

**WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ**

Kacper Rychel

Robot mobilny z nawigacją autonomiczną oparty na Arduino

PRACA INŻYNIERSKA

Opiekun pracy:
Dr inż. Mariusz Mączka

Rzeszów, 2026

Spis treści

1. Wprowadzenie	6
2. Zakres teoretyczny z dziedziny robotyki	8
2.1. Przegląd stanu wiedzy w zakresie robotów mobilnych	8
2.1.1. Modelowanie ruchu robota	8
2.1.2. Odtwarzanie danych sensorycznych	9
2.1.3. Technologia mapowania SLAM	10
2.2. Klasyfikacja i architektura robotów mobilnych	10
2.2.1. Klasyfikacja według sposobu lokomocji robota	10
2.2.2. Klasyfikacja według autonomiczności robota	11
2.2.3. Klasyfikacja według architektury rozwiązania	11
2.3. Modelowanie ruchu robotów mobilnych	12
2.4. Układy napędowe i sterowanie silnikami	15
2.4.1. Sterowanie silnikami robota	16
2.5. Percepcja otoczenia	17
2.6. Zaawansowane metody nawigacji	18
3. Montaż robota	21
3.1. Komponenty robota	21
3.1.1. Mikrokontroler Arduino Uno	21
3.1.2. Sterownik silników L293D	22
3.1.3. Silniki oraz koła	23
3.1.4. Czujnik ultradźwiękowy HC-SR04	24
3.1.5. Element obrotowy MicroServo 9G	25
3.1.6. Ogniwka litowo-jonowe 18650	26
3.1.7. Przełącznik kołyskowy	28
3.2. Wykorzystane materiały	28
3.3. Budowa robota	29
3.3.1. Lutowanie wyprowadzeń silników	30
3.3.2. Montaż silników oraz kół	30
3.3.3. Montaż mikrokontrolera Arduino Uno	31
3.3.4. Montaż osłony L293D	32
3.3.5. Montaż uchwytu na baterie	32

3.3.6. Montaż ogniw zasilających	33
3.3.7. Montaż serwomechanizmu z kątownikiem	34
3.3.8. Montaż czujnika ultradźwiękowego odległości	35
3.3.9. Schemat połączeń robota	35
4. Kod programu	38
4.1. Opis fragmentów kodu	38
4.1.1. Inicjalizacja komponentów	38
4.1.2. Funkcja pomiaru odległości	39
4.1.3. Funkcje ruchu robota	40
4.1.4. Pętla główna programu	42
5. Testy robota w środowisku	44
5.1. Jazda po prostej linii	45
5.2. Pokonywanie nierówności podłoża	46
5.3. Omijanie przeszkód w polu wykrywania czujnika	47
5.4. Omijanie przeszkód poza polem wykrywania czujnika	48
5.5. Omijanie przeszkód o nieregularnym kształcie	50
5.6. Omijanie przeszkód o chropowatej powierzchni	52
5.7. Omijanie przeszkód o gąbczastej strukturze	53
5.8. Zachowanie robota w sytuacji całkowitego otoczenia przeszkodami	54
6. Podsumowanie	57
Literatura	59

1. Wprowadzenie

Roboty mobilne należą do najbardziej dynamicznie rozwijających się obszarów współczesnej robotyki oraz inżynierii mechatronicznej. Ich znaczenie systematycznie rośnie wraz z postępem technologii obliczeniowych, czujnikowych oraz energetycznych, co przekłada się na coraz szerszy zakres zastosowań. Roboty mobilne wykorzystywane są obecnie m.in. w przemyśle, logistyce, eksploracji środowisk niebezpiecznych, a także w systemach autonomicznych. Zgodnie z definicją Międzynarodowej Federacji Robotyki, robot mobilny jest systemem mechatronicznym zdolnym do samodzielnego przemieszczania się w przestrzeni oraz realizacji określonych zadań bez stałego połączenia z infrastrukturą stacjonarną[?].

Rozwój mikrokontrolerów oraz popularyzacja platform open-source istotnie obniżyły próg wejścia w projektowaniu systemów robotycznych. Jak zauważa Monk, dostępność tanich i dobrze udokumentowanych platform, takich jak Arduino, umożliwia szybkie prototypowanie rozwiązań robotycznych również w warunkach edukacyjnych i amatorskich[?]. Dzięki temu możliwe stało się praktyczne łączenie zagadnień z zakresu elektroniki, programowania oraz automatyki w jednym projekcie.

Autonomia robota mobilnego rozumiana jest jako zdolność do samodzielnego podejmowania decyzji na podstawie informacji pozyskiwanych z czujników. System autonomiczny integruje procesy percepji otoczenia, podejmowania decyzji oraz sterowania ruchem. Siegwart, Nourbakhsh i Scaramuzza wskazują, że nawet w przypadku prostych algorytmów reaktywnych niezbędna jest spójna współpraca tych elementów, aby robot mógł funkcjonować w sposób przewidywalny i bezpieczny[?].

Centralnym elementem sterującym robota mobilnego jest mikrokontroler, odpowiedzialny za przetwarzanie danych sensorycznych oraz generowanie sygnałów sterujących elementami wykonawczymi. Platforma Arduino UNO, oparta na mikrokontrolerze ATmega328P, należy do najczęściej wykorzystywanych w projektach edukacyjnych ze względu na bogatą dokumentację techniczną, dostępność bibliotek programistycznych oraz szerokie wsparcie społeczności open-source[?]. Zastosowanie gotowych bibliotek znaczco skraca czas implementacji algorytmów sterowania oraz obsługi czujników.

Istotną rolę w systemach autonomicznych odgrywają czujniki umożliwiające pozyskiwanie informacji o otoczeniu. W robotach niskokosztowych powszechnie wykorzystuje się czujniki ultradźwiękowe, które pozwalają na pomiar odległości na podstawie

czasu przelotu fali akustycznej. Popularnym rozwiązaniem w projektach edukacyjnych jest moduł HC-SR04, który charakteryzuje się niewielkim kosztem, prostą obsługą oraz łatwą integracją z mikrokontrolerami[?]. Dokumentacja producenta wskazuje, że dokładność pomiaru jest wystarczająca dla podstawowych algorytmów nawigacyjnych, choć należy uwzględnić wpływ właściwości powierzchni odbijających oraz warunków środowiskowych[?].

Ruch robota mobilnego realizowany jest najczęściej za pomocą napędu różnicowego, składającego się z dwóch niezależnie sterowanych silników prądu stałego. Taki układ umożliwia realizację manewrów poprzez różnicowanie prędkości obrotowych kół. Jak wskazują Jones, Flynn i Seiger, napęd różnicowy cechuje się niewielką złożonością mechaniczną i sterowniczą, co czyni go jednym z najczęściej stosowanych rozwiązań w robotach mobilnych[?]. Do sterowania silnikami prądu stałego stosuje się układy typu mostek H, takie jak L293D, umożliwiające zmianę kierunku obrotów oraz regulację prędkości z wykorzystaniem sygnału PWM. Tego typu układy stanowią podstawowy element wykonawczy w systemach sterowania napędami niskiej mocy[?].

Praca podzielona jest na trzy rozdziały. W drugim rozdziale przedstawiony jest zakres teoretyczny zagadnień potrzebnych do skompletowania robota. Trzeci rozdział przeznaczony jest skupieniu najważniejszych elementów budowania fizycznej podstawy robota. W czwartym rozdziale opisany został kod programowy robota.

Celem niniejszej pracy jest zaprojektowanie i wykonanie autonomicznego robota mobilnego opartego na platformie Arduino UNO, zdolnego do samodzielnego poruszania się oraz unikania przeszkód na podstawie pomiarów odległości. Zakres pracy obejmuje analizę teoretyczną zagadnień robotyki mobilnej, projekt i realizację układu sprzętowego, implementację algorytmu sterowania oraz przeprowadzenie testów funkcjonalnych opracowanego rozwiązania.

2. Zakres teoretyczny z dziedziny robotyki

Robotyka mobilna stanowi jedną z kluczowych dziedzin współczesnej automatyki i mechatroniki, łącząc zagadnienia z zakresu mechaniki, elektroniki, informatyki oraz teorii sterowania. Jej przedmiotem są systemy zdolne do samodzielnego przemieszczania się w przestrzeni oraz podejmowania decyzji na podstawie informacji pozyskiwanych z otoczenia. W zależności od stopnia autonomii oraz złożoności środowiska pracy roboty mobilne mogą realizować zarówno proste zadania reaktywne, jak i zaawansowane algorytmy nawigacji i planowania ruchu.

Zakres teoretyczny niniejszej pracy obejmuje wybrane zagadnienia niezbędne do zrozumienia zasad działania autonomicznego robota mobilnego o prostej strukturze sprzętowej. Szczególny nacisk położono na aspekty związane z kinematyką ruchu, podstawowymi metodami sterowania napędem oraz wykorzystaniem czujników odległości w procesie omijania przeszkód. Omówione zagadnienia stanowią teoretyczne uzasadnienie przyjętych rozwiązań konstrukcyjnych i programowych, a jednocześnie wyznaczają ramy dla dalszych rozważań projektowych i implementacyjnych.

2.1. Przegląd stanu wiedzy w zakresie robotów mobilnych

Postęp technologiczny w obszarze robotyki mobilnej, obserwowany w ostatnich dekadach, doprowadził do wykształcenia się wielu zróżnicowanych podejść konstrukcyjnych oraz algorytmicznych. Dobór konkretnego rozwiązania zależy przede wszystkim od przeznaczenia robota, charakterystyki środowiska pracy oraz dostępnych zasobów sprzętowych i obliczeniowych. Jak podkreśla się w literaturze przedmiotu, pomimo dynamicznego rozwoju systemów autonomicznych i algorytmów sztucznej inteligencji, w dalszym ciągu istotną rolę odgrywają rozwiązania proste i deterministyczne, szczególnie w systemach o charakterze edukacyjnym oraz prototypowym [?].

2.1.1. Modelowanie ruchu robota

Jednym z podstawowych zagadnień w robotyce mobilnej jest modelowanie ruchu robota, stanowiące fundament projektowania algorytmów sterowania oraz nawigacji. W przypadku najczęściej spotykanego napędu różnicowego ruch robota opisywany jest za pomocą nieliniowego modelu kinematycznego, w którym prędkość liniowa oraz prędkość kątowa platformy są bezpośrednio zależne od prędkości obrotowych kół napędowych. Campion, Bastin oraz D'Andrea-Novel wskazują, że model kinematyczny robota

z napędem różnicowym cechuje się prostotą implementacji, jednak wprowadza istotne ograniczenie w postaci braku możliwości realizacji ruchu bocznego[?]. Ograniczenie to wymusza stosowanie odpowiednich strategii planowania trajektorii oraz manewrowania, zwłaszcza w środowiskach o ograniczonej przestrzeni roboczej.

W praktycznych realizacjach robotów mobilnych, szczególnie w systemach dysponujących niewielką mocą obliczeniową, sterowanie ruchem realizowane jest zazwyczaj w oparciu o regulatory dyskretne, bez jawnego uwzględniania pełnego modelu dynamicznego. Regulacja prędkości silników prądu stałego odbywa się najczęściej przy użyciu modulacji szerokości impulsu PWM, natomiast zmiana kierunku obrotów realizowana jest poprzez odpowiednie sterowanie mostkiem H. Jak zauważają Horowitz i Hill, takie podejście, mimo swojej prostoty, jest w pełni wystarczające w systemach, w których nie jest wymagana wysoka precyzja pozycjonowania ani zaawansowana regulacja momentu obrotowego[?].

2.1.2. Odtwarzanie danych sensorycznych

Istotnym elementem wpływającym na skuteczność autonomii robota mobilnego jest sposób pozyskiwania oraz przetwarzania danych sensorycznych. W systemach niskokosztowych dominują rozwiązania wykorzystujące pojedyncze lub kilka czujników odległości, których wskazania są bezpośrednio wykorzystywane do podejmowania decyzji ruchowych. Takie podejście określone jest w literaturze mianem nawigacji reaktywnej, w której robot nie buduje globalnej reprezentacji środowiska, lecz reaguje wyłącznie na aktualnie wykryte przeszkody[?]. Metody te, choć nieoptimalne z punktu widzenia długości trajektorii czy efektywności ruchu, charakteryzują się dużą odpornością na zmiany otoczenia oraz niewielkimi wymaganiami obliczeniowymi.

W algorytmach nawigacji reaktywnej powszechnie stosowane są czujniki ultradźwiękowe, cenione za prostą zasadę działania oraz łatwość integracji z popularnymi mikrokontrolerami. Badania porównawcze czujników odległości wskazują, że dokładność pomiarów ultradźwiękowych jest wystarczająca do wykrywania przeszkód o rozmiarach porównywalnych z wymiarami robota, jednak ulega pogorszeniu w przypadku powierzchni pochłaniających fale akustyczne lub ustawionych pod znacznym kątem względem osi pomiaru[?]. Z tego względu zaleca się stosowanie odpowiednich marginesów bezpieczeństwa w algorytmach decyzyjnych oraz filtrację wyników pomiarów w celu zwiększenia odporności systemu na błędne odczyty[?].

2.1.3. Technologia mapowania SLAM

W bardziej zaawansowanych systemach robotów mobilnych stosowane są techniki lokalizacji i mapowania jednociesnego (SLAM), wykorzystujące dane z czujników LiDARowych¹, kamer wizyjnych oraz jednostek inercyjnych. Metody te umożliwiają jednoczesną budowę mapy otoczenia oraz precyzyjne określanie położenia robota, jednak wiążą się z wysokimi wymaganiami obliczeniowymi oraz znaczną złożonością implementacyjną [?]. Jak podkreśla Thrun, implementacja algorytmów SLAM w systemach edukacyjnych jest często nieuzasadniona, gdyż poziom skomplikowania może przesłaniać podstawowe zagadnienia inżynierskie, takie jak sterowanie ruchem czy integracja czujników[?].

2.2. Klasyfikacja i architektura robotów mobilnych

Roboty mobilne mogą być klasyfikowane według wielu różnych kryteriów, co wynika z dużej różnorodności ich konstrukcji, przeznaczenia oraz środowisk pracy. W literaturze przedmiotu podkreśla się, że brak jednej uniwersalnej klasyfikacji robotów mobilnych jest konsekwencją interdyscyplinarnego charakteru tej dziedziny, łączącej zagadnienia z zakresu mechaniki, elektroniki, informatyki oraz teorii sterowania[?]. Najczęściej spotykane podejścia klasyfikacyjne obejmują podział ze względu na mechanizm lokomocji, stopień autonomii, środowisko eksploatacji oraz zakres realizowanych zadań.

2.2.1. Klasyfikacja według sposobu lokomocji robota

Jednym z podstawowych i najbardziej intuicyjnych kryteriów klasyfikacji robotów mobilnych jest sposób ich poruszania się. W tym ujęciu wyróżnia się roboty kołowe, gąsienicowe, kroczące oraz konstrukcje hybrydowe. Roboty kołowe, wyposażone w jedno lub więcej kół napędowych, charakteryzują się wysoką sprawnością energetyczną, niewielkimi stratami energii wynikającymi z oporów toczenia oraz prostotą konstrukcji mechanicznej. Jak wskazuje Siegwart, łatwość sterowania oraz stosunkowo niski koszt wykonania sprawiają, że roboty kołowe dominują w zastosowaniach edukacyjnych, przemysłowych oraz badawczych[?]. Ich istotnym ograniczeniem jest jednak ograniczona zdolność poruszania się po nierównym lub nieuporządkowanym terenie, co zawęża zakres potencjalnych zastosowań.

¹LiDAR (Light Detection and Ranging) - metoda pomiarowa używana do określania precyzyjnego dystansu obiektu.

Alternatywą dla konstrukcji kołowych stanowią roboty gąsienicowe, które dzięki większej powierzchni styku z podłożem cechują się lepszą przyczepnością oraz zdolnością pokonywania przeszkód terenowych. Z tego względu znajdują one zastosowanie w robotach eksploracyjnych, ratowniczych oraz wojskowych, gdzie wymagana jest wysoka stabilność ruchu w trudnych warunkach środowiskowych. Jak zauważa Borenstein, zwiększone opory ruchu oraz większa złożoność mechaniczna układów gąsienicowych prowadzą do wyższego zużycia energii oraz utrudniają precyzyjne sterowanie ruchem w porównaniu z robotami kołowymi[?].

Najbardziej złożoną grupę robotów mobilnych pod względem konstrukcyjnym i algorytmicznym stanowią roboty kroczące. Ich lokomocja opiera się na sekwencyjnym przemieszczaniu kończyn, co umożliwia poruszanie się w terenie niedostępnym dla robotów kołowych i gąsienicowych, takim jak schody, gruzowiska czy obszary o znaczących nierównościach. Projektowanie robotów kroczących wymaga zaawansowanego modelowania dynamiki, precyzyjnego sterowania ruchem oraz znaczących zasobów obliczeniowych[?].

2.2.2. Klasyfikacja według autonomiczności robota

Istotnym kryterium klasyfikacji robotów mobilnych, niezależnym od mechanizmu lokomocji, jest stopień autonomii systemu. W tym ujęciu wyróżnia się roboty zdalnie sterowane, półautonomiczne oraz autonomiczne. Roboty zdalnie sterowane wymagają stałej ingerencji operatora i nie podejmują samodzielnych decyzji, roboty autonomiczne są zdolne do samodzielnej analizy danych sensorycznych, planowania działań oraz realizacji zadań bez bezpośredniego nadzoru człowieka, natomiast roboty półautonomiczne są hybrydowym rozwiązaniem łączącym cechy maszyn autonomicznych oraz zdalnie sterowanych. Jak wskazuje Arkin, poziom autonomii robota jest ściśle powiązany z zastosowanymi algorytmami decyzyjnymi oraz architekturą systemu sterowania[?].

2.2.3. Klasyfikacja według architektury rozwiązania

Z punktu widzenia architektury systemowej robot mobilny stanowi złożony system mechatroniczny, który można opisać jako zbiór współpracujących ze sobą warstw funkcjonalnych. Wyróżnia się warstwę sensoryczną, odpowiedzialną za pozyskiwanie informacji o stanie robota i jego otoczeniu, warstwę decyzyjną, w której realizowane są algorytmy sterowania i nawigacji, oraz warstwę wykonawczą, obejmującą układy napędowe i ele-

menty wykonawcze. Taki podział funkcjonalny jest powszechnie stosowany w literaturze i umożliwia modularne oraz skalowalne projektowanie systemów robotycznych[?].

Warstwa sensoryczna obejmuje zestaw czujników, takich jak czujniki odległości, enkodery, czujniki inercyjne czy kamery wizyjne, których zadaniem jest dostarczanie informacji o położeniu robota, jego ruchu oraz otoczeniu. Rodzaj i jakość danych sensorycznych w istotny sposób wpływają na możliwości percepcyjne systemu oraz skuteczność podejmowanych decyzji.

Warstwa decyzyjna stanowi centralny element architektury robota mobilnego. W prostych konstrukcjach realizowana jest bezpośrednio na mikrokontrolerze, który przetwarza dane sensoryczne i generuje sygnały sterujące w czasie rzeczywistym. W bardziej zaawansowanych systemach warstwa ta może być rozproszona pomiędzy kilka jednostek obliczeniowych, takich jak komputery jednopłytowe lub procesory dedykowane, co umożliwia implementację złożonych algorytmów planowania trajektorii, lokalizacji czy uczenia maszynowego[?]. Rozwiązania te zwiększą jednak złożoność systemu oraz zapotrzebowanie na energię.

Warstwa wykonawcza obejmuje elementy odpowiedzialne za fizyczną realizację ruchu robota, w tym silniki, przekładnie mechaniczne oraz układy sterujące. Jej projekt musi uwzględnić wymagania mechaniczne i elektryczne, a także ograniczenia wynikające z zastosowanego źródła zasilania. Jak podkreślają Jones i Flynn, prawidłowa integracja warstwy wykonawczej z pozostałymi warstwami systemu jest kluczowa dla zapewnienia stabilnego, przewidywalnego i bezpiecznego zachowania robota mobilnego[?].

2.3. Modelowanie ruchu robotów mobilnych

Modelowanie ruchu robota mobilnego stanowi jeden z kluczowych etapów projektowania systemów sterowania oraz nawigacji, gdyż umożliwia formalny, matematyczny opis zależności pomiędzy sygnałami sterującymi a rzeczywistym ruchem platformy. W robotyce mobilnej wyróżnia się dwa podstawowe podejścia do opisu ruchu: modelowanie kinematyczne oraz dynamiczne. Model kinematyczny koncentruje się na zależnościach geometrycznych i prędkociowych, pomijając wpływ sił oraz momentów działających na robota, natomiast model dynamiczny uwzględnia masę, bezwładność oraz oddziaływanie z otoczeniem[?]. W przypadku większości robotów edukacyjnych i prototyp-

powych stosuje się modele kinematyczne, które zapewniają wystarczającą dokładność przy znacznie mniejszej złożoności obliczeniowej.

Najczęściej spotykanym rozwiązaniem konstrukcyjnym w robotach mobilnych jest napęd różnicowy, składający się z dwóch niezależnie sterowanych kół napędowych umieszczonych po przeciwnych stronach platformy oraz jednego lub więcej elementów podparcia biernego. Ruch robota wyposażonego w taki napęd może być opisany za pomocą zestawu równań kinematycznych, w których prędkość liniowa oraz prędkość kątowa platformy są bezpośrednią funkcją prędkości obrotowych lewego i prawego koła. Model ten cechuje się prostotą formalną oraz intuicyjnością, co czyni go szczególnie przydatnym w projektach dydaktycznych oraz na wczesnych etapach prototypowania[?].

Formalny opis kinematyki robota z napędem różnicowym prowadzi do nielinowego układu równań, w którym pozycja robota opisywana jest zazwyczaj za pomocą współrzędnych płaskich oraz kąta orientacji względem przyjętego układu odniesienia. Nieliniowość modelu wynika z faktu, że orientacja robota wpływa bezpośrednio na kierunek jego ruchu postępowego. W praktyce oznacza to konieczność ciągłego uwzględniania aktualnego położenia i orientacji robota podczas sterowania ruchem, nawet w przypadku realizacji prostych manewrów, takich jak jazda po łuku lub zmiana kierunku jazdy[?].

Istotnym ograniczeniem wynikającym z kinematyki napędu różnicowego jest brak możliwości realizacji ruchu bocznego. Robot nie jest w stanie przemieszczać się w kierunku prostopadłym do osi swojej orientacji bez wcześniejszego wykonania obrotu. W literaturze ograniczenia tego typu określane są mianem ograniczeń nieholonomicznych i mają one istotny wpływ na projektowanie algorytmów sterowania oraz nawigacji[?]. Ograniczenia nieholonomiczne powodują, że trajektorie ruchu robota muszą spełniać określone warunki ciągłości i gładkości, co komplikuje planowanie ruchu w środowiskach o ograniczonej przestrzeni.

Konsekwencją występowania ograniczeń nieholonomicznych jest konieczność stosowania odpowiednich strategii manewrowania, szczególnie podczas omijania przeszkód. W prostych systemach autonomicznych często wykorzystuje się sekwencje ruchów elementarnych, takich jak jazda do przodu, obrót w miejscu oraz cofanie, które w połączeniu umożliwiają realizację bardziej złożonych manewrów. Jak podkreśla Thrun, takie

podejście jest wystarczające w środowiskach o niewielkiej złożoności i stanowi rozsądny kompromis pomiędzy prostotą implementacji a funkcjonalnością systemu[?].

W praktycznych realizacjach robotów mobilnych model kinematyczny wykorzystywany jest głównie pośrednio, jako podstawa do projektowania algorytmów sterowania niskiego poziomu. Zamiast ciągłego rozwiązywania równań ruchu stosuje się sterowanie dyskretne, w którym decyzje podejmowane są w kolejnych krokach czasowych na podstawie aktualnych danych sensorycznych. Takie podejście jest szczególnie korzystne w systemach opartych na mikrokontrolerach o ograniczonej mocy obliczeniowej, gdzie kluczowe znaczenie ma krótki czas reakcji oraz stabilność działania[?].

W literaturze podkreśla się również, że w robotach edukacyjnych dokładność modelu kinematycznego ma często znaczenie drugorzędne w porównaniu z jego czytelnością i łatwością interpretacji. Błędy wynikające z poślizgu kół, nierówności podłożu czy niedokładności wykonania mechanicznego są zazwyczaj akceptowalne, o ile algorytm sterowania wykazuje odpowiednią odporność na zakłócenia[?]. Z tego względu w wielu konstrukcjach rezygnuje się z enkoderów oraz sprzężenia zwrotnego pozycji na rzecz prostych algorytmów otwartej pętli, opartych na czasie trwania ruchu oraz zadanej prędkości.

Ograniczenia kinematyczne robotów kołowych mają również bezpośredni wpływ na sposób planowania trajektorii. W systemach o wyższym stopniu zaawansowania stosuje się algorytmy planowania ruchu uwzględniające nieholonomiczność, takie jak planowanie w przestrzeni konfiguracyjnej czy metody oparte na krzywych Dubinsa. Jak zauważa LaValle, metody te wymagają precyzyjnego modelu robota oraz znacznych zasobów obliczeniowych, co w praktyce ogranicza ich zastosowanie w prostych platformach mobilnych[?].

Z tego względu w systemach o charakterze edukacyjnym i prototypowym często rezygnuje się z planowania globalnego na rzecz lokalnych algorytmów decyzyjnych, bazujących na bieżących pomiarach sensorycznych. Podejście to, mimo że nie gwarantuje optymalności trajektorii, umożliwia skuteczne poruszanie się robota w nieznanym środowisku i jest zgodne z ograniczeniami sprzętowymi oraz obliczeniowymi prostych platform mobilnych.

2.4. Układy napędowe i sterowanie silnikami

Układ napędowy robota mobilnego stanowi podstawowy element warstwy wykonawczej systemu, bezpośrednio odpowiedzialny za realizację poleceń generowanych przez warstwę decyzyjną. Jego zadaniem jest przekształcenie sygnałów sterujących w rzeczywisty ruch platformy przy zachowaniu możliwie wysokiej sprawności energetycznej, stabilności działania oraz przewidywalnego zachowania robota. W praktycznych realizacjach robotów mobilnych, najczęściej stosowane są silniki prądu stałego z magnesami trwałymi, które łączą prostotę sterowania z korzystnymi parametrami mechanicznymi i niewielkimi wymaganiami sprzętowymi.

Silniki prądu stałego charakteryzują się w przybliżeniu liniową zależnością momentu obrotowego od prądu twornika oraz zależnością prędkości obrotowej od napięcia zasilania. Właściwości te umożliwiają stosunkowo łatwą regulację prędkości obrotowej bez konieczności stosowania złożonych algorytmów regulacji. W robotach mobilnych silniki te zazwyczaj współpracują z przekładniami redukcyjnymi, które zwiększały dostępny moment obrotowy kosztem prędkości.

Regulacja prędkości obrotowej silników prądu stałego realizowana jest najczęściej przy wykorzystaniu modulacji szerokości impulsu PWM (Pulse Width Modulation). Metoda ta polega na okresowym załączaniu i wyłączaniu napięcia zasilającego silnika z ustaloną częstotliwością oraz zmiennym współczynnikiem wypełnienia. Ze względu na indukcyjność uzwojeń silnika oraz jego bezwładność mechaniczną sygnał PWM postrzegany jest jako napięcie o wartości średniej proporcjonalnej do wypełnienia impulsu. Jak podkreślają Horowitz i Hill, modulacja PWM umożliwia efektywną regulację prędkości przy minimalnych stratach mocy, co czyni ją standardowym rozwiązaniem w systemach sterowania napędami niskonapięciowymi[?].

Istotnym zagadnieniem w sterowaniu napędem robota mobilnego jest możliwość zmiany kierunku obrotów silnika. Wymaga to zastosowania układu umożliwiającego odwrócenie polaryzacji napięcia zasilającego, co najczęściej realizowane jest za pomocą mostka H. Układ ten składa się z zestawu elementów przełączających, takich jak tranzystory bipolarne lub MOSFET², które pozwalają na sterowanie kierunkiem przepływu prądu przez silnik. W rozwiązaniach edukacyjnych powszechnie stosowane są scalone

²MOSFET - podstawowa technologia produkcji większości półprzewodników z izolowaną bramką stosowanych w komputerach.

układy mostków H, integrujące elementy mocy oraz podstawowe zabezpieczenia w jednej obudowie.

Jednym z najczęściej wykorzystywanych układów tego typu jest L293D, który umożliwia niezależne sterowanie dwoma silnikami prądu stałego. Układ ten pozwala na realizację zmiany kierunku obrotów poprzez odpowiednie stany logiczne na wejściach sterujących, natomiast regulacja prędkości realizowana jest przez doprowadzenie sygnału PWM do wejścia aktywującego. Takie rozwiązanie jest szczególnie korzystne, gdyż umożliwia prostą i czytelną implementację sterowania ruchem bez konieczności projektowania własnych układów mocy.

W robotach mobilnych o niewielkich rozmiarach istotne znaczenie ma również sposób zasilania układu napędowego. Silniki prądu stałego charakteryzują się stosunkowo dużymi prądami rozruchowymi, które mogą prowadzić do chwilowych spadków napięcia oraz zakłóceń pracy układów logicznych. Z tego względu w literaturze zaleca się separację zasilania części wykonawczej i logicznej lub stosowanie odpowiednich filtrów oraz kondensatorów buforujących[?]. W prostych konstrukcjach edukacyjnych zagadnienie to bywa często pomijane, jednak jego uwzględnienie znacząco poprawia stabilność i niezawodność całego systemu.

2.4.1. Sterowanie silnikami robota

Sterowanie silnikami w robotach mobilnych może być realizowane zarówno w pętli otwartej, jak i zamkniętej. W systemach edukacyjnych i prototypowych dominującym rozwiązaniem jest sterowanie w pętli otwartej, w którym prędkość oraz czas trwania ruchu ustalane są na podstawie zadanych wartości sygnału PWM. Brak sprzężenia zwrotnego z enkoderów oznacza, że system nie kompensuje odchyłek wynikających z poślizgu kół, nierówności podłoża czy różnic parametrów silników. Jak zauważają Horowitz i Hill, w wielu zastosowaniach tego typu uproszczenie jest w pełni akceptowalne i nie wpływa istotnie na funkcjonalność systemu [?].

W bardziej zaawansowanych konstrukcjach stosuje się enkodery obrotowe, które umożliwiają pomiar prędkości oraz kąta obrotu kół. Pozwala to na implementację regulatorów prędkości, najczęściej typu PID, oraz na dokładniejsze sterowanie ruchem robota. Należy jednak podkreślić, że zastosowanie sprzężenia zwrotnego wiąże się ze wzrostem złożoności zarówno sprzętowej, jak i programowej, co może być niepożądane w projektach o charakterze dydaktycznym.

Z punktu widzenia architektury systemowej układ napędowy stanowi interfejs pomiędzy warstwą decyzyjną a fizycznym ruchem robota. Prostota jego realizacji ma istotne znaczenie dla przejrzystości projektu oraz możliwości dalszej rozbudowy systemu. W konstrukcjach opartych na mikrokontrolerach, takich jak Arduino, wykorzystanie standardowych bibliotek do obsługi PWM oraz gotowych sterowników silników pozwala skoncentrować się na zagadnieniach algorytmicznych, zamiast na niskopoziomowych detalach sprzętowych.

2.5. Percepcja otoczenia

Percepcja otoczenia stanowi jeden z fundamentalnych elementów autonomii robota mobilnego, gdyż to na jej podstawie system podejmuje decyzje dotyczące ruchu oraz reakcji na zmieniające się warunki środowiskowe. W przeciwnieństwie do systemów sterowanych zdalnie, robot autonomiczny musi samodzielnie interpretować sygnały pochodzące z czujników i przekształcać je w działania zgodne z przyjętą strategią sterowania. W literaturze podkreśla się, że jakość procesu percepacji ma bezpośredni wpływ na bezpieczeństwo oraz skuteczność poruszania się robota, niezależnie od stopnia zaawansowania zastosowanych algorytmów decyzyjnych[?].

Wśród czujników wykorzystywanych w systemach reaktywnych szczególnie szerokie zastosowanie znajdują czujniki ultradźwiękowe. Ich zasada działania polega na emisji krótkiego impulsu fali akustycznej o wysokiej częstotliwości oraz pomiarze czasu, po którym sygnał powraca do odbiornika po odbiciu od przeszkody. Znając prędkość rozchodzenia się dźwięku w powietrzu, możliwe jest wyznaczenie odległości do obiektu.

Badania porównawcze czujników odległości wskazują, że czujniki ultradźwiękowe zapewniają wystarczającą dokładność pomiaru dla podstawowych zadań uniwersalnego przeszkołdzenia, szczególnie w zakresie odległości od kilkunastu do kilkudziesięciu centymetrów[?]. Skuteczność tych czujników zależy jednak w dużym stopniu od właściwości powierzchni odbijającej falę akustyczną. Powierzchnie miękkie, porowate lub ustawione pod znacznym kątem względem osi pomiaru mogą powodować osłabienie lub rozproszenie sygnału odbitego, prowadząc do błędnych odczytów. Na dokładność pomiarów wpływają również warunki środowiskowe, takie jak temperatura powietrza czy obecność zakłóceń akustycznych.

Z tego względu istotnym elementem przetwarzania danych sensorycznych jest odpowiednia filtracja wyników pomiarów. W prostych systemach stosuje się najczęściej

uśrednianie kilku kolejnych odczytów lub odrzucanie wartości skrajnych, które mogą być skutkiem chwilowych zakłóceń. Zaleca się także wprowadzanie progów decyzyjnych, zwiększających odporność algorytmu sterowania na pojedyncze błędne pomiary[?]. Przykładowo, decyzja o zmianie kierunku ruchu może być podejmowana dopiero po kilkukrotnym potwierdzeniu obecności przeszkody w określonym zakresie odległości.

W bardziej rozbudowanych rozwiązaniach możliwe jest łączenie danych pochodzących z kilku czujników odległości, rozmieszczonych w różnych kierunkach wokół robota. Takie podejście pozwala na uzyskanie bardziej kompletnej informacji o otoczeniu bez konieczności stosowania zaawansowanych czujników. Integracja wielu źródeł danych sensorycznych, określana mianem fuzji sensorycznej, może istotnie zwiększyć niezawodność procesu percepcji, jednak wiąże się ze wzrostem złożoności algorytmów decyzyjnych oraz wymaga starannego doboru parametrów czasowych.

W kontekście mikrokontrolerów, takich jak Arduino UNO, przetwarzanie danych sensorycznych musi być realizowane w sposób efektywny czasowo. Pomiar odległości z wykorzystaniem czujnika ultradźwiękowego wymaga precyzyjnego odmierzania czasu trwania impulsu powrotnego, co realizowane jest z wykorzystaniem liczników sprzętowych lub mechanizmów przerwań. Odpowiednia organizacja programu, obejmująca cykliczny odczyt czujników oraz sekwencyjne podejmowanie decyzji, ma kolosalne znaczenie dla zapewnienia reakcji robota w czasie zbliżonym do rzeczywistego.

Należy również zwrócić uwagę na kompromis pomiędzy częstotliwością odczytu czujników a stabilnością działania algorytmu sterowania. Zbyt częste pomiary mogą prowadzić do nadmiernej liczby decyzji ruchowych i niestabilności toru jazdy, natomiast zbyt rzadkie odczyty zwiększą ryzyko kolizji z przeszkodami. W literaturze podkreśla się konieczność doboru parametrów czasowych w sposób dostosowany do prędkości robota oraz dynamiki środowiska pracy.

2.6. Zaawansowane metody nawigacji

Postęp w zakresie mocy obliczeniowej systemów wbudowanych oraz rosnąca dostępność zaawansowanych czujników pomiarowych przyczyniły się do dynamicznego rozwoju metod lokalizacji i nawigacji robotów mobilnych. W odróżnieniu od podejść reaktywnych, które bazują wyłącznie na bieżących pomiarach sensorycznych, metody zaawansowane zakładają istnienie wewnętrznej reprezentacji środowiska oraz ciągłą estymację położenia robota w czasie. Centralnym zagadnieniem w tym obszarze jest problem

jednoczesnej lokalizacji i mapowania, znany w literaturze jako SLAM (Simultaneous Localization and Mapping).

Algorytmy SLAM umożliwiają stopniowe tworzenie mapy nieznanego środowiska przy jednoczesnym określaniu pozycji robota względem tej mapy. Złożoność tego problemu wynika z faktu, że zarówno dane sensoryczne, jak i modele ruchu obarczone są niepewnością, której kumulacja prowadzi do narastania błędów lokalizacji. Jak podkreśla Thrun, SLAM stanowi jedno z fundamentalnych wyzwań współczesnej robotyki mobilnej, łącząc elementy probabilistyczne, estymacji stanu oraz sztucznej inteligencji [?].

W praktycznych implementacjach algorytmy SLAM wykorzystują dane pochodzące z wielu źródeł sensorycznych. Najczęściej stosowane są czujniki LiDARowe, dostarczające precyzyjnych informacji o geometrii otoczenia, kamery wizyjne umożliwiające analizę cech wizualnych środowiska oraz jednostki inercyjne, pozwalające na estymację przyspieszeń i prędkości kątowych robota. Fuzja danych z różnych czujników zwiększa dokładność lokalizacji oraz poprawia odporność systemu na zakłócenia lub chwilową utratę sygnału z jednego z kanałów pomiarowych[?].

W literaturze wyróżnia się kilka klas algorytmów SLAM, różniących się sposobem reprezentacji niepewności oraz złożonością obliczeniową. Do najwcześniej stosowanych należą metody oparte na filtrze Kalmana oraz jego rozszerzeniach, takich jak rozszerzony filtr Kalmana (EKF), które znajdują zastosowanie głównie w systemach o ograniczonej liczbie stanów. Algorytmy częsteczkowe, w tym FastSLAM, umożliwiają lepsze odwzorowanie nieliniowości oraz wielomodalności rozkładów prawdopodobieństwa, jednak wiążą się ze znacznym wzrostem zapotrzebowania na moc obliczeniową. W nowoczesnych systemach autonomicznych coraz częściej stosuje się metody grafowe, charakteryzujące się wysoką dokładnością estymacji, kosztem złożonej implementacji oraz dużych wymagań sprzętowych.

Poza lokalizacją i mapowaniem istotnym elementem zaawansowanej nawigacji jest planowanie ruchu. W systemach dysponujących mapą środowiska wykorzystuje się algorytmy planowania globalnego, których zadaniem jest wyznaczenie trajektorii pomiędzy pozycją początkową a celem. Do najczęściej stosowanych należą algorytmy grafowe, takie jak Dijkstra czy A*, a także algorytmy losowe, w tym RRT (Rapidly-exploring Random Trees). Planowanie globalne umożliwia efektywne omijanie przeszkód oraz minimalizację długości trajektorii, jednak jego skuteczność jest ściśle uzależniona od jakości mapy oraz dokładności lokalizacji robota[?].

Uzupełnieniem planowania globalnego są algorytmy planowania lokalnego, odpowiedzialne za reagowanie na dynamiczne zmiany w otoczeniu, takie jak pojawienie się przeszkód ruchomych lub nieprzewidziane zakłócenia. W praktycznych systemach autonomicznych często stosuje się architektury hybrydowe, w których plan globalny stanowi punkt odniesienia, natomiast algorytmy lokalne realizują bieżącą korektę ruchu. Takie podejście zwiększa skuteczność działania robota w złożonych środowiskach, lecz jednocześnie prowadzi do znacznego wzrostu złożoności całego systemu sterowania.

Pomimo licznych zalet, implementacja zaawansowanych metod lokalizacji i nawigacji napotyka na istotne ograniczenia. Algorytmy SLAM wymagają nie tylko znacznych zasobów obliczeniowych, lecz także precyzyjnych czujników oraz starannej kalibracji, co może utrudniać analizę działania systemu i przesyłać podstawowe cele dydaktyczne. Jak zauważa Thrun, nadmierna złożoność rozwiązań może ograniczać ich przydatność w kontekście nauczania podstaw robotyki mobilnej[?].

Z tego względu często świadomie rezygnuje się z implementacji pełnych algorytmów SLAM na rzecz uproszczonych metod nawigacji. Takie podejście umożliwia koncentrację na zagadnieniach integracji czujników, projektowania algorytmów sterowania ruchem oraz eksperymentalnej analizie zachowania robota w rzeczywistym środowisku. Jednocześnie znajomość zaawansowanych metod lokalizacji i nawigacji stanowi istotne zaplecze teoretyczne, pozwalające właściwie ocenić możliwości oraz ograniczenia prostych rozwiązań stosowanych w systemach o ograniczonych zasobach.

3. Montaż robota

Punktem kulminacyjnym realizacji całej pracy inżynierskiej jest zbudowanie fizycznej podstawy robota. Zadaniem robota jest omijanie przeszkód, na podstawie wykrywania ich w czasie rzeczywistym. Robot nie tworzy na bieżąco mapy otoczenia, jedynie reaguje na przeszkody pojawiające się w zasięgu czujnika ultradźwiękowego. Cała konstrukcja robota oparta jest na płycie akrylowej. Robot składa się z następujących komponentów:

- mikrokontroler Arduino Uno,
- sterownik silników L293D,
- silniki oraz koła,
- czujnik ultradźwiękowy HC-SR04,
- części obrotowej MicroServo9G,
- baterie litowo-jonowe 18650 3.3V,
- przełącznik kołyskowy.

3.1. Komponenty robota

W niniejszej części przedstawiono podstawowe komponenty wykorzystane do budowy robota mobilnego, które wspólnie tworzą kompletny system mechatroniczny zdolny do realizacji założonych funkcji. Dobór poszczególnych elementów został podektywany wymaganiami funkcjonalnymi projektu, dostępnością podzespołów oraz ich przydatnością w konstrukcjach edukacyjnych i prototypowych. Każdy z zastosowanych komponentów pełni ściśle określona rolę w systemie robota.

3.1.1. Mikrokontroler Arduino Uno

Mikrokontroler Arduino Uno (Rys. 3.1) stanowi centralny element elektroniczny konstrukcji robota mobilnego, odpowiadając za przetwarzanie danych sensorycznych oraz generowanie sygnałów sterujących dla elementów wykonawczych. Płytkę Arduino Uno dysponuje następującymi właściwościami technicznymi: mikrokontroler ATmega328P pracuje przy napięciu logicznym 5 V oraz częstotliwością taktowania 16 MHz, a jego pamięć obejmuje 32 KB pamięci Flash (z czego część zajmuje zainstalowany bootloader), 2 KB pamięci SRAM oraz 1 KB pamięci EEPROM. Ponadto płytka wyposażona jest

w 14 cyfrowych pinów I/O (w tym 6 mogących generować sygnały PWM) oraz 6 wejść analogowych.

Arduino Uno obsługuje popularne standardy komunikacyjne, takie jak UART, SPI oraz I²C, co umożliwia komunikację z wyświetlaczami oraz czujnikami. Programowanie odbywa się za pomocą Arduino IDE poprzez port USB, a wgrany bootloader eliminuje konieczność stosowania zewnętrznych programatorów.



Rys. 3.1. Mikrokontroler Arduino Uno

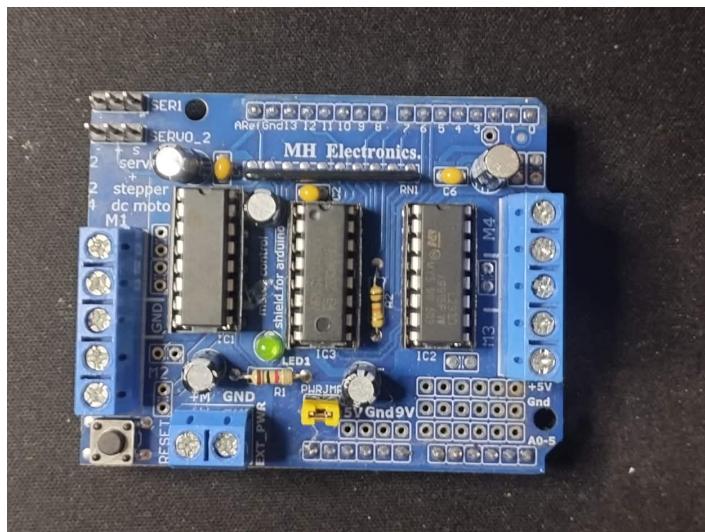
3.1.2. Sterownik silników L293D

Układ L293D (Rys. 3.2) pełni w konstrukcji robota mobilnego rolę pośredniego elementu wykonawczego pomiędzy mikrokontrolerem a silnikami prądu stałego. Jego podstawowym zadaniem jest umożliwienie sterowania kierunkiem obrotów oraz prędkością silników przy jednoczesnym odciążeniu mikrokontrolera od konieczności bezpośredniego dostarczania prądu o odpowiednich parametrach. Zastosowanie dedykowanego sterownika silników jest niezbędne ze względu na ograniczoną wydajność prądową wyjść mikrokontrolera Arduino Uno oraz konieczność separacji obwodów logicznych od obwodów mocy.

L293D jest układem scalonym typu podwójny mostek H, co oznacza, że umożliwia niezależne sterowanie dwoma silnikami prądu stałego lub jednym silnikiem krokowym. Każdy z mostków pozwala na realizację ruchu w obu kierunkach poprzez odpowiednie sterowanie sygnałami logicznymi doprowadzonymi do wejść układu. Zmiana kombinacji stanów logicznych na wejściach sterujących skutkuje zmianą polaryzacji napięcia na wyjściach, a tym samym zmianą kierunku obrotu silnika.

Układ L293D pracuje z dwoma niezależnymi napięciami zasilania. Pierwsze z nich, oznaczane jako V_{CC1} , służy do zasilania części logicznej układu i jest zazwyczaj zgodne z poziomem napięcia mikrokontrolera (5 V). Drugie napięcie, V_{CC2} , przeznaczone jest do zasilania silników i może osiągać wartość do 36 V, co umożliwia współpracę z szeroką gamą napędów. Maksymalny prąd wyjściowy pojedynczego kanału wynosi około 600 mA, co jest wystarczające dla niewielkich silników użytych w projekcie.

Istotną cechą układu L293D jest obecność wbudowanych diod zabezpieczających, które chronią układ przed przepięciami indukowanymi przez silniki podczas gwałtownych zmian prądu. Dzięki temu możliwe jest uproszczenie schematu elektrycznego oraz zwiększenie niezawodności całego systemu. Dodatkowo układ umożliwia regulację prędkości obrotowej silników poprzez zastosowanie modulacji szerokości impulsu (PWM) na wejściach sterujących, generowanej przez mikrokontroler.



Rys. 3.2. Układ L293D

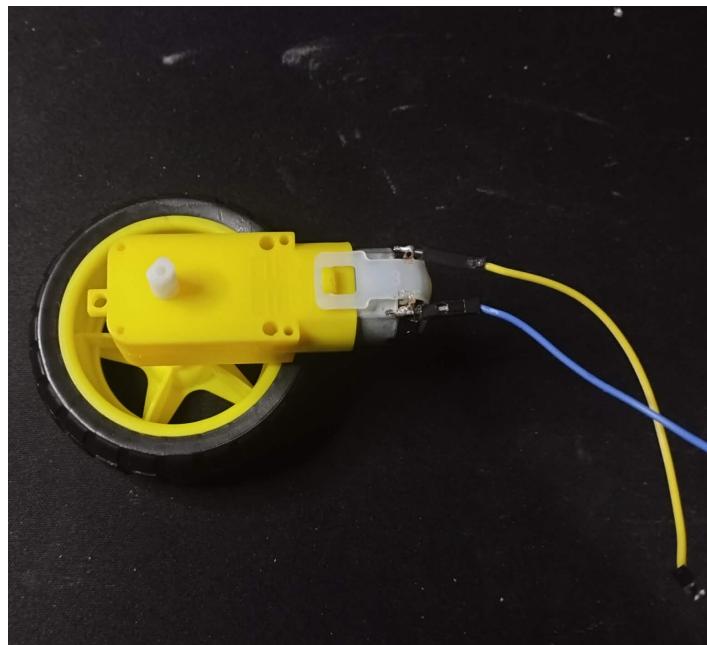
3.1.3. Silniki oraz koła

Elementy wykonawcze w postaci silników oraz kół (Rys. 3.3) stanowią podstawowy układ napędowy robota mobilnego i bezpośrednio odpowiadają za jego zdolność do poruszania się w środowisku.

Silniki prądu stałego charakteryzują się prostą zasadą działania, polegającą na wytwarzaniu momentu obrotowego w wyniku oddziaływania pola magnetycznego i prądu płynącego w uzwojeniu wirnika. Zmiana kierunku przepływu prądu powoduje zmianę kierunku obrotów silnika, co umożliwia realizację ruchu robota zarówno do

przodu, jak i do tyłu. Regulacja prędkości obrotowej silników realizowana jest poprzez zmianę wartości napięcia zasilającego lub poprzez zastosowanie modulacji szerokości impulsu (PWM).

W celu zwiększenia momentu obrotowego oraz dostosowania charakterystyki napędu do warunków ruchu robota, silniki wyposażone są w przekładnie redukcyjne. Przekładnia umożliwia zmniejszenie prędkości obrotowej przy jednoczesnym zwiększeniu momentu, co jest szczególnie istotne podczas ruszania z miejsca, pokonywania nierówności podłoża oraz manewrowania w ograniczonej przestrzeni. Zastosowanie przekładni wpływa również na poprawę precyzji sterowania ruchem robota.



Rys. 3.3. Koło oraz silnik wraz z wyprowadzeniami

3.1.4. Czujnik ultradźwiękowy HC-SR04

Czujnik ultradźwiękowy HC-SR04 (Rys. 3.4) stanowi podstawowy element systemu percepkcji robota mobilnego i odpowiada za wykrywanie przeszkód znajdujących się w jego bezpośrednim otoczeniu. Zasada działania czujnika HC-SR04 opiera się na pomiarze czasu przelotu fali ultradźwiękowej. Moduł składa się z nadajnika oraz odbiornika akustycznego, pracujących zazwyczaj z częstotliwością około 40 kHz. Po wyzwoleniu pomiaru czujnik emituje krótki impuls ultradźwiękowy, który rozchodzi się w powietrzu, a następnie odbija się od napotkanej przeszkody i powraca do odbiornika. Na podstawie czasu pomiędzy wysłaniem sygnału a jego odebraniem możliwe jest wyzna-

czenie odległości do obiektu, przy założeniu znanej prędkości rozchodzenia się dźwięku w powietrzu.

Moduł HC-SR04 wyposażony jest w cztery wyprowadzenia: zasilanie, uziemienie, wejście sygnału wyzwalającego (TRIG) oraz wyjście sygnału echa (ECHO). Pomiar realizowany jest poprzez podanie krótkiego impulsu logicznego na pin TRIG, po czym na pinie ECHO generowany jest impuls o długości proporcjonalnej do czasu powrotu fali ultradźwiękowej. Takie rozwiązanie umożliwia łatwą obsługę czujnika za pomocą mikrokontrolera, bez konieczności stosowania skomplikowanych interfejsów komunikacyjnych.

Zakres pomiarowy czujnika HC-SR04 wynosi typowo od kilku centymetrów do około czterech metrów, przy czym najwyższą dokładność uzyskuje się w przedziale od kilkunastu do kilkudziesięciu centymetrów.



Rys. 3.4. Czujnik ultradźwiękowy HC-SR04

3.1.5. Element obrotowy MicroServo 9G

Element obrotowy MicroServo 9G (Rys. 3.5) stanowi element wykonawczy robota mobilnego, wykorzystywany do realizacji ruchów obrotowych o zadanym zakresie kątowym. W prezentowanej konstrukcji pełni on funkcję mechanizmu pozycjonującego czujnik ultradźwiękowy, umożliwiając zmianę kierunku pomiaru oraz rozszerzenie obszaru percepcji otoczenia bez konieczności stosowania dodatkowych czujników.

MicroServo 9G należy do grupy niewielkich serwomechanizmów modelarskich, charakteryzujących się zową konstrukcją, niską masą oraz prostotą sterowania. Wewnątrz obudowy znajduje się silnik prądu stałego, przekładnia redukcyjna oraz układ

elektroniczny realizujący sprzężenie zwrotne na podstawie sygnału z potencjometru położenia. Dzięki temu serwomechanizm jest w stanie precyzyjnie ustawić wał wyjściowy w zadanej pozycji kątowej. Sterowanie serwomechanizmem odbywa się za pomocą sygnału PWM o stałej częstotliwości, zazwyczaj wynoszącej około 50 Hz. Informacja o pożądanym położeniu kątowym zawarta jest w szerokości impulsu sterującego, która najczęściej mieści się w zakresie od około 1 ms do 2 ms. Zmiana szerokości impulsu powoduje odpowiednią zmianę kąta obrotu wału serwomechanizmu, zazwyczaj w zakresie od 0° do 180° , w zależności od konkretnego modelu i jego konstrukcji mechanicznej. Serwomechanizm MicroServo 9G zasilany jest napięciem rzędu 4,8–6 V i charakteryzuje się stosunkowo niewielkim momentem obrotowym, wystarczającym jednak do napędu lekkich elementów, takich jak czujniki, niewielkie uchwyty czy elementy mechanizmu pomiarowego.



Rys. 3.5. Element obrotowy MicroServo 9G

3.1.6. Ogniwa litowo-jonowe 18650

Ogniwa litowo-jonowe typu 18650 (Rys. 3.6) stanowią podstawowe źródło zasilania robota mobilnego. Ze względu na korzystny stosunek pojemności do masy oraz szeroką dostępność są one powszechnie stosowane w systemach wbudowanych, urządzeniach przenośnych oraz konstrukcjach robotycznych o niewielkich i średnich wymaganiach energetycznych. Oznaczenie 18650 odnosi się do wymiarów ogniw, tj. średnicy wynoszącej około 18 mm oraz długości około 65 mm.

Pojedyncze ogniwo litowo-jonowe 18650 charakteryzuje się napięciem nominalnym wynoszącym około 3,6–3,7 V, przy napięciu maksymalnym rzędu 4,2 V oraz mi-

nimalnym napięciu rozładowania wynoszącym zazwyczaj około 2,5–3,0 V, w zależności od producenta i zastosowanego układu zabezpieczeń. Pojemność ogniw tego typu mieści się najczęściej w zakresie od 2000 do 3500 mAh.

Istotną zaletą ogniw 18650 jest możliwość dostarczania stosunkowo dużych prądów. Silniki prądu stałego wykorzystywane w robotach mobilnych generują krótkotrwałe, wysokie prądy rozruchowe, które mogą powodować spadki napięcia w przypadku zastosowania niewystarczająco wydajnego źródła zasilania. Wykorzystane ognia litowo-jonowe, w przeciwieństwie do klasycznych baterii alkalicznych, lepiej radzą sobie z takimi obciążeniami dynamicznymi.

Ze względu na specyfikę chemii litowo-jonowej niezbędne jest stosowanie układów zabezpieczających, które chronią ognia przed przeładowaniem, nadmiernym rozładowaniem, zwarciem oraz przegrzaniem. W praktyce realizowane jest to poprzez dedykowane moduły BMS (Battery Management System) lub ognia wyposażone w zintegrowane układy ochronne. Zastosowanie takich zabezpieczeń ma kluczowe znaczenie dla bezpieczeństwa użytkowania robota oraz trwałości zastosowanego źródła zasilania. W kontekście pracy robota mobilnego istotne znaczenie ma również filtracja i separacja zasilania. Wahania napięcia spowodowane pracą silników mogą negatywnie wpływać na stabilność działania mikrokontrolera oraz czujników. Z tego względu zaleca się stosowanie kondensatorów filtrujących oraz osobnych torów zasilania dla części logicznej i wykonawczej, co zwiększa niezawodność całego systemu.



Rys. 3.6. Ognia litowo-jonowe 18650

3.1.7. Przełącznik kołyskowy

Przełącznik kołyskowy (Rys. 3.6) pełni w konstrukcji robota mobilnego funkcję podstawowego elementu sterowania zasilaniem, umożliwiającego ręczne włączanie i wyłączanie całego układu. Jego zastosowanie pozwala na bezpieczne odłączanie źródła energii od pozostałych komponentów robota bez konieczności fizycznego demontażu baterii, co znacząco zwiększa komfort użytkowania oraz bezpieczeństwo eksploatacji systemu.



Rys. 3.7. Przełącznik kołyskowy

3.2. Wykorzystane materiały

Do budowy robota, oprócz elementów elektronicznych, wykorzystano również materiały konstrukcyjne o charakterze mechanicznym. Podstawę całej konstrukcji stanowi płyta akrylowa (Rys. 3.8) o wymiarach $148 \times 105 \times 3$ mm, pełniąca funkcję nośnej platformy montażowej dla pozostałych komponentów systemu. Wybór tego materiału podyktowany był jego niewielką masą, wystarczającą sztywnością, łatwością obróbki oraz brakiem własności przewodzenia prądu.

W celu zapewnienia stabilnego i powtarzalnego położenia czujnika ultradźwiękowego względem osi obrotu serwomechanizmu zastosowano stalowy kątownik montażowy (Rys. 3.9). Element ten został umocowany w odpowiedniej pozycji przy użyciu kleju termotopliwego aplikowanego za pomocą pistoletu do kleju (Rys. 3.10) oraz dodatkowo zabezpieczony opaską zaciskową. Pomimo zastosowania prostych metod montażu, konstrukcja spełnia założone wymagania funkcjonalne. Klej termotopliwy został wykorzystany również do mocowania silników napędowych, serwomechanizmu oraz czujnika

ultradźwiękowego, zapewniając wystarczającą trwałość połączeń przy niewielkim nakładzie czasowym.

Uchwyt na baterie oraz mikrokontroler Arduino Uno zostały zamocowane do płyty akrylowej za pomocą wkrętów. Połączenia elektryczne pomiędzy przewodami zasilającymi a elementami elektronicznymi wykonano przy użyciu bezołowiowej cyny lutowniczej, nanoszonej za pomocą lutownicy ręcznej (Rys. 3.11).



Rys. 3.8. Płytki akrylowe z otworami na wkręty



Rys. 3.9. Stalowy łącznik zamocowany na serwomechanizmie, kątownik zaznaczony czerwonym obramowaniem

3.3. Budowa robota

Przed przystąpieniem do realizacji właściwego projektu praktycznego zasadnym etapem procesu inżynierskiego jest wykonanie prototypu, którego celem jest weryfikacja przyjętych założeń projektowych oraz identyfikacja potencjalnych problemów konstrukcyjnych na wczesnym etapie prac.

Celem budowy robota było potwierdzenie możliwości realizacji głównego założenia projektowego, jakim było stworzenie autonomicznej konstrukcji mobilnej zdolnej



Rys. 3.10. Pistolet na klej termotopliwy wraz z wkładem



Rys. 3.11. Lutownica, cyna bezołowiowa oraz pasta lutownicza

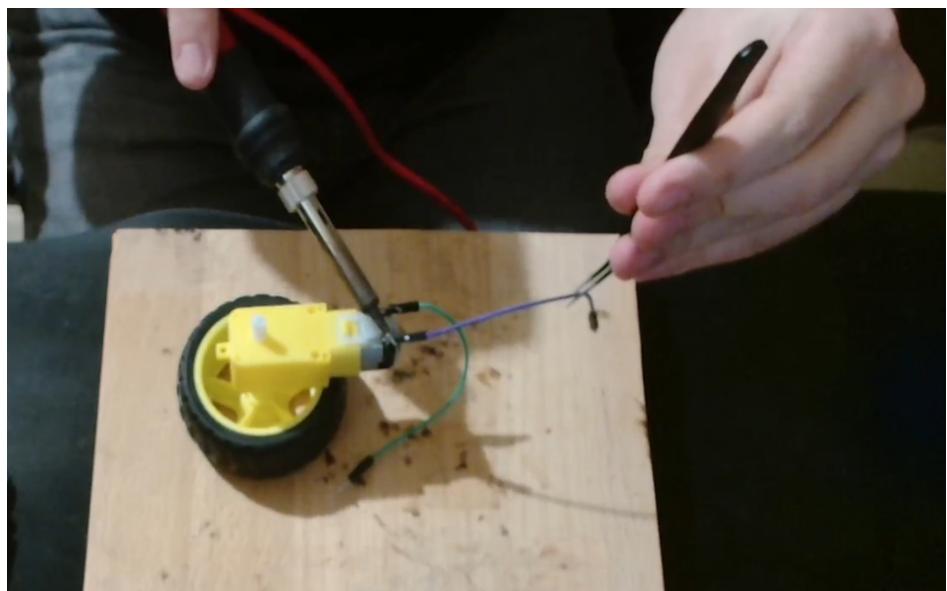
do poruszania się w środowisku oraz omijania przeszkód na podstawie danych pozyskiwanych z czujnika ultradźwiękowego.

3.3.1. Lutowanie wyprowadzeń silników

W pierwszym etapie montażu przeprowadzone zostało łączenie wyprowadzeń silników z kablami (Rys. 3.12). Wykonanie tego kroku przed połączeniem silników z podstawą akrylową jest wysoce wskazane ze względu na znacznie większą swobodę wykonywania ruchów lutownicą.

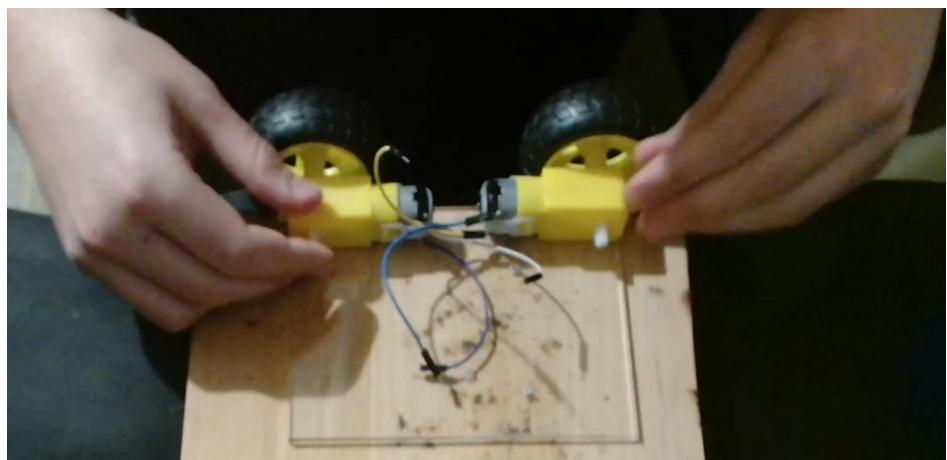
3.3.2. Montaż silników oraz kół

Silniki zostały zamocowane symetrycznie względem środka geometrycznego robota, przy rogach platformy (Rys. 3.13). Ich położenie dobrano w taki sposób, aby oś obrotu kół znajdowała się możliwie blisko tylnej krawędzi płyty nośnej, co zwiększa stabilność robota podczas jazdy oraz ułatwia wykonywanie manewrów skrętu. Do mocowania silników wykorzystano klej termotopliwy, który zapewnia wystarczającą sztywność połączenia przy jednoczesnym zachowaniu możliwości względnie łatwego demontażu w



Rys. 3.12. Łączenie wyprowadzeń silników z kablami

przypadku modyfikacji konstrukcji. Koła zostały osadzone bezpośrednio na wałach silników.

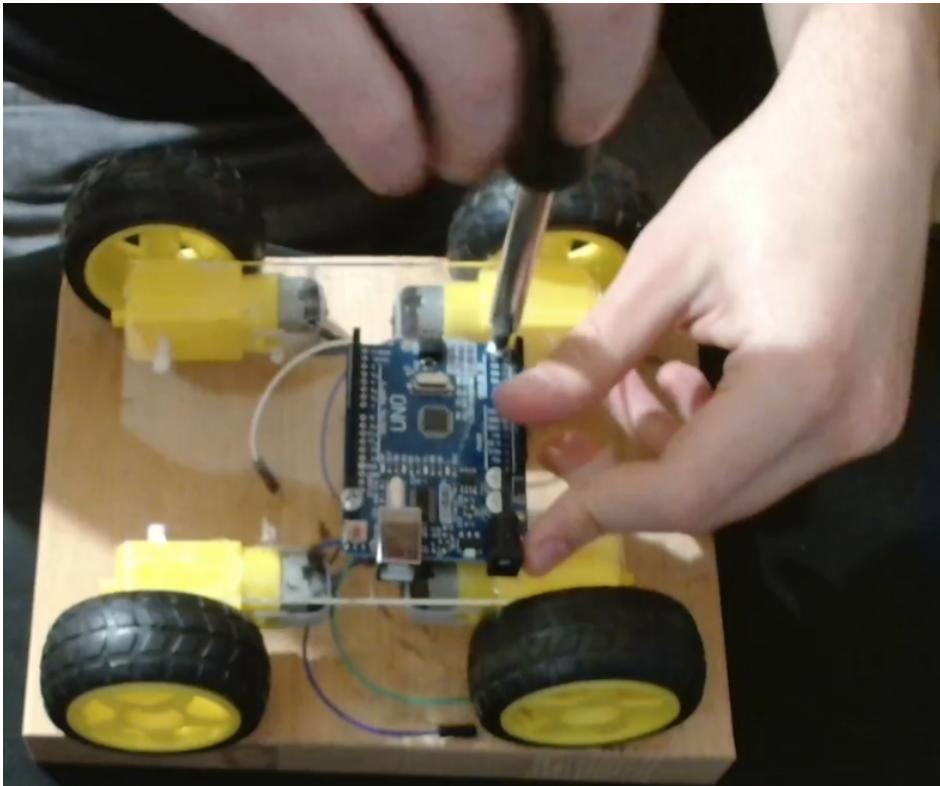


Rys. 3.13. Montaż silników z kołami

3.3.3. Montaż mikrokontrolera Arduino Uno

Płytkę Arduino Uno została zamocowana do płyty akrylowej konstrukcji nośnej za pomocą wkrętów dystansowych, co pozwoliło na jej uniesienie ponad powierzchnię podstawy (Rys. 3.14). Takie rozwiązanie minimalizuje ryzyko zwarć elektrycznych, umożliwia swobodny przepływ powietrza pod płytą oraz ułatwia prowadzenie przewodów połączeniowych. Otwory montażowe znajdujące się w narożnikach płytki mikrokontrolera zostały wykorzystane zgodnie z ich przeznaczeniem, co dodatkowo zwiększyło sztywność całego mocowania.

Orientacja mikrokontrolera została dobrana w sposób umożliwiający wygodny dostęp do portu USB oraz złącza zasilania, co znaczaco ułatwiało proces programowania oraz diagnostyki układu bez konieczności demontażu robota. Dodatkowo zachowano odpowiednie odstępy pomiędzy mikrokontrolerem a pozostałymi komponentami elektronicznymi, aby zapewnić przejrzystość połączeń.



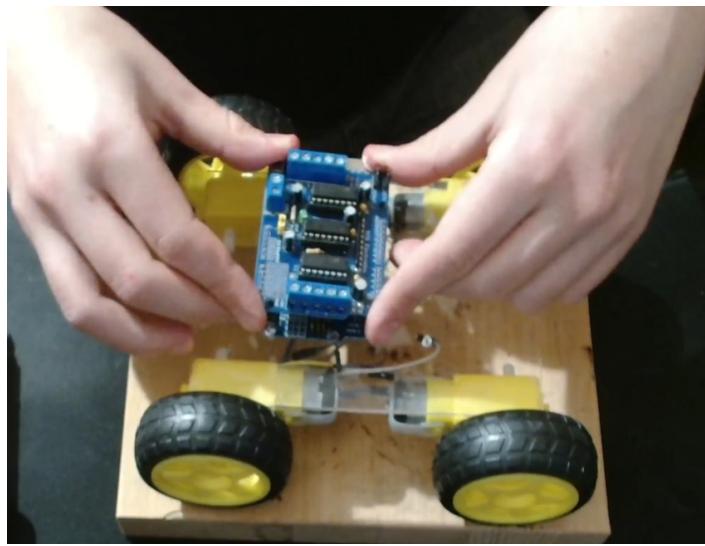
Rys. 3.14. Montaż mikrokontrolera Arduino Uno

3.3.4. Montaż osłony L293D

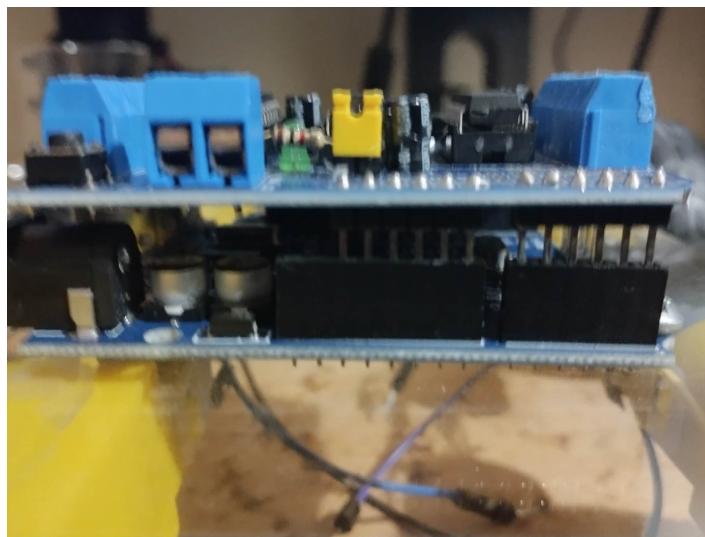
Osłona L293D została zamontowana bezpośrednio na mikrokontrolerze Arduino Uno (Rys. 3.15). Moduł ten został zaprojektowany z myślą o ścisłej współpracy z płytami Arduino, co umożliwia ich bezpośrednie połączenie bez konieczności stosowania dodatkowych przewodów połączeniowych. Rozmieszczenie wyprowadzeń osłony jednoznacznie determinuje sposób jej instalacji na płytce mikrokontrolera. Podczas montażu należy zwrócić uwagę na prawidłowe dopasowanie złączy pinowych, tak aby gniazda obu płytek pokrywały się ze sobą (Rys. 3.16).

3.3.5. Montaż uchwytu na baterie

Uchwyt na baterie został zamontowany w pobliżu mikrokontrolera Arduino Uno (Rys. 3.17), ponieważ zasilanie musi być połączone z nakładką płytki Arduino, skąd energia



Rys. 3.15. Montaż osłony L293D

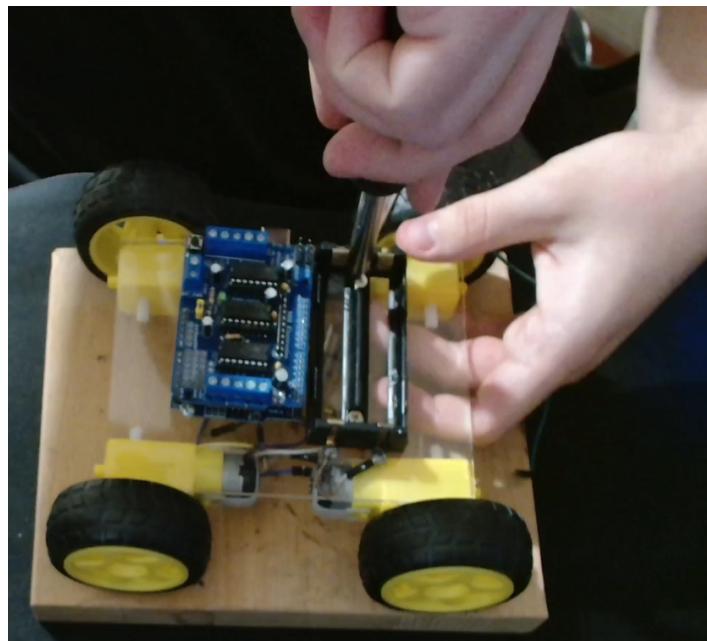


Rys. 3.16. Sposób łączenia płytki Arduino Uno z L293D

jest dystrybuowana do pozostałych komponentów. Baterie celowo zostały umieszczone nieintuicyjnie po przeciwej stronie wejść zasilania, co umożliwiło wykorzystanie kabli o domyślnej długości bez konieczności ówczesnej obróbki złączy. Zastosowanie krótszych kabli w przypadku umieszczenia zasilania po przeciwej stronie mikrokontrolera jest jak najbardziej zasadne, pod warunkiem poprawnego wykonania połączenia. Należy mieć na uwadze fakt, aby przewody nie były napięte oraz nie były poddawane nadmiernym zgięciom, gdyż może to doprowadzić do przerwania kabli.

3.3.6. Montaż ogniw zasilających

Baterie, zasilające cały układ, zostały umieszczone w uchwycie na baterie (Rys. 3.18). Czerpanie energii z baterii odbywa się za pośrednictwem metalowych blaszek zlokalizowanych na baterii i uchwycie baterii.

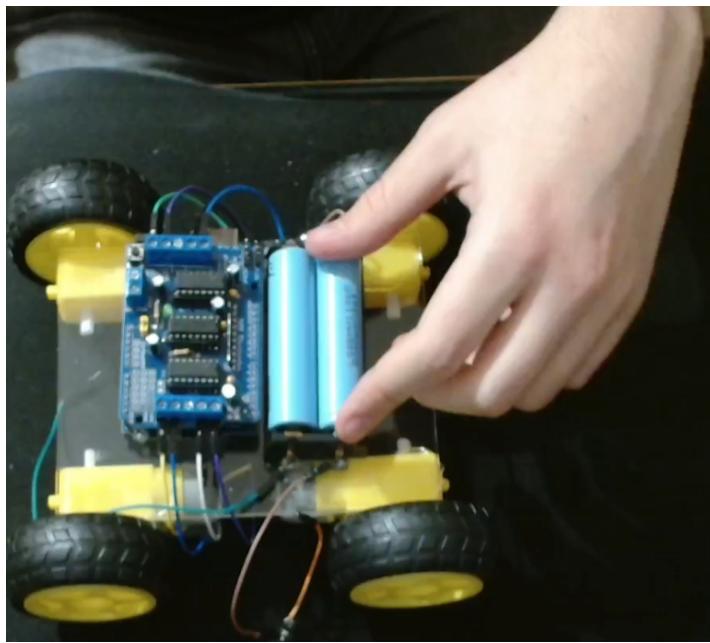


Rys. 3.17. Montaż uchwytu na baterie

zowanych na brzegach uchwytu. Technicznie, możliwe jest połączenie bezpośrednio z ogniwami, natomiast jest to silnie odradzana czynność ze względu na możliwe poprzez naruszenie struktury ognia nagłe uwolnienie energii w postaci eksplozji. Autor używa ogniw z zabezpieczeniem przepięciowym, co niweluje konsekwencje w przypadku błędego podłączenia zasilania do układu. Przy wykorzystaniu niezabezpieczonych ogniw, niepoprawne połączenie może skutkować zwarciem. Użyty uchwyt na baterie posiada specjalne oznaczenia, odnoszące się do prawidłowego rozmieszczenia polaryzacji baterii (Rys. 3.19).

3.3.7. Montaż serwomechanizmu z kątownikiem

Serwomechanizm z kątownikiem umieszczony został w części, przyjętej przez autora we wstępny projekcie jako "przednią część robota" (Rys. 3.18). To stwierdzenie jest czysto względne, ponieważ przód mógłby się znajdować po przeciwej stronie robota. Ważnym założeniem jest, aby serwomechanizm znajdował się w osi poruszania robota oraz był domyślnie zwrócony ku kierunkowi poruszania. W takim przypadku, do poprawnego działania robota koniecznym byłoby odwrócenie polaryzacji silników, tzn. odwrotne połączenie z płytą L293D lub zmiany w kodzie programu. Serwomechanizm przyklejony został do podstawy akrylowej warstwą kleju termotopliwego.



Rys. 3.18. Montaż baterii

3.3.8. Montaż czujnika ultradźwiękowego odległości

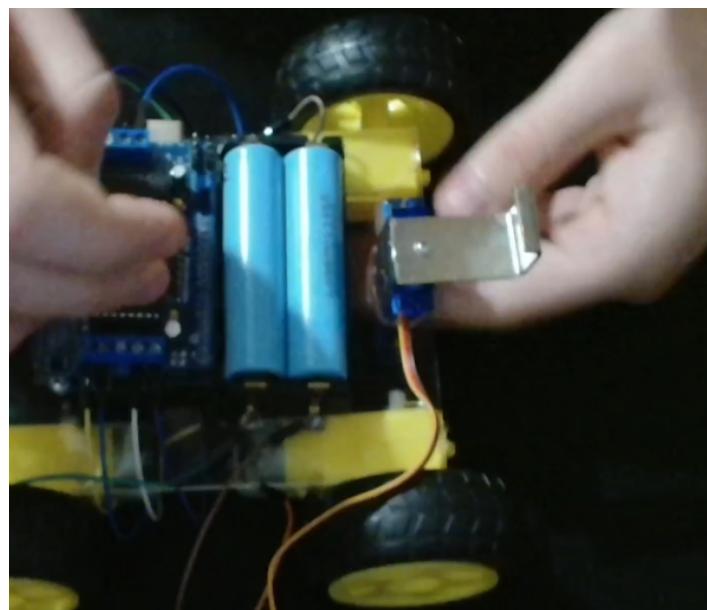
Czujnik ultradźwiękowy odległości HC-SR04 umieszczony został na przygotowanym kątowniku przykręconym do serwomechanizmu (Rys. 3.21). Czujnik został przyklejony klejem termotopliwym do konstrukcji w miejscułączenia kabli, ze względu na dużą płaską powierzchnię. Używanie kleju bezpośrednio na czujniku nie jest wskazane. Spособami montażu czujnika wartymi wspomnienia są również użycie trwalszego kleju, wprowadzenie innej formy podstawy dla czujnika lub wykorzystanie opaski uciskowej.

3.3.9. Schemat połączeń robota

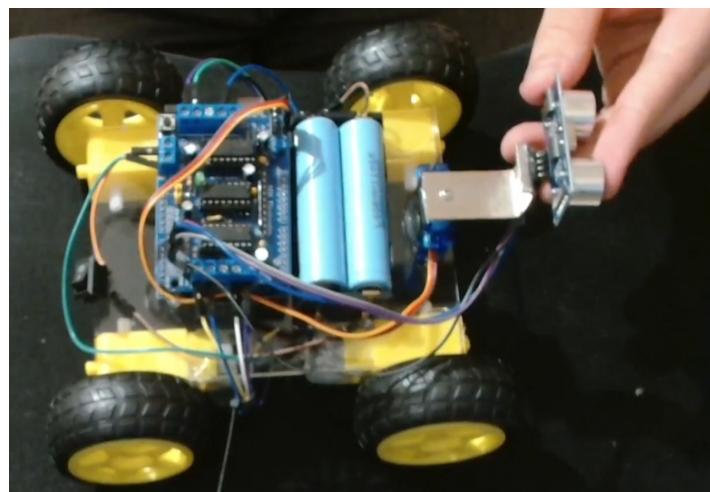
Schemat połączeń robota (Rys. 3.22) ilustruje wszystkie połączenia wykonane w projekcie. Kolorem czerwonym oznaczone zostały połączenia o dodatnim potencjale, czarnym zaś przewody uziemiające. Ogniwa podłączone zostały poprzez przełącznik kołyskowy do zasilania układu, skąd z portów M1, M2, M3, M4 wychodzą wyprowadzenia silników. Serwomechanizm został wpięty do dedykowanego pinu SER1. Czujnik ultradźwiękowy odległości wpięto do zasilania oraz uziemienia, przy czym ten czujnik posiada dwa dodatkowe wyprowadzenia. Wyzwalacz oznaczony symbolem TRIG został przypisany portowi funkcyjnemu A0, natomiast nasłuch oznaczony jako ECHO do portu A1.



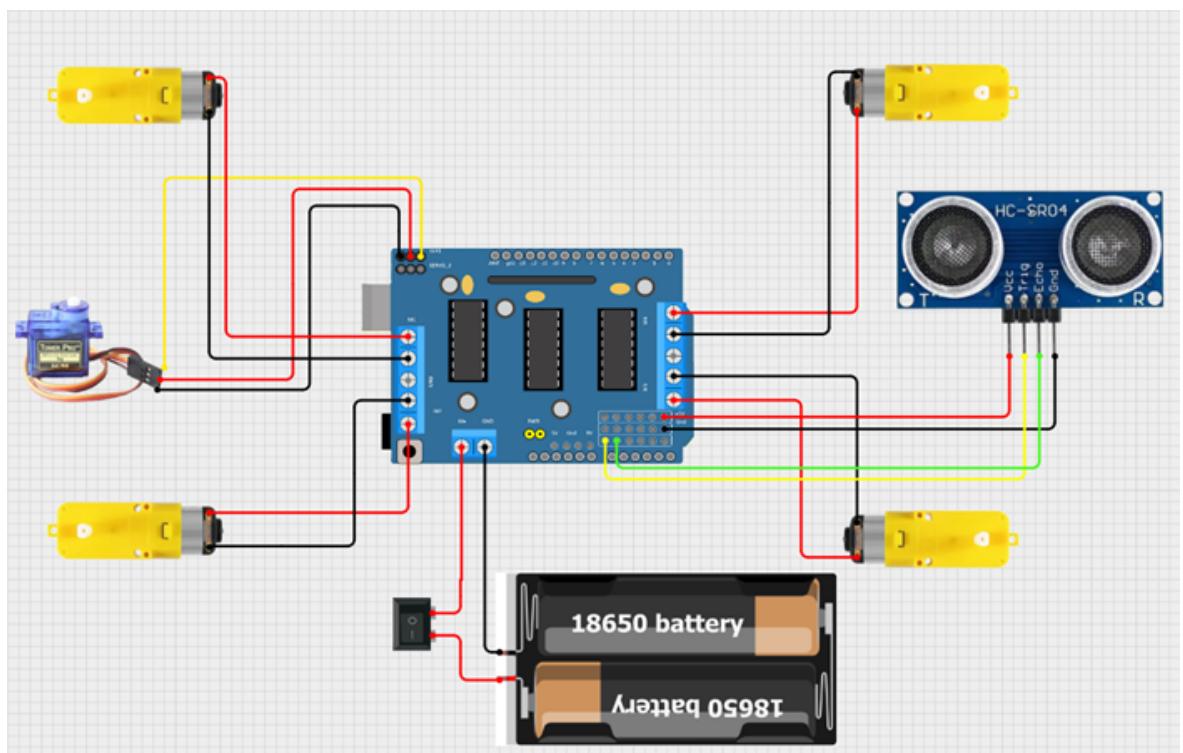
Rys. 3.19. Oznaczenie biegunów na uchwycie (czerwone symbole dorysowane nad faktycznymi symbolami)



Rys. 3.20. Montaż serwomechanizmu



Rys. 3.21. Montaż czujnika ultradźwiękowego odległości HC-SR04



Rys. 3.22. Schemat ideowy robota

4. Kod programu

Integralnym elementem realizacji projektu robota mobilnego jest warstwa programowa, która odpowiada za interpretację danych sensorycznych, podejmowanie decyzji sterujących oraz bezpośrednie zarządzanie elementami wykonawczymi. Kod programu stanowi zatem kluczowy komponent systemu, determinujący stopień autonomii robota, jego reakcję na zmienne warunki otoczenia oraz stabilność działania całej konstrukcji.

Cały kod znajduje się w załączonych plikach: Robot_arduino_kod.ino oraz Robot_arduino_kod.txt. Kod programu został zaprojektowany w sposób modularny i czytelny, co ułatwia zarówno jego analizę, jak i dalszą rozbudowę. Strukturę programu można podzielić na kilka logicznych bloków funkcjonalnych, z których każdy odpowiada za inny aspekt działania robota mobilnego.

4.1. Opis fragmentów kodu

Opracowane oprogramowanie zostało zaimplementowane na mikrokontrolerze Arduino Uno i napisane w języku C/C++ z wykorzystaniem środowiska Arduino IDE. Program realizuje podstawowy algorytm nawigacji reaktywnej, w którym decyzje ruchowe podejmowane są na podstawie bieżących pomiarów odległości uzyskiwanych z czujnika ultradźwiękowego. Robot nie tworzy mapy otoczenia ani nie zapamiętuje wcześniejszych stanów środowiska, lecz reaguje w czasie rzeczywistym na pojawiające się przeszkody, co jest zgodne z założeniami projektowymi oraz charakterem konstrukcji prototypowej.

4.1.1. Inicjalizacja komponentów

Początkowy etap programu obejmuje inicjalizację podstawowych elementów sprzętowych robota, niezbędnych do prawidłowego działania algorytmu sterowania. W tym celu wykorzystano bibliotekę AFMotor, która umożliwia sterowanie silnikami prądu stałego za pośrednictwem osłony L293D, oraz bibliotekę Servo, służącą do obsługi serwomechanizmu odpowiedzialnego za zmianę położenia czujnika ultradźwiękowego. Zastosowanie gotowych bibliotek upraszcza implementację oraz zwiększa czytelność kodu. W dalszej części zdefiniowane zostały cztery silniki napędowe, przypisane do odpowiednich kanałów osłony oraz częstotliwości sygnału PWM, co pozwala na niezależne sterowanie każdym z nich i realizację manewrów ruchu robota. Następnie określono wyprowadzenia mikrokontrolera Arduino Uno, do których podłączony jest czujnik ultradźwiękowy HC-SR04, umożliwiający pomiar odległości do przeszkód. Ostatnim kro-

kieminicjalizacji jest konfiguracja serwomechanizmu, który pozwala na skanowanie otoczenia w różnych kierunkach. Przedstawione definicje stanowią podstawę poprawnej komunikacji programu z warstwą sprzętową robota (listing 4.1).

Listing 4.1. Definicje sprzętowe i inicjalizacja komponentów

```
1 #include <AFMotor.h>
2 #include <Servo.h>
3
4 // --- Definicje silników ---
5 AF_DCMotor motor1(1, MOTOR12_1KHZ);
6 AF_DCMotor motor2(2, MOTOR12_1KHZ);
7 AF_DCMotor motor3(3, MOTOR34_1KHZ);
8 AF_DCMotor motor4(4, MOTOR34_1KHZ);
9
10 // --- Czujnik ultradźwiękowy ---
11 #define TRIG_PIN A0
12 #define ECHO_PIN A1
13
14 // --- Serwo ---
15 #define SERVO_PIN 10
16 Servo myServo;
```

4.1.2. Funkcja pomiaru odległości

Kolejną grupę elementów programu stanowią stałe konfiguracyjne oraz funkcje pomocnicze, które w istotny sposób wpływają na zachowanie robota w trakcie pracy (listing 4.2). Stałe definiują kluczowe parametry algorytmu sterowania, takie jak minimalna odległość uznawana za bezpieczną, prędkość obrotowa silników napędowych oraz zakres wychyleń serwomechanizmu odpowiedzialnego za zmianę kierunku pomiaru. Dzięki ich zastosowaniu możliwa jest łatwa modyfikacja charakterystyki ruchu robota bez konieczności ingerowania w główną logikę programu.

Szczególnie ważną rolę w tej części kodu pełni funkcja `readDistance()`, która realizuje pomiar odległości przy użyciu czujnika ultradźwiękowego HC-SR04. Funkcja ta generuje impuls wyzwalający na pinie `TRIG`, a następnie mierzy czas trwania sygnału powrotnego na pinie `ECHO`, co pozwala na obliczenie dystansu do przeszkody. W celu ograniczenia wpływu zakłóceń oraz przypadkowych błędów pomiarowych zastosowano prostą metodę filtracji polegającą na uśrednianiu kilku kolejnych odczytów oraz odrzucaniu wartości spoza fizycznie dopuszczalnego zakresu pracy czujnika. Takie rozwiązanie zwiększa stabilność i powtarzalność pomiarów, jednocześnie nie obciążając nadmiernie mikrokontrolera.

Listing 4.2. Funkcja pomiaru odległości czujnikiem ultradźwiękowym

```
1 const int DISTANCE_THRESHOLD = 20;
2 const int SERVO_CENTER = 90;
3 const int SCAN_ANGLE = 45;
4 const int MOTOR_SPEED = 170;
```

```

5 long readDistance() {
6     const int samples = 2;
7     long total = 0;
8     int validSamples = 0;
9
10    for (int i = 0; i < samples; i++) {
11        digitalWrite(TRIG_PIN, LOW);
12        delayMicroseconds(3);
13        digitalWrite(TRIG_PIN, HIGH);
14        delayMicroseconds(10);
15        digitalWrite(TRIG_PIN, LOW);
16
17        long duration = pulseIn(ECHO_PIN, HIGH, 30000);
18        if (duration <= 0) continue;
19
20        long distance = duration * 0.0343 / 2;
21        if (distance >= 2 && distance <= 400) {
22            total += distance;
23            validSamples++;
24        }
25    }
26
27    if (validSamples == 0) return -1;
28    return total / validSamples;
29}
30

```

4.1.3. Funkcje ruchu robota

Następnym etapem implementacji programu jest zdefiniowanie zestawu funkcji odpowiedzialnych za bezpośrednie sterowanie napędem robota. Funkcje te enkapsulują podstawowe operacje ruchu, takie jak jazda do przodu, cofanie oraz zatrzymanie pojazdu, i stanowią warstwę pośrednią pomiędzy algorytmem decyzyjnym a sprzętową realizacją napędu. Każda z funkcji przyjmuje jako parametr wartość prędkości, co pozwala na dynamiczne dostosowanie zachowania robota do aktualnej sytuacji w otoczeniu.

W przedstawionych procedurach sterowanie odbywa się poprzez jednoczesne ustawienie prędkości wszystkich silników oraz nadanie im odpowiedniego kierunku obrotu. Zapewnia to synchroniczną pracę napędu i stabilny ruch robota po linii prostej. Funkcja zatrzymania realizowana jest poprzez przejście silników w stan zwolnienia (RELEASE), co skutkuje ich natychmiastowym wyłączeniem bez wymuszania kierunku obrotu. Takie rozwiązanie zwiększa bezpieczeństwo pracy układu oraz ogranicza niepotrzebne zużycie energii.

Wyodrębnienie funkcji sterujących napędem do osobnych procedur znaczaco poprawia czytelność i modularność kodu. Umożliwia to również łatwą rozbudowę programu, na przykład o bardziej zaawansowane profile prędkości lub dodatkowe manewry, bez konieczności modyfikowania głównej pętli sterującej robota. Implementację podstawowych funkcji ruchu przedstawiono w listingu 4.3.

Listing 4.3. Funkcje sterowania napędem robota

```
1 void forward(int speed) {
2     motor1.setSpeed(speed); motor2.setSpeed(speed);
3     motor3.setSpeed(speed); motor4.setSpeed(speed);
4     motor1.run(FORWARD); motor2.run(FORWARD);
5     motor3.run(FORWARD); motor4.run(FORWARD);
6 }
7
8 void backward(int speed) {
9     motor1.setSpeed(speed); motor2.setSpeed(speed);
10    motor3.setSpeed(speed); motor4.setSpeed(speed);
11    motor1.run(BACKWARD); motor2.run(BACKWARD);
12    motor3.run(BACKWARD); motor4.run(BACKWARD);
13 }
14
15 void stopMotors() {
16     motor1.run(RELEASE); motor2.run(RELEASE);
17     motor3.run(RELEASE); motor4.run(RELEASE);
18 }
```

Uzupełnieniem podstawowych funkcji ruchu są procedury odpowiedzialne za zmianę kierunku jazdy robota, które wykorzystują zasadę różnicowego sterowania napędem. Wykonanie skrętu realizowane jest poprzez jednocześnie nadanie jednakowej prędkości wszystkim silnikom oraz wymuszenie przeciwnych kierunków ich obrotu po obu stronach konstrukcji. Takie sterowanie powoduje obrót robota wokół własnej osi, co jest rozwiązaniem powszechnie stosowanym w mobilnych platformach kołowych pozbawionych mechanicznego układu skrętnego.

Zachowanie symetrii prędkości silników wpływa na powtarzalność i stabilność manewru, natomiast rozdzielenie kierunków obrotu pomiędzy lewą i prawą stroną napędu jednoznacznie determinuje kierunek skrętu. Czas trwania manewru ograniczony jest poprzez wprowadzenie stałego opóźnienia, które pełni rolę prostego regulatora kąta obrotu. Pozwala to uniknąć nadmiernego skrętu oraz zapewnia płynne przejście do kolejnej fazy ruchu.

Przyjęte rozwiązanie charakteryzuje się niewielką złożonością implementacyjną i obliczeniową, a jednocześnie spełnia wymagania funkcjonalne stawiane przed robotem poruszającym się w środowisku o umiarkowanej zmienności. Implementację procedur odpowiedzialnych za skręt robota przedstawiono w listingu 4.4.

Listing 4.4. Funkcje realizujące manewr skrętu robota

```
1 void turnLeft(int speed) {
2     motor1.setSpeed(speed); motor2.setSpeed(speed);
3     motor3.setSpeed(speed); motor4.setSpeed(speed);
4     motor1.run(BACKWARD); motor2.run(BACKWARD);
5     motor3.run(FORWARD); motor4.run(FORWARD);
6     delay(150);
7 }
8
9 void turnRight(int speed) {
10    motor1.setSpeed(speed); motor2.setSpeed(speed);
```

```

11 motor3.setSpeed(speed); motor4.setSpeed(speed);
12 motor1.run(FORWARD); motor2.run(FORWARD);
13 motor3.run(BACKWARD); motor4.run(BACKWARD);
14 delay(150);
15 }

```

4.1.4. Pętla główna programu

Zasadniczą częścią oprogramowania jest pętla główna `loop()`, w której skupiona została cała logika decyzyjna odpowiedzialna za autonomiczne zachowanie robota. Każde jej wykonanie rozpoczyna się od odczytu aktualnej odległości przeszkody znajdującej się przed pojazdem, po czym uzyskana wartość porównywana jest z wcześniej zdefiniowanym progiem decyzyjnym. Jeżeli w polu detekcji nie zostanie wykryty obiekt, robot kontynuuje jazdę na wprost z ustaloną prędkością.

W sytuacji wykrycia przeszkody w odległości mniejszej od wartości granicznej algorytm przechodzi w tryb reaktywny. Robot zatrzymuje się, wykonuje krótki manewr cofania w celu zwiększenia przestrzeni manewrowej, a następnie dokonuje analizy otoczenia po obu stronach. Skanowanie realizowane jest poprzez wychylenie czujnika ultradźwiękowego w lewo oraz w prawo przy użyciu serwomechanizmu, co pozwala na ocenę dostępnej przestrzeni w każdym z kierunków. Porównanie uzyskanych pomiarów stanowi podstawę do wyboru kierunku skrętu, w którym robot dysponuje większą swobodą ruchu. Implementację pętli głównej programu przedstawiono w listingu 4.5.

Listing 4.5. Pętla główna programu i algorytm omijania przeszkód

```

1 void loop() {
2     int distCenter = readDistance();
3
4     if (distCenter < DISTANCE_THRESHOLD) {
5         stopMotors();
6         backward(140);
7         delay(250);
8         stopMotors();
9
10        myServo.write(SERVO_CENTER + SCAN_ANGLE);
11        delay(500);
12        int distLeft = readDistance();
13
14        myServo.write(SERVO_CENTER - SCAN_ANGLE);
15        delay(500);
16        int distRight = readDistance();
17
18        myServo.write(SERVO_CENTER);
19
20        if (distLeft > distRight) {
21            turnLeft(MOTOR_SPEED);
22        } else {
23            turnRight(MOTOR_SPEED);
24        }
25    } else {
26        forward(MOTOR_SPEED);
27    }
28 }

```

W trakcie tworzenia oprogramowania napotkano szereg problemów typowych dla systemów wbudowanych. Jednym z nich była niestabilność pomiarów czujnika ultradźwiękowego, objawiająca się sporadycznymi błędnymi odczytami lub brakiem odpowiedzi. Problem ten został częściowo rozwiązyany poprzez ograniczenie liczby próbek pomiarowych, wprowadzenie filtracji wartości skrajnych oraz zastosowanie limitu czasowego funkcji pulseIn. Kolejnym wyzwaniem było dobranie odpowiednich opóźnień czasowych, które z jednej strony zapewniały poprawne działanie serwomechanizmu i silników oraz pozwalały jednoznacznie określić punkt w kodzie, w którym robot obecnie się znajduje, a z drugiej nie powodowały nadmiernego spowolnienia reakcji robota.

Przyjęte rozwiązania nie są jedynymi możliwymi. Alternatywnie możliwe byłoby zastosowanie bardziej zaawansowanych algorytmów sterowania, takich jak regulatory rozmyte, algorytmy behawioralne o większej liczbie stanów czy też wykorzystanie enkoderów do precyzyjnej kontroli ruchu. Inną możliwością byłoby również zastosowanie technik lokalizacji i mapowania (SLAM), jednak wiązałoby się to ze znacznym wzrostem złożoności systemu oraz przekroczeniem możliwości sprzętowych zastosowanego mikrokontrolera. Z tego względu świadomie ograniczono zakres funkcjonalny programu, koncentrując się na czytelności kodu oraz niezawodności działania.

5. Testy robota w środowisku

Proces projektowania robota mobilnego nie kończy się na etapie implementacji sprzętowej oraz programowej, lecz wymaga również szczegółowej weryfikacji działania systemu w warunkach zbliżonych do rzeczywistego użytkowania. Testy środowiskowe robota stanowią o skuteczności zastosowanych algorytmów sterowania i percepji otoczenia. Ich celem jest sprawdzenie, w jakim stopniu robot jest w stanie funkcjonować autonomicznie w zróżnicowanych warunkach oraz reagować na sytuacje, które nie zawsze mogą zostać jednoznacznie przewidziane na etapie projektowania.

Środowisko pracy robota mobilnego charakteryzuje się dużą zmiennością, zarówno pod względem właściwości podłoża, jak i rodzaju oraz rozmieszczenia napotykanych przeszkód. Nawet w pozornie prostych warunkach domowych lub laboratoryjnych mogą występować czynniki wpływające na stabilność ruchu, dokładność pomiarów czujników czy skuteczność algorytmu omijania przeszkód. Z tego względu istotne jest przeprowadzenie serii testów, które pozwolą ocenić zachowanie robota w sytuacjach typowych, a także w warunkach granicznych, ujawniających ograniczenia zastosowanych rozwiązań.

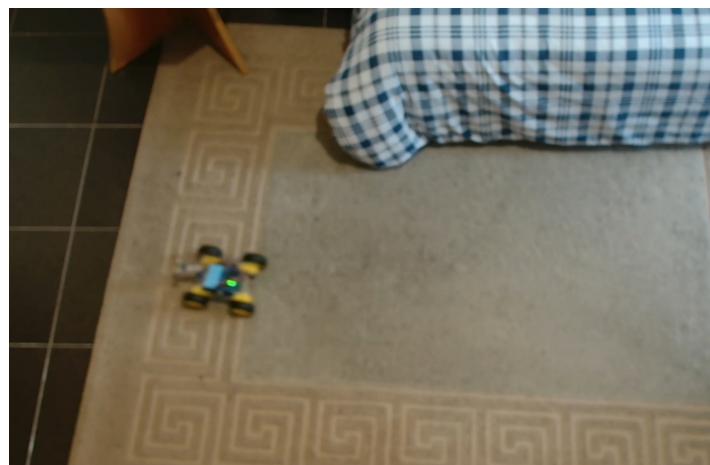
Szczególną uwagę podczas badań poświęcono współpracy pomiędzy układem napędowym a systemem percepji otoczenia. Stabilność jazdy, zdolność do pokonywania drobnych nierówności oraz reakcja na zmiany oporu ruchu mają bezpośredni wpływ na skuteczność realizowanego algorytmu sterowania. Równocześnie istotna jest analiza działania czujnika ultradźwiękowego, którego charakterystyka pomiarowa zależy od kształtu, orientacji oraz właściwości materiałowych napotykanych obiektów. Testy pozwalają zweryfikować, w jakim stopniu uproszczony model percepji otoczenia jest wystarczający do bezpiecznego i przewidywalnego poruszania się robota. Przeprowadzone próby umożliwiają również ocenę odporności systemu na sytuacje niejednoznaczne decyzyjnie, w których robot może otrzymywać sprzeczne lub niepełne informacje z czujników. W takich przypadkach znamienne znaczenie ma sposób, w jaki algorytm reaguje na brak jednoznacznej drogi przejazdu oraz czy zachowanie robota pozostaje stabilne i powtarzalne. Analiza tych scenariuszy pozwala wskazać potencjalne obszary wymagające dalszej optymalizacji, zarówno na poziomie oprogramowania, jak i konstrukcji mechanicznej.

5.1. Jazda po prostej linii

Celem przeprowadzenia testu jazdy w prostej linii była weryfikacja poprawności działania układu napędowego robota oraz sprawdzenie, czy zastosowany algorytm sterowania umożliwia stabilne poruszanie się bez konieczności wykonywania korekt kierunku (Rys. 5.1). Próba ta stanowi podstawowy test funkcjonalny, pozwalający ocenić symetrię pracy silników, równomierność przeniesienia napędu na koła oraz poprawność konfiguracji parametrów prędkości obrotowej.

Oczekiwany rezultatem testu było poruszanie się robota po możliwie prostym torze, bez wyraźnych odchyleń w lewo lub w prawo, przy zachowaniu stałej prędkości jazdy. Zakładano również brak niepożądanych drgań konstrukcji oraz płynne rozpędzanie i hamowanie, wynikające z jednoczesnego sterowania wszystkimi silnikami napędowymi. Taki rezultat świadczyłby o poprawnym doborze parametrów sterowania oraz właściwym zamocowaniu elementów mechanicznych.

W trakcie realizacji testu robot poruszał się po prostym torze ruchu, zgodnie z przyjętymi założeniami. Nie zaobserwowano istotnych odchyleń od kierunku jazdy ani tendencji do skręcania, co potwierdza poprawną synchronizację pracy silników oraz prawidłowe rozłożenie masy konstrukcji. Uzyskany wynik pozwala uznać, że układ napędowy robota działa stabilnie i stanowi solidną podstawę do realizacji bardziej złożonych manewrów w kolejnych testach środowiskowych.



Rys. 5.1. Jazda po prostej linii

5.2. Pokonywanie nierówności podłożą

Celem niniejszego testu była ocena zdolności robota do poruszania się po podłożu o zmiennej charakterystyce oraz sprawdzenie, w jaki sposób konstrukcja mechaniczna i układ napędowy reagują na występowanie niewielkich nierówności terenowych (Rys. 5.2). Test ten pozwala zweryfikować nie tylko przyczepność kół do nawierzchni, lecz również wpływ zmiany oporów toczenia na stabilność jazdy oraz ciągłość realizacji algorytmu sterowania.

Oczekiwany rezultatem było zachowanie płynności ruchu robota podczas przejazdu pomiędzy powierzchniami o różnej strukturze, bez utraty trakcji, zatrzymania lub niekontrolowanej zmiany kierunku jazdy. Zakładano, że zastosowany układ napędowy, przy odpowiednio dobranej prędkości silników, umożliwi pokonanie niewielkiej przeszkody terenowej wynikającej z różnic wysokości oraz zwiększonego oporu podłożu, bez konieczności ingerencji w algorytm sterujący.

W trakcie testu robot poprawnie pokonał nierówność terenową, polegającą na wjeździe z twardej, gładkiej powierzchni podłogi na miękkie i bardziej oporowe podłożo w postaci dywanu. Manewr został zrealizowany płynnie, bez zatrzymania napędu ani destabilizacji konstrukcji. Uzyskany wynik potwierdza wystarczającą moc układu napędowego oraz odpowiednią przyczepność kół, co umożliwia eksploatację robota w środowiskach o umiarkowanie zróżnicowanej nawierzchni.



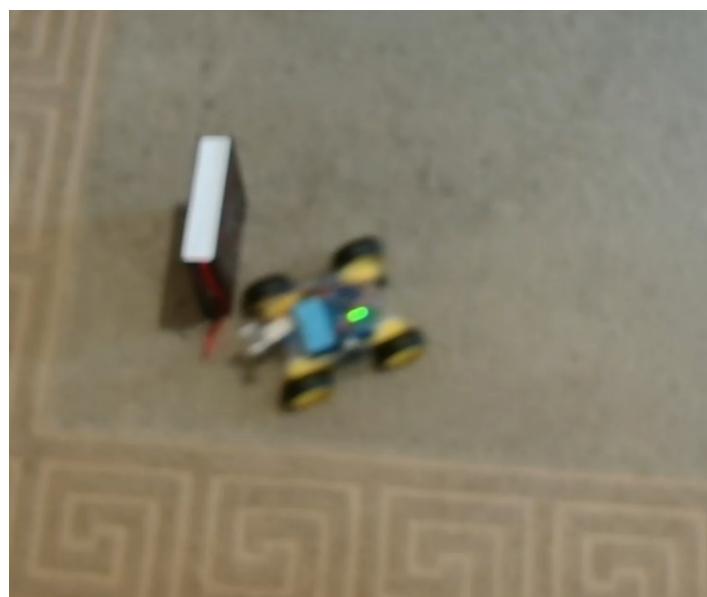
Rys. 5.2. Pokonywanie nierówności podłożą

5.3. Omijanie przeszkód w polu wykrywania czujnika

Celem tego testu była weryfikacja skuteczności algorytmu omijania przeszkód w sytuacji, gdy obiekt znajduje się w bezpośrednim polu widzenia czujnika ultradźwiękowego robota. Sprawdzano poprawność detekcji przeszkody, adekwatność reakcji decyzyjnej oraz wpływ przyjętego manewru skrętu na dalszy tor jazdy robota. Istotnym aspektem testu była również ocena powtarzalności zachowania robota przy identycznych parametrach sterowania, lecz nieznacznie zmienionych warunkach początkowych.

Oczekiwany efektem było jednoznaczne wykrycie przeszkody po przekroczeniu zdefiniowanego progu odległości, zatrzymanie ruchu postępowego oraz wykonanie manewru skrętu umożliwiającego bezkolizyjne jej ominięcie. Zakładano, że obrót robota wokół własnej osi oraz wybór kierunku skrętu na podstawie skanowania otoczenia pozwolą na wyznaczenie nowego, bezpiecznego kierunku jazdy i kontynuację ruchu bez kontaktu z przeszkodą.

W trakcie pierwszej próby robot prawidłowo zidentyfikował przeszkodę i rozpoczął procedurę omijania. Wykonany manewr skrętu spowodował jednak takie ustawienie robota, w którym przeszkoda znalazła się poza aktualnym polem detekcji czujnika ultradźwiękowego, mimo że wciąż znajdowała się na torze jazdy (Rys. 5.3). W konsekwencji w kolejnym cyklu algorytmu przeszkoda nie została ponownie wykryta, co doprowadziło do fizycznego kontaktu robota z obiektem.



Rys. 5.3. Omijanie przeszkód w polu wykrywania czujnika, przeszkoda została zahaczona

Test powtórzono przy nieznacznie zmienionej pozycji startowej robota. W drugim przypadku, pomimo zastosowania tego samego promienia skrętu oraz identycznych parametrów sterowania, robot zdołał poprawnie ominąć przeszkodę i kontynuować jazdę (Rys. 5.4). Uzyskane wyniki wskazują na ograniczenia wynikające z wąskiego pola widzenia pojedynczego czujnika ultradźwiękowego. Jednocześnie potwierdzają one poprawność działania logiki decyzyjnej w sprzyjających warunkach geometrycznych.



Rys. 5.4. Omijanie przeszkód w polu wykrywania czujnika, przeszkoda została ominięta

5.4. Omijanie przeszkód poza polem wykrywania czujnika

Celem niniejszego testu była ocena zachowania robota w sytuacji, gdy przeszkoda nie znajduje się w polu detekcji czujnika ultradźwiękowego, a zatem nie jest uwzględniana przez algorytm decyzyjny odpowiedzialny za omijanie obiektów. Badanie to miało na celu określenie wpływu takich przeszkód na stabilność ruchu robota oraz przewidywalność jego trajektorii w warunkach braku informacji sensorycznej.

Oczekiwany efektem testu była utrata zdolności świadomego reagowania na przeszkodę, przy jednoczesnym zachowaniu możliwie stabilnego toru jazdy. Zakładano, że ewentualny kontakt robota z obiektem spowoduje jedynie krótkotrwałe zakłócenie ruchu, bez istotnego wpływu na dalszy kierunek poruszania się. Taki scenariusz mógłby świadczyć o odpowiedniej sztywności konstrukcji oraz symetrycznym rozkładzie sił działających na układ jezdny.

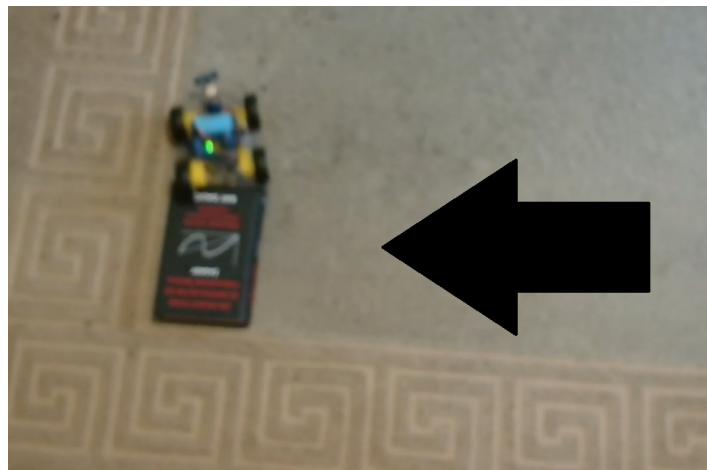
W ramach testu przeprowadzono dwie próby z wykorzystaniem przeszkód o różnej grubości. W pierwszej próbie zastosowano cienką książkę, która po kontakcie z robotem spowodowała jedynie niewielką zmianę trajektorii ruchu (Rys. ??). Odchylenie to miało charakter krótkotrwały i nie doprowadziło do znaczącej dezorientacji robota, jednak jego wartość była trudna do jednoznacznego przewidzenia.



Rys. 5.5. Omijanie przeszkód poza polem wykrywania czujnika, przeszkoda została ominięta, strzałka pokazuje pierwotny kierunek poruszania robota

W drugiej próbie wykorzystano książkę o znacznie większej grubości, co skutkowało gwałtowną i wyraźną zmianą kierunku jazdy robota po kontakcie z przeszkodą (Rys. ??). Zmiana ta miała charakter losowy i była bezpośrednio zależna od miejsca oraz kąta uderzenia, a nie od zaprogramowanej logiki sterowania. Nawet w przypadku minimalnych odchyleń kursu obserwowane zachowanie robota było niepowtarzalne i niemożliwe do kontrolowania, co jednoznacznie wskazuje na niedopuszczalność takich warunków pracy w kontekście systemów autonomicznych.

Uzyskane wyniki potwierdzają, że brak detekcji przeszkód znajdujących się poza polem widzenia czujnika prowadzi do nieprzewidywalnych reakcji robota i stanowi istotne ograniczenie zaprojektowanego systemu percepcji. W praktyce oznacza to konieczność rozszerzenia systemu sensorycznego lub zastosowania dodatkowych mechanizmów zabezpieczających, jeśli robot ma poruszać się w środowisku o podwyższonym stopniu nieuporządkowania.



Rys. 5.6. Omijanie przeszkód poza polem wykrywania czujnika, przeszkoda została ominięta, kierunek jazdy robota został zauważalnie zmieniony, strzałka pokazuje pierwotny kierunek poruszania robota

5.5. Omijanie przeszkód o nieregularnym kształcie

Celem przeprowadzonego testu była ocena skuteczności systemu detekcji przeszkód w przypadku obiektów o nieregularnym kształcie oraz analiza wpływu właściwości fizycznych przeszkody na zdolność jej wykrywania przez czujnik ultradźwiękowy. Szczególną uwagę poświecono rozróżnieniu, czy kluczowym czynnikiem ograniczającym detekcję jest przezroczystość materiału, czy raczej geometria obiektu i sposób odbicia fali ultradźwiękowej.

Oczekiwany efektem testu było wykrycie przeszkody niezależnie od jej kształtu i właściwości optycznych, przy założeniu, że znajduje się ona w polu widzenia czujnika i w zasięgu jego pracy. Zakładano, że robot zareaguje zgodnie z zaimplementowanym algorytmem, tj. zatrzyma się, wykona cofanie oraz podejmie próbę ominienia obiektu. W idealnym przypadku przezroczystość przeszkody nie powinna mieć znaczenia, ponieważ czujnik ultradźwiękowy opiera swoje działanie na emisji i odbiorze fal akustycznych, a nie promieniowania świetlnego. Istotnym czynnikiem pozostaje natomiast kształt powierzchni przeszkody, który może powodować rozproszenie fali i jej odbicie w kierunku innym niż czujnik.

W ramach badań przeprowadzono trzy próby eksperymentalne. W pierwszej z nich jako przeszkodę zastosowano szklankę. Robot zareagował dopiero po fizycznym kontakcie z obiektem, co sugeruje, że fala ultradźwiękowa nie została skutecznie odbita w stronę czujnika (Rys. ??). Po kontakcie algorytm zadziałał poprawnie — robot wycofał się, a następnie ominął przeszkodę zgodnie z założeniami programu.



Rys. 5.7. Omijanie przeszkód o nieregularnym kształcie, przeszkoda pomimo kontaktu została ominięta

W drugiej próbie ponownie wykorzystano szklankę, jednak w tym przypadku robot całkowicie zignorował jej obecność i nie podjął żadnej reakcji, nawet po kontakcie (Rys. ??). Oznacza to, że detekcja obiektu nie była powtarzalna i zależała od położenia przeszkody względem czujnika oraz kąta padania fali ultradźwiękowej.



Rys. 5.8. Omijanie przeszkód o nieregularnym kształcie, przeszkoda została zignorowana

W trzeciej próbie zastosowano butelkę o nieregularnym kształcie, wypełnioną wodą. Pomimo większych rozmiarów obiektu i jego masy, przeszkoda również nie została wykryta przez system sensoryczny robota (Rys. ??). Wskazuje to jednoznacznie,

że czynnikiem wpływającym na brak detekcji była geometria powierzchni obiektu, powodująca rozproszenie lub pochłanianie fali ultradźwiękowej.



Rys. 5.9. Omijanie przeszkód o nieregularnym kształcie, przeszkoda została zignorowana i spowodowała przewrócenie robota

5.6. Omijanie przeszkód o chropowatej powierzchni

Celem niniejszego testu było sprawdzenie skuteczności detekcji przeszkód o powierzchni nieregularnej i chropowatej, która z punktu widzenia czujnika ultradźwiękowego powinna sprzyjać odbiciu fali akustycznej w kierunku nadajnika. Test ten stanowił uzupełnienie wcześniejszych badań, w których analizowano zachowanie robota w kontakcie z obiektyami o gładkich i zaokrąglonych powierzchniach, charakteryzujących się obniżoną skutecznością odbicia ultradźwięków.

Oczekiwany efektem było poprawne wykrycie przeszkody znajdującej się w polu widzenia czujnika oraz uruchomienie standardowej sekwencji reakcji robota. Zakładano, że chropowata struktura powierzchni obiektu spowoduje wielokrotne, rozproszone odbicia fali ultradźwiękowej, co zwiększy prawdopodobieństwo jej powrotu do odbiornika czujnika, niezależnie od właściwości optycznych materiału.

W ramach testu jako przeszkodę zastosowano przezroczyste pudełko o wyraźnie chropowatej strukturze powierzchni (Rys. ??). Pomimo przezroczystości materiału, przeszkoda została wykryta w sposób jednoznaczny i powtarzalny. Robot prawidłowo zareagował na obecność obiektu, inicjując procedurę omijania zgodnie z założeniami algorytmu sterowania.



Rys. 5.10. Omijanie przeszkody o chropowatej powierzchni

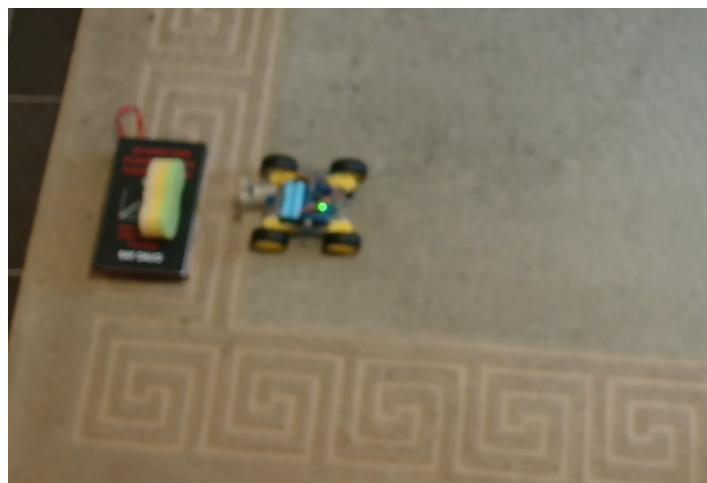
5.7. Omijanie przeszkód o gąbczastej strukturze

Celem przeprowadzonego testu była analiza zachowania robota w obecności przeszkód wykonanych z materiałów o wysokiej porowatości i zdolności pochłaniania energii fali ultradźwiękowej. Tego typu obiekty stanowią szczególne wyzwanie dla systemów detekcji opartych na czujnikach ultradźwiękowych, ponieważ struktura materiału może w istotny sposób wpływać na propagację oraz odbicie sygnału akustycznego.

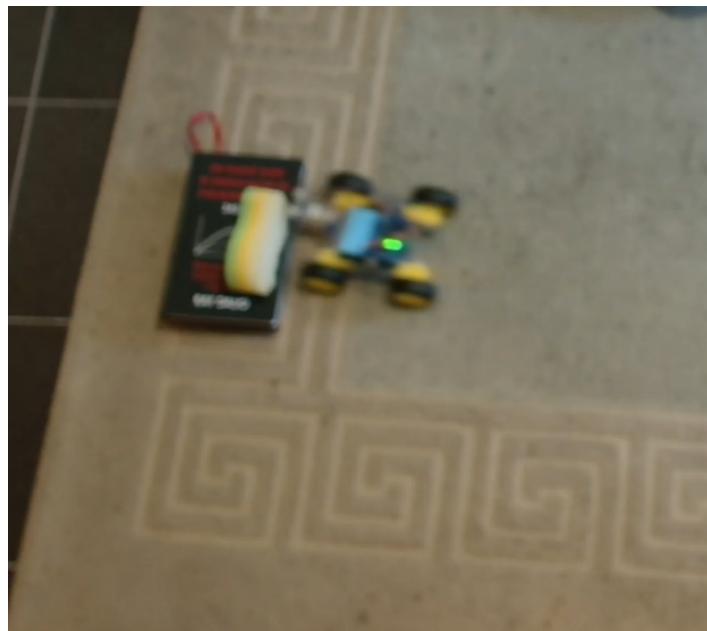
Oczekiwany efektem testu było wykrycie przeszkody znajdującej się w polu pomiarowym czujnika oraz uruchomienie standardowej procedury omijania. Zakładano jednak możliwość obniżonej skuteczności detekcji ze względu na właściwości fizyczne gąbczastego materiału.

W ramach badań przeprowadzono dwa testy różniące się orientacją przeszkody względem robota. W pierwszym przypadku gąbka została ustawiona stroną gładszą w kierunku czujnika (Rys. ??). W takiej konfiguracji robot reagował na przeszkodę w sposób niestabilny – w części prób obiekt był poprawnie wykrywany, natomiast w pozostałych przypadkach pomiar odległości nie inicjował reakcji algorytmu sterowania.

W drugim teście gąbka została odwrócona stroną bardziej porowatą i nieregularną w stronę robota (Rys. ??). W tej konfiguracji reakcja robota była sporadyczna lub nie występowała wcale, co skutkowało brakiem uruchomienia procedury omijania.



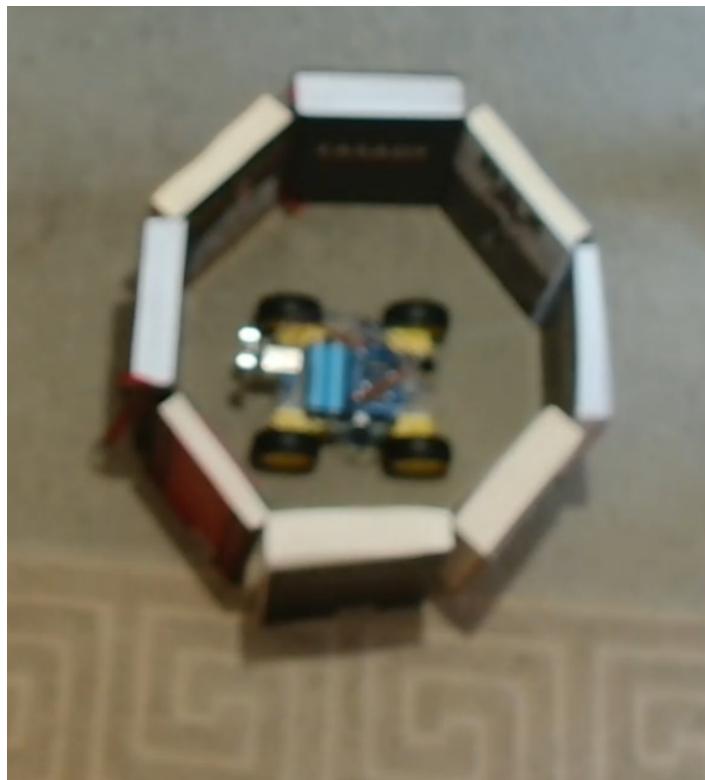
Rys. 5.11. Omijanie przeszkody o gąbczastej strukturze zwróconej gładką stroną



Rys. 5.12. Omijanie przeszkody o gąbczastej strukturze zwróconej porowatą stroną

5.8. Zachowanie robota w sytuacji całkowitego otoczenia przeszkodami

Celem przeprowadzonego testu była ocena zachowania robota w skrajnie niekorzystnym scenariuszu, w którym platforma mobilna zostaje otoczona przeszkodami z każdej strony, uniemożliwiając swobodny przejazd w dowolnym kierunku (Rys. ??). Tego typu sytuacja stanowi istotny przypadek graniczny dla prostych algorytmów reaktywnych, które nie wykorzystują globalnej mapy otoczenia ani pamięci wcześniejszych stanów środowiska.



Rys. 5.13. Robot otoczony przeszkodami

Oczekiwany efektem testu było wykrycie przeszkody znajdującej się bezpośrednio przed robotem oraz podjęcie próby zmiany trajektorii ruchu zgodnie z zaimplementowanym algorytmem omijania. Zakładano jednak, że w przypadku braku dostępnej wolnej przestrzeni robot nie będzie w stanie wyznaczyć bezpiecznego kierunku jazdy i jego zachowanie może odbiegać od poprawnego scenariusza omijania pojedynczej przeszkody. Test ten miał na celu weryfikację odporności algorytmu na sytuacje, w których lokalna percepceja nie dostarcza wystarczających informacji do podjęcia jednoznacznej decyzji ruchowej.

W trakcie próby robot poprawnie wykrył przeszkodę znajdującą się przed nim i zgodnie z logiką programu wykonał manewr cofania. Ruch wsteczny, realizowany bez dodatkowej weryfikacji otoczenia za robotem, doprowadził do kontaktu z przeszkodami znajdującymi się z tyłu konstrukcji, w wyniku czego część z nich została przewrócona (Rys. ??). Po zakończeniu manewru cofania robot powrócił do standardowego trybu pracy i kontynuował działanie algorytmu omijania przeszkód, nie identyfikując sytuacji jako stanu krytycznego.

Uzyskane rezultaty wskazują na istotne ograniczenie zastosowanego rozwiązania sterowania reaktywnego. Algorytm, oparty wyłącznie na pomiarze odległości w

jednym kierunku oraz prostym skanowaniu bocznym, nie posiada mechanizmów pozwalających na rozpoznanie sytuacji pełnego otoczenia ani na bezpieczne planowanie ruchu cofania. Brak czujników monitorujących przestrzeń za robotem oraz brak globalnej oceny sytuacji prowadzą do zachowań przypadkowych, które w rzeczywistych warunkach mogłyby skutkować uszkodzeniem robota lub elementów otoczenia. Test ten jednoznacznie pokazuje, że w bardziej wymagających scenariuszach niezbędne jest zastosowanie dodatkowych czujników, pamięci stanu lub bardziej zaawansowanych algorytmów decyzyjnych.



Rys. 5.14. Przewrócenie przeszkód znajdujących się poza obszarem wykrywania robota

6. Podsumowanie

Celem niniejszej pracy było zaprojektowanie, wykonanie oraz przetestowanie prostego robota mobilnego zdolnego do autonomicznego poruszania się i omijania przeszkód z wykorzystaniem mikrokontrolera Arduino. Realizacja zadania obejmowała zarówno część teoretyczną, dotyczącą podstaw robotyki mobilnej i sterowania ruchem, jak i część praktyczną, obejmującą montaż konstrukcji mechanicznej, implementację oprogramowania sterującego oraz przeprowadzenie testów w zróżnicowanych warunkach środowiskowych.

W części teoretycznej omówiono podstawowe zagadnienia związane z robotami mobilnymi, w szczególności modele ruchu platform z napędem różnicowym oraz proste, reaktywne algorytmy omijania przeszkód. Zaprezentowane podstawy stanowiły bezpośrednie odniesienie do przyjętych rozwiązań projektowych, umożliwiając świadomy dobór komponentów oraz metod sterowania adekwatnych do charakteru zadania i ograniczeń sprzętowych.

Montaż robota obejmował integrację elementów mechanicznych, napędowych oraz sensorycznych. Konstrukcja została oparta na gotowym podwoziu z silnikami prądu stałego. Czujnik ultradźwiękowy, zamontowany na serwomechanizmie, umożliwił skanowanie otoczenia w ograniczonym zakresie kątowym, co znacząco zwiększyło funkcjonalność robota w porównaniu do rozwiązania z nieruchomym sensorem.

Część programistyczna projektu obejmowała implementację algorytmu decyzyjnego w pętli głównej programu, obsługę czujnika ultradźwiękowego, sterowanie serwomechanizmem oraz kontrolę silników. Zastosowane rozwiązania charakteryzowały się prostotą i czytelnością, co jest szczególnie istotne w projektach edukacyjnych. Jednocześnie w trakcie prac ujawniły się typowe problemy systemów wbudowanych, takie jak niestabilność pomiarów odległości, konieczność doboru odpowiednich opóźnień czasowych oraz brak informacji o stanie otoczenia poza aktualnym polem widzenia czujnika.

Przeprowadzone testy środowiskowe pozwoliły na kompleksową ocenę działania robota w różnych warunkach. Robot poprawnie poruszał się po prostym torze oraz był w stanie pokonać niewielkie nierówności terenu. W przypadku przeszkód znajdujących się w polu wykrywania czujnika algorytm omijania działał poprawnie, choć ujawniono jego wrażliwość na pozycję startową oraz promień skrętu. Testy z przeszkodami niewidocznymi dla czujnika, obiekty mi przezroczystymi, o nieregularnych kształtach lub

wykonanymi z materiałów pochłaniających fale ultradźwiękowe wykazały liczne ograniczenia zastosowanego rozwiązania sensorycznego. Szczególnie istotnym przypadkiem była sytuacja całkowitego otoczenia robota przeszkodami, w której brak informacji o przestrzeni za robotem doprowadził do kolizji podczas manewru cofania. Na podstawie uzyskanych wyników można sformułować następujące wnioski:

- 1) Zastosowanie pojedynczego czujnika ultradźwiękowego znaczaco ogranicza zdolność robota do poprawnej percepji otoczenia, zwłaszcza w przypadku obiektów o nieregularnym kształcie, przezroczystych lub wykonanych z materiałów dźwiękołoszących,
- 2) Algorytm reaktywny, pozbawiony pamięci stanu oraz globalnej reprezentacji otoczenia, jest podatny na zachowania losowe i nie radzi sobie w sytuacjach skrajnych, takich jak pełne otoczenie przeszkodami.
- 3) W części montażowej istotnym problemem okazał się brak czujników monitorujących przestrzeń za robotem oraz brak elementów zabezpieczających przed kolizją podczas cofania.
- 4) W części programowej możliwe byłoby zwiększenie niezawodności poprzez filtrację pomiarów, uśrednianie wyników, dynamiczny dobór promienia skrętu oraz wprowadzenie prostych mechanizmów wykrywania stanu „utknięcia”.
- 5) Alternatywnym rozwiązaniem mogłoby być zastosowanie dodatkowych czujników (np. kilku czujników ultradźwiękowych, czujników podczerwieni lub zderzaków mechanicznych) bądź implementacja bardziej zaawansowanych algorytmów decyzyjnych.

Pomimo wskazanych ograniczeń i niedoskonałości należy stwierdzić, że główny cel projektu został osiągnięty. Zbudowany robot mobilny jest zdolny do autonomicznego poruszania się oraz omijania przeszkód w typowych, nieskomplikowanych warunkach środowiskowych. Projekt spełnił również swoje założenia edukacyjne, umożliwiając praktyczne zapoznanie się z problemami projektowania systemów robotycznych, integracji sprzętu z oprogramowaniem oraz analizy zachowania robota w rzeczywistym środowisku.

Literatura

- [1] International Federation of Robotics: World Robotics 2023 – Service Robots. IFR, Frankfurt am Main, 2023.
- [2] Monk S.: Programming Arduino: Getting Started with Sketches. McGraw-Hill Education, New York, 2016.
- [3] Siegwart R., Nourbakhsh I. R., Scaramuzza D.: Introduction to Autonomous Mobile Robots. MIT Press, Cambridge, 2011.
- [4] Arduino: Arduino Uno Rev3 – Technical Documentation. Arduino SA, 2022.
- [5] Cytron Technologies: HC-SR04 Ultrasonic Sensor User Manual. Dokumentacja techniczna, 2013.
- [6] Adafruit Industries: Adafruit Motor Shield. Dokumentacja techniczna, 2018.
- [7] Jones J. L., Flynn A. M.: Mobile Robots: Inspiration to Implementation. A K Peters, Natick, 1999.
- [8] Horowitz P., Hill W.: The Art of Electronics. Cambridge University Press, Cambridge, 2015.
- [9] Alimisis D.: Educational Robotics: Open Questions and New Challenges. Themes in Science and Technology Education, Vol. 6, No. 1, 2013, s. 63–71.
- [10] Campion G., Bastin G., D’Andrea-Novel B.: Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. IEEE Transactions on Robotics and Automation, Vol. 12, No. 1, 1996, s. 47–62.
- [11] Arkin R. C.: Behavior-Based Robotics. MIT Press, Cambridge, 1998.
- [12] Kurniawan A., Hadiyoso S.: Performance Comparison of Ultrasonic Sensors for Obstacle Detection. International Journal of Engineering Research, Vol. 7, No. 4, 2018, s. 210–215.
- [13] Welch G., Bishop G.: An Introduction to the Kalman Filter. University of North Carolina at Chapel Hill, 2006.

- [14] Durrant-Whyte H., Bailey T.: Simultaneous Localization and Mapping: Part I. IEEE Robotics & Automation Magazine, Vol. 13, No. 2, 2006, s. 99–110.
- [15] Thrun S., Burgard W., Fox D.: Probabilistic Robotics. MIT Press, Cambridge, 2005.
- [16] Borenstein J., Everett H. R., Feng L.: Where am I? Sensors and Methods for Mobile Robot Positioning. University of Michigan, 1996.
- [17] Borenstein J.: Control and kinematic design of multi-degree-of-freedom mobile robots. IEEE Transactions on Robotics and Automation, Vol. 14, No. 2, 1998, s. 315–322.
- [18] Raibert M.: Legged Robots That Balance. MIT Press, Cambridge, 1986.
- [19] Brooks R. A.: A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation, Vol. 2, No. 1, 1986, s. 14–23.
- [20] Siciliano B., Sciavicco L., Villani L., Oriolo G.: Robotics: Modelling, Planning and Control. Springer, London, 2010.
- [21] Canudas-de-Wit C., Siciliano B., Bastin G.: Theory of Robot Control. Springer, London, 1996.
- [22] Bloch A. M.: Nonholonomic Mechanics and Control. Springer, New York, 2003.
- [23] Kenjo T.: Stepping Motors and Their Microprocessor Controls. Oxford University Press, Oxford, 1994.
- [24] Eguchi A.: Educational Robotics for Promoting 21st Century Skills. Journal of Automation, Mobile Robotics & Intelligent Systems, Vol. 8, No. 1, 2014, s. 5–11.
- [25] LaValle S. M.: Planning Algorithms. Cambridge University Press, Cambridge, 2006.
- [26] Erickson R. W., Maksimović D.: Fundamentals of Power Electronics. Springer, New York, 2020.

STRESZCZENIE PRACY DYPLOMOWEJ INŻYNIERSKIEJ

ROBOT MOBILNY Z NAWIGACJĄ AUTONOMICZNĄ OPARTY NA ARDUINO

Autor: Kacper Rychel, nr albumu: EF-173701

Opiekun: Dr inż. Mariusz Mączka

Słowa kluczowe: Arduino Uno, Robot mobilny, Czujnik HC-SR04, Czujnik Ultradźwiękowy Odległości

Cel pracy: Stworzenie robota mobilnego, który potrafi omijać przeszkody i poruszać się autonomicznie w środowisku. Zakres pracy: zaprojektowanie platformy robota z silnikami i czujnikami ultradźwiękowymi do detekcji przeszkód; zaprogramowanie algorytmu omijania przeszkód i nawigacji w środowisku; możliwość rozbudowy o funkcje śledzenia wyznaczonego celu lub powrotu do bazy. Testy robota w różnych scenariuszach, takich jak omijanie przeszkód czy poruszanie się po określonej trasie.

BSC THESIS ABSTRACT

MOBILE ROBOT WITH AUTONOMOUS NAVIGATION BASED ON ARDUINO

Author: Kacper Rychel, Student's ID: EF-173701

Supervisor: Prof. Mariusz Mączka

Key words: Arduino Uno, Mobile robot, HC-SR04 sensor, Ultrasonic Distance Sensor

Objective: To create a mobile robot that can avoid obstacles and navigate autonomously within its environment. Scope of work: Designing a robot platform with motors and ultrasonic sensors for obstacle detection; programming an algorithm for obstacle avoidance and navigation within the environment; and the potential for expansion with target tracking or return-to-base functionality. Testing the robot in various scenarios, such as obstacle avoidance and navigating along a defined route.