AUTOREFERAT ROZPRAWY DOKTORSKIEJ

Adaptacyjny system wymiany informacji w rozproszonych układach sterowania

mgr inż. Przemysław Strzelczyk

Promotor:
dr hab. inż. Krzysztof Tomczewski, prof. uczelni

Opole 2022

1. Wstęp

W niniejszej pracy przedstawiono nową koncepcję oprogramowania pośredniczącego przeznaczoną do rozproszonych i modułowych systemów sterowania maszyn i urządzeń robotycznych. Jednym z poważniejszych problemów podczas tworzenia współczesnego oprogramowania dedykowanego dla systemów sterowania jest brak jego generyczności oraz możliwości ponownego wykorzystania w momencie zmiany warstwy sprzętowej. Obecnie na rynku istnieje kilka rozwiązań oprogramowania pośredniczącego. Najbardziej znanymi są ROS, RT-Middleware oraz Player/Stage [34]. Oprogramowanie pośredniczące pełni ważną rolę w tworzeniu dużych instalacji przemysłowych oraz badawczych konstrukcji robotycznych. Przykładem może być system CLARAty, który znajduje się w portfolio centrum badawczo-rozwojowego Narodowej Agencji Aeronautyki i Przestrzeni Kosmicznej (NASA). NASA podkreśla, że w ramach swoich badań nad robotami większość oprogramowania jest tworzona od nowa dla każdej nowo projektowanej konstrukcji. Instytucja ta podkreśliła również fakt, że oprogramowanie związane z przemysłem robotycznym jest coraz bardziej skomplikowane, a proces przygotowawczy wymaga coraz większego wysiłku oraz dłuższego czasu. Podkreśla również potrzebę opracowania oprogramowania wspomagającego tworzenie rozwiązań robotycznych oraz koncepcji oprogramowania wielokrotnego użytku [87]. Niniejsza praca przedstawia autorską architekturę oprogramowania, posiadającą funkcjonalności najważniejsze z punktu widzenia tworzenia oprogramowania dedykowanego dla systemów sterowania, takie jak: komunikacja lokalna, komunikacja globalna, zautomatyzowany proces wykrywania nowych węzłów oraz system dystrybucji i funkcjonalność reagowania na zdarzenia zewnętrzne. Zaprezentowany system pomimo rozbudowanej gamy funkcjonalności, posiada krótką ścieżkę integracji z nowym lub istniejącym już oprogramowaniem. Wykorzystanie opracowanego systemu pozwala na opracowanie środowiska komunikacyjnego w obrębie kilku aplikacji sterujących znajdujących się w jednym module sprzętowym oraz pomiędzy kilkoma współpracującymi ze sobą modułami. Istnieje również możliwość opracowania i zaimplementowania schematu reakcji oprogramowania na zdarzenia zewnętrzne. System umożliwia przekazywanie informacji o tym zdarzeniu do wszystkich modułów, np. w momencie awarii jednego z podzespołów. W pracy zaprezentowano podsystem posiadający własny protokół służący do automatycznego wykrywania węzłów systemu rozproszonego. Podsystem ten jest kluczowym elementem systemów rozproszonych o dynamicznej strukturze. Może być również wykorzystany do automatycznej konfiguracji w urządzeniach o konstrukcji modułowej. Dzięki jego wykorzystaniu można w prosty sposób utworzyć sieć współpracujących ze sobą napędów lub robotów. Podsystem ten pozwala również na bieżące monitorowanie ilości przyłączanych lub odłączanych węzłów sieci. W ramach niniejszej pracy system ten został zaimplementowany w układzie sterowania napędami górnej części aktywnego egzoszkieletu ręki. W ramach badań wykonano również stanowisko pomiarowe i uproszczony prototyp egzoszkieletu lewej ręki.

W autoreferacie zachowano numeracje bibliografii, rozdziałów, podrozdziałów, rysunków oraz tabel zgodne z rozprawą doktorską.

1.1.Wprowadzenie

Oprogramowanie pośredniczące jest rozwiązaniem oddzielającym warstwę systemu operacyjnego oraz warstwę sprzętową od oprogramowania bazującego na nich. [67, 57] Pełni ono rolę pośrednika w transakcjach wymiany informacji ze sprzętem oraz wprowadza dodatkową abstrakcję, umożliwiającą odwoływanie się do wewnętrznych peryferii systemu. Warstwa pośrednicząca ukrywa przed twórcą aplikacji wyższego rzędu rozwiązania szczegółowe niskiego poziomu. D. Bakken w roku 2001 przedstawił swoją definicję oprogramowania pośredniego. Brzmi ona następująco:
„Oprogramowanie pośredniczące, to klasa oprogramowania zaprojektowana, aby pomóc w zarządzaniu złożonością oraz niejednorodnością w rozproszonych systemach. Zostało ono zdefiniowane jako warstwa oprogramowania powyżej systemu operacyjnego, lecz poniżej warstwy aplikacji, która udostępnia wspólną abstrakcję programową w rozproszonym systemie". [10] Na rys. 1.1 przedstawiono umiejscowienie oprogramowania pośredniczącego w architekturze oprogramowania sterującego z uwzględnieniem warstwy sprzętowej.

Rys. 1.1. Umiejscowienie oprogramowania pośredniczącego w architekturze oprogramowania sterującego.

Głównym zadaniem tego typu oprogramowania w systemach sterowania jest dostarczenie mechanizmów niezbędnych do budowy dowolnego typu aplikacji sterujących: począwszy od prostych implementacji, skupiających się wokół jednego modułu sprzętowego, do konstrukcji wykorzystujących architekturę rozproszoną. Włączenie w proces tworzenia programu sterującego oprogramowania pośredniczącego pozwala na przyspieszenie procesu jego rozwoju. [63, 81] Tworzący oprogramowanie sterujące musi skupić się jedynie na zaprojektowaniu części logicznej swojej aplikacji wraz z niezbędnymi algorytmami sterującymi. Dzięki oprogramowaniu pośredniczącemu nie musi skupiać uwagi na trudnych kwestiach informatycznych, lecz wyłącznie na rozwiązaniu zdefiniowanego przez siebie problemu robotycznego. [34, 84]

2. Teza, cel i zakres pracy

Przedstawione w niniejszej pracy rozwiązanie opiera się na koncepcji oprogramowania pośredniczącego, umożliwiającego utworzenie izolacji oprogramowania wyższego rzędu od dostępnego systemu operacyjnego oraz peryferii sprzętowych. Izolacja powinna odbywać się za pomocą oprogramowania składającego się z warstw funkcjonalnych, umożliwiających wspomaganie projektowania systemów sterowania w napędach elektrycznych i robotyce. Zaproponowana w pracy warstwa pośrednicząca wspomaga szybkie tworzenie oprogramowania sterującego oraz umożliwia w odróżnieniu od innych systemów dostępnych na rynku krótką ścieżkę integracji już istniejących aplikacji sterujących z niniejszym systemem. W niniejszej pracy przedstawiono nowe, w pełni działające rozwiązanie warstwy pośredniczącej wraz z implementacją bazującego na niej systemu rozproszonego sterowania górnej części egzoszkieletu wspomaganego. W ramach pracy opracowano i wykonano również stanowisko pomiarowe wraz z modelem fizycznym aktywnego egzoszkieletu kończyny górnej oraz przeprowadzono badania eksperymentalne wydajności systemu wymiany informacji i działania egzoszkieletu.

Określenie odpowiednich celów oraz tezy pracy wymagało wykonania wstępnych czynności przygotowawczych:

1. analizy dostępnych na rynku podobnych rozwiązań, oferujących funkcjonalność oprogramowania pośredniczącego,
2. opracowania koncepcji oraz określenia podstawowych funkcjonalności oprogramowania pośredniczącego,
3. określenia obszarów potencjalnego zastosowania.

Przeprowadzone czynności przygotowawcze umożliwiły sformułowanie następującej tezy:

Możliwe jest opracowanie programowej warstwy komunikacyjnej, realizującej zadania wymiany informacji w rozproszonych systemach sterowania i urządzeniach o budowie modułowej, pełniącej funkcje kontrolne w zakresie konfiguracji i podstawowych zasad bezpieczeństwa systemu sterowania, będącej warstwą pośrednią pomiędzy aplikacjami sterującymi a warstwą sprzętową i systemem operacyjnym.

Do udowodnienia powyższej tezy niezbędne było przeprowadzenie poniżej wymienionych czynności badawczych.

1. Opracowanie koncepcji oraz implementacja podsystemu lokalnej wymiany informacji.
2. Opracowanie koncepcji oraz implementacja podsystemu globalnej wymiany informacji.
3. Opracowanie koncepcji oraz implementacja abstrakcyjnej warstwy sprzętowej.
4. Opracowanie koncepcji oraz implementacja podsystemu dystrybucji zdarzeń.
5. Opracowanie koncepcji oraz implementacja podsystemu detekcji elementów sieci.

6. Wykonanie badań wydajnościowych mechanizmów wymiany danych adaptacyjnego systemu wymiany informacji.
7. Opracowanie modelu kinematycznego oraz modelu symulacyjnego działania układu sterowania egzoszkieletu kończyny górnej.
8. Opracowanie metody i wykonanie optymalizacji nastaw regulatorów będących częścią układu sterowania egzoszkieletu kończyny górnej.
9. Opracowanie i wykonanie prototypu egzoszkieletu kończyny górnej o uproszczonej budowie, wraz z jednostkami napędowymi.
10. Opracowanie koncepcji oraz implementacja aplikacji sterujących górną częścią egzoszkieletu w oparciu o adaptacyjny system wymiany informacji.
11. Wykonanie pomiarów na stanowisku badawczym wyposażonym w prototyp egzoszkieletu kończyny górnej.

Po wstępnej analizie możliwości realizacji zostały wyznaczone cele szczegółowe pracy:

1. Opracowanie koncepcji działania adaptacyjnego systemu wymiany informacji, posiadającego cechy i funkcjonalność oprogramowania pośredniczącego wraz z warstwami abstrakcji sprzętowej, dystrybucji zdarzeń, detekcji elementów sieci i jego implementacja programowa.
2. Budowa rozproszonego systemu wymiany informacji.
3. Badanie wydajności warstw komunikacyjnych adaptacyjnego systemu wymiany informacji.
4. Opracowanie i budowa prototypu egzoszkieletu kończyny górnej.
5. Opracowanie modułowego systemu sterowania prototypu egzoszkieletu, bazującego na wykonanym adaptacyjnym systemie wymiany informacji, przeznaczonego do zastosowań rehabilitacyjnych umożliwiającego:
    - wspomaganie wykonywania ruchów,
    - wykonywanie ruchów zaprogramowanych,
    - zapewnienie podstawowych zasad bezpieczeństwa.

2.1. Przyjęta koncepcja oprogramowania pośredniczącego

Przegląd istniejących rozwiązań oprogramowania pośredniczącego pozwolił na określenie braków istniejących w tym zakresie. Jednym z głównych problemów, na który powołują się autorzy publikacji związani ze środowiskiem wytwarzania pośredniczącego oprogramowania dla rozwiązań robotycznych jest złożoność tych systemów. Dostępne środowiska cechują się dużym stopniem skomplikowania oraz znaczną objętością kodu. [15] Z tego powodu podstawowym problemem jest rozpoczęcie projektowania oprogramowania sterującego wspomaganego przez zastosowanie warstwy pośredniczącej. Kolejnym, często spotykanym problemem, jest integracja już istniejącego oprogramowania z oprogramowaniem pośredniczącym. Rozpoczęcie pracy z oprogramowaniem pośredniczącym jest zazwyczaj trudne z powodu skomplikowanego procesu integracyjnego lub szczególnych wymogów w zakresie architektury oprogramowania. Naukowcy zajmujący się zagadnieniami związanymi z oprogramowaniem pośredniczącym podkreślają, że obecnie dostępne mechanizmy nie są ukierunkowane na uproszczanie procesu ponownego wykorzystania kodu. Systemy stają się bardzo skomplikowane oraz rozszerzają gamę dostępnych narzędzi, budując przy tym siatkę ukrytych zależności nieznanych dla osób z poza danego projektu. [15]
Na tej podstawie w niniejszej pracy zaproponowano rozwiązanie posiadające krótką ścieżkę integracyjną z już istniejącymi systemami sterowania oraz nowymi projektami. Oprogramowanie pośredniczące będzie dostarczane i integrowane z zewnętrzną aplikacją za pomocą biblioteki. W celu wykorzystania funkcjonalności oferowanych przez bibliotekę aplikacja musi wykorzystać zestaw funkcji API, których deklaracje są dostarczane w plikach nagłówkowych wraz z biblioteką. Dodatkowo, w celu wykorzystania funkcjonalności platformy, do jej uruchomienia dostarczony zostanie gotowy skrypt. Integracja oprogramowania z platformą realizowana będzie w dwóch krokach: linkowanie biblioteki wraz z załączeniem interfejsu oraz uruchomienie oprogramowania pośredniczącego. Proponowane rozwiązanie oprócz krótkiej ścieżki integracji udostępnia szereg istotnych funkcjonalności, bez konieczności dołączania zewnętrznych mechanizmów. Podstawową funkcjonalnością niniejszego rozwiązania jest system komunikacyjny. System ten pozwala na szybką i efektywną wymianę informacji pomiędzy aplikacjami sterującymi. Dwie aplikacje sterujące mogą wymieniać między sobą dowolne informacje, bez ograniczeń ich formatu. Mechanizm komunikacyjny przyjęto taki, aby umożliwiał efektywną wymianę informacji zarówno w obrębie jednego modułu sprzętowego jak i całej sieci rozproszonej. Warstwa wymiany informacji została oparta o koncepcję wymiany wiadomości. W oparciu o niniejsze rozwiązanie komunikacyjne użytkownik może budować zarówno mechanizmy synchroniczne jak i asynchroniczne. Warstwa ta musi zapewniać wymianę informacji w czasie umożliwiającym realizację procesu sterowania. Kolejnym kluczowym elementem systemu jest warstwa detekcji aktualnej konfiguracji sieci węzłów w systemie. Warstwa odpowiedzialna jest za monitorowanie stanu i liczby węzłów należących do danego systemu rozproszonego. Odpowiedzialna jest ona za udostępnianie informacji na temat aktualnie dołączonych węzłów oraz dostarczanie informacji w momencie zmiany statusu węzła (z dostępnego na niedostępny i odwrotnie). Proponowane rozwiązanie posiada możliwość dystrybuowania informacji o zaistniałych zdarzeniach za pomocą mechanizmu publikacji-subskrypcji. Zdarzenia są

definiowane bezpośrednio przez użytkownika oprogramowania za pomocą interfejsu warstwy oferującej publikację zdarzeń. Dzięki tej warstwie użytkownik może poinformować subskrybentów o nagłej awarii aktualnie wykorzystywanego podzespołu lub niespodziewanej zmianie konfiguracji systemu. System nie wymaga również konfigurowania, ponieważ proces ten realizowany jest automatycznie. Oprogramowanie pośrednie w swoich ramach posiada również rozwiązanie abstrakcyjnej warstwy sprzętowej. Dzięki temu rozwiązaniu możliwa jest kontrola podzespołów wraz z utrzymaniem odpowiedniej generyczności warstwy logicznej aplikacji sterującej. Warstwa abstrakcji sprzętowej została wyposażona w mechanizm synchronizacji odwołań do sprzętu. Synchronizacja obejmuje zakres działania systemu operacyjnego. Oprogramowanie pośredniczące wyposażone zostało również w warstwę testową, pozwalającą na weryfikację poprawności jego własnych mechanizmów, poprzez wykonanie zdefiniowanych przypadków testowych. Warstwa posiada również szereg funkcji pozwalających na uruchomienie gotowej aplikacji testującej oraz wykonanie na niej określonego fragmentu kodu w celu weryfikacji poprawności działania danych funkcjonalności lub przeprowadzenia testów obciążeniowych. W ramach niniejszej pracy zostały przeprowadzone badania wydajnościowe warstw komunikacyjnych oraz przeprowadzono testy działania poszczególnych funkcjonalności na stanowisku badawczym poprzez wykorzystanie ich w implementacji systemu sterowania egzoszkieletu wspomaganego.

2.2.Potencjalny obszar zastosowania

Oprogramowanie pośredniczące o przyjętej architekturze może zostać zastosowane wszędzie tam, gdzie istnieje potrzeba zoptymalizowania oraz ujednolicenia elementów oprogramowania sterującego, np. w napędach złożonych maszyn i w robotyce. Z powodzeniem może zostać zastosowane w:

- robotach (mobilnych [85], humanoidalnych [105, 106], przemysłowych, medycznych [98]),
- systemach sterowania złożonych układów napędowych maszyn,
- rozwiązaniach wymagających wymiany informacji (lokalnej i w systemie rozproszonym),
- rozwiązaniach wykorzystujących mechanizm dynamicznej konfiguracji i rekonfiguracji (automatyczne rozpoznawanie konfiguracji układów modułowych),
- rozwiązaniach adaptujących się do czynników zewnętrznych oraz wewnętrznych.

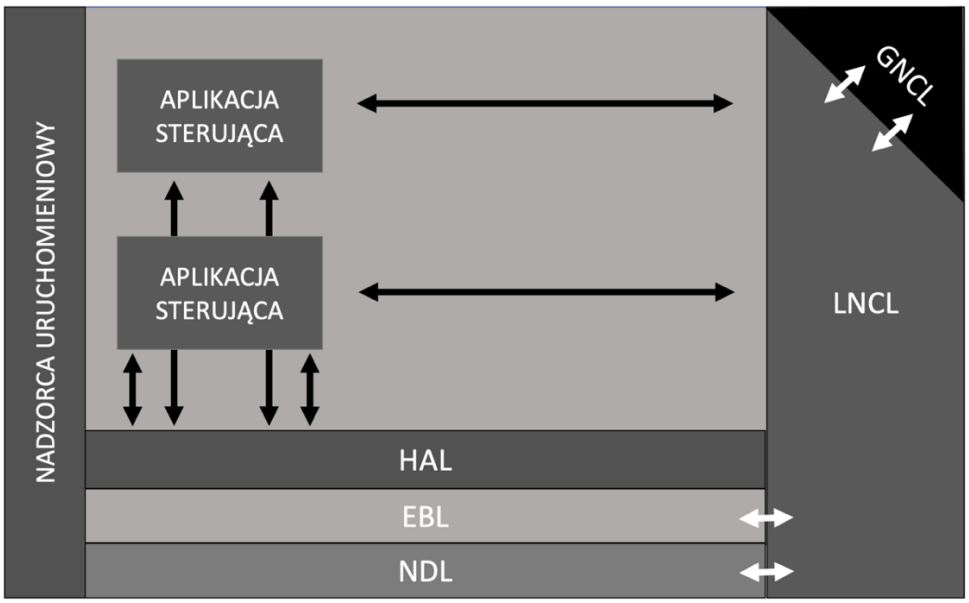
Obszar potencjalnego zastosowania opracowanego oprogramowania pośredniczącego jest bardzo szeroki. Przyjęte rozwiązanie może zostać wykorzystane podczas budowy prawie dowolnego urządzenia, które posiada system operacyjny dostosowany do jego obsługi. Platforma zapewnienia możliwość wprowadzania zmian sprzętowych w trakcie realizacji projektów i tworzenia kolejnych wersji urządzeń. Możliwa jest również komunikacja w zakresie systemu rozproszonego. [85] W zależności od potrzeb danego projektu umożliwia ono adaptację do zdarzeń zewnętrznych, np. w przypadku roju robotów mobilnych w sytuacji utraty pojedynczej jednostki mogą zostać wykorzystane mechanizmy automatycznej rekonfiguracji systemu. Mechanizmy pozwalające na dynamiczną rekonfigurację mogą również znaleźć

zastosowanie w rozwiązaniach robotyki oraz automatyki przemysłowej. Dzięki zapewnieniu możliwości rekonfiguracji, system może dokonać automatycznej reorganizacji w przypadku utraty lub dołączenia nowego elementu. Rekonfiguracja to jedna z kluczowych funkcjonalności opracowanego oprogramowania pośredniczącego. [30] Sposób reorganizacji systemu będzie uzależniony jedynie od wcześniej określonych reguł zaimplementowanych w programie sterującym. W przypadku urządzeń o budowie modułowej automatyczne rozpoznawanie struktury urządzenia umożliwia automatyczne dostosowanie układu i algorytmu sterowania do danej konfiguracji sprzętowej. Zastosowanie oprogramowania pośredniczącego w urządzeniach medycznych pozwala na budowę bardziej skomplikowanych i rozwojowych systemów. [131]

3. Przyjęta struktura adaptacyjnego systemu wymiany informacji

Przyjęty do realizacji system wymiany informacji pełni rolę oprogramowania pośredniczącego. Uproszczona struktura systemu została przedstawiona na rys. 3.1. System składa się z następujących, powiązanych ze sobą elementów:

- warstwy lokalnej wymiany informacji LNCL (*ang. Local Node Communication Layer*),
- warstwy globalnej wymiany informacji GNCL (*ang. Global Node Communication Layer*),
- warstwy wykrywania obecności węzłów NDL (*ang. Node Detection Layer*),
- warstwy umożliwiającej adaptację i rozgłoszenie zdarzeń EBL (*ang. Event Broadcast Layer*),
- abstrakcyjnej warstwy sprzętowej HAL (*ang. Hardware Abstraction Layer*),
- mechanizmu nadzorcy uruchomieniowego.



Rys. 3.1. Uproszczony schemat przedstawiający strukturę oprogramowania i interakcje pomiędzy poszczególnymi warstwami systemu.

3.2.Warstwa lokalnej wymiany informacji LNCL

Mechanizm LNCL odpowiada za komunikację odbywającą się w obrębie jednego węzła sieci rozproszonej. Warstwa została zaprojektowana specjalnie na potrzeby aplikacji sterujących,

wymagających przesyłania lokalnie informacji o dowolnym rozmiarze w szybki oraz efektywny sposób.

3.2.1. Architektura LNCL

Warstwa lokalnej komunikacji jest podstawową warstwą opracowanego oprogramowania pośredniczącego. Podczas wykonywania sekwencji uruchomieniowej przez nadzorcę, mechanizm startowy warstwy zostaje wykonany jako pierwszy spośród wszystkich dostępnych warstw. Umieszczenie warstwy jako pierwszej w kolejce jest ściśle związane z zależnościami pomiędzy warstwami systemu. Warstwa LNCL jest wykorzystywana w innych warstwach jako podstawowy mechanizm wewnątrz-modułowy, służący do wymiany informacji. [117, 118]

W związku z potrzebą udostępnienia niezawodnej oraz szybkiej metody komunikacji warstwa ta została wyposażona w mechanizm komunikacji bazujący na lokalnych domenowych gniazdach uniksowych, przeznaczonych do komunikacji międzyprocesowej (*ang. unix domain sockets*). Mechanizm bazujący na gniazdach domeny uniksowej wykazuje lepszą efektywność przesyłu danych (w zależności od sposobu implementacji gniazd), niż wykorzystanie gniazd sieciowych TCP opartych o komunikację w obrębie pętli lokalnej. [56] Dodatkowym atutem takiego rozwiązania jest udostępniony, podobny zestaw funkcji API, znany z gniazd sieciowych oferowanych przez standard POSIX. [100] Procedura uruchomieniowa warstwy LNCL rozpoczyna się od uruchomienia wątku *communicationServerThread*. Wątek ten odpowiedzialny jest za zainicjalizowanie serwera bazującego na uniksowych gniazdach domenowych. Adres nasłuchiwania ustalono na */tmp/LNCL_SERVICE*. Po poprawnej inicjalizacji serwer nasłuchuje klientów chcących ustalić połączenie. Po wykryciu próby podłączenia, serwer uruchamia nowy wątek dedykowany do obsługi zdarzeń pochodzących z nowego połączenia. Nowy wątek zostaje utworzony każdorazowo dla nowo dołączanego klienta LNCL. Do nowo utworzonego elementu zostają przekazane dane niezbędne do obsługi sesji klienckiej za pomocą obiektu klasy *Point*. Klasa *Point* udostępnia mechanizmy pozwalające na przechowywanie informacji o połączeniu. Jej głównym zadaniem jest umożliwienie przechowywania danych dotyczących obecnego stanu gniazda sieciowego oraz adresu transportowego LNCL.

3.2.12. Przykładowe zastosowanie warstwy LNCL

- Tworzenie, usuwanie adresów transportowych.
- Tworzenie, przesyłanie oraz odbieranie w obszarze lokalnym wiadomości o niezdefiniowanym rozmiarze oraz strukturze przesyłanej informacji.

3.3. Warstwa globalnej wymiany informacji GNCL

Mechanizm GNCL odpowiada za komunikację związaną z wymianą informacji pomiędzy odbiorcami znajdującymi się poza przestrzenią węzła lokalnego. [118] Architektura warstwy została zaplanowana w sposób, który pozwala na szybką wymianę informacji oraz zapewnienie niezawodności dostarczania wiadomości.

3.3.1. Architektura GNCL

Koncepcja globalnej warstwy komunikacyjnej bazuje na połączeniach opartych o protokół TCP (*ang. Transmission Control Protocol*). W związku ze specyfiką protokołu dostarcza on możliwość tworzenia kanałów komunikacyjnych charakteryzujących się niezawodnością dostarczania pakietów. [3, 134] Protokół ten został zastosowany również z powodu wbudowanego mechanizmu potwierdzeń oraz zachowania kolejności dostarczania informacji. [3, 134]. Architektura tej warstwy pozwala na bezinwazyjną zmianę wykorzystywanego protokołu na UDP. Zmiana wiąże się jednak z brakiem gwarancji dostarczenia informacji oraz zachowania kolejności pakietów, wynikającej z charakterystyki protokołu UDP. [3, 134] Warstwa GNCL podczas procedury startowej uruchamia dwa wątki niezbędne do jej poprawnego działania. Pierwszy z nich, obsługiwany za pomocą metody *connectionHandlerServerThread*, odpowiedzialny jest za zainicjalizowanie serwera opartego o protokół TCP/IP. Po udanej próbie inicjalizacji serwer w sposób ciągły rozpoczyna akceptacje nowych połączeń. Połączenia przyjmowane są z wykorzystaniem portu 9004. Mechanizm GNCL został skonfigurowany wstępnie w sposób, który pozwala na obsłużenie 100 połączeń zewnętrznych w tym samym czasie. Określono również rozmiar graniczny pojedynczego pakietu na 256 kB. Rozmiar graniczny określa wstępny rozmiar pamięci, do której zapisywane są dane odebrane przez warstwę GNCL z węzłów zewnętrznych. Rozmiar może być konfigurowany przez użytkownika. Po zaakceptowaniu połączenia, mechanizm tworzy dedykowany wątek służący do obsługi nowego klienta, który jest obsługiwany przez metodę *connectionHandlerThread*. Trzeci uruchamiany wątek, obsługiwany za pomocą metody *routingOUTThread*, jest odpowiedzialny za przekazywanie informacji z wewnątrz węzła do adresatów znajdujących się poza węzłem. W sekwencji uruchomieniowej oprogramowania pośredniczącego warstwa globalnej wymiany informacji zostaje uruchomiona jako trzecia w kolejności, zaraz po warstwie LNCL oraz NDL. Przedstawiona kolejność uruchomieniowa jest narzucona z powodu zależności warstwy GNCL od mechanizmów udostępnianych przez warstwy LNCL oraz NDL.

3.3.4. Przykładowe zastosowanie warstwy GNCL

- Przekazywanie wiadomości LNCL do adresatów znajdujących się w obszarze węzłów zewnętrznych.
- Przekazywanie wiadomości otrzymanych od nadawców znajdujących się poza lokalną instancją oprogramowania pośredniego.

3.4.Warstwa automatycznej detekcji węzłów systemu NDL

Oprogramowanie pośredniczące oraz zaimplementowana w jego ramach funkcjonalność automatycznej detekcji węzłów jest szczególnie ukierunkowana na zastosowanie w systemach rozproszonych i modułowych do automatycznego rozpoznawania konfiguracji układu. Warstwa NDL odpowiada za detekcję aktualnego stanu węzłów systemu. Funkcjonalność warstwy pozwala na pobranie informacji o aktualnie współpracujących węzłach systemu, korzystających z niniejszego oprogramowania pośredniego. Informacje przekazywane przez warstwę pozwalają również na uzyskanie informacji na temat utraconych węzłów.

3.4.1. Architektura NDL

Głównym zadaniem postawionym przed mechanizmami oraz architekturą warstwy NDL jest efektywność jej pracy. W związku z postawionymi wymaganiami, do realizacji mechanizmu wykrywania obecności węzłów systemu rozproszonego wybrano protokół UDP. Protokół dostarcza możliwość bezpołączeniowej wymiany informacji oraz nie posiada mechanizmów utrzymujących poprawną kolejność wysłanych datagramów oraz nie gwarantuje pełnego dostarczenia przesłanej informacji. [3] W przypadku niektórych zastosowań cechuje go jednak wyższa wydajność w porównaniu do protokołu TCP. [79] Wybór protokołu UDP pozwolił na osiągnięcie kompromisu pomiędzy wydajnością a funkcjonalnością. Warstwa wykrywania węzłów NDL nie jest wrażliwa na ewentualną utratę informacji spowodowaną poprzez niedostarczenie informacji. Procedura startowa warstwy NDL rozpoczyna swoje działanie poprzez uruchomienie wątku *discoverSendThread* odpowiedzialnego za przesyłanie zapytań o obecność węzłów. Kolejnym uruchamianym elementem jest wątek *discoverRecvRequestThread* dedykowany do odbioru zapytań przychodzących, związanych z mechanizmem detekcji. Trzecim, niezbędnym elementem w procedurze startowej, jest wątek *discoverRecvResponseThread*, posiadający mechanizm potwierdzający otrzymane zapytania pochodzące z węzłów zewnętrznych. Ostatnim elementem wymaganym do poprawnego funkcjonowania warstwy jest mechanizm odbioru zapytań pochodzących z wnętrza węzła lokalnego. Obsługa lokalnych zapytań realizowana jest za pomocą funkcjonalności udostępnianej przez wątek *handleMessageThread*. Żadna wymiana informacji pomiędzy instancjami warstw NDL, znajdującymi się w różnych modułach w systemie rozproszonym, nie wykorzystuje struktury wiadomości LNCL. Wiadomości NDL nie posiadają nagłówka określającego podstawowe informacje na temat odbiorcy oraz nadawcy, czy też rozmiaru wiadomości, jak w przypadku wiadomości LNCL. Informacje przekazywane przez warstwę zawierają jedynie informacje niezbędne do zidentyfikowania obecności węzła w systemie.

3.4.9. Przykładowe zastosowanie warstwy NDL

- Pobranie informacji na temat aktualnej topologii systemu rozproszonego.
- Pobranie informacji na temat wybranego węzła lub wszystkich węzłów systemu rozproszonego.
- Uzyskanie adresów węzłów zewnętrznych w systemie rozproszonym.

- Uzyskanie informacji na temat wersji oprogramowania w węzłach zewnętrznych.

3.5. Warstwa umożliwiająca adaptację i rozgłoszenie zdarzeń EBL

Jednym z głównych atutów prezentowanego adaptacyjnego systemu pośredniczącego jest możliwość adaptacji do zdarzeń zewnętrznych oraz wewnętrznych. Zdarzenie rozumiane jest jako sytuacja, po wystąpieniu której system powinien zareagować w zdefiniowany sposób. Zdarzenia dzielą się na zewnętrzne oraz wewnętrzne. Pierwszy przypadek obejmuje wszelkie zdarzenia pochodzące z modułów znajdujących się poza kontrolą lokalnej instancji systemu pośredniczącego. Do tej grupy zaliczane są wszystkie nielokalne instancje warstwy pośredniczącej. Drugi przypadek obejmuje zakres lokalnej warstwy pośredniczącej. Omawiany system, zajmujący się zarządzaniem mechanizmami rozgłoszeniowymi, udostępnia mechanizmy umożliwiające zdefiniowanie instrukcji wykonywanych w przypadku wystąpienia konkretnego czynnika w systemie. Mechanizm ten jest jednym z kluczowych elementów funkcjonalności automatycznej rekonfiguracji systemu. Automatyczna rekonfiguracja systemu bazuje na informacji pobranej z warstwy NDL i dostarczonej dzięki warstwie EBL, na temat zmiany struktury systemu. Dzięki udostępnieniu funkcjonalności wspierających adaptację systemu do warunków zewnętrznych, wszelkie rozwiązania oparte o opracowany system, posiadają możliwość autonomicznej zmiany sposobu działania. Aplikacje sterujące uzyskują mechanizm, pozwalający na implementację dodatkowych scenariuszy, umożliwiających wdrożenie akcji naprawczych w przypadku niespodziewanej awarii systemu, a w połączeniu z funkcjonalnością oferowaną przez pozostałe warstwy (NDL, GNCL, LNCL) mogą przekazywać lub delegować zadania w zależności od aktualnej konfiguracji systemu rozproszonego. Działania takie nie miałyby miejsca w przypadku braku mechanizmów adaptacyjnych oraz rozgłoszeniowych. Warstwa EBL udostępnia funkcjonalność pozwalającą na łatwą integrację z istniejącymi elementami oprogramowania poprzez odwołania do funkcji API.

3.5.1. Architektura EBL

Architektura warstwy EBL została podzielona na mechanizmy znajdujące się w przestrzeni tej warstwy oraz mechanizmy wymagające jawnego wywołania po stronie elementu wykorzystującego jej funkcjonalność. Założeniem warstwy jest udostępnienie funkcjonalności pozwalającej na transport określonej informacji do wszystkich obserwatorów. Transport odbywa się z określeniem z góry zdefiniowanego obszaru propagacji nadanej wiadomości. Mechanizmy transportowe bazują na wykorzystaniu funkcjonalności LNCL, jako głównego mechanizmu transportowego. Jeśli zakres propagacji wiadomości wykracza poza lokalną instancję oprogramowania pośredniego udział w transporcie bierze również warstwa GNCL.

Procedura uruchomieniowa warstwy EBL ogranicza się do uruchomienia jednego wątku *EBLServerHandlerThread*. Wątek ten odpowiedzialny jest za rejestrację warstwy w systemie komunikacyjnym LNCL, wykorzystując statyczny adres – 0x9999. Adres ten przypisany jest na stałe do warstwy EBL. Dzięki statycznemu przypisaniu, każda działająca aplikacja, zarówno

w obrębie lokalnym jak i globalnym, posiada informacje o tym, że wątek obsługujący zapytania skierowane do warstwy EBL będzie widnieć pod takim adresem transportowym. Kolejną częścią architektury warstwy jest wątek *EBLLocalThread* tworzony bezpośrednio w ramach przestrzeni adresowej użytkownika. Jest on odpowiedzialny za obsługę komunikacji pomiędzy instancją warstwy EBL a lokalną aplikacją, wykorzystującą funkcjonalność EBL.

3.6. Abstrakcyjna warstwa sprzętowa HAL

Główną rolą abstrakcyjnej warstwy sprzętowej w opracowanym oprogramowaniu pośredniczącym jest udostępnienie abstrakcji umożliwiającej obsługę dostępnych rozwiązań sprzętowych w module objętym kontrolą przez oprogramowanie pośredniczące. Podstawowym zadaniem warstwy sprzętowej jest umożliwienie łatwego dostępu do wybranego rozwiązania sprzętowego poprzez ogólnie dostępny interfejs oferowany przez warstwę HAL, bez potrzeby odwoływania się do specyficznych elementów związanych z danym urządzeniem. Brak specyficznych elementów w oprogramowaniu sterującym pozwala na zachowanie generyczności kodu. Abstrakcyjna warstwa sprzętowa odgrywa kluczową rolę w przypadku, w którym konieczna jest wymiana elementu sprzętowego, wykorzystywanego w gotowym oprogramowaniu. Dzięki udostępnionej abstrakcji w przypadku jednakowych wymagań funkcjonalnych elementu sprzętowego, oprogramowanie nie ulega zmianie. Dostosowana zostaje jedynie implementacja samego sterownika. Kolejnym kluczowym aspektem związanym z obsługą sprzętową realizowaną za pomocą tej warstwy jest odpowiednia synchronizacja odwołań.

3.6.1. Architektura HAL

Architektura abstrakcyjnej warstwy sprzętowej została podzielona na dwie części: część związaną z konkretnymi rozwiązaniami sprzętowymi oraz część odpowiadającą za mechanizm samej warstwy HAL.

Warstwa sprzętowa nie jest związana architektonicznie z poprzednio przedstawionymi warstwami. Nie wykorzystuje ona żadnej funkcjonalności oferowanej przez inne elementy systemu pośredniego. Abstrakcyjna warstwa sprzętowa jest całkowicie autonomiczną warstwą, pozbawioną zależności od systemu wymiany informacji. W szczególnych przypadkach mogłaby zostać wykorzystana jako osobny element wspomagający projekt jedynie pod względem odwołań do elementów sprzętowych. Użytkownik chcący odwołać się do poszczególnych urządzeń realizuje to poprzez operacje na obiekcie typu HAL.

4. Badania wydajności mechanizmów wymiany informacji LNCL oraz GNCL

Do przeprowadzenia badań opracowano dwa scenariusze testowania opracowanego systemu, służące do określenia wydajności mechanizmów wymiany danych zaimplementowanych w opracowanym oprogramowaniu pośredniczącym. Pierwszy scenariusz określa czynności prowadzące do określenia uwarunkowań czasowych związanych z wymianą informacji w obrębie lokalnym. Drugi określa czynności mające na celu określenie wydajności systemu w zakresie globalnej wymiany informacji.

4.1. Badania mechanizmu LNCL

Scenariusz obejmujący test lokalnej wymiany informacji przewiduje uruchomienie dwóch wątków A i B, znajdujących się w obrębie jednej aplikacji, będącej użytkownikiem oprogramowania pośredniego. Badania zostały przeprowadzone z wykorzystaniem jednego modułu sprzętowego Raspberry Pi 2 B v1.1. Wyniki zostały przedstawione w Tabeli 4.1.

Tabela 4.1. Wyniki testu komunikacji lokalnej.

Rozmiar pojedynczego pakietu [bajty]	256	512	1024	2048	4096
Średni czas przesyłu pojedynczego pakietu [μs]	97	102	104	141	131
Czas przesyłu informacji o rozmiarze 128MB [s]	52,28	27,51	13,98	9,46	4,41
Prędkość przesyłu [Mb/s]	19,58	37,22	73,21	108,16	231,92

4.3. Badania mechanizmu GNCL

Scenariusz testowy, związany z globalną warstwą komunikacyjną, wymagał utworzenia dwóch aplikacji testowych A oraz B. Aplikacje znajdują się w różnych węzłach systemu. Działanie aplikacji A oraz B jest analogiczne do działania wątku A i B w scenariuszu testowania lokalnej wymiany informacji. Jedyną różnicą jest to, że aplikacje nie znajdują się w obrębie tego samego modułu. W ramach badań warstwy GNCL wykonano testy obciążeniowe w następujących konfiguracjach:

- konfiguracja I: 4 x Raspberry Pi 2B v1.1.
- konfiguracja II: 8 x Raspberry Pi 2B v1.1.

Połączenie przewodowe pomiędzy modułami zostało zrealizowane za pomocą routera Mikrotik Cloud Router Switch CAS125-24G-15-2HnD-IN. Badania z wykorzystaniem powyżej wymienionych konfiguracji zostały przeprowadzone w dwóch wariantach:

- „każdy do każdego" - każdy z połączonych modułów Raspberry Pi ma za zadanie wysłać pakiet o wybranym rozmiarze do pozostałych modułów widocznych w konfiguracji sieci.
- „sekwencja" - z dostępnej listy modułów widocznych w konfiguracji sieci przed rozpoczęciem testu zostaje przygotowana struktura wymiany wiadomości pomiędzy dostępnymi modułami w układzie sekwencyjnym. Polega ona na tym, że każdy kolejny moduł posiada swojego następcę, do którego zostaje wysłana wiadomość. Pierwszy moduł na liście wysyła wiadomość do modułu drugiego. Ostatni moduł na liście wysyła wiadomość o wybranym rozmiarze do pierwszego z listy.

4.4. Wynik badań – wariant „każdy do każdego" z wykorzystaniem konfiguracji I

Poniżej przedstawiono wyniki badań dla wariantu „każdy do każdego" uruchomionego w konfiguracji składającej się z 4 modułów Raspberry Pi 2B v1.1.

Tabela 4.2 Wyniki badań dla wariantu „każdy do każdego" konfiguracja I.

Rozmiar pojedynczego pakietu (B)	256	512	2048	4096
Średni czas przesyłu pojedynczego pakietu (μs)	938	1022	1494	2172
Czas przesyłu informacji o rozmiarze 128MB (s)	493,70	269,02	98,25	71,41
Prędkość przesyłu (Mb/s)	2,07	3,80	10,42	14,33

4.5. Wynik badań wydajności w wariancie „każdy do każdego” z wykorzystaniem konfiguracji II

Poniżej przedstawiono wyniki badań wariantu „każdy do każdego” przeprowadzonego z wykorzystaniem konfiguracji zawierającej 8 modułów Raspberry Pi 2B v1.1.

Tabela 4.3 Wyniki badań dla wariantu „każdy do każdego” konfiguracja II.

Rozmiar pojedynczego pakietu [B]	256	512	2048	4096
Średni czas przesyłu pojedynczego pakietu [μs]	1639	1752	3217	3540
Czas przesyłu informacji o rozmiarze 128MB [s]	861,99	460,84	211,28	116,22
Prędkość przesyłu [Mb/s]	1,18	2,22	4,84	8,81

4.6. Wynik badań – wariant „sekwencja” z wykorzystaniem konfiguracji I

W Tabeli 4.4 przedstawiono rezultaty badań wykonanych w wariancie „sekwencja”.

Tabela 4.4 Wyniki badań dla wariantu „sekwencja” w konfiguracji I.

Rozmiar pojedynczego pakietu [B]	256	512	2048	4096
Średni czas przesyłu pojedynczego pakietu [μs]	608	670	1014	1337
Czas przesyłu informacji o rozmiarze 128MB [s]	320,93	176,58	66,75	43,98
Prędkość przesyłu [Mb/s]	3,19	5,79	15,33	23,27

4.7. Wynik badań dla wariantu „sekwencja” z wykorzystaniem konfiguracji II

Dane przedstawione w Tabeli 4.5 zawierają wyniki testu wydajnościowego globalnej wymiany informacji w wariancie „sekwencja”, przeprowadzonego z wykorzystaniem konfiguracji składającej się z 8 modułów Raspberry Pi 2B v1.1.

Tabela 4.5 Wyniki badań dla wariantu „sekwencja” w konfiguracji II.

Rozmiar pojedynczego pakietu (B)	256	512	2048	4096
Średni czas przesyłu pojedynczego pakietu (μs)	669	730	1052	1386
Czas przesyłu informacji o rozmiarze 128MB (s)	352,41	195,03	69,23	45,57
Prędkość przesyłu (Mb/s)	2,90	5,25	14,79	22,47

5. Badania układu napędowego fragmentu egzoszkieletu opartego na systemie wymiany informacji

System testowy obejmuje układ wspomagania ruchu ręki i składa się z następujących elementów:

- układu mechanicznego wraz z elementami napędowymi, czujnikami oraz elementami mocowania,
- układu sterowania, składającego się ze sterowników napędów wraz z medium umożliwiającym komunikację oraz modułów sprzętowych, odpowiadających za sterownie napędami złącz egzoszkieletu,
- programu sterującego, bazującego na opracowanym oprogramowaniu pośredniczącym, uwzględniającego modułową strukturę układu napędowego egzoszkieletu wspomaganego.

5.3. Przyjęta struktura ramienia egzoszkieletu

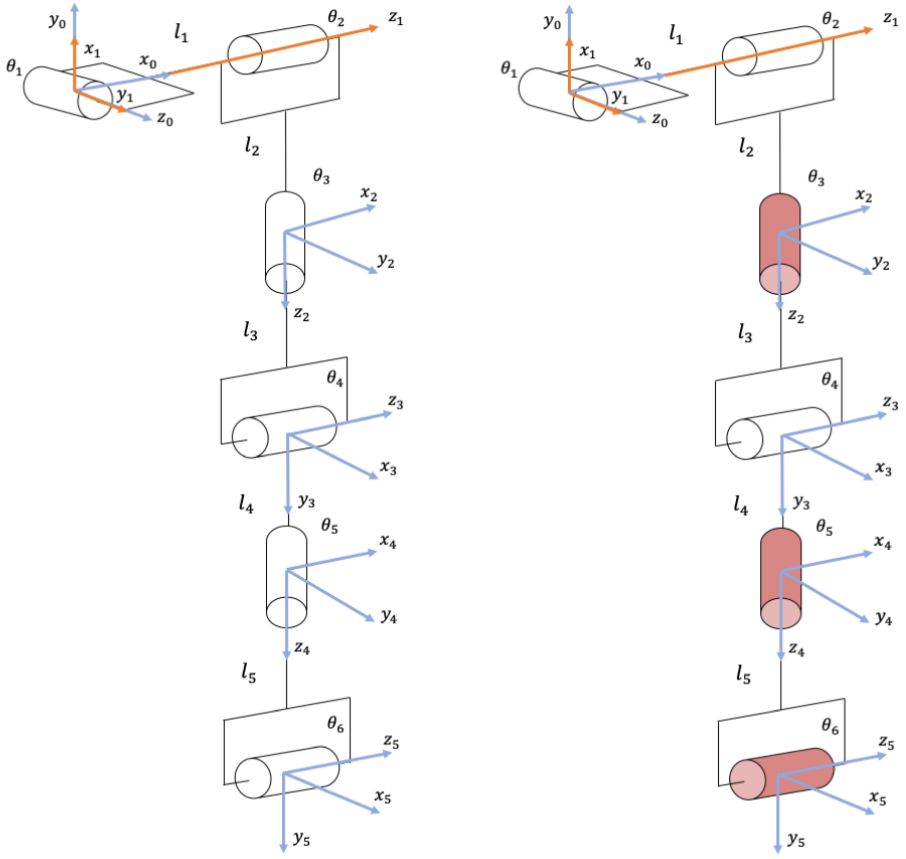
Opracowane oprogramowanie zostało zastosowane w układzie napędowym aktywnego egzoszkieletu lewej ręki, pełniącego funkcję wspomagania ruchów. Przyjęto strukturę egzoszkieletu posiadającą sześć złącz z napędami BLDC, pokazaną na rys. 5.1. Układ sterowania zaprojektowano w postaci modułowej (każdy napęd sterowany jest z dedykowanego modułu). Układ działa na podstawie sygnałów z czujników nacisku. Celem działania układu jest wzmacnianie ruchu kończyny. Ze względu na potencjalne zastosowanie egzoszkieletu do celów rehabilitacyjnych w napędach złącz wprowadzono ograniczenie zakresów ruchu, wartości momentu obrotowego i prędkości. Zgodnie z przyjętą koncepcją napędy poszczególnych złącz działają niezależnie od siebie na podstawie sygnałów z dedykowanych dla nich czujników. Pozycja manipulatora jest kontrolowana przez dodatkowy moduł z zaimplementowanym programem obliczającym położenie na podstawie zmierzonych za pomocą enkoderów inkrementalnych wartości kątów w złączach.

Wykonany w ramach pracy układ prototypowy ograniczono do trzech złącz kinematycznych o strukturze pokazanej na rys. 5.1, natomiast program sterujący przystosowano do obsługi pełnej konfiguracji egzoszkieletu zawierającej 6 złącz aktywnych. Dzięki zastosowaniu funkcjonalności oprogramowania pośredniczącego układ sterowania automatycznie rozpoznaje konfigurację prototypu, złożoną z mniejszej liczby modułów i dostosowuje sposób działania do rozpoznanej konfiguracji sprzętowej.

5.4. Wyznaczanie położenia egzoszkieletu w celu umożliwienia wprowadzenia ograniczeń zakresu ruchu

Na potrzeby pracy w układzie prototypowym zaimplementowano proste zadanie kinematyki dla konfiguracji składającej się z 6 złącz realizujących następujące ruchy lewej kończyny górnej: zgięcie, wyprost, odwodzenie, przywodzenie ramienia, rotacja zewnętrzna i

wewnętrzna, zginanie i wyprost łokcia oraz zginanie i wyprost nadgarstka. W celu sprawdzenia działania mechanizmu automatycznej detekcji napędów złącz przedstawiono również alternatywną konfigurację, reprezentującą strukturę układu kinematycznego egzoszkieletu wspomaganego w przypadku braku lub awarii poszczególnych złącz. Struktura kinematyczna egzoszkieletu została przedstawiona na rys 5.1. Po lewej stronie została przedstawiona konfiguracja pełna, ze wszystkimi dostępnymi złączami. Po prawej przedstawiono konfigurację uwzględniającą brak 3 złącz egzoszkieletu w układzie prototypowym. W konfiguracji prototypu niedostępne złącza zostały oznaczone kolorem czerwonym.



Rys. 5.1. Struktura kinematyczna pełna z 6 złączami (po lewej) oraz struktura kinematyczna prototypu, uwzględniająca brak 3 złącz (po prawej).

Brakujące w prototypie złącza realizują rotację zewnętrzną i wewnętrzną, zginanie i wyprost nadgarstka. W ramach niniejszej pracy pierwsza konfiguracja przedstawia możliwość rozbudowy prototypu egzoszkieletu wspomaganego, a druga jego aktualną strukturę, która została wykorzystana jako prototyp testowy. Oprogramowanie zostało dostosowane do obsługi pełnej struktury kinematycznej składającej się z 6 złącz. Za pomocą warstwy automatycznego wykrywania węzłów systemu sterowania oraz warstwy rozgłoszeniowej oprogramowania pośredniczącego, konfiguracja prototypu obsługiwana przez system sterowania może ulegać zmianie w trakcie działania systemu. Przykładowo, rozpoczynając pracę z prototypem posiadającym 6 złącz (rys. 5.1 konfiguracja po lewej stronie), w trakcie pracy 3 złącza ulegają awarii (rys. 5.1 konfiguracja po prawej stronie). W przypadku awarii system może dokonać automatycznej rekonfiguracji algorytmu sterowania. W przypadku awarii złącz zmienia się struktura kinematyczna układu, blokując możliwość wykonania ruchu w uszkodzonych

złączach. W ramach automatycznej rekonfiguracji system dokonuje ponownego przeliczenia prostego zadania kinematyki dla zaktualizowanej struktury egzoszkieletu (rys. 5.1 konfiguracja po prawej stronie). Niniejszy przykład został przedstawiony za pomocą Tabeli 5.1. Metodę kinematyczną prostą zaimplementowano w celu umożliwienia w przyszłości kontroli położenia oraz uwzględnienia dopuszczalnych zakresów ruchu i zadawania trajektorii ruchu. W ramach badań nie wprowadzano dodatkowych ograniczeń zakresów ruchów, jednak oprogramowanie umożliwia dostosowanie zakresów ruchu do możliwości operatora.

Tabela 5.1 Parametry DH dla łańcucha kinematycznego lewego ramienia egzoszkieletu

i	a_i	α_i	d_i	θ_i
1	0	$\frac{\pi}{2}$	0	$\theta_1 + \frac{\pi}{2}$
2	0	$-\frac{\pi}{2}$	l_1	$\theta_2 + \frac{\pi}{2}$
3	0	$\frac{\pi}{2}$	$l_2 + l_3$	θ_3
4	0	$-\frac{\pi}{2}$	0	θ_4
5	0	$\frac{\pi}{2}$	$l_4 + l_5$	θ_5

gdzie:
a_i – długość członu,
α_i- skręcenie członu,
d_i- odsunięcie przegubu,
θ_i- kąt przegubu,
$l_{1:6}$- długości poszczególnych elementów (Tabela 5.2).

Tabela 5.2 Długości elementów egzoszkieletu

Symbol	Długość [mm]	Opis długości
l_1	149	odległość od osi obrotu złącza barkowego 1 do osi złącza barkowego 2.
l_2	280	długość ramienia
l_3	240	długość przedramienia

W Tabeli 5.3 każde brakujące złącze w równaniu kinematyki prostej reprezentowane dotychczas za pomocą macierzy przekształceń jednorodnych A_n^{n+1} (I kolumna Tabeli 5.3) zostaje zastąpione przez macierz zastępczą Az_n^{n+1} (II kolumna Tabeli 5.3). Za pomocą dostępnych w oprogramowaniu pośredniczącym funkcjonalności zabieg ten pozwala na modyfikacje struktury kinematycznej egzoszkieletu w czasie działania programu sterującego.

Tabela 5.3 Macierze przekształceń jednorodnych egzoszkieletu lewej kończyny górnej oraz macierze zastępcze wykorzystane w przypadku braku lub awarii poszczególnych złącz.

Macierze przekształceń jednorodnych pełnej struktury egzoszkieletu wspomaganego	Macierze zastępcze brakujących złącz egzoszkieletu wspomaganego
$A_0^1 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$Az_0^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
$A_1^2 = \begin{bmatrix} c_2 & 0 & -s_2 & 0 \\ s_2 & 0 & c_2 & 0 \\ 0 & -1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$Az_1^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
$A_2^3 = \begin{bmatrix} c_3 & 0 & s_3 & 0 \\ s_3 & 0 & -c_3 & 0 \\ 0 & 1 & 0 & (l_2 + l_3) \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$Az_2^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & (l_2 + l_3) \\ 0 & 0 & 0 & 1 \end{bmatrix}$
$A_3^4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$Az_3^4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
$A_4^5 = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & (l_4 + l_5) \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$Az_4^5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & (l_4 + l_5) \\ 0 & 0 & 0 & 1 \end{bmatrix}$
$A_5^6 = \begin{bmatrix} c_6 & -s_6 & 0 & a_2 c_6 \\ s_6 & c_6 & 0 & a_2 s_6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$Az_5^6 = \begin{bmatrix} 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Macierz transformacji jednorodnej wyznaczono za pomocą poniższego równania:

$$\mathrm{A}_0^6 = \mathrm{A}_0^1 \mathrm{A}_1^2 \mathrm{A}_2^3 \mathrm{A}_3^4 \mathrm{A}_4^5 \mathrm{A}_5^6 \tag{5.1}$$

5.5. Napędy złącz i układ sterowania

Ze względu na zależność momentu obciążenia od położenia i kierunku ruchu ramion, w złączach kinematycznych zastosowano dyskretne regulatory quasi-adaptacyjne o strukturze bazującej na regulatorach PID, w których wartości współczynników wzmocnienia uzależnione zostały od pozycji poszczególnych złącz. W celu odpowiedniego doboru nastaw regulatorów

zastosowanych w napędach egzoszkieletu opracowano model symulacyjny napędów zainstalowanych w układzie prototypowym w programie Matlab z wykorzystaniem pakietu Simulink. Model składa się z trzech głównych elementów. Każdy z tych elementów reprezentuje lokalny układ regulacji i napęd danego złącza:

- napęd złącza barkowego nr 1, umożliwiający przywodzenie oraz odwodzenie ramienia w stawie ramiennym,
- napęd złącza barkowego nr 2, umożliwiający zginanie oraz prostowanie w stawie ramiennym,
- napęd złącza łokciowego, umożliwiający zginanie oraz prostowanie przedramienia w stawie łokciowym.

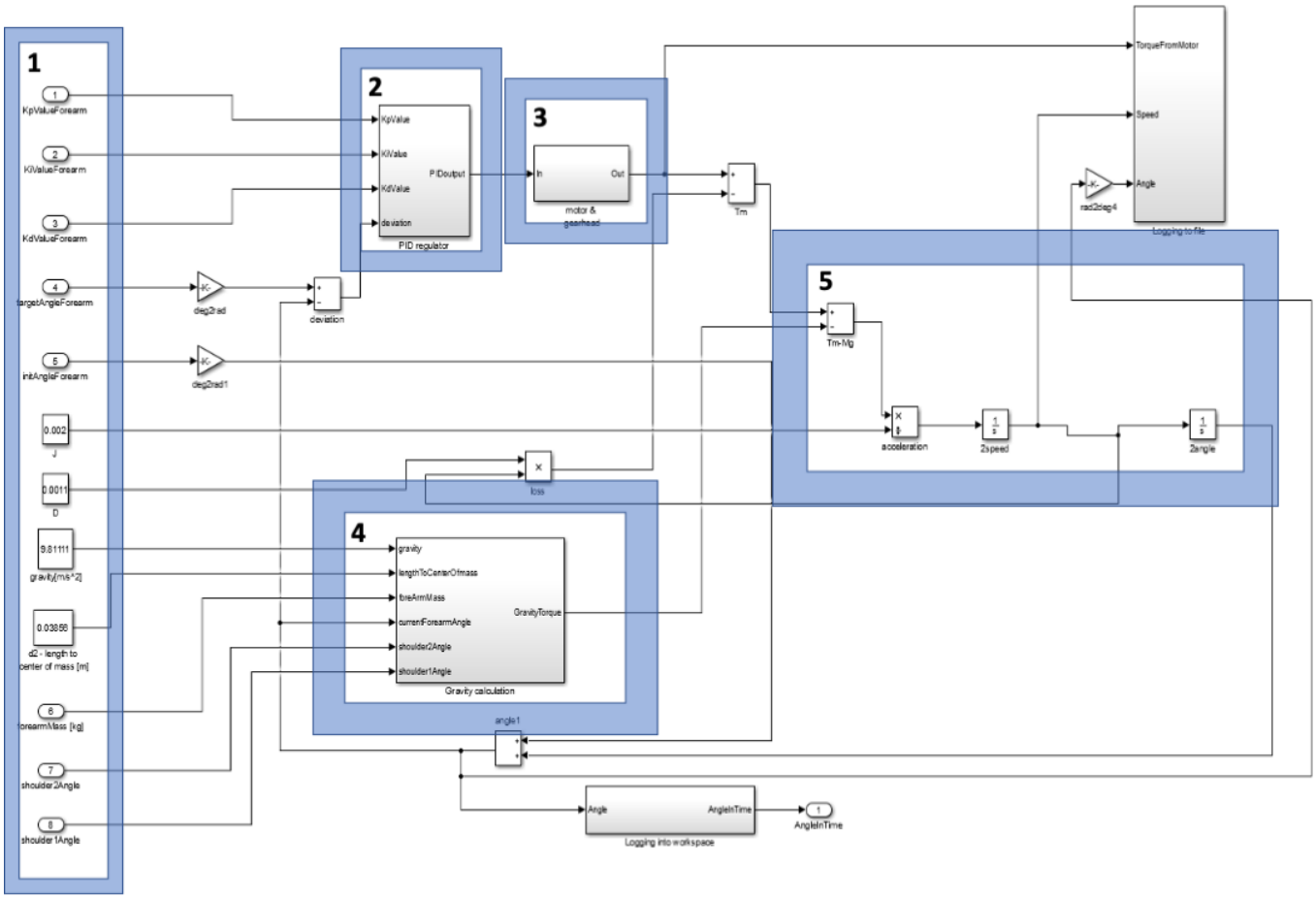
Wyżej wymienione elementy tworzą układ regulacji prototypu egzoszkieletu. Lokalne układy regulacji mają za zadanie reagować w sposób zależny od położenia i kierunku ruchu poszczególnych członów egzoszkieletu. Głównym założeniem implementacyjnym w programie symulacyjnym była modularyzacja modelu symulacyjnego zgodnie z budową układu sterowania w prototypie. Osiągnięto to przez wykorzystanie bloków funkcyjnych. Zastosowanie modularyzacji w programie symulacyjnym miało duży wpływ na projektowanie oprogramowania sterującego ramieniem egzoszkieletu. Pozwoliło to na zachowanie zbliżonej struktury programu modelu symulacyjnego do układu sterowania zaimplementowanego w prototypie egzoszkieletu. Głównym zadaniem programu symulacyjnego jest wyznaczenie parametrów regulatora w algorytmie optymalizacyjnym w różnych położeniach złącz kinematycznych. Do optymalizacji wykorzystano algorytm genetyczny. Jako kryterium optymalizacji przyjęto minimalizację czasu osiągnięcia przez napęd zadanej pozycji przy jednoczesnym ograniczeniu przeregulowań położenia docelowego złącza.

Ze względu na to, że w niniejszej pracy do napędu złącz egzoszkieletu wykorzystano gotowe zespoły napędowe, złożone z silników BLDC z wbudowanymi czujnikami położenia i przekładniami mechanicznymi oraz gotowych sterowników, nie prowadzono badań jednostek napędowych. Producent udostępnił jedynie charakterystyki wyjściowe całych napędów.

W związku z tym, że model symulacyjny został przewidziany wyłącznie do wyznaczenia zależności wartości współczynników wzmocnienia regulatorów od położenia kątowego i kierunku ruchu w złączach (podnoszenia lub opuszczania), w programie symulacyjnym nie implementowano modelu napędu, lecz zgodnie z danymi producenta zastąpiono go współczynnikiem proporcjonalności pomiędzy prądem a momentem obrotowym.

5.6. Model symulacyjny układu napędowego złącza łokciowego

Zadaniem układu regulacji w złączu łokciowym jest dostosowanie parametrów sterowania do zmian położenia kątowego przedramienia. W symulacji przyjęto, że tułów operatora znajduje się w pozycji pionowej. Założenie to jest spełnione w przypadku montażu egzoszkieletu np. do wózka inwalidzkiego. W przypadku egzoszkieletu mocowanego na ciele operatora w modelu należy dodatkowo uwzględnić kąt nachylenia tułowia. Model symulacyjny dla odcinka łokciowego został przedstawiony na rys. 5.3.



Rys. 5.3. Model symulacyjny dla złącza łokciowego z podziałem na moduły funkcjonalne.

Obszar 1 jest odpowiedzialny za przechowywanie parametrów wejściowych potrzebnych do przeprowadzenia obliczeń.

Obszar 2 odpowiedzialny jest za przemieszczanie złącza do położenia zadanego, a następnie pozycjonowanie złącza w pozycji zadanej. Obszar ten obejmuje regulator proporcjonalno-całkująco-różniczkujący. Współczynniki wzmocnienia regulatora są zależne od położenia kątowego przedramienia, wynikającego z sumy kątów w złączu barkowym 2 i złączu łokciowym oraz kąta w złączu barkowym 1.

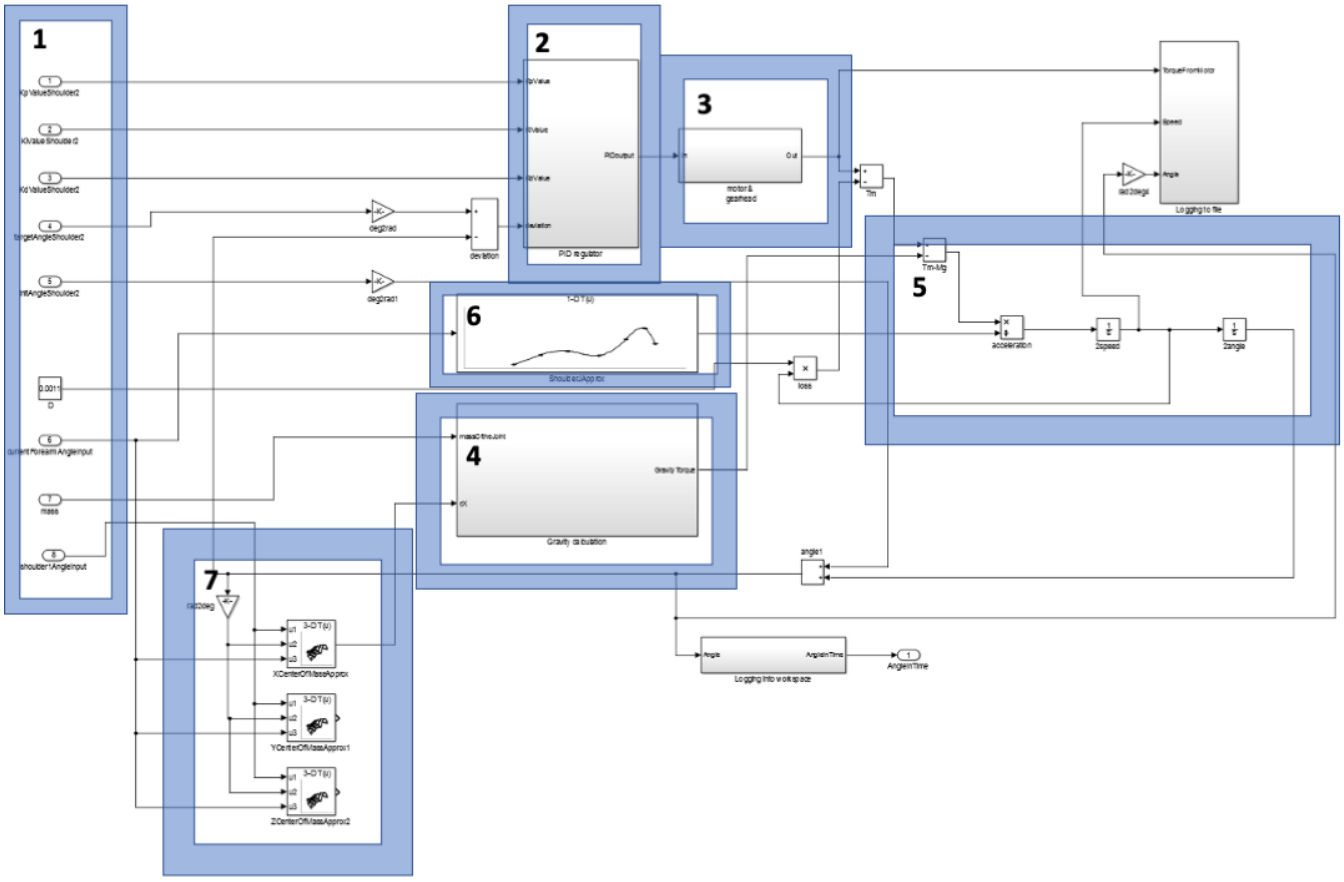
Obszar 3 odwzorowuje układ napędowy z silnikiem BLDC. Ze względu na zastosowanie gotowego zespołu napędowego napęd złącza reprezentowany jest przez współczynnik proporcjonalności pomiędzy prądem i momentem, wyznaczony na podstawie charakterystyk wyjściowych podanych przez producenta.

Obszar 4 odpowiedzialny jest za wyznaczenie chwilowych wartości momentu pochodzącego od siły ciężkości, działającego na złącze łokciowe. Kolejną wielkością wyznaczaną w obszarze 4. są opory ruchu wprowadzane przez przekładnię. Wartość momentu obrotowego mnożona jest przez współczynnik uwzględniający straty wprowadzane przez przekładnię mechaniczną. Następnie, na podstawie wartości momentu obliczane jest przyspieszenie kątowe w złączu, z jakim porusza się przedramię. Następnie kolejno wyznaczane są prędkość oraz kąt obrotu złącza. Uzyskana wartość położenia kątowego

przedramienia wykorzystywana jest w module regulatora, gdzie na jej podstawie wyznaczany jest uchyb położenia.

5.7. Model symulacyjny układu napędowego złącza barkowego 2

Układ napędowy złącza barkowego nr 2 został zaimplementowany w postaci oddzielnego bloku funkcyjnego w programie Simulink. Zadaniem układu regulacji jest dostosowanie parametrów regulatora do zmian położenia kątowego złącza barkowego 2. Założono, że tułów operatora znajduje się w pozycji pionowej. Schemat modelu symulacyjnego dla złącza barkowego 2 został przedstawiony na rys. 5.7.



Rys. 5.7. Model symulacyjny dla złącza barkowego 2 z podziałem na moduły funkcjonalne.

Obszar 1 w module symulacyjnym złącza barkowego 2. zawiera informacje na temat zadanych wartości współczynników wzmocnienia poszczególnych członów regulatora PID, wartości początkowej kąta w złączu barkowym 2. oraz wartości docelowej kąta. Obszar ten zawiera również informacje na temat wartości współczynnika oporów ruchu zastosowanej przekładni, masy ramienia z przedramieniem i aktualne pozycje kątowe złącz łokciowego i barkowego 1.

Obszar 2 reprezentuje regulator PID.

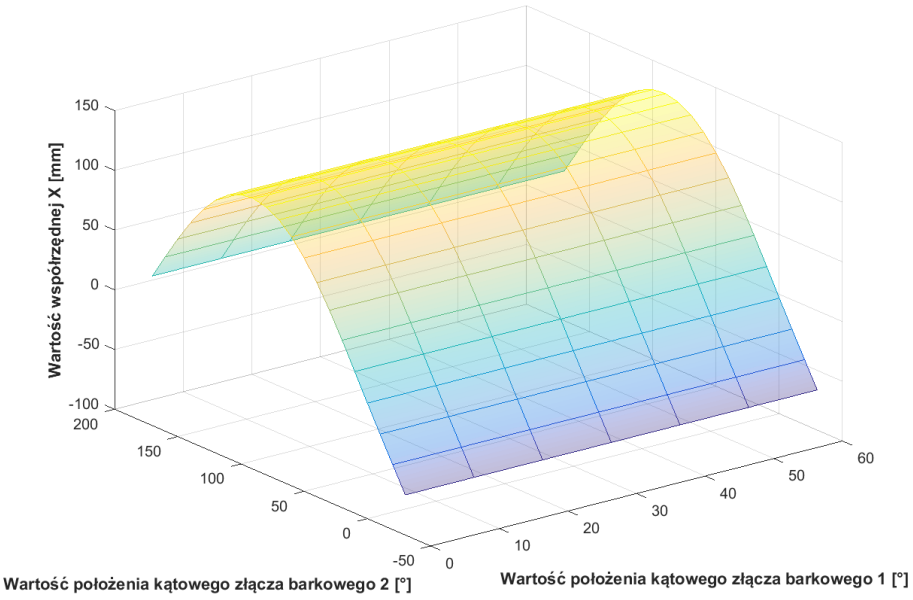
Obszar 3 zawiera informacje na temat zastosowanego napędu oraz dwóch przekładni. Informacje na temat napędu oraz przekładni zostały pobrane z danych dostarczonych przez producenta.

Obszar 4 wykorzystywany jest do wyznaczania momentu obrotowego w złączu, pochodzącego od siły ciężkości.

Obszar 5 wyznacza przyspieszenie z jakim porusza się złącze barkowe 2, jego prędkość obrotową oraz położenie kątowe.

Obszar 6 realizuje obliczenia związane z wyznaczeniem momentu bezwładności. Wartość momentu bezwładności względem złącza barkowego 2 zależna jest od położenia kątowego złącza łokciowego.

Obszar 7 zawiera elementy wyznaczające poszczególne współrzędne X, Y, Z środka masy od złącza barkowego 2. Położenie środka masy obliczane jest w celu wyznaczenia zależności momentu obrotowego pochodzącego od siły ciężkości od położenia kątowego złącz. Przykładowy wynik interpolacji współrzędnych X, Y, Z położenia środka masy został przedstawiony na rys. 5.9.



Rys. 5.9. Interpolowana zależność wartości współrzędnej X środka masy złącza barkowego 2 od położenia złącza barkowego 1 i 2 przy kącie złącza łokciowego 0°.

5.8. Model symulacyjny napędu złącza barkowego 1.

Model symulacyjny napędu złącza barkowego 1. został zawarty w osobnym bloku programu Simulink. Układ został podzielony na 7 obszarów funkcjonalnych w podobny sposób, jak w przypadku modelu złącza barkowego 2. Poszczególne elementy układu zostały przedstawione i szczegółowo opisane w pełnej wersji rozprawy doktorskiej.

5.9. Wyznaczenie nastaw regulatorów

Program symulacyjny układu napędowego prototypu egzoszkieletu został opracowany w celu wyznaczenia zależności nastaw regulatorów PID w napędach poszczególnych złącz od położenia kątowego członów egzoszkieletu względem pozycji początkowej. Model nie uwzględnia zmian pozycji wynikających z pochylenia tułowia. W celu wyznaczenia optymalnych wartości nastaw regulatorów zastosowano algorytm genetyczny. Sposób ewolucyjnego podejścia do rozwiązywania problemów został zaproponowany początkowo przez amerykańskiego naukowca Johna Henry'ego Hollanda. [121] Algorytm genetyczny jest metodą optymalizacyjną wykorzystującą ewolucję w kierunku lepszych rozwiązań. [121] W celu implementacji algorytmu genetycznego w opracowanym programie symulacyjnym wykorzystano bibliotekę udostępniającą funkcjonalność algorytmu genetycznego dostępną w pakiecie Matlab. Zadaniem algorytmu jest wyznaczanie optymalnych wartości współczynników wzmocnienia poszczególnych członów regulatora K_p, K_i, K_d w zadanych pozycjach złącz. Algorytm został skonfigurowany tak, że liczebność populacji wynosi 30 przy zadanej liczbie 100 generacji, osiągnięcie której stanowi warunek zakończenia obliczeń. Parametry te ustalono na podstawie wstępnie przeprowadzonych obliczeń. Wartość funkcji celu wyznaczana jest na podstawie parametrów przebiegu i wykorzystuje odchylenie standardowe wartości chwilowych od wartości zadanej. W procesie optymalizacji funkcja celu obliczana zgodnie z (5.16) jest minimalizowana.

$$f_k = \sqrt{\frac{\sum_{i=1}^{n}(x_i - x_z)^2}{n-1}} \tag{5.16}$$

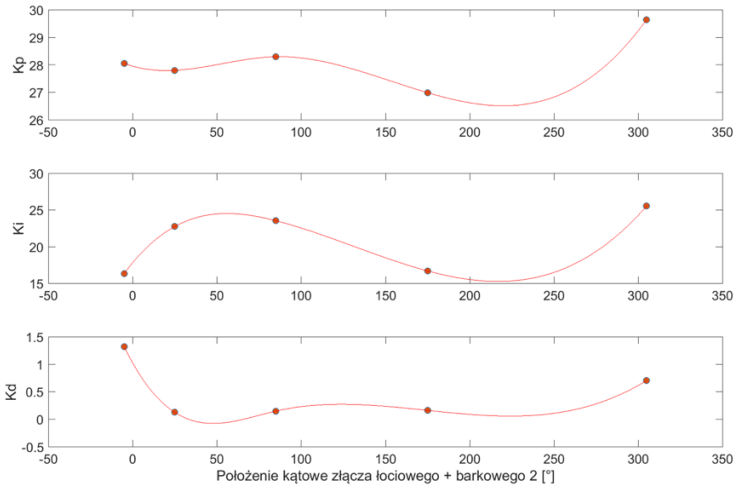
gdzie:
n - liczba próbek z przebiegu symulacji ruchu złącza,
x_i – wartość kąta w złączu dla i-tej iteracji,
x_z - wartość docelowa kąta w złączu.

Na bazie przeprowadzonych wstępnie obliczeń przyjęto następujące zakresy zmienności współczynników wzmocnień: $K_p \in < 0,35 >, K_i \in < 0,35 >, K_d \in < 0,6 >$.

5.10. Wyniki doboru parametrów regulatorów

Podczas optymalizacji wyznaczano przebiegi zmian położenia kątowego złącz w zadanych zakresach kątowych, przy różnych wartościach współczynników wzmocnienia. Zastosowana metoda doboru nastaw regulatora nie uwzględnia ograniczeń związanych z wytrzymałością konstrukcji mechanicznej wykonanej w ramach niniejszej pracy. W prototypie nie jest możliwe osiągnięcie tak dużych przyspieszeń ze względu na bezpieczeństwo operatora i wytrzymałość układu mechanicznego. Uzyskiwane w prototypie rzeczywiste wartości prędkości i przyspieszeń są ograniczane w układzie sterowania poprzez wprowadzenie ograniczenia wartości prędkości i momentu (poprzez ograniczenie maksymalnej wartości prądu) oraz w trybie wspomagania przez możliwości wykonania ruchu przez operatora, ponieważ sterowanie realizowane jest na podstawie sygnałów z czujników nacisku.

W celu adaptacji wartości współczynników wzmocnienia regulatorów PID do wartości chwilowych kątów obrotu w złączach wykonano obliczenia w wąskich zakresach zmian kątów obrotu w różnych pozycjach członów egzoszkieletu. Uzyskane wyniki umożliwiły utworzenie tablicy wartości współczynników wzmocnień wyznaczonych w tych punktach, które następnie interpolowano. Interpolowane wartości wykorzystano w układzie sterowania, uzależniając wartości współczynników wzmocnień regulatorów od położenia członów egzoszkieletu. Przykładową zależność współczynników wzmocnień regulatorów uzyskaną w wyniku interpolacji pokazano na rys. 5.39.



Rys. 5.39. Uzyskane zależności współczynników wzmocnień regulatora złącza łokciowego od położenia kątowego, przy kącie złącza barkowego 1. wynoszącym 0°, przy kącie złącza barkowego 2. równym 0°, dla ruchu w górę.

6. Stanowisko do badań eksperymentalnych

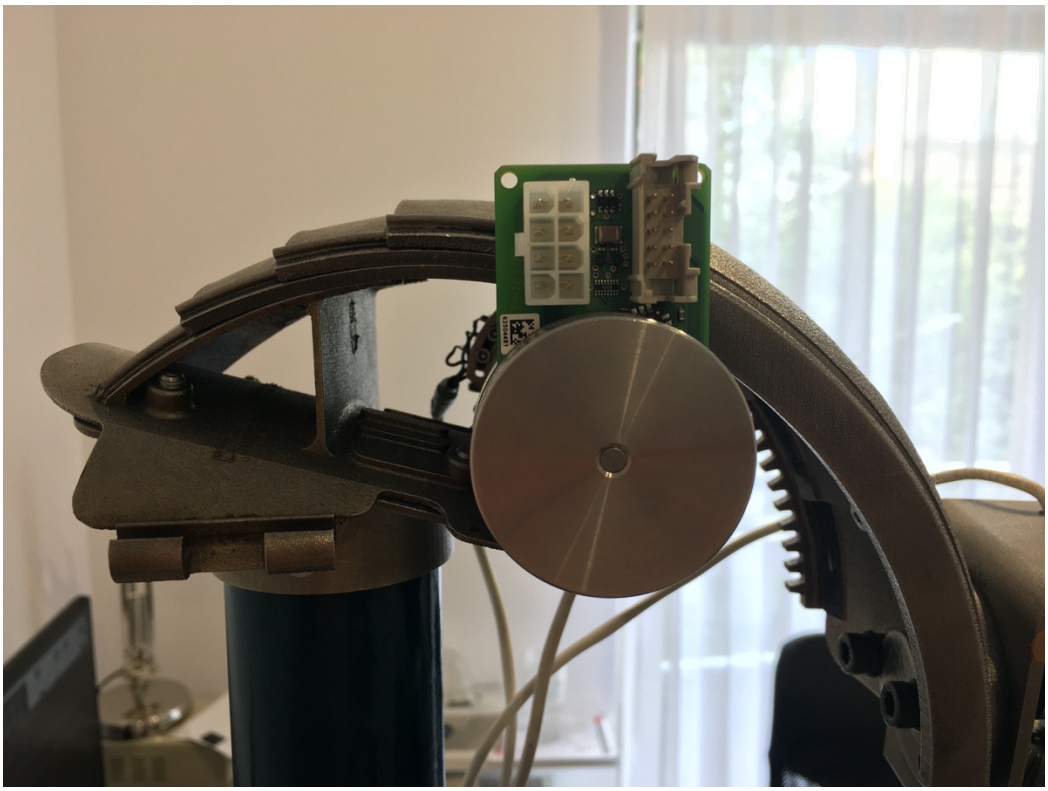
Stanowisko do badań eksperymentalnych zostało zaprojektowane oraz wykonane na potrzeby realizacji niniejszej rozprawy doktorskiej. Wykonana konstrukcja jest uproszczonym fragmentem przyjętego modelu górnej części egzoszkieletu. Zawiera elementy konstrukcyjne pozwalające na wykonanie wybranych ruchów lewej ręki operatora. Układ kinematyczny posiada trzy stopnie swobody mechanicznej: dwa w złączu barkowym, jako przeguby 1 i 2, oraz jedno w stawie łokciowym. Zdjęcie opracowanej konstrukcji zaprezentowano na rys. 6.1.

Rys. 6.1. Prototyp egzoszkieletu lewej kończyny górnej zrealizowany w ramach pracy.

Podczas testów konstrukcja egzoszkieletu została zamontowana w specjalnie przygotowanym uchwycie, umożliwiającym jej przytwierdzenie do powierzchni roboczej stołu. Uchwyt ten widoczny jest na rys. 6.1 Uchwyt o zbliżonej konstrukcji może być wykorzystany do zamocowania egzoszkieletu, np. do wózka inwalidzkiego. W przypadku przeprowadzenia testu konstrukcji zamontowanej na operatorze, należy wyeliminować uchwyt, a następnie wykorzystać specjalnie zaprojektowane pasy, pozwalające na montaż urządzenia na ciele operatora. Wymaga to jednak wykonania konstrukcji mechanicznej z lekkich materiałów.

Konstrukcja egzoszkieletu została podzielona funkcjonalnie na trzy części: złącze barkowe 1, złącze barkowe 2 oraz złącze łokciowe. Złącze barkowe 1. zostało zrealizowane z wykorzystaniem elementów tworzących wachlarz, pozwalających na jego składanie oraz rozkładanie. Zastosowanie takiego rozwiązania pozwoliło na uzyskanie zgodności położenia osi obrotu złącza barkowego 1. z osią obrotu barku człowieka. Umożliwiło to uniknięcie sytuacji wymuszania nienaturalnych ruchów człowieka związanych z przesunięciem osi obrotu barku i złącza egzoszkieletu. Element realizujący złącze barkowe 1. został wyposażony w

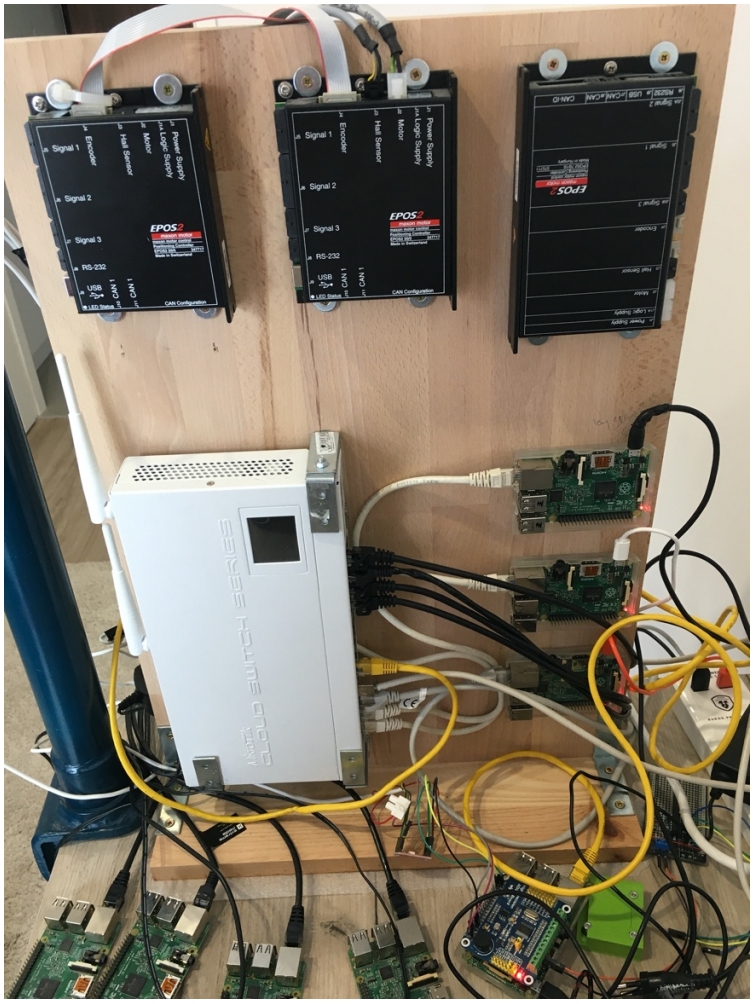
mechaniczny ogranicznik zakresu ruchu oraz czujniki krańcowe, informujące układ sterowania o osiągnięciu pozycji skrajnej. Napęd złącza barkowego 1. został zrealizowany poprzez wykonanie uzębienia wewnętrznego na wachlarzu zewnętrznym. Element napędzany jest silnikiem BLDC o mocy 50W, wyposażonym w przekładnię planetarną o przełożeniu 66:1 oraz koło zębate. Prowadzenie wachlarzy zrealizowano poprzez wykonanie prowadnic po obu stronach każdego elementu wachlarza. Wachlarz został zakończony stelażem, umożliwiającym umiejscowienie go na barku operatora. Stelaż został wyposażony w szybkozłącze i zaczepy pozwalające na umiejscowienie uprzęży na ciele człowieka. Całość została wykonana w technologii przyrostowej druku 3D. Wachlarz zewnętrzny został wyposażony w wałek. Złącze barkowe 2. wykonano poprzez zastosowanie przekładni stożkowej o przełożeniu 1:1. Unieruchomienie jednego z kół na wałku wachlarza powoduje wykonanie ruchu złącza barkowego 2. Obudowa złącza barkowego 2. została ułożyskowana za pomocą dwóch łożysk kulkowych. W tej części konstrukcji egzoszkieletu ograniczenie mechaniczne zakresu ruchu zostało zrealizowane za pomocą występu na wachlarzu zewnętrznym oraz rowka w korpusie. Do napędu złącza barkowego 2. wykorzystano silnik BLDC o mocy 90W wraz z przekładnią planetarną o przełożeniu 66:1. Napęd został umiejscowiony w dolnej części obudowy zewnętrznej złącza barkowego 2. Na wałku napędowym przekładni zostało zamocowane koło stożkowe. Z uwagi na różne rozmiary poszczególnych części kończyny górnej człowieka wynikających z budowy ciała operatora, umożliwiono regulację długości odcinka ramiennego. Za pomocą obejmy umieszczonej na przekładni silnika złącza barkowego 2. podłączono układ mechaniczny realizujący ruch złącza łokciowego. W napędzie złącza łokciowego wykorzystano przekładnię stożkową, połączoną z jednej strony z przekładnią planetarną o przełożeniu 43:1, a z drugiej z częścią ramienną. Do napędu złącza łokciowego zastosowano silnik BLDC o mocy 50 W. Koło napędzane połączone zostało sztywno z przedramieniem. W tym złączu zastosowano również mechaniczne ograniczenie zakresu ruchu. Większość elementów zostało wykonanych w technologii druku 3D. Dodatkowo wykorzystano fragmenty typowej ortezy dostępnej na rynku. Układy mechaniczne złącza barkowego 2. oraz złącza łokciowego zostały wyposażone w uchwyty pozwalające na zamocowanie uprzęży mocującej do kończyny górnej operatora. Złącze barkowe 1 zostało przedstawione na rys. 6.4.



Rys. 6.4. Widok boczny złącza barkowego 1. przedstawiający silnik BLDC wraz z modułem elektronicznym oraz fragment systemu mocowania pasa.

6.4. Stanowisko do badań adaptacyjnego systemu wymiany informacji

Stanowisko do badań prędkości transmisji danych zbudowane na potrzeby niniejszej pracy zawiera osiem modułów Raspberry Pi 2 B v 1.1. W układzie sterowania prototypu wykorzystane zostały trzy z nich. Moduły zostały połączone ze sobą za pomocą routera Mikrotik Cloud Router Switch CAS125-24G-15-2HnD-IN oraz kabla UTP ze złączem RJ45. [117] Oprogramowanie sterujące jest wgrywane do modułów i uruchamiane za pomocą komputera podłączonego do routera. Wszelkie komendy przekazywane do modułów są transportowane za pomocą protokołu SSH. Oprogramowanie wgrywane jest do poszczególnych modułów z wykorzystaniem protokołu SCP. Stanowisko to wyposażone zostało dodatkowo w trzy sterowniki silników BLDC firmy MAXON. Dwa sterowniki EPOS 50/5 przeznaczone są do sterowania pracą napędów złącza 1. oraz 3. W złączach tych zastosowane zostały silniki o mocy 50W. Trzeci sterownik typu EPOS 70/5 steruje pracą silnika w złączu 2. o mocy 90W. Opracowany układ sterowania napędami konstrukcji prototypowej pokazano na rys. 6.11.



Rys. 6.11. Układ sterowania napędami złącz prototypu egzoszkieletu.

6.5. Oprogramowanie sterujące

Oprogramowanie sterujące wykorzystuje do działania funkcjonalności oferowane przez opracowany adaptacyjny system wymiany informacji. Sterowanie układem prototypowym odbywa się za pomocą aplikacji *KinematicsApplication*. Aplikacja ta została połączona z biblioteką NodeAPI oprogramowania pośredniczącego, a w kodzie źródłowym wykorzystany został zestaw prototypów funkcji niniejszej biblioteki. Oprogramowanie to odpowiedzialne jest za sterowanie obiektem zgodnie z parametrami wejściowymi. Sterowanie napędami może odbywać się w dwóch trybach.

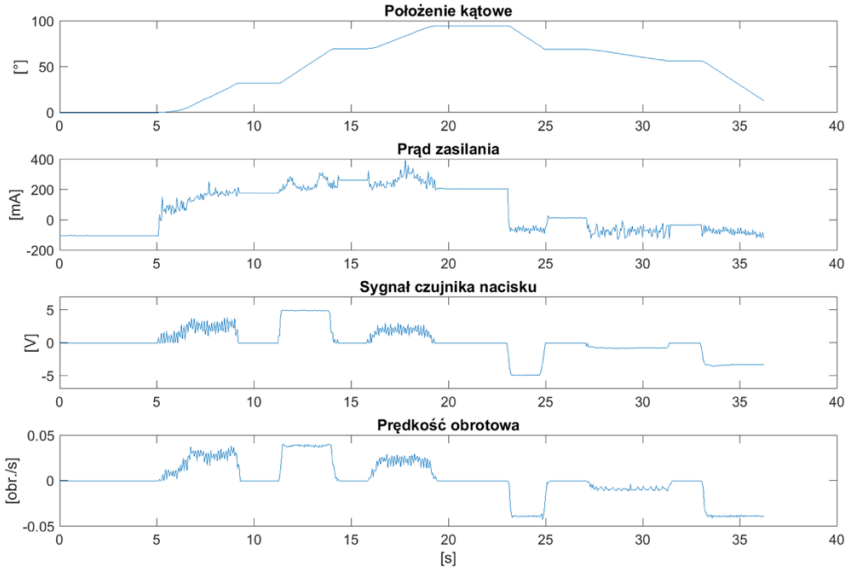
- *Tryb nadążny* - tryb ten bazuje na sygnałach z zestawu czujników FSR (*ang. Force Sensing Resistor*), wykorzystanych w prototypie górnej części egzoszkieletu. W tym trybie pracy obiekt nadąża za ruchem danej części ciała operatora, np. przedramienia.

- *Tryb rehabilitacyjny* - wariant bazujący na predefiniowanych trajektoriach ruchu. Zakres ruchu wprowadzany jest do programu w postaci zestawu kątów początkowych oraz końcowych, definiowanych oddzielnie dla każdego napędu. Tryb ten został uwzględniony w formie uproszczonej. Możliwa jest jego rozbudowa o zadawanie ciągłej trajektorii ruchu.

Niezależnie od trybu pracy w programie definiowana jest również prędkość maksymalna z jaką może poruszać się każde złącze oraz maksymalna wartość prądu zasilania silników (pośrednio maksymalna wartość momentu obrotowego).

6.5.10. Badania eksperymentalne prototypu egzoszkieletu

Na rys. 6.12 przedstawiono przykładowy przebieg sygnałów z czujników dostępnych dla złącza łokciowego w układzie prototypowym.



Rys. 6.12. Przebiegi sygnałów z czujników nacisku, położenia kątowego, prędkości kątowej oraz prądu silnika złącza łokciowego w przypadku nierównomiernego obciążenia czujnika nacisku i wykonania ruchu zginania i prostowania przedramienia.

7. Wnioski

Celem badań podjętych w trakcie realizacji niniejszej pracy było opracowanie i zbadanie oprogramowania pośredniczącego, oraz możliwości jego wykorzystania w układzie sterowania. Podczas etapu przygotowawczego określono plan składający się z trzech etapów realizacji. Pierwszy etap obejmował 3 wstępne obszary czynności przygotowawczych (rozdział 2, pkt. 1 – 3), które pozwoliły na sformułowanie tezy pracy. Czynności przygotowawcze pozwoliły na dokładną analizę rangi problemu oraz określenie potrzeb w zakresie oprogramowania pośredniczącego. Pozwoliły również na przegląd dostępnych rozwiązań, charakteryzujących się zbliżoną funkcjonalnością (rozdział 1.1 do 1.5), a także określenie własnej koncepcji oprogramowania pośredniczącego (rozdział 2.1). Następnie zdefiniowano potencjalny obszar zastosowania niniejszego oprogramowania pośredniego (rozdział 2.2). Wszelkie czynności zdefiniowane w etapie wstępnym zostały wykonane zgodnie z przyjętymi założeniami. Czynności badawcze określone w rozdziale 1 (pkt. 1 – 11), niezbędne do udowodnienia przyjętej tezy zostały również przeprowadzone. W rozdziale 3.2 przedstawiono opracowaną koncepcję oraz implementację lokalnego systemu wymiany informacji (czynność badawcza nr 1). W rozdziale 3.3 przedstawiono informacje na temat opracowanego globalnego podsystemu wymiany informacji (czynność badawcza nr 2). Opis implementacji abstrakcyjnej warstwy sprzętowej (czynność badawcza nr 3), został zawarty w rozdziale 3.6. Rozdział 3.5 zawiera opracowanie koncepcji oraz przedstawienie implementacyjnej części podsystemu dystrybucji zdarzeń EBL (czynność badawcza nr 4). Podsystem detekcji elementów sieci NDL został przedstawiony w rozdziale 3.4 (czynność badawcza nr 5). W celu sprawdzenia systemów komunikacyjnych przeprowadzono badania wydajności modułów GNCL oraz LNCL. Ich wyniki zostały przedstawione w rozdziale 4 (czynność badawcza nr 6). Badania zostały przeprowadzone w zakresie dostępnego sprzętu (8 modułów Raspberry Pi). Model kinematyczny wraz z symulacją działania fragmentu egzoszkieletu został opracowany w celu umożliwienia kontroli położenia egzoszkieletu. Został on przedstawiony w rozdziale 5 (czynność badawcza nr 7). Wyniki oraz przebieg optymalizacji parametrów regulatora wraz z opracowanym modelem napędów zaprezentowane zostały w rozdziale 5.9 oraz 5.10 (czynność badawcza nr 8). Prototyp egzoszkieletu został przedstawiony w rozdziale 6 (czynność badawcza nr 9). Aplikacje sterujące egzoszkieletem, wykorzystanym w niniejszej pracy jako przykład układu modułowego, zostały przedstawione w rozdziale 6.5 (czynność badawcza nr 10). Przeprowadzono badania pracy układu prototypowego na stanowisku badawczym (czynność badawcza nr 11), wyniki przedstawiono w rozdziale 6.5.10. Spełniono również cele szczegółowe pracy (rozdział 2, pkt 1 - 3) takie, jak:

- opracowanie koncepcji działania adaptacyjnego systemu wymiany informacji posiadającego cechy i funkcjonalność oprogramowania pośredniczącego wraz z warstwami abstrakcji sprzętowej, dystrybucji zdarzeń, detekcji elementów sieci i jego implementacja programowa (rozdział 2.1),
- budowa rozproszonego systemu wymiany informacji (rozdział 3),
- badanie wydajności warstw komunikacyjnych adaptacyjnego systemu wymiany informacji (rozdział 4),
- opracowanie i budowa prototypu egzoszkieletu kończyny górnej (rozdział 5-6),

- opracowanie modułowego systemu sterowania prototypu egzoszkieletu, bazującego na opracowanym i wykonanym adaptacyjnym systemie wymiany informacji, przeznaczonego do zastosowań rehabilitacyjnych, umożliwiającego wspomaganie wykonywania ruchów, wykonywanie ruchów zaprogramowanych, zachowanie podstawowych zasad bezpieczeństwa systemu sterowania (rozdział 6.5).

Wyniki badań zaprezentowane w niniejszej pracy pozwalają na stwierdzenie, że system może realizować zadania wymiany informacji w rozproszonych systemach sterowania posiadających zdefiniowane wymagania komunikacyjne nie pozwalające na przekroczenie czasu 1 ms dla transmisji pojedynczego pakietu. System może zostać wykorzystany jako podstawa programowa, przyspieszająca proces wytwarzania oprogramowania sterującego. Oprogramowanie pośredniczące opracowane w ramach niniejszej pracy udostępnia mechanizmy wspomagające osobę projektującą system rozproszony lub modułowy w kwestii obsługi zdarzeń oraz definiowania złożonych scenariuszy naprawczych. Według autora największymi osiągnięciami w pracy są:

- opracowanie autorskiego oprogramowania pośredniczącego składającego się z warstw realizujących czynności wspomagające tworzenie oprogramowania sterującego oraz jego implementacja programowa,
- opracowanie algorytmów testowania wraz z przeprowadzeniem badań wydajnościowych warstw komunikacyjnych LNCL i GNCL,
- opracowanie i wykonanie prototypu egzoszkieletu kończyny górnej,
- opracowanie modelu symulacyjnego napędów i układów sterowania oraz przeprowadzenie optymalizacji parametrów regulatorów,
- implementacja programów sterujących na bazie opracowanego oprogramowania pośredniczącego i wykonanie badań układu prototypowego.

Realizacja wszystkich zdefiniowanych czynności przygotowawczych oraz czynności badawczych, a także wszelkich celów szczegółowych, pozwoliła na zdefiniowanie stwierdzenia, **że możliwe** jest opracowanie programowej warstwy komunikacyjnej, realizującej zadania wymiany informacji w rozproszonych systemach sterowania i urządzeniach o budowie modułowej, pełniącej funkcje kontrolne w zakresie konfiguracji i podstawowych zasad bezpieczeństwa systemu sterowania, będącej warstwą pośrednią pomiędzy aplikacjami sterującymi a warstwą sprzętową i systemem operacyjnym.

Bibliografia – wybrane pozycje

[3] Al-Dhief F., Sabri N., Abdul L., Nurul M., Nik Abd M., Nik N., Abd Ghani M., Mohammed M., Al-Haddad R.N., Dawood Y., Ghani M., Ibrahim Obaid O.: "Performance comparison between TCP and udp protocols in different simulation scenarios", International Journal of Engineering & Technology 7, 2018, s.172-176. doi: 10.14419/ijet.v7i4.36.23739

[10] Bakken D.: "Middleware", Encyclopedia of Distributed Computing, J. Urban and P. Dasgupta, Eds., Kluwer Academic, 2001, źródło: https://eecs.wsu.edu/~bakken/middleware.htm, dostęp: 20.03.2022.

[15] Cervera E.: "Try to Start It! The Challenge of Reusing Code in Robotics Research", IEEE Robotics and Automation Letters vol. 4 no. 1, s. 49-56, 2019, doi: 10.1109/LRA.2018.2878604.

[30] Doose D., Grand C., Lesire C.: "MAUVE Runtime: A Component-Based Middleware to Reconfigure Software Architectures in Real-Time", 2017 First IEEE International Conference on Robotic Computing (IRC), 2017, s. 208-211, doi: 10.1109/IRC.2017.47.

[34] Elkady A., Sobh T.: "Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography." Journal of Robotics 2012 Vol 12, Hindawi Publishing Corporation, s.1-15, doi:10.1155/2012/959013

[56] Kerrisk M.: "The Linux Programming Interface A Linux and Unix System Programming Handbook", No Starch Press, 2010, s.1235, ISBN: 978-1-59327-220-3.

[57] Krakowiak S.: "Middleware Architecture with Patterns and Frameworks, Sacha Krakowiak", 2007.

[63] Liberatore V.: "Implementation challenges in real-time middleware for distributed autonomous systems", 2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06), 2006, s. 1-8, doi: 10.1109/SMC-IT.2006.36.

[67] Ma Q., Zou Y., Zhang T.: "Study of service robot architecture based on middleware and abstract environment", 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2012, s. 1200-1205, doi: 10.1109/ROBIO.2012.6491133.

[79] Milanovic A., Srbljic S., Sruk V.: "Performance of UDP and TCP communication on personal computers,", 2000 10th Mediterranean Electrotechnical Conference. Information Technology and Electrotechnology for the Mediterranean Countries. Proceedings. MeleCon 2000 (Cat. No.00CH37099), 2000, s. 286-289, doi: 10.1109/MELCON.2000.880422.

[81] Mohamed N., Al-Jaroodi J.: "Characteristics of middleware for networked collaborative robots", 2008 International Symposium on Collaborative Technologies and Systems, 2008, s. 524-531, doi: 10.1109/CTS.2008.4543973.

[84] Nagata K., Wakita Y., Yamanobe N., Ando N.: "Development of assistive robotic arm system with Robot Technology Middleware (RTM)", 19th International Symposium in Robot and Human Interactive Communication, 2010, s. 737-742, doi: 10.1109/ROMAN.2010.5598728.

[85] Naidoo N., Bright G. and Stopforth R.: "Navigation and control of cooperative mobile robots using a robotic middleware platform", 2016 12th IEEE International Conference on Control and Automation (ICCA), 2016, s. 927-932, doi: 10.1109/ICCA.2016.7505397.

[87] Nesnas I. A. D. et al..: “CLARAty: Challenges and Steps toward Reusable Robotic Software”, International Journal of Advanced Robotic Systems, 2006, doi: 10.5772/5766.

[98] Pulikottil T. B., Caimmi M., D'Angelo M. G., Biffi E., Pellegrinelli S., Tosatti L. M.: "A Voice Control System for Assistive Robotic Arms: Preliminary Usability Tests on Patients", 2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob), 2018, s. 167-172, doi: 10.1109/BIOROB.2018.8487200.

[100] Quigley M., Conley K., Gerkey B., Faust J., Foote T., Leibs J., Wheeler R., Ng A.: "ROS: an open-source Robot Operating System", ICRA Workshop on Open Source Software, 2009, s.1-6.

[105] Sarabia M., Ros R., Demiris Y.: "Towards an open-source social middleware for humanoid robots", 2011 11th IEEE-RAS International Conference on Humanoid Robots, 2011, s. 670-675, doi: 10.1109/Humanoids.2011.6100883.

[106] Sato R., Matsuda H., Fujieda M., Hata H., Ming A.: "Design and implementation of common platform for small humanoid robots", 2013 IEEE International Conference on Mechatronics and Automation, 2013, s. 855-860, doi: 10.1109/ICMA.2013.6618027.

[117] Strzelczyk P., Tomczewski K., "Data Exchange Platform Dedicated to Distributed Control Systems", Pomiary Automatyka Robotyka 21 nr 3/2017, s. 53-62, doi: 10.14313/PAR_225/53.

[118] Strzelczyk P., Tomczewski K.: "Realizacja mechanizmu lokalnej wymiany informacji w ramach platformy komunikacyjnej w rozproszonych systemach sterowania",Pomiary Automatyka Robotyka R. 20 nr 1/2016, s.65-68, doi: 10.14313/PAR_219/65.

[121] Tang K.S., Man K.F., Kwong S, He Q.: "Genetic Algorithms and their Applications", IEEE Signal Processing Magazine 13 nr 6, 1996, s 22-37., doi: 10.1109/79.543973.

[131] Wang W., Suga Y., Sugano S.: "Human in Loop Integration of an Arm Mounted Wheelchair Robot Based on RT Middleware", ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), 2010, s. 1-6, ISBN: 978-3-8007-3273-9.

[134] Wheeb A.: "Performance Comparison of Transport Layer Protocols", International Journals of Advanced Research in Computer Science and Software Engineering 5, s.121-125, 2015, ISSN: 2277 128X.