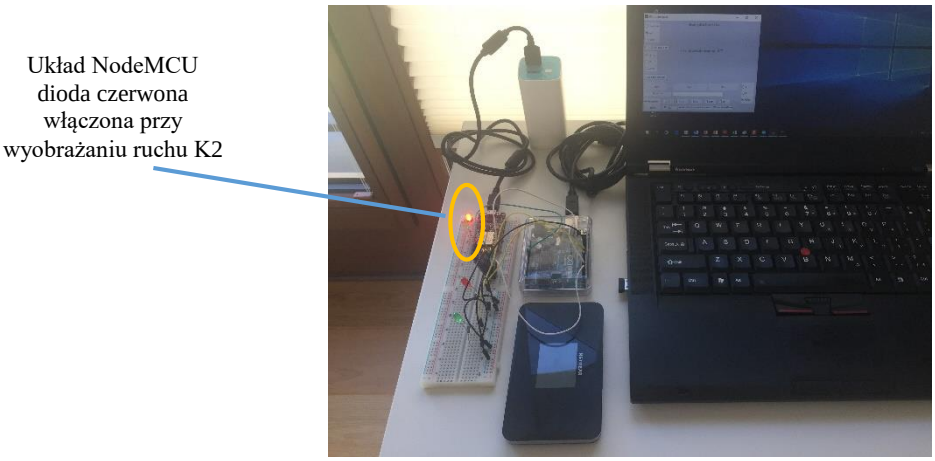
Autorka napisała program w środowisku programistycznym Arduino IDE, który został wgrany do modułu NodeMCU V2:

```
#include <ESP8266WiFi.h>

WiFiServer server(8888);
WiFiClient client;

const char* ssid     = "…"; // Nazwa wifi
const char* password = "…"; // Hasło do wifi

//int ledPin = 12; // D6

int ledPinGreen = 4; // D2
int ledPinRed = 15; // D8

void setup() {     //funkcja uruchamiana raz przy starcie

      Serial.begin(115200);
      delay(200);

      pinMode(ledPinRed, OUTPUT);
      pinMode(ledPinGreen, OUTPUT);

      digitalWrite(ledPinRed, HIGH);
      digitalWrite(ledPinGreen, HIGH);
      delay(1000);
      digitalWrite(ledPinRed, LOW);
      digitalWrite(ledPinGreen, LOW);

      // Connect to WiFi network
      Serial.println();
      Serial.print("Your are connecting to;");
      Serial.println(ssid);

      WiFi.begin(ssid, password);

      while (WiFi.status() != WL_CONNECTED) { //instrukcja
      sterująca przebiegiem programu wykonywana w pętli
        delay(500);
        Serial.print(".");
      }

      Serial.println("Your ESP is connected!");
      Serial.println("Your IP address is: ");
      Serial.println(WiFi.localIP());

      // Start the server
      server.begin();
}
```

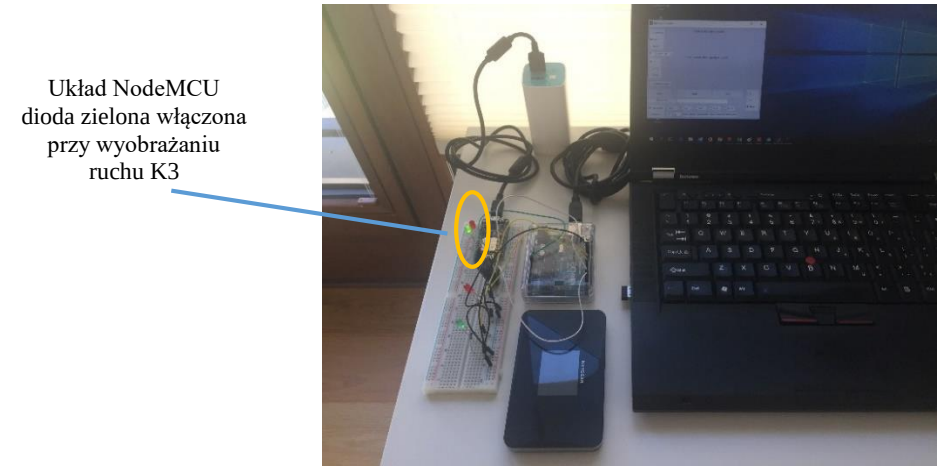
[17] Opis na podstawie: https://nettigo.pl/products/modul-wifi-nodemcu-v2-bezprzewodowy-modul-oparty-na-esp8266-12e - stan na dzień 21.10.2018 r.

```
void loop() {

    if (!client.connected()) {

        // try to connect to a new client
        client = server.available();
        if( client ) {
          Serial.println("Client connected");
        }

    } else {
        // read data from the connected client
        if (client.available() > 0) {
            char c = client.read();
            Serial.write(c);
            if( c == 'l' ){
              Serial.println("Diode L");
              digitalWrite(ledPinRed, HIGH); // włączmy diodę,
              podajemy stan wysoki
              delay(2000);
              digitalWrite(ledPinRed, LOW);
            }
            else if( c == 'r' ){
              Serial.println("Diode R");
                 digitalWrite(ledPinGreen, HIGH); // wyłączamy diodę,
                 podajemy stan
                 niski
              delay(2000);
              digitalWrite(ledPinGreen, LOW);
            }
            else
              Serial.println("Error");

            client.write('@');

        }
    }
}
```

Urządzenie z układem NodeMCU pozwalające na komunikację poprzez Wi-fi w przypadku wyobrażania ruchów K2 i K3 przedstawiono odpowiednio na rys. 15 i rys. 16 .

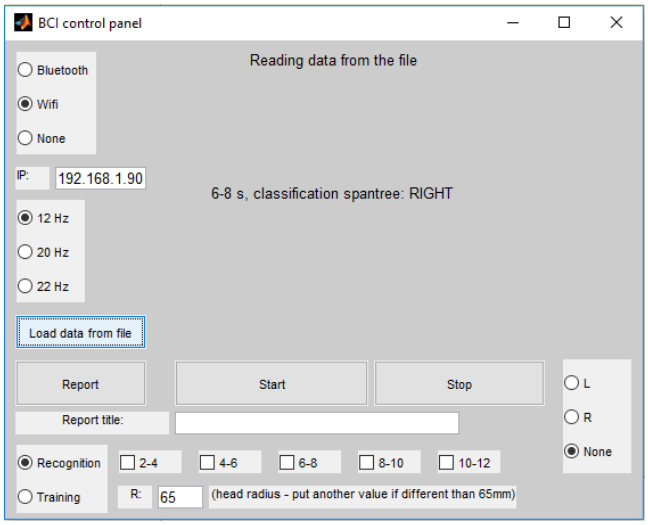


Rysunek 15 Urządzenie z układem NodeMCU podczas komunikacji poprzez Wi-fi przy wyobrażeniu ruchu K2 (dioda czerwona włączona)



Rysunek 16 Urządzenie z układem NodeMCU podczas komunikacji poprzez Wi-fi przy wyobrażeniu ruchu K3 (dioda zielona włączona)

Okno dialogowe umożliwiające wybór sterowania poprzez Wi-fi zawiera pole do podania adresu IP urządzenia NodeMCU sterującego diodami. Okno dialogowe z wybranym sterowaniem za pomocą Wi-fi w widoku czytania danych z pliku przedstawia rys 17.



Rysunek 17 Okno dialogowe przy wyborze komunikacji poprzez Wi-fi pomiędzy komputerem z programem sterującym a urządzeniem zewnętrznym w widoku czytania danych z pliku

5.2.2. Komunikacja poprzez Bluetooth

Przy komunikacji poprzez Bluetooth układ HC-05 (moduł Bluetooth master/slave) z którym komunikuje się komputer jest podłączony do układu Arduino Uno R3, zasilanego z banku energii z użyciem kabla USB. Na komputerze w środowisku Matlab uruchomiony jest program, który po zaklasyfikowaniu sygnału K2 lub K3, za pomocą łączności Bluetooth wysyła bezprzewodowo impulsy sterujące ("0" dla K2 lub "1" dla K3) do układu HC-05. HC-05 przesyła bezprzewodowo impulsy sterujące do układu Arduino Uno R3 wyposażonego w mikrokontroler ATMega328, do którego został wgrany kod sterujący urządzeniem. Układ Arduino Uno R3 przekazuje impulsy sterujące do odpowiedniej diody. Tak jak przy sterowaniu poprzez Wi-fi dla wysyłanego z programu Matlab impulsu "1" (K3), zapala się zielona dioda, dla impulsu "0" (K2) zapala się czerwona dioda. Napięcie podawane na diody zmniejszane jest przez dwa rezystory 220 Ohm.

Specyfikacja układu HC-05[18]:

- Wersja: Bluetooth Specification v2.0 + EDR
- Układ: HC-05 – master/slave
- Częstotliwość: 2.4GHz ISM
- Modulacja: GFSK (Gaussian Frequency Shift Keying)

[18] Specyfikacja na podstawie: https://nettigo.pl/products/modul-bluetooth-z-adapterem-do-plytki-stykowej (stan na dzień 21.10.2019 r.)

- Moc nadawania: ≤ 4dBm, Class 2
- Czułość: ≤ -84dBm przy 0.1% BER
- Prędkość: Asynchroniczna do 2.1Mbps(Max) / 160 kbps, Synchroniczna 1Mbps / 1Mbps
- Zasilanie +3.3VDC, 50mA
- Rozmiar: 26.9 mm x 13mm x 2.2 mm

Jest to moduł Bluetooth z adapterem do płytki stykowej. Moduł pracuje jako konwerter Bluetooth do TTL Serial. Wyprowadzone piny pozwalają podpiąć go do płytki stykowej. Maksymalny zasięg modułu Bluetooth HC-05 wynosi 10 m.

Do zaprogramowania mikrokontrolera ATMega328 tak jak w przypadku układu NodeMCU V2, autorka użyła środowiska programistycznego „Wiring". Napisała kod do sterowania diodami z użyciem Arduino IDE.

Urządzenie Bluetooth HC-05 widoczne jest jako port szeregowy, dlatego w programie została zastosowana biblioteka SoftwareSerial umożliwiająca komunikację z portem szeregowym.

Kod programu, napisany przez autorkę, który umożliwia sterowanie diodami wgrany na płytkę Arduino Uno:

```
#include <SoftwareSerial.h>

SoftwareSerial ss(4, 2);
int bdata;

void setup() {

  ss.begin(9600);
  Serial.begin(9600);
  Serial.println("setup ok");

  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
}

void loop() {

  if( ss.available() ) {
    Serial.println("read ...");
    bdata = ss.read();
    Serial.println("read ok");

    if( bdata == 1 ) {
      ss.println("data is 1");
      Serial.println("data is 1");

      digitalWrite(8, HIGH);
      digitalWrite(9, LOW);
      delay(2000);
      digitalWrite(8, LOW);
```

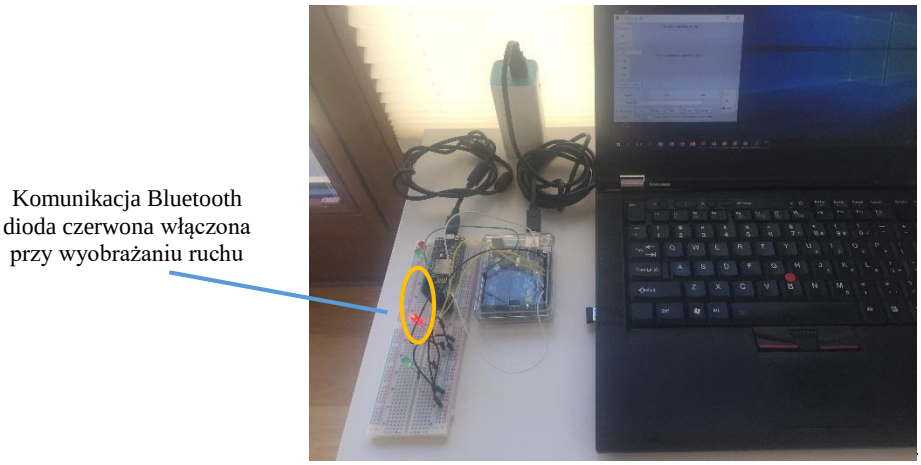
```
    }
    else if( bdata == 0 ) {
      ss.println("data is 0");
      Serial.println("data is 0");
      digitalWrite(9, HIGH);
      digitalWrite(8, LOW);
      delay(2000);
      digitalWrite(9, LOW);
    }
  } else {
    Serial.println("not available");
  }
  delay(500);
}
```

Widok płytki Arduino Uno R3 z rozmieszczeniem wyprowadzeń przedstawia rys. 18.

Rysunek 18 Płytka Arduino Uno R3

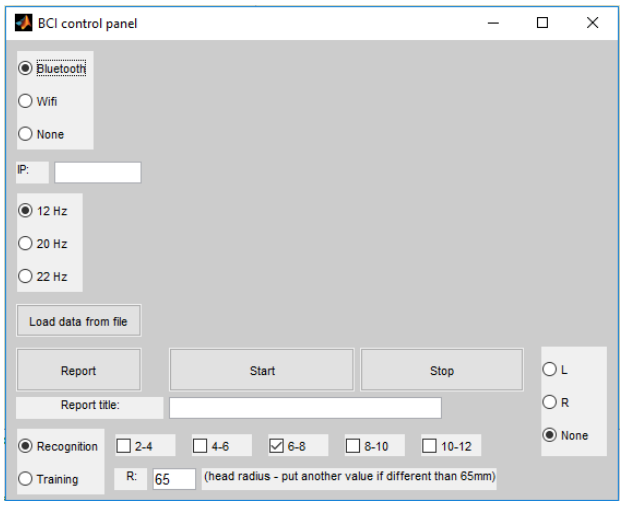
Płytka Arduino wyposażona jest w złącze USB, które umożliwia komunikację z komputerem – wgrywanie kodu programu i zasilanie platformy. Arduino może być też zasilana oddzielnie poprzez dodatkowe złącze zasilające (oddzielne zasilanie za pomocą banku energii wykorzystała autorka do bezprzewodowej komunikacji z urządzeniem przy sterowaniu poprzez Bluetooth.

Urządzenie zewnętrzne wskazujące przykładowy wynik klasyfikacji dla K2 (zapalona dioda czerwona) przy komunikacji poprzez Bluetooth przedstawia rys. 19.



Rysunek 19 Urządzenie zewnętrzne przy komunikacji Bluetooth z zapaloną diodą czerwoną sygnalizującą klasyfikację aktywności myślowej dla K2 „LEFT”

Okno dialogowe dla trybu pracy Bluetooth z przykładowymi ustawionymi parametrami początkowymi: analiza klasyfikowanego sygnału na częstotliwości 12 Hz, przedział czasowy analizowanego sygnału 6-8, tryb pracy – „Recognition” (normalna praca), nieokreślona aktywność K3 czy K2 – tryb „NONE”, promień głowy 65 mm, przedstawia rys. 20.

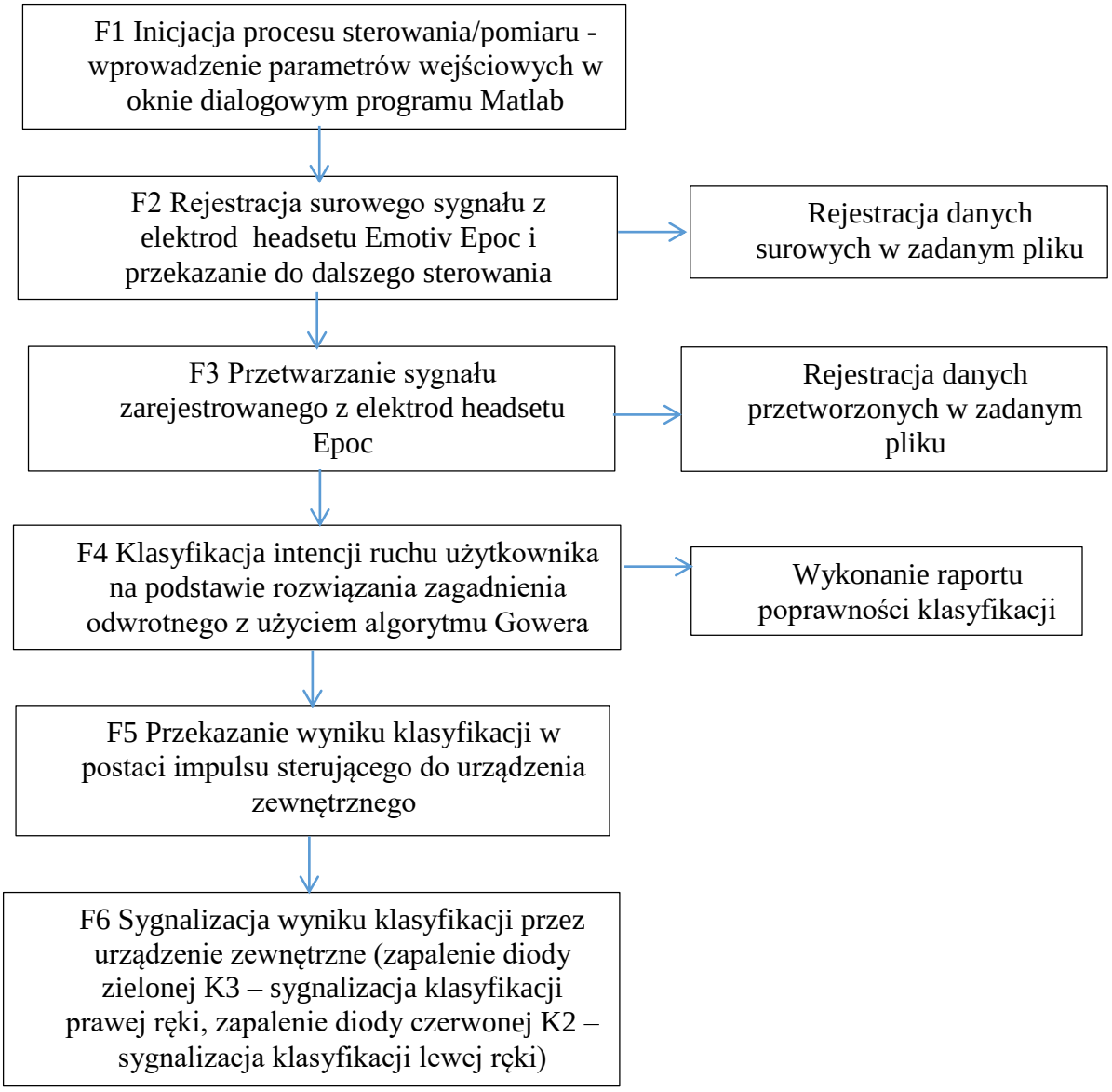


Rysunek 20 Okno dialogowe z przykładowymi ustawieniami początkowymi przy komunikacji z urządzeniem zewnętrznym za pomocą protokołu Bluetooth

5.3. Schemat działania urządzenia w systemie BCI

Schemat procesu sterowania urządzeniem zewnętrznym w systemie BCI od momentu wprowadzenia parametrów wejściowych w oknie dialogowym programu, poprzez proces klasyfikacji w oparciu o rozwiązanie zagadnienia odwrotnego i algorytm Gowera, do

momentu reakcji na sygnał sterujący przez urządzenie zewnętrzne został przedstawiony na rys. 21.



Rysunek 21 Schemat działania systemu BCI

Cały schemat sterowania urządzeniem zewnętrznym w systemie BCI, który został przedstawiony na rys. 21 wykonywany jest przez program napisany przez autorkę w języku Matlab (rozdział 6, rys. 22). Schemat procesu sterowania jest taki sam przy komunikacji przez Wi-fi i za pomocą protokołu Bluetooth. Dla ułatwienia analizy wyników pomiarów, autorka zaprojektowała możliwość rejestrowania danych w raportach zewnętrznych w programie MS Excel. Po odpowiednim uporządkowaniu, w oparciu o opracowaną aplikację MS Excel VBA, łatwo można przeanalizować poprawność wyników klasyfikacji na podstawie raportów.

6. Realizacja programowa klasyfikacji i sterowania urządzeniem zewnętrznym

Program opracowany z użyciem języka Matlab ma budowę modułową. Umożliwia rejestrację i przetwarzanie sygnałów zarejestrowanych z elektrod a następnie wyodrębnianie, selekcję i klasyfikację cech oraz sterowanie urządzeniem zewnętrznym. Autorka użyła wersji Matlab R2011b w wersji 32-bit, ponieważ SDK Emotive Epoc było dostarczone w wersji 32-bit. Na podstawie SDK Emotiv Epoc autorka napisała bibliotekę w języku C++ (eegloglib.dll), która umożliwia w programie Matlab pobieranie surowch danych z headsetu Emotive Epoc. Dane pobrane z headsetu zapisywane są do plików na lokalnym dysku w podkatalogach odpowiadającym poszczególnym pomiarom. Główne moduły programu wywołują szczegółowe funkcje wykonujące operacje zgodnie ze schemtem działania systemu BCI podanym na rys. 21:

F1. Inicjacja procesu sterowania/pomiaru

F2. Rejestracja surowego sygnału z elektrod headsetu Emotiv i zapis do pliku oraz przekazanie do dalszego przetwarzania

F3. Przetwarzanie sygnału zarejestrowanego z elektrod headsetu Emotiv i zapis do pliku danych przetworzonych

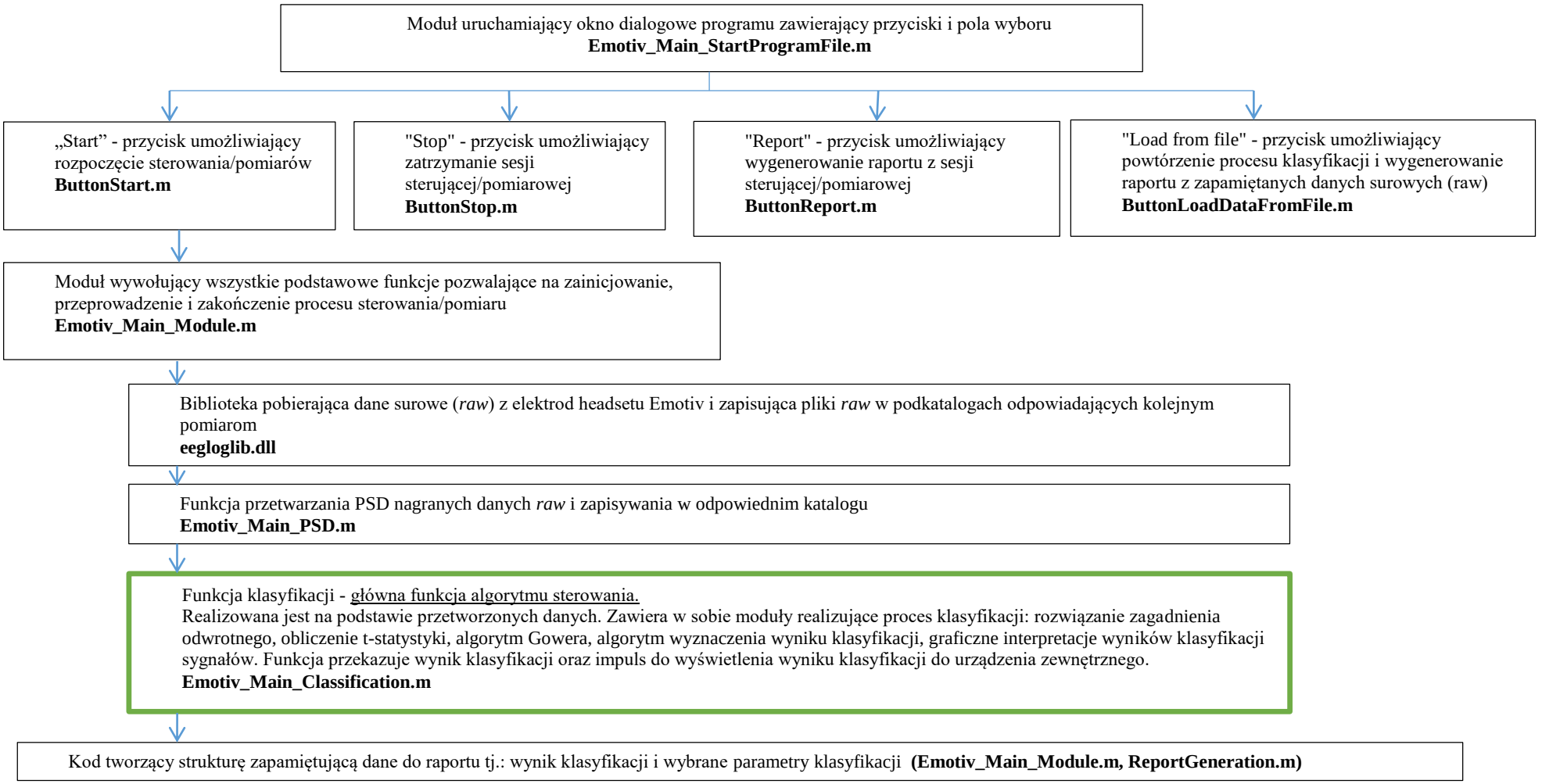
F4. Klasyfikacja intencji ruchu użytkownika – główna funkcja algorytmu sterowania urządzeniem zewnętrznym

F5. Przekazanie wyniku klasyfikacji w postaci impulsu sterującego do urządzenia zewnętrznego

F6. Sygnalizacja wyniku klasyfikacji przez urządzenie zewnętrzne (zapalenie diody zielonej – sygnalizacja klasyfikacji prawej ręki K3, zapalenie diody czerwonej – sygnalizacja klasyfikacji lewej ręki K2).

W środowisku Matlab pliki mają takie same nazwy jak funkcje w nich zawarte.

Architekturę programu klasyfikującego przedstawia rys. 22 .



Rysunek 22 Architektura programu klasyfikującego – pliki programu Matlab podano w zał. 4

F1. Inicjacja procesu sterowania/pomiaru - wprowadzenie parametrów wejściowych w oknie dialogowym programu Matlab

Po uruchomieniu programu otwiera się okno dialogowe (rys. 20), które pozwala na wprowadzenie parametrów wejściowych pomiaru lub pracy programu.

Okno dialogowe jest inicjowane przez moduł uruchamiający programu Emotiv_Main_StartProgramFile.m. Moduł otwiera okno dialogowe programu Matlab i ładuje bibliotekę dzieloną (eegloglib.dll) napisaną w języku C++, która czyta dane surowe z headsetu i zapisuje je w zadanym pliku.

Okno dialogowe zawiera przyciski. Z każdym przyciskiem skojarzona jest odpowiednia akcja uruchamiana przez wybranie danego przycisku:

– „*Start*" - przycisk umożliwiający rozpoczęcie sterowania/pomiarów. Wywołuje podstawowe pliki i funkcje programu pozwalające na zainicjowanie, przeprowadzenie i zakończenie procesu sterowania/pomiaru.

  Po wybraniu przycisku *„Start"* w oknie dialogowym rozpoczyna się realizacja kodu z pliku ButtonStart.m, która uaktywnia parametry wybrane w oknie dialogowym i umożliwia przejście do kolejnej funkcji Emotiv_Main_Module.m. Jest to plik programu, który wywołuje wszystkie podstawowe funkcje, tj.:

  ✓ Bibliotekę dzieloną (eegloglib.dll), która podłącza się do headsetu, pobiera dane surowe (*raw*) z 14 elektrod headsetu Emotiv i zapisuje pliki *raw* do wskazanego pliku, który następnie jest kopiowany przez program do podkatalogów odpowiadających kolejnym sesjom pomiarowym (F2).

  ✓ Funkcję przetwarzania PSD nagranych danych *raw* i zapisywanie w odpowiednim katalogu (F3) Emotiv_Main_PSD.m .

  ✓ Funkcję klasyfikacji na podstawie przetworzonych danych Emotiv_Main_Classification.m .

  Funkcja klasyfikacji jest podstawową funkcją algorytmu sterowania urządzeniem. Poprawny wynik klasyfikacji uruchamia proces sterowania urządzeniem – pozwala na realizację intencji użytkownika.

  Funkcja ta zawiera w sobie moduły realizujące proces klasyfikacji: rozwiązanie zagadnienia odwrotnego (wyznaczenie przybliżonej lokalizacji źródeł sygnałów), obliczenie t-statystyki, algorytm Gowera (drzewo rozpinające jest klasyfikatorem), algorytm wyznaczenia wyniku klasyfikacji, graficzne interpretacje wyników klasyfikacji sygnałów prezentowanych w rozprawie doktorskiej. Funkcja

przekazuje wynik klasyfikacji oraz impuls do wyświetlenia wyniku klasyfikacji do urządzenia zewnętrznego (F4, F5, F6):

- ✓ Kod tworzący strukturę zapamiętującą dane do raportu, tj.: wynik klasyfikacji i wybrane parametry klasyfikacji (częstotliwości sygnałów wykorzystywanych do klasyfikacji, przedziały czasowe wybrane do analizy sygnałów, symbol wyboru ręki dla danego pomiaru w sesji pomiarowej, obszary P19 biorące udział w procesie klasyfikacji sygnału) (F4)

– „*Stop*" - przycisk umożliwiający zatrzymanie sesji pomiarowej

Po wybraniu przycisku *„Stop"* w oknie dialogowym rozpoczyna się realizacja kodu z pliku ButtonStop.m, który zatrzymuje sesję pomiarową. Po naciśnięciu przycisku *„Stop"* można wygenerować raport z sesji sterującej/pomiarów. Jeżeli okno dialogowe nie zostanie zamknięte można ponownie wybrać przycisk *„Start"* i nagrać kolejną/kolejne sesje pomiarowe.

– „*Report*" - przycisk umożliwia wygenerowanie raportu z danej sesji/próby sterowania/pomiarowej

Po wybraniu przycisku *„Report"* w oknie dialogowym rozpoczyna się realizacja kodu z pliku ButtonReport.m, która generuje raport z sesji sterującej/pomiarowej. Raport generowany jest do pliku Microsoft Excel. Do wygenerowania raportu program musi być zatrzymany – *„Stop"* (jw.)

– „*Load data from file*" - przycisk umożliwia odtworzenie wyników klasyfikacji z zarejestrowanych danych surowych (*raw*) i wygenerowanie raportu na podstawie tych danych.

Po wybraniu przycisku „*Load data from file*" w oknie dialogowym rozpoczyna się realizacja kodu z pliku ButtonLoadDataFromFile.m , która umożliwia powtórzenie procesu klasyfikacji i wygenerowanie raportu z zapamiętanych danych surowych (*raw*).

W oknie dialogowym można wybrać też:

– Tryb pracy poprzez Wi-fi lub Bluetooth, czyli przekazywanie bezprzewodowo wyników klasyfikacji do urządzenia zewnętrznego. W przypadku braku wyboru tej opcji wyniki klasyfikacji są wyświetlane tylko w oknie dialogowym programu klasyfikującego

– Częstotliwość analizowanego sygnału

[19] Rysunek 7

- Przedziały czasowe sygnału, w których wyznaczany jest wynik klasyfikacji.
- Tryb pracy: normalna praca („*Recognition*”), tryb treningowy („*Training*”)
- Zadanie myślowe dotyczące wyobrażania ruchu: lewa ręka (K2), prawa ręka (K3), „*None*” - brak wyboru ręki. Wybór ręki ułatwia analizę poprawności klasyfikacji w raportach
- Okno dialogowe umożliwia nadanie nazwy raportu w polu „*Report title*” oraz podanie wartości promienia głowy.

F2. Rejestracja surowego sygnału z elektrod headsetu Emotiv i zapis do pliku

Rejestracja surowego sygnału z elektrod rozpoczyna się po wywołaniu biblioteki dzielonej eegloglib.dll opracowanej przez autorkę w języku C++, na podstawie przykładów dołączonych do SDK Emotive Epoc.

Dane wejściowe do biblioteki stanowią odczyty z elektrod headsetu Emotiv. Biblioteka zwraca surowe sygnały z elektrod (*raw*).

Biblioteka eegloglib.dll podłącza się do headsetu Emotiv, pobiera dane surowe (*raw*) z 14 elektrod headsetu Emotiv i zapisuje pliki *raw* do wskazanego pliku, który następnie jest kopiowany przez program do podkatalogów odpowiadających kolejnym sesjom pomiarowym. Jednorazowo pobieranych jest 960 odczytów z elektrod. Surowy sygnał zarejestrowany z elektrod headsetu Emotiv jest przekazywany do dalszego przetwarzania (rys. 21 i 22).

F3. Przetwarzanie sygnału zarejestrowanego z elektrod headsetu Emotive Epoc

Aby można było rozpocząć proces klasyfikacji sygnału zarejestrowanego z elektrod headsetu Emotiv Epoc, surowy sygnał musi zostać poddany wstępnej analizie i przetworzony. Wykonywana jest analiza widmowa zarejestrowanego sygnału, która pozwala na wyodrębnienie składowych częstotliwościowych (rozdział 3.2). Analiza widmowa i wyznaczanie mocy sygnałów dla poszczególnych częstotliwości (PSD) są wykonywane przez funkcję: Emotiv_Main_PSD.m (opis F1 – wybór przycisku *„Start”*).

Funkcja przyjmuje na wejściu surowe dane, a zwraca dane przetworzone. Liczność próby danych wejściowych wynosi 64, częstotliwość próbkowania headsetu Emotiv wynosi 128 Hz, co daje rozdzielczość 2 Hz (rozdział 3.2). Funkcja pozwala też na zapisanie przetworzonych danych w odpowiednim katalogu.

Wynikiem przetworzenia sygnału jest moc sygnału na poszczególnych częstotliwościach (PSD) uzyskana po policzeniu dyskretnej transformaty Fouriera z próby,

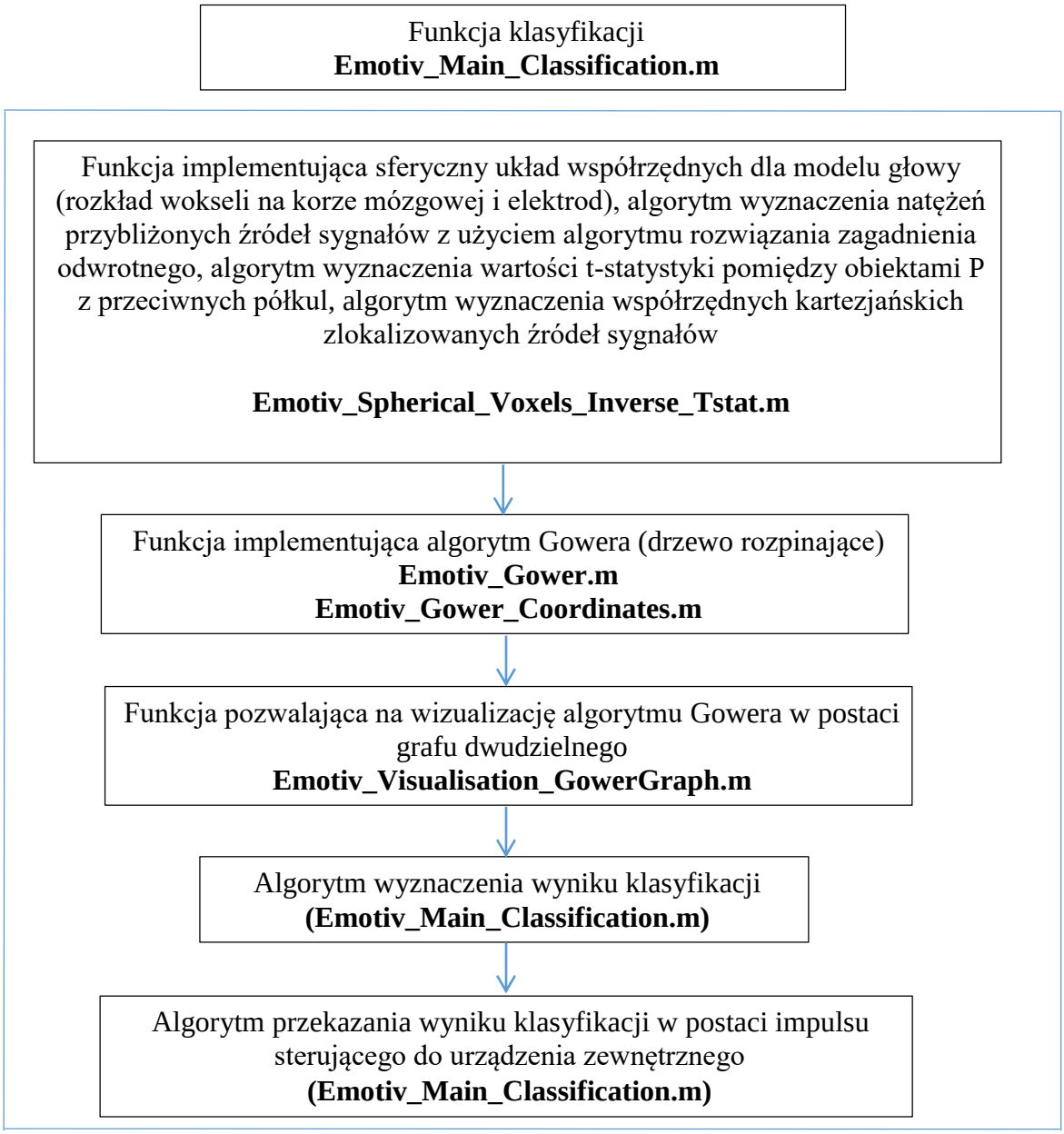
na poszczególnych częstotliwościach. Dane przetworzone są rejestrowane w odpowiednim katalogu (rys. 21 i 22).

F4. Klasyfikacja intencji ruchu użytkownika na podstawie rozwiązania zagadnienia odwrotnego z użyciem t-statystyki i algorytmu Gowera

Funkcja klasyfikacji jest podstawową funkcją algorytmu sterowania urządzeniem. Poprawny wynik klasyfikacji uruchamia proces sterowania urządzeniem – pozwala na realizację intencji użytkownika.

Funkcję klasyfikacji uruchamia plik Emotiv_Main_Classification.m , aktywujący się po wybraniu przycisku *„Start”* w oknie dialogowym programu (F1). Funkcja realizowana jest na podstawie przetworzonych danych. Zawiera w sobie moduły realizujące proces klasyfikacji: rozwiązanie zagadnienia odwrotnego, obliczenie t-statystyki, algorytm Gowera (drzewo minimalne jest klasyfikatorem), algorytm wyznaczenia wyniku klasyfikacji, graficzne interpretacje wyników klasyfikacji sygnałów. Funkcja przekazuje wynik klasyfikacji oraz impuls do wyświetlenia wyniku klasyfikacji do urządzenia zewnętrznego. W module F4 istnieje możliwość generowania raportów poprawności klasyfikacji w programie MS Excel (rys. 21, 22). Po odpowiednim uporządkowaniu, w oparciu o opracowaną aplikację MS Excel VBA, łatwo można przeanalizować poprawność wyników klasyfikacji na podstawie raportów. np. w procesie treningu użytkownika.

Schemat blokowy architektury funkcji realizującej proces klasyfikacji Emotiv_Main_Classification.m przedstawia rys. 23.



Rysunek 23 Architektura funkcji klasyfikującej - pliki programu Matlab podano w zał. 4

Funkcja Emotiv_Main_Classification.m zawiera w sobie następujące moduły i algorytmy realizujące proces klasyfikacji:

- Moduł Emotiv_Spherical_Voxels_Inverse_Tstat.m

  Funkcja implementująca:

  - sferyczny układ współrzędnych dla modelu głowy (rozkład elektrod i wokseli na korze mózgowej)
  - algorytm wyznaczenia natężeń przybliżonych źródeł sygnałów z użyciem algorytmu rozwiązania zagadnienia odwrotnego metodą minimalizacji normy z użyciem metody najmniejszych kwadratów

– algorytm wyznaczenia wartości t-statystyki pomiędzy obiektami P z przeciwnych półkul
– algorytm wyznaczenia współrzędnych sferycznych zlokalizowanych źródeł sygnałów

Dane wejściowe do funkcji Emotiv_Spherical_Voxels_Inverse_Tstat.m stanowią przetworzone sygnały z elektrod (PSD). Funkcja zwraca wartości t-statystyki wyznaczone pomiędzy obszarami z przeciwnych półkul (każde z każdym) opisane współrzędnymi sferycznymi (zgodnie z opisem w rozdziale 3.3.1.).

Przykładowe fragmenty kodów wykorzystywanych w funkcji Emotiv_Spherical_Voxels_Inverse_Tstat.m poniżej:

- Kod prezentujący rozwiązanie zagadnienia odwrotnego i zapisanie wyników w macierzy „Punktywyszukanejj”:

```
PseudoinvK=pinv(K);
szukanaJJ=PseudoinvK*zaimportowaneFi;

JX=zeros(Wnv,1);
m=1;
for n=1:3:trzyWnv;
    JX(m,1) = szukanaJJ(n,1); % [U,S,V] = svd(K);
    m=m+1;
end

JY=zeros(Wnv,1);
m=1;
for n=2:3:trzyWnv;
    JY(m,1) = szukanaJJ(n,1);
    m=m+1;
end

JZ=zeros(Wnv,1);
m=1;
for n=3:3:trzyWnv;
    JZ(m,1) = szukanaJJ(n,1);
    m=m+1;
end

Punktywyszukanejj=zeros(Wnv,1);
j=0;
for j=1:Wnv;
    Punktywyszukanejj(j,1) = sqrt(
    (JX(j,1)^2)+(JY(j,1)^2)+(JZ(j,1)^2));
end
```

- Kod liczenia t-statystyki:

```
%% T-statystyka
TetaStep = 5;
TetaCompartment = 15;
FiStep = 10;
FiCompartment = 30;
compartment_x = 360/FiCompartment;
```

```
compartment_y = 90/TetaCompartment;
wholeCompartment = compartment_x*compartment_y;
halfOfWholeCompartment = (wholeCompartment/2)+1;
compartment_number = 0;

%%% Wejściem do obliczenia t-statystyki są rozwiązania
    zagadnienia odwrotnego
%%% uśrednione na obszarach P zapisane w macierzy
    trMatrixFoundPointsJJ

%%% wartosc srednia w poszczególnych obszarach P
   (compartment)
for aa=1:1:sizeX;
    tab = trMatrixFoundPointsJJ( aa, : );
       compartment = Emotiv_przedzialy_90( tab, TetaStep,
       FiStep, TetaCompartment, FiCompartment );
    compartment = compartment(
    halfOfWholeCompartment:wholeCompartment );
    compartment_number = length( compartment );

    for i=1:1:compartment_number
        le = length( compartment(i).tab );
        my = sum( compartment(i).tab );
        array_my(aa,i) = my / le;
    end
end;

wholeMediumval_y = zeros(1, compartment_number);

for i=1:1:compartment_number
    tab = array_my( :, i );
    len = length( tab );
    sum_value = sum( tab );
    wholeMediumval_y(1, i) = sum_value / len;
end
%%% koniec: wartosc srednia w poszczególnych obszarach P
   (compartment)

%%% wariancje dla obszarów P
count_y = sizeX;
overallVariance_y = zeros(1, compartment_number);

for aa=1:1:sizeX;
    tab = trMatrixFoundPointsJJ( aa, : );
    compartment = Emotiv_przedzialy_90( tab, TetaStep,
    FiStep, TetaCompartment, FiCompartment );
    compartment =
    compartment(halfOfWholeCompartment:wholeCompartment);
    compartment_number = length( compartment );

    for i=1:1:compartment_number
        medium_value = wholeMediumval_y(i);
        tab = array_my(aa,i);
        overallVariance_y(1, i) = (overallVariance_y(1, i)
        + (tab - medium_value)^2);
    end
end;

for i=1:1:compartment_number
    overallVariance_y(1, i) = overallVariance_y(1, i) /
```

```
    count_y;
end
%%% koniec: wariancje dla obszarów P

%%% wartości t-statystyki
matrix = zeros( compartment_number, compartment_number );

for i=1:1:compartment_number
    for j=1:1:compartment_number
        x_d_1 = wholeMediumval_y(i);
        x_d_2 = wholeMediumval_y(j);
        s_1 = overallVariance_y(1, i);
        s_2 = overallVariance_y(1, j);
        t = (x_d_1-
        x_d_2)/sqrt((s_1/count_y)+(s_2/count_y));
        matrix(i,j) = t;
    end
end
%%% koniec: wartosci t-statystyki
```

- Obliczenie t-statystyki wymaga podzielenia głowy na zadaną liczbę obszarów P. Kod funkcji poniżej:

```
%%% Funkcja dzieląca korę mózgową na obszary P o zadanych
    wg zadanych parametrów Fi i Teta
%%% WTO, WFO, WT, WF: Tetakrok, Fikrok, Tetaprzedzial,
    Fiprzedzial
function compartment_array = przedzialy_90(tablica, WTO,
 WFO, WT, WF)

WT_MAX = 90;
WF_MAX = 360;

WCT = (WT_MAX / WTO) / (WT_MAX / WT); % liczba wokseli Teta
WCF = (WF_MAX / WFO) / (WF_MAX / WF); % liczba wokseli Fi
WC = (WF_MAX / WF) * (WT_MAX / WT); % liczba przedziałów

% inicjowanie tablic dla przedziałów w pionie
p = 1;
for wf=1:WF:WF_MAX
    compartment_teta(p).tab = [];
    p = p + 1;
end;

% wybranie wokseli z tablicy
i = 1;
o = 'a';
for wt=WT_MAX:-WTO:WTO
    p = 1;
    for wf=(1):(WF):(WF_MAX)
        for wc=1:1:WCF
            compartment_teta(p).tab = [
            compartment_teta(p).tab tablica( i ) ];
            compartment_teta(p).fi = wf + WF/2 - 1;
            compartment_teta(p).o = o;
            i = i + 1;
        end

        if mod(i-1,WCT*WCF) == 0
```

```
                if strcmp(o, 'a')
                    o = 'b1';
                elseif strcmp(o, 'b1')
                    o = 'b2';
                elseif strcmp(o, 'b2')
                    o = 'c';
                elseif strcmp(o, 'c')
                    o = 'a';
                end
            end

            p = p + 1;
        end
     end
    % inicjowanie tablic dla przedziałów
    p = 1;
    for wf=1:1:WC
        compartment_array(p).tab = [];
        p = p + 1;
    end;
    i = 1;
    j = 1;
    for wt=WT_MAX:-WT:WT
        p = 1;
        for wf=1:WF:WF_MAX
            % Definiowanie, co będzie w polach  przedzial.tab,
              przedzial.Tetakrok,
            % przedzial.Fikrok, itd...
              przedzial.Tetaprzedzial, przedzial.Fiprzedzial
            compartment_array(j).tab =
            compartment_teta(p).tab(i : i - 1 + WCT * WCF);
            compartment_array(j).o = compartment_teta(p).o;
            compartment_array(j).fi = compartment_teta(p).fi;
            teta = wt - WT/2;
            compartment_array(j).teta = teta;

            j = j + 1;
            p = p + 1;
        end
        i = i + WCT * WCF;
    end
```

– Moduł Emotiv_Gower.m

Funkcja implementująca algorytm Gowera. Dane wejściowe do funkcji stanowią wartości t-statystyki wyznaczone pomiędzy obszarami P z przeciwnych półkul (każde z każdym) opisane współrzędnymi sferycznymi. Funkcja zwraca całkowite drzewo rozpinające dla wybranych obszarów a, b i c, b dla K2 i K3 (rys. 9) oraz fragmenty drzew rozpinających po uwzględnieniu zadanych progów (wierzchołkami drzewa rozpinającego są obszary P a krawędziami wartości t-statystyki) (rozdział 3.5)

- Kod algorytmu Gowera wyznaczania drzewa rozpinającego:

```
%%% Algorytm Gowera:
%%% Z drzewa całkowitego wybierane są krawędzie tak, żeby
%%% ich suma  przy tworzeniu drzewa była jak największa
```

```
%%% i w poszczególnych krokach tworzona jest struktura
%%% drzewa rozpinającego
while mainKlineUniqueEmpty ~= 1

    Xprim = [ Xprim newXprim ];
    mainKlineUnique = setdiff(mainKlineUnique, newXprim);
    klineUnique = mainKlineUnique;
    arrayAiAsRPartial = [];
    arrayRWeightTmp = [];
    matrixSumOfWeight = [];

    klineUniqueEmpty = isempty(klineUnique);
    while klineUniqueEmpty ~= 1

        % wez pierwszy element
        ai = klineUnique(1);
        Xi = setdiff (Kdaszekuniquenastale, ai);
        productXprimXi = intersect(Xprim, Xi);

        % sprawdzenie 'checkProductXprimXi', czy zbiór
        % productXprimXi jest pusty. Jeżeli pusty
        % sprawdzenie = 1, jeżeli
        % nie jest pusty sprawdzenie = 0;
        checkProductXprimXi = isempty(productXprimXi);

        while checkProductXprimXi ~= 1

            as = productXprimXi;

            %Utwórz drzewo
            R = [];
            sumValue = [];
            uu = size(X,1);

            for b = 1:1:uu
                if X(b,2) == ai;
                        if isempty(intersect(X(b,3), as)) == 0
                         R = [R X(b,1)];
                    end
                elseif X(b,3) == ai
                    if isempty(intersect(X(b,2), as)) == 0
                        R = [R X(b,1)];
                    end;
                end;
            end;

            if length(R)~=0
                sumValue = [sumValue;R'];
                sumOfWeight = sum(sumValue);
                   matrixSumOfWeight = [matrixSumOfWeight;
                   sumOfWeight];
                arrayRWeightTmp = [arrayRWeightTmp; R'];

                for i=1:1:size(R,2)
                    valueR=R(i);
                    arrayAiAsRPartial = [
                    arrayAiAsRPartial; ai as
                    valueR];
                end;
```

```
                end;

                roznicasprawdzenieiloczynXprimXi = setdiff
                (productXprimXi,as);
                checkProductXprimXi =
                isempty(roznicasprawdzenieiloczynXprimXi);
            end;

            klineUnique = setdiff( klineUnique, ai );
            klineUniqueEmpty = isempty(klineUnique);

        end;

        cumulativeArrayROfWeightTmp_index =
        cumulativeArrayROfWeightTmp_index + 1;

        maxTR = -10e37;
        countT_ROW = size( arrayAiAsRPartial, 1 );
        countT_COL = size( arrayAiAsRPartial, 2 );
        for i=1:1:countT_ROW
            valueTR = arrayAiAsRPartial( i, countT_COL );
            if maxTR < valueTR
                maxTR = valueTR;
                cumulativeArrayROfWeightTmp(
                cumulativeArrayROfWeightTmp_index ).t =
                arrayAiAsRPartial( i,: );
            end
        end

        mainKlineUniqueEmpty = isempty(mainKlineUnique);

        if length(arrayAiAsRPartial)==0
            mainKlineUniqueEmpty = 1;
            cumulativeArrayROfWeightTmp_index=
            cumulativeArrayROfWeightTmp_index-1;
        end;

        newXprim = cumulativeArrayROfWeightTmp(
        cumulativeArrayROfWeightTmp_index).t(1);
    end;
```

– Moduł Emotiv_Gower_Coordinates.m

Funkcja zwraca współrzędne sferyczne pól P uwzględnione w drzewach minimalnych (całkowitym i po uwzględnieniu progów). Dane wejściowe do funkcji stanowią numery pól P rozpatrywanych przy wyznaczaniu t-statystyki uwzględnione w drzewach minimalnych i waga pomiędzy nimi.

– Moduł Emotiv_Visualisation_GowerGraph.m

Funkcja pozwalająca na wizualizację wyników algorytmu Gowera. Wynikiem wizualizacji jest graf dwudzielny prezentujący połączenia pomiędzy obszarami z lewej i prawej półkuli, wyróżniający najsilniejsze połączenia przy zadanym progu.

– Algorytm wyznaczenia wyniku klasyfikacji na podstawie wag istotnych połączeń pomiędzy obszarami P z poszczególnych obszarów głowy (rozdział 3.5).

- Kod wyznaczania wyniku klasyfikacji:

```
if suma_calkowitedek_2>abs(suma_calkowitedek_3)
    result_spantree = 'classification spantree: LEFT';
    result_spantree_tx = sprintf('%d-%d s, classification
    spantree: LEFT', save_time_i, save_time_j);
    result_arduino = '0';
elseif (suma_calkowitedek_2==0)&&(suma_calkowitedek_3==0)
    result_spantree = 'class spantree : -';
    result_spantree_txt = 'class spantree : -';
    result_arduino = '2';
else
    result_spantree = 'classification spantree: RIGHT';
    result_spantree_txt = sprintf('%d-%d s, classification
    spantree: RIGHT', save_time_i, save_time_j);
    result_arduino = '1';
end
if suma_wezelwaga_2>abs(suma_wezelwaga_3)
    result_spantree_thresh = sprintf( 'class spantree
    thresh  %1.1f: LEFT', prog_value );
    result_spantree_thresh_txt = sprintf( '%d-%d s, class
    spantree thresh  %1.1f: LEFT' , save_time_i,
    save_time_j, prog_value );
elseif (suma_wezelwaga_2==0)&&(suma_wezelwaga_3==0)
    result_spantree_thresh = sprintf( 'class spantree
    thresh %1.1f: - ', prog_value );
    result_spantree_thresh_txt = sprintf( '%d-%d s, class
    spantree thresh %1.1f: - ', save_time_i, save_time_j,
    prog_value );
else
    result_spantree_thresh = sprintf( 'class spantree
    thresh  %1.1f: RIGHT', prog_value );
    result_spantree_thresh_txt = sprintf( '%d-%d s, class
    spantree thresh  %1.1f: RIGHT', save_time_i,
    save_time_j, prog_value );
end
```

– Algorytm przekazania wyniku klasyfikacji w postaci impulsu sterującego do urządzenia zewnętrznego (rys. 21, 22 oraz moduł F5 poniżej).

F5. Przekazanie wyniku klasyfikacji

Wynik klasyfikacji w postaci impulsu sterującego jest przekazywany do urządzenia zewnętrznego (F4 i F6).

- Kod przekazujący impuls sterujący do urządzenia zewnętrznego w trybie Bluetooth lub Wi-fi:

```
if bluetooth_mode == 1 && TrainingMode == 0
    %wyjście na bluetooth
    bt = Bluetooth('HC-05', 1);
    fopen(bt);
    if result_arduino == '0'
        fwrite(bt, 0);
    else
```

```
            fwrite(bt, 1);
        end
        fclose(bt);
        % koniec wyjście na bluetooth
end
if bluetooth_mode == 2 && TrainingMode == 0
  %wyjście na wifi
  t = tcpip(IPSign, 8888)
  fopen(t);
  if result_arduino == '0'
      fwrite(t, '1');
  else
      fwrite(t, 'r');
  end
  fclose(t);
  % koniec wyjście na wifi
    end
```

Funkcje Bluetooth i Wi-fi zwracają deskryptor pliku reprezentującego urządzenia Bluetooth i Wi-fi.

F6. Sygnalizacja wyniku klasyfikacji przez urządzenie zewnętrzne (zapalenie diody zielonej – sygnalizacja klasyfikacji prawej ręki K3, zapalenie diody czerwonej – sygnalizacja klasyfikacji lewej ręki K2).

Impuls sterujący generowany przez program klasyfikujący przekazany do urządzenia zewnętrznego powoduje zapalenie diody zielonej w przypadku K3 i diody czerwonej w przypadku K2.

7. Podsumowanie

W rozprawie wykazano, że wykorzystanie do klasyfikacji intencji ruchu K2 i K3 algorytmu opartego na teorii grafów i na informacjach o źródłach sygnałów EEG (po rozwiązaniu zagadnienia odwrotnego) pozwala na prawidłową klasyfikację tych intencji, mimo że w headsecie Epoc nie ma elektrod na korze ruchowej.

W przeprowadzonych testach wykazano, że dokonując klasyfikacji w systemach BCI należy brać pod uwagę osobę, której intencje ruchu są klasyfikowane. Na etapie nauki klasyfikatora należy zatem wybrać dla danej osoby wyniki o najlepszej trafności (tj. przedziały czasowe oraz częstotliwości).

W rozprawie przedstawiono przegląd metod selekcji cech i klasyfikacji sygnałów EEG z ostatnich kilkunastu lat. W przeanalizowanej literaturze wykorzystanie teorii grafów do klasyfikacji sygnałów EEG było mało zbadane. Autorce nie są znane publikacje, w których wykorzystano by teorię grafów do klasyfikacji sygnałów EEG w oparciu o rozwiązanie zagadnienia odwrotnego, pozwalające uzyskać informacje o źródłach tych sygnałów.

Najważniejsze osiągnięcia autorki to:

- Opracowanie algorytmów, w których wykorzystano drzewo rozpinające (teoria grafów) jako klasyfikator rozpoznawania intencji ruchu lewą i prawą ręką
- Implementacja algorytmów klasyfikacji intencji ruchu w środowisku Matlab.

Dodatkowo autorka:

- Opracowała bliotekę dzieloną (eegloglib.dll) w języku C++ pobierającą surowe (*raw*) dane z elektrod headsetu Emotiv i zapisującą pliki *raw* w podkatalogach odpowiadających kolejnym pomiarom
- Napisała programy sterujące do układu NodeMCU (komunikacja WiFi) oraz do płytki Arduino Uno R3 (komunikacja Bluetooth)
- Opracowała program analizujący wyniki pomiarów w VBA w Excel.

Osiągnięciem autorki jest napisanie oprogramowania w środowiskach Matlab, C++, Wiring (NodeMCU, Arduino Uno R3) do całego systemu BCI od rejestracji danych EEG poprzez rozwiązanie zagadnienia odwrotnego, selekcję, klasyfikację aż do przekazania komendy sterującej do urządzenia zewnętrznego, realizującego tę komendę. Oprogramowanie opracowane dla systemu BCI zostało wykorzystane do wykonania testów.

Przedstawione w rozprawie algorytmy klasyfikacji oparte na teorii grafów w dalszych badaniach mogą zostać wykorzystane do klasyfikacji większej liczby obiektów niż dwa.

W kolejnych zadaniach badawczych autorka planuje opracować algorytmy dla klasyfikatorów opartych na geometrii Riemanna (kierunek aktualnie badany) oraz przeprowadzić testowanie tych klasyfikatorów i dokonać porównania uzyskanych wyników klasyfikacji.

Załącznik 1. Wyprowadzenie wzoru 24 określającego wartość zmierzonego potencjału

Wykorzystując wzory Maxwella (od (16) do (19)) zostało udowodnione, że problem odwrotny ma rozwiązanie.

Wzór (20) ustalający liniową zależność pól $\vec{D}$ i $\vec{E}$ jest szczegółowo uzasadniony w [82, str. 118]. W przypadku tkanki nerwowej pola te nie są duże. W ogólnym przypadku $\vec{D} = \varepsilon\vec{E} + \overrightarrow{P_c}$, gdzie $\overrightarrow{P_c}$ to polaryzacja z powodu ładunków związanych, ε przenikalność elektryczna ośrodka. Po przeprowadzeniu eksperymentów, gdy opisywane pole nie jest duże, można było stosować aproksymację $\vec{D} = \varepsilon\vec{E}$.

Podobnie wzór (21) (szczegółowe uzasadnienie w [82, str. 129]), ustalający liniową zależność pól $\vec{B}$ i $\vec{H}$ można stosować dla tkanki nerwowej która nie jest ferromagnetykiem[20]. W ogólnym przypadku $\vec{B} = \mu\,\vec{H} + \vec{M}$, gdzie $\vec{M}$ to magnetyzacja materiału, μ przenikalność magnetyczna ośrodka. Po zastosowaniu aproksymacji uzyskuje się $\vec{B}$=$\mu\vec{H}$.

Z wzorów Maxwella od wzoru (16) do wzoru (19) można wyznaczyć związek prądu mózgowego z potencjałami na powierzchni głowy.

Pole elektryczne w tkance nerwowej można opisać przy pomocy uproszczonej wersji równań Maxwella. Są to podstawowe równania liniowej elektrofizjologii. Z równań (17) i (18) wynika

$$0 = \nabla\cdot\left(\nabla\times\vec{H}\right) = \nabla\cdot\vec{J} + \nabla\cdot\frac{\partial\vec{D}}{\partial t} = \nabla\cdot\vec{J} + \frac{\partial}{\partial t}\left(\nabla\cdot\vec{D}\right) = \nabla\cdot\vec{J} + \frac{\partial\rho}{\partial t} \tag{63}$$

Wykorzystano tu tożsamość $\nabla\cdot(\nabla\,\mathrm{x}\,\overrightarrow{W}) = 0$ dla dowolnego pola wektorowego $\overrightarrow{W}$ klasy C^2. Zatem zasada zachowania ładunku jest postaci:

$$\nabla\cdot\vec{J} + \frac{\partial\rho}{\partial t} = 0 \tag{64}$$

Jeśli założy się wolną oscylację pól to można zaniedbać indukcję magnetyczną $\frac{\partial\vec{B}}{\partial t} \approx \vec{0}$, wówczas z równania (16) wynika, że $\nabla\times\vec{E} = \vec{0}$, tzn. rotacja pola elektrycznego $\vec{E}$ jest równa zeru, a więc pole $\vec{E}$ jest potencjalne. Istnieje więc potencjał skalarny $\emptyset$ taki, że

$$\vec{E} = -\nabla\emptyset \tag{65}$$

[20] Ferromagnetyki charakteryzują się dużą przenikalnością magnetyczną, występują tylko w postaci ciał stałych. Monokryształy ferromagnetyków mają różne zdolności magnesowania w różnych kierunkach (własność anizotropii). Tkanki są paramagnetykami.

Problem prosty w EEG polega na wyznaczeniu rozkładu potencjałów na powierzchni głowy na podstawie rozkładu gęstości źródeł prądowych w mózgu. Przyjęto założenia:
1^o – ośrodek nie wykazuje efektów pojemnościowych, tzn. jest czysto przewodzący, a zatem równanie zachowania ładunku jest postaci:

$$\nabla \cdot \vec{J} = 0 \tag{66}$$

2^o – jeżeli oznaczone zostanie $\vec{J}^i$ jako makroskopowa gęstość prądu źródeł (*impressed neural current*), prawo Ohma jest postaci:

$$\vec{J} = \sigma\vec{E} + \vec{J}^i \tag{67}$$

gdzie σ – przewodność elektryczne ośrodka.

W (67) prąd całkowity $\vec{J}$ w ośrodku przewodzącym jest sumą prądu Ohma $\sigma\vec{E}$ i prądu źródłowego jaki pojawia się na granicy między ośrodkami. Prąd ten pochodzi ze źródeł prądowych w błonach neuronalnych. Generatorami EEG są właśnie powyższe źródła prądowe, a nie potencjały ładunków które są zaniedbywalnie małe dla makroskopowych odległości.

Z (66) i (67) wynika

$$\nabla \cdot \left(\sigma(r) \cdot \vec{E} + \vec{J}^i\right) = 0 \tag{68}$$

$$\nabla \cdot \left(\sigma(r)\vec{E}\right) = -\nabla \cdot \vec{J}^i(r,t) \tag{69}$$

Jeżeli oznaczy się $s(r,t) = -\nabla \cdot \vec{J}^i(r,t)$, $s(r,t)$ – objętościowy prąd źródłowy, to uwzględniając (65) uzyska się równanie Poissona [82]:

$$\nabla \cdot \left(\sigma(r)\nabla\emptyset(r,t)\right) = -s(r,t) \tag{70}$$

W [82, str. 168] wykazano istnienie rozwiązania $\emptyset$ równania (70) dla przypadku N źródeł prądowych dla jednorodnego, izotropowego i czysto przewodzącego ośrodka. Rozwiązanie to dane jest wzorem (23).

Załącznik 2. Wprowadzenie do tematyki sygnałów generowanych przez mózg

Działanie mózgu jest podstawą codziennej aktywności, zachowań i procesów przebiegających w organizmie. Sygnały wzbudzane w mózgu są podstawą wykonywania różnych czynności, np. chodzenie, jedzenie, myślenie, nauka, mowa, pamięć, działanie twórcze [77]. Aktywność mózgu związana jest z komunikacją komórek nerwowych (neuronów) między sobą [54].

1.1. Ogólna charakterystyka sygnałów EEG[21]

Sygnał EEG jest pomiarem prądów, które przepływają przez neurony w mózgu. Sygnały prądowe są przenoszone pomiędzy neuronami za pomocą impulsu prądowego (potencjału czynnościowego), generowanego przez pojedynczą komórkę nerwową. Sygnały prądowe generowane przez neurony [74] tworzą wokół głowy mierzalne pole elektryczne, które można mierzyć za pomocą systemów EEG. Jest to jedna z technik pozwalająca na obserwowanie pracy mózgu wykorzystywana podczas wykonywania eksperymentów myślowych [54][22].

Systemy EEG bazujące na elektrodach umieszczonych na powierzchni głowy rejestrują modulacje wartości napięcia w dużej skali (wartości rejestrowane to uśrednione wartości dotyczące pewnego obszaru). Zapis EEG to różnica potencjałów pomiędzy daną elektrodą a elektrodą odniesienia. Często jednak wygodnie jest pokazywać różnice potencjałów pomiędzy wybranymi elektrodami. Pojedyncza elektroda zapewnia przybliżenie (potencjału elektrycznego mózgu) aktywności synaptycznej [82] uśrednionego z populacji 100 mln ÷ 1 bln neuronów. Różnica potencjałów pomiędzy dwiema elektrodami zależy od potencjału źródeł oraz od właściwości przewodnictwa głowy. Różnice w rozmiarach elektrod skutkują znaczącymi różnicami w rejestrowanych danych. Elektrody umieszczone na powierzchni głowy rejestrują źródła spójne w skali przynajmniej kilku cm. Skala pomiarowa to obszar, z którego potencjały są przestrzennie uśrednione [82].

Głównym problemem EEG jest odniesienie zmierzonego potencjału do odpowiedniego procesu fizjologicznego. Sygnały EEG zarejestrowane w pojedynczej elektrodzie

[21] Rozdział opracowany na podstawie [82], [54], [79].

[22] Inne techniki obrazowania pracy mózgu to np. MRI (Magnetic Resonance Imaging), PET (Positron Emission Tomography).

uzyskiwane są z wielu źródeł pochodzących z obszarów mózgu nie znajdujących się bezpośrednio pod elektrodą. Konieczne jest stosowanie przetwarzania wstępnego zarejestrowanego sygnału na potrzeby BCI [73].

Elektryczna aktywność mózgu dzieli się na dwie główne kategorie – odruchy i ruch dobrowolny polegający na wykonaniu zaplanowanego działania. Odruchy są odpowiedziami na bodźce środowiskowe, natomiast ruchy dobrowolne mogą być generowane wewnętrzne. Ich efektywność poprawia się wraz z doświadczeniem i na skutek nauki [54].

Sygnały rejestrowane za pomocą elektrod EEG odnoszą się głównie do kory mózgowej. Wielkość rejestrowanego potencjału w dużym stopniu zależy od synchronizacji źródeł [82].

1.2. Sygnały generowane przez komórki mózgowe[23]

Mózg ludzki to sieć ponad 100 bilionów indywidualnych komórek nerwowych połączonych w systemy, które tworzą odczucia świata zewnętrznego, skupiają uwagę i kontrolują działania. Są one zorganizowane w ścieżki przekazujące sygnały i komunikują się między sobą za pomocą transmisji synaptycznej. Komórka nerwowa odbiera sygnały, a także może przekazywać sygnał do innych neuronów lub komórek mięśniowych [54].

Neurony w odróżnieniu od innych komórek ciała mogą nie tylko wytwarzać sygnały prądowe, ale też przesyłać je na odległość. Są to sygnały bioelektryczne, których źródłem jest zmiana potencjału błony komórkowej neuronu. Dzięki możliwości generowania potencjału czynnościowego komórki nerwowe mogą przekazywać wytworzone sygnały na duże odległości. Są to regenerujące się sygnały elektryczne, których powstawanie powodowane jest zmianami przepływu jonów przez kanały napięciowe błony komórkowej. Potencjały czynnościowe składają się na sygnały, przez które mózg otrzymuje, analizuje i przewodzi informacje [54]. Jak podają przywoływane źródła, częstotliwości które te sygnały mogą osiągać znajdują się w paśmie do 100 Hz, a amplitudy znajdują się przeważnie w przedziale od 10 do ok. 100 μV [82].

Charakterystyka sygnału EEG rejestrowanego z elektrod na powierzchni głowy zależy nie tylko od właściwości i lokalizacji źródeł prądowych, ale również od elektrycznych i geometrycznych właściwości mózgu, czaszki oraz skóry. Zarówno głowa, jak i mózg mają

[23] Rozdział opracowany na podstawie [54], [82], [93], [79].

strukturę niejednorodną. Charakteryzują się anizotropowością[24] ośrodka. Długość fali zależy od tkanki, przez którą ona przenika. Pomimo obecnie posiadanej wiedzy o anizotropowości ośrodka, jakim jest mózg, ze względu na niewystarczającą ilość danych pozyskiwanych z przeprowadzonych eksperymentów i złożoność obliczeniową, modele głowy niemal zawsze zakładają nieskończoność, jednorodność i izotropowość ośrodka przewodzącego [82], (rozdział 3.3.1).

Tkanki mają właściwości liniowe w skali makroskopowej[25]. Oznacza to, że komórka jest liniowym przewodnikiem i obowiązuje prawo Ohma. Fakt, że komórka jest liniowym przewodnikiem oznacza, że obowiązuje zasada superpozycji, tj. potencjał w danym punkcie „n" jest liniową superpozycją wszystkich potencjałów ($V_1+V_2+\ldots+V_n$) z działających w tym samym momencie źródeł prądowych ($I_1+I_2+\ldots+I_n$). Przy założeniu nieliniowości, przewodniość błony komórkowej różni się w zależności od napięcia na błonie komórkowej i/lub czasu.

Sygnały elektryczne generowane przez mózg rejestrowane w systemach EEG są odbiciem aktywności elektrycznej Centralnego Układu Nerwowego (CUN)[26]. Neurony[27] przesyłają pomiędzy sobą informacje za pomocą impulsów elektrycznych, tzw. potencjałów czynnościowych (AP[28]). CUN człowieka składa się z dwóch podstawowych typów komórek: komórek nerwowych (neuronów) oraz komórek glejowych znajdujących się pomiędzy neuronami, otaczających je i wspierających. Jest ok. 10 000 różnych typów neuronów, ale podstawowa budowa neuronu jest jednakowa dla wszystkich komórek nerwowych [79]. Można przedstawić model neuronu, który ma cztery podstawowe elementy składowe/regiony generujące cztery typy sygnałów: sygnał wejściowy, sygnał wyzwalający (potencjał czynnościowy), sygnał przewodzący, sygnał wyjściowy. Różne rodzaje aktywności, które mogą wykonywać zależą bardziej od ich połączeń z innymi

[24] Anizotropia – przeciwieństwo izotropii, cecha materiału polegająca na tym, że w zależności od kierunku działania nań wykazuje on różne własności fizyczne (na podstawie [97]).

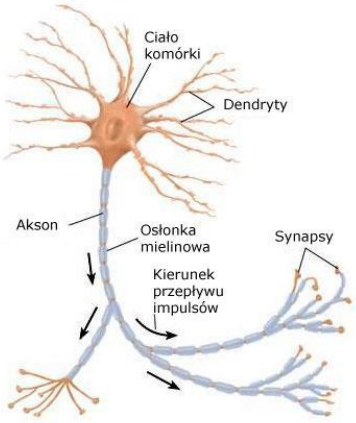
[25] Skala makroskopowa: definiująca obszar, który można podzielić na mniejsze elementy składające się z kolejnych coraz mniejszych elementów, aż do najmniejszego elementu błony komórkowej (na podstawie [82]).

[26] CUN – obustronna, symetryczna struktura składająca się z rdzenia kręgowego (spinal cord), rdzenia przedłużonego (medulla oblongata), mostu mózgu (pons), móżdżku (cerebellum), śródmózgowia (midbrain), międzymózgowia (diencephalon), półkuli mózgu (cerebral hemispheres) (na podstawie [54]).

[27] Neuron – każdy neuron jest dyskretną komórką z charakterystycznymi procesami wynikającymi z ciała komórki, przenoszącą sygnały w układzie nerwowym. Kształt komórek nerwowych jest złożony. Neurony różnią się kształtem i funkcją. Każdy neuron jest wyraźnie oddzielony od pozostałych (na podstawie [54]).

[28] AP – Action Potential (potencjał czynnościowy) – sygnał przewodzący w neuronie [54].

neuronami i komórkami niż od ich typów [79]. Pojedyncza komórka nerwowa – generator sygnału, zbudowana jest z ciała komórkowego zawierającego pojedyncze jądro komórkowe, dendrytów, jednego długiego aksonu, oraz zakończeń presynaptycznych w aksonie. Każdy z tych elementów odgrywa istotną rolę przy generowaniu sygnału i przekazywaniu go pomiędzy komórkami nerwowymi [54]. Sygnał neuronu jest propagowany od dendrytów, poprzez ciało komórki, do zakończeń aksonu. Jeden neuron komunikuje się z drugim poprzez synapsę[29] za pomocą małego impulsu elektrycznego[30]. Sygnały elektryczne na ogół przepływają przez komórkę nerwową tylko w jednym kierunku.



Rysunek 24 Schematyczna budowa neuronu[31]

Dendryty są głównym elementem odbierającym sygnały od innych neuronów. Z kolei akson jest głównym elementem przewodzącym, który wyprowadza sygnał z neuronu. Akson może przenieść sygnał elektryczny na dystans 0,1 mm ÷ 3 m. Sygnały te – potencjały czynnościowe – mają amplitudę do ok. 100 μV i trwają ok. 1 ms. Amplituda potencjału czynnościowego przemieszczającego się w dół aksonu pozostaje stała, ponieważ potencjał czynnościowy jest impulsem zerojedynkowym, który jest regenerowany w regularnych odstępach czasu wzdłuż aksonu [54].

Potencjały czynnościowe formują sygnały, poprzez które mózg odbiera, analizuje i przenosi informacje. Ze względu na podział funkcjonalny wyróżnia się trzy grupy neuronów: czuciowe, ruchowe i interneurony – czyli pozostałe neurony, które nie są czuciowe i ruchowe. Wszystkie behawioralne funkcje mózgu – przetwarzanie informacji

[29] Synapsa jest punktem komunikacji dwóch neuronów – na podstawie [54].
[30]Na podstawie http://www.opencolleges.edu.au/informed/learning-strategies/ - stan aktualny na dzień 09.08.2014 r.
[31] Rysunek 24, na podstawie [98].

czuciowej, programowanie odpowiedzi emocjonalnych i ruchowych, przechowywanie informacji (pamięć) – są przenoszone przez odpowiednie zestawy neuronów [54]. Czas trwania odczucia lub ruchu determinowany jest okresem, w jakim generowany jest potencjał czynnościowy. Intensywność odczuć lub prędkość ruchu zależą od częstotliwości powstawania potencjałów czynnościowych, natomiast nie zależą od wielkości lub czasu trwania. Informacja przenoszona przez potencjał czynnościowy jest określana przez ścieżkę, po której sygnał się porusza. Sygnały, które niosą informację o obrazie, mogą być takie same jak sygnały niosące informację o zapachu. W związku z tym mózg przetwarza informację o drodze przebytej przez sygnał, a nie o kształcie sygnału. Mózg analizuje i interpretuje wzory przychodzących sygnałów w ten sposób, że tworzy codzienne odczucia widzenia, dotyku, smaku, zapachu i dźwięku. Informacje czuciowe i ruchowe są przetwarzane w mózgu za pomocą wielu równoległych ścieżek. Wszystkie czuciowe i ruchowe systemy opierają się na wzorach hierarchicznego i równoległego przetwarzania. Każda ścieżka jest formowana przez kolejne połączenia zidentyfikowanych grup neuronów. Kolejne grupy przetwarzają bardziej złożone informacje. Przykładowo uczucia dotyku i bólu są przetwarzane za pomocą różnych ścieżek połączeń. Kształt, ruch i faktura trzymanego w rękach elementu są analizowane równocześnie za pomocą oddzielnych ścieżek, a rezultaty są integrowane w postaci świadomego odczucia - informacje są przetwarzane sekwencyjnie i równolegle [54]. Aktywność neuronów to aktywność zespołów komórek nerwowych. Do uzyskania mierzalnej wartości potencjału na powierzchni głowy ok. 6 cm^2 komórek kory mózgowej musi być synchronicznie aktywnych.[32]

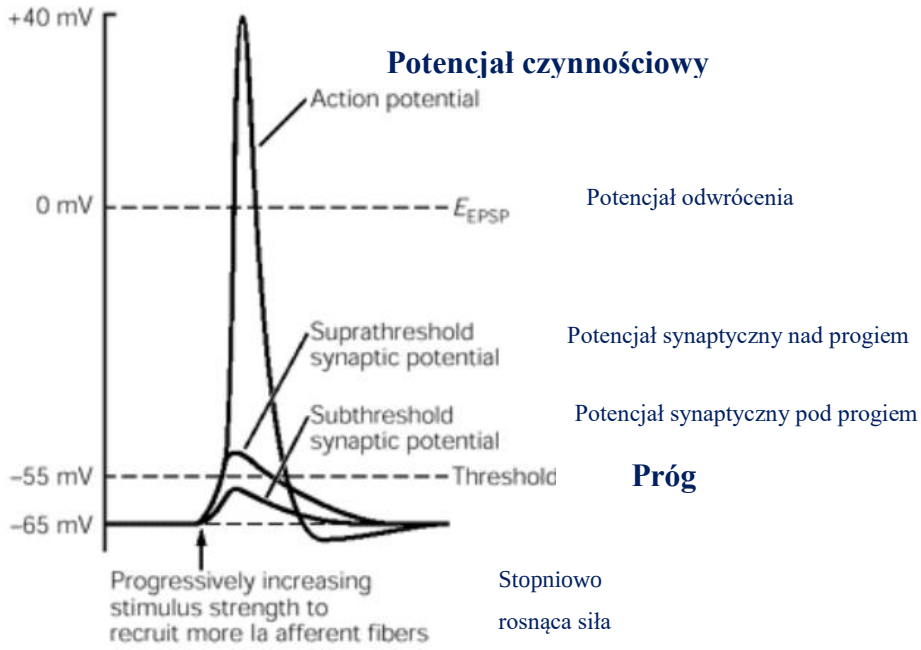
1.3. Potencjał czynnościowy

Aby przemieścić się i przenieść informację sygnał musi zostać wzmocniony, tj. komórka nerwowa musi wytworzyć potencjał czynnościowy. Jeżeli wypadkowy potencjał spoczynkowy przekroczy odpowiedni próg, to generowany jest impuls, który przemieszcza się wzdłuż aksonu – jest to potencjał czynnościowy (AP) [54].

Potencjał czynnościowy jest przewodzony wzdłuż komórki przez akson do zakończeń aksonu – synapsy, która kończy się w innych komórkach (nerwowych lub mięśniowych). Tam potencjał czynnościowy rozpoczyna komunikację komórki z kolejnymi komórkami.

[32] Na podstawie [81].

Potencjał czynnościowy nie wygasza się podczas przemieszczania w dół aksonu. Po ok. 1 ms (czas trwania typowego potencjału czynnościowego) komórka wraca do stanu spoczynkowego.[33]



Rysunek 25 Przebieg potencjału czynnościowego[34]

Po dotarciu do synapsy, impuls jest przekazywany do kolejnego neuronu drogą elektryczną lub drogą chemiczną (w przypadku synapsy chemicznej sygnał elektryczny uwalnia neurotransmiter chemiczny w zakończeniu nerwowym i w ten sposób sygnał przekazywany jest do kolejnej komórki). Jest to transmisja synaptyczna. W ten sposób indywidualne komórki nerwowe i komunikowanie się ich za pomocą synaps powoduje powstawanie prostych odruchów. Przeciętny neuron formuje i otrzymuje ok 10^3 połączeń synaptycznych a mózg ludzki zawiera przynajmniej 10^{11} neuronów. Stąd mózg formuje ok 10^{14} połączeń synaptycznych. Funkcje myślowe wynikają z biologicznych właściwości komórki nerwowej i wzorów połączeń komórek nerwowych. Złożone, połączone ze sobą sieci neuronów powodują powstawanie aktywności myślowych, takich jak: percepcja, działanie, motywacja, język, nauka i pamięć [54].

Komórki nerwowe budują zachowanie wykorzystując następujące zasady: wejście czuciowe, przetwarzanie z użyciem interneuronów, wyjście ruchowe. Poszczególne typy informacji są przetwarzane w odpowiadających im częściach mózgu [54].

[33] Na podstawie [54].
[34] Na podstawie [54] przykładowo dla nerwu czuciowego (tj. aferentnego).

Natężenie sygnału elektrycznego generowanego przez mózg jest osłabiane głównie przez czaszkę, której przybliżona rezystywność wynosi 177 Ωm [93]. Napięcie emitowanego przez mózg sygnału elektrycznego, który jest rejestrowany przez elektrody umieszczone na powierzchni głowy jest niskie (jest określane w mikrowoltach), dlatego żeby można było wykorzystać go do komunikacji z urządzeniem zewnętrznym musi być poddawane znacznemu wzmocnieniu. W swojej pracy autorka wykorzystuje komunikację headsetu Epoc firmy Emotive, który łączy się z komputerem bezprzewodowo poprzez interfejs Bluetooth. Headset Epoc posiada fabrycznie wbudowany wzmacniacz.

1.4. Opis wykorzystanych sygnałów EEG[35]

Rejestracja sygnałów elektrycznych mózgu za pomocą zapisów EEG wskazuje na aktywność konkretnych obszarów kory mózgowej skorelowanej z rodzajem wzbudzanej intencji do wykonania działania. W zależności od wykonywanej aktywności nie tylko obszar kory mózgowej, ale także częstotliwości sygnału generowanego przez mózg ulegają zmianie. Wszystkie obszary mózgu mają określone cechy, które pojawiają się w zależności od stanu jego aktywności. W eksperymencie wykorzystano fale β i fale Mu, czyli częstotliwości odpowiednio od 14 do około 26 Hz i od 8 do 13 Hz[36]. Fale β mają amplitudę poniżej 30μV. Związane są one z zaangażowaniem kory mózgowej w wykonywanie ruchu. Pośrednie rytmy β są powiązane z rytmami Mu i są redukowane podczas podejmowanej aktywności ruchowej lub dotyku - to znaczy wygaszają się podczas wykonywania ruchu a także przygotowania, planowania i wyobrażania ruchu (desynchronizacja[37] jest najbardziej widoczna w półkuli przeciwległej do tej części ciała, która wykazuje aktywność ruchową). Po zakończeniu wykonywania aktywności ruchowej następuje synchronizacja[38] fal β i Mu. Fale β wskazują na stan normalnej świadomości człowieka związanej z aktywnym myśleniem, skupianiem uwagi, koncentracją na świecie zewnętrznym, rozwiązywaniem konkretnych problemów, stanem niepokoju. Wysoki poziom fal β może wskazywać na stan paniki. Rytmiczne aktywności β występują głównie w przednich i centralnych obszarach

[35] Opisy aktywności fal mózgowych zostały wykonane na podstawie: [93], [79], [52], [54].
[36] W różnych źródłach można znaleźć różniące się zakresy częstotliwości fal β i fal Mu.
[37] Desynchronizacja – spadek aktywności. Na podstawie [22].
[38] Synchronizacja – wzrost aktywności. Na podstawie [93].

kory mózgowej[39](fale β związane z ruchem). Fale β są zlokalizowane po obu stronach kory mózgowej, mają rozkład symetryczny.

Oprócz fal β przedmiotem zainteresowania były fale Mu, które towarzyszą falom β. Wygaszają się one w przypadku ruchu, czy też przygotowania ruchu, planowania i wyobrażania sobie ruchu; po zakończeniu ruchu następuje wzrost wartości sygnału elektrycznego mózgu – tak jak w przypadku fal β.

Sygnał EEG składa się z połączenia wielu składowych częstotliwościowych, co staje się widoczne po wykonaniu analizy widmowej[40] (rozdział 3.2). Aby wyznaczyć składowe częstotliwości występujące w sygnale (widmie sygnału) autorka wykorzystała dyskretną transformatę Fouriera (rozdział 3.2).

Jednym z filarów współczesnej nauki o mózgu jest koncepcja mówiąca o tym, że różne obszary mózgu specjalizują się w różnych funkcjach. Przetwarzanie informacji w mózgu zachodzi w sposób równoległy. Na początku XX wieku Korbinian Brodmann[41] podzielił korę mózgową człowieka na 52 obszary zawierające charakterystyczne struktury komórek nerwowych, posiadające charakterystyczne zagospodarowanie warstw komórek. Stworzyło to precyzyjną mapę ciała odniesioną do właściwych obszarów kory mózgowej [54].

1.5. Pola Brodmanna związane z ruchem

Schemat kory mózgowej stworzony przez Brodmanna jest stosowany do dnia dzisiejszego i jest wciąż uaktualniany. Poszczególne obszary zdefiniowane przez Brodmanna zostały określone jako kontrolujące charakterystyczne funkcje mózgu.

Pole 4 – pierwszorzędowa kora ruchowa (*primary motor cortex*) kontroluje proste właściwości ruchu. Nawet najmniejsza stymulacja tego obszaru wywołuje ruch. Pole 4 kontroluje ruchy twarzy, palców, ręki, ramienia, tułowia, nóg i stóp.

Pole 6 – drugorzędowa kora ruchowa (kora przedruchowa (*premotor cortex*) i dodatkowa kora ruchowa (*supplementary motor area*)), odpowiada za ruchy kompleksowe obejmujące znaczne obszary ciała. Aby wywołać ruch potrzebna jest większa stymulacja tego obszaru niż pola 4 [54].

[39] Na podstawie [93].
[40] Na podstawie [82].
[41] Korbinian Brodmann (1868 – 1916) – niemiecki specjalista od neuroanatomii. Podzielił korę mózgową na dyskretne obszary na podstawie szczególnych cech [3].

Obszary kory ruchowej zawierają populacje neuronów, które przekazują informacje z kory mózgowej do pnia mózgu i rdzenia kręgowego. Więcej projekcji do rdzenia kręgowego odbywa się z pierwszorzędowej kory ruchowej, niż z drugorzędowej kory ruchowej. Drugorzędowa kora ruchowa przekazuje dodatkowo, oprócz projekcji do rdzenia kręgowego, informacje do pierwszorzędowej kory ruchowej. Wyjścia z pierwszorzędowej i drugorzędowej kory ruchowej nakładają się na siebie w rdzeniu kręgowym. Stymulacja pierwszorzędowej kory ruchowej przeważnie wywołuje proste ruchy pojedynczych stawów. Stymulacja drugorzędowej kory wywołuje natomiast bardziej złożone ruchy angażujące wiele stawów np. ruchy sięgania ręką. Stymulacja środkowych części pola 6 daje początek ruchom po obu stronach ciała [54].

Te same mięśnie mogą być aktywowane z różnych części kory mózgowej. Pojedyncze aksony rozdzielają się do więcej niż jednego mięśnia [54]. Półkule mózgu kontrolują procesy po przeciwnych stronach ciała. Prawa półkula kontroluje ruchy w lewej połowie ciała i odwrotnie. Półkule nie są identyczne w wyglądzie, nie mają identycznej struktury i identycznych funkcji [54]. Każdy obszar kory ruchowej posiada wyjątkowy wzór korowych i podkorowych wejść. Stąd wynika duża liczba sprzężeń z korą mózgową i obszarami podkorowymi, z których każde ma inny wkład do zachowania ruchowego [54]. Charakterystyczną cechą sekwencji dobrowolnych ruchów jest poprawa ich wykonywania (zwiększenie dokładności i prędkości) wynikająca z powtarzania. Na skutek powtarzania następują zmiany na korze mózgowej polegające na powiększeniu obszaru aktywującego się przy wykonywaniu sekwencji ruchu praktykowanego przez okres czasu w stosunku do nowej sekwencji ruchu. Wynika to z plastyczności mózgu[42]. Na skutek nauki ustala się minimalna liczba ścieżek potrzebnych do wykonania danego ruchu, następuje również zmiana obszarów mózgu odpowiadających za dane parametry ruchu [54].

Drugorzędowa kora ruchowa jest związana z przygotowaniem ruchu. Wykonanie ruchu w myślach ma podobny przebieg czasowy do rzeczywistego wykonania ruchu. Grupy neuronów ruchowych aktywowane w wyniku danej aktywności ruchowej zmieniają się, kiedy ruch staje się automatyczny [54].

Aktywność wywołana bodźcem odbywa się w pierwszorzędowej i drugorzędowej korze ruchowej. Aktywność w pierwszorzędowej korze ruchowej określa parametry danego

[42] Plastyczność mózgu – zmiany organizacji kory mózgowej będące następstwem nauki ruchu bądź uszkodzeń mózgu (uszkodzeń wejść czuciowych) związanych z tworzeniem nowych bądź zanikiem istniejących połączeń pomiędzy neuronami – zwiększenie/zmniejszenie obszaru reprezentującego dany obszar na korze mózgowej (na podstawie [54]).

ruchu. Aktywność w drugorzędowej korze ruchowej odnosi się do kolejności odpowiedzi na dany bodziec. Drugorzędowa kora ruchowa jest aktywna w przypadku nauki odpowiedzi na dany bodziec [92]. Staje się coraz mniej aktywna na skutek tej nauki. Kiedy odpowiedź na bodziec staje się automatyczna, aktywność drugorzędowej kory ruchowej ustaje [54]. Autorka w swojej pracy wykorzystała bodziec słuchowy.

Załącznik 3. Ostateczne wyniki BCI Competition III – Data Set V

Miarą wydajności klasyfikacji jest dokładność klasyfikacji. Pierwsza kolumna pokazuje średnią dla trzech osób testowanych, kolumny od 2 do 4 pokazują wyniki dla poszczególnych osób. Kolumna zatytułowana „psd" wskazuje, czy w testowaniu wykorzystano PSD (y), czy nie wykorzystano PSD (n).

Tabela 17 Wyniki BCI Competition III (Data Set V)

#.	contributor	psd	acc	s1	s2	s3	research lab	co-contributors
1.	Ferran Galan	y	**68.65**	79.60	70.31	56.02	University of Barcelona	Francesc Oliva, Joan Guardia
2.	Xiang Liao	y	**68.50**	78.08	71.66	55.73	University of Electronic Science and Technology of China (UESTC)	Yu Yin, Dezhong Yao
3.	Walter	y	**65.90**	77.85	66.36	53.44	???	
4.	Xiaomei Pei	y	**65.67**	76.03	69.36	51.61	Institute of Biomedical Engineering of Xi'an Jiaotong University	Guangyu Bin, Chongxun Zheng
5.	Irene Sturm	y	**64.91**	78.08	63.83	52.75	Fraunhofer FIRST (IDA), Berlin	Guido Dornhege
6.	Stephan Uray	y	**64.60**	81.05	73.04	39.68	TU Graz	
7.	Julien Kronegg	y	**64.04**	76.06	64.83	51.18	University of Geneva	Douglas Rofes
8.	John Q. Gan	y	**63.91**	77.40	63.83	50.46	University of Essex, Colchester	Louis C.S. Tsui
9.	Shiliang Sun	n	**62.83**	74.31	62.32	51.99	Tsinghua University, Beijing	Changshui Zhang, Jie Pan
10.	J. Ignacio Serrano M. D. del Castillo	y	**62.61**	75.80	61.75	50.23	Instituto de Automatica Industrial. CSIC. Madrid	
11.	Changshui Zhang	y	**60.47**	72.15	59.22	50.00	Tsinghua University, Beijing	Shiliang Sun, Feiping Nie
12.	Douglas Rofes	y	**59.81**	72.52	59.85	46.99	University of Geneva	
13.	Alois Schloegl	n	**52.71**	69.00	57.05	32.29	TU Graz	Carmen Vidaurre
14.	Ehsan Arbabi	n	**50.25**	55.41	51.79	43.61	Sharif University of Technology	Mohammad Bagher Shamsollahi
15.	Remy Lehembre	y	**50.23**	72.60	46.31	31.65	Universite Catholique de Louvain-la-Neuve (UCL-Belgium)	Simon Cedric
16.	Georgios Lappas	y	**45.72**	71.78	33.81	31.39	Technological Educational Institution (TEI) of Western Macedonia, University of Hertfordshire	Andreas Albrecht
17.	Mohammad Bagher Shamsollahi	y	**44.97**	71.46	32.52	30.76	Sharif University of Technology	Ehsan Arbabi
18.	Ikaro Silva	y	**30.68**	38.98	27.45	25.55	???	
19.	Ali Salehi	n	**27.97**	26.54	32.84	24.53	???	
20.	Ikaro Silva2	y	**14.24**	5.82	10.54	26.38	???	

Źródło: Idiap Research Institute Data Set V

Załącznik 4. Program klasyfikujący – pliki składowe programu Matlab

Plik „Emotiv_Main_StartProgramFile.m" jest głównym plikiem, od którego rozpoczyna się program klasyfikujący.

Plik „Emotiv_Main_StartProgramFile.m"

```
clear all
close all
global EndOfProgram;
global hTextWarning;
global resultWnd;
global DatapackageCounter;
global AllSumDatas;
global SumDataIndex;
global selectedTimeIntervals;
global selectedTrainingMode;
global selectedFrequency;
global selectedHand;
global selectedRadio;
global RdiameterSize;
global hText1;
global hText2;
global hText3;
global hText4;
global hText5;
global hText6;
global hTextWarning;
%% Ładuje biblioteke C++, która czyta dane surowe z headsetu
loadlibrary('eegloglib')

% główna ścieżka zapisu
global main_path;

main_path = 'BCI Results';
rawdata_folder = sprintf('%s\\raw_data', main_path);
mkdir( rawdata_folder );
PSDdata_folder = sprintf('%s\\PSDdata', main_path);
mkdir ( PSDdata_folder );
classified_data_report_folder = sprintf('%s\\classified_data_report',
main_path);
mkdir ( classified_data_report_folder );
% domyślnie włączone
selectedTimeIntervals = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
selectedRadio = 'None';
selectedTrainingMode = 'Recognition';
selectedFrequency = '12 Hz';
selectedHand = 'None';
RdiameterSize = 65;
DatapackageCounter = 0;
% index w tablicy AllSumDatas
SumDataIndex = 1;
AllSumDatas = [];
EndOfProgram = 0;
%% ResultWnd - identyfikator (uchwyt) okna
%Tworzenie figury i wyłączanie osi i menu
resultWnd = figure('name','BCI control panel');
axis off;
set(resultWnd, 'MenuBar', 'none');
```

```
set(resultWnd, 'ToolBar', 'none');
set(resultWnd, 'NumberTitle', 'off');
%koniec: Tworzenie figury i wyłączanie osi i menu


%Tworzenie przycisków Report, Start, Load Data From File, Stop
uicontrol(  resultWnd,...
            'Style', 'pushbutton',...
            'String', 'Report',...
            'Position', [10 100 115 40],...
            'Callback', @ButtonReport);        % Pushbutton string
callback - callback to funkcja wywoływana po naciśnięciu przycisku
"Report"
                                               % that calls a MATLAB
function
uicontrol(  resultWnd,...
            'Style', 'pushbutton',...
            'String', 'Start',...
            'Position', [150 100 170 40],...
            'Callback', @ButtonStart);         % Pushbutton string
callback - callback to funkcja wywoływana po naciśnięciu przycisku
"Start"
                                               % that calls a MATLAB
function
uicontrol(  resultWnd,...
            'Style', 'pushbutton',...
            'String', 'Load data from file',...
            'Position', [10 150 115 30],...
            'Callback', @ButtonLoadDataFromFile);         % Pushbutton
string callback - callback to funkcja wywoływana po naciśnięciu przycisku
"LoadDataFromFile"
                                               % that calls a MATLAB
function
uicontrol(  resultWnd,...
            'Style', 'pushbutton',...
            'String', 'Stop',...
            'Position', [325 100 150 40],...
            'Callback', @ButtonStop);          % Pushbutton string
callback - callback to funkcja wywoływana po naciśnięciu przycisku "Stop"
                                               % that calls a MATLAB
%koniec: Tworzenie przycisków Report, Start, Load Data From File, Stop
hTextWarning = 0;
%% Tworzenie przycisku zmainy częstotliwości
h = uibuttongroup(  'Units', 'pixels',...
                    'parent', resultWnd,...
                    'visible', 'off', 'Position', [0 0 .2 1]);
% Create three radio buttons in the button group.
u0 = uicontrol('Style','Radio','String','12 Hz',...
    'pos',[10 250 60 30],'parent', h,'HandleVisibility','off');
u1 = uicontrol('Style','Radio','String','20 Hz',...
    'pos',[10 220 60 30],'parent', h,'HandleVisibility','off');
u2 = uicontrol('Style','Radio','String','22 Hz',...
    'pos',[10 190 60 30],'parent', h,'HandleVisibility','off');
set(h,'SelectionChangeFcn', @ButtonSelectedFrequency);
set(h,'Visible','on');
%koniec: Tworzenie przycisku zmainy częstotliwości
%%Tworzenie przycisku "Training"
hTraining = uibuttongroup(  'Units', 'pixels',...
                    'parent', resultWnd,...
                    'visible', 'off', 'Position', [0 0 .2 1]);
```

```
% Create three radio buttons in the button group.
u0 = uicontrol('Style','Radio','String','Recognition',...
    'pos',[10 35 80 30],'parent', hTraining,'HandleVisibility','off');
u1 = uicontrol('Style','Radio','String','Training',...
    'pos',[10 5 80 30],'parent', hTraining,'HandleVisibility','off');
set(hTraining,'SelectionChangeFcn', @ButtonTraining);
set(hTraining,'Visible','on');
%koniec: Tworzenie przycisku "Training"
%% Tworzenie przycisków zmainy chwili czasowej
hTimeInterval = uibuttongroup(  'Units', 'pixels',...
                    'parent', resultWnd,...
                    'visible', 'off', 'Position', [0 0 .2 1]);
% Create two radio buttons in the button group.
u0 = uicontrol('Style','checkbox','String','2-4',...
    'pos',[100 40 55 20],'parent',
hTimeInterval,'HandleVisibility','off');
u1 = uicontrol('Style','checkbox','String','4-6',...
    'pos',[170 40 55 20],'parent',
hTimeInterval,'HandleVisibility','off');
u2 = uicontrol('Style','checkbox','String','6-8',...
    'pos',[240 40 55 20],'parent',
hTimeInterval,'HandleVisibility','off');
u3 = uicontrol('Style','checkbox','String','8-10',...
    'pos',[310 40 55 20],'parent',
hTimeInterval,'HandleVisibility','off');
u4 = uicontrol('Style','checkbox','String','10-12',...
    'pos',[380 40 55 20],'parent',
hTimeInterval,'HandleVisibility','off');

hCheckBoxGroup = [u0, u1, u2, u3, u4];
set(hCheckBoxGroup, 'Callback', {@ButtonSelectedTimeInterval,
hCheckBoxGroup});
set(hTimeInterval,'Visible','on');
%koniec: Tworzenie przycisków zmainy chwili czasowej
%% Tworzenie przycisku zmainy bluetooth/wifi/none
hHand = uibuttongroup(  'Units', 'pixels',...
                    'parent', resultWnd,...
                    'visible', 'off', 'Position', [0 0 .2 1]);
% Create two radio buttons in the button group.
u2 = uicontrol('Style','Radio','String','None',...
    'pos',[10 320 70 30],'parent', hHand,'HandleVisibility','off');
u0 = uicontrol('Style','Radio','String','Bluetooth',...
    'pos',[10 380 70 30],'parent', hHand,'HandleVisibility','off');
u1 = uicontrol('Style','Radio','String','Wifi',...
    'pos',[10 350 70 30],'parent', hHand,'HandleVisibility','off');
set(hHand,'SelectionChangeFcn', @ButtonSelectedBluetooth);
set(hHand,'Visible','on');
%koniec: Tworzenie przycisku zmainy bluetooth/wifi/none
%% Tworzenie przycisku zmainy oznaczenia ręki
hHand = uibuttongroup(  'Units', 'pixels',...
                    'parent', resultWnd,...
                    'visible', 'off', 'Position', [0 0 .2 1]);
% Create two radio buttons in the button group.
u2 = uicontrol('Style','Radio','String','None',...
    'pos',[490 40 60 40],'parent', hHand,'HandleVisibility','off');
u0 = uicontrol('Style','Radio','String','L',...
    'pos',[490 100 60 40],'parent', hHand,'HandleVisibility','off');
u1 = uicontrol('Style','Radio','String','R',...
    'pos',[490 70 60 40],'parent', hHand,'HandleVisibility','off');
set(hHand,'SelectionChangeFcn', @ButtonSelectedHand);
set(hHand,'Visible','on');
```

```
%koniec: Tworzenie przycisku zmainy LLL/RRR
%% tworzenie TextBox, w któym nadaje się nazwę raportu:
hText = uicontrol('Style','text',...
                'Position',[10, 75, 135 20],...
                'String','Report title:');

uicontrol('Style', 'edit', ...
          'Position', [150, 75, 250, 20], ...
          'FontSize', 10, ...
          'Callback', @EditReportFileName, ...
          'BackgroundColor', 'white',...
          'HorizontalAlignment', 'left');
% koniec: tworzenie TextBox, w któym nadaje się nazwę raportu:

%% tworzenie TextBox w któym podaje się promień głowy:
uicontrol('Style','text',...
                'Position',[100, 10, 30 20],...
                'String','R:');

hEditText = uicontrol('Style', 'edit', ...
          'Position', [135, 10, 40, 20], ...
          'FontSize', 10, ...
          'Callback', @EditRdiameter, ...
          'BackgroundColor', 'white',...
          'HorizontalAlignment', 'left');

uicontrol('Style','text',...
                'Position',[180, 10, 270 20],...
                'String','(head radius - put another value if different
than 65mm)');
set(hEditText, 'string', '65');
% koniec: tworzenie TextBox, w któym podaje się promień głowy
%% tworzenie TextBox, w któym podaje IP Wifi:
uicontrol('Style','text',...
                'Position',[10, 290, 30, 20],...
                'HorizontalAlignment', 'left',...
                'String','IP:');

hEditText = uicontrol('Style', 'edit', ...
          'Position', [45, 290, 80, 20], ...
          'FontSize', 10, ...
          'Callback', @EditIP, ...
          'BackgroundColor', 'white',...
          'HorizontalAlignment', 'left');

% koniec: tworzenie TextBox, w któym IP wifi
% Określenie obszarów do wyświetlania komunikatów tekstowych
hTextWarning = text(0.3, 0.4, '' ); % informacja o rozpoczęciu nagrywania
-  "Start of recording in ... sec" i nagrywaniu "recording"
hText1 = text(0.25, 0.7, '' ); % 1) wyświetlanie wyniku klasyfikacji -
drzewo minimalne całkowite;
hText2 = text(0.17, 0.6, '' ); % wyświetlanie wyniku klasyfikacji -
drzewo minimalne z progiem
hText3 = text(0.35, 0.7, '' ); % informacja o zatrzymaniu programu -
"Program stopped"
hText4 = text(0.29, 0.7, '' ); % wyświetlanie, który plik jest wczytywany
- przy czytaniu z pliku oraz komunikatu "End reading data from file";
hText5 = text(0.3, 0.7, ''); % komunikaty dot. generowania raportu
hText6 = text(0.25, 0.7, ''); % komunikaty dot. zakończenia generowania
raportu
% koniec: Określenie obszarów do wyświetlania komunikatów tekstowych
```

Plik „ButtonStart.m”

```
function ButtonStart(PushButton, EventData)
    global EndOfProgram;
    global hTextWarning;
    global hText1;
    global hText2;
    global hText3;
    global hText6;
    global resultWnd;
    global AllSumDatas;
    global SumDataIndex;
    global DatapackageCounter;
    global selectedTrainingMode;
    global selectedTimeIntervals;
    global selectedFrequency;
    global selectedHand;
    global selectedRadio;
    global bt;
    global main_path; % główna ścieżka zapisu
    global IPSign;
    if selectedTimeIntervals ~= 1
        errordlg('You have to select time interval','Error');
        return;
    end
    DatapackageCounter = DatapackageCounter + 1;
    % ile pomiarów w ramach jednego naciśnięcia ButtonStart
    dataCounter = 1;
    dataPackageFileCounter = 1;
    EndOfProgram = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%% Pętla wykonywana dokąd nie naciśnie się
STOP
    while EndOfProgram ~= 1
        % Zerowanie obszarów do wyświetlania komunikatów tekstowych
        set(hTextWarning, 'String', '' );
        set(hText1, 'String', '' );
        set(hText2, 'String', '' );
        set(hText3, 'String', '' );
        set(hText6, 'String', '' );
        % koniec: Zerowanie obszarów do wyświetlania komunikatów
tekstowych
%% I
        if strcmp(selectedTrainingMode, 'Training') == 1
           TrainingMode = 1;
        end
        if strcmp(selectedTrainingMode, 'Recognition') == 1
           TrainingMode = 0;
        end
        if TrainingMode == 1
            title('Command/Control: TRAINING');
        end
        if TrainingMode == 0
            title('Command/Control: RECOGNITION');
        end
        ShowMessages = 1;
%% II
        [y, Fs] = wavread('Concrete-Garage-Avenue--Hand-Clap-Sample--
(Schoeps-mk2-omni).wav');
        for i=3:-1:1
```

```
            time = sprintf( 'start of recording in: %d sec', i );
            set(hTextWarning, 'String', time );
            drawnow;
            sound(3*y, Fs);
        end
        set(hTextWarning, 'String', '              recording' );
%% III
        time_row_i = [];
        time_row_j = [];
        time_sec = {};
        if selectedTimeIntervals(1) == 1
           time_row_i = [time_row_i; 2];
           time_row_j = [time_row_j; 4];
           time_sec = [time_sec '2-4s'];
        end

        if selectedTimeIntervals(2) == 1
           time_row_i = [time_row_i; 4];
           time_row_j = [time_row_j; 6];
           time_sec = [time_sec '4-6s'];
        end

        if selectedTimeIntervals(3) == 1
           time_row_i = [time_row_i; 6];
           time_row_j = [time_row_j; 8];
           time_sec = [time_sec '6-8s'];
        end

        if selectedTimeIntervals(4) == 1
           time_row_i = [time_row_i; 8];
           time_row_j = [time_row_j; 10];
           time_sec = [time_sec '8-10s'];
        end

        if selectedTimeIntervals(5) == 1
           time_row_i = [time_row_i; 10];
           time_row_j = [time_row_j; 12];
           time_sec = [time_sec '10-12s'];
        end
%% IV
    if strcmp(selectedRadio, 'Wifi') == 1
        bluetooth_mode = 2;
    elseif strcmp(selectedRadio, 'Bluetooth') == 1
        bluetooth_mode = 1;
    else
        bluetooth_mode = 0;
    end

    if bluetooth_mode == 1
        bt = Bluetooth('HC-05', 1);
        %fopen(bt);
    end
%% V
    if strcmp(selectedFrequency, '12 Hz') == 1
       freq_value = 3;
    end
    if strcmp(selectedFrequency, '20 Hz') == 1
       freq_value = 7;
    end
    if strcmp(selectedFrequency, '22 Hz') == 1
```

```
        freq_value = 8;
    end
%% VI
        if strcmp(selectedHand, 'L') == 1
           mark_LR = 'L';
        end
        if strcmp(selectedHand, 'R') == 1
           mark_LR = 'R';
        end
        if strcmp(selectedHand, 'None') == 1
           mark_LR = 'None';
        end
 %% VII
        % Funkcja do normalnej pracy oraz nauki kalsyfikatora
        sum_data = Emotiv_Main_Module ( resultWnd, freq_value, mark_LR,
dataPackageFileCounter, DatapackageCounter, dataCounter, TrainingMode,
time_row_i, time_row_j, time_sec, bluetooth_mode, ShowMessages);
%% VIII
        AllSumDatas(SumDataIndex).sum_data = sum_data;
        SumDataIndex = SumDataIndex + 1;
        dataCounter = dataCounter + 1;
        dataPackageFileCounter = dataPackageFileCounter + 1;
        if( isempty(hText1) == 0 )
            set(hText1, 'String', num2str('') );
        end
        if( isempty(hText1) == 0 )
            set(hText1, 'String', num2str('') );
        end
    end
%%%%%%%%%%%%%%%%%%%%%% koniec: Pętla wykonywana dokąd nie naciśnie się
STOP
 %% IX
    if EndOfProgram == 1
        if( isempty(hText3) )
            hText3 = text(0.35, 0.7, 'Program stopped' );
        else
            set(hText1, 'String', ' ' );
            set(hText3, 'String', 'Program stopped' );
        end
        set(hText2, 'String', ' ' );
    end
end
```

Plik „ButtonStop.m”

```
function ButtonStop(PushButton, EventData)
    global EndOfProgram;
    EndOfProgram = 1;
End
```

Plik „ButtonReport.m”

```
function ButtonReport(PushButton, EventData)
    global AllSumDatas;
    global SumDataIndex;
    global hText1;
    global hText2;
    global hText3;
    global hText4;
    global hText5;
    global hText6;
    global hTextWarning;
```

```
    global main_path;
    set(hText1, 'String', ' ' );
    set(hText2, 'String', ' ' );
    set(hText3, 'String', ' ' );
    set(hTextWarning, 'String', ' ' );
    set(hText4, 'String', ' ' );
    set(hText5, 'String', 'Report generation' );
    drawnow; pause(.1);
    ReportGeneration(AllSumDatas);
    % wyczyszczenie raportów
    SumDataIndex = 1;
    AllSumDatas = [];
    set(hText5, 'String', '' );
    set(hText6, 'String', 'End of report generation' );
end
```

Plik „ButtonLoadDataFromFile.m”

```
function ButtonLoadDataFromFile(PushButton, EventData)
    global EndOfProgram;
    global hTextWarning;
    global hText1;
    global hText2;
    global hText4;
    global hText3;
    global hText6;
    global resultWnd;
    global AllSumDatas;
    global SumDataIndex;
    global selectedTrainingMode;
    global selectedTimeIntervals;
    global selectedFrequency;
    global selectedHand;
    global selectedRadio;
    global bt;
    global main_path;
    DatapackageCounter = 1;
    % ile pomiarów w ramach jednego naciśnięcia ButtonStart
    dataCounter = 1;
    EndOfProgram = 0;
%% pętla wykonywana po naciśnięciu przycisku START
    while EndOfProgram ~= 1
        % Zerowanie obszarów do wyświetlania komunikatów tekstowych
        set(hTextWarning, 'String', '' );
        set(hText4, 'String', '' );
        set(hText3, 'String', '' );
        set(hText6, 'String', '' );
        set(hText2, 'String', '' );
        % koniec: Zerowanie obszarów do wyświetlania komunikatów
tekstowych
%% I
        if strcmp(selectedTrainingMode, 'Training') == 1
           TrainingMode = 1;
        end
        if strcmp(selectedTrainingMode, 'Recognition') == 1
           TrainingMode = 0;
        end
        title('Reading data from the file');
        ShowMessages = 1;
%% IV
    if strcmp(selectedRadio, 'Wifi') == 1
        bluetooth_mode = 2;
```

```
    elseif strcmp(selectedRadio, 'Bluetooth') == 1
        bluetooth_mode = 1;
    else
        bluetooth_mode = 0;
    end
    if bluetooth_mode == 1
        bt = Bluetooth('HC-05', 1);
    end
 %% VII
        % Funkcja do normalnej pracy oraz nauki kalsyfikatora
        dataPackageraw_dataCatalogNameCounter = sprintf
('%s\\raw_data\\raw_data%d', main_path, DatapackageCounter);
        dataPackagePSDCatalogName = sprintf ('%s\\PSDdata\\PSD%d',
main_path, DatapackageCounter);
        sum_data =
Emotiv_Main_Classification_ReadFromFile(dataPackageraw_dataCatalogNameCo
unter, resultWnd, dataPackagePSDCatalogName, dataCounter, TrainingMode,
bluetooth_mode, ShowMessages);
        if( isempty(sum_data) )
            set(hText4, 'String', 'End of reading data from the file' );
        else
  %% VIII
            AllSumDatas(SumDataIndex).sum_data = sum_data;
            SumDataIndex = SumDataIndex + 1;
            dataCounter = dataCounter + 1;
            DatapackageCounter = DatapackageCounter + 1;

            set(hText1, 'String', ' ' );
            set(hText2, 'String', ' ' );
            set(hText4, 'String', 'End of reading data from the file' );
            break;
        end
        if( isempty(hText4) == 0 )
            set(hText4, 'String', num2str('') );
        end
    end
%% koniec: pętla wykonywana po naciśnięciu przycisku START
%% IX
    if EndOfProgram == 1
        if( isempty(hText3) )
            hText3 = text(0.35, 0.7, 'Program stopped' );
        else
            set(hText1, 'String', ' ' );
            set(hText3, 'String', 'Program stopped' );
        end
    end
end
```

Plik „ButtonSelectedBluetooth.m”

```
function ButtonSelectedBluetooth(source, eventdata)
    global selectedRadio;
    selectedRadio = get(eventdata.NewValue, 'String');
end
```

Plik „ButtonTraining.m”

```
function ButtonTraining(source, eventdata)
    global selectedTrainingMode;
    selectedTrainingMode = get(eventdata.NewValue, 'String');
end
```

Plik „ButtonSelectedFrequency.m”

```
function ButtonSelectedFrequency(source, eventdata)
    global selectedFrequency;
    selectedFrequency = get(eventdata.NewValue, 'String');
end
```

Plik „ButtonSelectedHand.m”

```
function ButtonSelectedHand(source, eventdata)
    global selectedHand;
    selectedHand = get(eventdata.NewValue, 'String');
end
```

Plik „ButtonSelectedTimeInterval.m”

```
function ButtonSelectedTimeInterval(hSrc, hEvent, hButtonGroup)
    global selectedTimeIntervals;
    selectedTimeIntervals = (get(hButtonGroup, 'Value'))';
    selectedTimeIntervals = cell2mat(selectedTimeIntervals);
end
```

Plik „Emotiv_Main_Classification_ReadFromFile”

```
function summary_data =
Emotiv_Main_Classification_ReadFromFile(dataPackageraw_dataCatalogNameCo
unter, resultWnd, dataPackagePSDCatalogName, DataCounter, TrainingMode,
bluetooth_mode, ShowMessages)
index_summary = 1;
summary_row = 1;
dataPackageFileCounter = 1;
global ReportFileNameSign;
global hTextWarning;
global main_path;
global bt;
reportPSDFileName = sprintf('PSD%d_rawdatapackage', DataCounter);
txtFilesDirectory = sprintf('%s\\*.txt',
dataPackageraw_dataCatalogNameCounter);
filesInDir = dir( txtFilesDirectory );
numberOfFiles = size( filesInDir, 1 );
% sortowanie plików
filesInDir = natsortfiles({filesInDir.name});
if( numberOfFiles == 0 )
    summary_data = [];
else
    mkdir( dataPackagePSDCatalogName );
    for i=1 : 1 : numberOfFiles
        fullFileName = filesInDir(i);
        [filePath, fileName, fileExt] = fileparts(char(fullFileName));
        dataPackageRawFileNameCounter = sprintf('%s\\%s.txt',
dataPackageraw_dataCatalogNameCounter, fileName);
        % nazwa pliku z ustawieniami
        dataSetingsFileNameCounter = sprintf('%s\\%s.mat',
dataPackageraw_dataCatalogNameCounter, fileName);
        load( dataSetingsFileNameCounter, 'measurementSettings' );
        f = measurementSettings.f;
        mark_LR = measurementSettings.mark_LR;
        time_sec = measurementSettings.time_sec;
        freq_value = 3;
        if strcmp(f, '12 Hz') == 1
```

```
            freq_value = 3;
        end
        if strcmp(f, '20 Hz') == 1
            freq_value = 7;
        end
        if strcmp(f, '22 Hz') == 1
            freq_value = 8;
        end
        time_row_i = [];
        time_row_j = [];
        for time = 1:1:length(time_sec)
            time_interval = char(time_sec(time));
            if strcmp( '2-4s', time_interval ) == 1
                time_row_i = [time_row_i; 2];
                time_row_j = [time_row_j; 4];
            end
            if strcmp( '4-6s', time_interval ) == 1
                time_row_i = [time_row_i; 4];
                time_row_j = [time_row_j; 6];
            end
            if strcmp( '6-8s', time_interval ) == 1
                time_row_i = [time_row_i; 6];
                time_row_j = [time_row_j; 8];
            end
            if strcmp( '8-10s', time_interval ) == 1
                time_row_i = [time_row_i; 8];
                time_row_j = [time_row_j; 10];
            end
            if strcmp( '10-12s', time_interval ) == 1
                time_row_i = [time_row_i; 10];
                time_row_j = [time_row_j; 12];
            end
        end
        % Nagrane raw data są przetwarzane - PSD i przenoszone do
katalogu data!!!
        dataPackagePSDFileName = sprintf('%s\\PSD%d_rawdatapackage.txt',
dataPackagePSDCatalogName, dataPackageFileCounter);
        Emotiv_Main_PSD(dataPackageRawFileNameCounter,
dataPackagePSDFileName);
        dataPackageFileCounter = dataPackageFileCounter + 1;
        directory_save = sprintf('%s\\classified_data_rep', main_path);
        LLL = [ 13, 24, 25, 36 ];
        RRR = [ 18, 19, 30, 31 ];
        sign = ReportFileNameSign;
        side = sprintf('1 side_%s', f);
        if TrainingMode == 1
            prog_value = [0.5, 0.7, 0.9];
        else
            prog_value = [0.5];
        end
        macierz_A = { 'A', 'F', 'K', 'P', 'U', 'Z'};
        macierz_B = { 'B', 'G', 'L', 'Q', 'V', 'AA' };
        macierz_C = { 'C', 'H', 'M', 'R', 'W', 'AB' };
        macierz_D = { 'D', 'I', 'N', 'S', 'X', 'AC'};
        macierz_E = { 'E', 'J', 'O', 'T', 'Y', 'AD'};
        prog_index = 0;
        counter_prog_index = 5;
        for index = 1:1:length(prog_value)
            prog_index = prog_index + 1;
            for time = 1:1:length(time_row_i)
                time_interval = char(time_sec(time));
```

```
                frequence = sprintf('_12Hz1s13,24,25,36,..._%s',
time_interval);
                save_time_i = time_row_i(time);
                save_time_j = time_row_j(time);
                A = char(macierz_A(time));
                B = char(macierz_B(time));
                C = char(macierz_C(time));
                D = char(macierz_D(time));
                E = char(macierz_E(time));
                row1 = 1;
                row2 = 2;
                row4 = 4;
                [~, fileName, fileExt] =
fileparts(dataPackagePSDFileName);
                directoryNamePath = sprintf( '%s\\results_%s%s_%s',
directory_save, fileName, frequence );
                set(hTextWarning, 'String', '              ' );
                [X, mark_name, subject_mark, hand_mark,
suma_calkowitedek_3, suma_calkowitedek_2, suma_wezelwaga_3,
suma_wezelwaga_2, result_spantree, result_spantree_thresh,
result_arduino] = Emotiv_Main_Classification(bt, resultWnd,
dataPackagePSDFileName, directoryNamePath, fileName, fileExt, LLL, RRR,
sign, frequence, save_time_i, save_time_j, prog_value(prog_index),
freq_value, TrainingMode, bluetooth_mode, ShowMessages, time_interval);
                side_description = sprintf('%s%s', side, hand_mark);
                summary_data(index_summary).trainingMode = TrainingMode;
                summary_data(index_summary).dataCounter = summary_row;
                summary_data(index_summary).fileName =
reportPSDFileName;
                summary_data(index_summary).prog_index = prog_index;
                summary_data(index_summary).side_description =
side_description;
                summary_data(index_summary).LLL = LLL;
                summary_data(index_summary).RRR = RRR;
                summary_data(index_summary).subject_mark = subject_mark;
                summary_data(index_summary).side_description =
side_description;
                summary_data(index_summary).A = A;
                summary_data(index_summary).B = B;
                summary_data(index_summary).C = C;
                summary_data(index_summary).D = D;
                summary_data(index_summary).E = E;
                summary_data(index_summary).row1 = row1;
                summary_data(index_summary).row2 = row2;
                summary_data(index_summary).row4 = row4;
                summary_data(index_summary).counter = 4;
                summary_data(index_summary).time_interval =
time_interval;
                summary_data(index_summary).mark_LR = mark_LR;
                summary_data(index_summary).suma_calkowitedek_2 =
suma_calkowitedek_2;
                summary_data(index_summary).suma_calkowitedek_3 =
suma_calkowitedek_3;
                summary_data(index_summary).suma_wezelwaga_2 =
suma_wezelwaga_2;
                summary_data(index_summary).suma_wezelwaga_3 =
suma_wezelwaga_3;
                summary_data(index_summary).result_spantree =
result_spantree;
                summary_data(index_summary).result_spantree_thresh =
result_spantree_thresh;
```

```
                summary_data(index_summary).counter_prog_index =
counter_prog_index;
                index_summary = index_summary+1;
            end
            summary_row = summary_row + 1;
            counter_prog_index = counter_prog_index+3;
        end
    end
end
```

Plik „Emotiv_Main_Module.m”

```
function summary_data = Emotiv_Main_Module( resultWnd, freq_value,
mark_LR, dataPackageFileCounter, DatapackageCounter, DataCounter,
TrainingMode, time_row_i, time_row_j, time_sec, bluetooth_mode,
ShowMessages)
global ReportFileNameSign;
global hTextWarning
global main_path;
global bt;
reportPSDFileName = sprintf('PSD%d_rawdatapackage', DatapackageCounter);
%dataPackageRawFileName = 'C:\matlab_doktorat\Matlab_main program_moje
dane\raw_data\rawdatapackage.txt';
dataPackageRawFileName = sprintf('%s\\raw_data\\rawdatapackage.txt',
main_path);
if freq_value == 3
    f = '12Hz';
end
if freq_value == 7
    f = '20Hz';
end
if freq_value == 8
    f = '22Hz';
end
% Wołana jest biblioteka dll (dzielona), która podłącza się do headsetu
i pobiera dane raw (surowe)!!!
calllib('eegloglib', 'eegLogRead', dataPackageRawFileName, 960, 5);
% zapamiętywanie plików raw
dataPackageraw_dataCatalogNameCounter = sprintf
('%s\\raw_data\\raw_data%d', main_path, DatapackageCounter);
mkdir (dataPackageraw_dataCatalogNameCounter);
dataPackagePSDCatalogName = sprintf ('%s\\PSDdata\\PSD%d', main_path,
DatapackageCounter);
mkdir (dataPackagePSDCatalogName);
dataPackageRawFileNameCounter = sprintf('%s\\rawdatapackage_%d.txt',
dataPackageraw_dataCatalogNameCounter, dataPackageFileCounter);
dataPackageSettingsFileNameCounter =
sprintf('%s\\rawdatapackage_%d.mat',
dataPackageraw_dataCatalogNameCounter, dataPackageFileCounter);
copyfile (dataPackageRawFileName, dataPackageRawFileNameCounter);
% struktura zawierająca ustawienia pomiaru
measurementSettings.f = f;
measurementSettings.mark_LR = mark_LR;
measurementSettings.time_sec = time_sec;
save( dataPackageSettingsFileNameCounter, 'measurementSettings' );
[y, Fs] = wavread('Concrete-Garage-Avenue--Hand-Clap-Sample--(Schoeps-
mk2-omni).wav');
processingvalue = sprintf('recording is done on %s', f);
set(hTextWarning, 'String', processingvalue);
drawnow;
sound(3*y, Fs);
```

```
% Nagrane raw data są przetwarzane - PSD i przenoszone do katalogu
data!!!
dataPackagePSDFileName = sprintf('%s\\PSD%d_rawdatapackage.txt',
dataPackagePSDCatalogName, dataPackageFileCounter);
Emotiv_Main_PSD(dataPackageRawFileNameCounter, dataPackagePSDFileName);
directory_save = sprintf('%s\\classified_data_rep', main_path);
LLL = [ 13, 24, 25, 36 ];
RRR = [ 18, 19, 30, 31 ];
sign = ReportFileNameSign;
side = sprintf('1 side_%s', f);
if TrainingMode == 1
    prog_value = [0.5, 0.7, 0.9];
else
    prog_value = [0.5];
end
macierz_A = { 'A', 'F', 'K', 'P', 'U', 'Z'};
macierz_B = { 'B', 'G', 'L', 'Q', 'V', 'AA' };
macierz_C = { 'C', 'H', 'M', 'R', 'W', 'AB' };
macierz_D = { 'D', 'I', 'N', 'S', 'X', 'AC'};
macierz_E = { 'E', 'J', 'O', 'T', 'Y', 'AD'};
prog_index = 0;
counter_prog_index = 5;
index_summary = 1;
for index = 1:1:length(prog_value)
    prog_index = prog_index + 1;
    for time = 1:1:length(time_row_i)
        time_interval = char(time_sec(time));
        frequence = sprintf('_12Hz1s13,24,25,36,..._%s', time_interval);
        save_time_i = time_row_i(time);
        save_time_j = time_row_j(time);
        A = char(macierz_A(time));
        B = char(macierz_B(time));
        C = char(macierz_C(time));
        D = char(macierz_D(time));
        E = char(macierz_E(time));
        row1 = 1;
        row2 = 2;
        row4 = 4;
        [~, fileName, fileExt] = fileparts(dataPackagePSDFileName);
        directoryNamePath = sprintf( '%s\\results_%s%s_%s',
directory_save, fileName, frequence );
        if TrainingMode == 1
            title('Command/Control: TRAINING - classification result');
        end
        if TrainingMode == 0
            title('Command/Control: RECOGNITION - classification
result');
        end
        [X, mark_name, subject_mark, hand_mark, suma_calkowitedek_3,
suma_calkowitedek_2, suma_wezelwaga_3, suma_wezelwaga_2,
result_spantree, result_spantree_thresh, result_arduino] =
Emotiv_Main_Classification(bt, resultWnd, dataPackagePSDFileName,
directoryNamePath, fileName, fileExt, LLL, RRR, sign, frequence,
save_time_i, save_time_j, prog_value(prog_index), freq_value,
TrainingMode, bluetooth_mode, ShowMessages);
        side_description = sprintf('%s%s', side, hand_mark);
        summary_data(index_summary).trainingMode = TrainingMode;
        summary_data(index_summary).dataCounter = DataCounter;
        summary_data(index_summary).fileName = reportPSDFileName;
        summary_data(index_summary).time_row_i = time_row_i;
        summary_data(index_summary).prog_index = prog_index;
```

```
        summary_data(index_summary).side_description = side_description;
        summary_data(index_summary).LLL = LLL;
        summary_data(index_summary).RRR = RRR;
        summary_data(index_summary).subject_mark = subject_mark;
        summary_data(index_summary).side_description = side_description;
        summary_data(index_summary).A = A;
        summary_data(index_summary).B = B;
        summary_data(index_summary).C = C;
        summary_data(index_summary).D = D;
        summary_data(index_summary).E = E;
        summary_data(index_summary).row1 = row1;
        summary_data(index_summary).row2 = row2;
        summary_data(index_summary).row4 = row4;
        summary_data(index_summary).counter = 4;
        summary_data(index_summary).time_interval = time_interval;
        summary_data(index_summary).mark_LR = mark_LR;
        summary_data(index_summary).suma_calkowitedek_2 =
suma_calkowitedek_2;
        summary_data(index_summary).suma_calkowitedek_3 =
suma_calkowitedek_3;
        summary_data(index_summary).suma_wezelwaga_2 = suma_wezelwaga_2;
        summary_data(index_summary).suma_wezelwaga_3 = suma_wezelwaga_3;
        summary_data(index_summary).result_spantree = result_spantree;
        summary_data(index_summary).result_spantree_thresh =
result_spantree_thresh;
        summary_data(index_summary).counter_prog_index =
counter_prog_index;
        index_summary = index_summary+1;
    end
    counter_prog_index = counter_prog_index+3;
end
```

Plik „Emotiv_Main_PSD.m”

```
function Emotiv_Main_PSD(dataPackageRawFileNameCounter, DataPSDFileName)
% Elektrody Emotiv: AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4,
F8, AF4
elektroda = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14];
el_sign = {'AF3', 'F7', 'F3', 'FC5', 'T7', 'P7', 'O1', 'O2', 'P8', 'T8',
'FC6', 'F4', 'F8', 'AF4'};
N = 64;     %liczba próbek
fpr = 128;    %[Hz]
Macierz = [];
Macierz_wiersze = [];
% załadowanie danych raw z pliku
directory_impdata = importdata(dataPackageRawFileNameCounter, ',', 1);
if isfield(directory_impdata, 'data') == 0
    errordlg('Błąd odczytu danych','Error');
    return;
else
    datax = directory_impdata.data(:, 2:15);
    for xx = 1:1:length(elektroda)
        x = [];
        X = [];
        licznik = 0;
        aaa = size(datax,1);
        floor_datax = floor(aaa/N);
        number_datax = floor_datax * N;
        for r=1:N:number_datax
            licznik = licznik + 1;
            x = datax(r:r+63,elektroda(xx));
```

```
            for czestotliwosc = 0:1:(length(x)-1)
                X_DFT = 0;
                for n = 0:1:(length(x)-1)
                    a = x(n+1);
                    b = a*exp(-1j*(2*pi*czestotliwosc*n)/N);
                    X_DFT = X_DFT + (a*exp(-
1j*(2*pi*czestotliwosc*n)/N));
                end
                X(czestotliwosc+1) = X_DFT;
            end
             X = X(1, 4:1:15);
             realX = real(X);
             imX = imag(X);
             marker = 0;
            %% liczenie modułu
            modX = [];
            PSD = [];
            for i=1:1:length(realX)
                modX(i) = sqrt(realX(i)^2+imX(i)^2);
                PSD(i) = realX(i)^2+imX(i)^2;
            end
            Macierz_wiersze = [ Macierz_wiersze; PSD ];
        end
        Macierz = [ Macierz Macierz_wiersze ];
        Macierz_wiersze = [];
    end
    % zapis do pliku
    fileID = fopen(DataPSDFileName, 'w');
    for i=1:1:size(Macierz,1)
        for j=1:1:size(Macierz,2)
            fprintf(fileID, '%f, ', Macierz(i,j));
        end
        fprintf(fileID, '\n');
    end
    fclose(fileID);
    Macierz = [];
end
disp ('koniec');
```

Plik „Emotiv_Main_Classification”

```
function [X, mark_name, subject_mark, hand_mark, suma_calkowitedek_3,
suma_calkowitedek_2, suma_wezelwaga_3, suma_wezelwaga_2,
result_spantree, result_spantree_thresh, result_arduino] =
Emotiv_Main_Classification (bt, resultWnd, directoryfile,
directoryNamePath, fileName, fileExt, LLL, RRR, sign, frequence,
save_time_i, save_time_j, prog_value, freq_value, TrainingMode,
bluetooth_mode, ShowMessages, time_interval)
global RdiameterSize
global hText6;
global IPSign;
%% Values
path = directoryNamePath;
mark = fileName;
subject_mark = mark;
hand_mark = mark;
mark_name = sprintf('%s_%s', mark, frequence);
title_mark_name = strrep(mark_name, '_', ' ');
title_mark = strrep(mark, '_', ' ');
title_prefix = sprintf('DATA: %s', title_mark);
date = 'REGISTRATION: EMOTIV data';
tol = 3;
```

```
WR = RdiameterSize;
%% Import RAW data
data_EMOTIV = directoryfile;
EMOTIV = importdata(data_EMOTIV);
decoding_testdata_file_name = sprintf( '%s\\decoding_testdata_%s', path,
mark_name );
spanning_tree_file_name = sprintf( '%s\\spanning_tree_%s', path,
mark_name );
spanning_tree_threshold_file_name = sprintf(
'%s\\spanning_tree_threshold_%s', path, mark_name );
spanning_tree_decoding_file_name = sprintf(
'%s\\spanning_tree_decoding_%s', path, mark_name );
detection_decoding_data = sprintf( '%s\\detection_decoding_data_%s',
path, mark_name );
detection_threshold_data = sprintf( '%s\\detection_threshold_data_%s',
path, mark_name );
figure_name = sprintf( '%s\\fig_%s', path, mark_name );
spanning_tree_decoding_file_name_xls = sprintf( '%s%s.xls',
spanning_tree_decoding_file_name, sign );
figure_name_png = sprintf( '%s%s.png', figure_name, sign );
decoding_testdata_file_name_xls = sprintf( '%s%s.xls',
decoding_testdata_file_name, sign );
spanning_tree_file_name_xls = sprintf( '%s%s.xls',
spanning_tree_file_name, sign );
detection_decoding_data_xls = sprintf( '%s%s.xls',
detection_decoding_data, sign );
detection_threshold_data_xls = sprintf( '%s%s.xls',
detection_threshold_data, sign );
spanning_tree_threshold_name_xls = sprintf( '%s%s.xls',
spanning_tree_threshold_file_name, sign );
% Data Decoding
[X, decoding_3ab_testdata_rotated, decoding_3cb_testdata_rotated,
decoding_2ab_testdata_rotated,
decoding_2cb_testdata_rotated]=Emotiv_Spherical_Voxels_Inverse_Tstat(EMO
TIV, WR, save_time_i, save_time_j, freq_value);
% % % ------------------- DRAW: Wyświetlanie figury do narysowania grafu
dwudzielnego
% ha = figure('units','normalized','outerposition',[0 0 1 1]);   %
enlarge figure for the whole screen
% % % ------------------- end DRAW: Wyświetlanie figury do narysowania
grafu dwudzielnego
%
% % % ------------------- DRAW: Wyświetlanie opisów na figurze z grafem
dwudzielnym
% annotation(ha,'textbox',...
%     [0.018037037037037 0.900740740740741 0.0804814814814815
0.0740740740740727],...
%     'String',{date},...
%     'FitBoxToText','off');
%
% annotation(ha,'textbox',...
%     [0.0177777777777778 0.774814814814815 0.0807407407407407
0.096296296296296],...
%     'String',{title_prefix},...
%     'FitBoxToText','off');
%
% LLLstr = num2str(LLL);
% left_nodes = sprintf ('Left nodes: %s', LLLstr)
% annotation(ha,'textbox',...
%     [0.017037037037037 0.41037037037037 0.08 0.0607407407407405],...
%     'String',{left_nodes},...
```

```
%     'FitBoxToText','off');
%
% RRRstr = num2str(RRR);
% left_nodes = sprintf ('Right nodes: %s', RRRstr)
% annotation(ha,'textbox',...
%     [0.017037037037037 0.321481481481481 0.0792592592592593
0.0562962962962957],...
%     'String',{left_nodes},...
%     'FitBoxToText','off');
% % % ------------------- end DRAW: Wyświetlanie opisów na figurze z
grafem dwudzielnym
%% Checking 2ab
suma_calkowitedek_2ab = [];
suma_wezelwaga_2ab = [];
suma_calkowitedek_2ab = 0;
suma_wezelwaga_2ab = 0;
if size (decoding_2ab_testdata_rotated,1)>0
    [calkowitedrzewomin_2ab,znacznik_2ab,wezelwaga_2ab] =
Emotiv_Gower(decoding_2ab_testdata_rotated, prog_value);
    [calkowitdekodowanie_2ab] =
Emotiv_Gower_ab_Coordinates(calkowitedrzewomin_2ab,
decoding_2ab_testdata_rotated);
     min_wezelawaga_2ab=min(wezelwaga_2ab(:,3));
    % Drawing (two-sided graph) - 2ab dla danych calkowitdekodowanie_2ab
bez uzględnienia pol Brodmanna
     [tab_left_2ab, tab_right_2ab, X_ab, Y_ab, linecolor_2ab,
wynik_calkowitedekodowanie_2ab, wynik_wezelwaga_2ab,
suma_calkowitedek_2ab, suma_wezelwaga_2ab ] =
Emotiv_Gower_left_Visualization( calkowitdekodowanie_2ab,
min_wezelawaga_2ab, LLL, RRR );
%     % % ------------------- DRAW: Rysowanie grafu dwudzielnego 2ab
%     subplot(2,2,1);
%     drawing_twosided_graph_left( tab_left_2ab, tab_right_2ab, X_ab,
Y_ab, linecolor_2ab );
%
%     title_string = sprintf( ' sum sptree 2ab: %0.2f, sum sptree
threshhold 2ab: %0.2f', suma_calkowitedek_2ab, suma_wezelwaga_2ab );
%     title( title_string );
%     % % ------------------- end DRAW: Rysowanie grafu dwudzielnego 2ab
end
%% Checking 2cb
suma_calkowitedek_2cb = [];
suma_wezelwaga_2cb = [];
suma_calkowitedek_2cb = 0;
suma_wezelwaga_2cb = 0;
if size (decoding_2cb_testdata_rotated,1)>0
    [calkowitedrzewomin_2cb,znacznik_2cb,wezelwaga_2cb] =
Emotiv_Gower(decoding_2cb_testdata_rotated, prog_value);
    [calkowitdekodowanie_2cb] =
Emotiv_Gower_ab_Coordinates(calkowitedrzewomin_2cb,
decoding_2cb_testdata_rotated);
     min_wezelawaga_2cb=min(wezelwaga_2cb(:,3));
    % Drawing (two-sided graph) - 2cb dla danych calkowitdekodowanie_2cb
bez uzględnienia pol Brodmanna
     [ tab_left_2cb, tab_right_2cb, X_2cb, Y_2cb, linecolor_2cb,
wynik_calkowitedekodowanie_2cb, wynik_wezelwaga_2cb,
suma_calkowitedek_2cb, suma_wezelwaga_2cb ] =
Emotiv_Gower_left_Visualization( calkowitdekodowanie_2cb,
min_wezelawaga_2cb, LLL, RRR );
%     % % ------------------- DRAW: Rysowanie grafu dwudzielnego 2cb
%     subplot(2,2,2);
```

```
%     drawing_twosided_graph_left( tab_left_2cb, tab_right_2cb, X_2cb,
Y_2cb, linecolor_2cb  );
%
%     title_string = sprintf( ' sum sptree 2cb: %0.2f, sum sptree
threshhold 2cb: %0.2f', suma_calkowitedek_2cb, suma_wezelwaga_2cb);
%     title( title_string );
%     % % ------------------- end DRAW: Rysowanie grafu dwudzielnego 2cb
end
suma_calkowitedek_2=suma_calkowitedek_2ab+suma_calkowitedek_2cb;
suma_wezelwaga_2=suma_wezelwaga_2ab+suma_wezelwaga_2cb;
% % % ------------------- DRAW: Wyświetlanie na figurze grafu
dwudzielnego wyniku klasyfikacji dla lewej ręki 2 ab i 2cb
% annotation(ha,'textbox',...
%     [0.0163609831029186 0.0404624277456647 0.0489231950844854
0.0390173410404623],...
%     'String',{'Results:'},...
%     'FitBoxToText','off');
%
%
% suma_calkowitedek_2_txt = sprintf( 'sum spantree 2=%0.2f',
suma_calkowitedek_2 );
% annotation(ha,'textbox',...
%     [0.07209375 0.0467836257309941 0.391061955469506
0.0324909747292419],...
%     'String',{suma_calkowitedek_2_txt},...
%     'FitBoxToText','off');
% suma_wezelwaga_2_txt = sprintf( 'sum spantree threshhold 2=%0.2f',
suma_wezelwaga_2 );
% annotation(ha,'textbox',...
%     [0.486754966887417 0.0459940652818991 0.398348003259984
0.0316265060240964],...
%     'String',{suma_wezelwaga_2_txt},...
%     'FitBoxToText','off');
% % % ------------------- DRAW: Wyświetlanie na figurze grafu
dwudzielnego wyniku klasyfikacji dla lewej ręki 2 ab i 2cb
%% Checking 3ab
suma_calkowitedek_3ab = [];
suma_wezelwaga_3ab = [];
suma_calkowitedek_3ab = 0;
suma_wezelwaga_3ab = 0;
if size (decoding_3ab_testdata_rotated,1)>0
    [calkowitedrzewomin_3ab,znacznik_3ab,wezelwaga_3ab] =
Emotiv_Gower(decoding_3ab_testdata_rotated, prog_value);
    [calkowitdekodowanie_3ab] =
Emotiv_Gower_ab_Coordinates(calkowitedrzewomin_3ab,
decoding_3ab_testdata_rotated);
      min_wezelawaga_3ab=max(wezelwaga_3ab(:,3));
    % Drawing (two-sided graph) - 3ab dla danych calkowitdekodowanie_3ab
bez uzględnienia pol Brodmanna
    [ tab_left_3ab, tab_right_3ab, X_3ab, Y_3ab, linecolor_3ab,
wynik_calkowitedekodowanie_3ab, wynik_wezelwaga_3ab,
suma_calkowitedek_3ab, suma_wezelwaga_3ab ] =
Emotiv_Gower_right_Visualization( calkowitdekodowanie_3ab,
min_wezelawaga_3ab, LLL, RRR );
%     % % ------------------- DRAW: Rysowanie grafu dwudzielnego 3ab
%     subplot(2,2,3);
%     drawing_twosided_graph_right( tab_left_3ab, tab_right_3ab, X_3ab,
Y_3ab, linecolor_3ab  );
%
%     title_string = sprintf( ' sum sptree 3ab: %0.2f, sum sptree
threshhold 3ab: %0.2f', suma_calkowitedek_3ab, suma_wezelwaga_3ab );
```

```
%     title( title_string );
%     % % ------------------- end DRAW: Rysowanie grafu dwudzielnego 3ab
end
%% Checking 3cb
suma_calkowitedek_3cb = [];
suma_wezelwaga_3cb = [];
suma_calkowitedek_3cb = 0;
suma_wezelwaga_3cb = 0;
if size (decoding_3cb_testdata_rotated,1)>0
    [calkowitedrzewomin_3cb,znacznik_3b,wezelwaga_3cb] =
Emotiv_Gower(decoding_3cb_testdata_rotated, prog_value);
    [calkowitdekodowanie_3cb] =
Emotiv_Gower_ab_Coordinates(calkowitedrzewomin_3cb,
decoding_3cb_testdata_rotated);
     min_wezelawaga_3cb=max(wezelwaga_3cb(:,3));
    % Drawing (two-sided graph) - 3cb dla danych calkowitdekodowanie_3cb
bez uzględnienia pol Brodmanna
    [ tab_left_3cb, tab_right_3cb, X_3cb, Y_3cb, linecolor_3cb,
wynik_calkowitedekodowanie_3cb, wynik_wezelwaga_3cb,
suma_calkowitedek_3cb, suma_wezelwaga_3cb ] =
Emotiv_Gower_right_Visualization( calkowitdekodowanie_3cb,
min_wezelawaga_3cb, LLL, RRR );
%     % % ------------------- DRAW: Rysowanie grafu dwudzielnego 3cb
%     subplot(2,2,4);
%     drawing_twosided_graph_right( tab_left_3cb, tab_right_3cb, X_3cb,
Y_3cb, linecolor_3cb  );
%
%     title_string = sprintf( ' sum spantree 3cb: %0.2f, sum spantree
threshhold 3cb: %0.2f', suma_calkowitedek_3cb, suma_wezelwaga_3cb );
%     title( title_string );
%     % % ------------------- end DRAW: Rysowanie grafu dwudzielnego 3cb
end
suma_calkowitedek_3=suma_calkowitedek_3ab+suma_calkowitedek_3cb;
suma_wezelwaga_3=suma_wezelwaga_3ab+suma_wezelwaga_3cb;
% % % ------------------- DRAW: Wyświetlanie na figurze grafu
dwudzielnego wyniku klasyfikacji dla prawej ręki 3ab i 3cb
% suma_calkowitedek_3_txt = sprintf( 'sum spantree 3=%0.2f',
suma_calkowitedek_3 );
% annotation(ha,'textbox',...
%     [0.07209375 0.00541516245487365 0.391061955469506
0.0324909747292419],...
%     'String',{suma_calkowitedek_3_txt},...
%     'FitBoxToText','off');
% suma_wezelwaga_3_txt = sprintf( 'sum spantree threshhold 3=%0.2f',
suma_wezelwaga_3 );
% annotation(ha,'textbox',...
%     [0.486754966887417 0.00451807228915663 0.398348003259984
0.0316265060240964],...
%     'String',{suma_wezelwaga_3_txt},...
%     'FitBoxToText','off');
% % % ------------------- end DRAW: Wyświetlanie na figurze grafu
dwudzielnego wyniku klasyfikacji dla prawej ręki 3ab i 3cb
%%
result_spantree = [];
result_spantree_thresh = [];
figure( resultWnd );
axis off;
 if suma_calkowitedek_2>abs(suma_calkowitedek_3)
    result_spantree = 'classification spantree: LEFT';
    result_spantree_txt = sprintf('%d-%d s, classification spantree:
LEFT', save_time_i, save_time_j);
```

```
    result_arduino = '0';
%     % % ------------------- DRAW
%     annotation(ha,'textbox',...
%     [0.0172962962962963 0.64 0.0804814814814815 0.105185185185184],...
%     'String',{result_spantree_txt},...
%     'FitBoxToText','off');
%     % % ------------------- end DRAW
elseif (suma_calkowitedek_2==0) && (suma_calkowitedek_3==0)
    result_spantree = 'class spantree : -';
    result_spantree_txt = 'class spantree : -';
    result_arduino = '2';
%     % % ------------------- DRAW
%      annotation(ha,'textbox',...
%     [0.0172962962962963 0.64 0.0804814814814815 0.105185185185184],...
%     'String',{result_spantree_txt},...
%     'FitBoxToText','off');
%     % % ------------------- end DRAW
 else
    result_spantree = 'classification spantree: RIGHT';
    result_spantree_txt = sprintf('%d-%d s, classification spantree:
RIGHT', save_time_i, save_time_j);
    result_arduino = '1';
%     % % ------------------- DRAW
%     annotation(ha,'textbox',...
%     [0.0172962962962963 0.64 0.0804814814814815 0.105185185185184],...
%     'String',{result_spantree_txt},...
%     'FitBoxToText','off');
%     % % ------------------- end DRAW
 end
hold on
global hText1;
global hText2;
global hText4;
%%
if suma_wezelwaga_2>abs(suma_wezelwaga_3)
    result_spantree_thresh = sprintf( 'classification spantree thresh
%1.1f: LEFT', prog_value );
    result_spantree_thresh_txt = sprintf( '%d-%d s, classification
spantree thresh  %1.1f: LEFT' , save_time_i, save_time_j, prog_value );
%     % % ------------------- DRAW
%      annotation(ha,'textbox',...
%     [0.017037037037037 0.503703703703704 0.0807407407407407
0.103703703703702],...
%     'String',{result_spantree_thresh_txt},...
%     'FitBoxToText','off');
%     % % ------------------- end DRAW
elseif (suma_wezelwaga_2==0) && (suma_wezelwaga_3==0)
    result_spantree_thresh = sprintf( 'classification spantree thresh
%1.1f: - ', prog_value );
    result_spantree_thresh_txt = sprintf( '%d-%d s, classification
spantree thresh %1.1f: - ', save_time_i, save_time_j, prog_value );
%     % % ------------------- DRAW
%      annotation(ha,'textbox',...
%     [0.017037037037037 0.503703703703704 0.0807407407407407
0.103703703703702],...
%     'String',{result_spantree_thresh_txt},...
%     'FitBoxToText','off');
%     % % ------------------- end DRAW
else
    result_spantree_thresh = sprintf( 'classification spantree thresh
%1.1f: RIGHT', prog_value );
```

```
    result_spantree_thresh_txt = sprintf( '%d-%d s, classification
spantree thresh  %1.1f: RIGHT', save_time_i, save_time_j, prog_value );
%     % % ------------------- DRAW
%      annotation(ha,'textbox',...
%     [0.017037037037037 0.503703703703704 0.0807407407407407
0.103703703703702],...
%     'String',{result_spantree_thresh_txt},...
%     'FitBoxToText','off');
%     % % ------------------- end DRAW
end
hold on
if ShowMessages == 1
    if TrainingMode == 1
        set(hText1, 'String', num2str(result_spantree_txt) );
        set(hText2, 'String', num2str(result_spantree_thresh_txt) );
    end
    if TrainingMode == 0
        set(hText1, 'String', num2str(result_spantree_txt) );
    end
    if bluetooth_mode == 1 && TrainingMode == 0
        %wyjście na bluetooth
        fopen(bt);
        if result_arduino == '0'
            fwrite(bt, 0);
        else
            fwrite(bt, 1);
        end
        fclose(bt);
        % koniec wyjście na bluetooth
    end
    if bluetooth_mode == 2 && TrainingMode == 0
        %wyjście na wifi
        t = tcpip(IPSign, 8888)
        fopen(t);
        if result_arduino == '0'
            fwrite(t, 'l');
        else
            fwrite(t, 'r');
        end
        fclose(t);
        % koniec wyjście na wifi
    end
end
if ShowMessages == 0
    FileNameShow = sprintf('File: %s, %s', strrep(fileName,'_','-'),
time_interval);
    set(hText6, 'String', ' ' );
    set(hText4, 'String', FileNameShow );
end
drawnow; pause(4);
% set (ha, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 35 19]);
% enlarge figure for the whole screen before save it
% print(ha, figure_name_png,'-dpng', '-r300')
% save of the whole screen figure to the file
% close(ha);
fprintf('Classification proces eneded\n');
end
```

Plik „Emotiv_Gower_left_Visualization.m”

```
function [ tab_left, tab_right, X, Y, linecolor,
wynik_calkowitedekodowanie, wynik_wezelwaga, suma_calkowitedek,
```

```
suma_wezelwaga ] = Emotiv_Gower_left_Visualization( calkowitdekodowanie,
min_wezelawaga, LLL, RRR )
labels_left = ['a 1 '; 'a 2 '; 'a 3 '; 'c 10'; 'c 11'; 'c 12'; 'a 13';
'a 14'; 'a 15'; 'c 22'; 'c 23'; 'c 24'; 'a 25'; 'a 26'; 'a 27'; 'c 34';
'c 35'; 'c 36'];
% 1 - a
% 2 - c
check_label_left = [1 1 1 1 1 2 2 1 1 1 1 2 2 2 2 2 2 2];   % pola
uwzgledniajace tylko cwiartke przednia polkuli lewej - czyli c (bez
10,11,22,23), oraz obszary 13,25,26,27
labels_right = ['4 a* '; '5 a* ';'6 a* '; '7 c* '; '8 c* '; '9 c* '; '16
a*'; '17 a*'; '18 a*'; '19 c*'; '20 c*'; '21 c*'; '28 a*';...
    '29 a*'; '30 a*'; '31 c*'; '32 c*'; '33 c*'];
% 3 - a*
% 4 - c*
check_label_right = [3 3 3 4 3 3 3 3 4 4 3 3 4 4 4 4 4 4];  % pola
uwzgledniajace tylko cwiartke przednia polkuli prawej - czyli c*(bez
9,8,20,21), oraz obszary 18,28,29,30
% Powyzsze pola check label sa niewykorzystywane
%% all nodes without excluding specific nodes
node_left = [  1, 2, 3, 10, 11, 12, 13, 14, 15, 22, 23, 24, 25, 26, 27,
34, 35, 36  ];
node_right = [ 4, 5, 6, 7, 8, 9, 16, 17, 18, 19, 20, 21, 28, 29, 30, 31,
32, 33 ];
%% only important nodes
important_node_left = LLL;
important_node_right = RRR;
matrix_important_node_left = zeros(1,size(node_left,2));
for i=1:1:size(important_node_left,2)
    for j=1:1:size(node_left,2)
        if important_node_left(i) == node_left(j)
            matrix_important_node_left (1,j) = 1;
        end
    end
end
matrix_important_node_right = zeros(1,size(node_right,2));
for i=1:1:size(important_node_right,2)
    for j=1:1:size(node_right,2)
        if important_node_right(i) == node_right(j)
            matrix_important_node_right (1,j) = 1;
        end
    end
end
y = linspace( 10, 190, 18 );
for i=1:1:size(node_left,2)
    tab_left( i, : ).x = 10;
    yleftspr = y(size(node_left,2)-i+1);
    tab_left( i, : ).y = y(size(node_left,2)-i+1);
    tab_left( i, : ).label = labels_left(i, :);
    tab_left( i, : ).node = node_left(i);
    tab_left( i, : ).importantnode = matrix_important_node_left(i);
    tab_left( i, : ).green = 0;
end
for i=1:1:size(node_right,2)
    tab_right( i, : ).x = 100;
    yrightspr = y(size(node_right,2)-i+1);
    tab_right( i, : ).y = y(size(node_right,2)-i+1);
    tab_right( i, : ).label = labels_right(i, :);
    tab_right( i, : ).node = node_right(i);
    tab_right( i, : ).importantnode = matrix_important_node_right(i);
    tab_right( i, : ).green = 0;
```

```
end
X = [];
Y = [];
linecolor = [];
check_counter = zeros(size(node_left));
suma_calkowitedek = 0;
suma_wezelwaga=0;
wynik_calkowitedekodowanie = [];
wynik_wezelwaga = [];
counter_left = 0;
% przerabiac wszystkie i dla important wykonywac to co jest!!!
for i=1:1:size(node_left,2)
    if tab_left( i, : ).importantnode ==1
        counter_left = counter_left + 1;
        for j=1:1:size(calkowitdekodowanie,1)
            a = important_node_left(counter_left);
            if calkowitdekodowanie (j,2) ==
important_node_left(counter_left)
                    for ii=1:1:size(node_right,2)
                        if tab_right( ii, : ).node ==
calkowitdekodowanie (j,5)
                            if calkowitdekodowanie (j,1) >=
min_wezelawaga
                                wynik_calkowitedekodowanie = [
wynik_calkowitedekodowanie; calkowitdekodowanie(j,:)];
                                wynik_wezelwaga = [ wynik_wezelwaga;
calkowitdekodowanie(j,:)];
                                X = [ X tab_left(i).x+10,
tab_right(ii).x ];
                                Y = [ Y tab_left(i).y, tab_right(ii).y];
                                linecolor = [linecolor; 'r'];
                                tab_left( i, : ).green = 1;
                                tab_right( ii, : ).green = 1;
suma_calkowitedek=suma_calkowitedek+calkowitdekodowanie(j,1);
suma_wezelwaga=suma_wezelwaga+calkowitdekodowanie(j,1);
                            else
                                wynik_calkowitedekodowanie = [
wynik_calkowitedekodowanie; calkowitdekodowanie(j,:)];
                                X = [ X tab_left(i).x+10,
tab_right(ii).x ];
                                Y = [ Y tab_left(i).y, tab_right(ii).y];
                                linecolor = [linecolor; 'b'];
                                tab_left( i, : ).green = 1;
                                tab_right( ii, : ).green = 1;
suma_calkowitedek=suma_calkowitedek+calkowitdekodowanie(j,1);
                            end
                        end
                    end
            end
        end

    end
end
end
```

Plik „Emotiv_Gower_right_Visualization.m”

```
function [  tab_left, tab_right, X, Y, linecolor,
wynik_calkowitedekodowanie, wynik_wezelwaga, suma_calkowitedek,
suma_wezelwaga ] = Emotiv_Gower_right_Visualization(
calkowitdekodowanie, min_wezelawaga, LLL, RRR )
for i=1:1:size(calkowitdekodowanie,1)
```

```
    if calkowitdekodowanie(i,1)<0
        calkowitdekodowanie(i,1)=abs(calkowitdekodowanie(i,1));
    end
end
min_wezelawaga = abs(min_wezelawaga);
labels_left = ['a 1 '; 'a 2 '; 'a 3 '; 'c 10'; 'c 11'; 'c 12'; 'a 13';
'a 14'; 'a 15'; 'c 22'; 'c 23'; 'c 24'; 'a 25'; 'a 26'; 'a 27'; 'c 34';
'c 35'; 'c 36'];
% 1 - a
% 2 - c
check_label_left = [1 1 1 1 1 2 2 1 1 1 1 2 2 2 2 2 2 2];   % pola
uwzgledniajace tylko cwiartke przednia polkuli lewej - czyli c (bez
10,11,22,23), oraz obszary 13,25,26,27
% Powyzsze pola check label sa niewykorzystywane
labels_right = ['4 a* '; '5 a* ';'6 a* '; '7 c* '; '8 c* '; '9 c* '; '16
a*'; '17 a*'; '18 a*'; '19 c*'; '20 c*'; '21 c*'; '28 a*';...
    '29 a*'; '30 a*'; '31 c*'; '32 c*'; '33 c*'];
% 3 - a*
% 4 - c*
check_label_right = [3 3 3 4 3 3 3 3 4 4 3 3 4 4 4 4 4 4];  % pola
uwzgledniajace tylko cwiartke przednia polkuli prawej - czyli c*(bez
9,8,20,21), oraz obszary 18,28,29,30
%% all nodes without excluding specific nodes
node_left = [  1, 2, 3, 10, 11, 12, 13, 14, 15, 22, 23, 24, 25, 26, 27,
34, 35, 36  ];
node_right = [ 4, 5, 6, 7, 8, 9, 16, 17, 18, 19, 20, 21, 28, 29, 30, 31,
32, 33 ];
%% only important nodes
important_node_left = LLL;
important_node_right = RRR;
matrix_important_node_left = zeros(1,size(node_left,2));
for i=1:1:size(important_node_left,2)
    for j=1:1:size(node_left,2)
        if important_node_left(i) == node_left(j)
            matrix_important_node_left (1,j) = 1;
        end
    end
end
matrix_important_node_right = zeros(1,size(node_right,2));
for i=1:1:size(important_node_right,2)
    for j=1:1:size(node_right,2)
        if important_node_right(i) == node_right(j)
            matrix_important_node_right (1,j) = 1;
        end
    end
end
y = linspace( 10, 190, 18 );
for i=1:1:size(node_left,2)
    tab_left( i, : ).x = 10;
    tab_left( i, : ).y = y(size(node_left,2)-i+1);
    tab_left( i, : ).label = labels_left(i, :);
    tab_left( i, : ).node = node_left(i);
    tab_left( i, : ).importantnode =matrix_important_node_left(i);
    tab_left( i, : ).green = 0;
end
for i=1:1:size(node_right,2)
    tab_right( i, : ).x = 100;
    tab_right( i, : ).y = y(size(node_right,2)-i+1);
    tab_right( i, : ).label = labels_right(i, :);
    tab_right( i, : ).node = node_right(i);
    tab_right( i, : ).importantnode =matrix_important_node_right(i);
```

```
    tab_right( i, : ).green = 0;
end
X = [];
Y = [];
linecolor = [];
check_counter = zeros(size(node_right));
suma_calkowitedek = 0;
suma_wezelwaga=0;
wynik_calkowitedekodowanie = [];
wynik_wezelwaga = [];
counter_right = 0;
for i=1:1:size(node_right,2)
    if tab_right( i, : ).importantnode ==1
        counter_right = counter_right + 1;
        for j=1:1:size(calkowitdekodowanie,1)
            a = important_node_right(counter_right);
            if calkowitdekodowanie (j,5) ==
important_node_right(counter_right)
                    for ii=1:1:size(node_left,2)
                        if tab_left( ii, : ).node == calkowitdekodowanie
(j,2)
                            if calkowitdekodowanie (j,1) >=
min_wezelawaga
                                wynik_calkowitedekodowanie = [
wynik_calkowitedekodowanie; calkowitdekodowanie(j,:)];
                                wynik_wezelwaga = [ wynik_wezelwaga;
calkowitdekodowanie(j,:)];
                                X = [ X tab_left(ii).x+10,
tab_right(i).x ];
                                Y = [ Y tab_left(ii).y, tab_right(i).y];

                                linecolor = [linecolor; 'r'];
                                tab_left( ii, : ).green = 1;
                                tab_right( i, : ).green = 1;
suma_calkowitedek=suma_calkowitedek+calkowitdekodowanie(j,1);
suma_wezelwaga=suma_wezelwaga+calkowitdekodowanie(j,1);
                            else
                                wynik_calkowitedekodowanie = [
wynik_calkowitedekodowanie; calkowitdekodowanie(j,:)];
                                X = [ X tab_left(ii).x+10,
tab_right(i).x ];
                                Y = [ Y tab_left(ii).y, tab_right(i).y];
                                linecolor = [linecolor; 'b'];
                                tab_left( ii, : ).green = 1;
                                tab_right( i, : ).green = 1;
suma_calkowitedek=suma_calkowitedek+calkowitdekodowanie(j,1);
                            end
                        end
                    end
            end
        end
    end
end
end
```

Plik „drawing_twosided_graph_left.m”

```
function [] = drawing_twosided_graph_left( tab_left, tab_right, X, Y,
linecolor )
axis([0 210 0 210]);
text (10, 200, 'L');
text (100, 200, 'P');
```

```
spr = size(tab_left,1);
for i=1:1:size(tab_left,1)
   text( tab_left(i).x, tab_left(i).y, tab_left(i).label )
   text( tab_right(i).x, tab_right(i).y, tab_right(i).label )
   if tab_left( i, : ).green == 1
       text( tab_left(i).x, tab_left(i).y, tab_left(i).label,
'color','g' )
   end
end
a=1;
if size(linecolor,1) ~= 0
    for i=1:2:size(X,2)
        line([X(i) X(i+1)],[Y(i) Y(i+1)],'color',linecolor(a,:),
'LineWidth',1);
        a=a+1;
    end
end
end
```

Plik „drawing_twosided_graph_right.m”

```
function [] = drawing_twosided_graph_right( tab_left, tab_right, X, Y,
linecolor )
axis([0 210 0 210]);
text (10, 200, 'L');
text (100, 200, 'P');
for i=1:1:size(tab_right,1)
   text( tab_left(i).x, tab_left(i).y, tab_left(i).label )
   text( tab_right(i).x, tab_right(i).y, tab_right(i).label )
   if tab_right( i, : ).green == 1
       text( tab_right(i).x, tab_right(i).y, tab_right(i).label,
'color','g' )
   end
end
a=1;
if size(linecolor,1) ~= 0
    for i=1:2:size(X,2)
        line([X(i) X(i+1)],[Y(i) Y(i+1)],'color',linecolor(a,:),
'LineWidth',1);
        a=a+1;
    end
end
end
```

Plik „Emotiv_Spherical_Voxels_Inverse_Tstat.m”

```
function [X, dekodowanie_3ab_danetest, dekodowanie_3cb_danetest,
dekodowanie_2ab_danetest, dekodowanie_2cb_danetest] =
Emotiv_Spherical_Voxels_Inverse_Tstat(EMOTIV, WR, save_time_i,
save_time_j, freq_value)
Nel=14;
wariancja=0;
%--------------------------------------------------------------------------
% WOKSELE !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
WstepTeta = -5*pi/180;
Wstepfi = 10*pi/180;
%% l. wokseli:
Wnv = (abs(((90*pi/180) / WstepTeta) * ((360*pi/180) / Wstepfi)));
WX=zeros(Wnv,1);
WY=zeros(Wnv,1);
WZ=zeros(Wnv,1);
Wspolfi=zeros(Wnv,1);
```

```
WspolTeta=zeros(Wnv,1);
Wnv=1;
a=1;
% fig = figure();
for WTeta=90*pi/180 : WstepTeta : -WstepTeta
    for Wfi=(0)*pi/180 : Wstepfi : (360)*pi/180-Wstepfi
        WX (Wnv,1) = WR*cos(Wfi)*sin(WTeta);
        WY (Wnv,1) = WR*sin(Wfi)*sin(WTeta);
        WZ (Wnv,1) = WR*cos(WTeta);

        Wspolfi (Wnv,1) = (Wfi * 180)/pi ;
        WspolTeta (Wnv,1) = (WTeta * 180)/pi;
        Wnv=Wnv+1;
%         figure(1); plot(WX, WY, 'x'); title('rozklad wokseli')
%         pause(1)
    end
end
Wnv=Wnv-1;
% %% ------------------- DRAW: Rysowanie rozkładu wokseli na sferze
% figure(2); plot3(WX, WY, WZ); title('rozklad wokseli')
% scatter3(WX, WY, WZ, 50)
% disp ('matrix WX');
% disp (WX);
% disp ('matrix WY');
% disp (WY);
% disp ('matrix WZ');
% disp (WZ);
% % % -------------------- end - DRAW: Rysowanie rozkładu wokseli na
sferze
%% ------------------------------------------------------------------------
% ELEKTRODY !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Estepfi1 = 36*pi/180;
Estepfi2 = 45*pi/180;
Estepfi3 = 60*pi/180;
Ene=32;
EX=zeros(Ene,1);
EY=zeros(Ene,1);
EZ=zeros(Ene,1);
ER=80;
Ene=1;
ETeta=72*pi/180 ;
% disp( ETeta );
for Efi1=(0)*pi/180 : Estepfi1 : (72)*pi/180;
    Eox=ER*cos(Efi1)*sin(ETeta);
    Eoy=ER*sin(Efi1)*sin(ETeta);
    Eoz=ER*cos(ETeta);
    EX (Ene,1)= Eox;
    EY (Ene,1)= Eoy;
    EZ (Ene,1)= Eoz;
    Ene=Ene+1;
end
% nazwy elektrod_____________________________
T7X = EX (1,1);
T7Y = EY (1,1);
T7Z = EZ (1,1);
RawX(1,1) = T7X;
RawY(1,1) = T7Y;
RawZ(1,1) = T7Z; label = ['T7']; RawLabel(1,1) = cellstr(label);
P7X = EX (2,1);
P7Y = EY (2,1);
P7Z = EZ (2,1);
```

```
RawX(2,1) = P7X;
RawY(2,1) = P7Y;
RawZ(2,1) = P7Z; label = ['P7']; RawLabel(2,1) = cellstr(label);
O1X = EX (3,1);
O1Y = EY (3,1);
O1Z = EZ (3,1);
RawX(3,1) = O1X;
RawY(3,1) = O1Y;
RawZ(3,1) = O1Z; label = ['O1']; RawLabel(3,1) = cellstr(label);
%-------------------------------------------------------------------------
-
Efi1 = (90)*pi/180;
ETeta = 72*pi/180;
EX(4,1)=ER*cos(Efi1)*sin(ETeta);
EY(4,1)=ER*sin(Efi1)*sin(ETeta);
EZ(4,1)=ER*cos(ETeta);
OZX=EX(4,1);
OZY=EY(4,1);
OZZ=EZ(4,1);
RawX(4,1) = OZX;
RawY(4,1) = OZY;
RawZ(4,1) = OZZ; label = ['OZ']; RawLabel(4,1) = cellstr(label);
%-------------------------------------------------------------------------
-
Ene = 5
ETeta=72*pi/180 ;
% disp( ETeta );
for Efi1=(108)*pi/180 : Estepfi1 : (360)*pi/180-Estepfi1
    Eox=ER*cos(Efi1)*sin(ETeta);
    Eoy=ER*sin(Efi1)*sin(ETeta);
    Eoz=ER*cos(ETeta);
    EX (Ene,1)= Eox;
    EY (Ene,1)= Eoy;
    EZ (Ene,1)= Eoz;
    Ene=Ene+1;
end
O2X = EX (5,1);
O2Y = EY (5,1);
O2Z = EZ (5,1);
RawX(5,1) = O2X;
RawY(5,1) = O2Y;
RawZ(5,1) = O2Z; label = ['O2']; RawLabel(5,1) = cellstr(label);
P8X = EX (6,1);
P8Y = EY (6,1);
P8Z = EZ (6,1);
RawX(6,1) = P8X;
RawY(6,1) = P8Y;
RawZ(6,1) = P8Z; label = ['P8']; RawLabel(6,1) = cellstr(label);
T8X = EX (7,1);
T8Y = EY (7,1);
T8Z = EZ (7,1);
RawX(7,1) = T8X;
RawY(7,1) = T8Y;
RawZ(7,1) = T8Z; label = ['T8']; RawLabel(7,1) = cellstr(label);
F8X = EX (8,1);
F8Y = EY (8,1);
F8Z = EZ (8,1);
RawX(8,1) = F8X;
RawY(8,1) = F8Y;
RawZ(8,1) = F8Z; label = ['F8']; RawLabel(8,1) = cellstr(label)
FP2X = EX (9,1);
```

```
FP2Y = EY (9,1);
FP2Z = EZ (9,1);
RawX(9,1) = FP2X;
RawY(9,1) = FP2Y;
RawZ(9,1) = FP2Z; label = ['FP2']; RawLabel(9,1) = cellstr(label);
FP1X = EX (10,1);
FP1Y = EY (10,1);
FP1Z = EZ (10,1);
RawX(10,1) = FP1X;
RawY(10,1) = FP1Y;
RawZ(10,1) = FP1Z; label = ['FP1']; RawLabel(10,1) = cellstr(label);
F7X = EX (11,1);
F7Y = EY (11,1);
F7Z = EZ (11,1);
RawX(11,1) = F7X;
RawY(11,1) = F7Y;
RawZ(11,1) = F7Z; label = ['F7']; RawLabel(11,1) = cellstr(label);
% nazwa elektrody_______________________________
Ene = 12;
ETeta=54*pi/180;
% disp( ETeta );
for Efi2=(18)*pi/180 : Estepfi2 : (360)*pi/180
    Eox=ER*cos(Efi2)*sin(ETeta);
    Eoy=ER*sin(Efi2)*sin(ETeta);
    Eoz=ER*cos(ETeta);
    EX (Ene,1)= Eox;
    EY (Ene,1)= Eoy;
    EZ (Ene,1)= Eoz;
    Ene=Ene+1;
end
% nazwy elektrod_______________________________
CP5X=EX(12,1);
CP5Y=EY(12,1);
CP5Z=EZ(12,1);
RawX(12,1) = CP5X;
RawY(12,1) = CP5Y;
RawZ(12,1) = CP5Z; label = ['CP5']; RawLabel(12,1) = cellstr(label);
PO3X=EX(13,1);
PO3Y=EY(13,1);
PO3Z=EZ(13,1);
RawX(13,1) = PO3X;
RawY(13,1) = PO3Y;
RawZ(13,1) = PO3Z; label = ['PO3']; RawLabel(13,1) = cellstr(label);
PO4X=EX(14,1);
PO4Y=EY(14,1);
PO4Z=EZ(14,1);
RawX(14,1) = PO4X;
RawY(14,1) = PO4Y;
RawZ(14,1) = PO4Z; label = ['PO4']; RawLabel(14,1) = cellstr(label);
CP6X=EX(15,1);
CP6Y=EY(15,1);
CP6Z=EZ(15,1);
RawX(15,1) = CP6X;
RawY(15,1) = CP6Y;
RawZ(15,1) = CP6Z; label = ['CP6']; RawLabel(15,1) = cellstr(label);
FC6X=EX(16,1);
FC6Y=EY(16,1);
FC6Z=EZ(16,1);
RawX(16,1) = FC6X;
RawY(16,1) = FC6Y;
RawZ(16,1) = FC6Z; label = ['FC6']; RawLabel(16,1) = cellstr(label);
```

```
AF4X=EX(17,1);
AF4Y=EY(17,1);
AF4Z=EZ(17,1);
RawX(17,1) = AF4X;
RawY(17,1) = AF4Y;
RawZ(17,1) = AF4Z; label = ['AF4']; RawLabel(17,1) = cellstr(label);
AF3X=EX(18,1);
AF3Y=EY(18,1);
AF3Z=EZ(18,1);
RawX(18,1) = AF3X;
RawY(18,1) = AF3Y;
RawZ(18,1) = AF3Z; label = ['AF3']; RawLabel(18,1) = cellstr(label);
FC5X=EX(19,1);
FC5Y=EY(19,1);
FC5Z=EZ(19,1);
RawX(19,1) = FC5X;
RawY(19,1) = FC5Y;
RawZ(19,1) = FC5Z; label = ['FC5']; RawLabel(19,1) = cellstr(label);
ETeta=36*pi/180;
% disp( ETeta );
for Efi2=(0)*pi/180 : Estepfi3 : (360)*pi/180-Estepfi3
    Eox=ER*cos(Efi2)*sin(ETeta);
    Eoy=ER*sin(Efi2)*sin(ETeta);
    Eoz=ER*cos(ETeta);
    EX (Ene,1)= Eox;
    EY (Ene,1)= Eoy;
    EZ (Ene,1)= Eoz;
    Ene=Ene+1;
end
% nazwy elektrod_____________________________
C3X=EX(20,1);
C3Y=EY(20,1);
C3Z=EZ(20,1);
RawX(20,1) = C3X;
RawY(20,1) = C3Y;
RawZ(20,1) = C3Z; label = ['C3']; RawLabel(20,1) = cellstr(label);
P3X=EX(21,1);
P3Y=EY(21,1);
P3Z=EZ(21,1);
RawX(21,1) = P3X;
RawY(21,1) = P3Y;
RawZ(21,1) = P3Z; label = ['P3']; RawLabel(21,1) = cellstr(label);
P4X=EX(22,1);
P4Y=EY(22,1);
P4Z=EZ(22,1);
RawX(22,1) = P4X;
RawY(22,1) = P4Y;
RawZ(22,1) = P4Z; label = ['P4']; RawLabel(22,1) = cellstr(label);
C4X=EX(23,1);
C4Y=EY(23,1);
C4Z=EZ(23,1);
RawX(23,1) = C4X;
RawY(23,1) = C4Y;
RawZ(23,1) = C4Z; label = ['C4']; RawLabel(23,1) = cellstr(label);
F4X=EX(24,1);
F4Y=EY(24,1);
F4Z=EZ(24,1);
RawX(24,1) = F4X;
RawY(24,1) = F4Y;
RawZ(24,1) = F4Z; label = ['F4']; RawLabel(24,1) = cellstr(label);
F3X=EX(25,1);
```

```
F3Y=EY(25,1);
F3Z=EZ(25,1);
RawX(25,1) = F3X;
RawY(25,1) = F3Y;
RawZ(25,1) = F3Z; label = ['F3']; RawLabel(25,1) = cellstr(label);
ETeta=18*pi/180;
% disp( ETeta );
for Efi2=(45)*pi/180 : Estepfi2 : (135)*pi/180
    Eox=ER*cos(Efi2)*sin(ETeta);
    Eoy=ER*sin(Efi2)*sin(ETeta);
    Eoz=ER*cos(ETeta);
    EX (Ene,1)= Eox;
    EY (Ene,1)= Eoy;
    EZ (Ene,1)= Eoz;
    Ene=Ene+1;
end
% nazwy elektrod_____________________________
CP1X=EX(26,1);
CP1Y=EY(26,1);
CP1Z=EZ(26,1);
RawX(26,1) = CP1X;
RawY(26,1) = CP1Y;
RawZ(26,1) = CP1Z; label = ['CP1']; RawLabel(26,1) = cellstr(label);
PZX=EX(27,1);
PZY=EY(27,1);
PZZ=EZ(27,1);
RawX(27,1) = PZX;
RawY(27,1) = PZY;
RawZ(27,1) = PZZ; label = ['PZ']; RawLabel(27,1) = cellstr(label);
CP2X=EX(28,1);
CP2Y=EY(28,1);
CP2Z=EZ(28,1);
RawX(28,1) = CP2X;
RawY(28,1) = CP2Y;
RawZ(28,1) = CP2Z; label = ['CP2']; RawLabel(28,1) = cellstr(label);
ETeta=18*pi/180;
% disp( ETeta );
for Efi2=(225)*pi/180 : Estepfi2 : (315)*pi/180
    Eox=ER*cos(Efi2)*sin(ETeta);
    Eoy=ER*sin(Efi2)*sin(ETeta);
    Eoz=ER*cos(ETeta);
    EX (Ene,1)= Eox;
    EY (Ene,1)= Eoy;
    EZ (Ene,1)= Eoz;
    Ene=Ene+1;
end
% nazwy elektrod_____________________________
FC2X=EX(29,1);
FC2Y=EY(29,1);
FC2Z=EZ(29,1);
RawX(29,1) = FC2X;
RawY(29,1) = FC2Y;
RawZ(29,1) = FC2Z; label = ['FC2']; RawLabel(29,1) = cellstr(label);
FZX=EX(30,1);
FZY=EY(30,1);
FZZ=EZ(30,1);
RawX(30,1) = FZX;
RawY(30,1) = FZY;
RawZ(30,1) = FZZ; label = ['FZ']; RawLabel(30,1) = cellstr(label);
FC1X=EX(31,1);
FC1Y=EY(31,1);
```

```
FC1Z=EZ(31,1);
RawX(31,1) = FC1X;
RawY(31,1) = FC1Y;
RawZ(31,1) = FC1Z; label = ['FC1']; RawLabel(31,1) = cellstr(label);
% Cz
EX(32,1)=0;
EY(32,1)=0;
EZ(32,1)=ER;
% nazwa elektrody_____________________________
CZX=EX(32,1);
CZY=EY(32,1);
CZZ=EZ(32,1);
RawX(32,1) = CZX;
RawY(32,1) = CZY;
RawZ(32,1) = CZZ; label = ['CZ']; RawLabel(32,1) = cellstr(label);
% A1 - elektroda odniesienia
A1X = ER;
A1Y = 0;
A1Z = ER*cos(100*pi/180);
% ------------------- DRAW: Rysowanie elektrod układu 10-20
    %%%% Nałożenie elektrod na woksele
%     hold on
%     scatter3(RawX, RawY, RawZ,50, 'filled', 'r');
%     xlabel('[mm]');
%     ylabel('[mm]');
%     zlabel('[mm]');
    %%%% koniec: Nałożenie elektrod na woksele
% scatter3(RawX, RawY, RawZ)
% figure(3); scatter3(RawX, RawY, RawZ); title('elektrody IDIAP - raw');
% text (RawX, RawY, RawZ, RawLabel);
%
% figure(4); scatter3(RawX, RawY, RawZ); title('elektrody IDIAP - raw');
% text (RawX, RawY, RawZ, RawLabel);
% figure(5); scatter(RawX, RawY); title('elektrody IDIAP - raw');
% text (RawX, RawY, RawZ, RawLabel);
% ------------------- end DRAW: Rysowanie elektrod układu 10-20
% PreprocX = zeros (8,1);
% PreprocY = zeros (8,1);
% PreprocZ = zeros (8,1);
%Elektrody dla danych IDIAP - 8 centroparietal channels
% % C3
% PreprocX (1,1) = C3X;
% PreprocY (1,1) = C3Y;
% PreprocZ (1,1) = C3Z; label = ['C3']; PreprocLabel(1,1) =
cellstr(label);
% % CZ
% PreprocX (2,1) = CZX;
% PreprocY (2,1) = CZY;
% PreprocZ (2,1) = CZZ; label = ['CZ']; PreprocLabel(2,1) =
cellstr(label);
% % C4
% PreprocX (3,1) = C4X;
% PreprocY (3,1) = C4Y;
% PreprocZ (3,1) = C4Z; label = ['C4']; PreprocLabel(3,1) =
cellstr(label);
% % CP1
% PreprocX (4,1) = CP1X;
% PreprocY (4,1) = CP1Y;
% PreprocZ (4,1) = CP1Z; label = ['CP1']; PreprocLabel(4,1) =
cellstr(label);
% % CP2
```

```
% PreprocX (5,1) = CP2X;
% PreprocY (5,1) = CP2Y;
% PreprocZ (5,1) = CP2Z; label = ['CP1']; PreprocLabel(5,1) =
cellstr(label);
% % P3
% PreprocX (6,1) = P3X;
% PreprocY (6,1) = P3Y;
% PreprocZ (6,1) = P3Z; label = ['P3']; PreprocLabel(6,1) =
cellstr(label);
% % PZ
% PreprocX (7,1) = PZX;
% PreprocY (7,1) = PZY;
% PreprocZ (7,1) = PZZ; label = ['PZ']; PreprocLabel(7,1) =
cellstr(label);
% % P4
% PreprocX (8,1) = P4X;
% PreprocY (8,1) = P4Y;
% PreprocZ (8,1) = P4Z; label = ['P4']; PreprocLabel(8,1) =
cellstr(label);
%end: Elektrody dla danych IDIAP - 8 centroparietal channels
PreprocX = zeros (14,1);
PreprocY = zeros (14,1);
PreprocZ = zeros (14,1);
% %Elektrody dla headsetu Emotiv
%14 channels: AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8,
AF4(2?)
%References: In the CMS/DRL noise cancellation configuration P3/P4
locations
% AF3 - "18"
PreprocX (1,1) = AF3X;
PreprocY (1,1) = AF3Y;
PreprocZ (1,1) = AF3Z; label = ['AF3']; PreprocLabel(1,1) =
cellstr(label);
% F7 - "11"
PreprocX (2,1) = F7X;
PreprocY (2,1) = F7Y;
PreprocZ (2,1) = F7Z; label = ['F7']; PreprocLabel(2,1) =
cellstr(label);
% F3 - "25"
PreprocX (3,1) = F3X;
PreprocY (3,1) = F3Y;
PreprocZ (3,1) = F3Z; label = ['F3']; PreprocLabel(3,1) =
cellstr(label);
% FC5 - "19"
PreprocX (4,1) = FC5X;
PreprocY (4,1) = FC5Y;
PreprocZ (4,1) = FC5Z; label = ['FC5']; PreprocLabel(4,1) =
cellstr(label);
% T7 - "1"
PreprocX (5,1) = T7X;
PreprocY (5,1) = T7Y;
PreprocZ (5,1) = T7Z; label = ['T7']; PreprocLabel(5,1) =
cellstr(label);
% P7 - "2"
PreprocX (6,1) = P7X;
PreprocY (6,1) = P7Y;
PreprocZ (6,1) = P7Z; label = ['P7']; PreprocLabel(6,1) =
cellstr(label);
% O1 - "3"
PreprocX (7,1) = O1X;
PreprocY (7,1) = O1Y;
```

```
PreprocZ (7,1) = O1Z; label = ['O1']; PreprocLabel(7,1) =
cellstr(label);
% O2 - "5"
PreprocX (8,1) = O2X;
PreprocY (8,1) = O2Y;
PreprocZ (8,1) = O2Z; label = ['O2']; PreprocLabel(8,1) =
cellstr(label);
% P8 - "6"
PreprocX (9,1) = P8X;
PreprocY (9,1) = P8Y;
PreprocZ (9,1) = P8Z; label = ['P8']; PreprocLabel(9,1) =
cellstr(label);
% T8 - "7"
PreprocX (10,1) = T8X;
PreprocY (10,1) = T8Y;
PreprocZ (10,1) = T8Z; label = ['T8']; PreprocLabel(10,1) =
cellstr(label);
% FC6 - "16"
PreprocX (11,1) = FC6X;
PreprocY (11,1) = FC6Y;
PreprocZ (11,1) = FC6Z; label = ['FC6']; PreprocLabel(11,1) =
cellstr(label);
% F4 - "24"
PreprocX (12,1) = F4X;
PreprocY (12,1) = F4Y;
PreprocZ (12,1) = F4Z; label = ['F4']; PreprocLabel(12,1) =
cellstr(label);
% F8 - "8"
PreprocX (13,1) = F8X;
PreprocY (13,1) = F8Y;
PreprocZ (13,1) = F8Z; label = ['F8']; PreprocLabel(13,1) =
cellstr(label);
% AF4 - "17"
PreprocX (14,1) = AF4X;
PreprocY (14,1) = AF4Y;
PreprocZ (14,1) = AF4Z; label = ['AF4']; PreprocLabel(14,1) =
cellstr(label);
% %end: Elektrody dla headsetu Emotiv
% ------------------- DRAW: Rysowanie rozkładu elektrod IDIAP dla danych
przetworzonych
    %%% Nałożene elektrod IDIAP/Emotiv dla danych przetworzonych na
woksele
%     hold on
%     scatter3(PreprocX,PreprocY,PreprocZ,50, 'filled', 'r')
%     xlabel('[mm]');
%     ylabel('[mm]');
%     zlabel('[mm]');
    %%% koniec: Nałożenie elektrod IDIAP dla danych przetworzonych na
woksele
% figure(6); scatter3(PreprocX, PreprocY, PreprocZ); title('elektrody
IDIAP-preprocessed');
% text (PreprocX, PreprocY, PreprocZ, PreprocLabel);
% figure(7); scatter(PreprocX, PreprocY); title('elektrody IDIAP-
preprocessed');
% text (PreprocX, PreprocY, PreprocZ, PreprocLabel);
% hold on
% %scatter3(PreprocX,PreprocY+100,PreprocZ,50, 'filled', 'r')
% scatter3(PreprocX,PreprocY,PreprocZ,50, 'filled', 'r')
% ------------------- end DRAW: Rysowanie rozkładu elektrod IDIAP dla
danych przetworzonych
%--------------------------------------------------------------------
```

```
% Generowanie macierzy K
trzyWnv=3*Wnv;
K=zeros(14,trzyWnv);
for i=1:14
    u=1;
    for j=1:Wnv
        a = PreprocX(i,1)-WX(j,1);
        b = PreprocY(i,1)-WY(j,1);
        c = PreprocZ(i,1)-WZ(j,1);
        A = 1/( (4*pi*0.00033) * sqrt(a^2 + b^2 + c^2));
        Ax = A*a;
        Ay = A*b;
        Az = A*c;
        Ra = A1X-WX(j,1);
        Rb = A1Y-WY(j,1);
        Rc = A1Z-WZ(j,1);
        B = 1/( (4*pi*0.33) * sqrt(Ra^2 + Rb^2 + Rc^2));
        RBx = B*Ra;
        RBy = B*Rb;
        RBz = B*Rc;
        X = Ax-RBx;
        Y = Ay-RBy;
        Z = Az-RBz;
        K(i,u) = X;
        u=u+1;
        K(i,u) = Y;
        u=u+1;
        K(i,u) = Z;
        u=u+1;
    end
end
X = [];
X = EMOTIV(save_time_i:save_time_j, :);
sizeX = size( X, 1 );
xx = 1;
wb = 1;
pp = 1;
wskaznik3 = 0;
wskaznik3proc = 0;
liczba3 = 0;
wskaznik2 = 0;
wskaznik2proc = 0;
liczba2 = 0;
%% ------------------- DRAW: rysowanie pustej figury z przyciskiem
"Pause"
% fig = figure(8);
% drawfig = 1;
% global bPressed;
% bPressed = false;
% if( drawfig == 1 )
%     hb = uicontrol( 'Style', 'pushbutton', 'String', 'Pause',...
%                     'Position', [20 20 50 20],...
%                     'Callback', @OnButton );
%
%     set (hb,'HandleVisibility','off');
% end
% ------------------- end DRAW: rysowanie pustej figury z przyciskiem
"Pause"
macPunktywyszukanejj = zeros (Wnv, 1);
klasyfikacja = zeros (sizeX,1);
macierzsigma = zeros (sizeX,1);
```

```
macierzm = zeros (sizeX, 1);
maczaimportowaneFi = [];
sprFi = [];
for aa=1:1:sizeX
% % ------------------- DRAW: Widok wokseli i elektrod dla danych
przetworzonych
% % Idiap (był już wcześniej)
%     if bPressed
%         waitforbuttonpress;
%         bPressed = false;
%         if ishandle( hb )
%             set( hb, 'Enable', 'on' );
%         end
%     end
%     figure (9);
%     if( drawfig == 1 )
%         clf;
%         hold on;
%         scatter3(WX, WY+100, WZ, 50)
%         hold on
%         scatter3(PreprocX,PreprocY+100,PreprocZ,50, 'r')
%     end
% % ------------------- end DRAW: Widok wokseli i elektrod dla danych
przetworzonych
    zaimportowaneFi = X( aa, freq_value:12:168 )';
    maczaimportowaneFi = [maczaimportowaneFi zaimportowaneFi];
    zaimportowaneFiMaximum = max (zaimportowaneFi);
    zaimportowaneFiMinimum = min (zaimportowaneFi);
    zaimportowaneFiMinMaxroznica = zaimportowaneFiMaximum -
zaimportowaneFiMinimum;
    intervalno = 20;
    zaimportowaneFiPrzedzial = zaimportowaneFiMinMaxroznica/intervalno;
    ZEWc=zeros(14,3);
% % ------------------- DRAW: rozrysowanie 14 elektrod headsetu Emotiv
Epoc
%     if( drawfig == 1 )
%         colors = jet(intervalno);
%         for i=1:14
%             val = zaimportowaneFi (i,1);
%             if val == zaimportowaneFiMinimum
%                 indexcolor = 1;
%             else
%                 indexcolor = ceil( ((val-
zaimportowaneFiMinimum)/(zaimportowaneFiMaximum-
zaimportowaneFiMinimum))*intervalno );
%             end
%             ZEWc(i,:)=colors(indexcolor,:);
%         end
%         hold on
%         scatter3 (PreprocX,PreprocY+100,PreprocZ, 50, ZEWc, 'filled');
%     end
% % ------------------- end DRAW: rozrysowanie 14 elektrod headsetu
Emotiv Epoc
%--------------------------------------------------------------------
%--------------------------------------------------------------------
    % pseudoinv K, qr decomposition K
    H=[];
    for a=1:Nel
        for b=1:Nel
            if a==b
                H(a,b)=1-(1/Nel);
```

```
            else
                H(a,b)=(-1/Nel);
            end
        end
    end
    K=H*K;
    zaimportowaneFi=H*zaimportowaneFi;
    rankK=rank(K);
%     rankKT = rank(K');
%     rankKTK = rank((K')*K);
%     rankKKT = rank(K*(K'));
%%%%%%%%%% Współczynnik uwarunkowania macierzy K
%     Ktran = K';
%     KtranK = Ktran*K;
%     D = eig(KtranK);
%     j = 1
%     for i = 1:1:size(D)
%         if D(i) > 10000
%             NowaD(j) = D(i);
%             j = j + 1;
%         end
%     end
%     MaxNowaD = max(NowaD);
%     MinNowaD = min(NowaD);
%     IlorazNowaD = sqrt(MaxNowaD)/sqrt(MinNowaD);
%%%%%%%%%% koniec: Współczynnik uwarunkowania macierzy K
    PseudoinvK=pinv(K);
    szukanaJJ=PseudoinvK*zaimportowaneFi;
    JX=zeros (Wnv,1);
    m=1;
    for n=1:3:trzyWnv;
        JX (m,1) = szukanaJJ(n,1);%     [U,S,V] = svd(K);
        m=m+1;
    end
    JY=zeros (Wnv,1);
    m=1;
    for n=2:3:trzyWnv;
        JY (m,1) = szukanaJJ(n,1);
        m=m+1;
    end
    JZ=zeros (Wnv,1);
    m=1;
    for n=3:3:trzyWnv;
        JZ (m,1) = szukanaJJ(n,1);
        m=m+1;
    end
    Punktywyszukanejj=zeros(Wnv,1);
    j=0;
    Maks1 = 0;
    Maks2 = 0;
    Maks3 = 0;
    WspolrzednaMaks1 = 1;
    WspolrzednaMaks2 = 1;
    WspolrzednaMaks3 = 1;
    for j=1:Wnv;
        Punktywyszukanejj(j,1) = sqrt(
(JX(j,1)^2)+(JY(j,1)^2)+(JZ(j,1)^2) );
    end;
    macPunktywyszukanejj( : ,pp ) = Punktywyszukanejj;
    trMatrixFoundPointsJJ = macPunktywyszukanejj';
    pp = pp + 1;
```

```
    % plot (1000000 * Punktywyszukanejj);
    PunktywyszukanejjMaximum = max (Punktywyszukanejj);
    PunktywyszukanejjMinimum = min (Punktywyszukanejj);
    PunktywyszukanejjMinMaxroznica = PunktywyszukanejjMaximum -
PunktywyszukanejjMinimum;
    posortowane = sort (Punktywyszukanejj);
    FiprosteJJ=K*szukanaJJ;
    macFiprosteJJ( : ,xx ) = FiprosteJJ;
    xx=xx+1;
end
    % spr Fi
    for i=1:size(macFiprosteJJ,2)
        for j=1:size(macFiprosteJJ,1)
            sprFi(j,i) = macFiprosteJJ(j,i) - maczaimportowaneFi(j,i);
        end
    end
    % end: spr Fi
%% Check
    T = PseudoinvK;
    Hn = K*T;
    Fi1 = K*szukanaJJ;
    Fi2 = Hn*zaimportowaneFi;
    difFi = Fi1 - Fi2;
    difFi1 = zaimportowaneFi - Fi1;
    difFi2 = zaimportowaneFi - Fi2;
    R = T*K;
    spr_szukanaJJ = R*szukanaJJ;
    sprJ = szukanaJJ - spr_szukanaJJ;
%% end: Check
%% T-statystyka ------------------------------------------------------------
-------------
TetaStep = 5;
TetaCompartment = 15;
FiStep = 10;
FiCompartment = 30;
compartment_x = 360/FiCompartment;
compartment_y = 90/TetaCompartment;
wholeCompartment = compartment_x*compartment_y;
halfOfWholeCompartment = (wholeCompartment/2)+1;
compartment_number = 0;
%%% Wejściem do obliczenia t-statystyki są rozwiązania zagadnienia
odwrotnego
%%% uśrednione na obszarach P zapisane w macierzy trMatrixFoundPointsJJ
%%% wartosc srednia w poszczególnych obszarach P (compartment)
for aa=1:1:sizeX;
    tab = trMatrixFoundPointsJJ( aa, : );
    compartment = Emotiv_przedzialy_90( tab, TetaStep, FiStep,
TetaCompartment, FiCompartment );
    compartment = compartment( halfOfWholeCompartment:wholeCompartment
);
    compartment_number = length( compartment );

    for i=1:1:compartment_number
        le = length( compartment(i).tab );
        my = sum( compartment(i).tab );
        array_my(aa,i) = my / le;
    end
end;
wholeMediumval_y = zeros(1, compartment_number);
for i=1:1:compartment_number
    tab = array_my( :, i );
```

```
    len = length( tab );
    sum_value = sum( tab );
    wholeMediumval_y(1, i) = sum_value / len;
end
%%% koniec: wartosc srednia w poszczególnych obszarach P (compartment)
%%% wariancje dla obszarów P
count_y = sizeX;
overallVariance_y = zeros( 1, compartment_number );
for aa=1:1:sizeX;
    tab = trMatrixFoundPointsJJ( aa, : );
    compartment = Emotiv_przedzialy_90( tab, TetaStep, FiStep,
TetaCompartment, FiCompartment );
    compartment = compartment(halfOfWholeCompartment:wholeCompartment);
    compartment_number = length( compartment );
    for i=1:1:compartment_number
        medium_value = wholeMediumval_y(i);
        tab = array_my(aa,i);
        overallVariance_y(1, i) = (overallVariance_y(1, i) + (tab -
medium_value)^2);
    end
end;
for i=1:1:compartment_number
    overallVariance_y(1, i) = overallVariance_y(1, i) / count_y;
end
%%% koniec: wariancje dla obszarów P
%%% wartości t-statystyki
matrix = zeros( compartment_number, compartment_number );
for i=1:1:compartment_number
    for j=1:1:compartment_number
        x_d_1 = wholeMediumval_y(i);
        x_d_2 = wholeMediumval_y(j);
        s_1 = overallVariance_y(1, i);
        s_2 = overallVariance_y(1, j);
        t = (x_d_1-x_d_2)/sqrt((s_1/count_y)+(s_2/count_y));
        matrix(i,j) = t;
    end
end
%%% koniec: wartosci t-statystyki
%% matrix - dekodowanie_____________________________________________
x2 = 1;
x3 = 1;
y2 = 1;
y3 = 1;
dekodowanie_ab_naukaklas=[];
dekodowanie_cb_naukaklas=[];
dekodowanie_3ab_danetest=[];
dekodowanie_2ab_danetest=[];
dekodowanie_3cb_danetest=[];
dekodowanie_2cb_danetest=[];
for i=1:1:compartment_number
    for j=1:1:compartment_number
        if j > i
            if strcmp(compartment(j).o, 'b1') ||
strcmp(compartment(j).o, 'b2')
                if strcmp(compartment(i).o, 'a')
                    value = matrix( i, j );
                    if value < 0
                        dekodowanie_3ab_danetest( x3, : ) = [value i
compartment(i).fi compartment(i).teta j compartment(j).fi
compartment(j).teta];
                        x3 = x3+1;
```

```
                    else
                        dekodowanie_2ab_danetest( x2, : ) = [value i
compartment(i).fi compartment(i).teta j compartment(j).fi
compartment(j).teta];
                        x2 = x2+1;
                    end
                end
            end
        end
        if i > j
            if strcmp(compartment(j).o, 'b1') ||
strcmp(compartment(j).o, 'b2')
                if strcmp(compartment(i).o, 'c')
                    value = matrix( i, j );
                    if value < 0
                        dekodowanie_3cb_danetest( y3, : ) = [value i
compartment(i).fi compartment(i).teta j compartment(j).fi
compartment(j).teta];
                        y3 = y3+1;
                    else
                        dekodowanie_2cb_danetest( y2, : ) = [value i
compartment(i).fi compartment(i).teta j compartment(j).fi
compartment(j).teta];
                        y2 = y2+1;
                    end
                end
            end
        end
    end
end
%%koniec: matrix - dekodowanie_____________________________________
disp ('koniec dekodowanie wstepne');
```

Plik „Emotiv_przedzialy_90”

```
%%% Funkcja dzieląca korę mózgową na obszary P o zadanych wg zadanych
%%% parametrów Fi i Teta
function compartment_array = przedzialy_90( tablica, WTO, WFO, WT, WF )
% WTO, WFO, WT, WF: Tetakrok, Fikrok, Tetaprzedzial, Fiprzedzial
% stopnie = ( radiany * 180/pi )
% radiany = ( stopnie * pi/180 )
WT_MAX = 90;
WF_MAX = 360;
WCT = (WT_MAX / WTO) / (WT_MAX / WT);    % liczba wokseli Teta
WCF = (WF_MAX / WFO) / (WF_MAX / WF);  % liczba wokseli Fi
WC = (WF_MAX / WF) * (WT_MAX / WT);    % liczba przedziałów
% inicjowanie tablic dla przedziałów w pionie
p = 1;
for wf=1:WF:WF_MAX
    compartment_teta(p).tab = [];
    p = p + 1;
end;
% wybranie wokseli z tablicy
i = 1;
o = 'a';
for wt=WT_MAX:-WTO:WTO
    p = 1;
    for wf=(1):(WF):(WF_MAX)
        for wc=1:1:WCF
            compartment_teta(p).tab = [ compartment_teta(p).tab tablica(
i ) ];
            compartment_teta(p).fi = wf + WF/2 - 1;
```

```
            compartment_teta(p).o = o;
            i = i + 1;
        end
        if mod(i-1,WCT*WCF) == 0
            if strcmp(o, 'a')
                o = 'b1';
            elseif strcmp(o, 'b1')
                o = 'b2';
            elseif strcmp(o, 'b2')
                o = 'c';
            elseif strcmp(o, 'c')
                o = 'a';
            end
        end
        p = p + 1;
    end
end
% inicjowanie tablic dla przedziałów
p = 1;
for wf=1:1:WC
    compartment_array(p).tab = [];
    p = p + 1;
end;
i = 1;
j = 1;
for wt=WT_MAX:-WT:WT
    p = 1;
    for wf=1:WF:WF_MAX
        % Definiowanie, co będzie w polach  przedzial.tab,
przedzial.Tetakrok,
        % przedzial.Fikrok, itd... przedzial.Tetaprzedzial,
przedzial.Fiprzedzial
        compartment_array(j).tab = compartment_teta(p).tab( i : i - 1 +
WCT * WCF );
        compartment_array(j).o = compartment_teta(p).o;
        compartment_array(j).fi = compartment_teta(p).fi;
        teta = wt - WT/2;
        compartment_array(j).teta = teta;

        j = j + 1;
        p = p + 1;
    end
    i = i + WCT * WCF;
end

%     for i=1:1:length(compartment_array)
%         fi(i) = compartment_array(i).fi
%         teta(i) = compartment_array(i).teta
%         figure(4); scatter(fi, teta); title('przedzialy');
%         pause(1);
%     end
```

Plik „Emotiv_Gower.m”

```
function [calkowitedrzewomin,znacznik,wezelwaga] =
Emotiv_Gower(dekodowanie_danetest, prog_value)
X=dekodowanie_danetest(:, [1, 2, 5] );
for z=1:1:size(X,1)
    if X(:,1)<=0
        znacznik(z,1)=1;
        z=z+1;
```

```
    else
        znacznik(z,1)=0;
        z=z+1;
end
end
for i=1:1:size(X,1)
    if X(i,1)<0
        X(i,1)=abs(X(i,1));
    end
end
calaX = [];
ROWSX = size(X,1);
COLSX = size(X,2);
for i=1:1:size(X,1)
    rzad = X(i,2);
    kol = X(i,3);

    calaX (rzad,kol) = X(i,1);
end;
Kdaszek2 = zeros (length(X(:,2)), 1);
Kdaszek3 = zeros (length(X(:,3)), 1);
Kdaszek23 = [Kdaszek2; Kdaszek3];
Kdaszekcaly = zeros (length (Kdaszek23(:,1)), 1);
klineUnique = zeros (length (Kdaszek23(:,1)),1);
Kdaszek = zeros (length (Kdaszekcaly(:,1)),2);

for i = 1:1:size(Kdaszek2)
    Kdaszek2 (i) = X(i,2);
end;
for i = 1:1:size(Kdaszek3)
    Kdaszek3 (i) = X(i,3);
end;
Kdaszekcaly = [Kdaszek2; Kdaszek3];
sprawedzeniewezlywezelwaga = unique (Kdaszekcaly);
sprawedzeniewezlycalkowitedrzewomin = unique (Kdaszekcaly);
klineUnique = unique (Kdaszekcaly);
Kdaszekuniquenastale = unique (Kdaszekcaly);
onesmatrix = ones (length (klineUnique),1);
Kdaszek = [klineUnique onesmatrix];
Xprim = [];
newXprim = klineUnique (2,1);
klineUniqueEmpty = 0;
mainKlineUniqueEmpty = 0;
mainKlineUnique = klineUnique;
cumulativeArrayROfWeightTmp = [];
calkowitedrzewomin = [];
cumulativeArrayROfWeightTmp_index=0;
%% Algorytm Govera: wybieranie pierwszego elementu Xprim=as i
sprawdzanie co z prawej lub lewej kolumny krawędzi (klineUnique) łączy
się z as.
%% Następnie wszystkie pary połączeń zapisywane są do macierzy i
wybierane jest to z największą wagą. Krawędź ai, która połączyła się z
krawędzią
%% as z tą największą wagą staje się kolejnym as. Odejmujemy ai, który
połączył się z as z największą wagą ze zbioru klineUnique
%% i dodajemy go do zbioru as (mamy 2 as, a potem odpowiednio więcej. I
sprawdzamy po kolei pozostałe ai w zbiorze klineUnique, czy łączą się
%% z którymś as. Jeżeli tak, to zapamiętujemy to ai w zbiorze
"arrayAiAsRPartial", z którego następnie wybierane jest najmocniejsze
połączenie
```

```
%% i dodawane do "cumulativeArrayROfWeightTmp". Wszystkie rekordy z
najmocniejszymi połączeniami po sprawdzeniu kolejnych ai są
przechowywane w
%% "cumulativeArrayROfWeightTmp". Następnie program sprawdza, z którym
as ostatnio przyłączone ai miało to najmocniejsze połączenie
(najmocniejsze
%% połączenie to osatni elementy każedego rekordu
"cumulativeArrayROfWeightTmp". Z tego powstaje "calkowitedrzewomin",
czyli dwie krawędzie wybrane
%% z każdego rekordu "cumulativeArrayROfWeightTmp", dla których
zaistniało to najmocniejsze połączenie.
%% Na koniec wprowadzany jest próg - w drzewie zostają tylko krawędzie o
połączeniach więszych niż próg.
%%% Algorytm Gowera:
%%% Z drzewa całkowitego wybierane są krawędzie tak, żeby ich suma przy
%%% tworzeniu drzewa była jak największa i w poszczególnych krokach
%%% tworzona jest struktura drzewa minimalnego
while mainKlineUniqueEmpty ~= 1
    Xprim = [ Xprim newXprim ];
    mainKlineUnique = setdiff( mainKlineUnique, newXprim );
    klineUnique = mainKlineUnique;
    arrayAiAsRPartial = [];
    arrayRWeightTmp = [];
    matrixSumOfWeight = [];
    klineUniqueEmpty = isempty(klineUnique);
    while klineUniqueEmpty ~= 1
        % wez pierwszy element
        ai = klineUnique(1);
        Xi = setdiff (Kdaszekuniquenastale, ai);
        productXprimXi = intersect( Xprim, Xi );
      % sprawdzenie 'checkProductXprimXi', czy zbiór productXprimXi
      % jest pusty. Jeżeli pusty sprawdzenie = 1, jeżeli nie jest
      % pusty sprawdzenie = 0;
        checkProductXprimXi = isempty(productXprimXi);
        while checkProductXprimXi ~= 1
            as = productXprimXi;
            %Utwórz drzewo
            R = [];
            sumValue = [];
            uu = size(X,1);
            for b = 1:1:uu
                if X(b,2) == ai;
                    if isempty( intersect( X(b,3), as ) ) == 0
                        R = [R X(b,1)];
                    end
                elseif X(b,3) == ai
                    if isempty( intersect( X(b,2), as ) ) == 0
                        R = [R X(b,1)];
                    end;
                end;
            end;
            if length(R)~=0
                sumValue = [sumValue;R'];
                sumOfWeight = sum(sumValue);
                matrixSumOfWeight = [matrixSumOfWeight; sumOfWeight];
                arrayRWeightTmp = [arrayRWeightTmp; R'];
                for i=1:1:size(R,2)
                    valueR=R(i);
                    arrayAiAsRPartial = [ arrayAiAsRPartial; ai as
valueR];
                end;
```

```
            end;
            roznicasprawdzenieiloczynXprimXi = setdiff
(productXprimXi,as);
            checkProductXprimXi =
isempty(roznicasprawdzenieiloczynXprimXi);
        end;
        klineUnique = setdiff( klineUnique, ai );
        klineUniqueEmpty = isempty(klineUnique);
    end;
    cumulativeArrayROfWeightTmp_index =
cumulativeArrayROfWeightTmp_index + 1;
    maxTR = -10e37;
    countT_ROW = size( arrayAiAsRPartial, 1 );
    countT_COL = size( arrayAiAsRPartial, 2 );
    for i=1:1:countT_ROW
        valueTR = arrayAiAsRPartial( i, countT_COL );
        if maxTR < valueTR
            maxTR = valueTR;
            cumulativeArrayROfWeightTmp(
cumulativeArrayROfWeightTmp_index ).t = arrayAiAsRPartial( i,: );
        end
    end
    mainKlineUniqueEmpty = isempty(mainKlineUnique);
    if length(arrayAiAsRPartial)==0
        mainKlineUniqueEmpty = 1;
cumulativeArrayROfWeightTmp_index=cumulativeArrayROfWeightTmp_index-1;
    end;
    newXprim =
cumulativeArrayROfWeightTmp(cumulativeArrayROfWeightTmp_index).t(1);
end;
%%% koniec: Algorytm Gowera:
%%% Z drzewa całkowitego wybierane są krawędzie tak, żeby ich suma przy
%%% tworzeniu drzewa była jak największa i w poszczególnych krokach
%%% tworzona jest struktura drzewa minimalnego
%-------------------------------------------------------------------------
--
%-------------------------------------------------------------------------
--
%%% Zapisanie informacji o drzewie minimalnym w tablicy
calkowitedrzewomin
dlugosc = length( cumulativeArrayROfWeightTmp );
ulala1 = 0;
ulala2 = 0;
maxpolaczenie = -10e40;
maxpolaczeniestare = -10e40;
for t=1:1:dlugosc
    tablica = cumulativeArrayROfWeightTmp( t );
    istotneCOLtablica = (size(tablica.t,2))-1;
    wierszeX = size(X,1);
    drzewominrobocze = [];
        for b = 1:1:size(X,1)
                ullla = X(b,2);
                t1 = tablica.t(1);
                if X(b,2) == tablica.t(1)
                    for i=2:1:istotneCOLtablica
                        ulala1 = intersect( X(b,3), tablica.t(i) );
                        if isempty(ulala1) == 0 &&
X(b,1)==tablica.t(istotneCOLtablica+1);
                            drzewominrobocze = [drzewominrobocze;
tablica.t(1) X(b,3) X(b,1)];
                        end;
```

```
                    end;
                elseif X(b,3) == tablica.t(1)
                    for i=2:1:istotneCOLtablica
                        ulala2 = intersect( X(b,2), tablica.t(i) );
                        if isempty(ulala2) == 0 &&
X(b,1)==tablica.t(istotneCOLtablica+1);
                           drzewominrobocze = [drzewominrobocze;
tablica.t(1) X(b,2) X(b,1)];
                        end;
                    end;
                end;
        end;
        ROWSdrzewominrobocze = size(drzewominrobocze,1);
    calkowitedrzewomin = [calkowitedrzewomin; drzewominrobocze];
end;
%%% Zapisanie informacji o drzewie minimalnym w tablicy
calkowitedrzewomin
%-------------------------------------------------------------------------
--
%-------------------------------------------------------------------------
--
%tworzenie zminimalizowanej macierzy po odcięciu poniżej progu
ROWSX = size(X,1);
COLSX = size(X,2);
ROWScalkowitedrzewomin = size(calkowitedrzewomin,1);
COLScalkowitedrzewomin = size(calkowitedrzewomin,2);
calaXmin = [];
for c=1:1:ROWScalkowitedrzewomin
    sprawdzenie = calkowitedrzewomin(c,3);
    for j=1:1:ROWSX
        if X(j,1) == sprawdzenie
            rzadmin = X(j,2);
            kolmin = X(j,3);
            calaXmin (rzadmin,kolmin) = sprawdzenie;
            abscalaXmin (rzadmin,kolmin) = abs(sprawdzenie);
        end;
    end;
end;
%poniższa zmienna próg wpisywana ręcznie
%próg obliczany jako pewnien procent
maxcalkowitedrzewomin = max (calkowitedrzewomin (:,3));
mincalkowitedrzewomin = min (calkowitedrzewomin (:,3));
roznicaminmax = (maxcalkowitedrzewomin - mincalkowitedrzewomin) *
prog_value;
prog = roznicaminmax + mincalkowitedrzewomin;
ROWScalaXmin = size (abscalaXmin,1);
COLScalaXmin = size (abscalaXmin,2);
wezelwaga = [];
for i=1:1:ROWScalaXmin
    for j=1:1:COLScalaXmin
         if abscalaXmin (i,j) >= prog;  %fls progu >0
         % abscalaXmin (i,j) < prog;  %fls progu <0
            wezelX = i;
            wezelY = j;
            waga = abscalaXmin (i,j);
            wezelwaga = [wezelwaga; wezelX wezelY waga];
         end;
    end;
end;
wezelwaga1 = [];
for i=1:1:size(calkowitedrzewomin,1)
```

```
    if calkowitedrzewomin(i,3) >= prog;
        wezelwaga1 = [wezelwaga1; calkowitedrzewomin(i,1)
calkowitedrzewomin(i,2) calkowitedrzewomin(i,3)];
    end
end
%% Przywracanie ""-"" znaku wagi jeżeli prawa ręka
    znacznik=unique(znacznik);
    for n=1:1:size(calkowitedrzewomin,1)
        if znacznik==1
            calkowitedrzewomin(n,3)=calkowitedrzewomin(n,3)*(-1);
        end;
    end;
    for n=1:1:size(X,1)
        if znacznik==1
            X(n,1)=X(n,1)*(-1);
        end;
    end;
    for n=1:1:size(wezelwaga,1)
        if znacznik==1
            wezelwaga(n,3)=wezelwaga(n,3)*(-1);
        end;
    end;
%% koniec: Przywracanie ""-"" znaku wagi jeżeli prawa ręka
%% wyznaczanie elementów niesparowanych w całkowitym drzewiemin oraz w
drzewie po odcięciu poniżej progu 0.9, 0.8, ...
Niesparowanecalkowitedrzewomin = [];
if isempty(calkowitedrzewomin) == 0;
    calkowitedrzewominwymienione = [calkowitedrzewomin(:,1)
calkowitedrzewomin(:,2)];
    calkowitedrzewominwymienioneunique = unique
(calkowitedrzewominwymienione);
    Niesparowanecalkowitedrzewomin = setdiff
(sprawedzeniewezlycalkowitedrzewomin,
calkowitedrzewominwymienioneunique);
else
    Niesparowanecalkowitedrzewomin =
sprawedzeniewezlycalkowitedrzewomin;
end
if isempty(Niesparowanecalkowitedrzewomin)==1
    Niesparowanecalkowitedrzewomin=0;
end
NiesparowaneWezelwaga = [];
if isempty(wezelwaga) == 0;
    wezelwagawymienione = [wezelwaga(:,1) wezelwaga(:,2)];
    wezelwagawymienioneunique = unique (wezelwagawymienione);
    NiesparowaneWezelwaga = setdiff (sprawedzeniewezlywezelwaga,
wezelwagawymienioneunique);
else
    NiesparowaneWezelwaga = sprawedzeniewezlywezelwaga;
end

if isempty(NiesparowaneWezelwaga)==1
    NiesparowaneWezelwaga=0;
end
%% koniec: wyznaczanie elementów niesparowanych w całkowitym drzewiemin
oraz w drzewie po odcięciu poniżej progu 0.9, 0.8, ...
%_____________________________________________
%szukanie ilości klastrow (grup) w calkowitym drzewie minimalnym
wierszecalkowitedrzewomin = size (calkowitedrzewomin,1);
kolumnycalkowitedrzewomin = size (calkowitedrzewomin,2)-1;
if isempty(calkowitedrzewomin) == 0;
```

```
    [cc,ccc] = size(calkowitedrzewomin);
    if cc == 1
        Grcalkowitedrzewomin = calkowitedrzewomin (1,:);
    end;
    pierwszywezelwiersz = calkowitedrzewomin (1,1);
    pierwszywezelkolumna = calkowitedrzewomin (1,2);
    pierwszywaga = calkowitedrzewomin (1,3);
    wezelpara = [pierwszywezelwiersz pierwszywezelkolumna pierwszywaga];
    grupa(1).para(1, :) = wezelpara;
    [GrCalkowitedrzewomin, ElwezelklasterCalk] =
Emotiv_poziomyprzeszukiwanie ( wezelpara, calkowitedrzewomin, grupa );
else
    calkowitedrzewomin = 0;
end
%koniec: szukanie ilości klastrow (grup) w calkowitym drzewie minimalnym
%szukanie ilości klastrow (grup) w zminimalizowanej macierzy z progiem
wierszewezelwaga = size (wezelwaga,1);
kolumnywezelwaga = size (wezelwaga,2)-1;
if isempty(wezelwaga) == 0;
    [cc,ccc] = size(wezelwaga);
    if cc == 1
        GrWezelwaga = wezelwaga (1,:);
    end;
    pierwszywezelwiersz = wezelwaga (1,1);
    pierwszywezelkolumna = wezelwaga (1,2);
    pierwszywaga = wezelwaga (1,3);
    wezelpara = [pierwszywezelwiersz pierwszywezelkolumna pierwszywaga];
    grupa(1).para(1, :) = wezelpara;
    [GrWezelwaga, ElwezelklasterWezelwaga] =
Emotiv_poziomyprzeszukiwanie ( wezelpara, wezelwaga, grupa );
else
    GrWezelwaga = 0;
end
%koniec: szukanie ilości klastrow (grup) w zminimalizowanej macierzy z
progiem
%%_____________________________________________________________
%koniec: tworzenie zminimalizowanej macierzy po odcięciu poniżej progu
%% remove of row with repeating weight
variable = [0 0 0];
next_variable = [0 0 0];
repeat_size = size(calkowitedrzewomin,1);
i=1;
while i<repeat_size
    variable = calkowitedrzewomin(i,3);
    next_variable = calkowitedrzewomin(i+1,3);
    if variable == next_variable
        calkowitedrzewomin(i+1,:)=[];
    else
        i=i+1;
    end
    repeat_size = size(calkowitedrzewomin,1);
end
%% ostateczne wyświetlenie wyników ilości grup w zminimalizowanej
macierzy z progiem
% disp (calkowitedrzewomin);
%
% disp (wezelwaga);
%
% disp (NiesparowaneWezelwaga);
%
% disp (Niesparowanecalkowitedrzewomin);
```

```
%
% disp (GrCalkowitedrzewomin);
% disp (ElwezelklasterCalk);
%
% disp (GrWezelwaga);
% disp (ElwezelklasterWezelwaga);
%koniec: ostateczne wyświetlenie wyników ilości grup w zminimalizowanej
macierzy z progiem
disp('koniec drzewomin');
```

Plik „Emotiv_poziomyprzeszukiwanie.m”

```
function [przeszukiwanie, wezelklasterunique] = poziomyprzeszukiwanie (
wezelpara, wezelwaga, klaster )
p = 1;
m = size( wezelwaga, 1 );
n = size( wezelwaga, 2 )-1;
wskaznik = [];
for i=2:1:m
    for j=1:1:n
        v = wezelwaga( i, j );
        wskaznik(1,1)=0;
        ke = size( klaster, 2 );
        ii = 1;
        while( ii <= ke )
            para = klaster(ii).para;
            pc = size( para, 1 );
            for jj=1:1:pc
                pierwszy = klaster(ii).para(jj, 1);
                drugi = klaster(ii).para(jj, 2);
                if v == pierwszy || v == drugi
                    wskaznik(1,1)=1;
                    a = wezelwaga( i, : );
                    pary = klaster(ii).para;
                    wynik = Emotiv_przeszukiwanieklastrow( pary, a );
                    if wynik == 0
                        p = p + 1;
                        klaster(ii).para(p, :) = wezelwaga( i, : );
                    end;
                end
            end
            ii = ii+1;
        end
    end
    if wskaznik(1,1)==0
    % dodanie nowego klastra
    klaster(ke+1).para(1, :) = wezelwaga( i, : );
    ke = size( klaster, 2 );
    end
end
ke = size( klaster, 2 );
for i=1:1:ke
    para = klaster(i).para;
    pc = size( para, 1 );
    pi = 1;
    wezel = [];
    for j=1:1:pc
        wezel(pi) = para(j, 1);
        pi = pi + 1;
        wezel(pi) = para(j, 2);
        pi = pi + 1;
```

```
    end
    wezelklasterunique(i).wezel = unique (wezel);
end
przeszukiwanie = klaster;
```

Plik „Emotiv_przeszukiwanieklastrow.m”

```
function wynik = przeszukiwanieklastrow (pary, a)
pc = size( pary, 1 );
for ii=1:1:pc
    b = pary( ii, : );
    if isequal(b,a) == 1
        wynik = 1;
        return;
    end
end
wynik = 0;
```

Plik „Emotiv_Gower_ab_Coordinates.m”

```
function
calkowitdekodowanie_ab=Emotiv_Gower_ab_Coordinates(calkowitedrzewomin_ab
, dekodowanie_ab_danetest)
% dekodowanie3_ab_danetest = dekodowaniewstepne3ab_danetestowe()
% (podprogram dekodowaniewstepne3ab_danetestowe)
%
% calkowitedrzewomin=drzewomin_danetestowe3ab(dekodowanie3_ab_danetest);
% (podprogram drzewomin_danetestowe3ab)
%
calkowitdekodowanie3_ab=tstatystyka_danetestowe3ab_calkowitedekod(calkow
itedrzewomin, dekodowanie3_ab_danetest)  % (podprogram
tstatystyka_danetestowe3ab_)
Spr = calkowitedrzewomin_ab;
wSpr = size (Spr,1);
w = size(dekodowanie_ab_danetest,1);
calkowitdekodowanie = [];
index = 1;
for mi=1:1:wSpr
    for i=1:1:w
        w1 = dekodowanie_ab_danetest(i,2)== Spr(mi,1) &&
dekodowanie_ab_danetest(i,5)== Spr(mi,2);
        w2 = dekodowanie_ab_danetest(i,2)== Spr(mi,2) &&
dekodowanie_ab_danetest(i,5)== Spr(mi,1);
        if( w1 || w2 )
                calkowitdekodowanie_ab(index,:) =
dekodowanie_ab_danetest(i,:);
                index = index+1;
        end;
    end;
end;
disp ('end calkowitdekodowanie_ab');
```

Plik „Emotiv_Gower_cb_Coordinates.m”

```
function
calkowitdekodowanie_cb=Emotiv_Gower_cb_Coordinates(calkowitedrzewomin_cb
, dekodowanie_cb_danetest)
% dekodowanie3_ab_danetest = dekodowaniewstepne3ab_danetestowe()
% (podprogram dekodowaniewstepne3ab_danetestowe)
%
% calkowitedrzewomin=drzewomin_danetestowe3ab(dekodowanie3_ab_danetest);
% (podprogram drzewomin_danetestowe3ab)
```

```
%
calkowitdekodowanie3_ab=tstatystyka_danetestowe3ab_calkowitedekod(calkow
itedrzewomin, dekodowanie3_ab_danetest)  % (podprogram
tstatystyka_danetestowe3ab_)
Spr = calkowitedrzewomin_cb;
wSpr = size (Spr,1);
w = size(dekodowanie_cb_danetest,1);
calkowitdekodowanie = [];
index = 1;
for mi=1:1:wSpr
    for i=1:1:w
        w1 = dekodowanie_cb_danetest(i,2)== Spr(mi,1) &&
dekodowanie_cb_danetest(i,5)== Spr(mi,2);
        w2 = dekodowanie_cb_danetest(i,2)== Spr(mi,2) &&
dekodowanie_cb_danetest(i,5)== Spr(mi,1);
        if( w1 || w2 )
                calkowitdekodowanie_cb(index,:) =
dekodowanie_cb_danetest(i,:);
                index = index+1;
        end;
    end;
end;
disp ('end calkowitdekodowanie_cb');
```

Plik „EditIP.m”

```
function EditIP(H, E)
    global IPSign
    IPSign = get(H, 'string');
    %disp(['The string in the editbox is: ',a]);
End
```

Plik „EditRdiameter.m”

```
function EditRdiameter(H, E)
    global RdiameterSize
    RdiameterSize = str2num(get(H, 'string'));
end
```

Plik „EditReportFileName.m”

```
function EditReportFileName(H, E)
    global ReportFileNameSign
    ReportFileNameSign = get(H, 'string');
    %disp(['The string in the editbox is: ',a]);
End
```

Do sortowania nazw plików Autorka użyła funkcji ‘natsortfiles’, której autorem jest Stephen Cobeldick. Funkcja umożliwia alfanumeryczne sortowanie tablic plików.

Kod biblioteki dzielonej eegloglib.dll opracowanej przez autrokę, która umożliwia odczyt surowych danych z headsetu Emotie Epoc

```
#include "windows.h"
#include "eegloglib.h"
#include <iostream>
#include <fstream>
#include <sstream>
#include <windows.h>
#include <map>
#include "EmoStateDLL.h"
#include "edk.h"
```

```
#include "edkErrorCode.h"

void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
}

EE_DataChannel_t targetChannelList[] = {
          ED_COUNTER,
          ED_AF3, ED_F7, ED_F3, ED_FC5, ED_T7,
          ED_P7, ED_O1, ED_O2, ED_P8, ED_T8,
          ED_FC6, ED_F4, ED_F8, ED_AF4, ED_GYROX, ED_GYROY, ED_TIMESTAMP,
          ED_FUNC_ID, ED_FUNC_VALUE, ED_MARKER, ED_SYNC_SIGNAL
};

const char header[] = "COUNTER,AF3,F7,F3, FC5, T7, P7, O1, O2,P8"
                      ", T8, FC6, F4,F8, AF4,GYROX, GYROY, TIMESTAMP, "
                      "FUNC_ID, FUNC_VALUE, MARKER, SYNC_SIGNAL,";

int eegLogRead( const char* fileName, int sampleCount, int errorCount )
{
    EmoEngineEventHandle eEvent                   = EE_EmoEngineEventCreate();
    EmoStateHandle eState                         = EE_EmoStateCreate();
    unsigned int userID                           = 0;
    float secs                                    = 1;
    unsigned int datarate                         = 0;
    bool readytocollect                           = false;
    int option                                    = 0;
    int state                                     = 0;

    try
    {
          if (EE_EngineConnect() != EDK_OK)
          {
                throw std::exception("Emotiv Engine start up failed.");
          }

           std::ofstream ofs(fileName, std::ios::trunc);
           ofs << header << std::endl;

           DataHandle hData = EE_DataCreate();
           EE_DataSetBufferSizeInSec(secs);

          for( int ii=0; ii<sampleCount; ++ii )
          {
                state = EE_EngineGetNextEvent(eEvent);
                if (state == EDK_OK)
                {
                      EE_Event_t eventType = EE_EmoEngineEventGetType(eEvent);
                      EE_EmoEngineEventGetUserId(eEvent, &userID);

                      // Log the EmoState if it has been updated
                      if (eventType == EE_UserAdded)
                      {
                            EE_DataAcquisitionEnable(userID,true);
                            readytocollect = true;
                      }
                }

                if (readytocollect)
                {
                      EE_DataUpdateHandle(0, hData);
```

```
                        unsigned int nSamplesTaken=0;
                        EE_DataGetNumberOfSample(hData,&nSamplesTaken);

                        if (nSamplesTaken != 0)
                        {
                                double* data = new double[nSamplesTaken];
                                for (int sampleIdx=0 ; sampleIdx<(int)nSamplesTaken
                                ; ++ sampleIdx)
                                {
                                        for (int i = 0 ;
                                        i<sizeof(targetChannelList)/sizeof(EE_DataChan
                                        nel_t) ; i++)
                                        {
                                                EE_DataGet(hData, targetChannelList[i],
                                                data, nSamplesTaken);
                                                ofs << data[sampleIdx] << ",";
                                        }

                                        ofs << std::endl;
                                }

                                delete [] data;
                        }

                        Sleep(250);
                }
                else
                {
                        Sleep(100);
                }
        }

        ofs.close();
        EE_DataFree(hData);
    }
    catch (const std::exception& e)
    {
         ss.str("");
         ss << e.what() << std::endl;
        ::OutputDebugString(ss.str().c_str());
    }

    EE_EngineDisconnect();
    EE_EmoStateFree(eState);
    EE_EmoEngineEventFree(eEvent);

    return 0;
}
```

8. Spis rysunków

9. Spis tabel

10. Spis załączników

11.Bibliografia

[1] Acharya, J. N., Hani, A., Cheek, J., Thirumala, P. i Tsuchida, T. (2016, August). American Clinical Neurophysiology Society Guideline 2: Guidelines for Standard Electrode Position Nomenclature. *Journal of Clinical Neurophysiology*.

[2] Allwein, E. L., Schapire, R. E. i Singer, Y. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, str. 113.

[3] Aminoff, M. J. i Daroff, R. B. (2014). *Encyclopedia of the Neurological Sciences.* ELSEVIER.

[4] Ang, K., Chin, Z. W., Guan, C. i Zhang, H. (2012). Filter bank common spatial pattern algorithm on BCI competition IV datasets 2a and 2b. *Front Neurosici*.

[5] Backus, G. i Freeman, G. (1968, October). The Resolving Power of Gross Earth Data. *Geophysical Journal International, Volume 16, Issue 2*, str. 169.

[6] Barachant, A. i Congedo, M. (2014). A plug&play P300 BCI using information geometry.

[7] Barachant, A., Bonnet, S. i Congedo, M. J. (2013). Classification of covariance matrieces using a Riemannian-based karnel for BCI applications. *Neurocomputing*, str. 172.

[8] Barachant, A., Bonnet, S., Condego, M. i Jutten, C. (2012). Multi-class brain computer interface classification by riemannian geometry. *IEEE Trans. Biomed. Eng.*, str. 920.

[9] Besserve, M., Martineri, J. i Garnero, L. (2011). Improving quantification of functional networks with EEG inverse problem: Evidence from a decoding point of view. *NeuroImage*, str. 1536.

[10] Birot, G., Spinelli, L., Vaulliemoz, S., Megavand, P., Brunet, D., Seeck, M. i Michel, C. (2014). Head model and electrical source imaging: A study of 38 epileptic patients. *Neuroimage: Clinical*, str. 77.

[11] Bishop, M. C. (2006). *Pattern Recognition and Machie Learning.* Berlin Springer.

[12] Blankertz, B., Lemm, S., Treder, M., Haufe, S. i Muller, K. R. (2010). Single-trial analysis and classification of ERP components - A tutorial. *NeuroImage*, str. 814.

[13] Brodu, N., Lotte, F. i Lecuyer, A. (2011). Comparative study of hand-power extraction techniques for motor imagery classification. *IEEE Symp. on Computational Intelligence, Cognitive Algorithms , Mind, and Brain*, str. 1.

[14] Broniec-Wójcik, A. (2013). Zastosowanie wybranych epizodów elektroencefalograficznych jagko sygnału sterującego w interfejsie człowiek-maszyna. *Akademia Górniczo-Hutnicza*.

[15] Cecotti, H. i Graser, A. (2011). Convolutional neural networks for P300 detectin with application to brain-computer interfaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, str. 433.

[16] Chmielnicki, W. (2012). Efektywne metody selekcji cech i rozwiązywania problemu wieloklasowego w nadzorowanej klasyfikacji danych. *Rozprawa doktorska, Instytut Podstawowych Problemów Techniki Polskiej Akademii Nauk*.

[17] Cho, H., Ahn, M., Kim, K. i Chan Jun, S. (2015). Increasing session-to-session transfer in a brain-computer interface with on-site background noise acquisition. *J. Neural Eng.*

[18] Cichocki, A. (2011). Tensor decompositions: a new concept in brain data analysis. *J. soc. Instrum. Control Eng.*, str. 507.

[19] Cichocki, A., Phan, A. H., Zhao, Q., Lee, N., Oseledets, I., Sugiyama, M. i Mandic, D. (2017). Tensor networks for dimensionality reduction and large-scale optimizations. Part 2 applications and future perspectives. *Found. Trends Mach. Learn.*, str. 431.

[20] Congedo, M. (2013). EEG Source Analysis. *Grenoble: Univ. Grenoble Alpes*.

[21] Congedo, M., Barachant, A. i Bhtia, R. (2017). Riemannian geometry for EEG-based brain-computer interfaces; a primer and a review. *Brain-Comput. Interfaces*, str. 155.

[22] Congedo, M., Lotte, F. i Lecuyer, A. (2006). Classification of movement intention by spatially filtered electromagnetic inverse solutions. *Physics in Medicine and Biology*.

[23] Dąbrowski, M. i Laus-Mączyńska, K. (1978). *Metody wyszukiwania i klasyfikacji informacji (Information Retrieval and Classification. Survey of Methods).* WNT.

[24] del Millan, J. (2004). On the need for on-line learning in brain-computer interfaces. *Proc. IEEE Int. Joint Con. on Neural Networks*, str. 2877.

[25] Devasahayam, S. (2000). *Signals and Systems in Biomedical Engineering - Signal Processing and Physiological Systems Modeling.* KLUWER ACADEMIC/PLENUM PUBLISHERS.

[26] Dietterich, T. G. i Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *J. Artif. Int. Res.*, str. 263.

[27] Dobosz, M. (2001). *Wspomagana komputerowo statystyczna analiza wyników badań.* EXIT.

[28] Duda, R. O., Hart, P. E. i Stork, D. G. (2001). *Pattern Classification.* Wiley.

[29] Durka, P., Kamiński, M., Kuś, R., Malinowska, U., Mikuła, I., Duszyk, A., . . . Żygierewicz, J. (2010). *EEG: skrypt.* brain.fuw.edu.pl/edu / EEG.

[30] Fazel-Rezai, R. (2011). *Recent Advances in Brain Computer Interface Systems.*

[31] Fukushima, K. i Miyake, S. (1982). Neocognitron: a self-organizing neural network model for a mechanism of visual pattern recognition. 267.

[32] Gergondet, P., Druon, S., Kheddar, A., Hintermuller, C., Guger, C. i Slater, M. (2011). Using brain-computer interface to steer a humanoid robot. *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference* (strony 192 - 197). Karon Beach, Phuket : IEEE.

[33] Grosse-Wentrup, M. (2009). Understanding brainconnectivity patterns during motor imagery for brain-computer interfacing. *Advances in Neural Information Processing Systems*.

[34] Grosse-Wentrup, M. (2011). What are the causes of performance variatio in brain-computer interfacing? *Int. J. Bioelectromagn.*, str. 115.

[35] Hasan, B. A. i Gan, J. Q. (2012). Hangman BCI: unsupervised self-paced brain-computer interface for playing games. *Comput. Biol. Med.*, str. 598.

[36] Hastie, T. i Tibshirani, R. (1997). Classification by pairwise coupling. *Advances in Neural Information Processing Systems Conf.*, (strony 507-13). Denver, CO, USA.

[37] Herman, P., Prasad, G., McGinnity, T. i Coyle, D. (2008). Comparative analysis of spectral approaches to festure extraction for EEG-based motor imagery classification. *IEEE Trans. Neural syst. Rehabil. Eng.*, str. 317.

[38] Hsu, W. Y. (2011). EEG-based motor imagery classification using enhanced active segment selection and adaptive classifier. *Comput. Biol. Med.*, str. 633.

[39] Jagodzińska, U. (2013, wrzesień). Applying Algorithms for inverse solutions in classifying EEG signals (Wykorzystywanie algorytmów do rozwiązywania zagadnień odwrotnych do klasyfikacji sygnałów EEG). *Elektronika Konstrukcje Technologie Zastosowania*, str. 144.

[40] Jagodzińska, U. (2013). Rozwiązanie zagadnienia odwrotnego jako klasyfikator sygnałów EEG w inerfejsach mózg-komputer. *Konferencja Urządzenia i Systemy Radioelektroniczne (UiSR), materiały konferencyjne.*

[41] Jagodzińska, U. (2013). The implementation of algorithms for inverse solutions in EEG brain-computer interfaces. *Signal Processing Symposium (SPS).* IEEE.

[42] Jagodzińska, U. (2013, marzec). Towards the Applications of Algorithms for Invese Solutions in EEG Brain-Computer Interfaces. *International Journal of Telecommunications, ISSN 2081-8491*, str. 277.

[43] Jagodzińska, U. i Oskwarek, Ł. (2012). Metoda LORETA jako przykład metody rozwiązywania zagadnienia odwrotnego w interfejsie mózg-komputer. *Urządzenia i Systemy Radioelektroniczne (UiSR), materiały konferencyjne.*

[44] Jagodzińska, U. i Oskwarek, Ł. (2012, grudzień). Metoda LORETA jako przykład rozwiązywania zagadnienia odwrotnego w interfejsie mózg-komputer (Low resolution electromagnetic tomography method as an example of solving the inverse problem in brain computer interface). *Elektronika Konstrukcje Technologie Zastosowania*, str. 89.

[45] Jagodzińska-Szymańska, U. (2014, październik). Implementacja wybranej metody klasterowej do klasyfikacji źródeł sygnałów EEG związanych z wyobrażaniem ruchu (Implementation of the chosen cluster method used for classification of EEG signal sources related to movement imagination). *Elektronika Konstrukcje Technologie Zastosowania*, str. 74.

[46] Jagodzińska-Szymańska, U. (2014). Zastosowanie teorii grafów do klasyfikacji sygnałów EEG (Implementation of graph theory for EEG signals classification). *Urządzenia i Systemy Radioelektroniczne (UiSR), materiały konferencyjne.*

[47] Jagodzińska-Szymańska, U. (2015, wrzesień). Implementacja BCI do klasyfikacji intencji ruchu w oparciu o lokalizację źródeł sygnałów EEG (Implementationof BCI for classifying the intention of movement based on the location of EEG signal sources. *Elektronika Konstrukcje Technologie Zastosowania*, str. 60.

[48] Jagodzińska-Szymańska, U. (2015). Zastosowanie interfejsu mózg-komputer (BCI) do odczytywania intencji ruchu prawą i lewą ręką w oparciu o lokalizację przestrzenną źródeł sygnałów bioelektrycznych. *Urządzenia i Systemy Radioelektroniczne (UiSR), materiały konferencyjne.*

[49] Jagodzińska-Szymańska, U. (2016, grudzień). Algorytm automatycznej klasyfikacji w oparciu o sygnały źródłowe EEG i jego implementacja (Algorithm for the automatic classification based on the signal sources of EEG and its implementation). *Elektronika Konstrukcje Technologie Zastosowania*, str. 31.

[50] Jagodzińska-Szymańska, U. (2016, listopad). Testowanie algorytmu wykorzystującego rozwiązanie zagadnienia odwrotnego do klasyfikacji sygnałów sterujących. *Elektronika Kontstukcje Technologie Zastosowania*, str. 81.

[51] Jayaram, V., Alamgir, M., Altun, Y., Scholkopf, B. i Grosse-Wentrup, M. (2016). Transfer learning in brain-computer interfaces. *IEEE Comput. Intell. Mag.*, str. 20.

[52] Johnston, D. i Miao-Sin, S. (1995). *Foundations of Cellular Neurophysiology.* Massachusetts Institute of Technology.

[53] Kachenoura, A., Albera, L., Senhadji, L. i Comon, P. (2008). ICA: a potential tool for BCI systems. *IEEE Signal Process. Mag.*, str. 57.

[54] Kandel, E. R., Schwartz, J. H. i Jessell, T. M. (2000). *Principles of neural science, Fourth Edition.* McGraw-Hill Companies, Inc.

[55] Kołodziej, M. (2011). *Przetwarzanie, analiza i klasyfikacja sygnału EEG na użytek interfejsu mózg-komputer*. Rozprawa Doktorska Politechnika Warszawska.

[56] Kołodziej, M., Majkowski, A. i Rak, R. (2011). Wykorzystanie t-statystyk do szybkiej selekcji sygnału EEG na użytek interfejsu mózg-komputer. *Przegląd Elektrotechniczny*, str. 187.

[57] Kołodziej, M., Majkowski, A. i Rak, R. J. (2011). Wykorzystanie maszyny wektorów wspierających (SVM) do klasyfikacji sygnłu EEG na uzytek interfejsu mózg-komputer. *Pomiary, Automatyka, Kontrola*, str. 1546.

[58] Krauledat, M., Schroder, M., Blankertz, B. i Muller, K. R. (2007). Reducing calibration time for brain-computer interfaces: a clustering approach. *Advances in Neural Information Processing Sstems vol 19.* Cambridge, MA: MIT Press.

[59] Krusienski, D., Grosse-Wentrup, M., Galan, F., Coyle, D., Miller, K., Forney, E. i Anderson, C. (2011). Critical Issues in state-of-the-art brain-cmputer interface signal processing. *J. Neural eng.*

[60] Krusienski, D., McFarland, D. i Wolpaw, J. (2012). Value o amplitude, phase and coherence features for a sensorimotor rhytm-based brain-computer interface. *Brain Res. Bull.*, str. 130.

[61] Lawson, C. L. i Hanson, R. J. (1995). *Solving Least Squares Problems* . Philadelphia SIAM: Classics in Applied Mathematics, Society for Industrial and Applied Mathematics.

[62] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. i Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, str. 541.

[63] Li, S. (2009). *Markov Random Field Modeling in Image Analysis.* Berlin: Springer.

[64] Lindig-Leon, C. i Bougrain, L. (2015). Comparison of sensorimotor rhytms in EEG signals during simple and combined motor imageries over the contra and ipsilateral

hemispheres. *37th Anual Int. Conf. of the IEEE Engineering in Medicine and Biology Society.* Milan, Italy.

[65] Llera, A., Gomez, V. i Kappen, H. (2012). Adaptive classification on brain-computer interfaces using reinforcement signals. *Neural Comput.*

[66] Lotte, F. (2015). Signal procesing approaches to minimize of suppress calibration time in oscillatory activity-based brain-computer interfaces. *Proc. IEEE*, str. 871.

[67] Lotte, F. (2016). Towards usuable electroencephalograpy-based brain-computer interfaces. *Habilitation Thesis Habilitation a diriger des recherches (HDR) Univ Bordeaux*.

[68] Lotte, F. i Guan, C. (2009). An efficient P300-based brain-computer interface with minimal calibration time. *Assistive Machine Learning for People with Disabilities Symp.*

[69] Lotte, F., Bougrain, L., Cichocki, A., Clerc, M., Congedo, M., Rakotomamonjy, A. i Yger, F. (2018). A review of classification algorithms for EEG-based brain-computer interfaces: a 10 year update. *J. Neural Eng.* .

[70] Lotte, F., Bourgrain, L. i Clerc, M. (2015). *Electroenecephalogrphy (EEG)-based brain-computer interfaces.* Wiley Encyclopedia on Electrical and Electronices Engineering (New York: Wiley).

[71] Lotte, F., Congedo, M., Lecuyer, A., Lamarche, F. i Arnaldi, B. (2007). A review of classification algorithms for EEG-based brain-cmputer interfaces. *J. Neural Eng.*

[72] Madjarov, G., Kocev, D., Gjorgjevikj, D. i Dzeroski, S. (2012). An extensive experimental comparison of methods for muti-label learning. *Pattern Recogn(Best Papers of Iberian Conf. on Pattern Recognition and Image Analysis)*, (strony 3084-104).

[73] Mainardi, L. T., Bianchi, A. M. i Cerutti, S. (2006). Digital Biomedical Signal Acquisition and Processing. W J. D. Bronzino, *Medical Devices and Systems.*

[74] Majchrowski, K. (2017). Podstawy elektroniki i akustyki, Wykład dla elektroradiologii, Pomiar elektryczności w organiźmie człowieka. *Wykład 6*.

[75] Mayaud, L. i et al. (2016). Brain-computer interface for the communcation of acute patients: a feasibility study and a randomized cntrolled trial cmparing performance with healthy participants and a traditional assistive device. *Brain-Comput. Interfaces*, str. 197.

[76] Mc Farland, D., Sarnacki, W. i Wolpaw, J. (2011). Should be parameters of a BCI translation algorithm be continually adapted? *J. Neurosci. Methods*, str. 103.

[77] McFarland, D. i Wolpaw, J. R. (2017). EEG-based brain-computer interfaces. *Current Opinion in Biomedical Engineering*, str. 194.

[78] Niedermeyer, E. i da Silva, F. L. (2005). *Electoencephalography: Basic Principles, Clinical Applications, and Related Fields 5th edn.* Philadelphia, PA: Lipincott Williams&Wilkins.

[79] Noirhomme, Q. (2006). *Localization of Brain Functions: Stimuling Brain Activity and Source Reconstruction for Classification*. Louvain: These presentee en vue de l'obtention du grande de Docteur en Science Appliquees, Universitate Catholique de Louvain.

[80] Noirhomme, Q., Kitney, R. I. i B., M. (2008, May). Single-trial EEG Source Recontruction for Brain-Computer Interface. *IEEE Transactions on Biomedical Engineering*, str. 1592.

[81] Nunez, P. L. i Silberstein, R. B. (2000). *On teh Relationship of Synaptic Activity to Macroscopic Measurements: Does Co-Registration of EEG with fMRI make sense.* Brain Topohraphy, Volume 13, Number 2.

[82] Nunez, P. L. i Srinivasan, R. (2006). *Electric Fields Of The Brain.* OXFORD University Press.

[83] Oskwarek, Ł. (2014). Zastosowanie zagadnienia odwrotnego EEG do oceny aktywości elektrycznej kory mózgowej na użytek interfejsu BCI. *Pomiary Automatyka Kontrola*.

[84] Osowski, S. (2013). *Sieci neuronowe do przetwarzania informacji.* OWPW.

[85] Osowski, S., Cichocki, A. i Siwek, K. (2006). *Matlab w zastosowaniu do obliczeń obwodowych i przetwarzania sygnałów.* Oficyna Wydawnicza Politechniki Warszawskiej.

[86] Pan, S. J. i Yang, Q. (2010). A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, str. 345.

[87] Pascual-Marqui, R. D. (1999). Review of Methods for Solving the EEG Inverse Problem. *International Journal of Bioelectromagnetism, Volume 1, Number 1*, 75.

[88] Pascual-Marqui, R. D. (2007). Discrete, 3D distributed linear imaging methods of electric neuronal activity. Part 1: exact, zero error localization. *The Key Institute for Brain-Mind Research University Hospital of Psychiatry Lenggstr. 31, CH-8032 Zurich, Switzerland, http://www.uzh.ch/keyinst/loreta*.

[89] Phan, A. H. i Cichocki, A. (2010). Tensor decompositions for feature extraction and classification of high dimensional datasets. *Nonlinear Teory Appl.*, str. 37.

[90] Ramoser, H., Muller-Gerking, J. i Pfurtscheller, G. (2000). Optimal spatial filtering of single trial EEG during imagined hand movement. *IEEE Trans. Rehabil. Eng.*, str. 441.

[91] Regińska, T. (2013). *Metody numeryczne w zagadnieniach niestabilnych.* Centrum Studiów Zaawansowanych PW.

[92] Roland, P. E., Larsen, B., Lassen, N. A. i Skinhoj, E. (1980). Supplementary motor area and other cortical areas in organization of voluntary movements in man. *Journal of Neurophysiology Vol. 43*, str. 118.

[93] Sanei, S. i Chambers, J. A. (2007). *EEG Signal Processing.* John Wiley&Sons, Ltd.

[94] Schlogl, A., Vidaurre, C. i Muller, K. R. (2010). *Adaptive methods in BCI research - an introductry tutorial.* Brain Computer Interfaces Berlin: Springer.

[95] Shenoy, P., Krauledat, M., Blankertz, B., Rao, R. i Muller, K. R. (2006). Towards adaptive classification for BCI. *J. Neural Eng.*

[96] Stąpor, K. (2011). *Metody klasyfikacji obiektów w wizji komputerowej.* PWN.

[97] Suchodolski, B., Bromberg, A., Marszałek, L., Stachoń, B., Windholz, A., Wolański, J. i Zabłudowski, T. (1959). *Mała Encyklopedia Powszechna.* Warszawa: Państwowe Wydawnictwo Naukowe.

[98] Tadeusiewicz, R. (2011). Elektryczna aktywność mózgu. *Seminarium Polskiego Towarzystwa Elektrotechniki Teoretycznej i Stosowanej*.

[99] Tsoumakas, G. i Katakis, I. (2007). Multilabel classification: an overview. *Int. J. Data Warehousing Mining*, str. 1.

[100] Vidaurre, C., Sanelli, C., Muller, K. R. i Blankertz, B. (2011). Co-adaptive calibration to improve BCI efficiency. *J. Neural Eng.*

[101] Vidaurre, C., Schlogl, A., Cabeza, R., Scherer, R. i Pfurtscheller, G. (2007). Study of on-line adaptive discriminant analysis for EEG-based brain computer interfaces. *IEEE Trans. Biomed. Eng.*, str. 550.

[102] Wei, Q., Wang, Y. i Gao, X. G. (2007). Amplitude and phase coupling measures for feature extraction in an EEG-based brain-computer interface. *J. Neural Eng.*, str. 120.

[103] Wierzgała, P., Zapała, D., Wójcik, G. M. i Masiak, J. (2018, November). Most Polpular Signal Processing Methods in Motor Imagery BCI: A Review and Meta-Analysis. *Frontiers in Neuroinformatics*.

[104] Wolpaw, J. i Wolpaw, E. (2012). *Brain-Computer Interfaces: Principles and Practice.* Oxford: Oxford University Press.

[105] Wolpaw, J., Birbaumer, N., McFarland, D., Pfurtscheller, G. i Vaughan, T. (2002). Brain-computer interfaces for communication and control. *Clin. Neurophysio.*, str. 767.

[106] Yger, F., Berar, M. i Lotte, F. (2017). Riemannian approaches in brain-computer interfaces: a review. *IEEE Trans. Neural Syst. Rehabil. Eng.*, str. 1753.

[107] Yi, W., Qiu, S., Qi, H., Zhang, L., Wan, B. i Ming, D. (2013). EEG feature comparison and classification of simple and compound limb motor imagery. *J. Neuroeng. Rahabil.*, str. 106.

[108] Zhou, Z., Wan, B., Ming, D. i Qi, H. (2010). A novel technique for phase synchrony measurement from the complex motor imaginary potential of combined body and limb action. *J. Neural Eng*.