



**Adam Banuszewicz**

nr albumu: 33816

kierunek studiów: Teleinformatyka

specjalność: Sieci teleinformatyczne i systemy mobilne

forma studiów: studia stacjonarne

**ALGORYTMY POLECEŃ MENTALNYCH W INTERFEJSACH  
MÓZG-KOMPUTER**

**ALGORITHMS OF MENTAL COMMANDS IN BRAIN-COMPUTER  
INTERFACES**

Praca dyplomowa magisterska

napisana pod kierunkiem:

**dr. inż. Roberta Krupińskiego**

Katedra Przetwarzania Sygnałów i Inżynierii Multimedialnej

Data wydania tematu pracy: 01.11.2018 r.

Data dopuszczenia pracy do egzaminu: .....

Szczecin, 2019



## **OŚWIADCZENIE AUTORA PRACY DYPLOMOWEJ**

Oświadczam, że praca dyplomowa magisterska pn.

„Algorytmy poleceń mentalnych w interfejsach mózg–komputer”

napisana pod kierunkiem:

dr. inż. Roberta Krupińskiego

jest w całości moim samodzielnym autorskim opracowaniem, sporządzonym przy wykorzystaniu wykazanej w pracy literatury przedmiotu i materiałów źródłowych.

Złożona w Dziekanacie Wydziału Elektrycznego treść mojej pracy dyplomowej w formie elektronicznej jest zgodna z treścią w formie pisemnej.

Oświadczam ponadto, że złożona w Dziekanacie praca dyplomowa, ani jej fragmenty nie były wcześniej przedmiotem procedur procesu dyplomowania związanych z uzyskaniem tytułu zawodowego w uczelniach wyższych.

.....  
podpis dyplomanta

Szczecin, dn. ..... r.

## **Streszczenie pracy**

W niniejszej pracy przedstawiono zagadnienia związane z opracowaniem układu wirtualnej klaviatury sterowanej przy użyciu urządzenia do rejestracji aktywności mózgu. Główną motywacją projektu było stworzenie interfejsu mózg–komputer (ang. BCI – brain–computer interface), który umożliwiłby na przykład przywrócenie zdolności sprawnej komunikacji verbalnej osobom z zaburzeniami mowy.

Praca została podzielona na cztery rozdziały. W rozdziale pierwszym zawarto wstęp do tematyki interfejsów mózg–komputer. W rozdziale drugim scharakteryzowano wybrane urządzenia komercyjne, umożliwiające rejestrację aktywności mózgu. W rozdziale trzecim przedstawiono opracowany projekt wirtualnej klaviatury. W ostatnim, czwartym rozdziale, zawarto wyniki badań opracowanego systemu. Pracę zamyka spis bibliograficzny opiewający na 49 pozycji.

## **Słowa kluczowe**

Interfejs mózg–komputer, Wirtualna klaviatura, Elektroencefalografia

## **Abstract**

This thesis presents development process of virtual keyboard controlled with brain activity acquisition device. The main motivation of this project was to create a brain–computer interface, which would allow, for example, to restore verbal communication ability for people with speech disorders.

Thesis was divided into four chapters. The first chapter contains an introduction to the subject of brain–computer interfaces. The second chapter characterizes some of commercially–available devices that allow brain activity recording. The third chapter presents design of virtual keyboard. The last, fourth chapter, contains results of system tests. This thesis provides bibliographic list with total of 49 items.

## **Keywords**

Brain–computer interface, Virtual keyboard, Electroencephalography

# **Spis treści**

<b>Wykaz ważniejszych oznaczeń i skrótów</b>	7
<b>Wprowadzenie</b>	8
<b>1. Wstęp do tematyki interfejsów mózg–komputer</b>	9
1.1. Czym jest oraz jak działa interfejs mózg–komputer	9
1.2. Rodzaje interfejsów mózg–komputer	10
1.2.1. Interfejsy inwazyjne	10
1.2.2. Interfejsy nieinwazyjne	13
1.2.3. Inne rodzaje interfejsów	14
1.3. Przetwarzanie sygnałów	16
1.3.1. Usunięcie szumów	16
1.3.2. Ekstrakcja cech	17
1.3.3. Klasyfikacja sygnałów	18
1.4. Przegląd potencjalnych zastosowań	20
1.4.1. Interfejs komunikacyjny	20
1.4.2. Sterowanie wózkiem inwalidzkim	21
1.4.3. Rehabilitacja	22
1.4.4. Nadzór skupienia	22
1.4.5. Gry komputerowe	24
<b>2. Charakterystyka wybranych urządzeń komercyjnych</b>	25
2.1. Emotiv Insight	25
2.2. Emotiv EPOC+	28
2.3. Muse/Muse 2	29
2.4. MindWave Mobile 2	31
2.5. OpenBCI Ultracortex Mark IV	33
<b>3. Projekt wirtualnej klawiatury</b>	39
3.1. Założenia projektowe	39
3.2. Wybór narzędzi	40
3.2.1. Urządzenia rejestrujące	40
3.2.2. Język programowania	41
3.2.3. Środowisko programistyczne	42
3.2.4. System kontroli wersji	43
3.3. Omówienie aplikacji	43
3.3.1. Struktura kodu źródłowego	43
3.3.2. Algorytm działania	45
3.3.3. Trening detekcji komend mentalnych	47
3.3.4. Komunikacja z urządzeniem	48
3.3.5. Proces decyzyjny	54
3.4. Omówienie interfejsu użytkownika	56
3.4.1. Zakładka <i>Virtual Keyboard</i>	56
3.4.2. Zakładka <i>Messenger</i>	57
3.4.3. Zakładka <i>About</i>	58

3.4.4. Zakładka <i>Connection</i> . . . . .	58
3.4.5. Zakładka <i>Settings</i> . . . . .	60
<b>4. Badania opracowanego systemu</b> . . . . .	61
4.1. Wpływ ilości komend oraz ilości sesji treningowych na poprawność detekcji . . . . .	61
4.2. Wpływ parametrów opracowanej aplikacji na poprawność detekcji . . . . .	66
<b>Zakończenie</b> . . . . .	68
<b>Bibliografia</b> . . . . .	69
<b>Spis tabel</b> . . . . .	73
<b>Spis rysunków</b> . . . . .	74
<b>Spis kodów źródłowych</b> . . . . .	75

# Wykaz ważniejszych oznaczeń i skrótów

<b>API</b>	Application programming interface – Interfejs programistyczny aplikacji
<b>BCI</b>	Brain–computer interface – Interfejs mózg–komputer
<b>BLE</b>	Bluetooth Low Energy
<b>CNS</b>	Central nervous system – Ośrodkowy układ nerwowy
<b>ECoG</b>	Electrocorticography – Elektrokortykografia
<b>EEG</b>	Electroencephalography – Elektroencefalografia
<b>EKG</b>	Electrocardiography – Elektrokardiografia
<b>EMG</b>	Electromyography – Elektromiografia
<b>EOG</b>	Electrooculography – Elektrookulografia
<b>ERD</b>	Event-related desynchronization – Desynchronizacja związana z bodźcem
<b>ERP</b>	Event-related potentials – Potencjały wywołane
<b>ERS</b>	Event-related synchronization – Synchronizacja związana z bodźcem
<b>IDE</b>	Integrated Development Environment – Zintegrowane środowisko programistyczne
<b>JSON</b>	JavaScript Object Notation
<b>MEA</b>	Multielectrode array – Układ mikroelektrod
<b>MEG</b>	Magnetoencephalography – Magnetoencefalografia
<b>MRI</b>	Magnetic resonance imaging – Obrazowanie metodą rezonansu magnetycznego
<b>PNS</b>	Peripheral nervous system – Obwodowy układ nerwowy
<b>SDK</b>	Software development kit – Zestaw narzędzi do tworzenia oprogramowania
<b>SMR</b>	Sensorimotor rhythm – Rytm sensoromotoryczny
<b>SNR</b>	Signal-to-noise ratio – Stosunek sygnału do szumu
<b>SSVEP</b>	Steady-state visual evoked potentials – Wzrokowe potencjały wywołane stanu ustalonego
<b>WPF</b>	Windows Presentation Foundation

# **Wprowadzenie**

W związku z postępującą komputeryzacją otaczającego nas świata zmienia się sposób interakcji człowieka z oprogramowaniem. Konwencjonalne metody prezentacji danych są coraz częściej wspierane przez innowacyjne technologie, w związku z czym konieczne staje się również opracowanie nowych metod ich akwizycji. Szczególnie obiecujące wydają się tutaj stosowane w interfejsach mózg–komputer urządzenia rejestrujące aktywność mózgu, które, zapewniając sposób pozyskiwania poleceń pozostający do tej pory w sferze fantastycznej naukowej, coraz śmielej wkraczają w codzienne życie.

Spośród wielu zastosowań interfejsów mózg–komputer wyróżnić można przywrócenie zdolności komunikacji osobom, które z powodu poważnych schorzeń nie były do tej pory w stanie korzystać z dostępnych interfejsów komunikacyjnych. Medycyna nie jest jednak jedyną dziedziną nauki, która może zyskać na swojej integracji z BCI: sfera ich zastosowań jest znacznie szersza, praktycznie nieograniczona, co przekładać się będzie na wzrost znaczenia tej technologii w procesie interakcji z komputerem.

Badaniom interfejsów mózg–komputer, nad wyraz skomplikowanym ze względu na ich silną interdyscyplinarność, zostało poświęconych w ostatnich latach wiele artykułów. Wśród nich można zauważać pewną właściwość – znaczna część przeprowadzana jest z wykorzystaniem specjalistycznych urządzeń oraz oprogramowania, nierzadko w warunkach laboratoryjnych. Oczywiście jest, iż użytkownik końcowy nie będzie wykorzystywał sprzętu tak wysokiej jakości oraz przebywał w tak odizolowanym środowisku; koniecznością wydaje się zatem weryfikacja możliwości coraz to bardziej osiągalnych, szczególnie w kontekście wzrostu dostępności oraz spadku ceny, urządzeń komercyjnych.

## **Cel pracy**

Celem pracy było wykonanie układu wirtualnej klawiatury sterowanej przy użyciu urządzenia do rejestracji aktywności mózgu. Na realizację celu składała się analiza dostępnych komercyjnie urządzeń rejestrujących, opracowanie algorytmu sterowania wykorzystującego komendy mentalne oraz zbadanie wpływu zakłóceń oraz parametrów algorytmu na jego skuteczność.

## **Zakres pracy**

W ramach pracy przeanalizowano sześć urządzeń do rejestracji aktywności mózgu, spośród których wybrano jedno, na bazie którego wykonano układ wirtualnej klawiatury. Opracowano aplikację w języku C#, która, wykorzystując sterowanie za pomocą rejestrowanych komend mentalnych, pozwala na wprowadzanie tekstu, a następnie jego syntezę do postaci dźwiękowej. (TODO podsumować badania)

## **ROZDZIAŁ 1**

# **Wstęp do tematyki interfejsów mózg–komputer**

W rozdziale pierwszym zawarto wstęp do tematyki interfejsów mózg–komputer. Na treść rozdziału składa się sformułowanie definicji BCI, omówienie podstawowych, niezbędnych do zrozumienia pracy pojęć związanych z układem nerwowym oraz przedstawienie charakteru parametrów rejestrowanych przez urządzenia do akwizycji aktywności mózgu. Omówione zostały również rodzaje interfejsów mózg–komputer, z wyszczególnieniem podziału na inwazyjne, nieinwazyjne oraz *inne*, zawierające podziały rzadziej spotykane w literaturze. W tym rozdziale został poruszony również problem przetwarzania pozyskanych przebiegów aktywności mózgu, w szczególności ich filtracji, ekstrakcji cech oraz klasyfikacji. Rozdział zamyka zestawienie pięciu potencjalnych zastosowań interfejsów mózg–komputer, omówienie sposobu integracji w nie BCI, korzyści z niej wynikających, ich ograniczeń oraz znanych implementacji.

### **1.1. Czym jest oraz jak działa interfejs mózg–komputer**

Interfejs mózg–komputer jest układem, który przekształca aktywność ośrodkowego układu nerwowego (ang. CNS – central nervous system) w polecenia dla zewnętrznego urządzenia wykonawczego. BCI pracuje w zamkniętej pętli, w której można wyróżnić sześć etapów: (1) rejestrację aktywności mózgu, (2) usunięcie szumów, (3) ekstrakcję cech, (4) klasyfikację sygnałów, (5) wydanie polecenia oraz (6) sprzężenie zwrotne [7]. Taka definicja prowadzi do konkluzji, iż interfejs mózg–komputer stwarza dodatkowy efektor dla układu nerwowego [51].

Jak wcześniej wspomniano, BCI przekształca sygnały pozyskane z ośrodkowego układu nerwowego, który razem z obwodowym układem nerwowym (ang. PNS – peripheral nervous system) składa się na układ nerwowy człowieka.

W skład PNS wchodzi układ somatyczny, który stanowi połączenie z narządami zmysłu, mięśniami szkieletowymi i skórą, oraz układ autonomiczny, który unerwia narządy wewnętrzne, przez co zapewnia reakcje niezależne od woli, na przykład bicie serca i oddychanie [46].

CNS składa się z mózgowia oraz rdzenia kręgowego. Rdzeń kręgowy przewodzi impulsy nerwowe pomiędzy mózgiem a obwodowym układem nerwowym. Neurony w rdzeniu kręgowym tworzą również lokalne pętle sprzężenia zwrotnego, które odpowiadają za odruchy bezwarunkowe [46]. Mózgowie odpowiada za kontrolę działań, samoregulację procesów biologicznych oraz funkcje poznawcze, takie jak na przykład uczenie się oraz pamięć.

Wspomniana w sformułowanej definicji aktywność CNS obejmuje zjawiska elektrofizjologiczne, neurochemiczne oraz metaboliczne, które mogą być rejestrowane przez monitorowanie pola elektrycznego, magnetycznego albo innych parametrów za pomocą czujników na skórze głowy, powierzchni mózgu lub wewnątrz mózgu [51]; charakter rejestrowanych parametrów zależy od przyjętej koncepcji realizacji BCI.

Większość współczesnych interfejsów mózg–komputer działa w oparciu o potencjały wywołane [6, 20]. Inną często wykorzystywaną metodą jest detekcja wyobrażenia ruchu [20].

Potencjały wywołane (ang. ERP – event-related potentials) są odpowiedzią na bodźce poznawcze, czuciowe lub ruchowe [6]. Wymagają one zewnętrznej stymulacji, która może być na przykład słuchowa lub wzrokowa. Przykładowa realizacja BCI z wykorzystaniem ERP w postaci potencjałów P300 została omówiona w rozdziale 1.4.1 na stronie 20. Inną metodą jest wykorzystanie wzrokowych potencjałów wywołanych stanu ustalonego (ang. SSVEP – steady-state visual evoked potentials). W tym podejściu każda komenda jest podświetlana ze stałą, unikalną częstotliwością [20]. W aktywności CNS użytkownika skupiającego się na komendzie o danej częstotliwości możliwe będzie zaobserwowanie SSVEP o takiej samej częstotliwości.

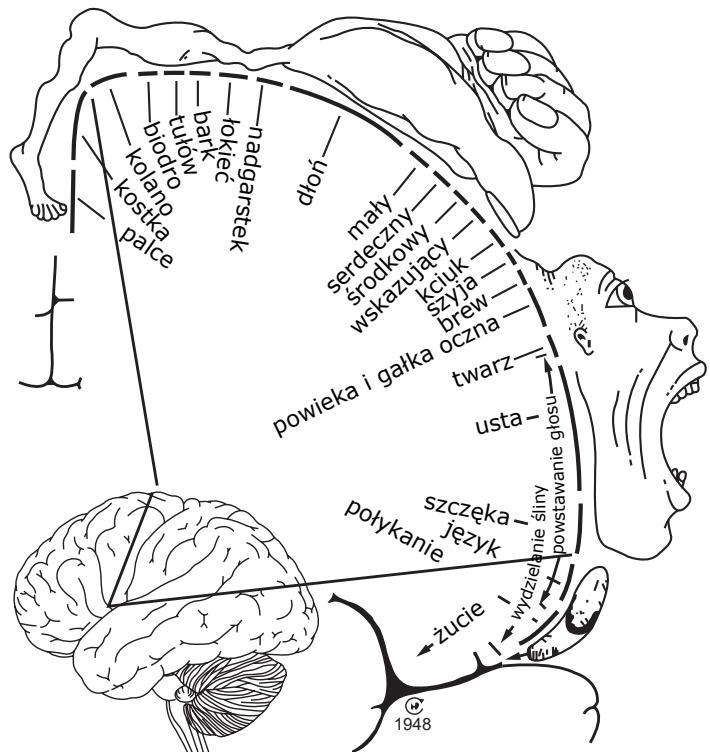
Wyobrażenia ruchu powodują zmiany w rytmach sensoromotorycznych (ang. SMR – sensorimotor rhythms), rejestrowane w obszarach odpowiedzialnych za czucie oraz motorykę [20]. Zmiana jest szczególnie widoczna w paśmie mu ( $8\div12$  Hz) oraz beta ( $13\div30$  Hz) [46]. Jest to tak zwana desynchronizacja/synchronizacja wywołana (ang. ERD/ERS – event-related desynchronization/synchronization), czyli kolejno spadek oraz wzrost mocy w rzeczonych pasmach. Na podstawie *map neurologicznych*, takich jak na przykład przedstawiona na Rysunku 1.1 na następnej stronie, możliwe jest określenie części ciała, której wyobrażenie ruchu spowodowało aktywność neuronów w danym miejscu mózgu.

## 1.2. Rodzaje interfejsów mózg–komputer

### 1.2.1. Interfejsy inwazyjne

Interfejsy pozwalające na pozyskiwanie sygnałów bezpośrednio z komórek mózgowych nazywane są inwazyjnymi. Technika ta stosowana jest najczęściej w badaniach przeprowadzanych z wykorzystaniem zwierząt [46]. U ludzi pomiary inwazyjne pobierane są zazwyczaj w warunkach klinicznych, podczas operacji mózgu lub monitorowania pacjenta bezpośrednio przed lub po operacji.

Umieszczenie inwazyjnego BCI wymaga skomplikowanej operacji neurochirurgicznej, w której część czaszki jest usuwana, elektroda lub implant jest umieszczany w mó-



**Rysunek 1.1.** Homunculus ruchowy Penfielda; mapa kory mózgu odzwierciedlająca ośrodkie ruchowe

Źródło: [51]

zgu, a następnie kość jest przytwierdzana z powrotem [46]. Operacja obarczona jest ryzykiem uszkodzenia tkanki oraz w konsekwencji połączeń pomiędzy neuronami [20].

Interfejsy inwazyjne charakteryzują się najwyższą jakością pozyskanych sygnałów. Spowodowane jest to wyeliminowaniem problemów związanych z niską przewodnością czaszki<sup>1</sup> oraz zredukowaniem artefaktów poprzez zapewnienie dodatkowej warstwy filtrującej sygnały. Inną zaletą jest możliwość ich pozycjonowania bezpośrednio w płatach istotnych z perspektywy konkretnego zastosowania, na przykład w obszarach odpowiedzialnych za funkcje ruchowe lub emocje.

Do inwazyjnej akwizycji sygnałów wykorzystywane są:

**Mikroelektrody** – zostały opracowane do akwizycji sygnałów oraz stymulacji mózgu. Przenikając oponę miękką oraz korę mózgową, umieszczone są w odległości około 150 µm od docelowego neuronu [1]. Są w stanie rejestrować aktywność na poziomie pojedynczych neuronów.

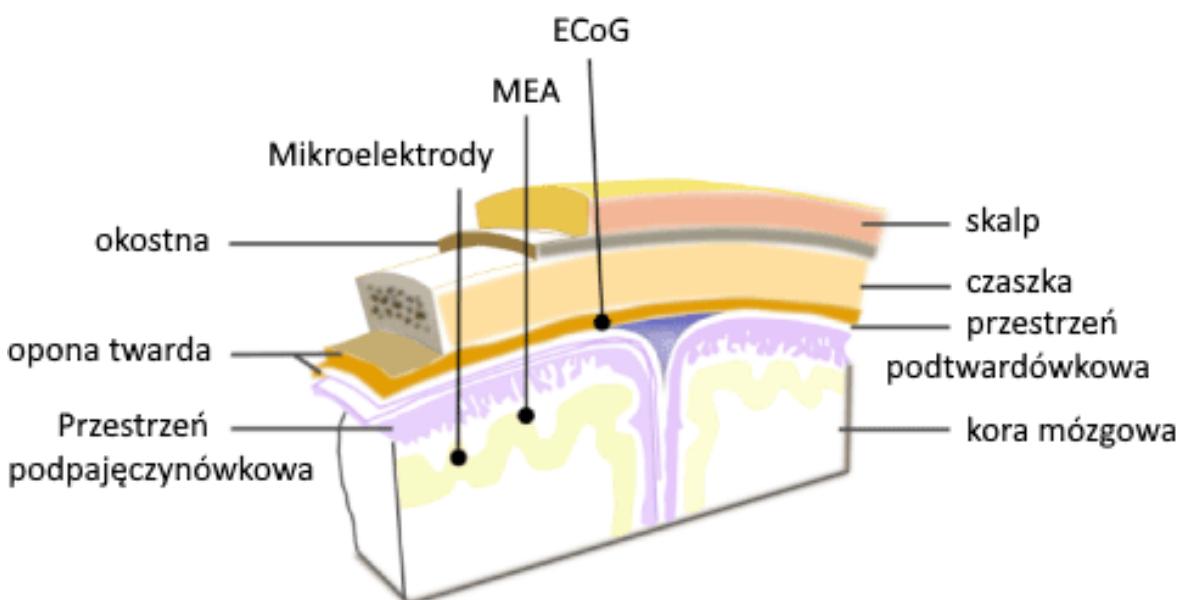
**MEA** – układy mikroelektrod (ang. multielectrode array, również microelectrode array) umieszczanych w korze mózgowej. Są pozycjonowane w pobliżu docelowej populacji neuronów. Odległości między poszczególnymi elektrodami są różne w zależności od realizacji i wynoszą przykładowo 400 µm dla układów Utah oraz 100 µm dla układów Michigan [19]. Ze względu na sposób wykonania wśród głównych typów

<sup>1</sup>Przewodność czaszki jest około 20÷80 krotnie niższa niż mózgu [8]

mikroelektrod można wyróżnić tak zwane *microwire arrays*, *micro-machined arrays* oraz *flexible arrays* [19]. Więcej na temat MEA można znaleźć w literaturze [1, 19, 51].

**ECoG** – elektrokortykografia (ang. *electrocorticography*), czasem nazywana iEEG (ang. *intracranial electroencephalogram* – wewnętrzczaszkowa elektroencefalografia), jest wykonywana przy użyciu elektrod umieszczonych w warstwie podtwardówkowej [1]. Jest wykorzystywana do rejestracji aktywności neuronalnej bez konieczności penetracji tkanki korowej. Jest najmniej inwazyjną metodą z omówionych w tym rozdziale, ponieważ nie ingeruje w strukturę opony miękkiej oraz kory mózgowej. Z racji umieszczenia bliżej kory charakteryzuje się wyższą amplitudą rejestrowanych sygnałów, szerszym zakresem ich częstotliwości oraz lepszymi parametrami topograficznymi niż zewnętrzczaszkowe badanie EEG [20].

Ich typowe umiejscowienie zostało pokazane na Rysunku 1.2.



**Rysunek 1.2.** Umiejscowienie elektrod wykorzystywanych w inwazyjnych BCI w zależności od zastosowanej technologii

Źródło: [1]

Wprowadzenie na rynek inwazyjnych interfejsów mózg–komputer wymaga dalszych badań w zakresie ich wpływu na użytkownika podczas długotrwałego stosowania [20]. Ze względu na ich specjalistyczny charakter oraz skomplikowaną procedurę umieszczenia urządzenia, ich koszt będzie prawdopodobnie znacznie wyższy niż interfejsów nieinwazyjnych.

## 1.2.2. Interfejsy nieinwazyjne

Nieinwazyjne interfejsy mózg–komputer pozwalają na akwizycję sygnałów niewymagającą ingerencji w strukturę czaszki ani nawet skórę głowy. Technologie nieinwazyjne do estymacji sygnałów wykorzystują zmiany w ciśnieniu krwi lub fluktuacje pola elektrycznego oraz magnetycznego, spowodowane aktywnością neuronów w konkretnej części mózgu [46].

Trendem na przestrzeni ostatnich lat jest stosowanie interfejsów wykorzystujących EEG (ang. electroencephalography – elektroencefalografia). Pomiary elektroencefalograficzne są niedrogim, wygodnym oraz uniwersalnym pod względem środowiska wykorzystania sposobem rejestracji aktywności mózgu, czego konsekwencją jest rozwój komercyjnych rozwiązań<sup>2</sup> oraz narastająca liczba publikacji naukowych w tym temacie. Z drugiej strony, interfejsy wykorzystujące MEG (ang. magnetoencephalography – magnetoencefalografia) są kosztowne oraz niepraktyczne w codziennym użytkowaniu, przez co stanowią jedynie narzędzie badawcze dla technologii BCI [51].

Interfejsy nieinwazyjne są bardziej podatne na zakłócenia pochodzące ze środowiska z racji braku ekranowania elektrod czaszką, jak ma to miejsce w pomiarach inwazyjnych.

Jako nieinwazyjne sposoby rejestracji aktywności mózgu stosuje się:

**EEG** – badanie elektroencefalograficzne jest rejestracją aktywności neuronów, które podczas swojej aktywacji wytwarzają potencjał elektryczny. W zależności od miejsca wzmożonej aktywności mózgu, możliwe jest określenie aktualnego stanu psychofizycznego użytkownika urządzenia. Amplitudy sygnałów EEG są zazwyczaj rzędu dziesiątek do setek mikrowoltów [1]. Badanie elektroencefalograficzne wykonuje się poprzez elektrody umieszczone na skalpie. Sensory EEG są małe, lekkie oraz łatwe do założenia [1, 21].

**MEG** – badanie magnetoencefalograficzne rejestruje zmiany pola magnetycznego wywołane przez prąd przepływający przez akson<sup>3</sup>. Do pozyskiwania sygnałów wykorzystywane są bardzo czułe czujniki magnetyczne, na przykład SQUID (ang. superconducting quantum interference device), które są w stanie rejestrować pole magnetyczne rzędu 50÷500 fT<sup>4</sup> [1]. Ta metoda pozyskiwania sygnałów wymaga specjalistycznych osłon przed zakłóceniami elektromagnetycznymi, co dyskwalifikuje możliwość jej użycia poza warunkami laboratoryjnymi.

**MRI** – obrazowanie metodą rezonansu magnetycznego (ang. magnetic resonance imaging) wykorzystuje rezonowanie płynów, głównie krwi, pod wpływem silnego pola magnetycznego [1]. W BCI wykorzystywane są jego dwie odmiany: fMRI oraz dMRI, kolejno funkcjonalne oraz dyfuzyjne MRI. Najczęściej stosowaną techniką rejestracji fMRI jest detekcja lokalnego natlenienia krwi BOLD (ang. blood oxygenation le-

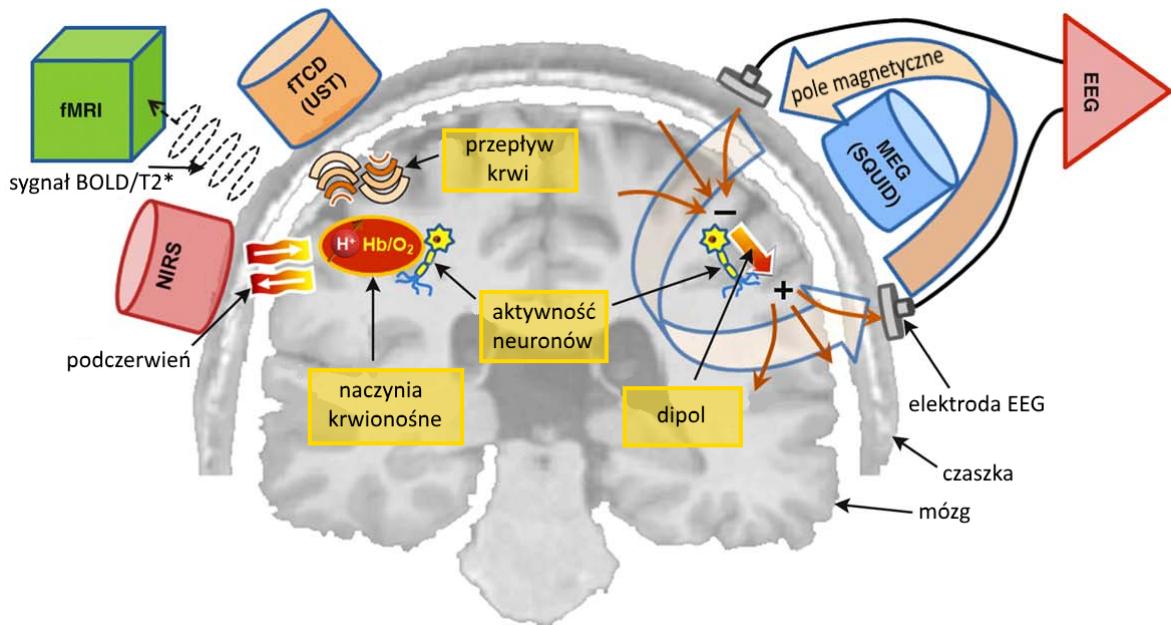
<sup>2</sup>Przykłady dostępnych komercyjnie nieinwazyjnych interfejsów mózg–komputer wykorzystujących sygnały EEG zostały omówione w rozdziale 2 na stronie 25.

<sup>3</sup>Akson – część neuronu odpowiedzialna za przenoszenie informacji.

<sup>4</sup>1 fT =  $1 \times 10^{-15}$  T; T – Tesla

vel dependent) podczas aktywacji neuronów przy użyciu ważonych sekwencji T2-zależnych [30].

Literatura [30] wyszczególnia również metody NIRS oraz fTCD, jednak z racji ich małej popularności zostały pominięte w niniejszej pracy. Schematyczną zasadę działania poszczególnych interfejsów nieinwazyjnych pokazano na Rysunku 1.3.



**Rysunek 1.3.** Sposoby nieinwazyjnej akwizycji aktywności mózgu

Źródło: Opracowanie własne na podstawie [30]

Nieinwazyjne interfejsy mózg–komputer są postrzegane jako najbezpieczniejsze oraz najtańsze BCI [21]. Należy mieć jednak na uwadze, że w tej gamie interfejsów istnieje znaczna rozpiętość cenowa, od około 200 \$ za urządzenie w przypadku tych wykorzystujących EEG, do aż 2–3 milionów \$ w przypadku rejestratorów magnetoencefalograficznych [30].

### 1.2.3. Inne rodzaje interfejsów

Pomimo iż w literaturze dominuje klasyfikacja na interfejsy inwazyjne oraz nieinwazyjne, niektórzy autorzy definiują również inne rodzaje BCI. Wśród nich możemy wyróżnić interfejsy częściowo-inwazyjne, stymulujące oraz dwukierunkowe. Ponieważ nie znajdą one zastosowania w niniejszej pracy oraz są w pewnym sensie pochodnymi omówionych rozdziałów, w celu dopełnienia tematu postanowiono zamieścić jedynie ich zwięzły opis oraz odniesienia do literatury, w której można uzyskać na ich temat więcej informacji.

#### Interfejsy częściowo-inwazyjne

W zależności od definicji inwazyjności, ten interfejs może zawierać się w interfejsach inwazyjnych, omówionych w rozdziale 1.2.1 na stronie 10. Rajesh P.N. Rao definiuje interfejsy

częściowo-inwazyjne jako urządzenia, które nie ingerują w strukturę mózgu [46]. Autor niniejszej pracy nie zgadza się z taką klasyfikacją i postrzega jakąkolwiek ingerencję w czaszkę jako przejaw inwazyjności.

Interfejsy częściowo-inwazyjne wykorzystują na przykład elektrokortykografię, czyli wewnętrzczaszkową elektroencefalografię, lub sygnały pozyskane z nerwów znajdujących się poza mózgiem. W celu instalacji wymagają operacji, jednak nie wiąże się ona z penetracją mózgu. Ten typ interfejsów charakteryzuje się słabszymi parametrami sygnałów niż interfejsy inwazyjne, jednak posiada mniejsze ryzyko uszkodzenia mózgu [21].

Interfejsy częściowo-inwazyjne zostały poruszone w literaturze [20, 21, 46].

### **Interfejsy stymulujące**

Interfejsy stymulujące pozwalają na przesyłanie informacji do mózgu. Możliwość stymulacji mózgu pozwala BCI na bezpośrednie przekazanie danych do mózgu [46], zapewniając tym samym swoisty *dodatkowy zmysł*. Stymulatory mogą być realizowane zarówno w technologii IBS (ang. invasive brain stimulation – inwazyjnej stymulacji mózgu) jak i NIBS (ang. non-invasive brain stimulation – nieinwazyjnej stymulacji mózgu).

Interfejsy stymulujące znajdują aktualnie zastosowanie przede wszystkim w medycynie. Rozwiązania oparte o DBS (ang. deep brain stimulation – głęboką stymulację mózgu) są uznawaną metodą terapii między innymi w przypadku choroby Parkinsona oraz drżenia samoilistnego. Przeprowadzane są również badania dotyczące przywrócenia zmysłu wzroku osobom niewidomym [46].

Interfejsy stymulujące zostały poruszone w literaturze [3, 18, 46].

### **Interfejsy dwukierunkowe**

Dwukierunkowe interfejsy mózg-komputer, zwane również rekurencyjnymi, pozwalają na jednoczesne odczytywanie sygnałów z mózgu oraz przesyłanie do niego informacji. Są realizowane z wykorzystaniem dekodera, który tłumaczy aktywność neuronów na sygnały dla urządzenia wykonawczego, oraz enkodera, który dostarcza informacje z otoczenia bezpośrednio do mózgu, tworząc w ten sposób zamknięty układ sterowania [5].

Dzięki zastosowaniu interfejsów rekurencyjnych mózg nie musi polegać już wyłącznie na ciele w kwestii pozyskiwania sygnałów oraz wykonywania różnych czynności [46]. Jest to szczególnie istotne w odniesieniu do osób niepełnosprawnych, w których przypadku interfejsy te mogą zastąpić uszkodzone struktury organizmu.

Wyzwaniami stawianymi przed tym rodzajem interfejsów są [46]:

- opracowanie sposobu dostarczenia rozmaitych informacji do mózgu przez jego stymulację, rejestrując w tym samym czasie sygnały pochodzące zeń,
- utrzymanie komunikacji dwukierunkowej przez maksymalnie długi czas,
- uwzględnienie oraz wykorzystanie plastyczności mózgu, czyli możliwości do tworzenia nowych połączeń w celu adaptacji oraz w wyniku rozwoju.

Interfejsy dwukierunkowe zostały poruszone w literaturze [5, 46, 50].

## 1.3. Przetwarzanie sygnałów

### 1.3.1. Usunięcie szumów

Sygnały EEG charakteryzują się bardzo niskim współczynnikiem SNR<sup>5</sup> [21]. Z tego powodu przed przystąpieniem do ich analizy należy najpierw usunąć z nich możliwie najwięcej ilość artefaktów. Przykładowe przebiegi zakłóceń zostały pokazane na Rysunku 1.4 na stronie 18.

Pierwszym typem zakłóceń są zakłócenia pochodzące ze środowiska, w którym jest rejestrowany sygnał EEG. Można do nich zaliczyć między innymi zakłócenia od sieci energetycznej oraz elektroniki (komputerów, telefonów, routerów Wi-Fi i tym podobnych). Najprostszym sposobem minimalizacji ich wpływu na sygnał wyjściowy jest eliminacja źródeł zakłóceń – przeprowadzenie akwizycji sygnałów z dala od miast, usunięcie zbędnych urządzeń z otoczenia, zastąpienie, o ile to możliwe, zasilania niezbędnych urządzeń prądem przemiennym na rzecz prądu stałego. Innym sposobem jest wykorzystanie klatki Faradaya.

Drugim typem zakłóceń są tak zwane zakłócenia fizjologiczne. Powstają one na skutek ruchu ciała lub innych fluktuacji potencjałów bioelektrycznych. Ich źródła są niemożliwe do wyeliminowania. Typowymi przykładami takich zakłóceń są sygnały EOG, EMG oraz EKG. W szczególności dwa pierwsze, z racji małej odległości od miejsca akwizycji sygnałów EEG, mają duży wpływ na SNR. Wpływ EOG oraz EMG można zminimalizować poprzez uniknięcie nadmiernego mrugania, ruchu oczu oraz napinaniamięśni.

Typowymi technikami usunięcia artefaktów z sygnału EEG są [46]:

**Progowanie** — jeżeli jakakolwiek charakterystyka sygnału EOG lub EMG przekracza zdefiniowany próg (ang. threshold), próbki sygnałów EEG w tej epoce zostają uznane za skażone i są odrzucane. Ten sposób może być zastosowany również dla akwizycji wyłącznie sygnałów EEG, jednak to podejście wymaga wstępnej kalibracji z użytkownikiem. Wadą tego rozwiązania jest utrata informacji zawartych w odrzuconych próbkach.

**Filtracja** — wycięcie z pobranego sygnału składowych o określonym paśmie częstotliwości przy użyciu filtra pasmowo-zaporowego. W celu przeprowadzenia filtracji należy przekształcić sygnał do dziedziny częstotliwości (na przykład przy użyciu FFT<sup>6</sup>), wyciąć składowe o niepożądanej częstotliwości, a następnie przekształcić sygnał z powrotem do dziedziny czasu. Usunąć w ten sposób można między innymi zakłócenia sieci energetycznej<sup>7</sup> oraz artefakty EOG (około 1÷4 Hz). Filtrację należy stosować, tylko jeżeli podlegające jej składowe mają inną częstotliwość niż sygnały, które chcemy uzyskać.

**Regresja liniowa** — przy założeniu, iż szum sygnału EEG jest addytywny można sformu-

<sup>5</sup>SNR (ang. signal-to-noise ratio) – stosunek sygnału do szumu

<sup>6</sup>FFT (ang. Fast Fourier Transform) – Szybka transformacja Fouriera.

<sup>7</sup>W zależności od kraju zakłócenia te mogą występować w innym paśmie częstotliwości. W Europie jest to 50 Hz, ale na przykład w większości państw Ameryki Północnej częstotliwość sieci wynosi 60 Hz.

łować zależność w postaci [46]:

$$EEG_i(t) = EEG_i^{true}(t) + K \times EOG(t)$$

gdzie  $EEG_i^{true}(t)$  jest czystym sygnałem EEG pozyskanym z elektrody  $i$  w czasie  $t$ ,  $EOG(t)$  jest sygnałem EOG w czasie  $t$ , a  $K$  stałą, która może być estymowana. Posiadając estymację stałej  $K$ , na podstawie prostego przekształcenia można uzyskać estymację czystego sygnału EEG w postaci:

$$EEG_i^{true}(t) = EEG_i(t) - K \times EOG(t)$$

Ta metoda jest trudniejsza do usunięcia artefaktów EMG, ponieważ pochodzą one z wielu źródeł (wielu grup mięśni) i wymagają opracowania bardziej skomplikowanego modelu.

**Analiza składowych głównych** – inaczej PCA (ang. Principal Component Analysis); polega na redukcji współczynników potrzebnych do opisania dużej liczby skorelowanych ze sobą zmiennych, przy jednoczesnym zachowaniu jak największej liczby składowych znajdujących się w sygnale właściwym. Umożliwia zmniejszenie ilości informacji zawartych w sygnale poprzez eliminację pewnych składowych zawierających artefakty [45]. PCA pozwala na usunięcie szumów związanych z EOG [46].

**Analiza składowych niezależnych** – inaczej ICA (ang. Independent Component Analysis); pozwala na estymację nieznanych sygnałów źródłowych oraz ekstrakcję zakłóceń w celu ich późniejszej eliminacji [45]. ICA stosuje się w celu eliminacji zakłóceń EOG oraz EMG [46].

### 1.3.2. Ekstrakcja cech

Celem ekstrakcji cech (ang. feature extraction) jest przekształcenie surowych danych EEG do postaci nadającej się do wykorzystania w procesie klasyfikacji sygnałów [49].

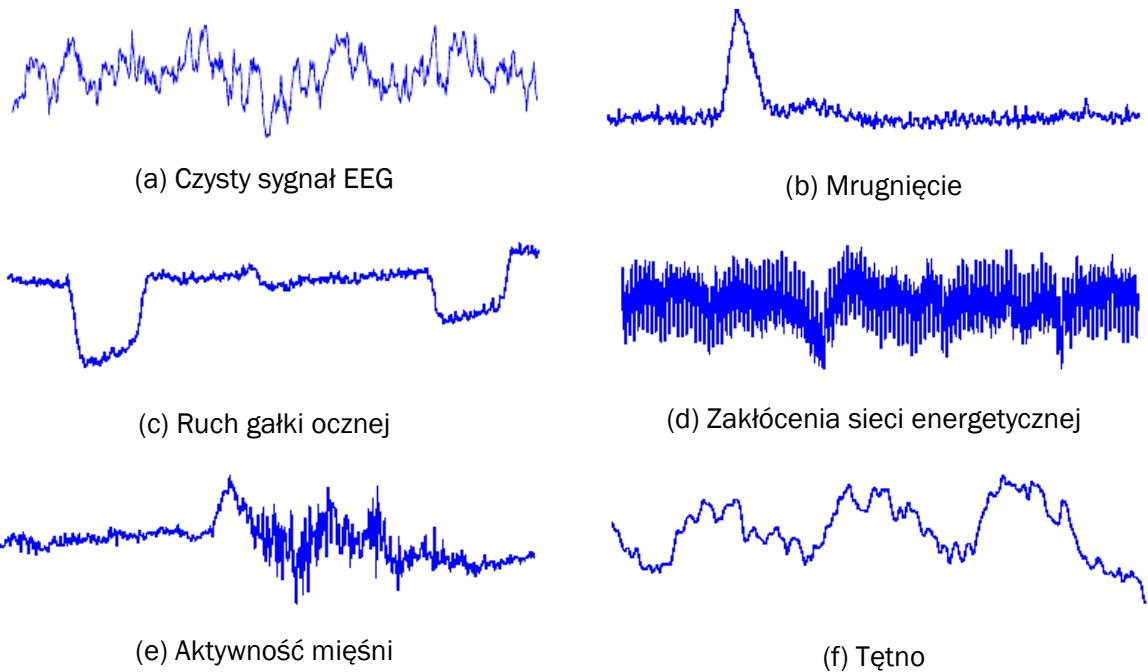
Cecha jest właściwością opisującą sygnał EEG [7]. Cechą podstawową sygnału jest jego bezpośredni pomiar [51], na przykład różnica potencjałów pomiędzy dwoma elektrodami w chwili  $t$ . Podstawowe cechy są zazwyczaj przedstawiane w postaci cech złożonych, które stanowią ich liniowe oraz nieliniowe kombinacje, stosunki lub miary statystyczne. W celu jak najdokładniejszej detekcji intencji użytkownika wiele cech jest pozyskiwanych jednocześnie; są one zazwyczaj grupowane w tak zwane wektory cech [7].

Przykładem wykorzystania ekstrakcji cech może być rozpoznanie wyobrażenia ruchu ręką – cechą używaną do detekcji jest moc pasma  $\mu$  (8÷12 Hz) oraz  $\beta$  (16÷24 Hz) [7].

W interfejsach mózg–komputer najczęściej wykorzystywane są cechy opisujące sygnał [6]:

**Przestrzenne** – służą do przybliżonego wyznaczania źródeł sygnału [6].

**Spektralnie** – opisują moc sygnału w zależności od częstotliwości [7].



**Rysunek 1.4.** Rodzaje artefaktów występujących w sygnałach EEG

Źródło: [10]

**Czasowo** – opisują zmianę sygnału w czasie.

Innymi cechami wykorzystywanymi w BCI są reprezentacje czasowo-częstotliwościowe sygnałów, transformacja Hilberta oraz parametry Hjortha [49].

Problem ekstrakcji cech został poruszony szerzej w literaturze [6, 7, 46, 49, 51].

### 1.3.3. Klasyfikacja sygnałów

Klasyfikatory służą do rozpoznawania wzorców w wektorach cech sygnałów EEG. Ich zadaniem jest przekształcenie aktywności mózgu w sygnały użyteczne dla komputera.

Wśród klasyfikatorów można wyróżnić między innymi:

**Klasyfikator liniowy** – algorytm dyskryminacyjny posługujący się funkcjami liniowymi w celu rozróżnienia poszczególnych klas [28]. Wśród tego rodzaju klasyfikatorów w BCI najczęściej wykorzystywane są LDA<sup>8</sup> oraz SVM<sup>9</sup> [21].

LDA jest liniowym klasyfikatorem binarnym, który przy pomocy hiperpłaszczyzny rozdziela dane reprezentujące różne klasy [28]. Metoda jest stosunkowo prosta i posiada małą złożoność obliczeniową, przez co jest często stosowana w BCI. Jej wadą jest osiąganie słabych rezultatów klasyfikacji w przypadku złożonych, nieliniowych wektorów cech sygnałów EEG [21].

SVM, podobnie jak LDA, używa hiperpłaszczyzny w celu rozdzielenia poszczególnych klas. W przypadku SVM poszukiwana hiperpłaszczyzna powinna rozdzielać z

<sup>8</sup>LDA (ang. linear discriminant analysis) – liniowa analiza dyskryminacyjna

<sup>9</sup>SVM (ang. support vector machine) – maszyna wektorów nośnych

maksymalnym marginesem dane należące do odrębnych klas. Metoda posiada dobre właściwości generalizacji oraz jest niewrażliwa na przetrenowanie [6].

**Sieci neuronowe** – wśród sieci neuronowych wykorzystywanych w BCI najczęściej zastosowanie znajduje MLP<sup>10</sup> [28].

MLP składa się z kilku warstw neuronów: warstwy wejściowej, warstw ukrytych oraz warstwy wyjściowej. Wielowarstwowy perceptron jest wrażliwy na przetrenowanie, w szczególności w przypadku zaszumionych danych, takich jak sygnały EEG [21].

Innymi rodzajami sieci neuronowych wykorzystywanymi w BCI są między innymi LVQ, Fuzzy ARTMAP, RBF oraz BLRNN, ALN oraz PeGNC [28].

**Naiwny klasyfikator bayesowski** – jest klasyfikatorem probabilistycznym opartym na założeniu niezależności cech. Z tego powodu jest również zwany modelem cech niezależnych (*ang. independent feature model*). Rozpatrując klasyfikację jako zadanie przyporządkowania klasy do konkretnego wejścia na podstawie wektorów cech  $F_1, F_2, \dots, F_n$ , decyzję otrzymujemy poprzez wybranie klasy z największym prawdopodobieństwem:

$$P(C = i | F_1, \dots, F_n)$$

Na drodze przekształceń, korzystając z twierdzenia Bayesa, otrzymujemy prawdopodobieństwo w postaci:

$$P(C = i | F_1, \dots, F_n) = P(C = i)P(F_1 | C = i)P(F_2 | C = i) \dots P(F_n | C = i)$$

Mając równanie w tej postaci, wynik klasyfikacji otrzymujemy poprzez obliczenie prawdopodobieństwa dla każdej z klas, a następnie wybranie tej o największym prawdopodobieństwie [46].

**Algorytm najbliższego sąsiada** – klasa wyjściowa jest przydzielana na podstawie najbliższego sąsiada, który jest wyłaniany na przykład przy pomocy odległości euklidesowej między wektorami, danej wzorem [46]:

$$d_{x,y} = \sqrt{\sum_{n=1}^M (x_n - y_n)^2}$$

gdzie  $d_{x,y}$  jest odlegością między wektorami  $x$  oraz  $y$ .

Klasyfikacja algorytmem najbliższego sąsiada NN (*ang. nearest neighbor*) jest podatna na szum [46]. Z tego powodu często jest zastępowana bardziej odporną odmianą w postaci algorytmu  $k$  najbliższych sąsiadów k-NN (*ang. k nearest neighbors*), w którym klasa jest przyporządkowywana na podstawie klasyfikacji  $k$  najbliższych sąsiadów.

---

<sup>10</sup>MLP (*ang. multilayer perceptron*) – wielowarstwowy perceptron

W celu poprawy klasyfikacji można zastosować połączenie kilku różnych algorytmów. Ostateczne przyporządkowanie do konkretnej klasy następuje wtedy na drodze wzmacniania (ang. boosting), głosowania (ang. voting) lub nakładania (ang. stacking).

Wzmacnianie wykorzystuje klasyfikatory połączone w kaskadę. Każdy klasyfikator stara się wyeliminować błąd wprowadzony przez poprzednie. Ten sposób łączenia pozwala na stworzenie wydajnego klasyfikatora z połączenia kilku słabych. Charakteryzuje się odpornością na przetrenowanie [21].

Przy głosowaniu kilka klasyfikatorów samodzielnie dokonuje przydzielenia danych wejściowych do odpowiedniej klasy. Klasa wynikowa zostaje wyłoniona na drodze głosowania, to jest zostaje nią klasa, która została wyłoniona najwięcej razy podczas samodzielnych klasyfikacji.

Nakładanie wykorzystuje dwa poziomy klasyfikatorów. Pierwszy, tak zwany poziom 0, klasyfikuje dane wejściowe do odpowiednich klas. Wyjścia klasyfikatorów poziomu 0 są połączone z wejściami klasyfikatora poziomu wyższego, zwanego poziomem 1, który podejmuje ostateczną decyzję dotyczącą przynależności do danej klasy [28].

Połączone klasyfikatory uzyskują lepsze rezultaty. Ich połączenie obniża wariancję oraz błąd klasyfikacji [28].

## 1.4. Przegląd potencjalnych zastosowań

### 1.4.1. Interfejs komunikacyjny

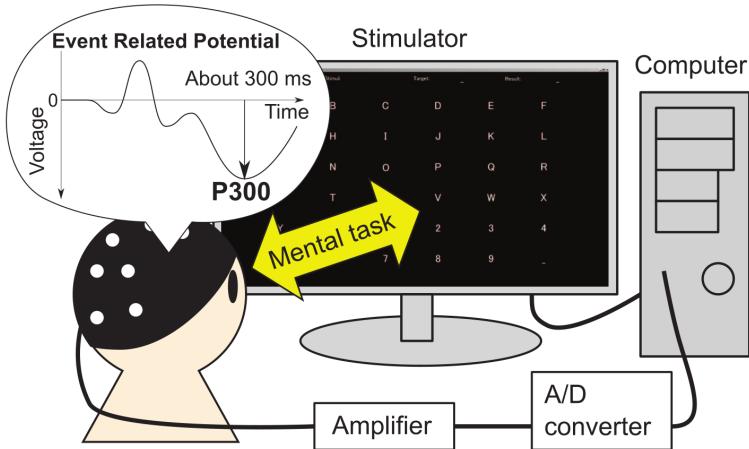
Najprostsze interfejsy komunikacyjne można zrealizować dając użytkownikowi do wyboru dwie opcje: *tak* lub *nie*. Podejście dwuwartościowe pozwala na stosowanie różnorodnych technik akwizycji wyboru, między innymi poprzez analizę mrugnięć, ruchu gałek ocznych, czy sygnałów pochodzących z BCI. Przy tak skonstruowanym interfejsie osoba z niego korzystająca jest w stanie odpowiadać na pytania rozmówcy.

Taki sposób komunikacji może być zmodyfikowany w celu umożliwienia osobie użytkującej interfejs nadawania toku rozmowy. Dzięki zastosowaniu tablicy ze znakami, jej rozmówca może przemieszczać po kolej swój palec po literach alfabetu i notować te, dla których osoba korzystająca z interfejsu zakomunikowała odpowiedź *tak*. Wadą takiego rozwiązania jest fakt, iż rozmówca musi asystować osobie niepełnosprawnej, co może wprowadzać uczestników rozmowy w zakłopotanie. Taki interfejs charakteryzuje się małą stopą błędu, jednak komunikacja przy jego użyciu jest długotrwała i wynosi około jedno słowo na minutę [21].

Innym podejściem jest stworzenie systemu opartego o wirtualny kursor, który pozwoliłby użytkownikowi na samodzielnie wybieranie liter lub całych wyrazów spośród dostępnych opcji. Taki system może zawierać elementy autokorekty, aby zminimalizować występujące błędy.

W literaturze można również spotkać się z systemami badającymi potencjały wywołane. Taką metodą jest P300. W takim interfejsie użytkownik skupia swoją uwagę na literze, którą chce wybrać. Litery są samoczynnie podświetlane w losowej kolejności. 300 milisekund po podświetleniu litery, na której użytkownik systemu jest aktualnie skupiony,

można zaobserwować zmianę amplitudy rejestrowanego sygnału EEG [46]. Przykładową realizację interfejsu opartego o analizę potencjału P300 pokazano na Rysunku 1.5.



**Rysunek 1.5.** Struktura interfejsu opartego o potencjał P300

Źródło: [36]

#### 1.4.2. Sterowanie wózkiem inwalidzkim

Osoby sparaliżowane niekiedy są w stanie sterować wózkiem inwalidzkim, używając do tego celu wydmuchiwanego powietrza, mowy lub, w przypadku częściowego paraliżu, sprawnych części ciała. Integracja z interfejsami mózg–komputer stwarza nowe możliwości dla osób, które z powodu poważniejszych schorzeń nie są w stanie skorzystać z wyżej wymienionych metod sterowania.

Kontrola wózka inwalidzkiego może odbywać się przy wykorzystaniu sterowania nisko- lub wysokopoziomowego.

Sterowanie niskopoziomowe można zrealizować przez translację sygnałów odebranych z urządzenia rejestrującego aktywność mózgu na komendy opisujące ruch wózka (*ruch do przodu, zawróć, przyspiesz*), z zastrzeżeniem, że użytkownik wydaje je w sposób bezpośredni. Ten rodzaj sterowania wymaga zamontowania czujników na wózku, które uniemożliwią wykonanie niedozwolonych manewrów, na przykład uderzenia w inną osobę lub przedmiot. Do zalet rozwiązania można zaliczyć wysoki stopień kontroli osoby użytkującej wózek nad sposobem przemieszczania oraz stosunkowo małą ilość wymaganych komend mentalnych do podstawowego sterowania. Do wad należy konieczność wydawania poleceń ruchu w sposób ciągły, co może prowadzić do znużenia użytkownika, a w konsekwencji samoczynnego *rozstrajania się* układu sterowania.

Sterowanie wysokopoziomowe polega na wydawaniu poleceń dotyczących celu ruchu (*kuchnia, łazienka*). Podejście to wiąże się z koniecznością stosowania wózków o wysokim stopniu autonomiczności. Zaletą tego rozwiązania jest odciążenie użytkownika od konieczności utrzymywania ciągłego skupienia na kierunku oraz prędkości ruchu, wyeliminowanie wpływu zakłóceń odbieranych w trakcie przemieszczania się oraz, po zapewnieniu skutecznych algorytmów i odpowiedniej ilości czujników, bezpieczniejsze po-

ruszanie się w środowisku. Do wad zaliczyć należy większą ilość komend, które należy zdefiniować (co najmniej po jednej dla każdego pomieszczenia), mniejszą precyzję ruchu, spowodowaną odgórny okrešeniem miejsca w pomieszczeniu, do którego użytkownik chce się przemieścić oraz konieczność zdefiniowania i zmapowania każdego środowiska, w którym będzie poruszał się wózek.

Mimo obiecujących wstępnych rezultatów, wykorzystanie interfejsów mózg–komputer do zadania sterowania wózkami inwalidzkimi jest trudne do zrealizowania z powodu braku niezawodnych, przenośnych i łatwych w użyciu urządzeń rejestrujących. Inną przeszkodą jest brak wózków o wystarczającym stopniu autonomiczności, które byłyby zdalne do pracy w codziennym środowisku [46].

### **1.4.3. Rehabilitacja**

Można wyróżnić trzy główne sposoby wsparcia rehabilitacji medycznej przez interfejsy mózg–komputer [9]:

**mózg–akcja** – polega na nauczeniu pacjenta przy pomocy BCI wydawania odpowiednich poleceń, wymaganych do usprawnienia motoryki kończyn,

**mózg–kończyna** – polega na wykorzystaniu BCI do kontroli urządzenia wspierającego poruszanie się, a w konsekwencji odbudowania połączeń potrzebnych do poruszania się bez niego,

**mózg–mózg** – na podstawie informacji odebranych z systemu BCI następuje stymulacja mózgu w celu poprawy jego aktywności, aż do osiągnięcia zadowalających rezultatów.

Należy nadmienić, iż te sposoby nie są wzajemnie wykluczające – ich połączenie pozwala na zwiększenie znaczenia BCI w tej dziedzinie nauki.

Interfejsy mózg–komputer, poprzez stopniową integrację z neurorehabilitacją, mogą pomóc między innymi osobom dotkniętym udarem mózgu, który często powoduje długotrwałą niepełnosprawność ruchową oraz zaburzenia funkcji poznawczych. Wiele badań wykazuje, iż technika nieinwazyjnej stymulacji mózgu jest skuteczna, nawet w przypadku przewlekłego uszkodzenia mózgu [9]. Duża ilość osób rokrocznie dotykanych udarem stanowi badania w tym zakresie niezwykle istotnymi dla społeczeństwa.

Stosowanie BCI do wspomagania leczenia funkcji poznawczych wymaga skrupulatnej identyfikacji sygnałów mózgowych, które są możliwe do uzyskania. Może to być trudne, ponieważ nie wszystkie stany mózgu są tak dobrze scharakteryzowane, jak te związane z funkcjami motorycznymi [31].

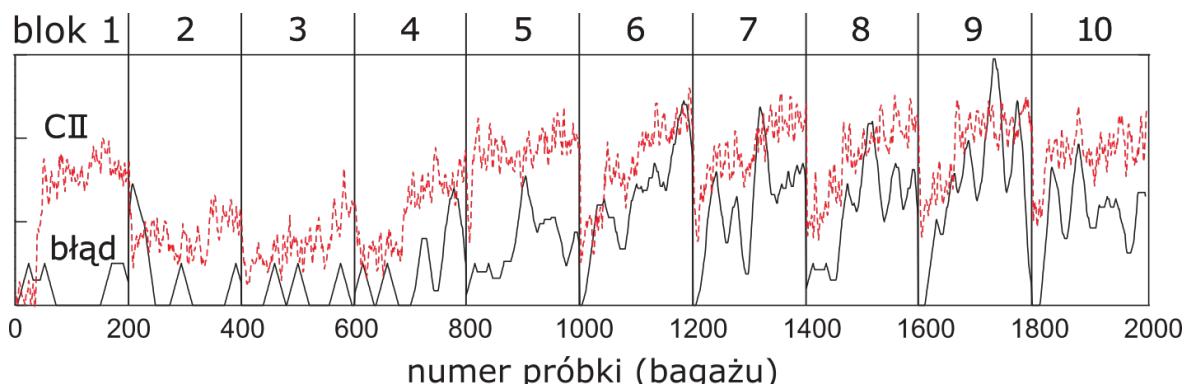
### **1.4.4. Nadzór skupienia**

Wykorzystanie interfejsów mózg–komputer do nadzorowania skupienia użytkownika jest szczególnie istotne w asystowaniu pracowników wykonujących monotonne prace, takie jak prowadzenie samochodu czy monitorowanie systemu zabezpieczeń. Popełnienie błędu w przypadku takich profesji powoduje nie tylko ryzyko dla samego pracownika, ale

również dla innych osób znajdujących się w jego otoczeniu. Wiele wypadków jest spowodowanych zmęczeniem, nieuwagą lub nawet zaśnięciem podczas wykonywania pracy. Choć znużenie lub senność może być wykryta przez analizowanie obszaru okolic oczu, taka detekcja może mieć miejsce za późno, aby było możliwe zapobiegnięcie wypadkowi. Integracja BCI z urządzeniami do nadzoru niesie możliwość wykrywania takich sytuacji, zanim doprowadzą do katastrofalnych w skutkach wydarzeń.

W związku z postępującą automatyzacją pojazdów, kierowca jest odciążany od wykonywania żmudnych zadań, takich jak utrzymywanie stałej prędkości czy przełączanie między światłami mijania oraz drogowymi. Wraz z dalszym progresem coraz więcej czynności wykonywanych jest przez sam samochód, podczas kiedy rola osoby znajdującej się w fotelu kierowcy sprowadza się do nadzoru i reagowania w niebezpiecznych sytuacjach. Może to prowadzić, w szczególności na trasach o znacznej długości, do sytuacji nadmiernego zaufania do systemu sterowania, w której to kierowca przestanie zwracać uwagę na działania pojazdu. Interfejs mózg–komputer mógłby w tym momencie sygnalizować spadek skupienia kierowcy lub wymuszać bezpieczny postój pojazdu do momentu, w którym kierujący pojazdem będzie w stanie kontynuować podróż.

Badania przeprowadzone przez naukowców z Berlin BCI wykazały, iż istnieje zależność pomiędzy spadem koncentracji, a wzrostem mocy w paśmie alpha<sup>11</sup> [4]. Badanie polegało na sklasyfikowaniu 2000 bagażów jako bezpiecznych lub niebezpiecznych na podstawie ich prześwietleń. Wykonano je w 10 turach po 200 bagażów. Zarejestrowaną zależność pomiędzy parametrem CII (ang. concentration insufficiency index), a stopą błędu przedstawiono na Rysunku 1.6. Na podstawie badania można wywnioskować, iż spadek koncentracji (wzrost parametru CII) prowadził do wzmożenia ilości popełnianych błędów. Uczestnicy rzeczonego badania na okres następujący bezpośrednio po przerwie wykazywali znacznie niższy stopień błędu. Tę zależność można wykorzystać do określenia momentów, w których pracownik powinien udać się na przerwę lub, w przypadku dłużej nikłej koncentracji, zakończyć swoją zmianę.



**Rysunek 1.6.** Zależność pomiędzy stopniem skupienia CII, a stopą błędu. Pionowe linie po zakończeniu dwustuprobkowych bloków oznaczają moment przerwy.

Źródło: [4]

<sup>11</sup>Pasmo alpha - pasmo fal elektromagnetycznych o częstotliwości 8÷12 Hz i amplitudzie w zakresie 20÷80 µV [31, str. 17].

#### **1.4.5. Gry komputerowe**

Wprowadzenie dedykowanych rozwiązań dla przemysłu gier komputerowych wydaje się być kolejnym krokiem po rozwoju aplikacji korzystających z technologii VR<sup>12</sup> oraz AR<sup>13</sup>. To, co sprawia, że gracze są odpowiednim odbiorcą wczesnych systemów BCI jest fakt, iż często są oni zainteresowani nowymi technologiami, są skłonni do jej wdrażania w celu uzyskania przewagi nad przeciwnikami oraz przyzwyczajeni do konieczności treningu, który pozwala na progres w grach [35]. W związku z dynamicznym rozwojem tej branży, opracowanie rozwiązań wykorzystujących BCI może być bardzo opłacalne w przyszłości.

Wykorzystanie BCI w grach komputerowych pozwala na integrację gry z wrażeniami użytkownika. Dostarczenie informacji na temat skupienia, zainteresowania, frustracji czy znudzenia umożliwia grze dostosowanie się do potrzeb gracza. W przypadku narastania frustracji gra może samoczynnie obniżyć poziom trudności, a w przypadku znudzenia – podwyższyć. Wykorzystując inne odczucia gra może również uprościć interfejs użytkownika, czy też wyświetlić lub ukryć podpowiedzi dotyczące aktualnie wykonywanego zadania (ang. quest). Dynamiczne dostosowywanie gry do aktualnego stanu gracza pozwala jej na zapewnienie użytkownikowi zwiększonego komfortu podczas rozgrywki.

Innym zastosowaniem interfejsów mózg–komputer jest wydawanie poleceń za pomocą komend mentalnych. Polecienniem może być na przykład wyobrażenie sobie ruchu postaci, co prowadziłoby do faktycznego przemieszczania się kontrolowanego bohatera, czy wybieranie odpowiedzi w dialogach występujących w grze. Wiąże się to z całkowitą zależnością gry od sygnałów otrzymywanych z urządzenia rejestrującego i wnosi konieczność znacznej modyfikacji istniejących silników gier.

Zintegrowanie BCI z grami komputerowymi wiąże się nie tylko ze znacznymi inwestycjami ze strony producentów gier, ale również samych graczy, którzy będą musieli zakupić urządzenie rejestrujące aktywność mózgu. Należy zwrócić uwagę na fakt, że rozgrywki często prowadzone są przez długi okres, a więc wymogiem stawianym przed systemami rejestrującymi jest ich wygoda. Nie mogą one ograniczać ruchów użytkownika ani jego pola widzenia. Wadą rozwiązań jest fakt, iż gracze często generują napięcia mięśniowe, co będzie prowadziło do powstawania artefaktów w przypadku urządzeń rejestrujących sygnały EEG. Rozwiązaniem tego problemu może być ulepszenie algorytmów filtracji sygnałów lub stosowanie systemów hybrydowych EEG/EMG, a więc rejestrujących oraz wykorzystujących aktywność zarówno mózgu, jak i mięśni [46].

---

<sup>12</sup>VR (ang. virtual reality) – rzeczywistość wirtualna

<sup>13</sup>AR (ang. augmented reality) – rzeczywistość rozszerzona

## **ROZDZIAŁ 2**

# **Charakterystyka wybranych urządzeń komercyjnych**

W rozdziale drugim zostały omówione wybrane urządzenia pozwalające na rejestrację aktywności mózgu. Zebrane urządzenia stanowią nieinwazyjne jednostki akwizycyjne, które są dostępne w sprzedaży detalicznej w cenie nieprzekraczającej 1000 \$. Przenalizowano sześć urządzeń czterech różnych producentów, w szczególności ich parametry, rodzaj oraz umiejscowienie czujników oraz dostępne narzędzia programistyczne. W przypadku każdego urządzenia wyszczególniono również jego cenę oraz, jeśli dotyczy, model licencjonowania.

### **2.1. Emotiv Insight**

Insight (patrz Rysunek 2.1 na następnej stronie) jest produktem wprowadzonym na rynek w roku 2015 przez firmę Emotiv przy wsparciu crowdfundingu na portalu kickstarter. Jest produktem do użytku codziennego, głównie za sprawą minimalistycznego designu oraz braku konieczności stosowania żelów przewodzących, przeznaczonym do mniej precyzyjnych zastosowań.

Jest wyposażony w pięć czujników właściwych oraz dwa referencyjne. Lokalizacja czujników została przedstawiona na Rysunku 2.2 na następnej stronie. Czas ubrania oraz ustawienia urządzenia oscyluje w granicach 1–2 minut. Parametry urządzenia zostały zestawione w Tabeli 2.1 na stronie 27.

Koszt produktu na dzień 21 kwietnia 2019 roku wynosi 299 \$.

Firma Emotiv dostarcza do swoich rozwiązań API<sup>1</sup> o nazwie Cortex. Stanowi on podstawkę do budowania aplikacji wykorzystujących pobrane z hełmów strumienie danych dzięki wykorzystaniu JSON oraz WebSocket [12]. Cortex ułatwia tworzenie gier, aplikacji oraz rejestrowania danych do późniejszego ich wykorzystania do badań.

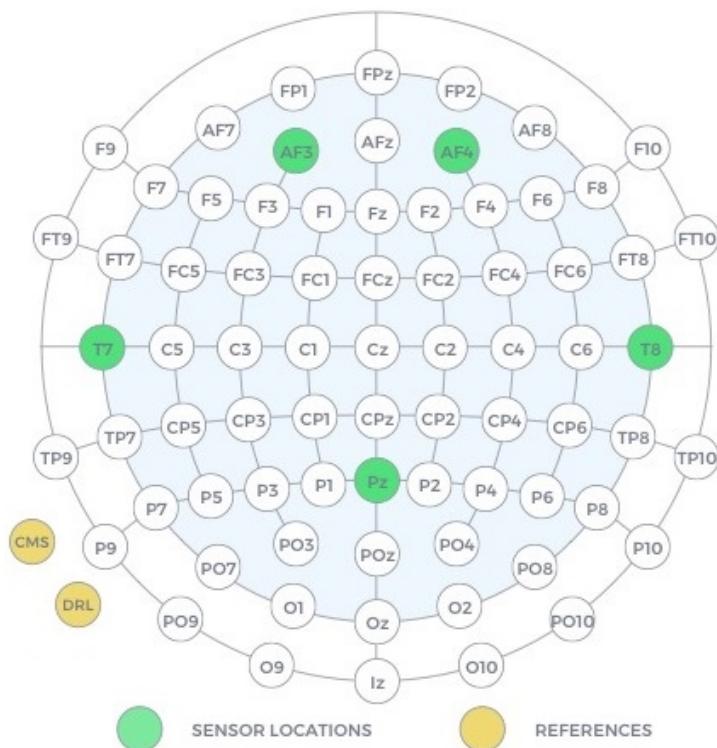
---

<sup>1</sup>API (ang. application programming interface) – Interfejs programistyczny aplikacji. Zawiera zestaw reguł i ich opisów, które definiują sposób komunikacji między programami komputerowymi.



**Rysunek 2.1.** Hełm Emotiv Insight

Źródło: [14]



**Rysunek 2.2.** Rozmieszczenie sensorów w hełmie Emotiv Insight

Źródło: [14]

Cortex jest wrapperem SDK<sup>2</sup> firmy EMOTIV. Zapewnia on, w zależności od rodzaju zakupionej licencji, dostęp do różnych strumieni danych z hełmów. Jest kompatybilny z systemami Mac OS oraz Windows. Umożliwia programowanie w językach Java, C#, C++, Python, Ruby, JavaScript (Node.js) oraz PHP.

<sup>2</sup>SDK (ang. software development kit) – Zestaw narzędzi dla programistów niezbędny w tworzeniu aplikacji korzystających z danej biblioteki.

**Tabela 2.1.** Parametry Emotiv Insight

Źródło: Opracowanie własne na podstawie [17]

Ilość kanałów	5 (+2 referencyjne)
Umiejscowienie elektrod	AF3, AF4, T7, T8, Pz
Czujniki referencyjne	DMS/DRL
Rodzaj czujników	Półsuchy polimer
Rozdzielcość	14 bit na kanał
Rozdzielcość LSB	0,51 µV @ 14 bit
Detekcja ruchu	9-osiowy czujnik (3x żyroskop, 3x akcelerometr, 3x magnetometr)
Łączność	Bezprzewodowa 2,4GHz/Bluetooth 4.0
Zasilanie	Li-Po 480 mAh, do 8 godzin pracy

Licencja Cortex jest dostępna w trzech planach omówionych poniżej.

### **Darmowa**

- Mental Commands API,
- Performance Metrics API (do 0,1 Hz),
- Frequency Bands API,
- Facial Expressions API,
- Motion data API,
- nielimitowana ilość sesji na 3 urządzeniach.

### **Niekomercyjna pro – 55–99 \$/miesiąc**

- Wszystkie API z licencji darmowej,
- Raw EEG API,
- oprogramowanie EmotivPRO,
- nielimitowana ilość sesji na 3 urządzeniach.

### **Komercyjna**

- Performance Metrics API o wysokiej rozdzielcości,
- konfigurowanie API pod swoje potrzeby,
- tworzenie komercyjnych rozwiązań.

Oprogramowanie EmotivPRO [15], dostępne w licencjach niekomercyjnej pro oraz komercyjnej, stanowi wsparcie dla badań wykorzystujących EEG. Pozwala ono na akwizycję oraz prezentację strumieni danych w czasie zbliżonym do rzeczywistego, zapisywanie sesji w chmurze oraz szybką analizę wbudowanym algorytmem FFT, bez konieczności eksportu danych.

## 2.2. Emotiv EPOC+

EPOC+, pokazany na Rysunku 2.3, został wprowadzony na rynek w 2013 roku przez firmę Emotiv. Został zaprojektowany do badań wykorzystujących EEG oraz zaawansowanych zastosowań BCI [13].

Jest wyposażony w 14 kanałów właściwych oraz 2 referencyjne (dokładna lokalizacja sensorów została przedstawiona na Rysunku 2.4 na następnej stronie). W odróżnieniu od Emotiv Insight, omówionego w rozdziale 2.1 na stronie 25, wymaga stosowania *mokrych* elektrod, pokrytych nasączonym solą fizjologiczną filcem. Ze względu na większą ilość czujników niż w Emotiv Insight, czas ubrania oraz przygotowania urządzenia do pracy wynosi około 3–5 minut. Parametry hełmu zostały przedstawione w Tabeli 2.2.

Koszt produktu na dzień 21 kwietnia 2019 roku wynosi 799 \$.



**Rysunek 2.3.** Hełm Emotiv EPOC+

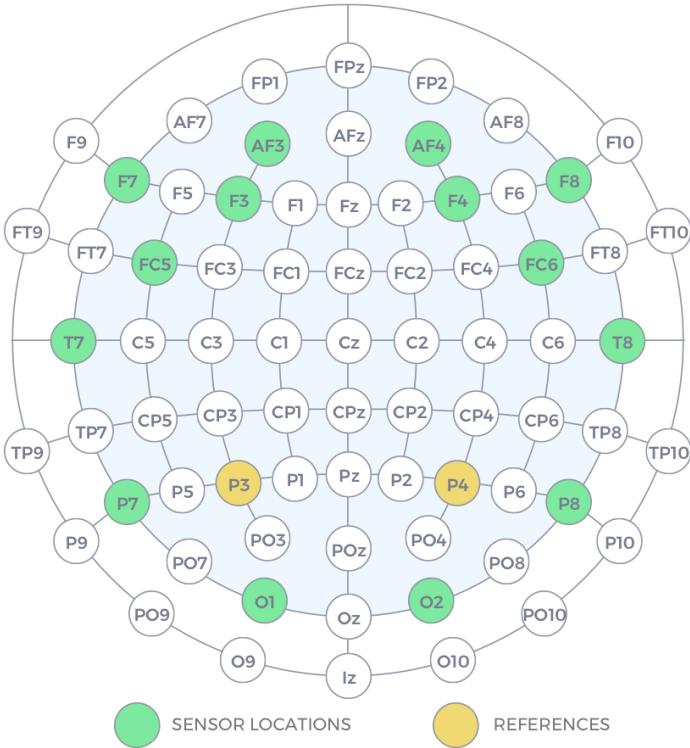
Źródło: [13]

**Tabela 2.2.** Parametry Emotiv EPOC+

Źródło: Opracowanie własne na podstawie [17]

Ilość kanałów	14 (+2 referencyjne)
Umiejscowienie elektrod	AF3, AF4, F3, F4, FC5, FC6, F7, F8, T7, T8, P7, P8, O1, O2
Czujniki referencyjne	DMS/DRL
Rodzaj czujników	Nasączane solą fizjologiczną
Rozdzielcość	14/16 bit na kanał
Rozdzielcość LSB	0,51 µV @ 14 bit/0,13 µV @ 16 bit
Detekcja ruchu	9-osiowy czujnik (3x żyroskop, 3x akcelerometr, 3x magnetometr)
Łączność	Bezprzewodowa 2,4GHz/Bluetooth 4.0
Zasilanie	Li-Po 680 mAh, do 12 godzin pracy

Od strony programistycznej urządzenie wykorzystuje to samo API oraz SDK co Emotiv Insight; zostały one omówione w rozdziale 2.1 na stronie 25.



**Rysunek 2.4.** Rozmieszczenie sensorów w hełmie Emotiv EPOC+

Źródło: [13]

## 2.3. Muse/Muse 2

Muse/Muse 2 są urządzeniami wspomagającymi medytację, które pozwalają na rejestrację w czasie rzeczywistym aktywności mózgu, tętna, oddechu oraz ruchu ciała<sup>3</sup> [23]. Przekształcają one zmierzonyą aktywność mózgu w predefiniowane dźwięki, takie jak szum wody czy deszczu; w zależności od poziomu skupienia dźwięk będzie spokojny lub gwałtowny, co pozwala osobom uczącym się medytować na efektywniejszą naukę wyciszenia umysłu.

Oba urządzenia są z wyglądu bardzo do siebie podobne. Nowsze, Muse 2 (pokazane na Rysunku 2.5 na następnej stronie), w odniesieniu do poprzedniej wersji, zostało *odchudzone*, przez co nabralo bardziej eleganckiego wyglądu oraz zyskało niższy profil z dodatkowymi czujnikami [26]. Dodano również miękkie w dotyku wykończenie.

Obie opaski są wyposażone w 7 czujników, w tym 3 referencyjne (patrz Rysunek 2.6 na następnej stronie). Zestawienie parametrów oferowanych przez obie opaski znajduje się w Tabeli 2.3 na stronie 31.

Koszt Muse wynosi 219 €; Muse 2 – 269 €.

Muse posiada oferty skierowane do następujących grup:

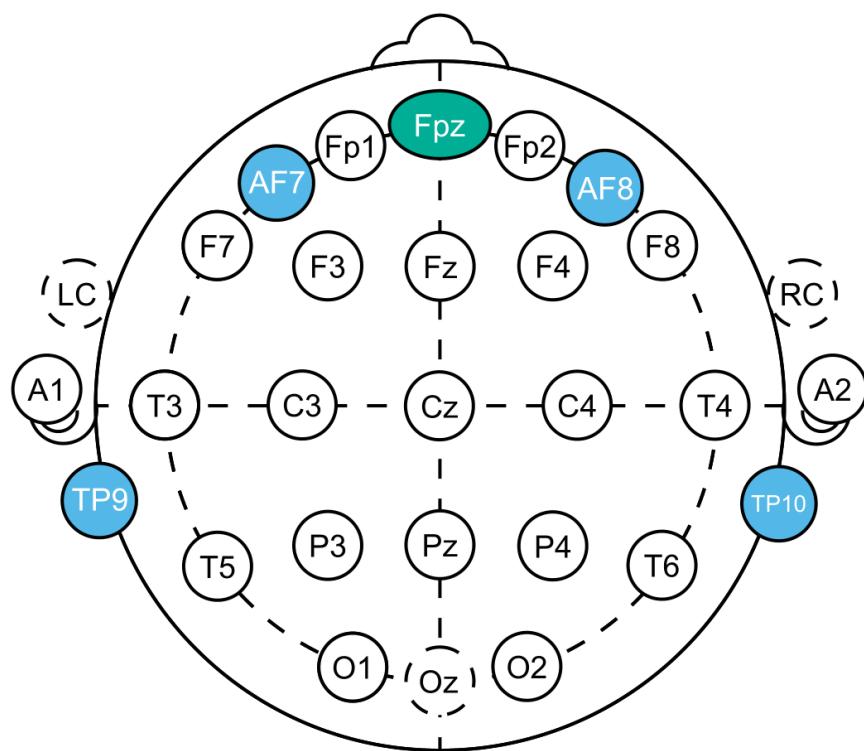
**Deweloperów** – na dzień 23 kwietnia 2019 roku Muse nie wspiera aktywnie swojego SDK. Ostatnią dostępną wersją jest v6.0.3, wydana w marcu 2018 roku. Opaska Muse 2, z racji późniejszej daty premiery, **nie jest wspierana przez SDK**. Na stronie

<sup>3</sup>Rejestracja poszczególnych parametrów w zależności od wersji opaski.



**Rysunek 2.5.** Opaska Muse 2

Źródło: [23]



**Rysunek 2.6.** Rozmieszczenie sensorów w opasce Muse

Źródło: [47]

dla deweloperów [24] znajduje się odnośnik do innych narzędzi, takich jak Muse Direct czy projektów open source, np. MuseLSL, EEG Notebooks.

**Profesjonalistów** – w ramach subskrypcji Muse Connect profesjonalisci otrzymują program wspomagający rozwój ich biznesu poprzez uczenie ich klientów technik medytacji [25]. W ten sposób otrzymują dostęp do różnych wskazówek m.in. webina-

**Tabela 2.3.** Parametry Muse oraz Muse 2

Źródło: Opracowanie własne na podstawie [22] oraz [47]

Parametr	Muse	Muse 2
Ilość kanałów	4 (+ 3 referencyjne)	4 (+3 referencyjne)
Umiejscowienie elektrod	TP9, AF7, AF8, TP10 <sup>a</sup>	TP9, AF7, AF8, TP10 <sup>a</sup>
Czujniki referencyjne	CMS/DRL	CMS/DRL
Rodzaj czujników	Suche srebrne/silikonowe	Suche srebrne/silikonowe
Rozdzielcość	12 bit na próbkę	12 bit na próbkę
Rejestrowane parametry	EEG	EEG, tętno, ruch ciała, oddech
Kompatybilność	iOS, Android	iOS, Android
Łączność	Bezprzewodowa Bluetooth 4.0	Bezprzewodowa Bluetooth 5.0
Zasilanie	Li-Ion, do 5 godzin pracy	Li-Ion, do 5 godzin pracy

<sup>a</sup> Dokładna lokalizacja zależy od wielkości głowy użytkownika; zamieszczono lokalizację zgodną z [47]. W literaturze można spotkać również T9, FP1, FP2, T10 [2].

riów<sup>4</sup>, studiów przypadków oraz informacji, które pomogą wprowadzić Muse do ich biznesu. Muse Connect wspomaga prowadzenie podopiecznych: ustalanie dla nich celów do realizacji oraz śledzenie ich progresu (również w czasie rzeczywistym).

Aplikacja oferuje dwa rodzaje subskrypcji:

1. miesięczną w cenie 39 \$/miesiąc,
2. roczną w cenie 33 \$/miesiąc; w tej opcji dodatkowo otrzymujemy za darmo urządzenie Muse.

**Naukowców** – w ramach narzędzi dla naukowców Muse oferuje dostęp do MusePlayer oraz MuseLab. MusePlayer służy do rejestrowania, ponownego odtwarzania, przekierowywania oraz przetwarzania danych z opasek. Umożliwia konwersję z natywnego typu danych (.muse) na inne (.txt, .mat, .csv). MuseLab wykorzystuje się do wizualizacji danych.

## 2.4. MindWave Mobile 2

MindWave Mobile 2 zbudowane jest z opaski na głowę, klipsu na ucho oraz ramienia z zamocowanym czujnikiem (patrz Rysunek 2.7 na następnej stronie). Elektrody referencyjne oraz uziemiające znajdują się na klipsie na uchu, a elektroda EEG na ramieniu dotykającym czoła w pozycji FP1. Umożliwia pomiar sygnałów EEG, sygnałów NeuroSky eSense, na które składają się skupienie oraz medytacja, oraz mrugnięcie. Do zasilania wykorzystywana jest pojedyncza bateria AAA, która starcza na 8 godzin pracy [32].

Parametry urządzenia zostały zestawione w Tabeli 2.4 na następnej stronie.

Koszt urządzenia na dzień 25 czerwca 2019 roku wynosi 100 \$.

<sup>4</sup>Webinarium – Internetowe seminarium realizowane przy wykorzystaniu streamingu wideo.



**Rysunek 2.7.** Hełm MindWave Mobile 2

Źródło: [32]

**Tabela 2.4.** Parametry MindWave Mobile 2

Źródło: Opracowanie własne na podstawie [32]

Ilość kanałów	1
Umiejscowienie elektrod	FP1
Czujniki referencyjne	<i>brak danych</i>
Rodzaj czujników	<i>brak danych</i>
Rozdzielcość	12 bit
Częstotliwość próbkowania	512 Hz
Detekcja ruchu	Brak
Łączność	Bezprzewodowa Bluetooth (BT/BLE)
Zasilanie	1,5 V (1 bateria AAA), do 8 godzin pracy

NeuroSky dostarcza swoje API dla systemów iOS, Android, macOS oraz Windows. Darmowe narzędzia dla deweloperów składają się z trzech osobnych API [33]:

**ThinkGear Connector (TGC)** — program działający w tle systemu operacyjnego; zarządza komunikacją z hełmem przy pomocy TCP/IP.

**ThinkGear Communication Driver (TGCD)** — biblioteka zawierająca funkcje pozwalające na połączenie oraz przetwarzanie danych z hełmu. Można ją wykorzystywać z językami C/C++, Java oraz C#.

**Protokół komunikacji MindSet** — zawiera specyfikacje protokołu komunikacyjnego; pozwala na integrację hełmu z dowolną platformą/językiem programowania.

Dodatkowo można wykupić narzędzia dla naukowców w cenie 500 \$, na które składają się [34]:

**NeuroView** — aplikacja do wyświetlania oraz zapisywania danych EEG w czasie rzeczywistym. Pozwala na rejestrację czystego sygnału EEG, jego poszczególnych składowych oraz odczytów NeuroSky eSense dla skupienia oraz medytacji. Umożliwia eksport do plików CSV.

**NeuroSkyLab** — aplikacja zapewniająca integrację ze środowiskiem MATLAB.

## 2.5. OpenBCI Ultracortex Mark IV

Ultracortex, pokazany na Rysunku 2.8 na następnej stronie, jest open source'owym hełmem zaprojektowanym do pracy ze wszystkimi układami OpenBCI [44]. Pozwala na rejestrację sygnałów EEG, EMG oraz ECG. Wspiera do 16 kanałów rozmieszczonych na 35 różnych lokalizacjach według systemu 10-20<sup>5</sup>.

W projekcie zastosowano *suche* sensory EEG. Ich rozmieszczenie pokazano na Rysunku 2.9 na następnej stronie.

Czas założenia oraz uruchomienia hełmu wynosi poniżej 30 sekund.

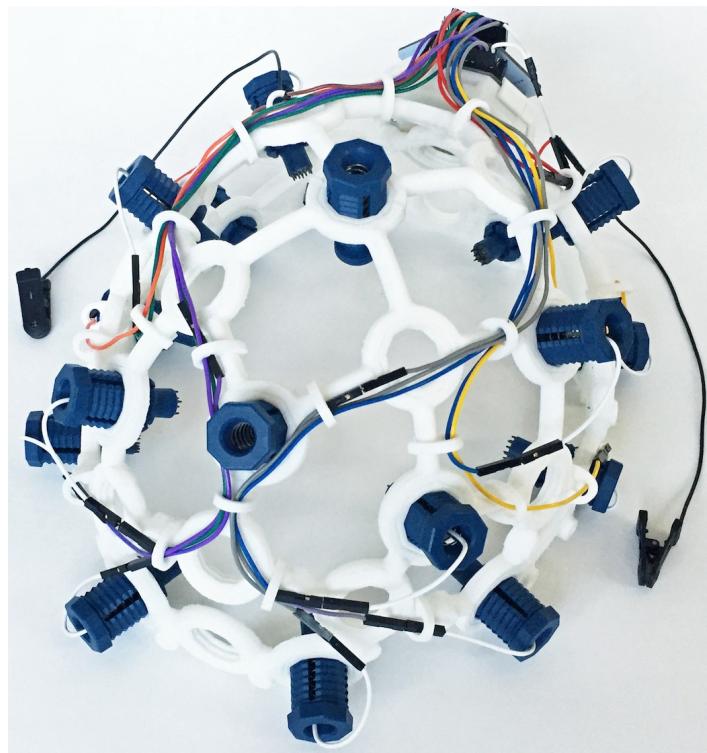
Produkt można kupić w trzech wariantach:

1. Do samodzielnego druku — dostarczane są wszystkie części hełmu oprócz tych, które można wydrukować na drukarce 3D. Hełm należy zmontować samodzielnie na podstawie dokumentacji. Cena: 300–400 \$.
2. Niezmontowany — dostarczane są wszystkie części hełmu, również z tymi, które można wydrukować na drukarce 3D. Hełm należy zmontować samodzielnie na podstawie dokumentacji. Cena: 500–600 \$.
3. Zmontowany — dostarczany jest całkowicie zmontowany hełm. Cena: 700–850 \$.

Dodatkowo należy zakupić wybraną płytę OpenBCI. W dalszej części rozdziału zamieszczono opisy dostępnych układów. Parametry hełmu *Mark IV* w połączeniu z kompatybilnymi płytami zamieszczono w Tabeli 2.5 na stronie 37.

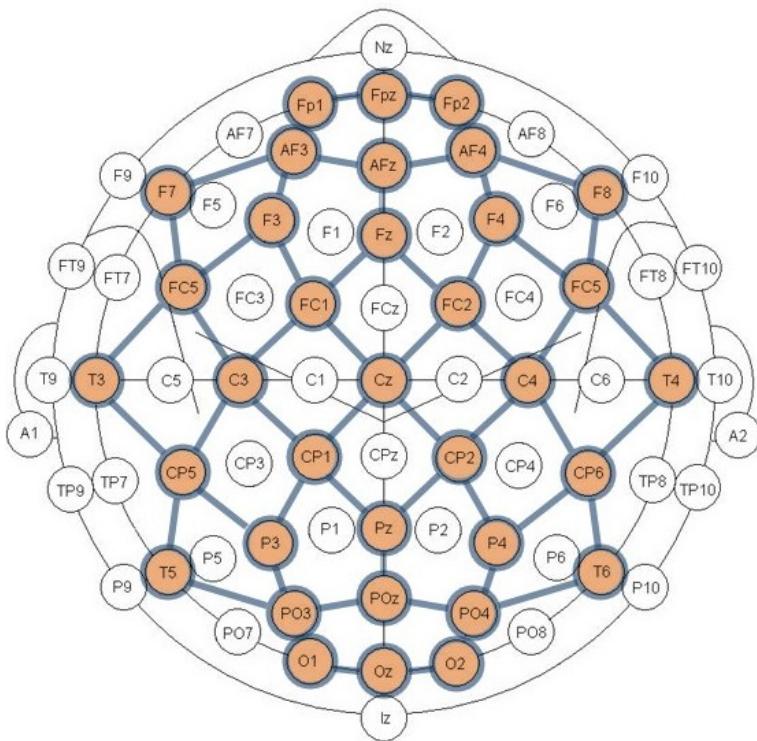
**Ganglion Board** — Ganglion (patrz Rysunek 2.10 na stronie 35) posiada 4 kanały, które mogą być użyte do mierzenia sygnałów EMG, EKG lub EEG [40]. Częstotliwość próbkowania wynosi 200 Hz.

<sup>5</sup>System 10-20 – system opisu umiejscowienia elektrod; składa się z 21 elektrod. Podstawą tego standardu jest zdefiniowanie konturów między punktami orientacyjnymi czaszki (np. nasion, inion), a następnie podzielenie ich na proporcjonalne odległości 20% całkowitej długości. Istnieją również systemy pokrewne, np. 10-10 oraz 10-5, które używają odpowiednio 10% oraz 5% całkowitej długości [51, rozdz. 6].



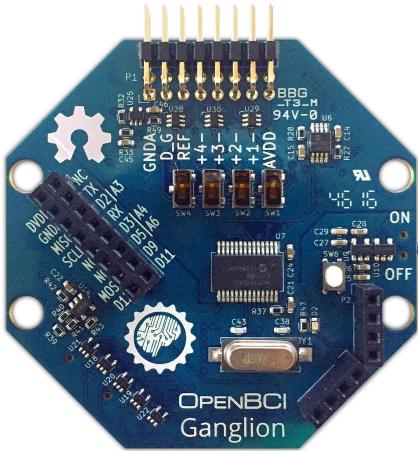
**Rysunek 2.8.** Hełm Ultracortex Mark IV

Źródło: [43]



**Rysunek 2.9.** Rozmieszczenie sensorów w hełmie Ultracortex Mark IV

Źródło: [44]



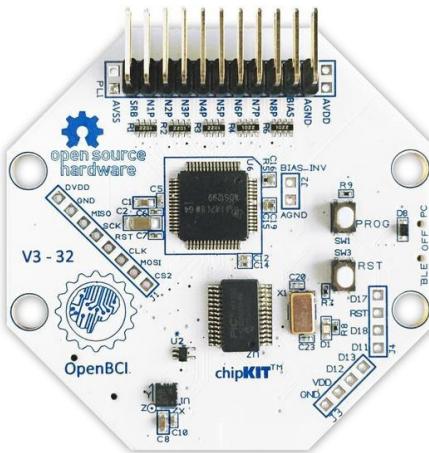
## Rysunek 2.10. OpenBCI Ganglion Board

Źródło: [40]

Komunikacja bezprzewodowa jest realizowana z wykorzystaniem Simblee BLE, modułu Bluetooth 4.0 kompatybilnego z Arduino, który pozwala na łatwą integrację z IoT<sup>6</sup>. Moduł umieszczony na płytce Ganglion jest zaprogramowany i gotowy do pracy, jednak część jego wyprowadzeń pozostaje dostępna dla użytkownika do celów własnych.

Koszt układu to 200 \$.

**Cyton Board** – Cyton, pokazany na Rysunku 2.11, jest kompatybilny z Arduino, posiada 8 kanałów oraz 32-bitowy procesor [39]. Podobnie jak Ganglion, może być wykorzystany do mierzenia aktywności mięśni, serca lub mózgu. Dane próbkiowane są z częstotliwością 250 Hz.



**Rysunek 2.11.** OpenBCI Cyton Board

Źródło: [39]

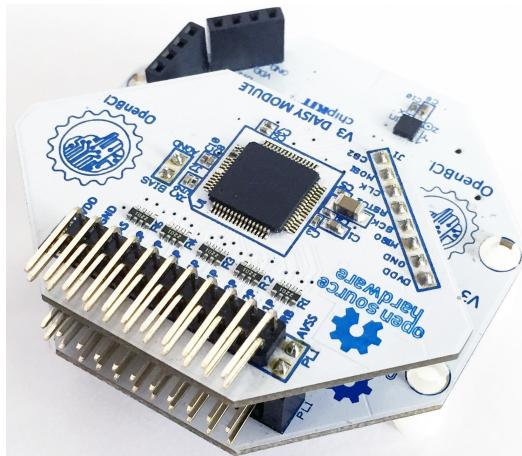
<sup>6</sup>IoT (ang. Internet of Things) – Internet rzeczy. Koncepcja zgodnie z którą urządzenia mogą komunikować się ze sobą, wykorzystując do tego celu internet.

Do komunikacji bezprzewodowej wykorzystywany jest moduł RFduino. Komunikacja z komputerem może być realizowana przy pomocy adaptera OpenBCI USB lub, jeżeli komputer posiada tę technologię, wbudowanego modułu BLE.

Koszt płytki to 500 \$.

**Cyton + Daisy Board** — Ten układ zawiera opisaną wcześniej płytę Cyton oraz Daisy, która jest wpinana w płytę Cyton. Układ pokazano na Rysunku 2.12.

Daisy zapewnia dodatkowe 8 kanałów, które mogą być użyte do mierzenia EMG, EKG lub EEG [38]. Częstotliwość próbkowania wynosi 250 Hz.



**Rysunek 2.12.** OpenBCI Cyton + Daisy Board

Źródło: [38]

Koszt układu to 950 \$. Płytkę Daisy nie może być zakupiona oddziennie.

Oprogramowanie OpenBCI jest open source'owe i jest dostępne na ich profilu na GitHubie (patrz [41]).

Głównym narzędziem do wizualizacji, nagrywania oraz komunikacji z płytami jest OpenBCI GUI [42]. Narzędzie jest kompatybilne z systemami macOS, Windows oraz Linux. Pozwala na filtrowanie oraz przetwarzanie danych w czasie rzeczywistym; umożliwia również przekierowywanie danych wykorzystując protokoły UDP, OSC, LSL oraz port szeregowy.

Do komunikacji z układami stosowane są narzędzia OpenBCI Ganglion SDK oraz OpenBCI Cyton SDK, wykorzystujące protokół oparty na znakach ASCII<sup>7</sup>. OpenBCI dostarcza również bibliotekę Brainflow, która służy do analizy danych EMG, EKG oraz EEG [37]. Umożliwia ona programowanie w językach Python, Java, R, C++<sup>8</sup>, Matlab oraz C#.

Platforma OpenBCI umożliwia również współpracę z innymi programami do analizy sygnałów, m.in. MATLAB, Neuromore, OpenVIBE, LabStreamingLayer oraz BrainBay.

<sup>7</sup>ASCII (ang. American Standard Code for Information Interchange) – siedmiobitowy system służący do kodowania znaków.

<sup>8</sup>Mimo iż biblioteka została napisana jest w C/C++, dodano nakładkę na język C++ w celu zapewnienia warstwy abstrakcji.

**Tabela 2.5.** Parametry Ultracortex Mark IV w połączeniu z kompatybilnymi układami  
 Źródło: Opracowanie własne na podstawie [38, 39, 40, 44]

Parametr	Ganglion	Cyton	Cyton + Daisy
Ilość kanałów	4	8	16
Umiejscowienie elektrod	Konfigurowalne <sup>a</sup>	Konfigurowalne <sup>a</sup>	Konfigurowalne <sup>a</sup>
Rodzaj czujników	Suche	Suche	Suche
Rozdzielcość	24 bit	24 bit	24 bit
Częstotliwość próbkowania	200 Hz	250 Hz	250 Hz
Rejestrowane parametry	EMG, EKG, EEG <sup>b</sup>	EMG, EKG, EEG <sup>b</sup>	EMG, EKG, EEG <sup>b</sup>
Detekcja ruchu	3 osiowy akcelerometr	3 osiowy akcelerometr	3 osiowy akcelerometr
Kompatybilność	macOS, Windows, Linux	macOS, Windows, Linux	macOS, Windows, Linux
Łączność	BLE 4.0 przez adapter	BLE	BLE
Zasilanie	6V (4 baterie AA)	6V (4 baterie AA)	6V (4 baterie AA)

<sup>a</sup> Mark IV wspiera 35 lokalizacji według standardu 10-20. Punkty w których można umieścić elektrody pokazano na Rysunku 2.9 na stronie 34.

<sup>b</sup> Sensory nie są dołączane do płytek i należy je zakupić oddzielnie.



## **ROZDZIAŁ 3**

# **Projekt wirtualnej klawiatury**

W rozdziale trzecim omówiono opracowaną aplikację. W pierwszej części rozdziału sformułowano założenia projektowe. Dalej omówiono zagadnienia związane z doborem narzędzi wymaganych do realizacji pracy, w tym urządzenia rejestrującego aktywność mózgu, języka programowania, środowiska programistycznego oraz systemu kontroli wersji. Kolejną częścią jest omówienie samej aplikacji, z wyszczególnieniem struktury jej kodu źródłowego, algorytmu działania, metodyki treningu detekcji komend mentalnych, protokołu komunikacyjnego urządzenia rejestrującego oraz procesu decyzyjnego aplikacji. Rozdział zawiera również omówienie interfejsu użytkownika.

### **3.1. Założenia projektowe**

Przed przystąpieniem do realizacji projektu wirtualnej klawiatury sformułowano założenia przedstawione poniżej.

1. Aplikacja powinna umożliwiać wprowadzanie następujących znaków: liter A–Z, cyfr 0–9 oraz symboli przecinka i kropki. Powinna również umożliwiać wstawianie spacji, przejście do nowej linii oraz usunięcie ostatniego znaku.
2. Wprowadzona wiadomość powinna być wyświetlana na interfejsie użytkownika. Wyślanie (zaakceptowanie) wiadomości powinno odczytać ją przy pomocy syntezatora mowy.
3. Domyślnym językiem aplikacji powinien być język angielski.
4. Nawigacja po klawiaturze oraz selekcja (wciśnięcie) poszczególnych klawiszy powinna odbywać się głównie z wykorzystaniem poleceń mentalnych. Opcjonalnie, jeżeli jest to konieczne, oprócz poleceń mentalnych mogą zostać wykorzystane do tego celu inne sygnały pozyskane z jednostki akwizycyjnej, na przykład EOG czy EMG.

5. Aplikacja powinna umożliwiać sterowanie manualne (bez wykorzystania hełmu) w celu łatwego przetestowania algorytmów selekcji.
6. Poszczególne parametry algorytmu obsługi klawiatury powinny być konfigurowalne dla użytkownika.
7. Aplikacja powinna mieć zaimplementowany mechanizm zapisywania ostatnio sprecyzowanych parametrów algorytmu.
8. Użytkownik aplikacji nie powinien musieć w sposób bezpośredni porozumiewać się z API urządzenia; nie powinien w ogóle musieć znać jego protokołu komunikacyjnego.
9. Aplikacja powinna mieć zaimplementowany mechanizm bezpośredniej wymiany wiadomości z urządzeniem, jednak jego użytkowanie powinno być całkowicie opcjonalne i niewymagane do sprawnej pracy z urządzeniem.

## 3.2. Wybór narzędzi

### 3.2.1. Urządzenia rejestrujące

Wybierając urządzenie rejestrujące poszukiwano urządzenia, które (1) jest nieinwazyjne, (2) posiada rozwinięte SDK oraz bazę przykładowych aplikacji, (3) zapewnia sprzętowe wsparcie w zakresie detekcji komend mentalnych, (4) jest przystępne cenowo – kosztuje poniżej 500 \$. Dodatkową zaletą urządzenia, aczkolwiek niewymagana do realizacji niższej pracy, byłaby (6) możliwość rejestracji dodatkowych sygnałów, na przykład przy użyciu akcelerometru. Wyboru dokonywano spośród urządzeń omówionych w rozdziale 2 na stronie 25.

W przypadku OpenBCI Ultracortex Mark IV, jednym z jego mankamentów jest wysoka cena – nawet w przypadku zakupu najtańszego wariantu, a więc hełmu do samodzielnego druku oraz montażu wraz z układem Ganglion Board, cena wynosiła 500 \$. Po doliczeniu do tej kwoty wymaganych elektrod cena znacznie wzrasta. W przypadku oferowanego przez firmę OpenBCI Starter Kitu, który zawiera wszystkie wymagane peryferiale, cena wynosi blisko 950 \$. Samodzielność montażu w odczuciu autora pracy jest ryzykownym posunięciem w przypadku tak delikatnych oraz precyzyjnych urządzeń, tym bardziej że nie jest to standardowy model sprzedaży tej gamy urządzeń. Niewątpliwą zaletą urządzeń OpenBCI jest ich otwartoźródłowość – oprogramowanie jest dostępne na serwisie GitHub. Inną zaletą jest jego duża popularność oraz integracja z wieloma programami do analizy danych, jak na przykład MATLAB.

Urządzenia Muse 2 oraz MindWave Mobile 2 w odczuciu autora są narzędziami mało precyzyjnymi, nie nadającymi się do wykorzystania w celach badawczych. Ich zaletami jest łatwość nakładania, niska cena oraz smukłość, która prawdopodobnie pozwala na długie sesję bez odczuwania dyskomfortu związanego z naciskiem urządzenia. Wadą Muse 2 jest brak SDK dla programistów. Wadą MindWave Mobile 2 jest liczba elektrod rejestrujących – urządzenie jest wyposażone w wyłącznie jedną.

Emotiv Insight oraz EPOC+ są bardzo popularne wśród osób zajmujących się tematyką BCI. Za pomocą rozbudowanego API pozwalają na detekcję nie tylko aktywności mózgu w poszczególnych pasmach, ale również komend mentalnych oraz mimiki. Ze względu na wykorzystywany protokół komunikacyjny, są uniwersalne pod względem platformy programistycznej oraz języka programowania. Ich największym mankamentem jest brak darmowego dostępu do surowych danych EEG – żeby je uzyskać trzeba wykupić licencję. Urządzenie EPOC+ charakteryzuje się lepszymi parametrami niż Insight, jednak jest od niego droższe. Inną jego wadą jest rodzaj czujników; EPOC+ używa nasączanych solą fizjologiczną, podczas kiedy czujniki w Insight są wykonane z półsuchego polimeru.

Ostatecznie zakupiono urządzenie Emotiv Insight. Nabyto najnowszą wersję urządzenia; premiera modelu odbyła się w maju 2019 roku. Niniejsza praca jest jednym z pierwszych projektów powstających na jego nowej odsłonie. Głównym czynnikiem przemawiającym na jego korzyść był dostęp do zaawansowanego API, które dobrze wpasowuje się w założenia pracy. Urządzenie posiada również atrakcyjną cenę; w opinii autora spośród omówionych modeli Emotiv Insight zapewnia najlepszy stosunek ceny do oferowanych możliwości.

### **3.2.2. Język programowania**

Wybierając język programowania poszukiwano języka, który (1) jest językiem wysokiego poziomu, (2) jest językiem obiektowym, (3) umożliwia łatwe tworzenie interfejsu użytkownika, (4) posiada kompleksową dokumentację, (5) charakteryzuje się potencjałem przyszłego rozwoju oraz (6) dużą popularnością.

Języki wysokiego poziomu ułatwiają proces tworzenia oprogramowania, zwiększając czytelność kodu oraz poziom jego abstrakcji. Pozwalają na zdystansowanie się od warstwy sprzętowej komputera oraz większe skupienie się na aspektach związanych z logiką programu.

Języki obiektowe ułatwiają tworzenie dużych aplikacji. Programuje się w nich z użyciem obiektów, które stanowią instancje klas, które z kolei są definicją zmiennych, pól oraz metod dla obiektów danego typu. Wśród paradygmatów programowania obiektowego możemy wyróżnić abstrakcję, hermetyzację, polimorfizm oraz dziedziczenie.

Interfejs użytkownika umożliwia osobie użytkującej system na interakcję z oprogramowaniem. Wybrany język programowania powinien mieć zapewnioną łatwość tworzenia graficznych interfejsów; ich tworzenie powinno być możliwe natywnie, bez konieczności używania zewnętrznych bibliotek.

Język powinien mieć kompleksową oraz dobrze sformułowaną dokumentację. Dokumentacja techniczna stanowi źródło częstych odniesień w procesie powstawania oprogramowania, a zatem powinna być łatwo dostępna i umożliwiać szybkie odnajdywanie w niej potrzebnych informacji.

Potencjał przyszłego rozwoju pozwoli autorowi na wykorzystanie umiejętności zdobytych w trakcie realizacji niniejszej pracy w przyszłym życiu zawodowym. Uwzględnienie tego kryterium przy wyborze danej technologii stanowi dobrą formę motywacji, która jest kluczowa do sprawnej realizacji pracy.

Używając języków o dużej popularności znacznie łatwiej jest posiłkować się serwisami dla programistów, takimi jak na przykład Stack Overflow. Dzięki swojej zaangażowanej społeczności, ułatwiają one proces rozwiązywania ewentualnych problemów, które mogą wystąpić w trakcie tworzenia oprogramowania.

Wśród wstępnie wyselekcjonowanych języków programowania znalazły się języki C++, C#, Java oraz JavaScript.

JavaScript został wyeliminowany jako pierwszy, ze względu na fakt, iż interpretacja kodu w nim tworzony jest zależna od przeglądarki internetowej. Wymusza to testowanie oprogramowania na różnych wersjach przeglądarek, co w efekcie wydłuża czas powstawania aplikacji. Inną wadą JavaScript jest brak dostępu do zaawansowanych technik debugowania, do których dostęp istnieje w przypadku języków kompilowanych.

C++, pomimo wielu dostępnych materiałów oraz zaawansowanego narzędzia do tworzenia interfejsów użytkownika w postaci Qt, został wykluczony ze względu na brak jego znajomości przez autora pracy.

Dokonując ostatecznego wyboru pomiędzy C# oraz Java, zdecydowano się zrealizować aplikację w języku C#. Autor pracy posiada większe doświadczenie w pracy z tym językiem, przez co realizacja w nim oprogramowania będzie w większym stopniu pozbacona antywzorców oraz wykonana w zgodzie z ogólnie przyjętymi standardami.

Wybranie języka C# wiąże się również z wyborem technologii tworzenia interfejsu graficznego; wybór dokonywano spośród Windows Forms (WinForms), Windows Presentation Foundation (WPF) oraz Universal Windows Platform (UWP). Zdecydowano się skorzystać z WPF ze względu na dodatkowe możliwości jakie oferuje pod kątem dostosowywania interfejsu użytkownika w odniesieniu do WinForms oraz brakiem ograniczeń, którymi charakteryzują się aplikacje uniwersalne [29].

### **3.2.3. Środowisko programistyczne**

Najpopularniejszymi środowiskami programistycznymi do języka C# są Visual Studio, Rider oraz Visual Studio Code.

Zarówno Visual Studio jak i Rider są bardzo zaawansowanymi środowiskami. Oba programy posiadają funkcjonalności usprawniające tworzenie kodu, jego debugowanie, integrację testów oraz zarządzanie projektem, co czyni je niezwykle potężnymi narzędziami do pracy z językiem C#. W odczuciu autora główne różnice sprowadzają się do dwóch cech:

1. Visual Studio posiada darmową wersję Community, podczas gdy Rider, pomijając miesięczną wersję trial, wymaga zakupienia licencji,
2. Rider zapewnia znacznie więcej funkcjonalności dotyczących inspekcji kodu, czyli wykrywania oraz zastępowania jego fragmentów w celu ich uproszczenia lub uniknięcia potencjalnych błędów [27].

Visual Studio Code, pomimo iż z definicji jest edytorem kodu, a nie zintegrowanym środowiskiem programistycznym, posiada wiele funkcjonalności pełnoprawnego IDE. Posiada między innymi wsparcie dla debugowania kodu, wersjonowania oprogramowania,

zupełniania kodu oraz kolorowania składni. Jego możliwości mogą być powiększane przez obszerną bazę rozszerzeń. Do jego zalet należy znaczna szybkość, przejrzysty interfejs oraz cross-platformowość. Wadami narzędzi są niskie możliwości w zakresie inspekcji kodu oraz debugowania. VS Code jest darmowe.

Ostatecznie zdecydowano się na zrealizowanie projektu, wykorzystując do tego celu Visual Studio. Rider został odrzucony ze względu na konieczność zakupienia licencji oraz jego krótki okres obecności na rynku, przez co opinie na jego temat mogą nie być obiektywne. Visual Studio Code nie zostało wybrane ze względu na swoje niskie możliwości w zakresie refaktoryzacji kodu.

### **3.2.4. System kontroli wersji**

Jako system kontroli wersji wybrano Git. Opracowana aplikacja jest hostowana w prywatnym, niedostępnym dla osób postronnych, repozytorium na serwisie GitHub.

## **3.3. Omówienie aplikacji**

### **3.3.1. Struktura kodu źródłowego**

Rozwiązanie (ang. Solution) zostało podzielone na dwa projekty: HeadsetController oraz VirtualKeyboard. Wydzielenie osobnych projektów odseparowało logikę związaną z obsługą samego hełmu oraz z wirtualną klawiaturą – umożliwia to dołączanie projektu HeadsetController do innych aplikacji, nawet tych niewykonanych w języku C#, bez konieczności ingerencji w jego kod. Struktura rozwiązania w postaci drzewa została przedstawiona na Rysunku 3.1 na następnej stronie.

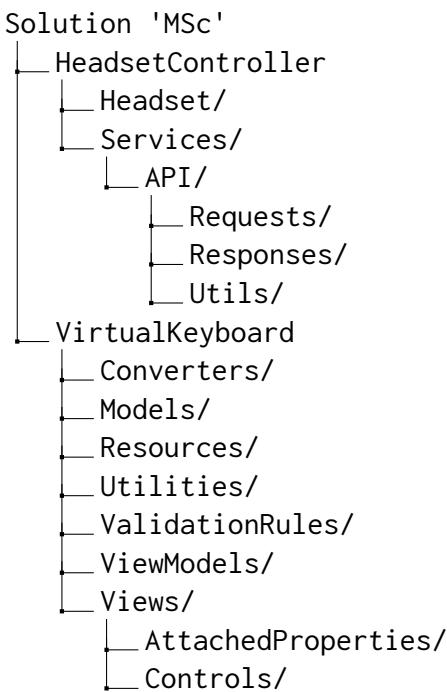
HeadsetController jest biblioteką służącą do komunikacji z urządzeniem. Jest projektem o typie wynikowym DLL<sup>1</sup> (output type: class library). Nie stanowi samodzielnej aplikacji; jest przeznaczony do bycia dołączonym do innych projektów. W projekcie znajdują się dwa główne foldery: Headset oraz Services.

Do folderu Headset należą klasy Headset.cs oraz Insight.cs. Headset.cs jest klasą abstrakcyjną realizującą niskopoziomową komunikację z urządzeniami Emotiv – zajmuje się przetwarzaniem danych otrzymanych protokołem WebSocket, a następnie, na podstawie zawartości otrzymanej wiadomości, zgłaszaniem odpowiedniego zdarzenia. Insight.cs stanowi programową reprezentację hełmu Emotiv Insight; umożliwia dostęp do jego parametrów, na przykład poziomu naładowania, jakości połączenia poszczególnych elektrod lub siły komend mentalnych. Dziedziczy po klasie Headset.cs.

W folderze Services znajdują się klasy JsonParser.cs oraz WebSocket.cs, które stanowią klasy opakowujące (ang. wrappers) na operacje kolejno na formacie JSON oraz protokole WebSocket. Dzięki zastosowaniu takiego podejścia, wewnętrzna implementacja tych klas jest ukryta przed użytkownikami biblioteki. Dla programisty dostępne są wyłącznie wysokopoziomowe metody niezbędne do pracy z urządzeniem, co wpisuje się

---

<sup>1</sup>DLL (ang. Dynamic-link library) – biblioteka łączona dynamicznie; rodzaj biblioteki łączonej z programem dopiero w trakcie jego wykonywania.



**Rysunek 3.1.** Struktura rozwiązania przedstawiona w postaci drzewa

Źródło: Opracowanie własne

w zasadę hermetyzacji. W tym folderze znajduje się również folder API, który zawiera trzy podkatalogi: Requests, Responses oraz Utils. Pierwsze dwa z nich zawierają definicję zapytań oraz odpowiedzi, pogrupowanych w dalsze foldery zgodnie z nomenklaturą zawartą w dokumentacji protokołu komunikacyjnego urządzenia. Folder Utils zawiera klasę Enums.cs, w której umieszczone są typy wyliczeniowe wykorzystywane do komunikacji.

Projekt VirtualKeyboard jest aplikacją Windows (output type: Windows application) wykorzystującą strukturę interfejsu Windows Presentation Foundation (WPF). Zawiera 7 głównych folderów: Converters, Models, Resources, Utilities, ValidationRules, ViewModels oraz Views.

Katalog Converters zawiera implementacje interfejsów IValueConverter oraz IMultiValueConverter. Są w nim zawarte konwertery umożliwiające stosowanie nie-standardowej logiki do tworzenia powiązań między interfejsem użytkownika, a właściwościami (ang. properties) w aplikacji. Przykładem zaimplementowanego konwertera może być BooleanToVisibilityConverter.cs, który zwraca wartość typu wyliczeniowego Visibility na podstawie przekazanej zmiennej typu boolean. I tak, dla parametru **true** zwraca wartość Visibility.Visible, a dla **false** – Visibility.Hidden.

Folder Models zawiera klasy odpowiedzialne za logikę biznesową aplikacji. Umieszczona jest w nim klasa Settings.cs, przechowująca ustawienia dotyczące progów aktywacji dla poszczególnych komend mentalnych, czasu skupienia wymaganego do ich aktywacji oraz wartości określającej czy włączony został tryb offline aplikacji. Klasa jest deserializowana z pliku settings.json wraz ze startem aplikacji oraz serializowana do niego podczas jej zamknięcia. Dzięki zastosowaniu mechanizmu serializacji użytkownik

nie traci ustawień poprzedniej sesji. W folderze Models znajduje się również definicja klasy KeyboardNavigator.cs, która odpowiada za proces decyzyjny dotyczący wyboru aktualnie zazначенego klawisza klawiatury oraz jego wcisnięcia.

Folder Resources zawiera graficzne zasoby wykorzystywane w aplikacji. Przechowywana jest tu między innymi ikona aplikacji – keyboard.ico.

W katalogu Utilities znajduje się klasa SettingsSerializer.cs, która odpowiada za serializację oraz deserializację klasy Settings.cs oraz jej zapis/odczyt z pliku. Plik wynikowy settings.json zostaje umieszczony w folderze, w którym znajduje się aplikacja.

Folder ValidationRules zawiera klasę NotEmptyValidationRule.cs, która jest wykorzystywana do potwierdzenia poprawności wprowadzonych przez użytkownika ustawień połączenia z hełmem Emotiv Insight.

Katalog ViewModels zawiera tak zwane ViewModele, czyli klasy implementujące interfejs INotifyPropertyChanged, które stanowią centrum wykorzystywanego w aplikacji wzorca MVVM<sup>2</sup>. Za pomocą mechanizmu komend oraz wiązania danych (ang. data binding) są one połączone z widokami.

Ostatni katalog, Views, zawiera definicje widoków w postaci plików XAML<sup>3</sup>. Zawarte są w nim również foldery AttachedProperties, w których przechowywane są właściwości rozszerzające zdefiniowane kontrolki, oraz Controls, w którym znajduje się definicja stworzonej przez autora pracy kontrolki KeyView, która jest składową opracowanej klawiatury.

### 3.3.2. Algorytm działania

Na Rysunku 3.2 na następnej stronie przedstawiono schemat blokowy algorytmu aplikacji.

W momencie uruchomienia aplikacji następuje wczytanie ustawień zapisanych w pliku settings.json; plik zapisany jest w formacie JSON. W celu załadowania jego danych następuje deserializacja do obiektu klasy Settings.cs.

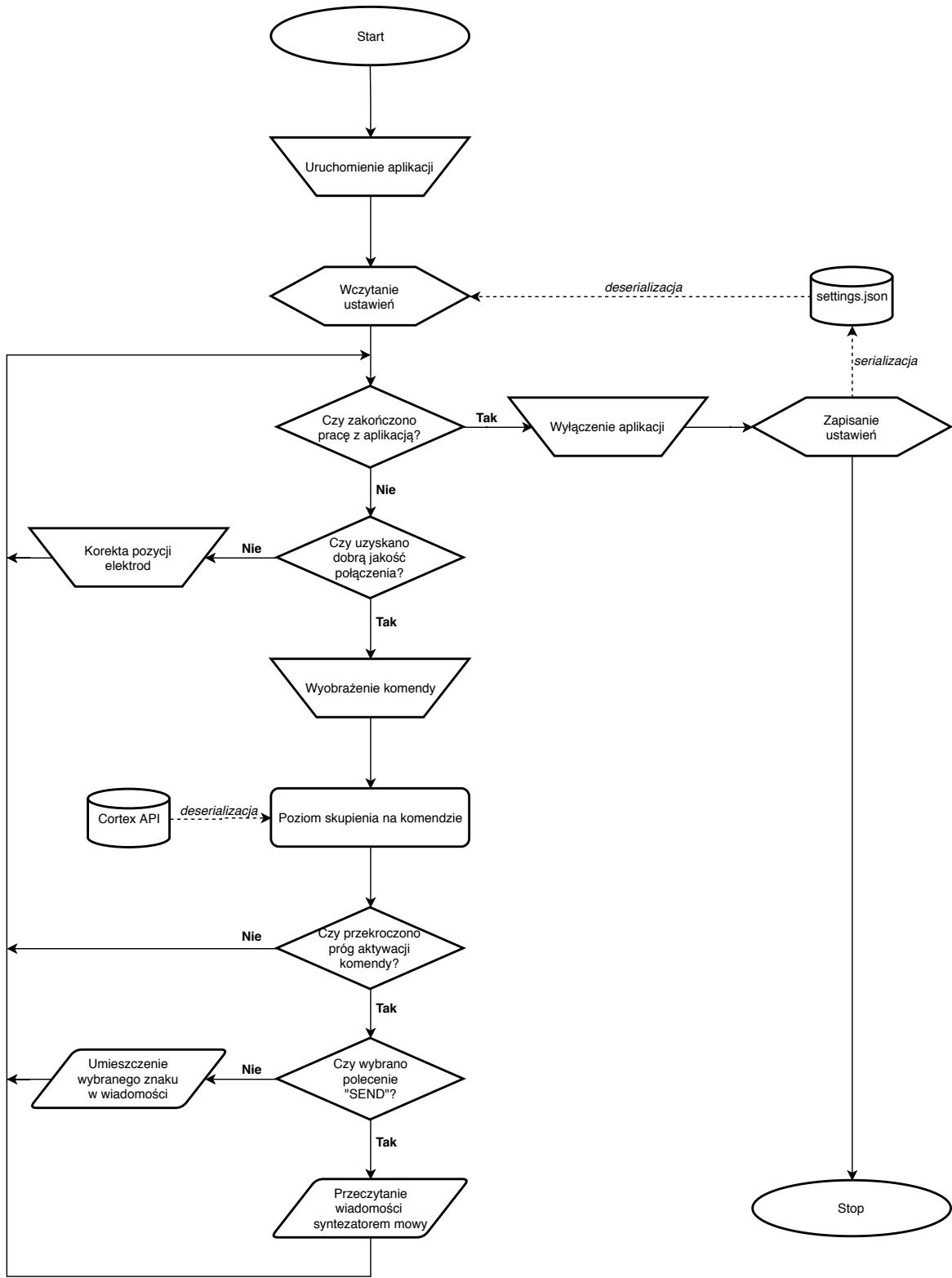
Po uruchomieniu aplikacji użytkownik powinien nałożyć hełm Emotiv Insight, dając przy tym do uzyskania jak najwyższej jakości sygnału. Jakość sygnału jest uśredniana do pojedynczej wartości procentowej oznaczonej na interfejsie użytkownika jako Contact Quality (patrz rozdział 3.4.1 na stronie 56). Użytkownik może korygować położenie czujników na głowie, bazując na diagramie przedstawiającym głowę ze schematycznie oznaczonymi czujnikami, który został umieszczony w interfejsie użytkownika. Maksymalna możliwa do osiągnięcia jakość będzie różna w zależności od użytkownika; niezależnie od osiągniętych wartości praca z hełmem będzie możliwa, jednak im wyższa jakość połączenia, tym bardziej miarodajne będą sygnały odebrane z urządzenia.

W celu wykonania dowolnej komendy, a więc przesunięcia selekcji w góre, w dół,

---

<sup>2</sup>MVVM (ang. Model–View–ViewModel) – wzorzec architektury oprogramowania zapewniający sepeację warstw UI oraz biznesowej; silnie zakorzeniony w aplikacjach WPF.

<sup>3</sup>XAML (ang. Extensible Application Markup Language) – oparty na XML język znaczników wykorzystywany do opisu interfejsu użytkownika w między innymi aplikacjach Windows Presentation Framework.



**Rysunek 3.2.** Schemat blokowy algorytmu aplikacji

Źródło: Opracowanie własne

w prawo, w lewo lub wciśnięcie klawisza, należy skupić się na danej akcji. Aplikacja, w momencie przekroczenia poziomu skupienia dla danej komendy zdefiniowanego w zakładce *Settings*, wykona żądaną akcję. Jeżeli akcją będzie wciśnięcie klawisza z danym znakiem, zostanie on dodany do okna wiadomości znajdującego się w interfejsie użytkownika. Jeżeli akcją będzie wciśnięcie klawisza *SEND*, tekst znajdujący się aktualnie w oknie wiadomości zostanie przeczytany przy użyciu syntezatora mowy, a następnie skasowany.

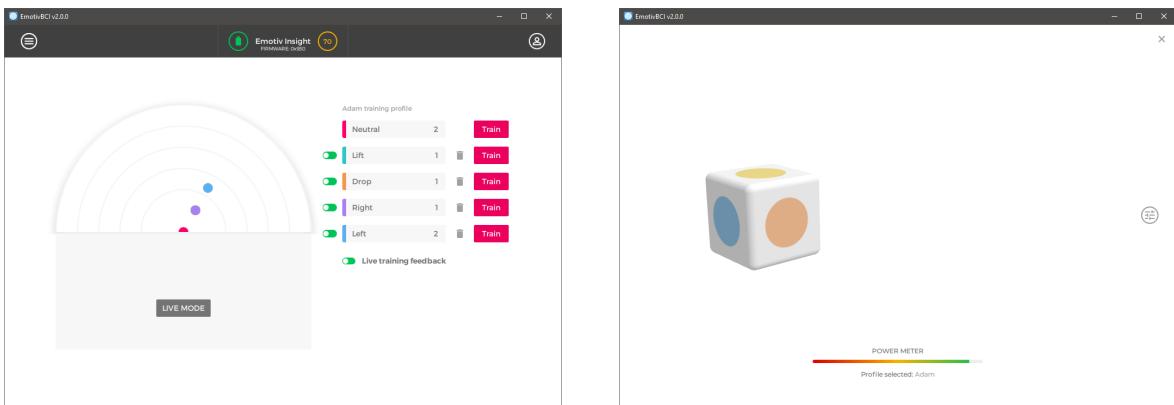
W momencie zamknięcia aplikacji wywoływana jest metoda zapisująca aktualnie zdefiniowane ustawienia. Odbywa się to poprzez serializację obiektu klasy *Settings.cs* do pliku *settings.json*.

### 3.3.3. Trening detekcji komend mentalnych

Poprzez swoje API Emotiv Insight umożliwia wykrywanie maksymalnie do 5 komend mentalnych: stanu neutralnego oraz wybranych czterech spośród następujących trzynastu wyobrażeń: pchnięcia, przyciągnięcia, uniesienia, opuszczenia, przesunięcia w lewo, przesunięcia w prawo, obrotu zgodnie ze wskazówkami zegara, obrotu przeciwne do wskazówek zegara, obrotu w przód, obrotu w tył, obrotu w lewo, obrotu w prawo oraz zniknięcia. Dodatkowo wspomaga detekcję stanów emocjonalnych, to jest ekscytacji, długotrwałej ekscytacji, stresu, zaangażowania, odpreżenia, zainteresowania oraz skupienia. Spośród mimiki umożliwia wykrywanie mrugnięcia, puszczenia oczka, zmarszczenia brwi, uniesienia brwi, uśmiechu i zaciśnięcia zębów.

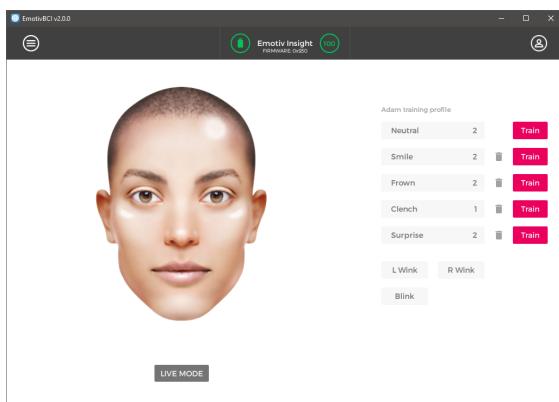
Wykorzystując aplikację EmotivBCI możliwe jest trenowanie detekcji poszczególnych komend mentalnych w czasie rzeczywistym. Trening jest interaktywny; użytkownik, obserwując wirtualny sześciian, wyobraża sobie kolejne komendy, a system rejestruje aktywność mózgu związaną z danym wyobrażeniem. W miarę postępu oraz nasilania skupienia na danej akcji sześciian wykonuje wyobrażoną czynność, na przykład obraca się w prawo lub znika. Trening detekcji mimiki, podobnie jak komend mentalnych, jest również interaktywny – użytkownik obserwuje postęp na wirtualnym awatarze.

Istotną kwestią jest wybór odpowiednich poleceń mentalnych do obsługi wirtualnej klawiatury. Intuicyjność interakcji z systemem pozwoli użytkownikowi na szybsze przyzwojenie zasad sterowania oraz zniweluje ilość błędów związanych z mylением poleceń. Opracowany projekt wymaga pięciu komend: przesunięcia selekcji w góre, w dół, w prawo, w lewo oraz akceptację (wciśnięcie) aktualnie oznaczonego klawisza. O ile w przypadku przesunięcia selekcji dobór komend mentalnych wydaje się oczywisty (są to polecenia *uniesienie*, *opuszczenie*, *przesunięcie w lewo* i *przesunięcie w prawo*), to komenda akceptacji wyboru wymaga głębszej analizy; czy powinno to być *pchnięcie*, symulujące wciśnięcie klawisza klawiatury, a może *przyciągnięcie*, czyli zwolnenie klawisza i przejście do wyboru kolejnej litery? Należy mieć również na uwadze, iż zwiększoną ilość komend możliwych do wykrywania będzie prowadziła do powiększenia stopy błędu detekcji. Ponieważ w niniejszej aplikacji akceptacja selekcji jest kluczowym poleceniem, postanowiono wykorzystać do niego detekcję wyrazów twarzy. Zdecydowano, iż będzie za nią odpowiadać *zmarszczenie brwi*, które osiągało najkorzystniejszy stosunek poprawnych doomyłkowych.

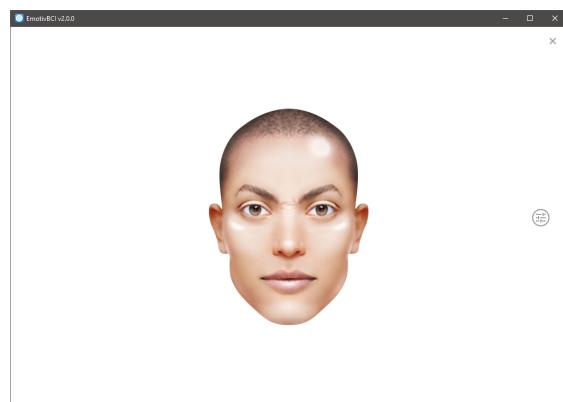


(a) Zestawienie wszystkich wytrenowanych komend mentalnych

(b) Obserwacja w trybie *Live Mode* podczas wyobrażenia komendy *w lewo*



(c) Zestawienie wszystkich wytrenowanych detekcji mimiki



(d) Obserwacja w trybie *Live Mode* podczas wykonania akcji *zmarszczenia brwi*

**Rysunek 3.3.** Proces treningu komend z wykorzystaniem aplikacji *EmotivBCI*

Źródło: Opracowanie własne

kowych klasyfikacji spośród dostępnych możliwości detekcji.

Zrzuty ekranu z procesu treningu komend pokazano na Rysunku 3.3. Na Rysunkach 3.3a oraz 3.3c pokazano zestawienie wszystkich wytrenowanych modeli detekcji. Na Rysunkach 3.3b oraz 3.3d zaprezentowano zrzuty ekranu wykonane w trakcie rozpoznania przez aplikację *EmotivBCI* wytrenowanych wzorców.

### 3.3.4. Komunikacja z urządzeniem

Komunikacja z Cortex API, czyli wrapperem SDK Emotiv, odbywa się przy wykorzystaniu protokołu WebSocket oraz formatu danych JSON. Do nawiązania połączenia z hełmem wymagany jest Bluetooth w wersji 4.0 lub klucz sprzętowy USB (ang. USB dongle).

W celu uzyskania identyfikatora aplikacji (ang. application ID) koniecznego do wygenerowania identyfikatora klienta (ang. client ID) oraz sekretu (ang. client secret), zarejestrowano aplikację *ict\_master* na stronie Emotiv; utworzono również konto EmotivID. Uzyskane w ten sposób dane umożliwiają autoryzację w Cortex API, pozwalając tym sa-

mym na otrzymywanie wyników wystosowanych zapytań.

Podczas uruchomienia aplikacji nawiązywane jest szyfrowane połączenie wss (ang. WebSocket Secure). Komunikacja odbywa się z wykorzystaniem protokołu JSON-RPC 2.0, w którym każde zapytanie składa się z pól definiujących wersję JSON-RPC (*jsonrpc*), nazwę wywoływanej metody (*method*), jej parametrów (*params*, o ile wymagane) oraz identyfikatora zapytania (*id*). Otrzymany wynik zawiera specyfikację wersji JSON-RPC (*jsonrpc*), wynik (*result*) lub błąd (*error*), oraz identyfikator zapytania (*id*). Przykładowe zapytanie zostało pokazane na Kodzie źródłowym 3.1; przykładowa odpowiedź na nie znajduje się na Kodzie źródłowym 3.2.

### Kod źródłowy 3.1. Przykładowe zapytanie Cortex API

Źródło: [11]

```
1  {
2      "id": 1,
3      "jsonrpc": "2.0",
4      "method": "authorize",
5      "params": {
6          "clientId": "xxx",
7          "clientSecret": "yyy"
8      }
9 }
```

### Kod źródłowy 3.2. Przykładowy wynik zapytania Cortex API

Źródło: [11]

```
1  {
2      "id": 1,
3      "jsonrpc": "2.0",
4      "result": {
5          "cortexToken": "zzz",
6          "warning": {
7              "code": 6,
8              "message": "...",
9              "licenseUrl": "https://..."
10         }
11     }
12 }
```

W celu minimalizacji liczby błędów komunikacji postanowiono zaimplementować kompletną bibliotekę zapytań oraz ich wyników. Opracowano w tym celu klasy Request, Response oraz klasy reprezentujące wszystkie możliwe zapytania, ich parametry oraz wyniki.

Klasa Request, pokazana na Kodzie źródłowym 3.3 na następnej stronie, staje się klasą bazową dla wszystkich zapytań Cortex API. Klasy zapytań, dziedzicząc po niej, uzyskują dostęp do zawartych w niej właściwości. Dzięki temu zyskują mechanizm automatycznego przydzielania *id* zapytania, którego niepowtarzalność jest kluczowa do po-

prawnej identyfikacji odebranych wyników; uzyskują również automatyczne przypisanie wersji protokołu JSON-RPC, która jest stała w obrębie Cortex API. Dodatkowo, dzięki zastosowaniu słowa kluczowego `abstract`, klasa bazowa wymusza redefinicję właściwości `method`, która jest unikalna dla każdego rodzaju zapytania.

### Kod źródłowy 3.3. Kod klasy Request

Źródło: Opracowanie własne

```
1 public abstract class Request : IRequest
2 {
3     private static int _id;
4
5     public int id { get; } = _id++;
6     public string jsonrpc { get; } = "2.0";
7     public abstract string method { get; }
8 }
```

Przykładem klasy dziedziczącej po `Request` może być `AuthorizeRequest`. Jej kod został przedstawiony na Kodzie źródłowym 3.4. Oprócz nadpisania w wierszu 3 właściwości `method` wartością odpowiednią dla swojego zapytania, wprowadza ona właściwość `@params4` typu `AuthorizeParameter`.

### Kod źródłowy 3.4. Kod klasy AuthorizeRequest

Źródło: Opracowanie własne

```
1 public class AuthorizeRequest : Request
2 {
3     public override string method { get; } = "authorize";
4     public AuthorizeParameter @params { get; }
5
6     public AuthorizeRequest(AuthorizeParameter parameter)
7     {
8         @params = parameter;
9     }
10 }
```

Kod klasy `AuthorizeParameter` został pokazany na Kodzie źródłowym 3.5 na sąsiedniej stronie. Poprzez swoją sygnaturę konstruktora, zawartą w wierszu 8, wymusza ona przekazanie parametrów, które są wymagane dla danego zapytania (`clientId` oraz `clientSecret`) na etapie tworzenia obiektu; parametry opcjonalne (`license` oraz `debit`) nie wymagają przekazania przez konstruktor i mogą pozostać nieprzydzielone.

<sup>4</sup>W języku C# `params` jest słowem kluczowym i nie może być użyte jako identyfikator w programie, o ile nie zostanie poprzedzone prefiksem `@`.

### Kod źródłowy 3.5. Kod klasy AuthorizeParameter

Źródło: Opracowanie własne

```
1 public class AuthorizeParameter
2 {
3     public string clientId { get; }
4     public string clientSecret { get; }
5     public string license { get; set; }
6     public int? debit { get; set; }
7
8     public AuthorizeParameter(string clientId, string clientSecret)
9     {
10         this.clientId = clientId;
11         this.clientSecret = clientSecret;
12     }
13 }
```

Kod klasy Response został pokazany na Kodzie źródłowym 3.6. Posiada on definicje wszystkich pól definiowanych przez standard JSON-RPC. W celu utworzenia jej instancji należy poprzez parametr generyczny T sprecyzować jej typ, który musi implementować interfejs IResult.

### Kod źródłowy 3.6. Kod klasy Response

Źródło: Opracowanie własne

```
1 public class Response<T> where T : IResult
2 {
3     public int id { get; }
4     public string jsonrpc { get; }
5     public T result { get; }
6
7     [JsonConstructor]
8     private Response(int id, string jsonrpc, T result)
9     {
10         this.id = id;
11         this.jsonrpc = jsonrpc;
12         this.result = result;
13     }
14 }
```

Przykładem implementacji interfejsu IResult jest klasa AuthorizeResponse, pokazana na Kodzie źródłowym 3.7 na następnej stronie.

Wysłanie zapytania odbywa się przy użyciu metody SendRequest przedstawionej na Kodzie źródłowym 3.8. Jako parametr przyjmuje ona zapytanie implementujące interfejs IRequest, a więc wszystkie zapytania dziedziczące po klasie Request. Wartością zwracaną jest Response<T>, gdzie T jest typem generycznym implementującym interfejs IResult. W ciele tej metody odbywa się kojarzenie uzyskanych odpowiedzi z wysłanymi zapytaniami. W linii 13 następuje zasubskrybowanie wydarzenia (ang. event) OnResponse, dzięki czemu każdorazowe jego zgłoszenie spowoduje wywołanie lokalnej funkcji WaitResponseMatch, w której, jeżeli id zapytania oraz otrzymanej odpowiedzi są takie same, następuje odsubskrybowanie rzeczonej funkcji oraz przydzielenie otrzymanej odpowiedzi do wyniku zapytania. Ponieważ metoda SendRequest jest asynchroniczna, nie blokuje ona interfejsu użytkownika w trakcie oczekiwania na odpowiedź.

### Kod źródłowy 3.7. Kod klasy AuthorizeResponse

Źródło: Opracowanie własne

```
1  public class AuthorizeResponse : IResult
2  {
3      public string cortexToken { get; }
4      public object warning { get; }
5
6      [JsonConstructor]
7      private AuthorizeResponse(string cortexToken, object warning)
8      {
9          this.cortexToken = cortexToken;
10         this.warning = warning;
11     }
12 }
```

### Kod źródłowy 3.8. Kod metody SendRequest

Źródło: Opracowanie własne

```
1  public async Task<Response<T>> SendRequest<T>(IRequest request) where T : IResult
2  {
3      var tcs = new TaskCompletionSource<Response<T>>();
4
5      void WaitResponseMatch(string response)
6      {
7          if (Parser.GetTokenAsString(response, "id") != request.id.ToString())
8              return;
9
10         OnResponse -= WaitResponseMatch; //unsubscribe after match
11         tcs.SetResult(Parser.Deserialize<Response<T>>(response));
12     }
13     OnResponse += WaitResponseMatch;
14     SendRequest(request);
15
16     return await tcs.Task;
17 }
```

Na przykład, chcąc uzyskać Cortex Token, który jest niezbędny do autoryzacji większości zapytań Cortex API, należy wywołać metodę SendRequest, przekazując do niej obiekt klasy AuthorizeRequest z parametrem typu AuthorizeParameter. Z uzyskanej odpowiedzi typu Response<AuthorizeResponse> możliwe jest odczytanie cortexToken. Operacje te, zawarte w metodzie Authorize, pokazano na Kodzie źródłowym 3.9. Wartością "xxx" oznaczono clientId, a "yyy" clientSecret.

### Kod źródłowy 3.9. Kod metody Authorize

Źródło: Opracowanie własne

```
1 public async Task Authorize()
2 {
3     var authorizeResponse = SendRequest<AuthorizeResponse>(new
4         ↳ AuthorizeRequest(new AuthorizeParameter("xxx", "yyy")));
5     CortexToken = (await authorizeResponse).result?.cortexToken;
6 }
```

Wykorzystując wewnętrzny mechanizm serializacji zapytanie jest przekształcane do postaci pokazanej na Kodzie źródłowym 3.10, a następnie wysyłane przy użyciu protokołu WebSocket. Odpowiedź na to zapytanie została pokazana na Kodzie źródłowym 3.11. Obie wiadomości są tożsame z przykładami zaczerpniętymi z dokumentacji, pokazanymi na Kodach źródłowych 3.1 na stronie 49 oraz 3.2 na stronie 49.

### Kod źródłowy 3.10. Zserializowane zapytanie AuthorizeRequest

Źródło: Opracowanie własne

```
1 {
2     "method": "authorize",
3     "params": {
4         "clientId": "xxx",
5         "clientSecret": "yyy"
6     },
7     "id": 1,
8     "jsonrpc": "2.0"
9 }
```

### Kod źródłowy 3.11. Odpowiedź otrzymana na zapytanie AuthorizeRequest

Źródło: Opracowanie własne

```
1 {
2     "id": 1,
3     "jsonrpc": "2.0",
4     "result": {
5         "cortexToken": "zzz"
6     }
7 }
```

### 3.3.5. Proces decyzyjny

Proces decyzyjny, obejmujący zmianę selekcji aktualnie wybranego klawisza oraz jego wciśnięcie (akceptację), odbywa się w oparciu o dane otrzymane od Cortex API. Po otwarciu sesji następuje zasubskrybowanie trzech strumieni danych: informacyjnego, komend mentalnych oraz mimiki. Subskrypcja powoduje przesyłanie w regularnych odstępach czasu obiektów w postaci formatu JSON. Zawartość wiadomości oraz jej częstotliwość zależą od rodzaju strumienia danych. Częstotliwość próbkowania dla strumienia informacyjnego, komend mentalnych oraz mimiki wynosi kolejno 128 Hz, 8 Hz oraz 32 Hz [11].

O ile strumień informacyjny nie jest bezpośrednio wykorzystywany do podejmowania decyzji, tak informacje w nim zawarte są niezwykle istotne dla użytkownika urządzenia. Przesłane paczki zawierają dane na temat stanu hełmu: poziom naładowania jego baterii oraz jakość (siła) sygnału z poszczególnych czujników. Im lepsza łączność elektrod ze skórą głowy, tym wyniki pozostałych strumieni można uznać za bardziej miarodajne.

Dane strumieni komend mentalnych oraz mimiki mają zbliżony format, dlatego zostaną omówione na przykładzie wyłącznie jednej komendy – wyobrażenia ruchu w prawo. Pozostałe komendy, czyli wyobrażenie ruchu w dół, w lewo, w górę oraz zmarszczenie brwi, zostały zaimplementowane w analogiczny sposób.

Przy stworzeniu instancji klasy KeyboardNavigator (patrz Kod źródłowy 3.12) następuje przekazanie do niej w konstruktorze obiektów Insight oraz Settings; referencje te zostają zapisane w klasie. Za sprawą interfejsu INotifyPropertyChanged, implementowanego przez obie te klasy, następuje subskrypcja zmian wartości ich właściwości. W konstruktorze następuje również ustawienie wartości czasomierza (ang. timer) na wartość sprecyzowaną w zakładce Settings oraz zasubskrybowanie wydarzenia informującego o upłynięciu odliczanego przez niego czasu.

#### Kod źródłowy 3.12. Kod konstruktora klasy KeyboardNavigator

Źródło: Opracowanie własne

```
1 public KeyboardNavigator(Insight insight, Settings settings)
2 {
3     Insight = insight;
4     Settings = settings;
5
6     Insight.PropertyChanged += Insight_PropertyChanged;
7     Settings.PropertyChanged += Settings_PropertyChanged;
8
9     _rightTimer.Interval = TimeSpan.FromMilliseconds(Settings.FocusTime);
10    _rightTimer.Tick += _rightTimer_Tick;
11 }
```

W momencie zgłoszenia wydarzenia PropertyChanged przez obiekt Insight następuje wywołanie metody Insight\_PropertyChanged, pokazanej na Kodzie źródłowym 3.13 na sąsiedniej stronie. Metoda ta sprawdza jaka wartość została zmodyfikowana; jeżeli jest to siła skupienia na komendzie *w prawo*, sprawdzane jest czy został przekroczony próg jej aktywacji, definiowany przez użytkownika w zakładce Settings. Jeżeli tak –

następuje włączenie czasomierza \_rightTimer; jeżeli nie – timer jest wyłączany.

### Kod źródłowy 3.13. Kod metody Insight\_PropertyChanged

Źródło: Opracowanie własne

```
1  private void Insight_PropertyChanged(object sender, PropertyChangedEventArgs e)
2  {
3      if (e.PropertyName == "RightLevel" && Insight.RightLevel >=
4          Settings.RightThreshold)
5          _rightTimer.Start();
6      else if (e.PropertyName == "RightLevel")
7          _rightTimer.Stop();
}
```

Załączenie czasomierza, zgodnie z subskrypcją złożoną w Kodzie źródłowym 3.12 na poprzedniej stronie w linii 10, powoduje wywołanie metody \_rightTimer\_Tick co każde upłynięcie jego okresu, który, podobnie jak próg aktywacji poszczególnych komend, jest definiowany przez użytkownika w zakładce *Settings*. Wywołana wtedy metoda, pokazana na Kodzie źródłowym 3.14, zgłasza wydarzenie OnRightCommand, które odpowiada za przesunięcie selekcji na klawiaturze o jeden klawisz w prawo.

### Kod źródłowy 3.14. Kod metody \_rightTimer\_Tick

Źródło: Opracowanie własne

```
1  private void _rightTimer_Tick(object sender, EventArgs e) =>
2      OnRightCommand?.Invoke();
```

W celu umożliwienia edycji ustawień w trakcie działania aplikacji (a nie wyłącznie na etapie jej uruchomienia) śledzone są również zmiany w klasie *Settings*. Każdorazowa zmiana właściwości określającej częstotliwość zgłaszania komend powoduje uaktualnienie wartości przypisanej do timera. Odpowiada za to metoda *Settings\_PropertyChanged*, pokazana na Kodzie źródłowym 3.15.

### Kod źródłowy 3.15. Kod metody Settings\_PropertyChanged

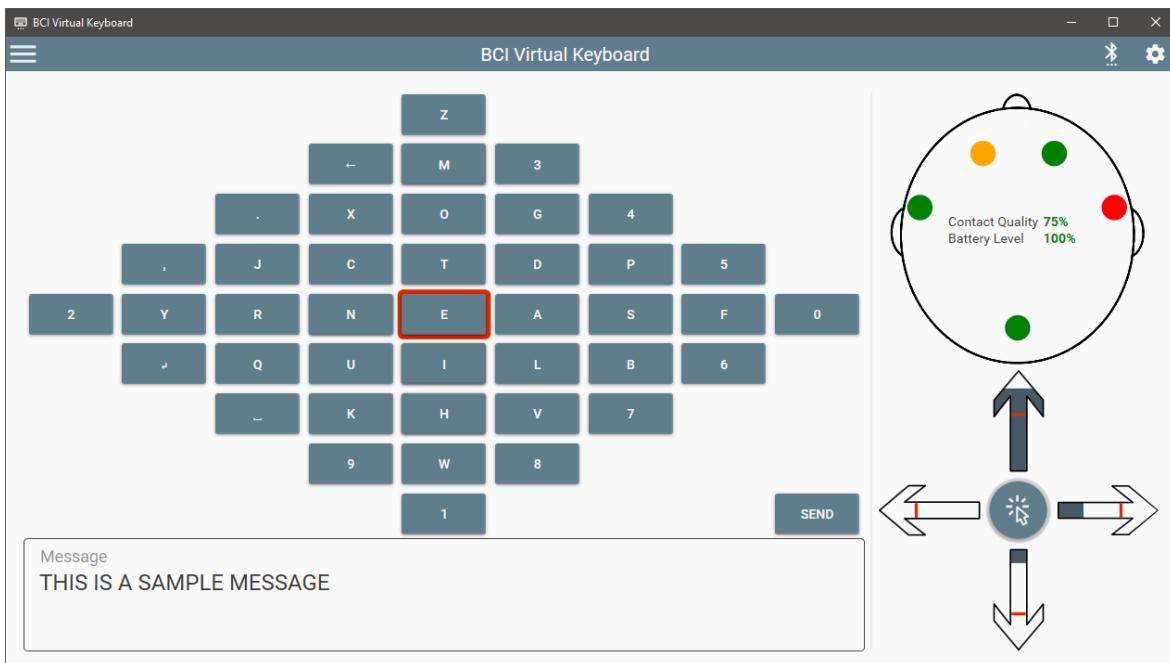
Źródło: Opracowanie własne

```
1  private void Settings_PropertyChanged(object sender,
2      System.ComponentModel.PropertyChangedEventArgs e)
3  {
4      if (e.PropertyName != "FocusTime")
5          return;
6
7      _rightTimer.Interval = TimeSpan.FromMilliseconds(Settings.FocusTime);
}
```

## 3.4. Omówienie interfejsu użytkownika

### 3.4.1. Zakładka Virtual Keyboard

Widok zakładki *Virtual Keyboard* został pokazany na Rysunku 3.4.



Rysunek 3.4. Widok zakładki *Virtual Keyboard*

Źródło: Opracowanie własne

*Virtual Keyboard* jest domyślnym widokiem wyświetlonym po uruchomieniu aplikacji. Można w nim wydzielić cztery omówione poniżej segmenty.

**Moduł klawiatury** — znajdująca się w lewym górnym rogu aplikacji klawiatura składa się z 42 klawiszy: liter A–Z, cyfr 0–9, znaków spacji, enter (przejście do nowej linii), przecinka, kropki, *backspace* oraz znajdującego się w jej prawym dolnym rogu klawisza *SEND*. Aktualnie wybrany klawisz jest oznaczany czerwoną ramką.

Układ klawiatury jest rozwiązaniem autorskim, mającym na celu minimalizację liczby ruchów potrzebnych do wprowadzenia wiadomości. Bazuje on na częstotliwości występowania liter w alfabetie angielskim, którą przyjął Samuel Morse opracowując swój alfabet [48]. Przyjętym przez autora priorytetem było (1) minimalizacja liczby ruchów oraz (2) minimalizacja liczby zakrętów koniecznych do zaznaczenia danej litery, zakładając że poruszamy się ze środka klawiatury (litery *E*).

**Moduł okna wiadomości** — znajduje się pod klawiaturą. Wyświetla tekst aktualnie wprowadzanej wiadomości. W momencie wyboru znaku, jest on dodawany do wiadomości. Po wyborze klawisza *SEND* następuje odczytanie tekstu wiadomości z użyciem syntezatora mowy, a następnie jego wyczyszczenie.

**Moduł informacyjny** — znajduje się w prawym górnym rogu aplikacji. Dostarcza w czasie rzeczywistym informacji na temat stanu Emotiv Insight: poziomu naładowania, jakości połączenia poszczególnych czujników oraz średniej jakości połączenia *Contact Quality*. Dzięki schematycznemu przedstawieniu czujników na diagramie głowy, możliwe jest łatwe zlokalizowanie czujników o niskiej jakości połączenia, a następnie korekta ich powierzchni styku. W zależności od aktualnej jakości połączenia, kolor czujników będzie się zmieniał z czarnego przez czerwony, pomarańczowy aż do zielonego dla jakości połączenia kolejno bardzo złej, złej, dobrej oraz bardzo dobrej.

**Moduł pada kierunkowego** — pad kierunkowy (ang. directional pad, d-pad) znajduje się w prawym dolnym rogu aplikacji. Składa się on z czterech strzałek kierunkowych oraz przycisku znajdującego się między nimi.

Poziom wypełnienia strzałki reprezentuje siłę skupienia na odpowiadającej mu komendzie. Czerwoną linią został oznaczony próg aktywacji komendy, po przekroczeniu którego następuje wykonanie powiązanej akcji. Użytkownik może edytować próg każdej z komend w zakładce *Settings*.

Przycisk pomiędzy strzałkami odpowiada za *wciśnięcie* (selekcję) aktualnie oznaczonego przycisku. Podobnie jak strzałki kierunkowe, threshold przycisku jest możliwy do edycji w zakładce *Settings*. Poziom pewności rozpoznania komendy selekcji jest sygnalizowany użytkownikowi przez wypełnianie okrągłego paska postępu znajdującego się wokół przycisku.

### 3.4.2. Zakładka *Messenger*

Zakładka *Messenger*, pokazana na Rysunku 3.5 na następnej stronie, umożliwia bezpośrednią wymianę wiadomości z Cortex API.

W górnej części okna znajduje się historia, zawierająca zarówno wysłane jak i odebrane wiadomości. Wiadomości dodawane są w kolejności chronologicznej zgodnie z następującym formatem:

*data    czas    typ :*  
*tekst*

gdzie:

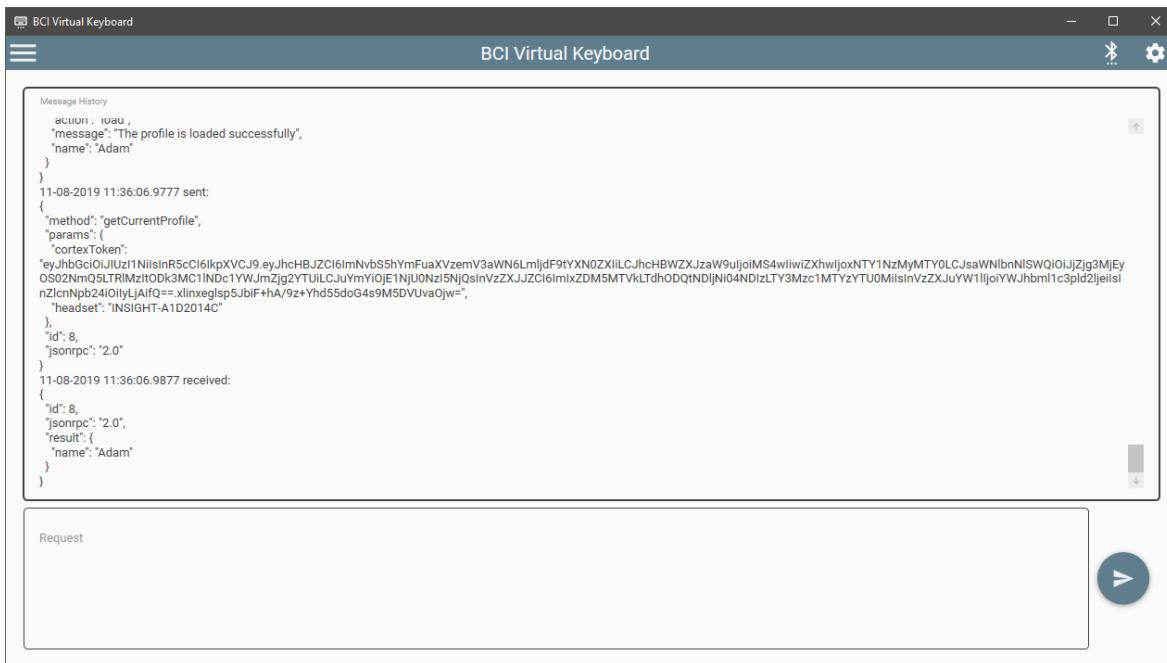
*data* – data wysłania/odebrania wiadomości w formacie DD-MM-RRRR

*czas* – czas wysłania/odebrania wiadomości w formacie HH-MM-SS.FFFF

*typ* – typ wiadomości; sent lub received dla kolejno wysłanej lub odebranej

*tekst* – zawartość wiadomości

W oknie historii wyświetlane są zarówno wiadomości wysłane bezpośrednio przez użytkownika, jak i wszystkie wysłane podczas przez oprogramowanie podczas pracy aplikacji.



**Rysunek 3.5.** Widok zakładki *Messenger*

Źródło: Opracowanie własne

W dolnej części okna znajduje się pole do wpisywania wiadomości. Obok pola znajduje się przycisk służący do wysłania aktualnie wprowadzonego tekstu.

### 3.4.3. Zakładka *About*

Widok zakładki *About* został pokazany na Rysunku 3.6 na sąsiedniej stronie.

Zawiera ona sekcję *About*, zawierającą informacje na temat aplikacji: czym jest wirtualna klawiatura oraz czego używa do umożliwienia użytkownikowi interakcji z nią, *Author*, przedstawiającą autora aplikacji oraz *Get In Touch*, zawierającą odnośniki umożliwiające kontakt z autorem; są to odnośniki do strony na GitHubie autora oraz adresu e-mail.

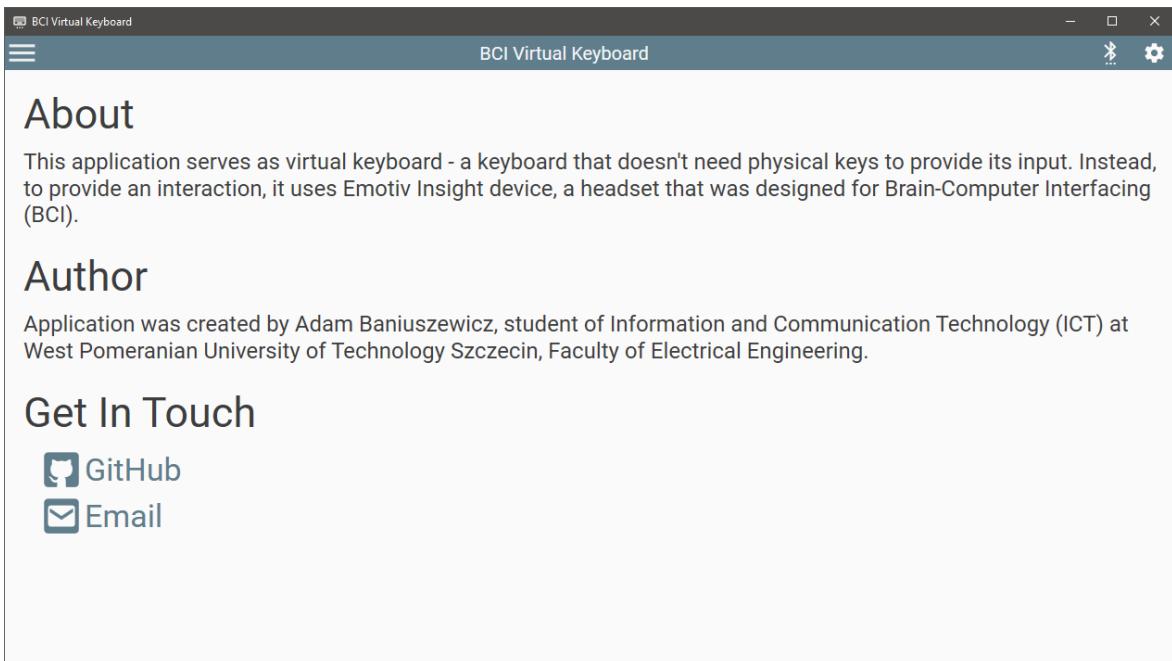
### 3.4.4. Zakładka *Connection*

Zakładka *Connection* służy do definiowania ustawień połączenia z Cortex API. Jej widok przedstawiono na Rysunku 3.7 na następnej stronie.

Zawiera ona dwa pola tekstowe pozwalające na wprowadzenie *Client ID* oraz *Client Secret*, kluczy unikalnych dla opracowanej przez autora aplikacji. Dodatkowo zawiera dwie listy rozwijalne służące do wyboru hełmu oraz profilu użytkownika.

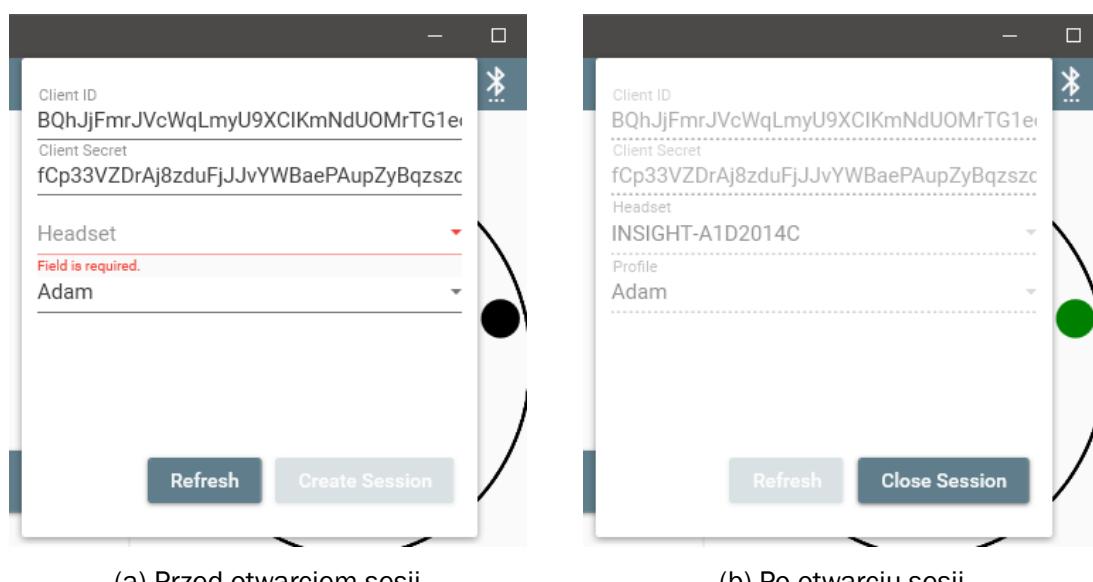
Pola w aplikacji posiadają weryfikację wprowadzonych danych; w przypadku wprowadzenia niepoprawnych wartości pole zostanie podkreślone na czerwono, a poniżej niego pojawi się komunikat błędu. Przykład działania tego mechanizmu pokazano na Rysunku 3.7a na sąsiedniej stronie w polu *Headset*.

Przycisk *Refresh* pozwala na odpytanie Cortex API na temat znajdujących się w po-



**Rysunek 3.6.** Widok zakładki *About*

Źródło: Opracowanie własne



**Rysunek 3.7.** Widok zakładki *Connection*

Źródło: Opracowanie własne

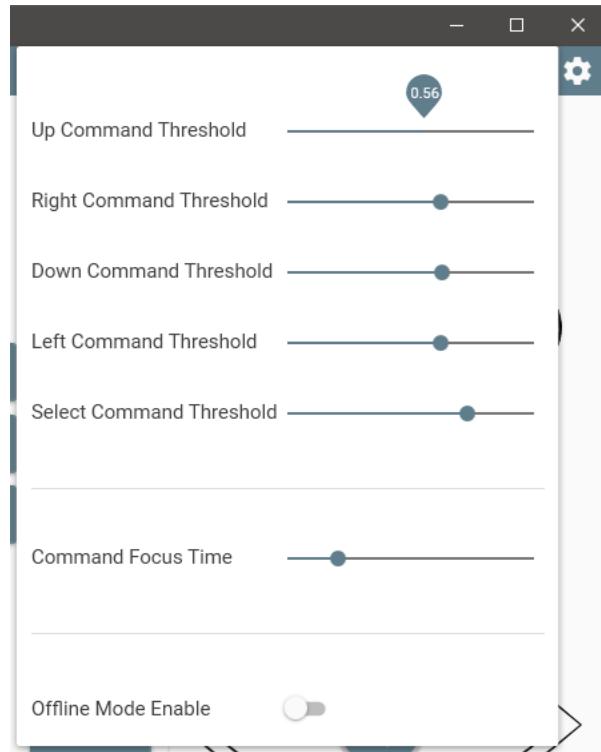
bliży hełmów Emotiv oraz dostępnych profili. Po kliknięciu tego przycisku uzyskane dane są możliwe do wyboru z list rozwijalnych *Headset* oraz *Profile*.

Wykonanie operacji *Create Session* jest możliwe wyłącznie w przypadku poprawnego sprecyzowania wszystkich dostępnych pól. Kliknięcie tego przycisku spowoduje otwarcie sesji z wybranym hełmem oraz zalogowanie na odpowiednim profilu. Po otwarciu sesji wszystkie kontrolki w tym widoku, za wyjątkiem przycisku służącego do zamknięcia sesji,

stają się nieaktywne do momentu zakończenia sesji. Widok okna z otwartą sesją pokazano na Rysunku 3.7b na poprzedniej stronie.

### 3.4.5. Zakładka **Settings**

Zakładka *Settings* umożliwia użytkownikowi zmianę parametrów algorytmu aplikacji; jej widok pokazano na Rysunku 3.8.



**Rysunek 3.8.** Widok zakładki *Settings*

Źródło: Opracowanie własne

Przy użyciu zawartych w niej suwaków użytkownik może zmieniać próg aktywacji poszczególnych komend mentalnych. Wartość progu aktywacji wyrażona jest liczbą z zakresu [0..1], przy czym 0 oznacza bardzo niski próg, a 1 bardzo wysoki.

Zakładka ta umożliwia dodatkowo zmianę parametru *Command Focus Time*, definiującego czas przez który użytkownik musi skupić się na komendzie w poziomie przekraczającym jej threshold, aby powiązana z nią komenda została wykonana. Wartość czasu aktywacji wyrażona jest w milisekundach i może przyjmować wartości z zakresu [1..2500].

Aktualna wartość definiowana przy pomocy suwaka jest widoczna po kliknięciu w niego; zostaje wtedy wyświetlony dymek z jego aktualną wartością. Mechanizm ten został pokazany na Rysunku 3.8 przy polu *Up Command Threshold*.

Na dole zakładki znajduje się przełącznik pozwalający na włączenie tak zwanego *Offline Mode*. W tym trybie na oknie *Virtual Keyboard* widoczne są suwaki, pozwalające na ręczną symulację aktualnego skupienia dla poszczególnych komend.

## **ROZDZIAŁ 4**

# **Badania opracowanego systemu**

## **4.1. Wpływ ilości komend oraz ilości sesji treningowych na poprawność detekcji**

Trening komend mentalnych, detekcji mimiki oraz badanie poprawności ich wykrywania wykonano w oprogramowaniu EmotivBCI.

W przypadku komend mentalnych zastosowano inną strategię niż początkowo zakładano: autor, zamiast skupiać się na wyobrażeniu ruchu w górę, w prawo, w dół lub w lewo, powiązał te komendy z innymi, bardziej abstrakcyjnymi wyobrażeniami. Było to spowodowane niemożnością osiągnięcia wystarczającej siły komendy w przypadku początkowo zakładanej strategii. Kierując się wskazówkami zawartymi w dokumentacji EmotivBCI, akcje komend mentalnych zostały powiązane z wyobrażeniami angażującymi różne części mózgu. I tak, kolejno dla *uniesienia*, *przesunięcia w prawo*, *opuszczenia* oraz *przesunięcia w lewo*, jest to wyobrażenie ruchu języka, wyobrażenie ruchu prawego kciuka, liczenie w głowie co 3 (3, 6, 9, 12, ...) oraz przywołanie w pamięci wspomnienia z gry komputerowej.

Każdą komendę trenowano dziesięciokrotnie. Trening komend poprzedzało dziesięciokrotne wytrenowanie stanu *neutralnego*, zarówno dla komend mentalnych jak i detekcji mimiki. Komendy mentalne trenowano w kolejności *uniesienie*, *przesunięcie w prawo*, *opuszczenie*, *przesunięcie w lewo*, przy czym każdą z komend trenowano najpierw dziesięciokrotnie, a dopiero później rozpoczynano trening kolejnej.

Badaniu podlegała poprawność detekcji komendy podczas jej wyobrażenia lub wykonania, oraz jej korelacja z niepożądaną detekcją w trakcie utrzymywania stanu *neutralnego*<sup>1</sup>. Badanie wykonywano po każdym treningu. Składało się ono z dwudziestu piętnastosekundowych prób, z których dziesięć było skupieniem się na komendzie, a dziesięć zachowywaniem stanu *neutralnego*. Skupienie się oraz zachowywanie stanu *neutralnego*

---

<sup>1</sup>Stan neutralny – stan, w którym nie następuje skupienie na poleceniu lub wykonanie akcji; rozluźnienie, odprężenie.

było wykonywane na zmianę.

W przypadku próby skupienia (S) komenda była uznawana za wykonaną poprawnie jeżeli w trakcie jej wyobrażenia nastąpił ruch sześcianu treningowego do odpowiadającej komendzie strefy (patrz Rysunek 4.1). W przypadku detekcji mimiki komenda była zaliczana jeżeli nastąpiło powtórzenie jej przez awatara (patrz Rysunek 3.3d na stronie 48). Jeżeli w ciągu piętnastosekundowej próby komenda nie została wykonana lub została wykonana inna komenda niż oczekiwana, próba nie była zaliczana. Próba braku skupienia (NS) była uznawana za zaliczoną jeżeli w jej trakcie nie nastąpiło wykonanie żadnej akcji, która byłaby definiowana jako zaliczona próba skupienia.



**Rysunek 4.1.** Sposób uznawania komendy mentalnej; jeżeli sześciyan wychylił się co najmniej na oznaczone (przezroczyste) pozycje, komenda była uznawana za zaliczoną

Źródło: Opracowanie własne

Uzyskane wyniki zestawiono w Tabeli 4.1 na sąsiedniej stronie oraz przedstawiono na Rysunku 4.2 na stronie 65.

W odczuciu autora trening pierwszej komendy (patrz Rysunek 4.2a na stronie 65) był najbardziej satysfakcjonujący; jego rezultat – poruszenie się sześcianu treningowego – był możliwy do uzyskania z dwudziestoprocentową skutecznością już od pierwszej sesji treningowej. Przy sesji drugiej oraz trzeciej skuteczność spadła do dziesięciu procent. Autor zaobserwował wtedy znacznie większe *drganie*<sup>2</sup> sześcianu podczas skupiania się na komendzie, co nie miało miejsca w przypadku pierwszej sesji, w której, jeżeli nastąpił jakaś ruch sześcianu, to było to przemieszczenie aż do samej krawędzi ekranu. Przy sesjach od piątej do ósmej, obok bardzo silnego wzrostu poziomu prawidłowych detekcji podczas skupienia, nastąpił również wzrost błędnych detekcji podczas utrzymywania stanu neutralnego. Sesja dziewiąta wydaje się być najbardziej optymalnym wynikiem osiągniętym dla komendy *uniesienia*: podczas niej skuteczność detekcji w trakcie skupienia wyniosła osiemdziesiąt procent przy jedynych trzydziestu procent podczas braku skupienia. Maksymalny stosunek poprawnych do złych/braku detekcji wyniósł dziewięćdziesiąt

<sup>2</sup>Poprzez *drganie* sześcianu autor ma na myśli jego przemieszczenie się w pożądany kierunek, jednak bez przekroczenia progu definiowanego jako zaliczenie komendy. Takie wychylenia najczęściej następowały kilkukrotnie, jedno po drugim.

**Tabela 4.1.** Wyniki treningu uzyskane w programie EmotivBCI  
 Źródło: Opracowanie własne

Numer treningu	Komenda																				
	Uniesienie				Przesunięcie w prawo				Opuszczenie				Przesunięcie w lewo				Zmarszczenie brwi				
	S +	S -	NS +	NS -	S +	S -	NS +	NS -	S +	S -	NS +	NS -	S +	S -	NS +	NS -	S +	S -	NS +	NS -	
1	2	8	10	0	0	10	6	4	0	10	10	0	0	10	10	0	10	0	10	3	7
2	1	9	10	0	0	10	8	2	0	10	10	0	0	10	10	0	10	0	10	4	6
3	1	9	10	0	0	10	10	0	0	10	10	0	0	10	10	0	10	0	10	4	6
4	8	2	6	4	0	10	10	0	1	9	8	2	1	9	10	0	10	0	10	4	6
5	6	4	8	2	1	9	10	0	1	9	9	1	1	9	10	0	10	0	10	6	4
6	8	2	5	5	0	10	10	0	4	6	9	1	2	8	9	1	10	0	10	5	5
7	9	1	4	6	0	10	10	0	2	8	10	0	1	9	10	0	10	0	10	0	6
8	9	1	3	7	1	9	10	0	2	8	10	0	3	7	10	0	10	0	10	4	6
9	8	2	7	3	2	8	10	0	3	7	10	0	0	2	8	10	0	10	0	10	5
10	9	1	5	5	1	9	10	0	3	7	10	0	3	7	10	0	10	0	10	0	5

S+ Podczas skupienia na komendzie została ona wykonana (więcej = lepiej)

S- Podczas skupienia na komendzie nie została ona wykonana lub została wykonana inną komendą (mniej = lepiej)

NS+ Podczas braku skupienia na komendzie (stan neutralny) nie została wykonana komenda (więcej = lepiej)

NS- Podczas braku skupienia na komendzie (stan neutralny) została wykonana komenda (mniej = lepiej)

procent i został osiągnięty w sesji siódmej, ósmej oraz dziesiątej, przy czym spośród nich ostatnia sesja charakteryzowała się najniższym błędem podczas braku skupienia wynoszącym pięćdziesiąt procent.

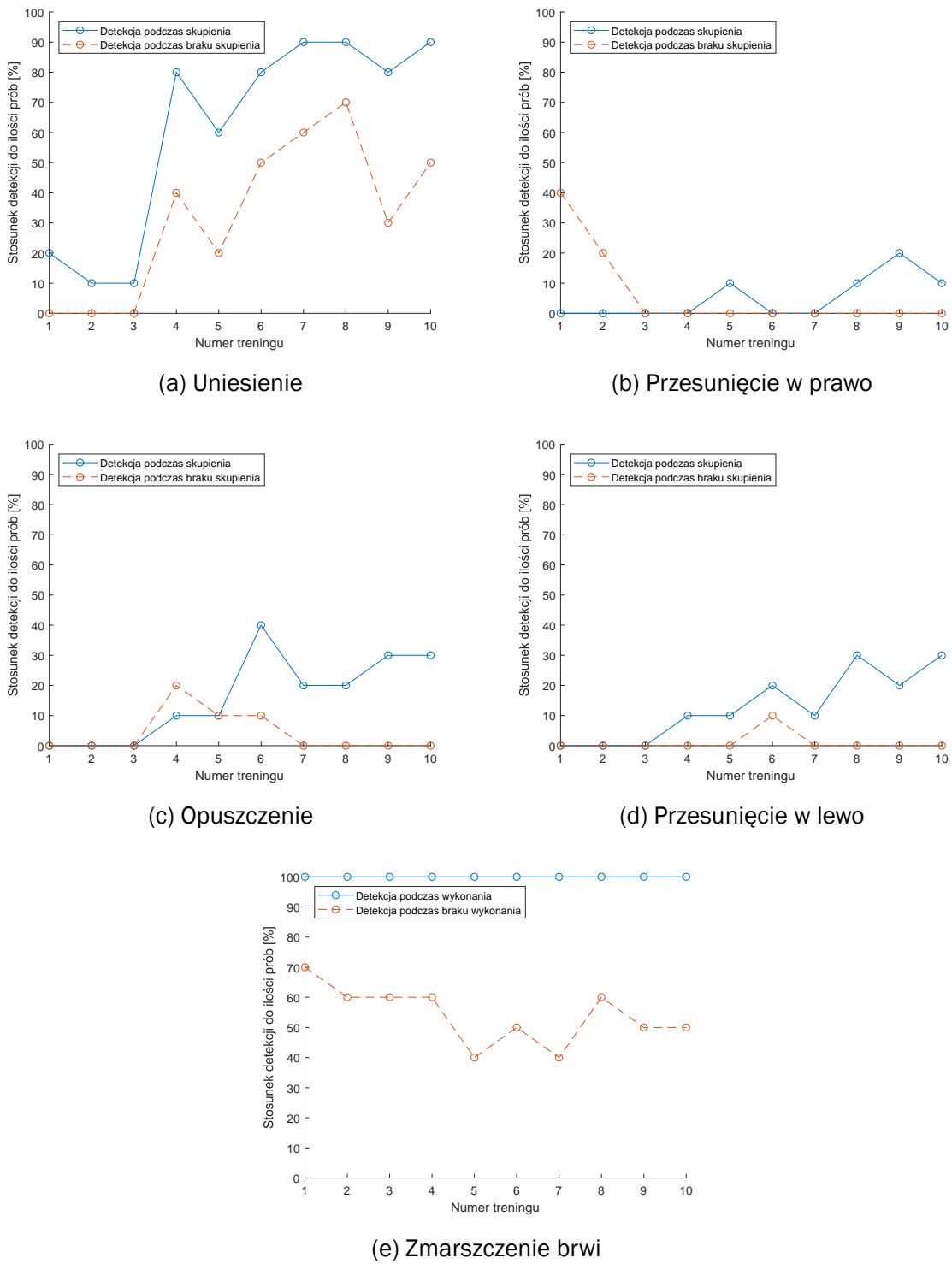
Wyniki uzyskane podczas treningu drugiej komendy pokazano na Rysunku 4.2b na następnej stronie. Detekcja podczas skupienia na komendzie osiąga zauważalnie gorsze wyniki niż podczas treningu pierwszej komendy. Przez pierwsze cztery sesje treningowe autor nie był w stanie sprawić, aby sześciian wychylił się na tyle, by zostało to uznane jako zaliczona próba. Przy sesji piątej poprawność detekcji wyniosła jedynie dziesięć procent, jednak podczas sesji szóstej i siódmej spadła ona z powrotem do zera procent, co pozwala sądzić, iż detekcja podczas sesja piątej mogła być błędnie zmierzona. Od sesji ósmej do dziesiątej następuje liniowy wzrost poprawności detekcji podczas skupienia do poziomu dwudziestu procent, co stanowi maksimum osiągnięte dla tej komendy. Przy ostatniej sesji stosunek zaliczonych detekcji do ilości prób wyniósł dziesięć procent. Przy drugiej komendzie osiągnięto bardzo niski współczynnik detekcji podczas braku skupienia. Małejąc z czterdziestu procent po pierwszym treningu, osiągnął zero procent przy treningu trzecim i pozostał na tym pułapie przez wszystkie pozostałe treningi tej komendy.

Trzecia komenda związana była z innym rodzajem wyobrażenia niż dwie poprzednie. Nie angażowała ona kory ruchowej; autor spodziewał się więc, iż pozwoli ona na łatwiejszą detekcję polecenia, a tym samym na osiągnięcie lepszego rezultatu w stanie skupienia. Wyniki uzyskane dla tej komendy przedstawiono na Rysunku 4.2c na sąsiedniej stronie. Detekcja podczas skupienia na komendzie nie była osiągalna przez pierwsze trzy sesje treningowe, jednak podczas kolejnych dwóch usytuowała się ona na poziomie dziesięciu procent. Po szóstym treningu stosunek poprawnych detekcji do ilości prób wyniósł czterdzieści procent, co stanowiło wartość szczytową dla tej komendy. Sesje od siódmej do dziesiątej zapewniły kolejno dwadzieścia, dwadzieścia, trzydzieści oraz trzydzieści procent poprawnych detekcji. Detekcja podczas braku skupienia charakteryzowała się wartością niezerową jedynie podczas sesji od czwartej do szóstej. Jej maksymalna wartość wyniosła dwadzieścia procent. Charakterystyczna dla treningu tej komendy była łatwość, z jaką autor był w stanie poruszyć sześciianem w pożądanym kierunku, jednak bez możliwości utrzymania poziomu skupienia pozwalającego na wychylenie się aż do progu zaliczenia próby. Na uwagę zasługuje również fakt większej ilości uzyskanych komunikatów *Great training*<sup>3</sup> niż miało to miejsce w przypadku innych komend.

Podczas treningu ostatniej komendy mentalnej (patrz Rysunek 4.2d na następnej stronie) można było zaobserwować trend rosnący w przypadku detekcji podczas skupienia. Jej maksymalna skuteczność osiągnęła trzydzieści procent podczas ósmej oraz dziesiątej sesji treningowej. Detekcja podczas braku skupienia pozostała zerowa; wyjątkiem była sesji szósta, w której osiągnęła dziesięć procent.

Wyniki poszczególnych sesji treningowych detekcji zmarszczenia brwi pokazano na Rysunku 4.2e na sąsiedniej stronie. Na pierwszy rzut oka widać znaczną przewagę detekcji mimiki nad detekcją komend mentalnych, przyjmując jako kryterium porównawcze

<sup>3</sup>Aplikacja EmotivBCI po zakończeniu sesji treningowej wyświetla ocenę jakości treningu w skali od zera do stu, definiującą spójność z poprzednimi sesjami [16]. Wartości powyżej siedemdziesięciu pięciu opatrzone są komunikatem *Great training*.



**Rysunek 4.2.** Wyniki treningu uzyskane w programie EmotivBCI

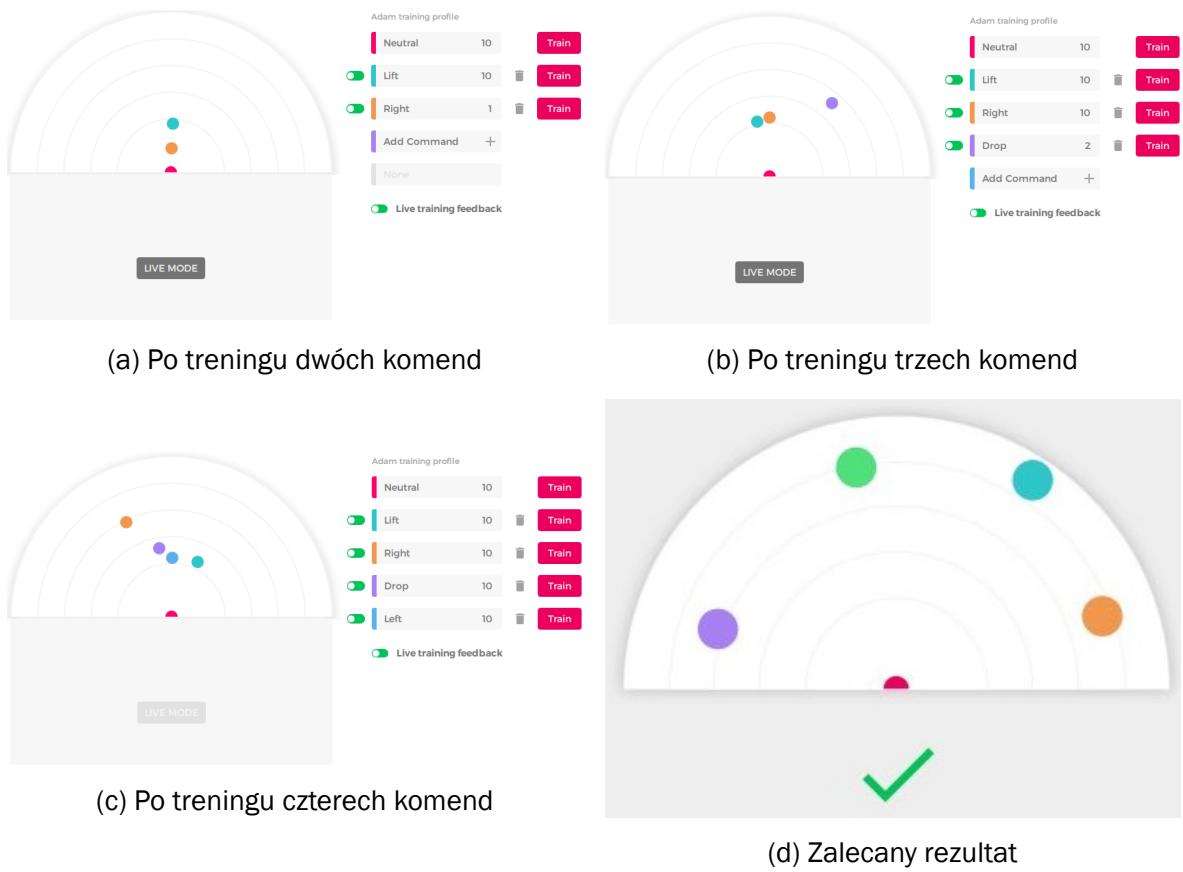
Źródło: Opracowanie własne

stosunek poprawnych detekcji w stanie skupienia/wykonania do odbytej ilości prób. Komendy mentalne, nie licząc *uniesienia*, nie osiągnęły poziomu nawet pięćdziesięciu procent, podczas kiedy zmarszczenie brwi charakteryzuje się stałym, niezależnym od liczby treningów, stuprocentowym poziomem detekcji. Wysoki poziom detekcji wiąże się jednak z wysoką stopą błędnych klasyfikacji podczas braku wykonania akcji. Kolejne sesje treningowe obniżają ją jednak z poziomu siedemdziesięciu procent dla pierwszej do pięćdziesięciu procent w przypadku ostatniej sesji. Autor zaobserwował, iż detekcje podczas braku skupienia następowały jedynie w przypadku wykonywania innych czynności, jak na przykład ruchu gałek ocznych czy wzięcia głębokiego oddechu, jednak postanowił nie ograniczać tych akcji, gdyż w jego opinii stanowić będą one naturalny proces użytkowania urządzenia.

Na podstawie opracowanych wyników można wysunąć wniosek, iż dodawanie kolejnych możliwych do detekcji komend mentalnych powoduje obniżenie stopy poprawności ich detekcji podczas skupienia. Odwrotna sytuacja ma miejsce w przypadku detekcji w stanie *neutralnym* – po dodaniu drugiej komendy i przeprowadzeniu trzech treningów, osiąga ona wartość zerową, przekraczając ją jedynie w przypadku czterech spośród dwudziestu czterech sesji treningowych dla trzech ostatnich komend mentalnych. W przypadku detekcji mimiki, która charakteryzowała się stuprocentową poprawnością detekcji podczas wykonania zmarszczenia brwi, kolejne treningi powodują obniżenie błędnych detekcji w przypadku stanu *neutralnego*.

Stosunkowa słaba detekcja komend mentalnych mogła być spowodowana nikłą separacją poszczególnych poleceń. Separację można oszacować na podstawie schematu przestrzeni mózgowej wygenerowanego przez program EmotivBCI, który pokazano na Rysunku 4.3 na następnej stronie. Należy zwrócić uwagę, iż izolacja poleceń nie jest stała i zmienia się w przypadku kolejnych treningów. Na podstawie Rysunków 4.3a na sąsiedniej stronie oraz 4.3b na następnej stronie można zauważyć, iż polecenia *uniesienia* oraz *przesunięcia w prawo* są początkowo oddalone siebie, po czym, po dodaniu polecenia *opuszczenia*, znacznie się zbliżają. Podobna sytuacja ma miejsce w przypadku polecenia *opuszczenia*, jak pokazano na Rysunku 4.3c na sąsiedniej stronie. Według producenta urządzenia pożądaną sytuacją jest jak największa separacja sygnałów, na przykład do poziomu pokazanego na Rysunku 4.3d na następnej stronie. Wiązać się ona będzie z łatwiejszym wywoływaniem poszczególnych komend [16].

## **4.2. Wpływ parametrów opracowanej aplikacji na poprawność detekcji**



**Rysunek 4.3.** Schemat przestrzeni mózgowej uzyskany w programie EmotivBCI

Źródło: a, b, c: Opracowanie własne, d: [16]

# **Zakończenie**

TODO

# Bibliografia

- [1] Bashir I. M.: A Brief Review of Brain Signal Monitoring Technologies for BCI Applications: Challenges and Prospects. *Journal of Bioengineering & Biomedical Science*, t. 04, nr 01, 2014, DOI: 10.4172/2155-9538.1000128.
- [2] Bashivan P., Rish I., Heisig S.: Mental state recognition via wearable EEG. *CoRR*, t. abs/1602.00985, 2016, URL: <http://arxiv.org/abs/1602.00985>.
- [3] Benabid A. L. i in.: Deep brain stimulation: BCI at large, where are we going to?, *Brain Machine Interfaces: Implications for Science, Clinical Practice and Society, Progress in Brain Research*, t. 194, rozd. 5, s. 71–82, Elsevier, 2011, DOI: 10.1016/B978-0-444-53815-4.00016-9.
- [4] Blankertz B. i in.: The Berlin Brain–Computer Interface: Non-Medical Uses of BCI Technology. *Frontiers in Neuroscience*, t. 4, s. 198, 2010, DOI: 10.3389/fnins.2010.00198.
- [5] Boi F. i in.: A Bidirectional Brain-Machine Interface Featuring a Neuromorphic Hardware Decoder. *Frontiers in Neuroscience*, t. 10, grud. 2016, DOI: 10.3389/fnins.2016.00563.
- [6] Cegielska A., Olszewski M.: Nieinwazyjny interfejs mózg–komputer do zastosowań technicznych. *Pomiary Automatyka Robotyka*, t. R. 19, nr 3, s. 5–14, 2015.
- [7] Clerc M., Bougrain L. i Lotte F., red.: *Brain-Computer Interfaces 1: Foundations and Methods*, John Wiley & Sons, Inc., Hoboken, NJ, USA, lip. 2016, DOI: 10.1002/9781119144977.
- [8] Clerc M., Bougrain L. i Lotte F., red.: *Brain-Computer Interfaces 2: Technology and Applications*, John Wiley & Sons, Inc., Hoboken, NJ, USA, sierp. 2016, DOI: 10.1002/9781119332428.
- [9] Coyle D., red.: *Brain-Computer Interfaces: Lab Experiments to Real-World Applications*, Elsevier, Netherlands, 2016.
- [10] Dhiman R., Saini J., Mittal A.: Artifact removal from EEG recordings – An overview, *National Conference on Computational Instrumentation NCCI-2010*, s. 62–66, Chandigarh, India, mar. 2010.
- [11] Emotiv: *Cortex 2.0 API documentation*, URL: <https://emotiv.gitbook.io/cortex-api/> (dostęp: 04.08.2019).
- [12] Emotiv: *Develop with Emotiv*, URL: <https://www.emotiv.com/developer/> (dostęp: 08.04.2019).
- [13] Emotiv: *Emotiv EPOC+ 14 Channel Mobile EEG*, URL: <https://www.emotiv.com/product/emotiv-e poc-14-channel-mobile-eeg/> (dostęp: 21.04.2019).
- [14] Emotiv: *Emotiv Insight 5 Channel Mobile EEG*, URL: <https://www.emotiv.com/product/emotiv-insight-5-channel-mobile-eeg/> (dostęp: 21.04.2019).

- [15] Emotiv: *Emotiv PRO*, URL: <https://www.emotiv.com/emotivpro/> (dostęp: 08.04.2019).
- [16] Emotiv: *EmotivBCI documentation*, URL: <https://emotiv.gitbook.io/emotivbci/> (dostęp: 22.08.2019).
- [17] Emotiv: *Headsets comparison chart*, URL: <https://www.emotiv.com/comparison/> (dostęp: 04.04.2019).
- [18] Finisguerra A., Borgatti R., Urgesi C.: Non-invasive Brain Stimulation for the Rehabilitation of Children and Adolescents With Neurodevelopmental Disorders: A Systematic Review. *Frontiers in Psychology*, t. 10, lut. 2019, DOI: 10.3389/fpsyg.2019.00135.
- [19] Ghane-Motlagh B., Sawan M.: Design and Implementation Challenges of Micro-electrode Arrays: A Review. *Materials Sciences and Applications*, t. 04, nr 08, s. 483–495, 2013, DOI: 10.4236/msa.2013.48059.
- [20] Graimann B., Pfurtscheller G. i Allison B., red.: *Brain-Computer Interfaces: Revolutionizing Human-Computer Interaction*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, DOI: 10.1007/978-3-642-02091-9.
- [21] Hassanien A. E. i Azar A. T., red.: *Brain-Computer Interfaces: Current Trends and Applications*, Springer International Publishing, 2015, DOI: 10.1007/978-3-319-10978-7.
- [22] InteraXon Inc. *Compare Muse Devices*, URL: <https://choosemuse.com/product-comparison/> (dostęp: 23.04.2019).
- [23] InteraXon Inc. *Introducing Muse 2*, URL: <https://choosemuse.com/muse-2/> (dostęp: 23.04.2019).
- [24] InteraXon Inc. *Muse developer*, URL: <https://choosemuse.com/development/> (dostęp: 23.04.2019).
- [25] InteraXon Inc. *Muse professional*, URL: <https://choosemuse.com/muse-professionals/> (dostęp: 23.04.2019).
- [26] InteraXon Inc. *What are the differences between Muse 2 and Muse the brain sensing headband?*, URL: <https://choosemuse.force.com/s/article/How-is-Muse-2-different-than-Muse-the-brain-sensing-headband?> (dostęp: 23.04.2019).
- [27] JetBrains: *JetBrains Rider vs Visual Studio (with and without ReSharper)*, URL: <https://www.jetbrains.com/rider/compare/rider-vs-visual-studio/> (dostęp: 03.08.2019).
- [28] Lotte F., Congedo M., Lécuyer A., Lamarche F., Arnaldi B.: A review of classification algorithms for EEG-based brain-computer interfaces. *Journal of Neural Engineering*, t. 4, nr 2, R1–R13, sty. 2007, DOI: 10.1088/1741-2560/4/2/r01.

- [29] Microsoft: *Choose your app platform*, mar. 2019, URL: <https://docs.microsoft.com/en-us/windows/apps/desktop/choose-your-platform> (dostęp: 03.08.2019).
- [30] Min B.-K., Marzelli M. J., Yoo S.-S.: Neuroimaging-based approaches in the brain-computer interface. *Trends in Biotechnology*, t. 28, nr 11, s. 552–560, 2010, DOI: 10.1016/j.tibtech.2010.08.002.
- [31] Nam C. S., Nijholt A., Lotte F.: *Brain-Computer Interfaces Handbook: Technological and Theoretical Advances*, CRC Press, Inc., Boca Raton, FL, USA, 2018.
- [32] NeuroSky, Inc. *MindWave Mobile 2 - Brainwave Sensing Headset*, URL: <https://store.neurosky.com/pages/mindwave> (dostęp: 24.04.2019).
- [33] NeuroSky, Inc. *NeuroSky Developer*, URL: <http://developer.neurosky.com/docs/doku.php> (dostęp: 24.04.2019).
- [34] NeuroSky, Inc. *NeuroSky Research Tools*, URL: <https://store.neurosky.com/products/mindset-research-tools> (dostęp: 24.04.2019).
- [35] Nijholt A.: BCI for Games: A ‘State of the Art’ Survey, *Entertainment Computing - ICEC 2008*, s. 225–228, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [36] Onishi A., Natsume K.: Overlapped Partitioning for Ensemble Classifiers of P300-Based Brain-Computer Interfaces. *PLOS ONE*, t. 9, nr 4, kw. 2014, DOI: 10.1371/journal.pone.0093045.
- [37] OpenBCI: *Brainflow*, URL: <https://github.com/OpenBCI/brainflow> (dostęp: 04.06.2019).
- [38] OpenBCI: *Cyton + Daisy Biosensing Boards (16-channels)*, URL: <https://shop.openbci.com/collections/frontpage/products/cyton-daisy-biosensing-boards-16-channel> (dostęp: 29.04.2019).
- [39] OpenBCI: *Cyton Biosensing Board (8-channels)*, URL: <https://shop.openbci.com/collections/frontpage/products/cyton-biosensing-board-8-channel> (dostęp: 29.04.2019).
- [40] OpenBCI: *Ganglion Board (4-channels)*, URL: <https://shop.openbci.com/collections/frontpage/products/pre-order-ganglion-board> (dostęp: 29.04.2019).
- [41] OpenBCI: *OpenBCI GitHub profile*, URL: <https://github.com/OpenBCI> (dostęp: 19.08.2019).
- [42] OpenBCI: *The OpenBCI GUI*, URL: [https://github.com/OpenBCI/OpenBCI\\_GUI](https://github.com/OpenBCI/OpenBCI_GUI) (dostęp: 04.06.2019).
- [43] OpenBCI: *Ultracortex Mark IV*, URL: <https://docs.openbci.com/Headware/01-Ultracortex-Mark-IV> (dostęp: 28.04.2019).
- [44] OpenBCI: *Ultracortex Mark IV EEG Headset*, URL: <https://shop.openbci.com/products/ultracortex-mark-iv> (dostęp: 28.04.2019).

- [45] Paszkiel S.: Wykorzystanie metody PCA i ICA do analizy sygnału EEG w kontekście usuwania zakłóceń. *Pomiary Automatyka Kontrola*, t. R. 59, nr 3, s. 204–207, 2013.
- [46] Rao R. P.: *Brain-Computer Interfacing: An Introduction*, Cambridge University Press, New York, NY, USA, 2013.
- [47] *Technical specifications, validation, and research use*, InteraXon Inc., kw. 2017.
- [48] *The frequency of the letters of the alphabet in English*, URL: <https://www3.nd.edu/~busiforc/handouts/cryptography/letterfrequencies.html> (dostęp: 12.08.2019).
- [49] Vaibhav G.: *Brain-Computer Interfacing for Assistive Robotics*, Academic Press, San Diego, 2015, DOI: 10.1016/C2013-0-23408-5.
- [50] Wander J. D., Rao R. P.: Brain-computer interfaces: a powerful tool for scientific inquiry. *Current Opinion in Neurobiology*, t. 25, s. 70–75, kw. 2014, DOI: 10.1016/j.conb.2013.11.013.
- [51] Wolpaw J., Wolpaw E.: *Brain-Computer Interfaces: Principles and Practice*, Oxford University Press, USA, 2012.

# **Spis tabel**

2.1. Parametry Emotiv Insight . . . . .	27
2.2. Parametry Emotiv EPOC+ . . . . .	28
2.3. Parametry Muse oraz Muse 2 . . . . .	31
2.4. Parametry MindWave Mobile 2 . . . . .	32
2.5. Parametry Ultracortex Mark IV w połączeniu z kompatybilnymi układami . . . . .	37
4.1. Wyniki treningu uzyskane w programie EmotivBCI . . . . .	63

# Spis rysunków

1.1.	Homunkulus ruchowy Penfielda . . . . .	11
1.2.	Umiejscowienie elektrod wykorzystywanych w inwazyjnych BCI . . . . .	12
1.3.	Sposoby nieinwazyjnej akwizycji aktywności mózgu . . . . .	14
1.4.	Rodzaje artefaktów występujących w sygnałach EEG . . . . .	18
1.5.	Struktura interfejsu opartego o potencjał P300 . . . . .	21
1.6.	Zależność między stopniem skupienia, a stopą błędu . . . . .	23
2.1.	Hełm Emotiv Insight . . . . .	26
2.2.	Rozmieszczenie sensorów w hełmie Emotiv Insight . . . . .	26
2.3.	Hełm Emotiv EPOC+ . . . . .	28
2.4.	Rozmieszczenie sensorów w hełmie Emotiv EPOC+ . . . . .	29
2.5.	Opaska Muse 2 . . . . .	30
2.6.	Rozmieszczenie sensorów w opasce Muse . . . . .	30
2.7.	Hełm MindWave Mobile 2 . . . . .	32
2.8.	Hełm Ultracortex Mark IV . . . . .	34
2.9.	Rozmieszczenie sensorów w hełmie Ultracortex Mark IV . . . . .	34
2.10.	OpenBCI Ganglion Board . . . . .	35
2.11.	OpenBCI Cyton Board . . . . .	35
2.12.	OpenBCI Cyton + Daisy Board . . . . .	36
3.1.	Struktura rozwiązania przedstawiona w postaci drzewa . . . . .	44
3.2.	Schemat blokowy algorytmu aplikacji . . . . .	46
3.3.	Proces treningu komend z wykorzystaniem aplikacji <i>EmotivBCI</i> . . . . .	48
3.4.	Widok zakładki <i>Virtual Keyboard</i> . . . . .	56
3.5.	Widok zakładki <i>Messenger</i> . . . . .	58
3.6.	Widok zakładki <i>About</i> . . . . .	59
3.7.	Widok zakładki <i>Connection</i> . . . . .	59
3.8.	Widok zakładki <i>Settings</i> . . . . .	60
4.1.	Sposób uznawania komendy mentalnej . . . . .	62
4.2.	Wyniki treningu uzyskane w programie EmotivBCI . . . . .	65
4.3.	Schemat przestrzeni mózgowej uzyskany w programie EmotivBCI . . . . .	67

# **Spis kodów źródłowych**

3.1. Przykładowe zapytanie Cortex API . . . . .	49
3.2. Przykładowy wynik zapytania Cortex API . . . . .	49
3.3. Kod klasy Request . . . . .	50
3.4. Kod klasy AuthorizeRequest . . . . .	50
3.5. Kod klasy AuthorizeParameter . . . . .	51
3.6. Kod klasy Response . . . . .	51
3.7. Kod klasy AuthorizeResponse . . . . .	52
3.8. Kod metody SendRequest . . . . .	52
3.9. Kod metody Authorize . . . . .	53
3.10. Zserializowane zapytanie AuthorizeRequest . . . . .	53
3.11. Odpowiedź otrzymana na zapytanie AuthorizeRequest . . . . .	53
3.12. Kod konstruktora klasy KeyboardNavigator . . . . .	54
3.13. Kod metody Insight_PropertyChanged . . . . .	55
3.14. Kod metody _rightTimer_Tick . . . . .	55
3.15. Kod metody Settings_PropertyChanged . . . . .	55