

SAMSON – System Analizy MioSygnałów Oparty Na sieci neuronowej

Dokumentacja projektu

Jakub Wilczyński
Kinga Michalak
Maksymilian Tulewicz
Mikołaj Strużykowski
Paulina Piorun
Rozalia Nowicka

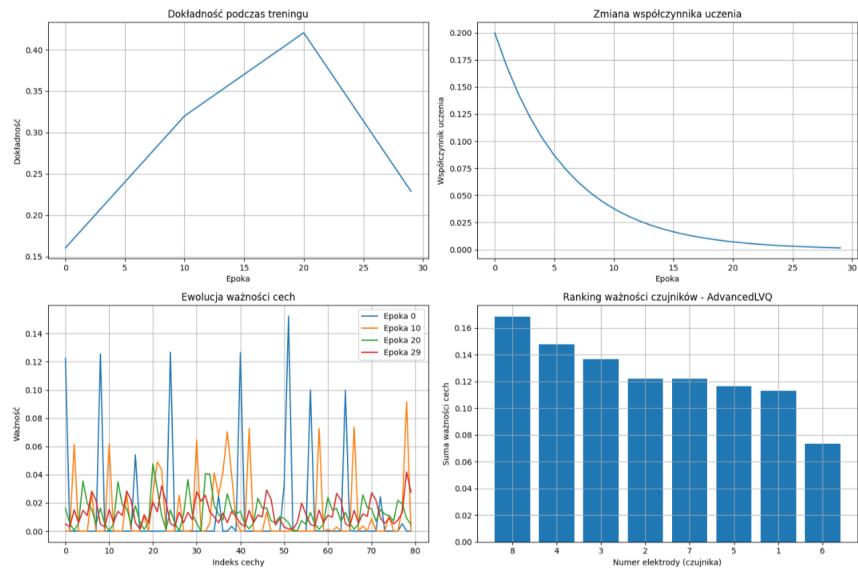
1 Kamień milowy I – raporty dotyczące postępu prac poszczególnych członków zespołu, a zaplanowane cele

Jakub Wilczyński

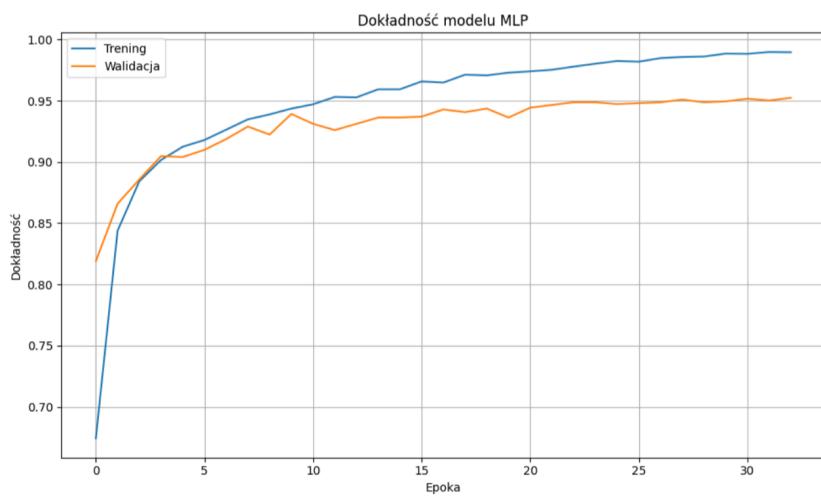
Do dziś wszystkie prace nad projektem **SAMSON — System Analizy MioSygnałów Oparty Na Sieci Neuronowej** przebiegają zgodnie z założonym planem. Pierwszy kamień milowy, wyznaczony na 25 kwietnia, został zrealizowany w pełni. Zakończyłem pracę z danymi testowymi, wykorzystując trzy różne typy modeli: **Learning Vector Quantization (LVQ)**, **Multilayer Perceptron (MLP)** oraz **jednowymiarowe sieci konwolucyjne (1D-CNN)**.

- **LVQ** — klasyfikator bazujący na prototypach, uczący się przez modyfikację wektorów reprezentujących klasę. Jego główną zaletą jest interpretowalność, jednak w testach uzyskał najgorsze wyniki pod względem skuteczności klasyfikacji i stabilności nauki.
- **MLP** — klasyczna sieć neuronowa składająca się z wielu warstw w pełni połączonych. Pokazała dobrą efektywność oraz czas uczenia zbliżony do 1D-CNN.
- **1D-CNN** — sieć konwolucyjna działająca na sekwencjach jednowymiarowych, skuteczna w rozpoznawaniu lokalnych wzorców czasowych. Zaprezentowała podobny poziom dokładności i szybkości uczenia jak MLP.

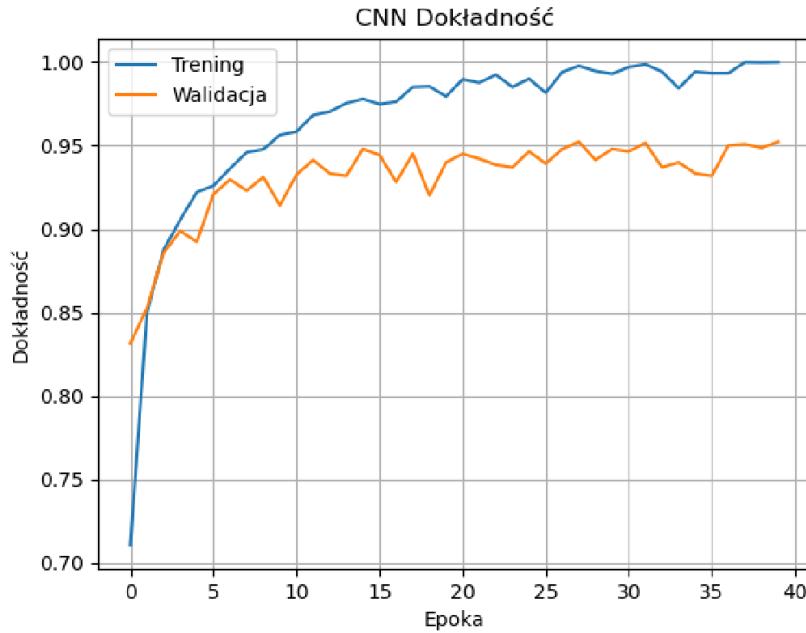
We wszystkich modelach zastosowano mechanizmy przeciwdziałające przeuczeniu, takie jak regularizacja L2 oraz technika *early stopping*. Testy przeprowadzono na zbiorze danych EKG udostępnionym na platformie Kaggle, traktowanym jako analogiczny przypadek przetwarzania sygnałów biologicznych. Poniżej przedstawiono przykładowe wykresy z przebiegu uczenia dla każdej z architektur: Obecnie realizuję drugi etap projektu, którego kamień milowy przypada na 24 maja. Zajmuję się implementacją oraz testowaniem danych rzeczywistych przekazanych przez zespół odpowiedzialny za analizę danych. Dane te charakteryzują się istotnymi zależnościami czasowymi, których brakowało w zbiorach testowych używanych wcześniej. Na ich podstawie planuję przeprowadzić analizę efektywności różnych konfiguracji rozmieszczenia elektrod na przedramieniu, w celu określenia optymalnego układu pod kątem skuteczności klasyfikacji mio-sygnałów.



Rysunek 1: Wykres uczenia modelu LVQ.



Rysunek 2: Wykres uczenia modelu MLP.



Rysunek 3: Wykres uczenia modelu 1D-CNN.

Kinga Michalak

Wybranie materiału

Dokonano researchu na temat materiału, z którego powinna zostać wykonana elektroda. Uwzględniono czynniki takie jak bezpieczeństwo osoby badanej, interakcje materiału (metalu) ze skórą oraz żelom przewodzącym, a także koszty i dostępność surowca. Na tej podstawie zdecydowano się na użycie srebra jako materiału przewodzącego. Jako żel wspomagający przesyłanie sygnałów wybrano [?].

Dobranie kształtu elektrody

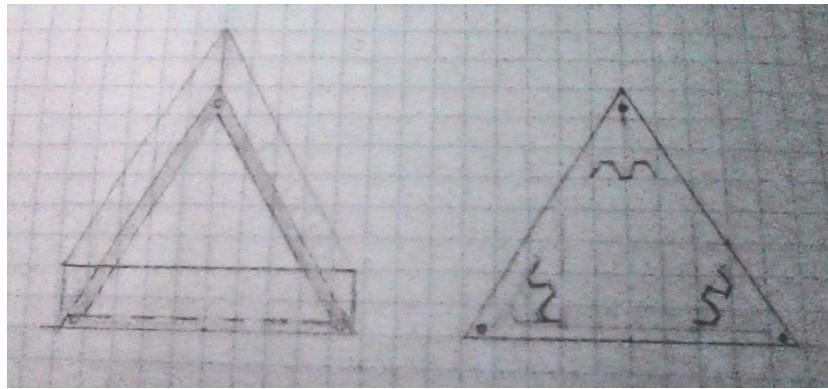
Po analizie wielu rozwiązań używanych na rynku oraz prac naukowych wybrano dla elektrody kształt trójkątny z 3 drutami w każdym rogu, gdzie sygnał odbierany z jednego będzie sygnałem referencyjnym. Odległość między drutami będzie wynosiła 1cm tak, żeby sygnały nie były zakłócane. W celu uzyskania bardziej dokładnych rezultatów i stabilnego sygnału, druty nie będą pojedynczym „paskiem”, a zgęstą „sprzęzyną” o zagięciach 90°.

Projekt elektrody

Elektroda składać się będzie z płytka, do której od spodu przyjmocowane zostaną druty, a od góry elementy elektroniczne. Całość zostanie umieszczona w ochronnym pojemniku w kształcie trójkąta z otworem umożliwiającym kontakt drutów ze skórą. Pojemnik będzie wyposażony w ścianki i pokrywkę, które zabezpieczą elementy elektroniczne. W rogach przewidziano otwory umożliwiające przykręcenie płytka do obudowy. Elektrody będą mocowane do ciała za pomocą ręcznie wykonanych, szydełkowych opasek z akrylu, z użyciem żelu poprawiającego przewodnictwo elektryczne.

Pojemniki zostaną zaprojektowane w Autodesk Fusion 360 i wydrukowane na drukarce 3D.

Pierwotny projekt przedstawia opakowanie (po lewej) oraz płytka z drutami (po prawej) na rysunku 4. Zakłada on elektrodę w kształcie trójkąta równobocznego o długości boku 4,8 cm, a rozmieszczenie drutów, oddalonych od siebie o co najmniej 1,5 cm, znajduje się na kole o średnicy 1,6 cm. Przeprowadzone zostaną testy mające na celu sprawdzenie, czy zmniejszenie odległości między drutami wpłynie na jakość otrzymywanej sygnału.



Rysunek 4: Projekt elektrody

Wybranie rozmieszczeń elektrod

Zostało wybrane rozmieszczenie 3 elektrod zbierających odpowiednie sygnały:

- Elektroda na zginacze palców II–V
Lokalizacja: 1/3 odległości między stawem łokciowym a nadgarstkiem, na linii środkowej przedramienia – po stronie wewnętrznej przedramienia, tam gdzie mięśnie się napinają przy robieniu pięści.
- Elektroda na zginacz kciuka
Lokalizacja: jest to mięsień wewnętrzny, nie ma o nim na SENIAM informacji. Należy umieścić po wewnętrznej stronie ręki, ale bliżej promieniowej strony (kciuka), ok. 1/3 długości przedramienia od nadgarstka.
- Elektroda na prostowniki palców
Lokalizacja: 1/4 odległości od nadkłykcia bocznego kości ramiennej do wyrostka rylcowatego kości łokciowej – na grzbietowej stronie przedramienia, w górnej 1/3 jego długości.

Maksymilian Tulewicz

Na początku zająłem się stworzeniem podstawowego modelu dłoni w środowisku CAD Autodesk Inventor. Projekt opierał się na analizie proporcji ludzkiej dłoni, a inspirację czerpałem z mechanicznych dloni przedstawionych w filmie *Terminator* oraz w grze *Cyberpunk 2077*.

Po opracowaniu modelu rozpoczęłem analizę środowisk symulacyjnych, w których chciałem przetestować działanie dloni. Postanowiłem przystosować model do symulacji w Gazebo, co wymagało eksportu pliku do formatu URDF.

Pierwsze podejście wykonałem w programie Blender z użyciem nakładki Phobos. Niestety, mimo wielu godzin pracy, napotkałem trudności z obsługą formatów plików oraz niekompatybilnością eksportu. W związku z tym zdecydowałem się zmienić strategię.

Kolejną próbę podjąłem w środowisku Fusion 360 z wykorzystaniem nakładki Fusion2URDF do której w repozytorium GitHub. Program ten okazał się bardziej kompatybilny z Inventorem, jako że pochodzi od tego samego producenta.

Po wygenerowaniu pliku URDF napotkałem kolejne wyzwanie — plik był przystosowany do pracy w środowisku ROS1, podczas gdy celem projektu była praca z ROS2, używanym na zajęciach. Po przeanalizowaniu różnic pomiędzy wersjami ROS zdecydowałem się jednak pozostać przy ROS1.

W kolejnym etapie planuję stworzyć program sterujący gestami dloni, oparty na danych wejściowych z sieci neuronowej.

W odniesieniu do kamienia milowego projektu: zakończenie pracy każdej z osób z danymi testowymi oraz działające fragmenty projektu — mogę stwierdzić, że etap ten został niemal osiągnięty. Model dloni został zaimplementowany w środowisku symulacyjnym, jednak obecnie nie obsługuje jeszcze danych wejściowych, co będzie celem dalszych prac.

Mikołaj Strużykowski

Projekt elektroniki

Jako główny element wzmacniający sygnał elektrod wybrałem wzmacniacz różnicujący, dzięki temu porównując sygnał z dwóch sąsiadujących elektrod do referencyjnej wszelkie sygnały wspólne zostaną od siebie odjęte. Takim sygnałem jest szum, który dla dwóch elektrod położony blisko siebie powinien być bardzo podobny.

Kolejnym elementem jest wzmacniacz nieodwracający, który dzięki potencjometrowi będzie mógł podnieść poziom sygnału do 3.3V co będzie poprawnie odczytywane przez mikrokontroler.

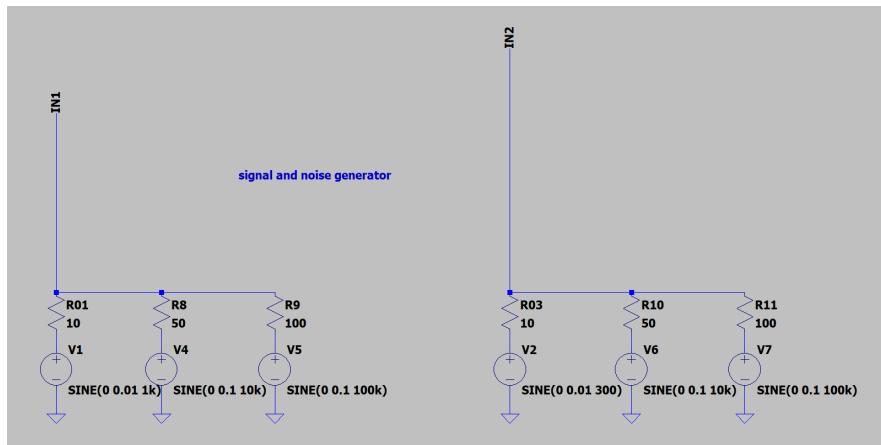
Ostatnim elementem poprzedzającym mikrokontroler jest filtr dolnoprzepustowy odcinający częstotliwości powyżej 4.8k Ω . Wiemy, że nie jest to interesujący nas zakres.

Przed pierwszym wzmacnieniem sygnału wymagane jest dodanie stałego napięcia do sygnału, tak żeby nie była odcinana część sygnału poniżej 0V.

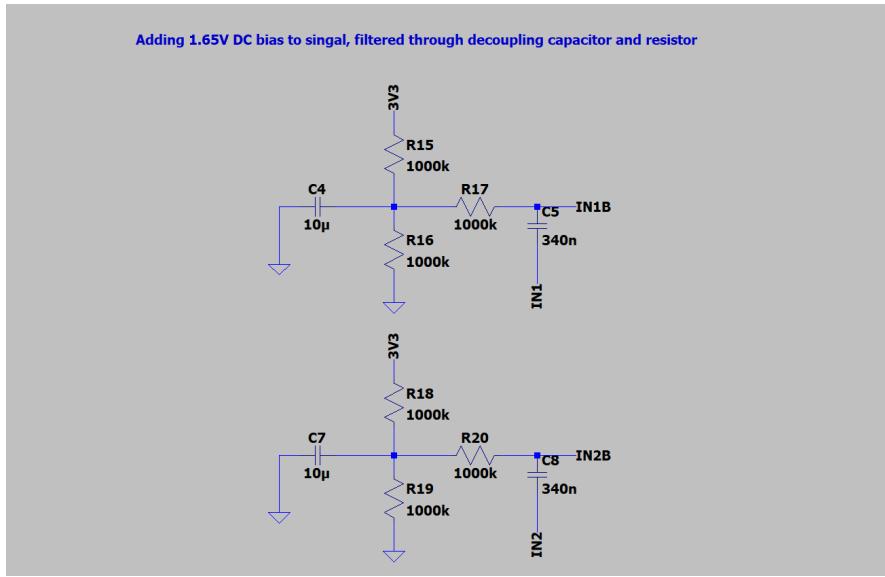
Do programu LTspice nie udało się dodać wzmacniacza, który będzie końcowo używany jednak został zastąpiony podobnym modelem. Używany końcowo wzmacniacz to MCP6021 wzmacniacz typu *Rail-to-rail* z dużą impedancją wejściową oraz dostosowany do używania w układach wzmacniacza różnicującego.

Symulacje w programie LTspice

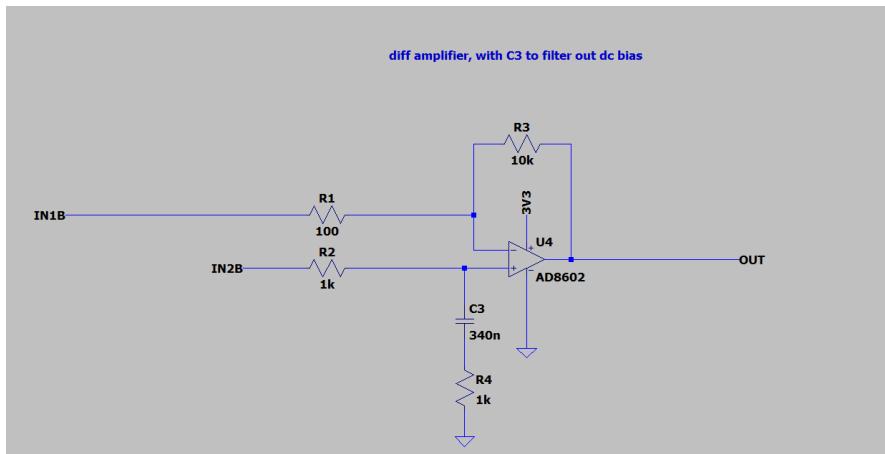
Budowa układu:



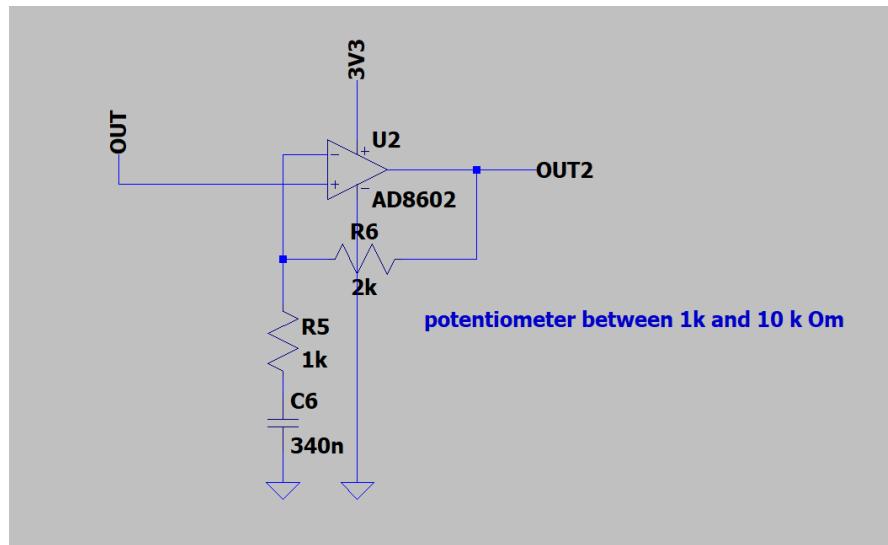
Rysunek 5: Generator sygnału i szumu.



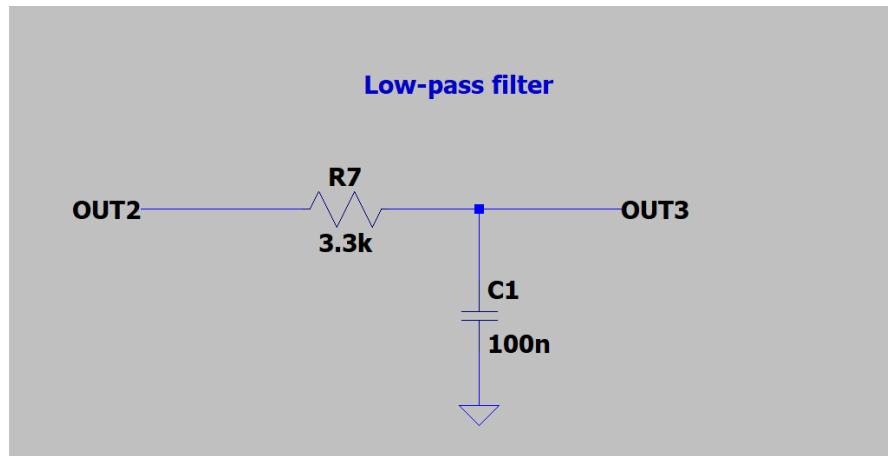
Rysunek 6: Dodawanie stałej.



Rysunek 7: Wzmacniacz różnicujący.

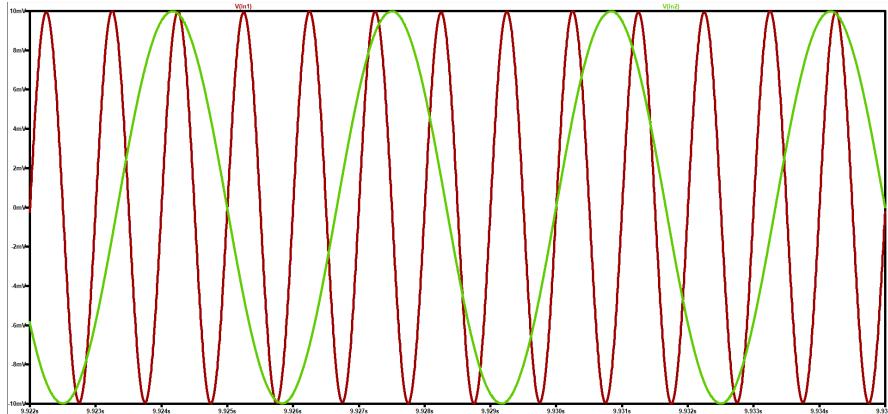


Rysunek 8: Wzmacniacz nieodwracający z potencjometrem.

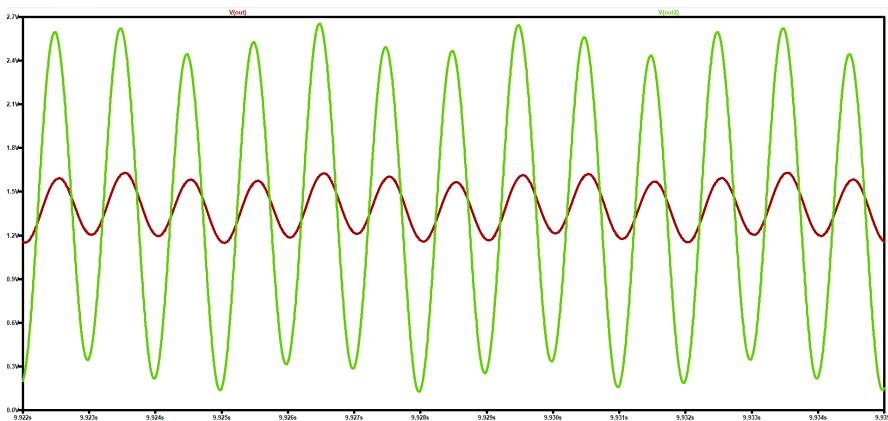


Rysunek 9: Filtr dolnowprzepustowy.

Wyniki:



Rysunek 10: Sygnały wyjściowe z elektrod.



Rysunek 11: Sygnał po przejściu przez wzmacniacz różnicujący(czerwony) i nieodwracający(zielony).

Fizyczny model elektrody

Czekamy na elementy elektryczne, które pozowią na stworzenie prototypu w technologii THT, gdy okaże się spełniać swoje zadanie zostanie utworzony model płytki drukowanej PCB. W ten sposób elektroda zostanie odpowiednio zminiaturyzowana.

Paulina Piorun

Zakres tematyczny

Opisana w kolejnych rozdziałach część projektu dotyczy następujących tematów i zagadnień:

- cyfrowa Filtracja sygnałów,
- analiza sygnału,
- wyodrębnienie cech sygnału,
- normalizacja.

Zrealizowane zadania

W ramach pierwszego etapu prac zrealizowano następujące zadania:

- Zaprojektowano filtr pasmowoprzepustowy oraz pasmowozaporowy.

- Wyodrębniono 25 cech sygnału.
- Poddano wybrane cechy normalizacji.
- Przygotowano pliki zawierające wyodrębnione cechy, wykorzystywane w dalszym procesie rozpoznawania gestów.

Dane użyte do testów

Do testowania wykorzystano publicznie dostępną bazę danych GrabMyo udostępnioną przez PhysioNet [4], zawierającą sygnały EMG zarejestrowane podczas wykonywania gestów ręką. Dane obejmują trzy oddzielne sesje pomiarowe przeprowadzone w różne dni, z udziałem 43 uczestników w każdej sesji. Każdy uczestnik wykonał 17 gestów (w tym stan spoczynku), z których każdy był powtarzany siedmiokrotnie, co daje obszerny i zróżnicowany zestaw prób. Każda próba trwała 5 sekund, a dane były rejestrowane z częstotliwością 2048 Hz przy użyciu 32-kanałowego układu elektrod rozmieszczonej na przedramieniu i nadgarstku. Pliki pomiarowe zawierają zarówno sygnały EMG (.dat), jak i metadane (.hea), a dodatkowe informacje o uczestnikach dostępne są w pliku subject-info.

Wykorzystane oprogramowanie

Do realizacji projektu wykorzystano środowisko **MATLAB** w wersji:

- **MATLAB**: R2024b (wersja 24.2.0.2863752, Update 5)
- **System operacyjny**: Microsoft Windows 10 Home, wersja 10.0 (komplikacja 19045)
- **Java**: Java 1.8.0_202-b08 z Oracle Corporation (Java HotSpot™ 64-Bit Server VM)

W projekcie wykorzystano następujące toolboxy MATLAB-a:

- **MATLAB (core)** - wersja 24.2
- **Signal Processing Toolbox** - wersja 24.2
- **Wavelet Toolbox** - wersja 24.2

Filtracja sygnału

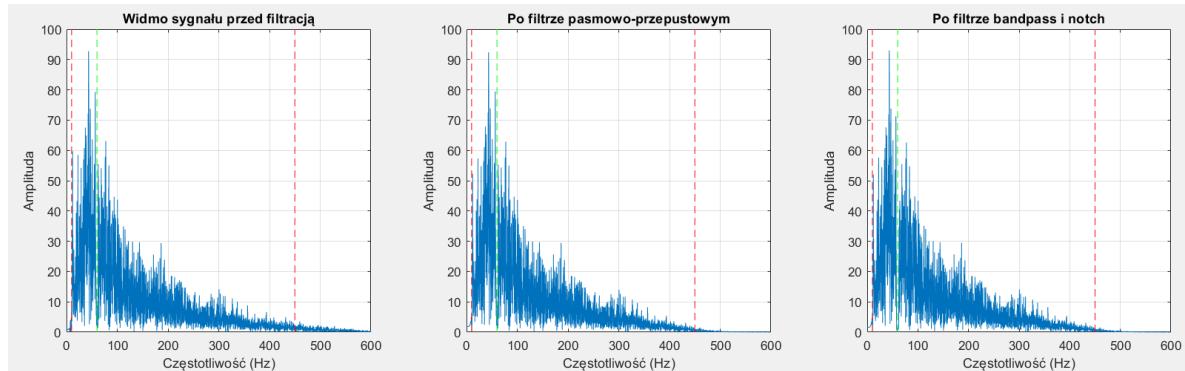
Przed procesem ekstrakcji cech sygnału przystąpiono do filtracji sygnału. Największe zakłócenia występujące w sygnale to te związane z częstotliwością napięcia sieci energetycznej (50 Hz lub 60 Hz). Za zakres częstotliwości, który niesie istotne informacje przyjęto 10 Hz do 450 Hz (na podstawie informacji z artykułu [6]).

Jak zauważono w innym artykule [5], wartości te mogą być inne. Cytując: „*It is seen that 95 % of the signal energy is included in the frequency range from 30 Hz to 250 Hz and 99 % do not exceed range above 300 Hz*” (czyli 95 % energii sygnału zawiera się w paśmie 30 Hz do 250 Hz, a 99 % nie przekracza 300 Hz).

W tym przypadku zastosowano **filt Butterwortha**, ponieważ ma on następujące cechy:

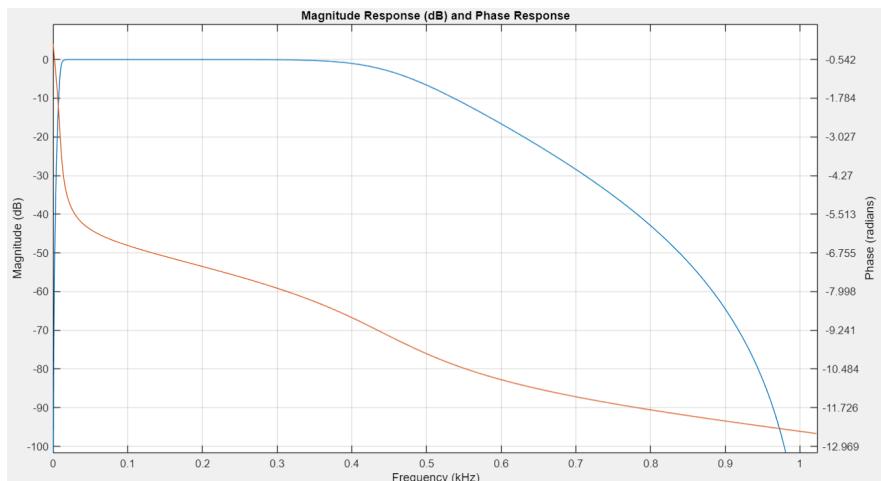
- Płaska charakterystyka amplitudowa – nie zniekształca amplitudy sygnału w paśmie przepustowym.
- Brak przesunięcia fazowego – przy użyciu funkcji `filtfilt()` sygnał nie jest opóźniony.
- Stabilność i prostota – łatwy w implementacji i zawsze stabilny.
- Uniwersalność – można go używać jako filtr dolno-, górnego, pasmowoprzepustowy lub zaporowy.

Na rysunku 12 przedstawiono widmo sygnału na trzech etapach filtracji: przed filtracją, po dodaniu filtru pasmowoprzepustowego, po dodaniu filtru pasmowozaporowego. Liniami czerwonymi zaznaczono częstotliwości graniczne dla filtru pasmowoprzepustowego, natomiast zielona linia odpowiada częstotliwości, której należy się pozbyć używając filtru pasmowozaporowego.

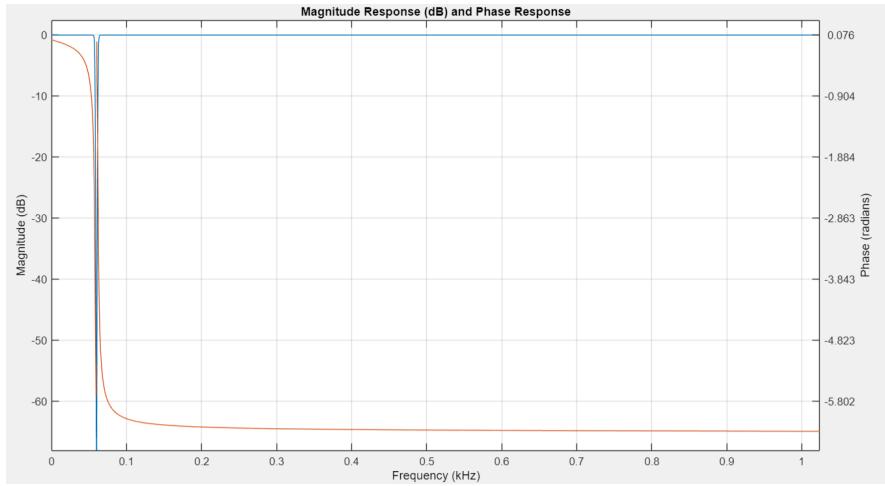


Rysunek 12: Efekt filtracji: przed filtracją, po zastosowaniu filtru pasmowoprzepustowego, po filtracji pasmowozaporowej.

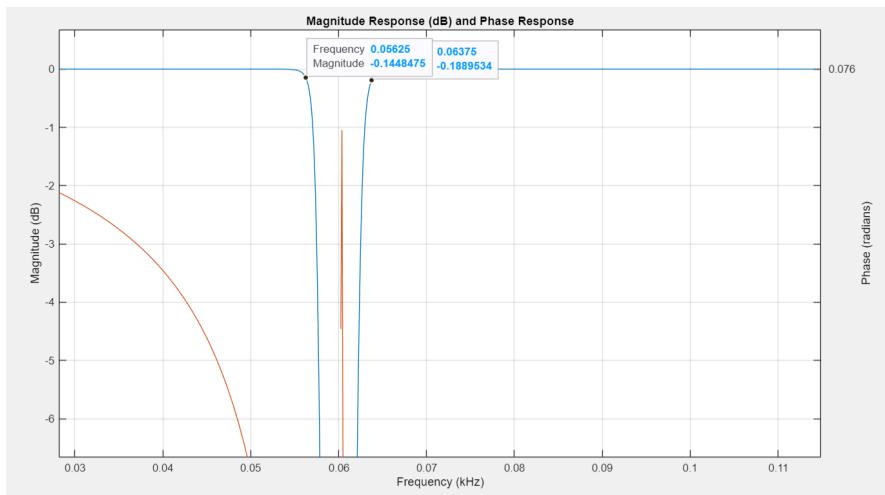
Charakterystyki filtrów W poniższym rozdziale przedstawiono charakterystyki zaprojektowanych filtrów. Na rysunku 13 przedstawiono charakterystykę amplitudowo-fazową dla filtru pasmowoprzepustowego. Natomiast rysunki 14 i 15 przedstawiają charakterystykę amplitudowo-fazową filtru pasmowozaporowego – wycinającego wąski zakres częstotliwości związanych z siecią energetyczną. Kolorem czerwonym przedstawiona jest charakterystyka fazowa a kolorem niebieskim charakterystyka amplitudowa.



Rysunek 13: Charakterystyka amplitudowo-fazowa dla filtru pasmowoprzepustowego



Rysunek 14: Charakterystyka amplitudowo-fazowa dla filtra pasmowozaporowego



Rysunek 15: Charakterystyka amplitudowo-fazowa dla filtra pasmowozaporowego – widok w przybliżeniu

Wyodrębnione cechy

W ramach analizy wyodrębniono 25 cech sygnału EMG, które zgodnie z literaturą charakteryzują się wysoką skutecznością w rozróżnianiu wzorców aktywności mięśniowej i mogą być efektywnie wykorzystywane do identyfikacji gestów. W tabeli 1 przedstawiono zbiór cech wykorzystywanych do identyfikacji gestów, wybranych na podstawie dwóch artykułów [5] oraz [6]. Zastosowana transformacja falkowa umożliwia jednoczesną analizę sygnału EMG w dziedzinie czasu i częstotliwości, co pozwala skuteczniej wykrywać zmiany oraz charakterystyczne cechy aktywności mięśniowej. Z kolei podział sygnału EMG na krótkie okna czasowe pozwala na uchwycenie lokalnych właściwości sygnału, zwiększając precyzję analizy. W przypadku sygnału o długości 5 sekund, zastosowanie okien o długości 200 ms (410 próbek) z przesunięciem co 100 ms (205 próbek) umożliwia rejestrowanie dynamicznych zmian aktywności mięśni z wysoką rozdzielczością czasową, przy zachowaniu odpowiedniego nakładania się okien, co sprzyja ciągłości i dokładności ekstrakcji cech.

Tabela 1: Zestawienie cech wykorzystywanych do identyfikacji gestów na podstawie sygnału EMG.

Nr	Nazwa cechy (ang)	Skrót	Nazwa cechy (pl)
1	Integrated EMG	IEMG	Zintegrowany sygnał EMG
2	Mean Absolute Value	MAV	Średnia wartość bezwzględna
3	Modified Mean Absolute Value	MMAV	Zmodyfikowana średnia wartość bezwzględna
4	Simple Square Integral	SSI	Prosty całek kwadratowy
5	Variance	VAR	Wariancja
6	Root Mean Square	RMS	Pierwiastek średniokwadratowy
7	v-Order 2	V2	Rząd v = 2
8	v-Order 3	V3	Rząd v = 3
9	Log Detector	LOG	Wykrywacz logarytmiczny
10	Waveform Length	WL	Długość przebiegu
11	Average Amplitude Change	AAC	Średnia zmiana amplitudy
12	Difference Absolute Standard Deviation	DASDV	Różnicowa odchylenie standardowe bezwzględne
13	Maximum Fractal Length	MFL	Maksymalna długość fraktalna
14	Myopulse Percentage Rate	MYOP	Procentowa liczba impulsów mięśniowych
15	Zero Crossing	ZC	Liczba przejść przez zero
16	Willison Amplitude	WAMP	Amplituda Willisona
17	Mean Power	MNP	Średnia moc
18	Total Power	TTP	Całkowita moc
19	Median Frequency	MDF	Mediana częstotliwości
20	Mean Frequency	MNF	Średnia częstotliwość
21	Frequency Ratio	FR	Stosunek częstotliwości
22	Power Spectrum Ratio	PSR	Stosunek widma mocy
23	Spectral Moment 1	SM1	Spektralny moment 1. rzędu
24	Spectral Moment 2	SM2	Spektralny moment 2. rzędu
25	Spectral Moment 3	SM3	Spektralny moment 3. rzędu

Tabela 2: Tabela przedstawia optymalne komponenty falkowe oraz indeksy RES dla różnych cech EMG
 Źródło: [Feature Extraction and Reduction of Wavelet Transform Coefficients for EMG Pattern Classification](#)

Feature extraction	Optimal wavelet component	Optimal WF	RES index
ZC	D2	Db7	12.52
WAMP	D2	Db7	12.22
MAV, IEMG	D2	Db7	11.11
VAR, SSI, TTP, MNP	D2	Db7	10.43
MMAV	D2	Db7	10.69
SM1	D2	Db7	10.45
LOG	D2	Db7	9.53
RMS, V2	cD1	Db10	11.06
V3	cD1	Db10	10.27
SM2	cD1	Db10	10.23
SM3	cD1	Db10	9.83
MYOP	D1	Db5	12.95
MFL	D1	Db8	9.86
WL, AAC	Raw	-	11.69
DASDV	Raw	-	11.15
MNF	Raw	-	5.48
MDF	Raw	-	5.38
PSR	Raw	-	4.99
FR	Raw	-	3.46

Tabela 3: Lista wybranych cech sygnału EMG wraz z informacją o normalizacji do zakresu 0-1

Nr	Feature Name	Normalizacja
1	Frequency Ratio	-
2	Power Spectrum Ratio	-
3	Median Frequency	-
4	Mean Frequency	-
5	Difference Absolute Standard Deviation	+
6	Waveform Length	+
7	Average Amplitude Change	+
8	Maximum Fractal Length	-
9	Myopulse Percentage Rate	-
10	Spectral Moment 2	-
11	Spectral Moment 3	-
12	v-Order 2	-
13	v-Order 3	-
14	Root Mean Square	+
15	Log Detector	-
16	Spectral Moment 1	+
17	Modified Mean Absolute Value	+
18	Variance	-
19	Simple Square Integral	+
20	Total Power	+
21	Mean Power	+
22	Mean Absolute Value	+
23	Integrated EMG	-
24	Willison Amplitude	-
25	Zero Crossing	-

Zapis do pliku

Dane zostały zapisane do pliku w formacie CSV, gdzie wartości w poszczególnych wierszach są oddzielone średnikami (;). Pierwsza kolumna zawiera numer kanału, druga kolumna to numer okna czasowego, a ostatnia kolumna wskazuje numer gestu. Kolejne kolumny odpowiadają wybranym cechom sygnału EMG, zapisanym w postaci skrótów, takich jak IEMG, MAV, RMS, itp. Taki format umożliwia łatwą organizację danych i ich późniejszą analizę, zapewniając szybki dostęp do informacji o cechach, kanałach i oknach czasowych, co jest istotne w procesie klasyfikacji gestów.

Podsumowanie

Wszystkie zaplanowane zadania związane z przetwarzaniem sygnałów EMG do etapu I projektu zostały zrealizowane. Zaimplementowano odpowiednie filtry do przetwarzania sygnałów, a także zdefiniowano sposób zapisów wyników do pliku w formacie CSV. Kolejnym krokiem jest integracja z siecią neuronową, która będzie odpowiedzialna za rozpoznawanie gestów na podstawie przetworzonych cech sygnałów EMG. W tej chwili potrzeba przeprowadzenia testów, aby sprawdzić, jak system radzi sobie z klasyfikowaniem gestów oraz czy można wprowadzić zmiany w sposobie ekstrakcji cech, które wpływają na poprawę wyników.

Istnieją jeszcze kwestie, które mogą wymagać dopracowania i dalszej analizy:

- Normalizacja: proces normalizacji danych jest kluczowy, ale wymaga przemyślenia, które cechy powinny być normalizowane, aby zachować istotne informacje, jednocześnie unikając utraty cennych danych.
- Redukcja cech: choć redukcja wymiarowości jest niezbędna do uproszczenia modelu, należy zwrócić uwagę, aby nie utracić ważnych cech, które mogą wpływać na jakość rozpoznawania gestów.

- Okna czasowe: wybór rozmiaru i przesunięcia okna może wpływać na rozdzielcość czasową i dokładność analizy. Dobór tych parametrów może wymagać dalszych eksperymentów w zależności od specyfiki gestów.

Te kwestie powinny zostać szczegółowo przeanalizowane w kontekście wyników uzyskanych z siecią neuronową oraz przeprowadzonych testów, aby zoptymalizować cały proces rozpoznawania gestów. Kolejny etap projektu będzie skupiał się na dostosowaniu zaprojektowanych rozwiązań do danych rzeczywistych.

Rozalia Nowicka

Opis

W ramach realizacji projektu podjęto działania dotyczące przygotowania komunikacji mikrokontrolera z komputerem oraz rozpoczęto prace nad odczytem sygnału analogowego. Wykonywane zadania wynikały z ustalonego wcześniej harmonogramu.

Protokół komunikacyjny

W ramach prac dotyczących protokołu komunikacyjnego opracowano format ramki danych, określono prędkość transmisji oraz wybrano sposób liczenia sumy kontrolnej. Ramka rozpoczyna się zdefiniowanym nagłówkiem i ma określoną długość pięciu bajtów. Struktura dla jednego kanału ADC (aktualnie używana): `0xAA<adc_msб><adc_lsb><crc_lsb><crc_msб>`

gdzie:

- `0xAA` – nagłówek,
- `<adc_msб>` – starszy bajt wartości ADC,
- `<adc_lsb>` – młodszy bajt wartości ADC,
- `<crc_lsb>` – młodszy bajt CRC16,
- `<crc_msб>` – starszy bajt CRC16.

Docelowo planuje się ramkę zawierającą informacje z trzech kanałów ADC. Jej strukturę przedstawiono poniżej:

`0xAA<adc1_msб><adc1_lsb><adc2_msб><adc2_lsb><adc3_msб><adc3_lsb><crc_lsb><crc_msб>`

gdzie:

- `0xAA` – nagłówek,
- `<adc1_msб>` – starszy bajt wartości ADC1,
- `<adc1_lsb>` – młodszy bajt wartości ADC1,
- `<adc2_msб>` – starszy bajt wartości ADC2,
- `<adc2_lsb>` – młodszy bajt wartości ADC2,
- `<adc3_msб>` – starszy bajt wartości ADC3,
- `<adc3_lsb>` – młodszy bajt wartości ADC3,
- `<crc_lsb>` – młodszy bajt CRC16,
- `<crc_msб>` – starszy bajt CRC16.

Do obliczania sumy kontrolnej zastosowano metodę CRC-16 z uwagi na jej skuteczność i akceptowalną złożoność obliczeniową. Poniżej przedstawiono jej implementację.

```

1 uint16_t CRC16( const uint8_t *data , uint16_t length ) {
2     uint16_t crc = 0xFFFF;
3     for (uint16_t i = 0; i < length; i++) {
4         crc ^= data[ i ];
5         for (uint8_t j = 0; j < 8; j++) {
6             if (crc & 0x0001) {
7                 crc >>= 1;
8                 crc ^= 0xA001;
9             } else {
10                 crc >>= 1;
11             }
12         }
13     }
14     return crc;
15 }
```

UART

Dokonano odpowiedniej konfiguracji UART ustalając prędkość transmisji na 115200 bps. Opracowano prosty program na komputer (Python) odbierający dane przez UART i weryfikujący poprawność struktury ramek oraz sumy kontrolnej.

ADC

Przeanalizowano materiały dotyczące rejestracji sygnałów EMG i ustaloną częstotliwość próbkowania na 1 kHz. Opracowano i uruchomiono uproszczony program konwertujący napięcie analogowe na wartość cyfrową przy pomocy wbudowanego przetwornika ADC i przerwań. Wyniki wstępnie wyświetlono przez UART.

2 Kamień milowy II – raporty dotyczące postępu prac poszczególnych członków zespołu, a zaplanowane cele

Jakub Wilczyński

Postęp prac nad projektem SAMSON — System Analizy MioSygnałów Oparty Na Sieci Neuronowej

W ostatnim etapie projektu skoncentrowałem się na przetwarzaniu rzeczywistych danych EMG i do pracowania architektury sieci neuronowej dedykowanej do zadania rozpoznawania gestów na podstawie sygnałów mioelektrycznych. Po przeprowadzeniu wstępnych eksperymentów z różnymi architekturami, zdecydowałem się na rozwijanie i strojenie modelu 1D-CNN+LSTM, który najlepiej odpowiadał specyfice problemu. Modele klasyczne nie były w stanie uchwycić złożonych zależności czasowych oraz lokalnych wzorców w sekwencjach EMG, przez co ich skuteczność była dalece niezadowalająca. W efekcie, cała uwaga została skupiona na dopracowaniu i analizie działania zaawansowanego modelu konwolucyjno-rekurencyjnego.

Przetwarzanie danych EMG

Otrzymane dane rzeczywiste wymagały złożonego procesu przygotowania:

- Połączono wiele źródeł sygnałów w jeden spójny zbiór danych,
- Opracowano algorytm budowania sekwencji nakładających się okien czasowych dla każdego kanału i gestu,
- Wprowadzono normalizację cech, by zapewnić porównywalność wartości pomiędzy różnymi próbami i kanałami,
- Oznaczono etykiety gestów w postaci numerycznej, umożliwiając skuteczne uczenie modelu,

- Zastosowano stratyfikowany podział na zbiory treningowe i testowe, by zachować proporcje klas.

Podczas tego procesu analizowałem także rozkład cech, liczbę próbek na klasę i kanał oraz koreacje pomiędzy cechami w celu optymalnego przygotowania danych wejściowych.

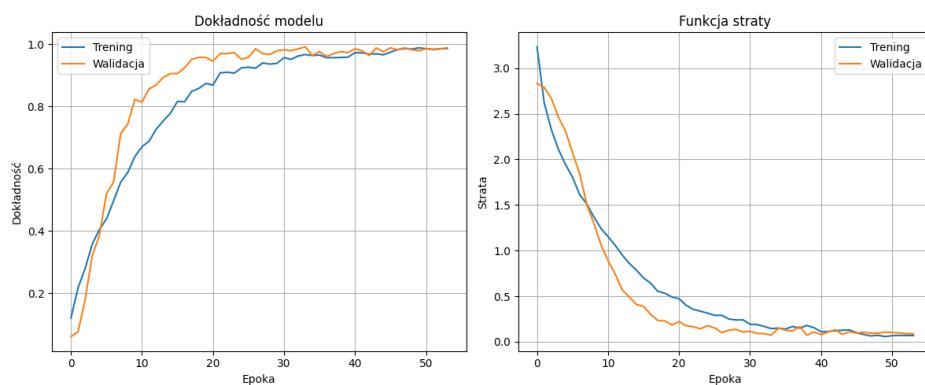
Architektura i strojenie sieci 1D-CNN+LSTM

Wybrany model składa się z warstw konwolucyjnych (1D-CNN), które wydobywają lokalne wzorce czasowe z sygnału, oraz warstw LSTM pozwalających na uchwycenie długoterminowych zależności czasowych w oknie sekwencji. Zastosowałem rozbudowaną regularyzację (normalizacje, dropouts, wczesne zatrzymywanie, dynamiczną zmianę tempa nauki) oraz dokładnie dobrałem długość sekwencji, rozmiary filtrów i głębokość sieci. Model okazał się bardzo elastyczny, a wyciągnięte cechy pozwoliły na skuteczne rozróżnianie nawet podobnych gestów.

Analiza i wizualizacja wyników

W celu pełnej oceny działania modelu przygotowałem szereg wykresów i analiz. Poniżej każdy wykres został opisany szczegółowo wraz z informacją o tym, co znajduje się na osiach, interpretacją i wyciągniętymi wnioskami.

Wykres 1: Przebieg uczenia modelu



Rysunek 16: Historia uczenia modelu.

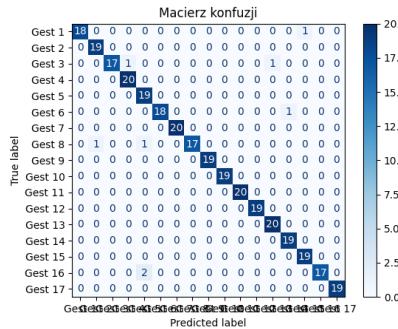
Opis: Wykres przedstawia przebieg uczenia się modelu 1D-CNN+LSTM: po lewej znajduje się krzywa dokładności (accuracy), a po prawej przebieg funkcji straty (loss) zarówno dla zbioru treningowego, jak i walidacyjnego.

Oś X: Numer epoki (epochs, kolejne przebiegi uczenia modelu).

Oś Y: Dla wykresu po lewej — dokładność (accuracy, wartość od 0 do 1); dla wykresu po prawej — wartość funkcji straty (loss).

Wnioski: Model bardzo szybko osiąga wysoką dokładność zarówno na zbiorze treningowym, jak i walidacyjnym. Już po kilku epokach dokładność przekracza 90%, a po około 20 epokach stabilizuje się na poziomie powyżej 95%. Różnice pomiędzy przebiegiem dla treningu i walidacji są minimalne, co świadczy o dobrej generalizacji i braku przeuczenia. Funkcja straty systematycznie maleje i również osiąga bardzo niskie wartości, potwierdzając skuteczność procesu uczenia. Brak gwałtownych wzrostów straty czy spadków dokładności wskazuje, że zastosowane mechanizmy regularyzacyjne (early stopping, redukcja tempa nauki) i architektura modelu są dobrze dobrane do problemu.

Wykres 2: Macierz konfuzji



Rysunek 17: Macierz konfuzji dla rozpoznawanych gestów.

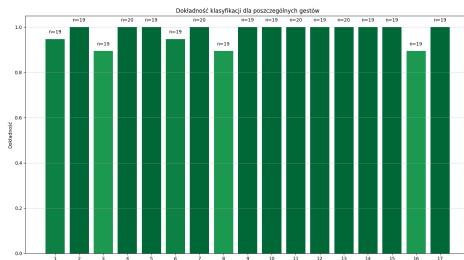
Opis: Macierz konfuzji obrazuje szczegółowo, jak model radzi sobie z klasyfikacją poszczególnych gestów. Każda komórka pokazuje liczbę przypadków, w których gest rzeczywisty został zaklasyfikowany do danej klasy. Diagonalne wartości reprezentują poprawne klasyfikacje.

Oś X: Klasa gestu przewidziana przez model (predykcja).

Oś Y: Rzeczywista klasa gestu (prawdziwa etykieta).

Wnioski: Praktycznie wszystkie próbki zostały poprawnie zaklasyfikowane — zdecydowana większość wartości znajduje się na przekątnej macierzy. Jedynie w przypadku gestów 1 i 16 pojawiły się pojedyncze błędne klasyfikacje, co widać po niewielkich wartościach poza główną przekątną. Oznacza to, że model skutecznie rozróżnia nawet bardzo liczne klasy (aż 17 różnych gestów), a większość błędów jest sporadyczna i nie wykazuje regularnych pomyłek pomiędzy konkretnymi parami gestów.

Wykres 3: Dokładność klasyfikacji poszczególnych gestów



Rysunek 18: Dokładność klasyfikacji dla każdego gestu.

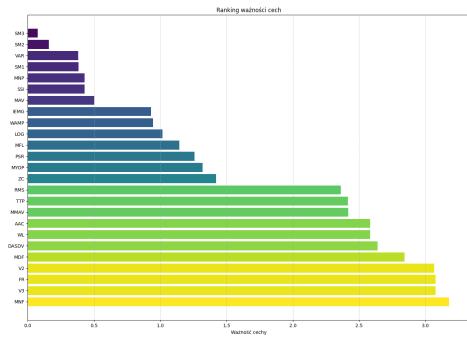
Opis: Każdy słupek przedstawia dokładność klasyfikacji osobno dla każdego gestu. Nad słupkami podano liczbę próbek testowych w każdej klasie (n). Kolorystyka słupków odzwierciedla wartość dokładności.

Oś X: Numer gestu (każda kolumna to inny gest).

Oś Y: Dokładność klasyfikacji (proporcja poprawnych klasyfikacji dla danego gestu, od 0 do 1).

Wnioski: Model osiąga niemal perfekcyjną skutecznosć dla większości gestów — dokładność klasyfikacji dla 14 z 17 gestów wynosi 100%. Jedynie gesty 2, 3, 6, 8 i 16 mają nieco niższą skutecznosć, sięgającą jednak nadal powyżej 90%. Warto zauważać, że liczba próbek dla wszystkich gestów jest bardzo zbliżona (19 lub 20), co świadczy o dobrze zbalansowanym zbiorze testowym. Sporadyczne spadki dokładności dla pojedynczych gestów mogą wynikać z ich podobieństwa do innych ruchów lub większej zmienności sygnału u poszczególnych użytkowników.

Wykres 4: Ranking ważności cech



Rysunek 19: Ranking cech sygnału według ich wpływu na klasyfikację.

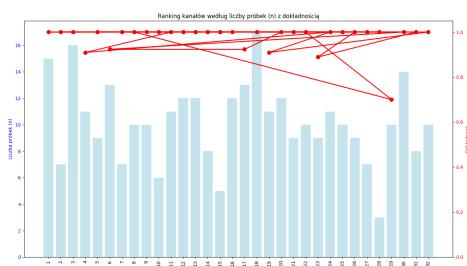
Opis: Wykres przedstawia uszeregowanie cech sygnału EMG według ich znaczenia dla prawidłowej klasyfikacji gestów. Każda cecha została oceniona na podstawie wpływu na rozróżnianie klas (efektu rozdzielczości pomiędzy gestami).

Oś X: Ważność cechy (liczbową miarą wpływu danej cechy na klasyfikację, im wyższa tym lepiej rozróżnia gesty).

Oś Y: Nazwa cechy (każdy wiersz to inna cecha sygnału).

Wnioski: Najważniejszymi cechami okazały się parametry częstotliwościowe (MNF, V3, FR, V2, MDF) oraz wybrane miary amplitudy i zmienności (DASDV, WL, AAC). Cechy takie jak SM3, SM2, VAR, SM1 mają najmniejszy wpływ na rozpoznawanie gestów. Wskazuje to, że analiza widma sygnału EMG oraz cechy opisujące zmienność są kluczowe dla skutecznej klasyfikacji, podczas gdy niektóre proste miary statystyczne mają marginalne znaczenie. Wyniki sugerują możliwość uproszczenia modelu przez eliminację cech o najmniejszej ważności.

Wykres 5: Ranking kanałów według liczby próbek (n) z dokładnością



Rysunek 20: Ranking kanałów według liczby próbek z dokładnością.

Opis: Wykres przedstawia ranking kanałów (elektrod) według liczby próbek (słupki, lewa oś) oraz odpowiadającej im dokładności klasyfikacji (linia, prawa oś). Każdy kanał reprezentuje oddzielną elektrodę zbierającą sygnał EMG.

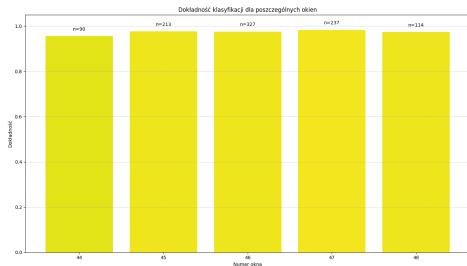
Oś X: Numer kanału (elektrody).

Oś Y lewa: Liczba próbek przypisanych do danego kanału.

Oś Y prawa: Dokładność klasyfikacji dla danego kanału (od 0 do 1).

Wnioski: Większość kanałów charakteryzuje się bardzo wysoką dokładnością (bliską 1.0), niezależnie od liczby próbek. Mimo że niektóre kanały mają znacznie mniej danych, nie przekłada się to na pogorszenie skuteczności klasyfikacji. Spadki dokładności obserwowane są sporadycznie i dotyczą pojedynczych kanałów o najmniejszej liczbie próbek, co sugeruje, że zapewnienie minimalnej liczby próbek dla każdej elektrody pozwala osiągnąć stabilną jakość klasyfikacji w systemie wielokanałowym.

Wykres 6: Dokładność klasyfikacji dla poszczególnych okien



Rysunek 21: Dokładność klasyfikacji dla poszczególnych okien czasowych.

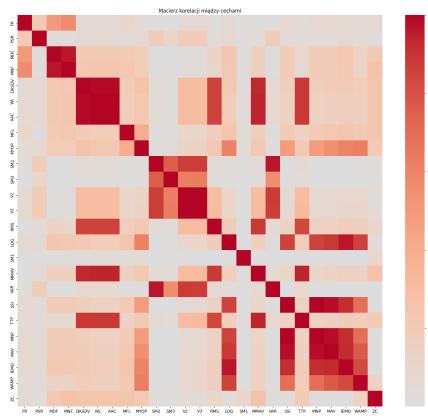
Opis: Wykres prezentuje dokładność klasyfikacji dla poszczególnych okien czasowych, czyli fragmentów sygnału EMG analizowanych przez model. Nad słupkami podano liczbę próbek w danym oknie.

Oś X: Numer okna czasowego.

Oś Y: Dokładność klasyfikacji dla danego okna (od 0 do 1).

Wnioski: Skuteczność klasyfikacji w każdym z analizowanych okien jest bardzo wysoka — dla wszystkich okien przekracza 95%. Najlepsze wyniki osiągane są dla okien, które zawierają najwięcej próbek, jednak nawet najmniej liczne okno nie powoduje istotnego spadku jakości. To potwierdza, że model jest odporny na niewielkie różnice w liczbie danych pomiędzy oknami i efektywnie wykorzystuje dostępne informacje w całym przebiegu sygnału.

Wykres 7: Macierz korelacji między cechami



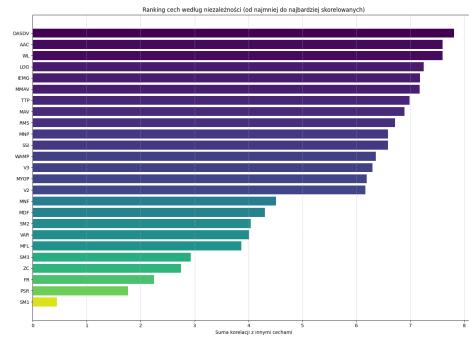
Rysunek 22: Macierz korelacji pomiędzy cechami sygnału.

Opis: Macierz korelacji pomiędzy wszystkimi wyekstrahowanymi cechami sygnału EMG. Każdy piksel przedstawia wartość bezwzględnej korelacji pomiędzy dwiema cechami. Kolor czerwony oznacza wysoką korelację, szary — niską.

Oś X/Y: Nazwy cech sygnału (ta sama lista na obu osiach).

Wnioski: Widoczne są wyraźne grupy silnie skorelowanych cech, zwłaszcza wśród cech częstotliwościowych i niektórych miar amplitudy. Część cech wykazuje niewielką korelację z pozostałymi, co sugeruje ich dużą niezależność informacyjną i potencjalną wartość dla dalszej optymalizacji modelu (np. DASDV, AAC lub LOG). Przyszła selekcja cech powinna rozważyć eliminację nadmiarowych, silnie skorelowanych parametrów, by uprościć model bez utraty skuteczności.

Wykres 8: Ranking cech według niezależności



Rysunek 23: Ranking cech według niezależności (od najmniej do najbardziej skorelowanych).

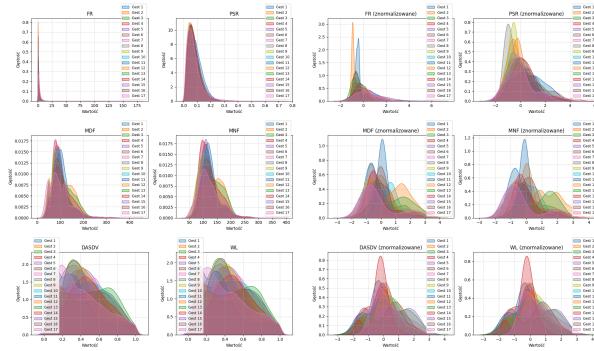
Opis: Wykres słupkowy prezentuje ranking cech według niezależności. Im mniejsza suma korelacji cechy z innymi, tym wyższa jej pozycja w rankingu (lewa strona wykresu). Kolejność cech pozwala wyłonić te, które wnoszą unikalną informację do procesu klasyfikacji.

Oś X: Suma korelacji z innymi cechami (im mniejsza, tym bardziej niezależna cecha).

Oś Y: Nazwa cechy.

Wnioski: Najbardziej niezależne cechy (DASDV, AAC, WL, LOG, IEMG, MMAV) mogą być kluczowe dla dalszego rozwoju systemu i odporności na przeuczenie. Cecha SM1 wyróżnia się jako najbardziej redundantna, co uzasadnia jej potencjalną eliminację w dalszych iteracjach projektu. Ranking ten jest cenną wskazówką przy wyborze zestawu cech do kolejnych eksperymentów modelu.

Porównanie rozkładów cechy ZC przed i po normalizacji



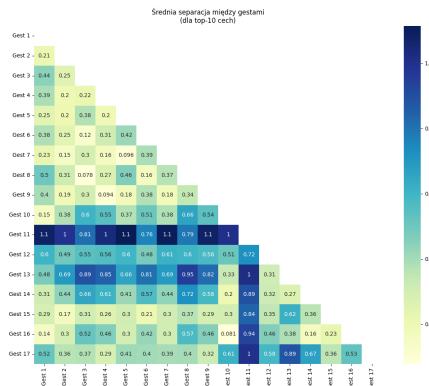
Rysunek 24: Porównanie rozkładów gęstości przykładowych cechy ZC przed normalizacją (lewy panel) i po normalizacji (prawy panel) dla wszystkich gestów.

Opis: Wykresy przedstawiają rozkład wartości cechy ZC (Zero Crossing) dla wszystkich gestów przed (lewy panel) i po normalizacji (prawy panel). Każda krzywa reprezentuje rozkład tej cechy dla jednej klasy gestu.

Oś X: Wartość cechy ZC (przed lub po normalizacji). **Oś Y:** Gęstość — prawdopodobieństwo wystąpienia danej wartości cechy.

Interpretacja: Przed normalizacją rozkłady wartości ZC dla różnych gestów są bardzo mocno na siebie nałożone, przez co sieć neuronowa ma trudność z rozpoznawaniem gestów na podstawie tej cechy. Po normalizacji rozkłady są wyraźnie bardziej rozseparowane — nakładają się w mniejszym stopniu, dzięki czemu sieć może łatwiej wyodrębnić różnice między gestami i poprawnie je klasyfikować. Na podstawie tych danych został stworzony wykres ze średnią separacją między gestami (rozdział poniżej).

Średnia separacja między gestami (po normalizacji, top-10 cech)



Rysunek 25: Macierz średniej separacji pomiędzy gestami po normalizacji, wyznaczona dla 10 cech najlepiej rozróżniających klasy.

Opis: Wykres przedstawia macierz średniej separacji (odległości) pomiędzy rozkładami poszczególnych gestów, wyznaczoną na podstawie top-10 najważniejszych cech po normalizacji. Każda komórka macierzy zawiera wartość miary rozdzielnosci pomiędzy dwoma gestami — im wyzsza wartość, tym rozkłady tych gestów są bardziej odseparowane i łatwiejsze do rozróżnienia przez klasyfikator. Skala kolorów od żółtego (mała separacja) do ciemnoniebieskiego (wysoka separacja) pozwala szybko zidentyfikować pary gestów trudne i łatwe do rozróżnienia.

Oś X i Y: Numery gestów (klas).

Wnioski:

- Najwyższą separację wykazują **Gest 11**, **Gest 13** oraz **Gest 12**, których rozkłady są najbardziej oddalone od pozostałych. Są one tym samym najlepiej rozróżnialne przez sieć neuronową.
- Gesty takie jak **Gest 2**, **Gest 6** czy **Gest 7** wykazują niską separację wobec wielu innych gestów, co może utrudniać ich jednoznaczną klasyfikację i prowadzić do pomyłek.
- Macierz potwierdza, że normalizacja i właściwy wybór cech pozwala na skuteczne rozdzielenie większości gestów, szczególnie tych o wysokich wartościach w macierzy separacji.

Integracja z symulacją w ROS1

Po dopracowaniu modelu skupiam się na jego integracji z systemem ROS1, współpracując z zespołem odpowiedzialnym za symulację wirtualnej ręki. Największymi wyzwaniami są:

- zapewnienie możliwie niskiego opóźnienia przekazu rozpoznanych gestów do modułu sterowania kończyną,
- stabilna komunikacja pomiędzy systemem analizy sygnału a symulatorem ruchu,
- przetestowanie całego pipeline'u w warunkach zbliżonych do rzeczywistych.

Celem jest osiągnięcie płynnego i wiarygodnego odwzorowania ruchów ręki na podstawie sygnałów EMG w czasie rzeczywistym.

Plany na dalsze etapy

W najbliższym czasie planuję:

- Dalszą optymalizację modelu na podstawie wyników z integracji,
- Testy systemu w pełnej symulacji,

Najważniejsze wyniki i rankingi

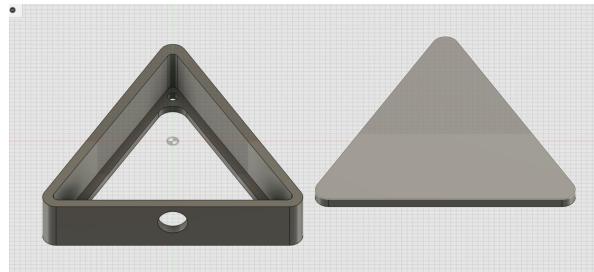
Poniżej przedstawiam podsumowanie kluczowych rezultatów:

- **Najwyższe dokładności klasyfikacji osiągnięto dla gestów:** Gest 11, Gest 13, Gest 12, Gest 1, Gest 10 (na podstawie macierzy separacji i wykresu dokładności klasyfikacji).
- **Najważniejsze cechy sygnału według wpływu na klasyfikację:** MNF, V3, FR, V2, MDF (na podstawie rankingu ważności cech).
- **Najbardziej wartościowe kanały:** Kanał 5, Kanał 8, Kanał 12, Kanał 3, Kanał 10 (na podstawie rankingu kanałów według liczby próbek i dokładności).
- **Najbardziej niezależne cechy:** DASDV, AAC, WL, LOG, IEMG (na podstawie rankingu cech według niezależności).

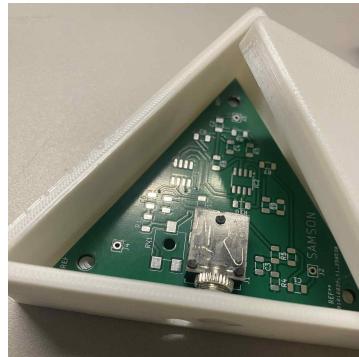
Kinga Michalak

Projekt pudełka i pokrywki dla elektrody

W ramach realizacji tego etapu projektu zaprojektowano pierwszą wersję pudełka na płytę z elektrodą oraz pokrywkę w programie Fusion 360 (patrz rysunek 26). Po wydrukowaniu i ocenie prototypu wykonano jeszcze dwie wersje, gdzie ostatnia została wykonana w oparciu o rzeczywistą płytę (patrz rysunek 27).



Rysunek 26: Wersja pierwsza



Rysunek 27: Wersja druga

Mikołaj Strużykowski

Integracja z mikrokontrolerem

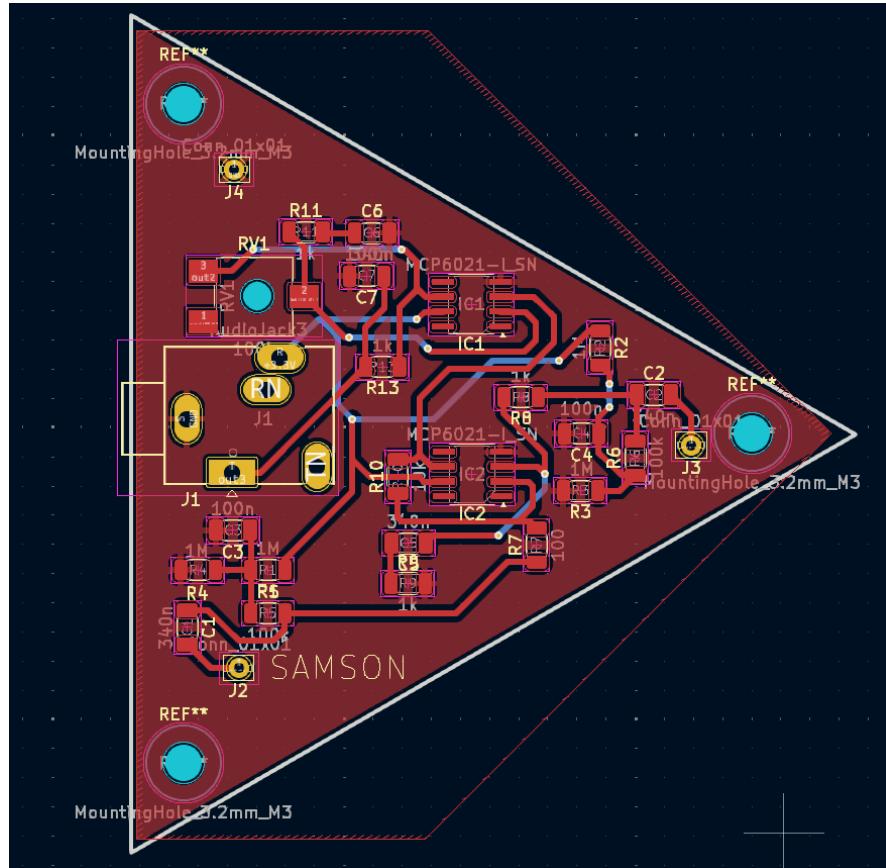
Integracja z mikrokontrolerem przebiegła pomyslnie. Przy pierwszym połączeniu go z elektrodą występowały liczne błędy związane z przesyłaną ramką danych, jednak po drobnych poprawkach udało się odczytać pierwsze sygnały.

Ich wizualizacja oraz dalsza analiza znajdują się w części Pauli dotyczącej sygnałów.

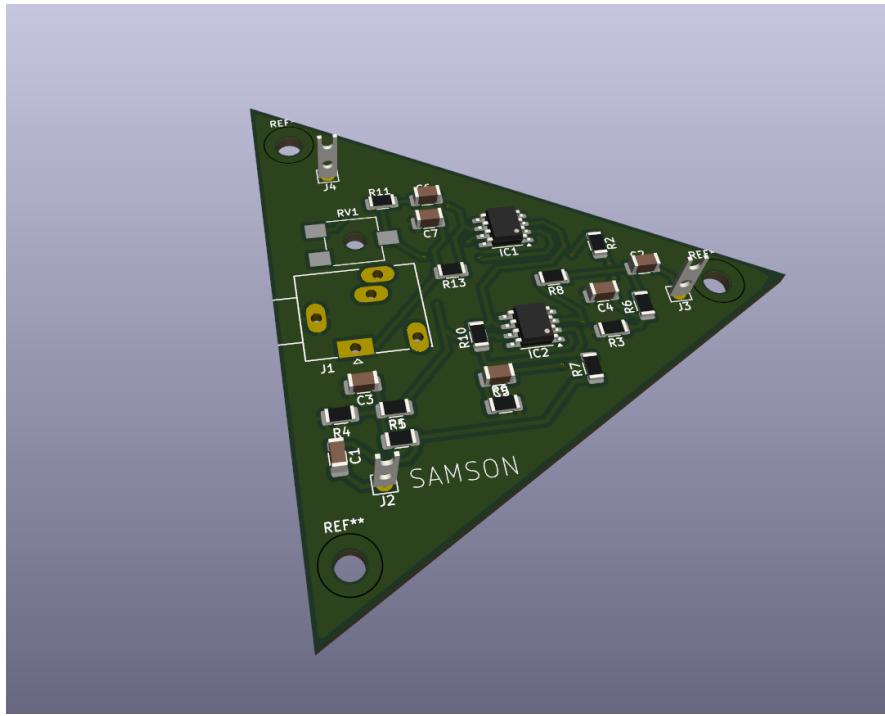
Projekt płytki drukowanej

Przed przystąpieniem do projektowania płytki drukowanej, zostały przeprowadzone testy jakości odczytywanych sygnałów. Dzięki temu mogły zostać wykryte błędy w projekcie przed zamówieniem płytek PCB.

Działanie urządzenia było na tyle spodziewane i zadowalające, że projekt nie został zmieniony przed ostatecznym modelem. Poniżej znajduje się schemat oraz model 3D.



Rysunek 28: Elektroda SAMSON model .brd



Rysunek 29: Elektroda SAMSON model 3D

Płytki PCB trafiły już w ręce zespołu, w najbliższym tygodniu rozpoczęte zostanie pobieranie sygnałów od jego członków. Dotychczasowe pomiary z elektrody złożonej na płytce prototypowej były znacznie zaszumione. Mamy nadzieję, że lepiej skonstruowane elektrody pozwolą na zredukowanie tych błędów. Do wykonania projektu został użyty program KiCad. Trójkątny kształt płytek pozwala na równomierne rozłożenie srebrnych sond na skórze.

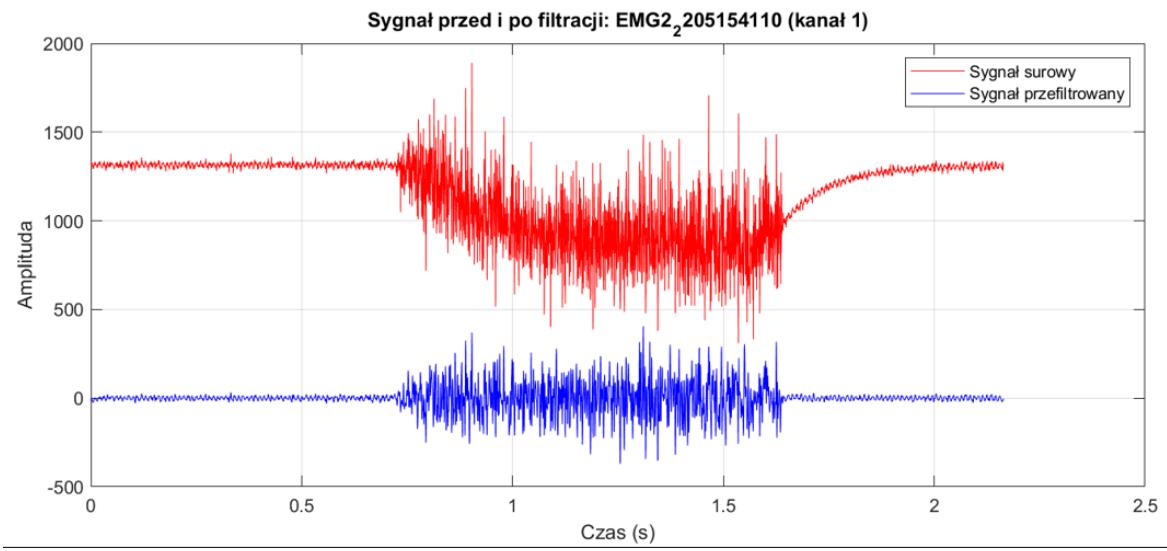
Paulina Piorun

Odczyt danych z próbnych pomiarów

W ramach tego etapu przygotowano skrypt do odczytu danych zapisanych do pliku pochodzących z rzeczywistych pomiarów. Do testów wykorzystano pliki zawierające testowe dane, których format odpowiada docelowej formie odczytu pomiarów rejestrujących wykonanie gestu. Ustalono, że dane będą zapisywane w pliku w formacie .txt, w którym każda linia będzie zawierała pomiary z kolejnych kanałów w danym momencie czasu. W efekcie pomiar z każdego kanału będzie stanowił oddzielną kolumnę danych.

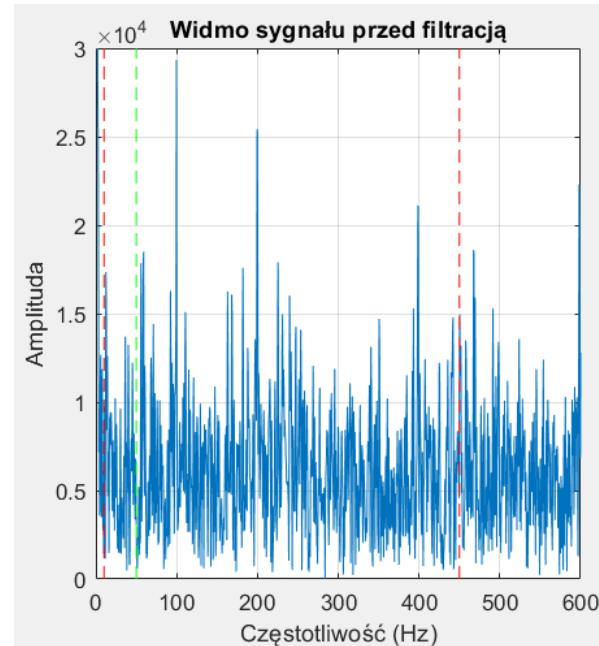
Filtracja i analiza częstotliwościowa

Odczytane dane zostały poddane filtracji. Filtracja została przeprowadzona z użyciem filtra pasmowo-przepustowego o zakresie 10 Hz do 450 Hz oraz filtra wycinającego (notch) o częstotliwości 50 Hz. Tak dobrane parametry filtracji umożliwiają redukcję zakłóceń pochodzących z sieci energetycznej oraz eliminację składowych niskoczęstotliwościowych niezwiązań z właściwym sygnałem EMG. Na rysunku 30 przedstawiono porównanie sygnału przed filtracją i po filtracji. Wartości amplitudy odpowiadają surowym danym nieprzeskalowanym do fizycznej wielkości jaką jest napięcie.

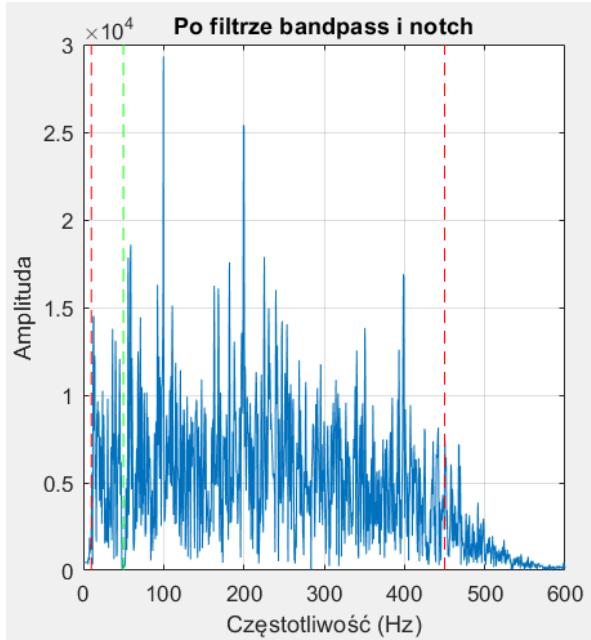


Rysunek 30: Testowe próbki rzeczywistego sygnału

Analiza częstotliwościowa sygnału wykazała (patrz rysunek 31), że znaczący wpływ na kształt sygnału mają składowe o niskich częstotliwościach, które w procesie filtracji zostają odfiltrowane. Charakterystyka widmowa po filtracji została przedstawiona na rysunku 32.



Rysunek 31: Widmo sygnału przed filtracją – dane testowe



Rysunek 32: Widmo sygnału po filtracji – dane testowe

Podsumowanie

Podczas tego etapu prac dostosowano program do odczytu danych pochodzących z rzeczywistych pomiarów testowych, zapewniając możliwość ich filtracji i analizy częstotliwościowej. Przygotowano mechanizm umożliwiający wizualizację sygnału surowego oraz przefiltrowanego, co pozwala na szybką weryfikację poprawności przetwarzania. Dalsze testy i modyfikacje będą możliwe po otrzymaniu pomiarów rejestrujących konkretne gesty. Wtedy poza filtracją zostaną uruchomione funkcje obliczające wybrane cechy sygnału, które będą podstawą do trenowania i walidacji algorytmów klasyfikujących gesty.

Rozalia Nowicka

W ramach realizacji projektu podjęto działania dotyczące dostosowania komunikacji mikrokontrolera z komputerem do trzech kanałów oraz rozpoczęto testy pomiaru rzeczywistego sygnału EMG. Wykonywane zadania wynikały z ustalonego wcześniej harmonogramu.

Konfiguracja ADC i obsługa trzech kanałów

Skonfigurowano przetwornik ADC do pracy z trzema kanałami przy wykorzystaniu pojedynczego przetwornika – decyzja ta została podjęta z uwagi na łatwiejszą synchronizację danych. Kolejne kanały mierzone są sekwencyjnie, nie powinno to jednak wpływać na jakość pomiarów. Dostosowano format ramki danych zgodnie ze wcześniejszymi założeniami. Ustawiono częstotliwość taktowania na 80 MHz, co pozwoliło zwiększyć dokładność pomiarów poprzez wydłużenie czasu próbkowania (640,5 cykli zegara), bez istotnej utraty szybkości. Rozważono częstotliwość pomiaru 2 kHz, jednak ze względu na konieczność podniesienia wtedy prędkości UART i potencjalnej utraty pakietów, zrezygnowano z tego pomysłu i pozostało przy częstotliwości 1 kHz. Wprowadzono również sprzętowe uśrednianie wyników (z 32 próbek), które przyczynia się do zwiększenia precyzji przy zachowaniu właściwej szybkości działania.

Transmisja danych

Zrealizowano transmisję UART z mikrokontrolera do komputera, utrzymując prędkość transmisji równą 15 200 bps. Zmodyfikowano odpowiednio program do odbioru danych na komputerze, tak aby był dostosowany do trzech kanałów. Dane zapisywane są do pliku w następującym formacie:

```
adc1;adc2;adc3  
adc1;adc2;adc3  
...
```

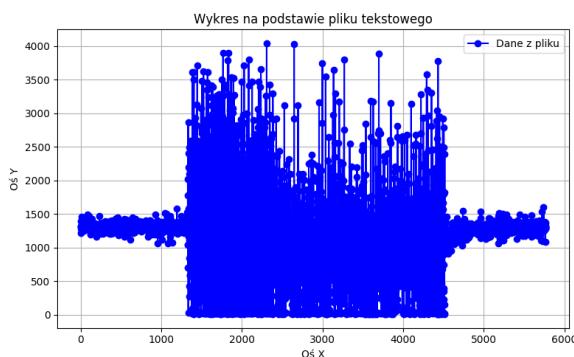
Prosta struktura ułatwia przetwarzanie pliku. Nazwa pliku nadawana jest zgodnie z datą i godziną rozpoczęcia danego pomiaru.

Sekwencja pomiarów

Podczas testowania wstępnych wersji programu, zauważono niefunkcjonalną obsługę sekwencji pomiarów. W celu poprawy tego aspektu opracowano działanie przycisku. Każde naciśnięcie powoduje zmianę statusu pomiaru (pomiar lub brak pomiaru). Natomiast program odbierający dane na komputerze jest uruchomiony podczas trwania całej sesji pomiarów, zapisując dane z każdego pomiaru do osobnego, odpowiednio nazwanego pliku. Zrealizowano ten mechanizm poprzez uwzględnienie dwóch dodatkowych ramek składających się z dziewięciu jednakowych elementów każda, oznaczających odpowiednio rozpoczęcie ($0xCC$) oraz zakończenie ($0x99$) pomiaru.

Testy rzeczywistego sygnału

Wykonano szereg prób odbioru rzeczywistego sygnału EMG, podczas których wykryto błędy transmisji. Szczególne problemy zauważono z samą ramką danych. Podjęto próby jej naprawy, jednak z uwagi na niemożliwość przeprowadzenia kolejnych testów na rzeczywistym sygnale, testy wykonano z użyciem potencjometrów. Podczas ich przeprowadzania nie zauważono żadnych nieprawidłowości. Przykładowy pomiar wykonany na rzeczywistym sygnale z pominięciem ramki danych zobrazowano na rys. 33.



Rysunek 33: Przykładowy pomiar rzeczywistego sygnału EMG.

Maksymilian Tulewicz

Modelowanie dloni i symulacja w Gazebo

W ramach projektu stworzono model 3D dloni, który został zaprojektowany do symulacji w środowisku Gazebo. Głównym celem było uzyskanie wizualizacji zebranych sygnałów mio-elektrycznych, przetworzonych przez sieć neuronową, w postaci naturalnych gestów dloni. Model został oparty na proporcjach dloni użytkownika przy uwzględnieniu średnich długości palców u ludzi oraz anatomicznej struktury przegubów. Początkowo stworzono szczegółowy model w Autodesk Inventor, który następnie przekształcono do formatu URDF (Unified Robot Description Format), niezbędnego do pracy w systemie ROS.

Przekształcenie modelu na format URDF

Model 3D dloni, stworzony w Autodesk Inventor, został przekształcony do formatu URDF za pomocą różnych narzędzi. Pierwotnie próbowało wygenerować plik URDF za pomocą środowiska Blender z nakładką Phobos, jednak napotkano znaczące trudności związane z kompatybilnością formatów oraz niewystarczającym wsparciem dla złożonych modeli anatomicznych. Zmiana podejścia na środowisko Fusion 360 z wykorzystaniem nakładki Fusion2URDF okazała się kluczowa dla sukcesu projektu. Nakładka ta umożliwiła automatyczne wygenerowanie wymaganych plików symulacyjnych, w tym:

- Pliku URDF z pełną strukturą kinematyczną
- Plików meshowych (.stl) dla wizualizacji
- Plików konfiguracyjnych Gazebo (.gazebo)
- Plików materiałowych (.xacro)

```
1 <joint name="point_base_joint" type="revolute">
2   <origin xyz="-0.01955 -0.003981 0.073116" rpy="0 0 0"/>
3   <parent link="base_link"/>
4   <child link="point_base_link_1"/>
5   <axis xyz="1.0 0.0 0.0"/>
6   <limit upper="1.570796" lower="0.0" effort="100" velocity="100"/>
7 </joint>
8
9 <link name="point_base_link_1">
10   <inertial>
11     <origin xyz="-0.0035493370884052654 0.02169003855191419
12       0.00039656698602429097" rpy="0 0 0"/>
13     <mass value="0.1"/>
14     <inertia ixx="0.01" iyy="0.01" izz="0.01" ixz="0.0" iyz="0.0" ixy="0.0"/>
15   </inertial>
16   <visual>
17     <geometry>
18       <mesh filename="package://samson_final_description/meshes/
19         point_base_link_1.stl" scale="0.001 0.001 0.001"/>
20     </geometry>
21   </visual>
22 </link>
```

Listing 1: Fragment pliku URDF przedstawiający strukturę przegubu palca

Zmiana konstrukcji przegubu kciuka

Początkowo kciuk został zamodelowany z przegubem kulkowym (spherical joint), który lepiej oddawał naturalną anatomię ludzkiej dłoni. Okazało się jednak, że nakładka Fusion2URDF nie obsługiwała tego typu złożień poprawnie - przeguby kulkowe nie były konwertowane do odpowiadających im kombinacji przegubów obrotowych w formacie URDF. W związku z tym przegub kciuka został przeprojektowany na zwyczajny przegub obrotowy, zamontowany pod specjalnie obliczonym kątem:

```

1 <joint name="thumb_joint" type="revolute">
2   <origin xyz="-0.029809 0.007485 0.044426" rpy="0 0 0"/>
3   <parent link="base_link"/>
4   <child link="thumb_link_1"/>
5   <axis xyz="0.228002 -0.711691 0.664463"/>
6   <limit upper="0.0" lower="-1.570796" effort="100" velocity="100"/>
7 </joint>

```

Listing 2: Konfiguracja przegubu kciuka z niestandardową osią obrotu

Niestandardowa oś obrotu ‘xyz="0.228002 -0.711691 0.664463,’ umożliwiła zachowanie naturalnego ruchu kciuka przy jednocześnie kompatybilności z formatem URDF.

Problemy z niestabilnością modelu dloni

Po zimportowaniu modelu do Gazebo napotkano poważny problem niestabilności symulacji. Palce, po podłączeniu kontrolerów PID typu `effort_controllers/JointPositionController`, zaczęły wykazywać niepożądane oscylacje, które z czasem nasilały się i prowadziły do całkowitej destabilizacji modelu. Analiza problemu wykazała kilka potencjalnych przyczyn:

1. Nieodpowiednie wartości PID kontrolerów
2. Błędne pozycje startowe przegubów względem ich limitów
3. Niewłaściwe momenty bezwładności elementów
4. Konflikty w detekcji kolizji (self-collision)

Pierwsze próby rozwiązymania problemu koncentrowały się na dostosowaniu wartości PID dla przegubów. Wartości te zostały systematycznie testowane z wykorzystaniem metody prób i błędów:

```

1 thumb_joint_position_controller:
2   type: effort_controllers/JointPositionController
3   joint: thumb_joint
4   pid: {p: 0.1, i: 0.0, d: 0.01} # Zmiany w tej linii

```

Listing 3: Przykład konfiguracji kontrolera PID w pliku controller.yaml

Analiza i rozwiązywanie problemów stabilności

Dogłębna analiza ujawniła, że główną przyczyną niestabilności były niewłaściwie dobrane momenty bezwładności wygenerowane automatycznie przez wtyczkę Fusion2URDF. Wartości te, wynoszące zaledwie ‘3e-06’ do ‘7e-06’ kgm², były zbyt małe dla stabilnej symulacji numerycznej w Gazebo. Problem pogłębiała niewielka masa komponentów (0.011-0.24 kg) oraz arbitralnie dobrane właściwości materiału.

Dodatkowo zidentyfikowano problem z pozycjami startowymi przegubów - niektóre z nich (szczególnie kciuk i końcówki palców) startowały na granicy swoich limitów kinematycznych, co powodowało natychmiastowe próby korekcji przez kontrolery PID.

```

1 <!-- Oryginalne wartości (niestabilne) -->
2 <inertial>
3   <mass value="0.030406967296379917"/>
4   <inertia ixx="3e-06" iyy="2e-06" izz="2e-06" ixy="0.0" iyz="-1e-06" ixz="0.0" />
5 </inertial>
6
7 <!-- Poprawione wartości (stabilne) -->
8 <inertial>
9   <mass value="0.1"/>
10  <inertia ixx="0.01" iyy="0.01" izz="0.01" ixy="0.0" iyz="0.0" ixz="0.0"/>
11 </inertial>

```

Listing 4: Porównanie oryginalnych i poprawionych momentów bezwładności

Zmiana momentów bezwładności na wartości rzędu ‘0.01’ kgm² oraz zwiększenie mas do ‘0.1-0.5’ kg całkowicie rozwiązały problem niestabilności.

System kontroli gestów

Po rozwiązaniu problemów z niestabilnością modelu opracowano kompletny system kontroli gestów składający się z trzech głównych komponentów:

1. Kontroler gestów (`hand_gesture_controller`) - node ROS nasłuchujący komend gestów i przekształcający je na sekwencje pozycji przegubów
2. Klient testowy (`test_gesture_client`) - narzędzie do szybkiego testowania gestów z linii komend
3. Interaktywne menu (`gesture_menu`) - przyjazny interfejs użytkownika z graficznym menu wyboru gestów

System implementuje płynne przejścia między pozycjami za pomocą interpolacji liniowej, co zapewnia naturalne wykonywanie gestów:

```
1 // Plynne przejście - 50 krok w przez 2 sekundy
2 int steps = 50;
3 double step_duration = 0.04; // 40ms per step
4
5 for(int step = 0; step <= steps; ++step) {
6     double progress = (double)step / (double)steps;
7
8     for(const auto& joint_cmd : gesture_positions) {
9         double current_pos = start_pos + progress * (target_pos - start_pos);
10
11         std_msgs::Float64 msg;
12         msg.data = current_pos;
13         joint_publishers_[controller].publish(msg);
14     }
15
16     ros::Duration(step_duration).sleep();
17 }
```

Listing 5: Fragment kodu implementujące płynne przejście między gestami

Zaimplementowane gesty

System obsługuje siedem predefiniowanych gestów, każdy z precyzyjnie dobranymi pozycjami przegubów:

- **open_hand** - wyprostowana dłoń z palcami skierowanymi do przodu
- **closed_fist** - zaciśnięta pięść z wszystkimi palcami zgodzonymi
- **peace_sign** - znak pokoju (palec wskazujący i środkowy wyprostowane)
- **pointing** - wskazywanie (tylko palec wskazujący wyprostowany)
- **thumbs_up** - kciuk w górę z pozostałymi palcami zaciśniętymi
- **ok_sign** - znak OK (kciuk i palec wskazujący tworzą okrąg)
- **rock_sign** - znak rock (kciuk, wskazujący i mały palec wyprostowane)

Pozycje te w aktualnym stadium są tylko prezentacją możliwości modelu – zostaną one dostosowane do oczywiście odczytywanych sygnałów.

Integracja z systemem klasyfikacji EMG

Opracowany system gestów został zaprojektowany z myślą o integracji z modułem klasyfikacji sygnałów EMG. Komunikacja odbywa się przez standardowy topic ROS `/hand_gesture_command`, co umożliwia łatwe podłączenie klasyfikatora sieci neuronowej:

```
1 # Pseudokod integracji z klasyfikatorem EMG
2 gesture_publisher = rospy.Publisher('/hand_gesture_command', String, queue_size
=10)
3
4 def on_emg_classification(predicted_gesture):
5     gesture_msg = String()
6     gesture_msg.data = predicted_gesture # np. "closed_fist"
7     gesture_publisher.publish(gesture_msg)
```

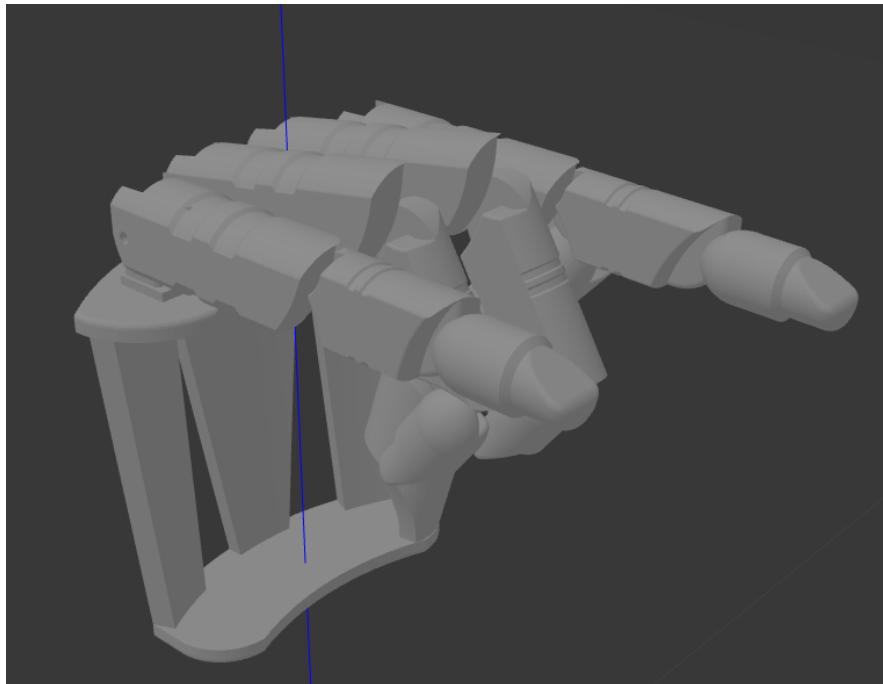
Listing 6: Przykład wysyłania komendy gestu z modułu klasyfikacji

Podsumowanie

Stworzony model 3D dłoni w środowisku Gazebo stanowi solidną podstawę do integracji z rzeczywistymi danymi EMG. Kluczowe osiągnięcia projektu obejmują:

- Stworzenie anatomicznie poprawnego modelu dłoni z 14 stopniami swobody
- Rozwiążanie problemów numerycznej stabilności symulacji
- Implementację systemu płynnej kontroli gestów
- Opracowanie intuicyjnego interfejsu użytkownika
- Przygotowanie architektury do integracji z klasyfikatorem EMG

Przeprowadzone korekty momentów bezwładności i optymalizacja parametrów kontrolerów PID umożliwiły uzyskanie stabilnego i przewidywalnego zachowania modelu. System ten może być bezpośrednio wykorzystany w dalszych etapach projektu do realizacji sterowania gestami w oparciu o dane z sieci neuronowej przetwarzającej sygnały EMG.



Rysunek 34: Symulacja ręki w Gazebo po implementacji poprawek stabilności.

Kamienie milowe – podsumowanie

Kamień Milowy 1 kwietnia – 2 maja

Większość zadań zaplanowanych do wykonania przed pierwszym kamieniem milowym zostało wykonyane. Cały zespół zakończył przygotowania teoretyczne do pracy nad projektem, a utworzone systemy są gotowe na przyjęcie rzeczywistych danych. W ciągu najbliższego tygodnia zostanie utworzony prototyp elektrody pozwalający na przetestowanie utworzonych części projektu oraz komunikacji między nimi. Zgodnie z harmonogramem zostanie wybrana grupa ochotników (spoza zespołu projektowego), którzy wezmą udział w zbieraniu danych mio-sygnalów do dalszej analizy.

Kamień Milowy 2 maja – 6 czerwca

Również tym razem zespołowi udało się wykonać większość zadań. Każdy jego członek przygotował oraz wykonał swoje zadania oraz rozwiązał napotkane problemy. Dodatkowo w przeciągu ostatnich tygodni osobne części projektu zostały ze sobą skomunikowane, a co najistotniejsze, mogliśmy pracować na danych zbieranych przez nas. Dzięki czemu przynajmniej niektóre funkcje projektu zostały przetestowane w warunkach, w których będą pracować docelowo.

3 Etap końcowy projektu

3.1 Zbieranie danych

W końcowym etapie projektów jego członkowie spotkali się by zebrać dane, które posłużyły do nauuczenia sieci neuronowej rozpoznawania gestów.

W procesie zbierania sygnałów wzięło udział trzech członków projektu oraz jeden ochotnik wybrany spośród studentów 3. roku Automatyki i Robotyki.

Do rozpoznawania zostały wybrane 3 gesty, które według wcześniejszych analiz były najbardziej wyróżniające się spośród reszty. Był to:



Rysunek 35: Wrists flexion

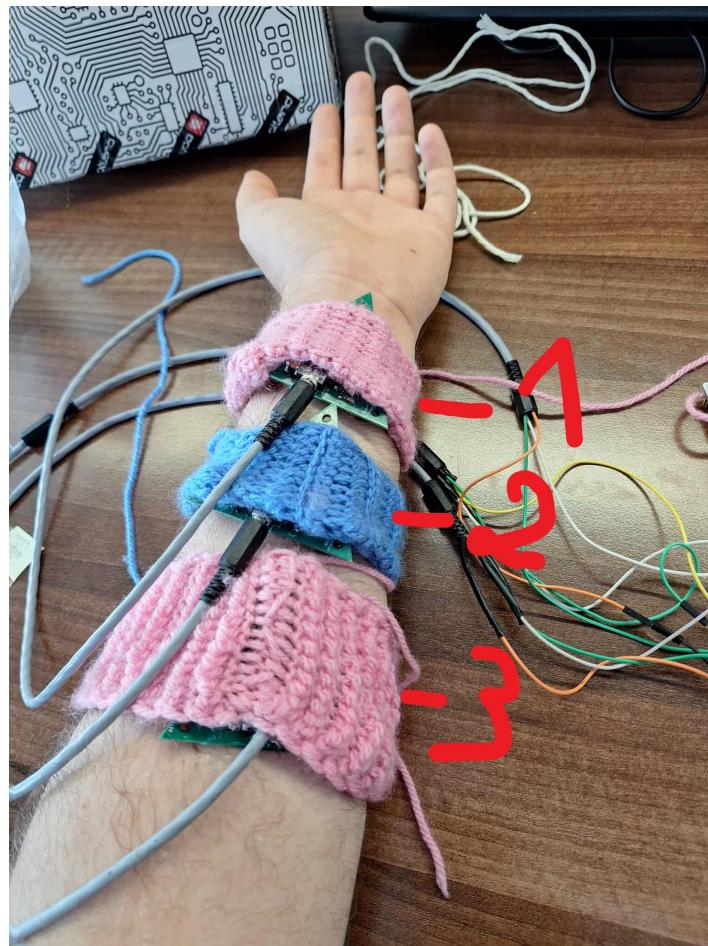


Rysunek 36: Wrists extension



Rysunek 37: Forearm pronation

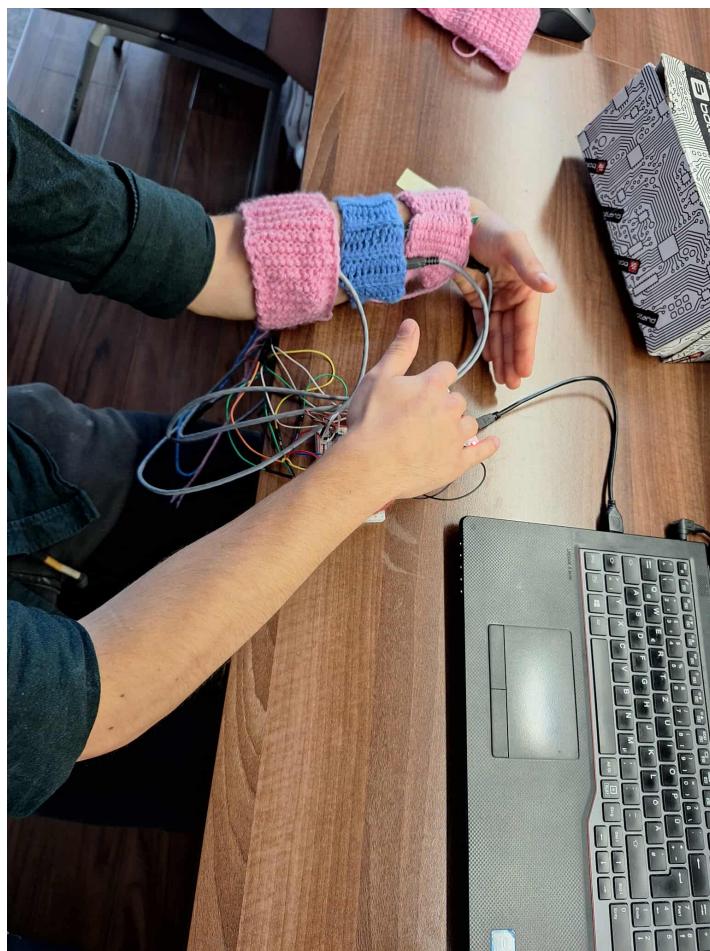
Każdy gest został wykonany 100 razy przez każdą z trzech wybranych osób co dało 300 pomiarów dla każdego gestu. Elektrody zostały oznaczone, tak żeby za każdym razem znajdowały się w tych samych miejscach ręki.



Rysunek 38: Umiejscowienie elektrod w trakcie pomiarów.

Elektrody były przytrzymywane przez akrylowe opaski co zapewniało kontakt sond ze skórą, gdy to nie wystarczało elektroda dodatkowo była przywiązywana bawełnianym sznurkiem do ręki. W celu poprawy przewodności skóry uczestników przy każdym pomiarze używany był żel kontaktowy do elektrod.

Każdy uczestnik samodzielnie rozpoczynał pomiar przez wcisnięcie przycisku na mikrokontrolerze, wykonanie gestu i zatrzymanie pomiaru kolejnym wcisnięciem przycisku. Przed wykonaniem każdego gestu uczestnik wracał do pozycji początkowej – ręki luźno położonej na stole. Pomiar był zatrzymywany w trakcie wykonywania gestu.



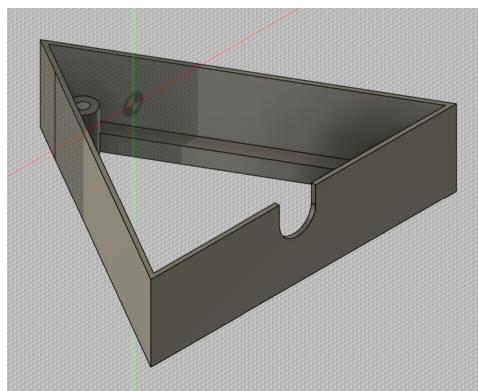
Rysunek 39: Przykładowy pomiar.

3.2 Obudowa elektrody

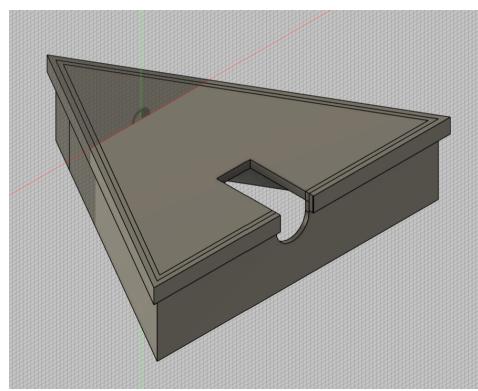
W końcowej fazie realizacji projektu opracowano dwie wersje obudowy dla płytki. Pierwsza z nich zawierała zbyt daleko idące modyfikacje — zbyt cienkie ścianki sprawiły, że konstrukcja nie zachowała wymaganych wymiarów podczas druku. Po wprowadzeniu odpowiednich poprawek udało się uzyskać w pełni funkcjonalną osłonę na elektrodę, która umożliwia zarówno wygodne wykonywanie pomiarów, jak i bezpieczne przechowywanie urządzenia.

Ostateczny projekt obudowy pozwala na trwałe zamocowanie płytki za pomocą śrubek oraz łatwe zamknięcie całej konstrukcji. Zarówno dolna część obudowy, jak i pokrywka, zostały wyposażone w otwór na przewód, co zapewnia skuteczną ochronę elementów elektronicznych bez ograniczania ich funkcjonalności. Dodatkowo, dzięki zastosowaniu podwyższeń pod płytke oraz otworowi w dnie obudowy, możliwe jest sprawne wyprowadzenie przewodów sygnałowych, co umożliwia stabilny i niezakłócony odczyt danych.

Finalna wersja konstrukcji została przedstawiona na rysunkach [40](#) i [41](#).



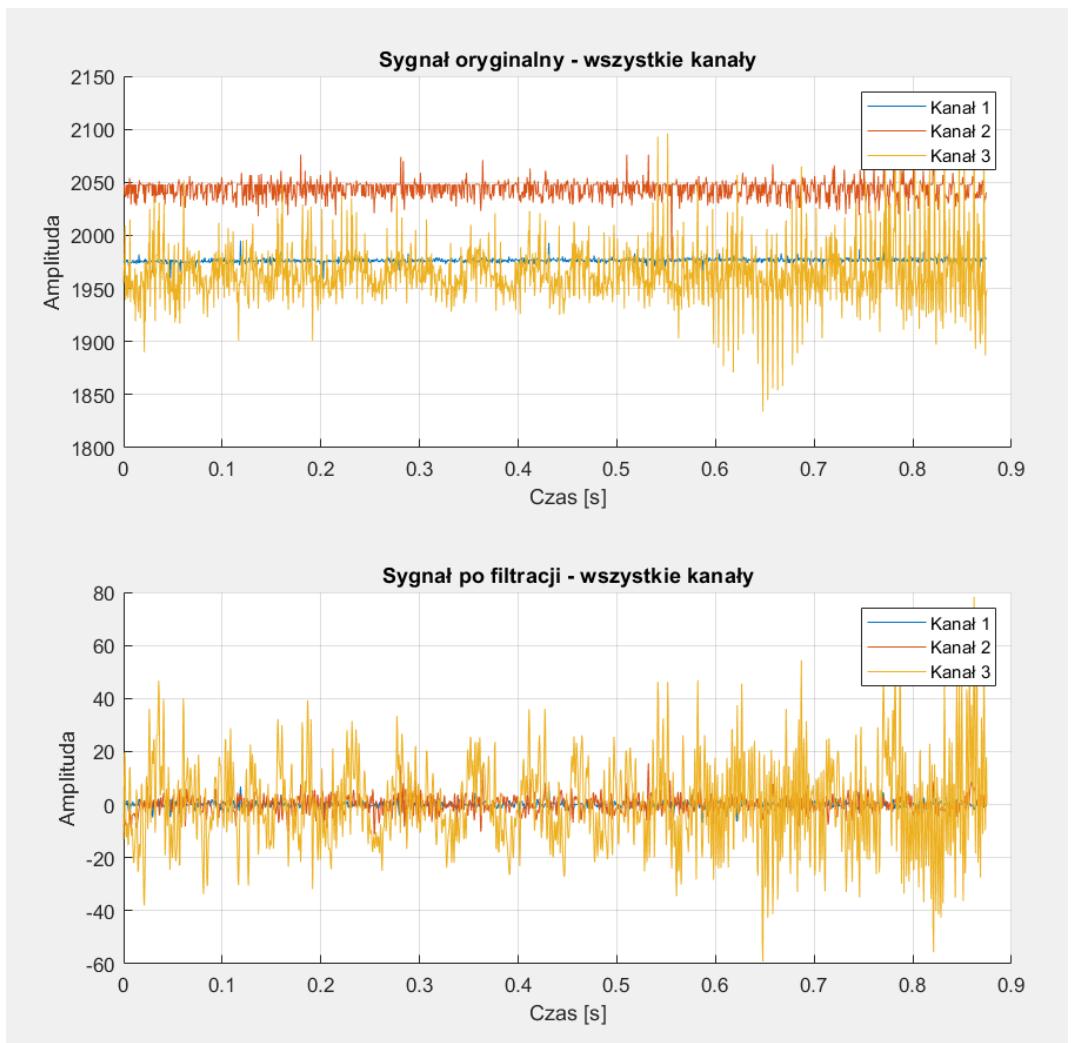
Rysunek 40: Projekt pudełka



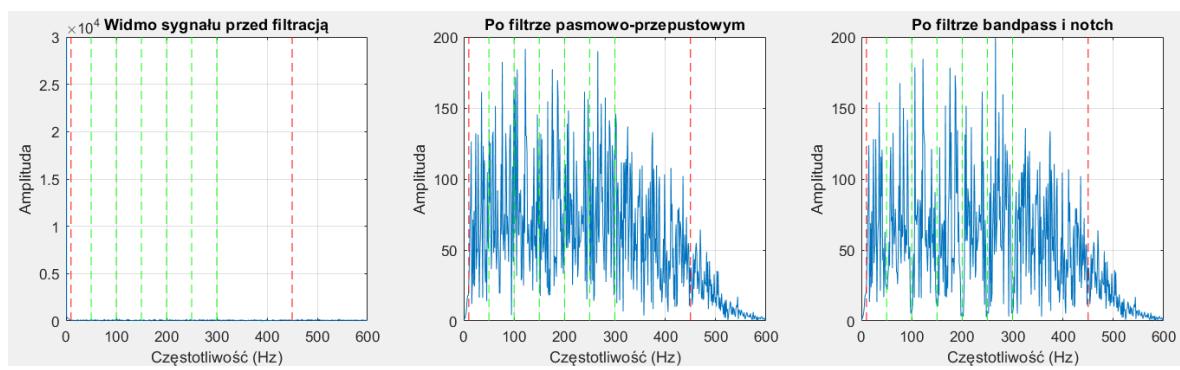
Rysunek 41: Projekt całej obudowy

3.3 Przetwarzanie sygnałów

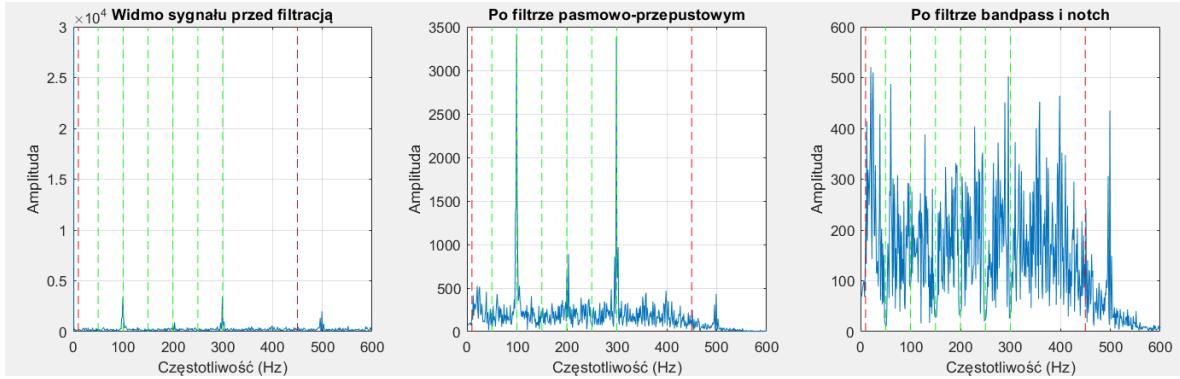
W końcowym etapie realizacji projektu wykonano zadania związane z przygotowaniem i przetwarzaniem danych potrzebnych do rozpoznawania gestów. Na początkowym etapie współpracując z resztą zespołu została sprawdzona jakość zbieranych danych w sposób wyrywkowy, co pozwoliło na wcześnie wychwycenie ewentualnych nieprawidłowości i zakłóceń w sygnale. Polegała ona na przeglądzie kształtu sygnału, charakterystyk częstotliwościowych oraz efektów filtracji. Zaobserwowano istotny wpływ poprawnego ułożenia wszystkich sond na jakość sygnału i ilość rejestrowanych zakłóceń.



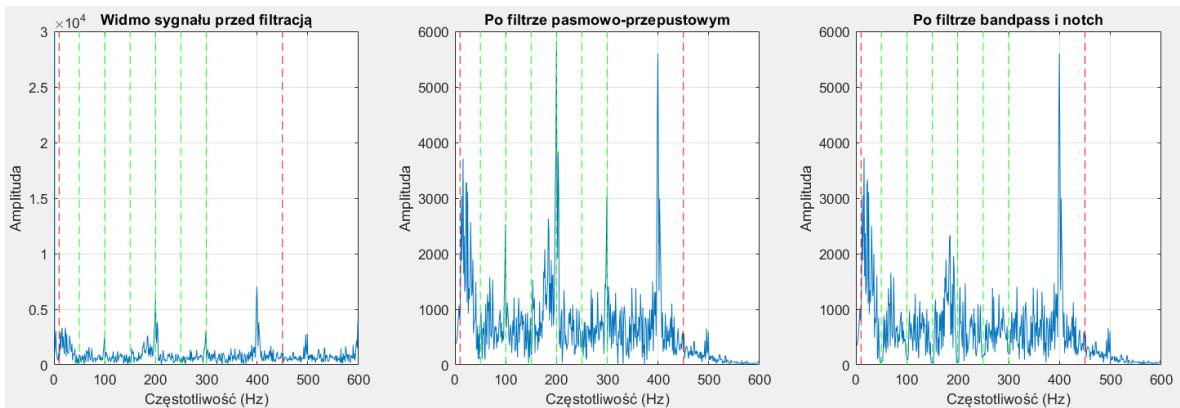
Rysunek 42: Sygnał przed filtracją i po filtracji



Rysunek 43: Charakterystyka częstotliwościowa – kanał 1

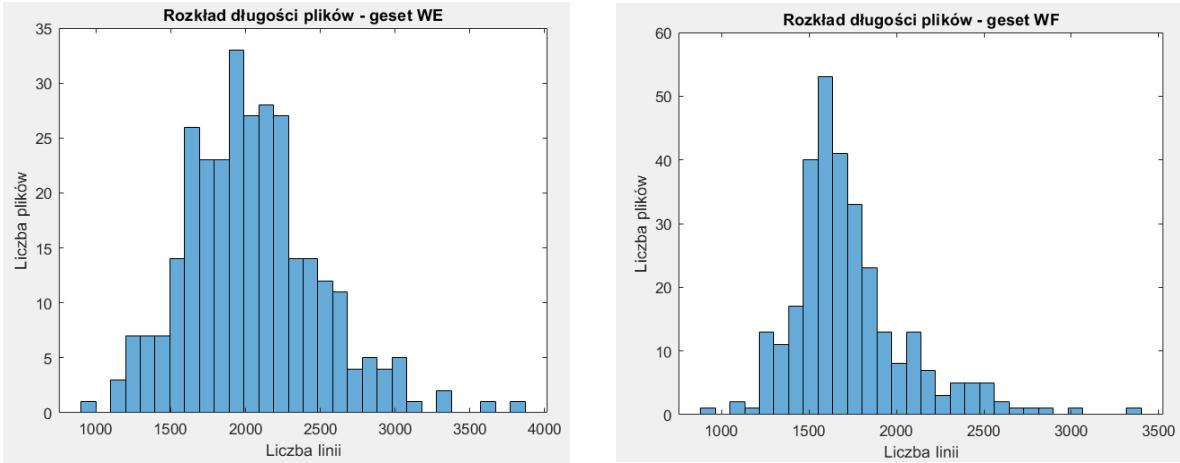


Rysunek 44: Charakterystyka częstotliwościowa – kanał 2

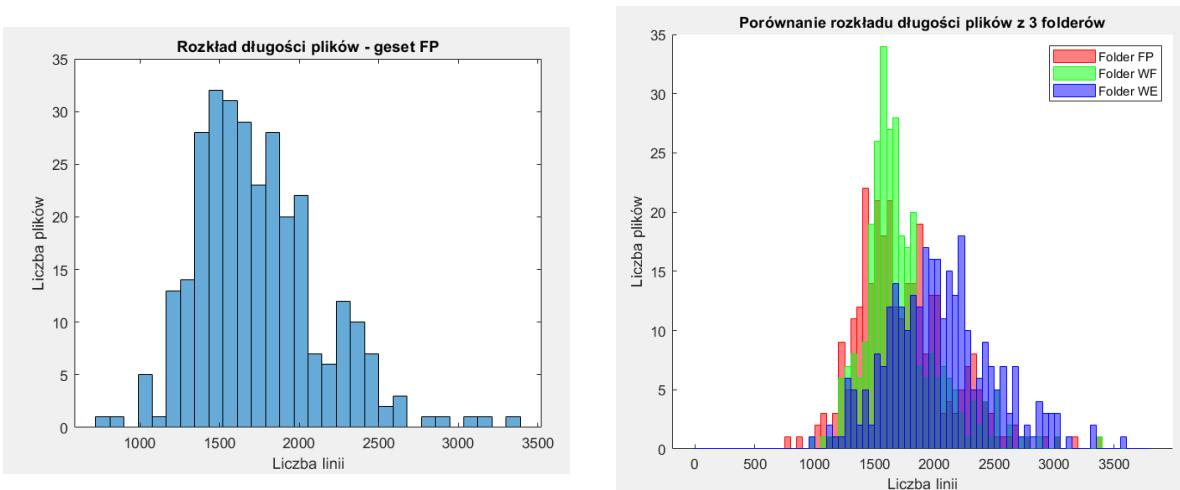


Rysunek 45: Charakterystyka częstotliwościowa – kanał 3

Wstępnie zakładano odfiltrowywanie filtrem typu Notch jedynie częstotliwości 50 Hz. Podczas analizy sygnałów pochodzących z rzeczywistych pomiarów zaobserwowano, że konieczne będzie odfiltrowanie również kolejnych harmonicznych, które stanowiły istotne zakłócenia sygnału. Dodano odpowiednie poprawki w przygotowanym na poprzednich etapach prac kodzie. W tym podrozdziale przedstawiono przykładowe wyniki pomiarów dla gestu WF (wrist flexion). Na rysunkach: 43, 44, 45 przedstawiono charakterystyki częstotliwościowe z podziałem na kanały pomiarów. Można na nich zaobserwować skalę zakłóceń, które wpływają na mierzony sygnał. Rysunek 42 pokazuje zmiany sygnału w czasie przed i po filtracji. Najbardziej istotna zmiana dotyczy usunięcia składowych stałych, której efektem jest przesunięcie sygnału i zmniejszenie amplitudy. Amplituda została podana w surowych wartościach przesyłanych z mikrokontrolera bez przeskalowania do fizycznej wielkości jaką jest napięcie. Ostatecznie zarejestrowano oraz poddano analizie trzy gesty, które stanowiły podstawę dalszych etapów opracowania danych.



Rysunek 46: Porównanie długości pomiarów dla dwóch różnych gestów



Rysunek 47: Histogram długości pomiarów gestu FP oraz zestawienie trzech histogramów w celu porównania

Została również przeprowadzona analiza statystyczna długości dostępnych plików, w których zapisywano pomiary. Na rysunkach 46 i 47 widoczne są histogramy, które ułatwiały proces podjęcia decyzji o ostatecznej długości analizowanego sygnału. Na jej podstawie podjęto decyzję o przyjęciu danych do wspólnej długości, tak aby dla każdego pomiaru uzyskać jednakową liczbę okien czasowych. Taki zabieg miał umożliwić późniejsze efektywne trenowanie sieci rozpoznajającej gesty.

Po przygotowaniu danych zostały one zapisane do pliku csv. Gotowy plik został następnie udostępniony członkowi zespołu odpowiedzialnemu za dalsze przetwarzanie i trenowanie sieci.

3.4 Integracja sieci neuronowej z rzeczywistymi danymi EMG

W celu adaptacji istniejącej architektury sieci neuronowej do przetwarzania rzeczywistych danych pochodzących z prostego układu EMG wyposażonego w 3 elektrody, konieczne było przeprowadzenie kilku modyfikacji na poziomie danych wejściowych oraz liczby klas wyjściowych:

- Liczbę kanałów wejściowych zredukowano z 32 do 3, co odpowiada liczbie fizycznych elektrod w badanym systemie.
- Liczbę okien czasowych analizowanych jednocześnie zwiększyliśmy z 5 do 10, z zachowaniem nakładania 50%, by uchwycić płynne przejścia między stanami mięśni.
- Liczbę gestów zredukowano z 17 do 3 (klasy: *wrist flexion, wrist extension, forearm pronation*), co lepiej odzwierciedla uproszczony scenariusz testowy systemu rzeczywistego.

3.4.1 Architektura sieci CNN + LSTM dla sygnałów EMG

Zastosowany model neuronowy bazuje na połączeniu konwolucyjnych warstw (CNN) oraz warstw rekurencyjnych typu LSTM (Long Short-Term Memory), co czyni go szczególnie skutecznym w analizie danych EMG, które mają zarówno cechy lokalne (w krótkim czasie) jak i zależności czasowe (w dłuższych sekwencjach).

Warstwy konwolucyjne (CNN)

Warstwy typu Conv1D służą do wyodrębniania lokalnych cech z każdego okna czasowego, takich jak zmiany amplitudy sygnału, impulsy aktywności czy lokalne wzorce mięśniowe. Zostały one opakowane w strukturę TimeDistributed, co umożliwia ich niezależne zastosowanie do każdego z 10 okien w sekwencji.

Dodatkowo zastosowano warstwy BatchNormalization (stabilizacja rozkładu aktywacji) oraz MaxPooling1D (redukcja wymiaru poprzez wybór maksimum lokalnego), co pozwala zwiększyć odporność modelu na szum oraz zmniejszyć liczbę parametrów w kolejnych warstwach.

Warstwy LSTM

Po ekstrakcji cech konwolucyjnych dane trafiają do dwóch kolejnych warstw LSTM, które uczą się relacji czasowych między kolejnymi oknami. Pierwsza warstwa LSTM zwraca sekwencję (czyli zachowuje informację czasową dla każdej ramki), druga natomiast aggreuguje ją do pojedynczego wektora reprezentującego całą sekwencję.

Dzięki mechanizmowi pamięci długoterminowej, LSTM potrafi skutecznie uchwycić kontekst sekwencyjny – np. jak zmienia się sygnał podczas wykonywania gestu od początku do końca – co czyni go idealnym komponentem dla analizy EMG.

Warstwy gęste i klasyfikator

Na zakończenie zastosowano dwie warstwy Dense (gęsto połączone), każda z BatchNorm i Dropout, co pozwala zwiększyć zdolność uogólniania i zapobiegać przeuczeniu. Warstwa końcowa to klasyczny Dense z aktywacją softmax, zwracająca prawdopodobieństwo przynależności próbki do jednej z trzech klas gestów.

3.4.2 Wyniki walidacji na rzeczywistych danych

W celu oceny zdolności generalizacyjnych modelu przeprowadzono walidację na niezależnym zbiorze danych pomiarowych, który nie był wykorzystywany podczas treningu. Zbiór ten zawierał 147 sekwencji (od 45 do 51 przypadających na każdy z trzech gestów). Model osiągnął wysoką dokładność klasyfikacji, wynoszącą około 94%.

Tabela 4: Metryki skuteczności modelu na danych testowych

Gest	Precision	Recall	F1-score	Support
Gest 1 – Wrist flexion	0.86	0.96	0.91	45
Gest 2 – Wrist extension	1.00	1.00	1.00	51
Gest 3 – Forearm pronation	0.96	0.86	0.91	51
Makrośrednia	0.94	0.94	0.94	147
Średnia ważona	0.94	0.94	0.94	147

Opis metryk:

- **Precision** — miara precyzji: jaki odsetek próbek oznaczonych jako dana klasa był faktycznie poprawny,
- **Recall** — czułość: jaki procent rzeczywistych przypadków danej klasy został poprawnie wykryty,
- **F1-score** — średnia harmoniczna precision i recall, będąca zbalansowaną oceną skuteczności,

- **Support** — liczba przypadków danej klasy w zbiorze testowym.

Wysokie wartości metryk w tabeli świadczą o bardzo dobrej skuteczności modelu, przy czym najbardziej jednoznaczną klasą okazał się *wrist extension*, który został sklasyfikowany bezbłędnie.

3.4.3 Analiza przestrzeni cech – t-SNE

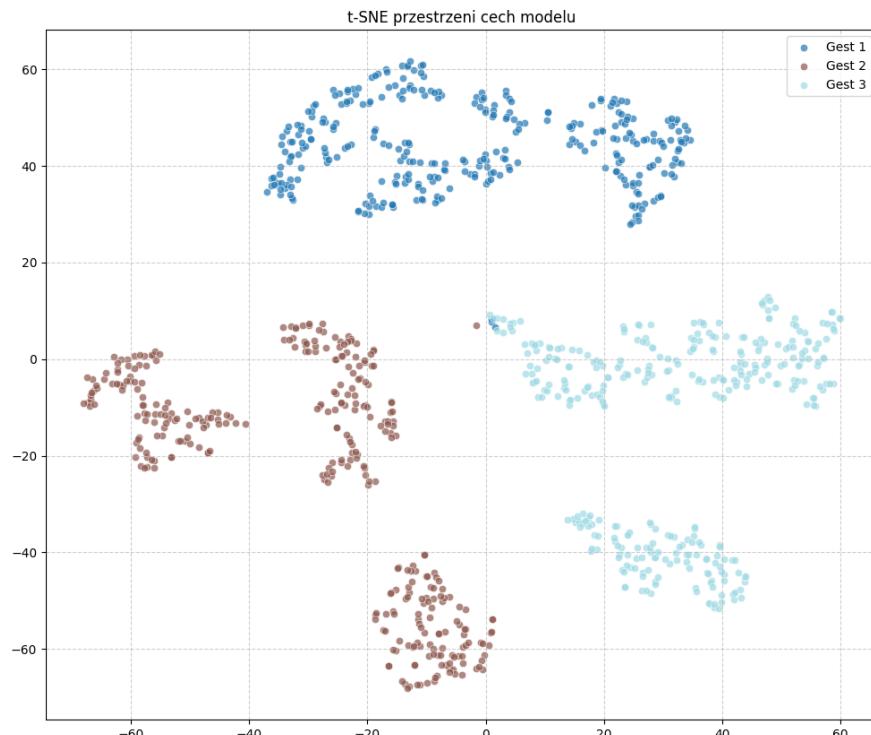
Aby lepiej zrozumieć, w jaki sposób sieć reprezentuje dane wejściowe w swojej warstwie ukrytej, przeprowadzono redukcję wymiarowości cech przy pomocy algorytmu **t-SNE** (t-distributed Stochastic Neighbor Embedding).

Algorytm ten działa w kilku etapach:

- Oblicza prawdopodobieństwa bliskości punktów w oryginalnej przestrzeni (tutaj: 64-wymiarowej reprezentacji sekwencji EMG po warstwie **Dense**).
- Przypisuje każdemu punktowi nowe współrzędne w przestrzeni 2D, tak aby podobieństwa między punktami (lokalne sąsiedztwa) były zachowane.
- Minimalizuje rozbieżność między rozkładem podobieństw w oryginalnej i zredukowanej przestrzeni poprzez dywergencję Kullbacka-Leiblera.

Wynikowy wykres t-SNE (Rysunek 48) pokazuje wyraźne klastry odpowiadające poszczególnym gestom, co potwierdza zdolność sieci do wyodrębnienia reprezentacji umożliwiającej poprawną klasyfikację. Zaobserwowano również pewne podgrupy wewnętrz klas gestów, co może wynikać z różnic między osobami wykonującymi pomiary lub specyficznych cech ich aktywacji mięśni.

Warto podkreślić, że t-SNE nie zachowuje metryki odległości globalnych, a jedynie lokalne struktury – dlatego oddalenie między grupami nie powinno być interpretowane ilościowo, a jakościowo.



Rysunek 48: Wizualizacja przestrzeni cech przy użyciu algorytmu t-SNE. Każdy punkt reprezentuje jedną sekwencję EMG, kolor odpowiada klasie gestu. Widoczne są odrębne klastry, co świadczy o wysokiej separowalności danych przez model.

3.4.4 Szczegóły struktury sieci

Tabela 5: Struktura warstw sieci neuronowej i liczba parametrów

Warstwa (typ)	Kształt wyjściowy	Liczba parametrów
InputLayer	(10, 25, 1)	0
TimeDistributed(Conv1D)	(10, 25, 32)	128
TimeDistributed(BatchNorm)	(10, 25, 32)	128
TimeDistributed(MaxPooling1D)	(10, 12, 32)	0
TimeDistributed(Conv1D)	(10, 12, 64)	6,208
TimeDistributed(BatchNorm)	(10, 12, 64)	256
TimeDistributed(MaxPooling1D)	(10, 6, 64)	0
TimeDistributed(Flatten)	(10, 384)	0
LSTM (1)	(10, 64)	114,944
LSTM (2)	(64)	33,024
Dense (64)	(64)	4,160
BatchNormalization	(64)	256
Dropout	(64)	0
Dense (32)	(32)	2,080
BatchNormalization	(32)	128
Dropout	(32)	0
Dense (3, softmax)	(3)	99
Suma parametrów:		161,411

Każda warstwa konwolucyjna zawiera jądra konwolucyjne uczące się filtrów, stąd widoczna liczba parametrów (np. 6,208 w drugiej konwolucji). Warstwy LSTM mają znaczną liczbę parametrów, ponieważ każda komórka LSTM zawiera bramki wejścia, zapominania, aktualizacji i wyjścia – każda z własnymi wagami i biasami. To właśnie te warstwy decydują o zdolności modelu do modelowania kontekstu czasowego.

Na rysunku 50 przedstawiono pełną architekturę zastosowanej sieci neuronowej CNN+LSTM. Graf ten uwzględnia wszystkie warstwy sieci, w tym ich nazwy, rozmiary tensorów oraz kolejność połączeń. Wizualizacja ta stanowi uzupełnienie tabeli 5 i ułatwia intuicyjne zrozumienie przepływu danych w modelu.

3.4.5 Struktura danych wejściowych – wyjaśnienie kształtu (10, 25, 1)

Choć dane pochodzą z trzech kanałów (elektrod), każda sekwencja analizowana przez sieć pochodzi z jednego kanału – a więc ma wymiar (10, 25), czyli 10 okien z 25 cechami. Następnie jest dodawany wymiar kanału głębokości (jeden kanał) – stąd ostateczny `input_shape = (10, 25, 1)`.

Alternatywą byłoby przetwarzanie sygnałów z wszystkich kanałów jednocześnie (np. (10, 25, 3)), jednak obecne podejście pozwala:

- uzyskać więcej przykładów treningowych (każda elektroda to osobny przykład),
- lepiej zróżnicować sygnały i analizować je niezależnie,
- uprościć strukturę modelu i skrócić czas treningu.

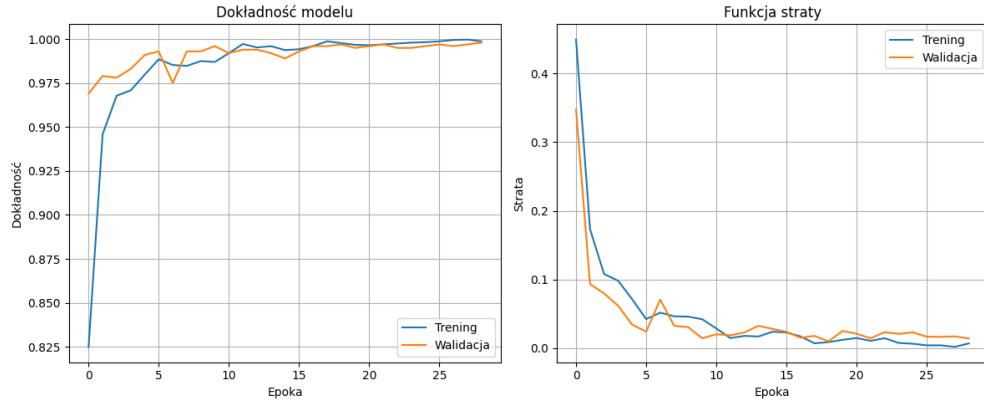
3.4.6 Optymalizacja i funkcja straty

Model został wytrenowany z wykorzystaniem optymalizatora **Adam**, który łączy zalety podejścia *Momentum* i *RMSProp*. Dzięki temu adaptacyjnie dostosowuje tempo uczenia do każdego parametru, co zwiększa stabilność i szybkość konwergencji, szczególnie w przypadku danych czasowych o dużej zmienności – takich jak sygnały EMG.

Funkcja straty zastosowana w modelu to **sparse categorical crossentropy**, czyli wariant entropii krzyżowej, w którym klasy przekazywane są jako liczby całkowite zamiast wektorów one-hot. Jest

to podejście bardziej efektywne obliczeniowo, zachowując przy tym pełną funkcjonalność klasycznej entropii krzyżowej.

3.4.7 Wykres treningu modelu

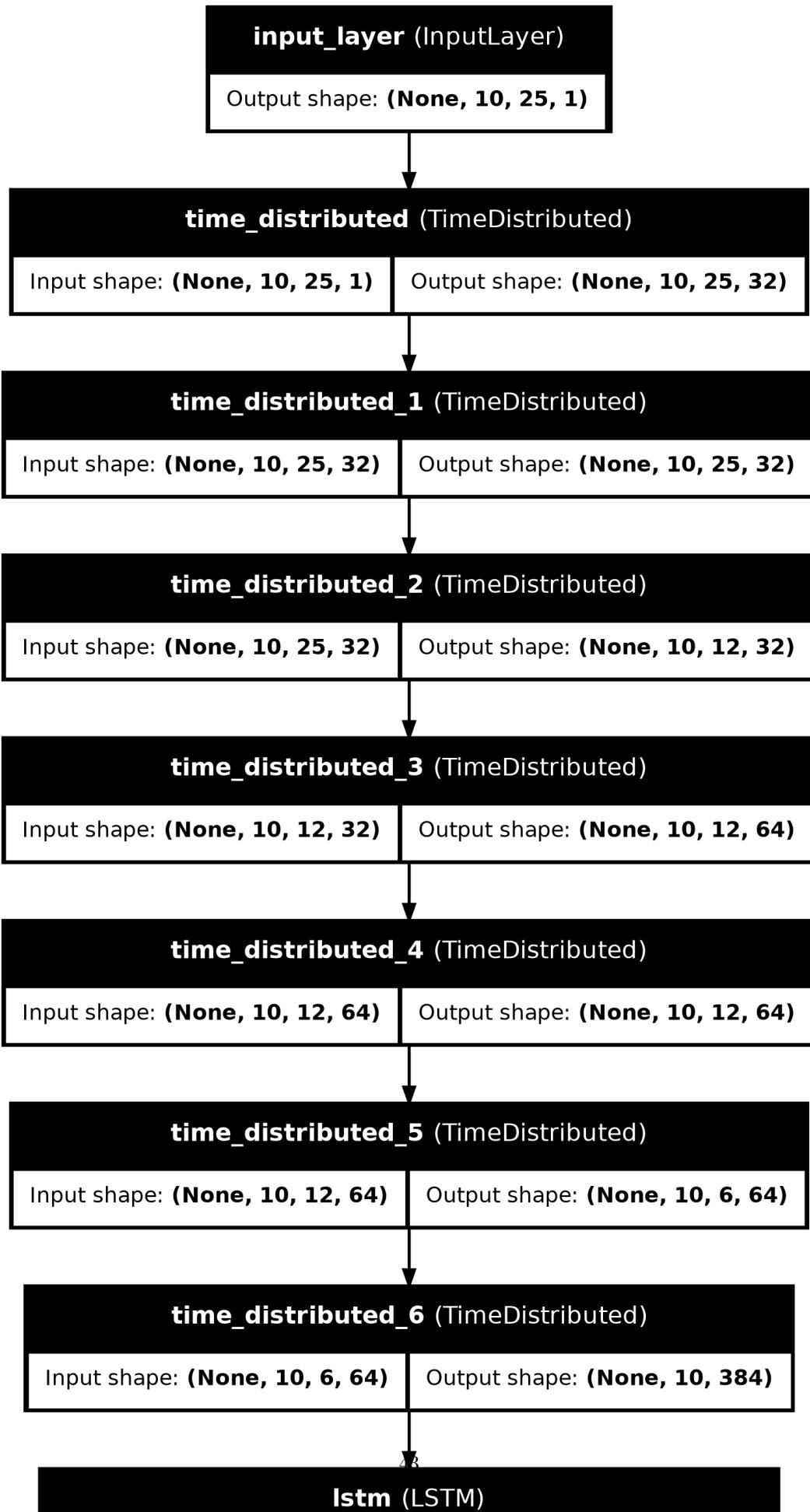


Rysunek 49: Przebieg treningu – dokładność i strata dla zbioru treningowego i walidacyjnego

Sieć zakończyła trening po 29 epokach dzięki zastosowaniu mechanizmu `early stopping`, który automatycznie przerwał uczenie po wykryciu braku poprawy dokładności na zbiorze walidacyjnym. Krzywe dokładności i straty wskazują na dobrą konwergencję i brak przeuczenia.

4 Podsumowanie projektu

Analiza i przetwarzanie sygnałów umożliwiły wyodrębnienie istotnych cech sygnału, które posłużyły w identyfikacji gestów. Niezwykle przydatna okazała się cyfrowa filtracja sygnału, która pozwoliła pozbyć się niechcianych zakłóceń na zadowalającym poziomie. Rozpoczęcie prac na próbnych danych zaczerpniętych z internetu pozwoliły na rozpoczęcie prac nad projektem od początku czasu przewidzianego na jego realizację. Prace nad rzeczywistymi pomiarami były stosunkowo krótkie, a do osiągnięcia lepszych efektów potrzeba więcej czasu na testy i weryfikację poprawności zastosowanych rozwiązań.



Literatura

- [1] Analiza falkowa. Politechnika Warszawska – e-sezam. <https://esezam.okno.edu.pl/mod/book/view.php?id=9&chapterid=78>.
- [2] Informacje na temat mięśni. <https://fizjoterapeuty.pl/uklad-miesniowy>.
- [3] SENIAM – Techniczne informacje na temat elektrod. <https://www.seniam.org/>.
- [4] M. Alattar, J. K. Aggarwal, E. Tunik. Gesture recognition and biometrics electromyogram (grab-my). PhysioNet. <https://physionet.org/content/grabmyo/1.1.0/>.
- [5] N. Arjunan, D. Kumar. Feature extraction and reduction of wavelet transform coefficients for emg pattern classification. ResearchGate. https://www.researchgate.net/publication/262011091_Feature_Extraction_and_Reduction_of_Wavelet_Transform_Coefficients_for_EMG_Pattern_Classification.
- [6] S. Faizan, M. Zahoor, T. Kanwal, A. A. Shah. Machine learning-based feature extraction and classification of emg signals for intuitive prosthetic control. *Applied Sciences*. <https://www.mdpi.com/2076-3417/14/13/5784>.
- [7] P. Wołczowski, P. Mróz, K. Pladzyk, A. Szymański. The system for emg and mmg signals recording for the bioprosthetic hand control. BazTech. https://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-27229fd5-6b50-43af-8f37-eb9e576533f6/c/Wolczowski__the_system_for_EMG.pdf.
- [8] W. J. Wołczowski A., Błędowski M. The system for emg and mmg signals recording for the bioprosthetic hand control. *Journal of Automation, Mobile Robotics & Intelligent Systems*, 11(3):22–29, 2017.