



POLITECHNIKA KRAKOWSKA im. T. Kościuszki
Wydział Inżynierii Elektrycznej i Komputerowej
Katedra Automatyki i Informatyki



Kierunek studiów: Informatyka

Specjalność: -

STUDIA NIESTACJONARNE

PRACA DYPLOMOWA INŻYNIERSKA

Jakub SZKLARCZYK

PROJEKT I KONSTRUKCJA ROBOTA JEŽDŽĄCEGO
ORAZ APLIKACJI STERUJĄcej

DESIGN AND CONSTRUCTION OF DRIVING ROBOT
WITH CONTROL APPLICATION

Opiekun pracy:
dr inż. Dariusz Dorota

Kraków, 2024.

Spis Treści

1 Wstęp	3
1.1 Wprowadzenie	3
1.2 Cel i zakres pracy	3
2 Projekt robota	4
2.1 Zebranie wymagań	4
2.2 Zakres możliwości	4
2.3 Scenariusze użytkowania	5
2.4 Wykorzystywane technologie	6
2.5 Wymagania projektu	7
2.5.1 Wymagania funkcjonalne	7
2.5.2 Wymagania niefunkcjonalne	8
3 Istniejące rozwiązania	9
3.1 Pierwsze roboty mobilne	9
3.2 Rozwój robotów mobilnych	10
3.3 Klasyfikacja robotów	12
3.4 Systemy operacyjne robotów	13
3.5 Stosowane czujniki oraz interakcja	13
3.6 Kierunki rozwoju robotów	14
4 Konstrukcja	15
4.1 Modele 3D	15
4.1.1 Projekt modeli	15
4.1.2 Wydruk modeli	16
4.2 Główne moduły sterujące	17
4.2.1 Opis połączeń	17
4.2.2 Specyfikacja głównych modułów	18
4.3 Ekspandery pinów	20
4.3.1 Ekspander analogowy MCP3008	20
4.3.2 Ekspander cyfrowy MCP23017	20
4.4 Czujniki zamontowane w robocie	20
4.4.1 Czujniki odległości HC-SR04	21
4.4.2 Czujniki płomieni Waveshare 9521	21
4.4.3 Czujnik temperatury i wilgotności DHT11	22

4.4.4	Zestaw czujników powietrza MQ	23
4.4.5	Diody LED Adafruit Neopixel	24
4.5	Poruszanie się robota	25
4.6	Przenoszenie przedmiotów	26
4.7	Zasilanie całego układu	27
4.7.1	Wykorzystane elementy	27
4.7.2	Czas działania robota	28
4.8	Pokaz robota	29
5	Oprogramowanie	30
5.1	Program na arduino	30
5.1.1	Zasada działania kodu	30
5.1.2	Sterowanie silnikami	31
5.1.3	Odczytywanie wartości sensorów	33
5.2	Konfiguracja i oprogramowanie mikrokomputera	35
5.2.1	Konfiguracja systemu	35
5.2.2	Sterowanie ramieniem	36
5.3	Aplikacja mobilna	37
5.3.1	Ogólny opis aplikacji	37
5.3.2	Ustawienia aplikacji	38
5.3.3	Sterowanie robotem	41
5.3.4	Dane z czujników	43
5.3.5	Pozostałe strony	46
5.4	Kod źródłowy	46
5.5	Działanie robota	47
6	Podsumowanie	48
6.1	Robot na tle innych rozwiązań	48
6.2	Zrealizowane założone cele	48
Spis rysunków	49	
Spis tabel	50	
Spis kodów programów	50	
Bibliografia	51	

1 Wstęp

W tym rozdziale przedstawiono ogólny opis projektu, a także założenia, cel i zakres pracy. Znajdują się tu także informacje o wykorzystanych kontrolerach oraz czujnikach. Opisano tutaj również możliwe zastosowania robota.

1.1 Wprowadzenie

W dobie stale rosnącej informatyzacji i rozwoju technologii komputerowych, coraz większą rolę w codziennym życiu odgrywają roboty, wykonujące najróżniejsze czynności. Szeroka gama modułów i czujników, duża kompatybilność podzespołów, a także coraz szybszy rozwój obszaru IoT (ang. Internet of Things) sprawia, że jesteśmy w stanie skonstruować urządzenie o szerokim wachlarzu zastosowań. W tej pracy postanowiono skupić się na aspektach bezpieczeństwa, tworząc robota, którego głównym zadaniem jest sprawdzanie pomieszczeń i wykrywanie płomieni, a także innych, niebezpiecznych dla zdrowia i życia substancji. Całość porusza się na czterech kołach, które sterowane są poprzez aplikację, wykorzystując w tym celu komunikację Bluetooth. Urządzenie posiada też kamerę do podglądu obrazu na żywo, który transmitowany jest do aplikacji za pomocą technologii WiFi. W aplikacji wyświetlają się również dane z zainstalowanych czujników. Dodatkową funkcjonalnością robota jest ramię, pozwalające na transport drobnych przedmiotów. Wszystkimi elementami steruje mikrokontroler w połączeniu z mikrokomputerem, a całość zamontowana jest na specjalnie przygotowanej ramie, która została wydrukowana dzięki popularnej obecnie technologii druku 3D. Wszystkie elementy elektroniczne połączone są ze sobą łatwymi do montażu i demontażu przewodami, co pozwala na przebudowę lub rozbudowę robota o nowe elementy w przyszłości, oraz dostosowanie go do specjalistycznych potrzeb.

1.2 Cel i zakres pracy

Celem pracy dyplomowej jest przygotowanie projektu, rozpoczynając od analizy istniejących rozwiązań, zebrania wymagań systemowych i określenia zakresu możliwości urządzenia. Następnym krokiem jest dobranie odpowiednich części, przygotowanie modeli obiektów, a także napisanie odpowiedniego oprogramowania. Wszystkie elementy po złożeniu w całość mają za zadanie utworzyć działającego robota, poruszającego się na kołach, zdolnego do skanowania pomieszczeń i przenoszenia drobnych przedmiotów. Urządzenie ma być sterowane za pomocą dedykowanej aplikacji, w której znajdą się również dane z zainstalowanych czujników.

2 Projekt robota

Rozdział ten koncentruje się na opisaniu kolejnych kroków wykonanych w projekcie. Znajduje się tu m.in. zbieranie wymagań dotyczących projektu podzielonych na dwie grupy i przedstawionych w formie diagramów. Opisany został również zakres możliwości robota oraz scenariusze jego użytkowania, a także wykorzystane technologie, zarówno ze strony hardware'owej (użyte podzespoły), jak i software'owej (napisane programy).

2.1 Zebranie wymagań

W celu poprawnego wykonania pracy, kluczowe jest zrozumienie ogólnego zamysłu projektu, co stanowi podstawę dla dalszych działań. W tym celu postanowiono zebrać odpowiednie wymagania, gdyż jest to kluczowy etap w każdym projekcie inżynierskim. Poprawne zidentyfikowanie, analiza i dokumentacja zapewnia, że końcowy produkt będzie spełniał oczekiwania użytkowników i będzie zgodny z założeniami. Przeprowadzenie tego procesu w sposób systematyczny i zorganizowany jest niezbędne do osiągnięcia sukcesu. Jedną z najpopularniejszych metodyk zbierania wymagań jest analiza interesariuszy (osób potencjalnie zainteresowanych produktem), gdyż mogą oni dostarczyć cennych informacji na temat swoich oczekiwaniń oraz potrzeb, co pozwala na lepsze dopasowanie końcowego rozwiązania do rzeczywistych wymagań. Kolejnym istotnym elementem jest analiza istniejących już rozwiązań oraz najlepszych praktyk w danej dziedzinie, co umożliwia zidentyfikowanie mocnych i słabych stron dostępnych rozwiązań, a także uniknięcie błędów, które mogłyby wpłynąć na ostateczny kształt produktu końcowego. Ostatnim krokiem podczas zbierania wymagań było sporządzenie dokumentacji całego projektu, a także diagramów wymagań systemowych (podzielonych na funkcjonalne oraz niefunkcjonalne), które zostały przedstawione niżej. Dzięki tak kompleksowo przeprowadzonej analizie możliwe było zebranie odpowiednich wymagań, które są niezbędne do osiągnięcia sukcesu projektu. To z kolei dało możliwość stworzenia robota, który w pełni odpowiada oczekiwaniom użytkowników oraz spełnia wszystkie założenia.

2.2 Zakres możliwości

Robot dzięki dużej ilości modułów i czujników posiada bardzo szeroki wachlarz możliwości. Jego podstawowym zadaniem jest sprawdzanie pomieszczeń. Do tego celu zastosowano osiem różnych sensorów detekcji substancji w powietrzu. Zastosowanie czujniki z serii MQ pozwalają na wykrycie takich substancji jak Alkohol, Butan, CNG, Etanol, LPG, Metan, Tlenek Węgla (CO_2), Wodór (H_2), dym oraz inne łatwopalne gazy, co daje możliwość sprawdzenia stężenia szerokiej gamy różnych substancji, a także różnego rodzaju awarii, np. wyciek gazu. Dodatkowo robot posiada też czujnik temperatury i wilgotności powietrza. W urządzeniu znajdują się również dwa sensory odległości,

pozwalające na wykrywanie przeszkód, znajdujących się zarówno przed jak i za urządzeniem. Dzięki temu osoba sterująca robotem może mieć pewność, że nie trafi przez nieuwagę na jakąś przeszkodę. Obok czujników odległości, znajdują się również dwa sensory będące w stanie wykryć płomienie. Dzięki temu w bezpieczny sposób możemy skontrolować dane pomieszczenie, czy nie znajduje się w nim pożar. Kolejną podstawową funkcją jest przemieszczanie się na czterech kołach. Każde z kół sterowane jest oddziennie, za pomocą silnika z enkoderem, co pozwala na precyzyjne kierowanie robotem, a także na zawracanie w miejscu. Urządzenie posiada również kamerę, dzięki czemu operator może bez problemu sprawdzić kto lub co znajduje się w pomieszczeniu. Dodatkową funkcjonalnością jest uchwyt składający się z dwóch ramion, umieszczony z przodu robota, dzięki któremu możemy transportować przedmioty w niebezpieczne obszary. Całość zasilana jest z wydajnej baterii, pozwalającej na długi czas pracy urządzenia, którą można bezproblemowo naładować podpinając kabel USB (ang. Universal Serial Bus). Urządzenie posiada przełącznik ON / OFF pozwalający sterować zasilaniem, a także przycisk do bezpiecznego wyłączenia lub zrestartowania systemu (ze względu na to, że w projekcie wykorzystywany jest mikrokomputer z zainstalowanym systemem operacyjnym Linux, dodano przycisk który po naciśnięciu wysyła komendę bezpiecznego wyłączenia systemu operacyjnego).

2.3 Scenariusze użytkowania

Robot dzięki swojej szerokiej gamie czujników i modułów, a także możliwości rozbudowy posiada wiele zastosowań. Może on służyć do penetracji pomieszczeń w trakcie pożarów, trzęsień ziemi czy w przypadku podejrzeń rozszczelnienia instalacji gazowych. Urządzenie po podłączeniu zewnętrznego zasilania może służyć również jako stacjonarny punkt obserwacji pomieszczenia. Robot nadaje się również do obsługi innych maszyn, które mogą stanowić zagrożenie dla człowieka w kontakcie bezpośrednim. Innym ciekawym zastosowaniem może być również opieka szpitalna, np. podawanie leków pacjentom chorym zakaźnie. Urządzenie można wykorzystać nawet do różnych celów badawczych, takich jak monitorowanie stanu roślin w laboratoriach, a także warunków środowiskowych ich rozwoju. Finalnie, robota można również wykorzystać jako świetne narzędzie edukacyjne, pomagające uczniom i studentom w zrozumieniu programowania, robotyki i automatyki. Aby lepiej zobrazować jego potencjalne zastosowania, warto rozważyć kilka scenariuszy, które pokazują jego wszechstronność w różnych sytuacjach:

- Scenariusz podstawowy:** W sytuacji zagrożenia wyciekiem gazu w budynku, robot używany jest do sprawdzenia źródła i skali wycieku. Dzięki modułowi do detekcji gazów, urządzenie jest w stanie ocenić poziom niebezpieczeństwa i przekazać te informacje operatorowi. W przypadku wykrycia wysokiego stężenia gazu, można zarządzić ewakuację całego budynku, a w przypadku gazów łatwopalnych grożących wybuchem nawet całej pobliskiej okolicy.

2. **Scenariusz alternatywny:** W trakcie trzęsienia ziemi robot zostaje wykorzystany do przeszukiwania zawalonych budynków w celu lokalizacji ocalałych osób. Dzięki swoim czujnikom i kamerze, urządzenie jest w stanie wykrywać uwięzionych ludzi, a także omijać przeszkody i poruszać się w trudnym terenie. W przypadku zidentyfikowania poszkodowanych poprzez operatora robota, ratownicy mogą przystąpić do akcji ratunkowej.
3. **Scenariusz opcjonalny:** W środowisku szpitalnym robot zostaje przydzielony do opieki nad pacjentami zakaźnymi. Jego zadaniem jest dostarczanie leków oraz monitorowanie stanu zdrowia pacjentów, co pozwala ograniczyć kontakt personelu medycznego z osobami chorymi. W razie potrzeby, urządzenie może być również wykorzystywane do dezynfekcji pomieszczeń, przemieszczając się po korytarzach z odpowiednimi środkami.

2.4 Wykorzystywane technologie

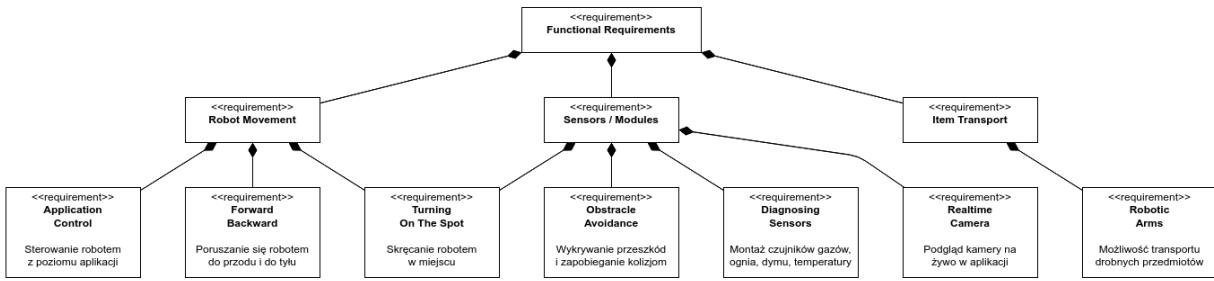
Robot składa się z trzech głównych modułów sterujących. Pierwszym z nich, odpowiedzialnym za poruszanie się czy też odczytywanie czujników jest mikrokontroler (Arduino Nano), na który został napisany dedykowany program w języku C++. Za całość pracy robota, a także za dostarczanie obrazu z kamery odpowiada mikrokomputer (Raspberry Pi Zero 2W) pod kontrolą systemu Linux, który sterowany jest za pomocą oprogramowania napisanego w języku Python. Wybrany został wyżej wymieniony model ze względu na jego niewielkie rozmiary i małe zużycie prądu, co przekłada się na dłuższy czas pracy na baterii. Komunikację pomiędzy dwoma wyżej wymienionymi urządzeniami zrealizowano za pomocą omawianego wcześniej przewodu USB, wykorzystując do tego protokół UART (ang. Universal Asynchronous Receiver Transmitter). Całość komunikuje się z telefonem z systemem Android za pomocą takich technologii jak Bluetooth oraz Wifi, z wykorzystaniem dedykowanej aplikacji, napisanej w języku C# w technologii MAUI (ang. Multi Platform Application User Interface). Dzięki zastosowaniu wyżej wymienionej wielo-platformowej technologii, aplikację można w razie potrzeby uruchomić również na innych systemach operacyjnych, jak np. iOS na urządzeniach mobilnych, a nawet na systemie Windows na urządzeniach stacjonarnych. Aplikacja pozwala na sterowanie robotem, podgląd obrazu z kamery, a także odczyt czujników. Całe urządzenie po złożeniu zostało umieszczone na specjalnie zaprojektowanej platformie, która została wydrukowana dzięki wykorzystaniu technologii druku 3D. Do tego celu wykorzystano najpopularniejsze w tej technologii materiały, takie jak PLA (ang. Polylactic Acid) oraz TPU (ang. Thermoplastic Polyurethane).

2.5 Wymagania projektu

Do poprawnego wykonania projektu zgodnie z założeniami zgromadzono szereg oczekiwani, które następnie podzielono na dwie podgrupy - wymagania funkcjonalne oraz niefunkcjonalne. Aby jak najlepiej przedstawić różnice pomiędzy tymi wymaganiami, można przyjąć określenie, mówiące że wymagania funkcjonalne opisują co system ma robić, a wymagania niefunkcjonalne opisują jak system ma działać. Do zobrazowania wymagań zdecydowano się użyć SysML (ang. Systems Modeling Language) [7], ze względu na swoje zaawansowane możliwości zintegrowanego modelowania systemów, obejmujących zarówno aspekty mechaniczne, elektryczne, jak i programowe. Umożliwia on precyzyjne definiowanie i śledzenie wymagań na różnych poziomach szczegółowości, co jest kluczowe dla skutecznego zarządzania projektem i weryfikacji spełnienia wymagań. Dzięki graficznym modelom oraz diagramom, jakie oferuje SysML, możliwe jest przedstawienie skomplikowanych zależności i wymagań w sposób bardziej przystępny i zrozumiały, co ułatwia współpracę i podejmowanie decyzji na różnych poziomach zarządzania projektem. Narzędzie to pozwala także na łatwiejsze identyfikowanie potencjalnych problemów czy nieścisłości już na wczesnym etapie, co może znacząco wpływać na efektywność całego procesu projektowego. Dzięki integracji wymagań funkcjonalnych i niefunkcjonalnych za pomocą SysML, możliwe jest także skuteczne zarządzanie weryfikacją i validacją projektu. Umożliwia to sprawdzenie, czy wszystkie zdefiniowane wymagania zostały spełnione oraz czy system działa zgodnie z oczekiwaniemi na każdym etapie realizacji.

2.5.1 Wymagania funkcjonalne

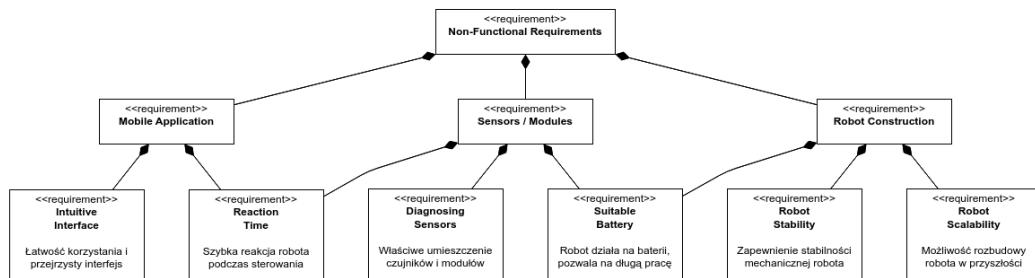
Są to wyznaczniki, które określają, co robot ma robić. Dotyczą one specyficznych funkcji, zachowań lub usług, które urządzenie ma zapewnić użytkownikom (rys. 1). Innymi słowy, są to konkretne działania, które system ma wykonywać, aby spełnić cele projektu. Do takich wymagań, w kontekście omawianego projektu należą m.in. poruszanie się (robot powinien poruszać się do przodu i do tyłu, a także skracać w miejscu) czy sterowanie zdalne (powinna być możliwość sterowania poprzez dedykowaną aplikację na telefon). Urządzenie powinno unikać przeszkód (wyposażenie w czujniki zapobiegające zderzeniu się z przeszkodą), a także posiadać moduły bezpieczeństwa (zainstalowanie sensorów takich jak np. czujnik dymu czy ognia). Ponadto powinna być możliwość podglądu z kamery (obraz powinien być transmitowany na żywo w aplikacji), oraz możliwość transportu przedmiotów (robot powinien posiadać ramię pozwalające na transport drobnych rzeczy). Wszystkie wyżej wymienione wymagania funkcjonalne zgrupowano i przedstawiono na poniższym diagramie (rys. 1).



Rys. 1: Wymagania funkcjonalne przedstawione w SysML, źródło: opracowanie własne

2.5.2 Wymagania niefunkcjonalne

Są to wyznaczniki, które określają, jak system ma działać, dotyczą jego właściwości, jakości i wydajności (rys. 2). Takimi wymaganiami w kontekście omawianego projektu są m.in. szybkość reakcji (pojazd powinien reagować natychmiastowo (maksymalne opóźnienie pół sekundy) po kliknięciu przycisku w aplikacji), wydajność baterii (urządzenie powinno działać jak najdłużej (minimum jedną godzinę, ze względu na czas potrzebny do sprawdzenia pomieszczenia w typowym scenariuszu) na zasilaniu z baterii), czy dokładność czujników (sensory powinny być odpowiednio dobrane i rozmieszczone, tak aby zminimalizować ryzyko błędного odczytu). Aplikacja powinna mieć przejrzysty interfejs (całość powinna być intuicyjna, tak aby każda osoba mogła bezproblemowo sterować robotem czy odczytywać wartości sensorów), robot powinien być stabilny mechanicznie (pojazd musi zapewnić bezpieczne poruszanie się w każdym terenie), a całość powinna być skalowalna (przygotowana na instalowanie dodatkowych czujników w przyszłości). Wszystkie wyżej wymienione wymagania niefunkcjonalne, tak samo jak w poprzednim punkcie zgrupowano i przedstawiono na poniższym diagramie (rys. 2).



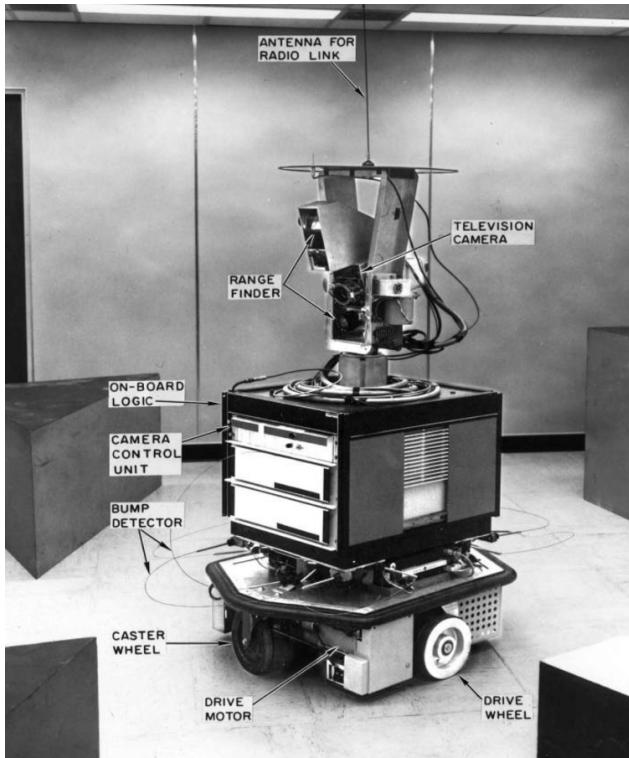
Rys. 2: Wymagania niefunkcjonalne przedstawione w SysML, źródło: opracowanie własne

3 Istniejące rozwiązania

W niniejszym rozdziale skupiono się na omówieniu historii robotyki, używanych systemach operacyjnych oraz ogólnej klasyfikacji tych urządzeń. Przedstawiono pierwsze rozwiązania, postęp technologiczny w dziedzinie robotyki w kolejnych latach, a także kierunki w jakich zmierza rozwój tej dziedziny nauki.

3.1 Pierwsze roboty mobilne

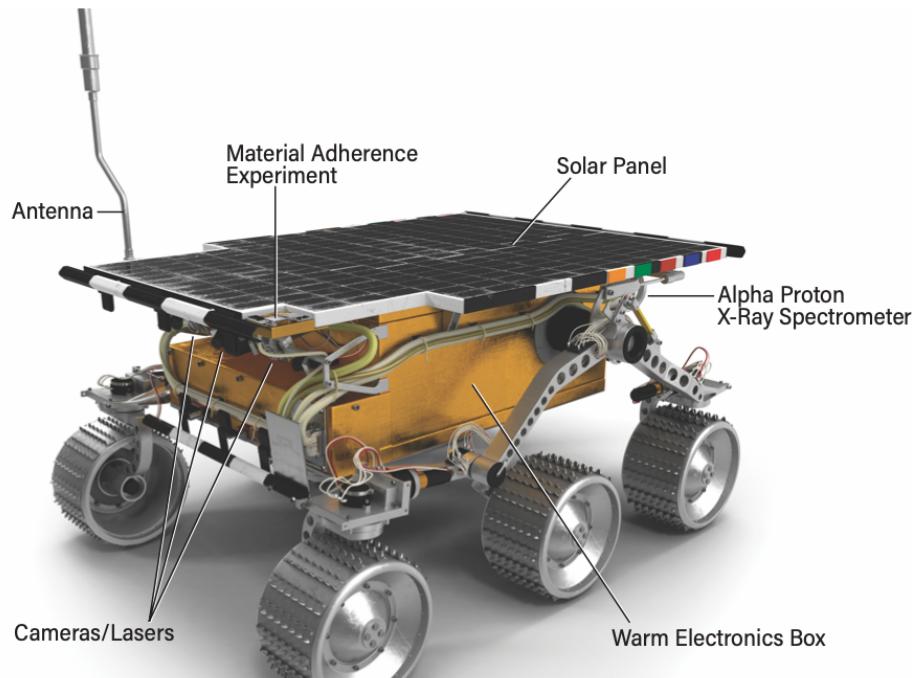
Na początku były to proste urządzenia, które miały na celu eksplorację i nawigację w kontrolowanych środowiskach. Te konstrukcje stanowiły podstawę dla współczesnych, zaawansowanych robotów, które w dzisiejszych czasach zdolne są do autonomicznego poruszania się i wykonywania skomplikowanych zadań. Pierwsze roboty powstały w połowie XX wieku, gdy naukowcy zaczęli eksperymentować z maszynami zdolnymi do samodzielnego poruszania się. Jednym z bardziej znanych jest robot o nazwie Shakey (rys. 3) [8], stworzony w latach 60. Był on pionierskim projektem, który łączył pierwsze elementy sztucznej inteligencji z podstawowymi funkcjami mobilności, co umożliwiało poruszanie się w złożonym środowisku i podejmowanie decyzji na podstawie otoczenia. Zrewolucjonizował on dziedzinę robotyki, pokazując potencjał robotów mobilnych. Dzięki takim urządzeniom, możliwe stało się dalsze badanie i rozwój technologii.



Rys. 3: Shakey, jeden z pierwszych robotów mobilnych, źródło: [9]

3.2 Rozwój robotów mobilnych

Postęp w dziedzinie robotyki mobilnej jest wynikiem licznych innowacji technologicznych i badawczych. Na przestrzeni dekad, roboty mobilne ewoluowały od prostych, zdalnie sterowanych urządzeń do zaawansowanych systemów autonomicznych, zdolnych do samodzielnego podejmowania decyzji i adaptacji do zmieniających się warunków. W latach 80. i 90. XX wieku, roboty takie jak *Sojourner* (rys. 4) [10], pierwszy łazik marsjański stworzony przez organizację NASA (ang. The National Aeronautics and Space Administration), pokazały, że mobilne maszyny mogą być użyteczne w eksploracji kosmosu. Zademonstrował on zdolność autonomicznego poruszania się po powierzchni innej planety, zbierając dane i przesyłając je na Ziemię. Był to przełomowy krok w robotyce mobilnej, pokazujący, że maszyny mogą wykonywać złożone zadania w ekstremalnych warunkach bez bezpośredniego nadzoru człowieka. To z kolei przyczyniło się do coraz to szybszej eksploracji kosmosu.



Rys. 4: Sojourner, pierwszy łazik marsjański opracowany przez NASA, źródło: [11]

Współczesne roboty, takie jak Boston Dynamics Spot (rys. 5) [12], pokazują niezwykłe możliwości w zakresie zwinności i adaptacji do różnorodnych środowisk, takich jak schody, nierówne powierzchnie czy przeszkody. Wyżej wymieniony czteronożny robot wyposażony w zaawansowane czujniki i kamery, może precyzyjnie nawigować i wykonywać zadania autonomicznie lub zdalnie, co czyni go idealnym narzędziem do zastosowań w przemyśle, czy podczas operacji ratunkowych. Kluczowym elementem rozwoju robotów mobilnych jest integracja zaawansowanych sensorów, które pozwalają na precyzyjne mapowanie otoczenia i unikanie przeszkód. Rozwój sztucznej inteligencji i uczenia maszynowego również odegrał kluczową rolę w umożliwieniu robotom bardziej zaawansowanej autonomii i lepszego rozumienia otaczającego świata.



Rys. 5: Spot, robot stworzony przez Boston Dynamics, źródło: [13]

Wyżej pokazane urządzenia są tylko jednymi z wielu przykładów robotów dostępnych na rynku. W dzisiejszym świecie, ze względu na coraz mniejsze koszty produkcji, coraz większą dostępność części, a także szybki rozwój sztucznej inteligencji, coraz więcej firm produkuje swoje własne roboty, dopasowane do konkretnych wymagań danej firmy.

3.3 Klasyfikacja robotów

Roboty mobilne można podzielić na kilka głównych kategorii w zależności od ich sposobu poruszania się oraz przeznaczenia. Urządzenia kołowe są najczęściej stosowane w środowiskach domowych i przemysłowych do zadań takich jak sprzątanie czy inspekcja. Charakteryzują się prostą konstrukcją i stosunkowo niskim kosztem produkcji. Roboty kroczące, jak np. Spot przedstawiony wyżej, są zdolne do poruszania się w trudnym terenie, gdzie koła nie są praktyczne. Ich złożona mechanika i zaawansowane algorytmy stabilizacji umożliwiają im wykonywanie skomplikowanych zadań w zróżnicowanych warunkach. Urządzenia pływające, takie jak autonomiczne pojazdy podwodne AUV (ang. Autonomous Underwater Vehicle), są używane głównie w badaniach oceanograficznych oraz w przemyśle naftowym do inspekcji i konserwacji podwodnych instalacji. Latające roboty, czyli drony (rys. 6), mają szerokie zastosowanie w monitoringu, fotografii, dostawach oraz w działaniach wojskowych. Jest to również kategoria robotów, które najczęściej możemy zobaczyć w codziennym życiu.



Rys. 6: Dron firmy Dji, sterowany pilotem, źródło: [14]

3.4 Systemy operacyjne robotów

Najczęściej stosowane systemy operacyjne dla robotów mobilnych to często systemy czasu rzeczywistego RTOS (ang. Real Time Operating System) [15] - specjalistyczny system operacyjny przeznaczony do obsługi aplikacji, które wymagają wysokiej precyzji czasowej i deterministycznego przetwarzania danych. Jest on stosowany w środowiskach, w których duża liczba zdarzeń musi zostać przetworzona w jak najkrótszym czasie lub w określonych terminach. RTOS jest kluczowy w systemach wbudowanych, gdzie czas reakcji na zdarzenia jest krytyczny. Innym popularnym rozwiązaniem jest system operacyjny dedykowany do tego typu urządzeń, czyli ROS (ang. Robot Operating System) [16] - oprogramowanie, które zapewnia zbiór narzędzi, bibliotek i konwencji, ułatwiających tworzenie złożonych i odpornych na błędy zachowań robotów w różnych dziedzinach, od przemysłowych urządzeń produkcyjnych po roboty badawcze. Całość jest dostępna za darmo, na licencji otwarto źródłowej, co oznacza że każdy może go pobrać, używać, modyfikować i dystrybuować bez ponoszenia kosztów licencyjnych. ROS jest rozwijany przez dużą społeczność deweloperów, co przyczynia się do jego ciągłego udoskonalania i rozszerzania funkcjonalności.

3.5 Stosowane czujniki oraz interakcja

Czujniki odgrywają kluczową rolę w funkcjonowaniu robotów mobilnych, umożliwiając im zbieranie informacji o otoczeniu, co jest niezbędne do nawigacji, unikania przeszkód oraz wykonywania określonych zadań. Można je podzielić na dwie grupy. Pierwsza z nich to czujniki kontaktowe, do której należą np. sensory siły czy dotyku. Druga to czujniki bezkontaktowe, w której znajdują się np. sensory magnetyczne czy laserowe, a także wykorzystane w projekcie optyczne (do wykrywania płomieni), ultradźwiękowe (do mierzenia odległości), temperaturowe czy chemiczne (do detekcji gazów). Czujniki optyczne, takie jak kamery i fotodiody, są powszechnie stosowane do wykrywania obiektów i analizy wizualnej, pozwalają też na tworzenie trójwymiarowych map otoczenia, co jest szczególnie przydatne w nawigacji i planowaniu ruchu. Czujniki ultradźwiękowe i LiDAR (ang. Light Detection and Ranging) są używane do precyzyjnego mierzenia odległości do obiektów i wykrywania przeszkód. Roboty mobilne często wykorzystują również czujniki inercyjne, takie jak akcelerometry i żyroskopy, które monitorują ruch i orientację w przestrzeni, co jest kluczowe dla stabilności i precyzyjnej kontroli ruchu. Czujniki dotykowe pozwalają robotom na wyczuwanie siły nacisku, co jest niezbędne podczas manipulacji obiektem. Ponadto, czujniki temperatury i wilgotności mogą być używane do monitorowania warunków środowiskowych, co jest szczególnie ważne w misjach ratunkowych lub eksploracyjnych. Integracja tych różnorodnych czujników z zaawansowanymi algorytmami przetwarzania danych umożliwia robotom nie tylko nawigację, ale także interakcję z ludźmi czy podejmowanie decyzji w czasie rzeczywistym, a w miarę rozwoju technologii oraz sztucznej inteligencji, roboty stają się coraz bardziej autonomiczne.

3.6 Kierunki rozwoju robotów

Rozwój robotów nieustannie napędzany jest przez postępy w technologii oraz zmieniające się potrzeby społeczne i ekonomiczne, a także sztuczną inteligencję, która w ostatnich czasach staje się coraz bardziej popularna. Początkowo te urządzenia wykorzystywane były głównie w przemyśle i w takich miejscach jak fabryki czy laboratoria, jednak coraz częściej pojawiają się także w naszych domach. Różnego rodzaju maszyny, takie jak np. roboty sprzątające, lub wykonujące za nas inne prace stają się codziennością. Dzięki szybkiemu rozwojowi tego obszaru, urządzenia te coraz częściej przypominają innych ludzi lub zwierzęta, porozumiewają się z nami dzięki sztucznej inteligencji i coraz częściej pełnią również funkcje rozrywkowe (rys. 7).



Rys. 7: Kerfuś, robot do zabawiania klientów, maskotka sklepu, źródło: [17]

4 Konstrukcja

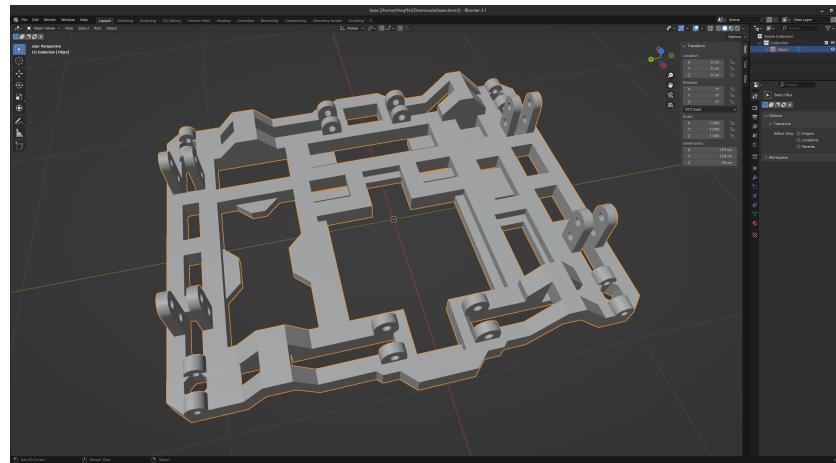
W niniejszym rozdziale przedstawiono kolejne etapy tworzenia robota. Zostały tutaj opisane wszystkie elementy, które zostały użyte do konstrukcji urządzenia. Przedstawiono tu podzespoły elektroniczne, wliczając w to zarówno główne elementy sterujące jak i różnego rodzaju moduły, a także obiekty stworzone specjalnie na potrzeby tego projektu, takie jak rama czy uchwyty, tworzące bazę dla pozostałych podzespołów.

4.1 Modele 3D

Po zebraniu wszystkich modułów wykorzystywanych w urządzeniu, postanowiono stworzyć i wydrukować modele, tak aby móc złożyć wszystkie elementy ze sobą. Na projekt oraz druk zdecydowano się głównie ze względu na wysokie dopasowanie wymiarów ramy robota do posiadanych podzespołów. Pozwoliło to dobrze zagospodarować przestrzeń i zmniejszyć wymiary całego urządzenia. Ponadto elementy wydrukowane na drukarce 3D są lekkie i stosunkowo tanie, co pozwoliło zmniejszyć zarówno wagę robota, jak i koszt jego produkcji.

4.1.1 Projekt modeli

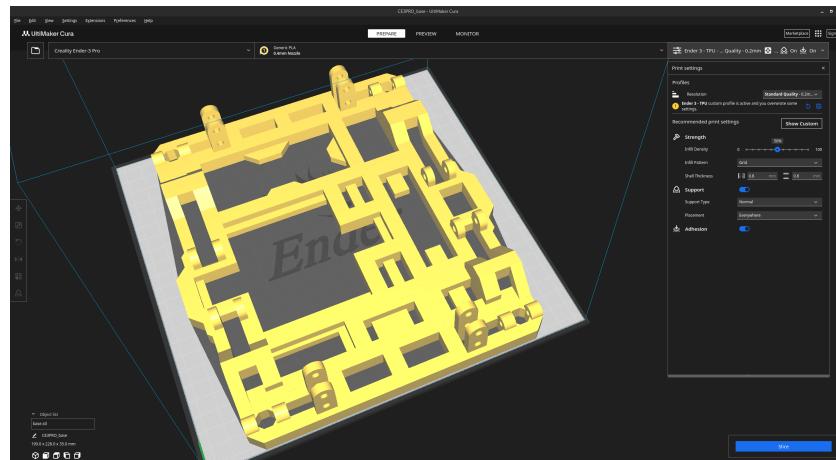
W celu złożenia urządzenia w całość zaprojektowano kilka modeli 3D w programie Blender. Podstawowym modelem jest rama robota (rys. 8) na której umieszczone zostały otwory przygotowane specjalnie pod wymiar wszystkich modułów i czujników. Zaprojektowano też otwory na śruby w odpowiednich miejscach, tak aby do ramy można było przymocować również inne obiekty. Następnie przygotowano modele uchwytów trzymających czujniki odległości oraz płomieni, tak aby zamontowane były pionowo, w kierunku jazdy robota. Zamodelowano także uchwyty na ramię urządzenia, służące do przenoszenia przedmiotów, a także same ramiona. Koła (zaplanowane do wydruku z materiałów typu plastik) zostały zaprojektowane z dużą średnicą 80 cm, tak aby robot poruszał się bez problemu również na nierównym terenie. Czym większy ich promień, tym łatwiej pokonywać przeszkody, takie jak kamienie czy dziury, bez zakłócania płynności jazdy. Dodatkowo, mniejszy kąt natarcia sprawia, że koła mniej się zapadają w nierównościach, co zwiększa kontrolę nad pojazdem. Zaprojektowane zostały również opony (z myślą o wydruku z materiałów gumowych) pozwalające zwiększyć przyczepność. Ze względu na ich materiał są także bardziej elastyczne i pochłaniają wstrząsy, co poprawia komfort poruszania się i chroni koła oraz inne części pojazdu przed uszkodzeniami.



Rys. 8: Okno programu Blender z zaprojektowaną ramą robota, źródło: opracowanie własne

4.1.2 Wydruk modeli

Wszystkie modele przygotowano do druku w programie Ultimaker CURA, a następnie wydrukowano na drukarce 3D. Do elementów takich jak rama (rys. 9), uchwyty czy koła wykorzystano poliaktyd, znany szerzej jako PLA (ang. Polylactic Acid) [18]. Jest to biodegradowalny materiał do druku 3D, który jest łatwy w użyciu i pochodzi z odnawialnych surowców, takich jak skrobia kukurydziana. Z kolei do wydruku uchwytów ramienia oraz opon wykorzystano termoplastyczny poliuretan, znany bardziej jako TPU (ang. Thermoplastic Polyurethane) [19]. Jest to elastyczny filament do druku 3D, znany z wytrzymałości i odporności na ścieranie, idealny do tworzenia elastycznych i wytrzymały części. Wszystkie elementy połączono ze sobą wykorzystując wspomniane wcześniej otwory montażowe oraz nakrętki i śruby o średnicy 3mm i różnej długości, dobranej odpowiednio do danego otworu montażowego.



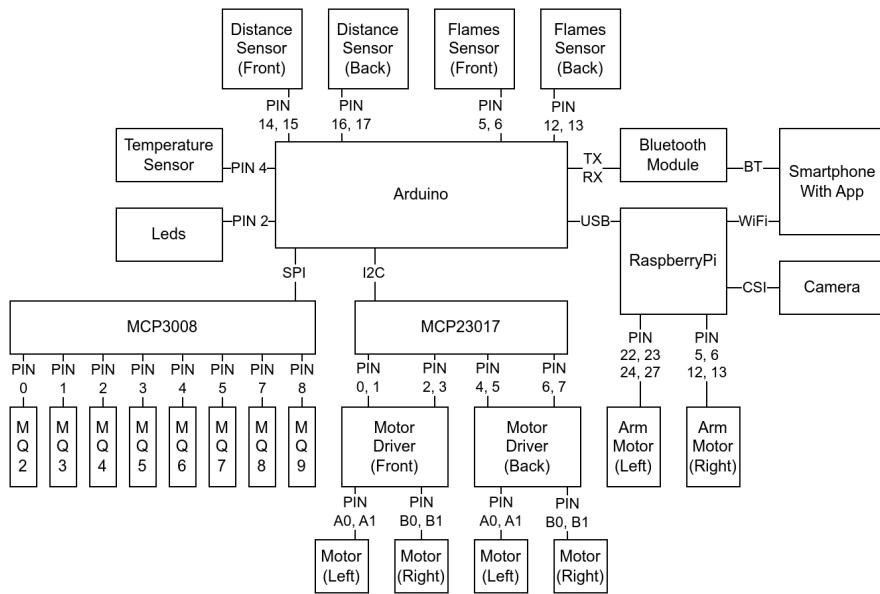
Rys. 9: Okno programu CURA z elementem do druku, źródło: opracowanie własne

4.2 Główne moduły sterujące

W tym podrozdziale przedstawiono bazowe układy robota, na które składa się mikrokontroler Arduino Nano oraz mikrokomputer Raspberry Pi Zero. Zebrano także ich szczegółową specyfikację w tabeli, a także opis wyprowadzeń dla każdej z wyżej wymienionych płyt. W tym rozdziale znajduje się również opis w jaki sposób poszczególne moduły komunikują się ze sobą. Przedstawiono także diagram połączeń pomiędzy tymi elementami.

4.2.1 Opis połączeń

Każdy z podzespołów robota został podłączony wykorzystując odpowiedni dla danego elementu interfejs. Głównymi modułami sterującymi są mikrokontroler Arduino Nano oraz mikrokomputer RaspberryPi Zero 2W, połączone za pomocą przewodu USB (ang. Universal Serial Bus). Do mikrokomputera podłączono silniki sterujące ramieniem wykorzystując piny GPIO (ang. General Purpose Input Output), a także kamerę za pomocą złącza CSI (ang. Camera Serial Interface). Następnie na karcie microSD został zainstalowany i odpowiednio skonfigurowany system Linux Raspberry Pi OS. Do mikrokontrolera podłączono bezpośrednio czujniki odległości oraz płomieni, czujnik temperatury i wilgotności oraz moduł LED, za pomocą pinów GPIO. W celu podłączenia ośmiu czujników powietrza oraz czterech silników napędzających robota wykorzystano ekspandery pinów, opisane w rozdziale 4.3. W celu udokumentowania wszystkich połączeń utworzono diagram (rys. 10) przedstawiający wszystkie elementy robota.



Rys. 10: Diagram połączeń pomiędzy poszczególnymi elementami, źródło: opracowanie własne

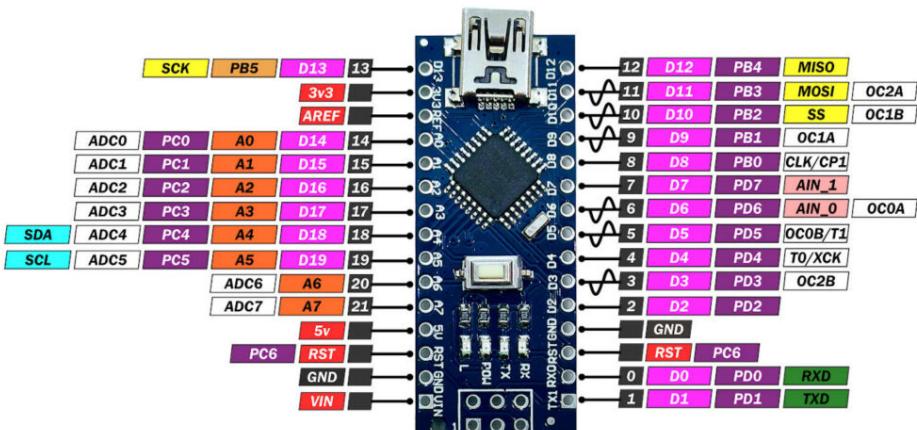
4.2.2 Specyfikacja głównych modułów

W celu dokładniejszego pokazania możliwości wykorzystanych układów, w poniższych tabelach przedstawiono ich specyfikację. Pozwala to zobrazować moc obliczeniową potrzebą do stworzenia robota. Głównym mikrokontrolerem odpowiadającym za sterowanie napędem i odczytem czujników jest Arduino Nano (tab. 1), posiadające łącznie 36 pinów, w tym 20 GPIO (rys. 11), z kolei za obsługę kamery odpowiada mikrokomputer Raspberry Pi Zero 2W (tab. 2), posiadający m.in. 40 pinów GPIO (rys. 12) oraz złącze CSI. Do wyżej wymienionego mikrokomputera dodano kamerę OV5647, której specyfikację również zamieszczono w tabeli ze specyfikacją Raspberry Pi.

Arduino Nano

Parametr	Wartość
Procesor	ATmega328P
Piny cyfrowe	14
Piny analogowe	8
Pamięć flash	32 KB
Pamięć SRAM	2 KB
Pamięć EEPROM	1 KB
Częstotliwość taktowania	16 MHz

Tab. 1: Specyfikacja mikrokontrolera, źródło: [20]



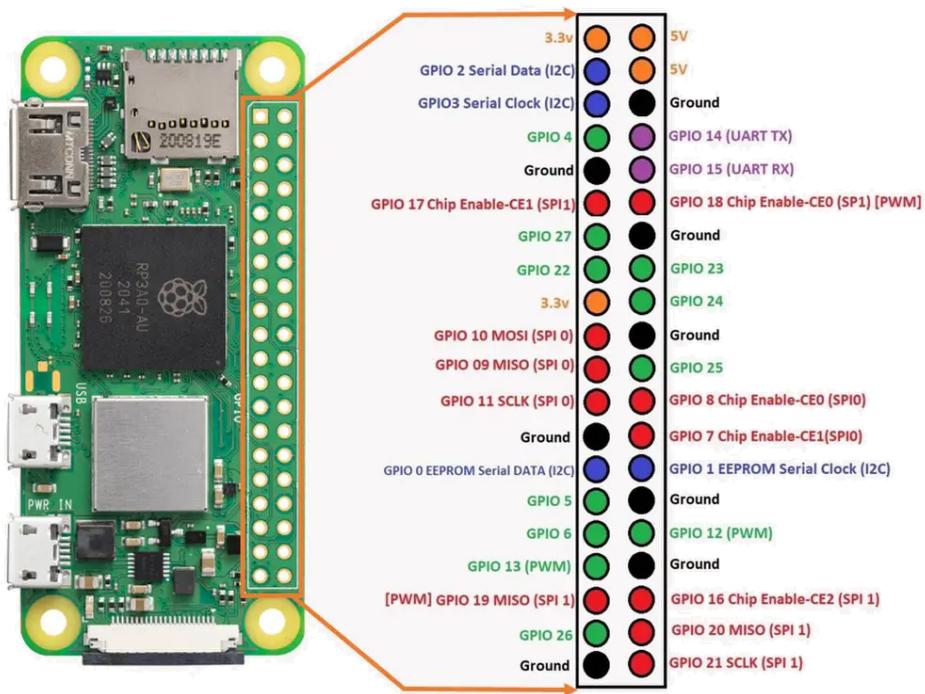
Rys. 11: Opis wyprowadzeń GPIO w Arduino Nano, źródło: [21]

RaspberryPi Zero 2W

+ Kamera OV5647

Parametr	Wartość
Procesor	Cortex-A53
Liczba rdzeni	4
Taktowanie procesora	1 GHz
Pamięć RAM	512 MB
WiFi	802.11 b/g/n/ac
Bluetooth	5.0
Interfejsy	1x USB, 1x HDMI
Złącze GPIO	40-pinowe
Karta Pamięci	MicroSD 8 GB
Rozdzielcość Kamery	5 Mpx
Kąt Widzenia Kamery	160°

Tab. 2: Specyfikacja mikrokomputera i kamery, źródło: [22]



Rys. 12: Opis wyprowadzeń GPIO w Raspberry Pi Zero 2W, źródło: [23]

4.3 Ekspandery pinów

W celu zainstalowania dużej ilości sensorów w robocie wykorzystano tzw. ekspander pinów. Jest to układ scalony pozwalający na dodanie pinów GPIO do urządzenia, wykorzystując najczęściej magistralę SPI (ang. Serial Peripheral Interface) lub IIC (ang. Inter Integrated Circuit).

4.3.1 Ekspander analogowy MCP3008

Do podłączenia ośmiu czujników z serii MQ (MQ2 - MQ9) zastosowano ekspayer pinów analogowych MCP3008, zwany inaczej przetwornikiem ADC (ang. Analog Digital Converter). Układ ten wyposażony jest w osiem pinów GPIO i pracuje z dokładnością 10 bitów, co oznacza, że operuje wartościami w przedziale od 0 do 1023. Moduł komunikuje się za pomocą magistrali SPI. Jest to szeregowy interfejs urządzeń peryferyjnych, jeden z najczęściej używanych interfejsów komunikacyjnych pomiędzy systemami mikroprocesorowymi a układami peryferyjnymi. Wykorzystuje $3+N$ pinów, gdzie N oznacza liczbę podłączonych urządzeń. Do komunikacji wykorzystuje trzy linie wspólne, MOSI (ang. Master Output Slave Input), MISO (ang. Master Input Slave Output) oraz SCLK (ang. Serial Clock). Oprócz tego wykorzystuje także jedną linię SS (ang. Slave Select), osobną dla każdego urządzenia, służącą do wyboru aktywnego urządzenia w transmisji danych [24].

4.3.2 Ekspander cyfrowy MCP23017

Do podłączenia silników odpowiadających za napęd robota wykorzystano ekspayer pinów cyfrowych MCP23017, posiadający łącznie 16 pinów GPIO. Pozwala on na odczyt i zapis wartości LOW oraz HIGH, oznaczających odpowiednio binarne 0 (`false`) oraz 1 (`true`). Układ komunikuje się korzystając z magistrali I²C, zwanej również IIC. Jest to szeregowa, dwukierunkowa magistrala służąca do przesyłania danych w urządzeniach elektronicznych. W przeciwieństwie do wcześniej omawianego SPI, którego ilość wykorzystywanych pinów zależy od ilości urządzeń, I²C charakteryzuje się stałą liczbą wykorzystywanych pinów bez względu na liczbę podłączonych elementów. Wykorzystuje on wspólne dwie linie dla wszystkich urządzeń, SDA (ang. Serial Data Line) oraz SCL (ang. Serial Clock Line), a wybór aktywnego elementu odbywa się za pomocą określenia adresu konkretnego urządzenia [25].

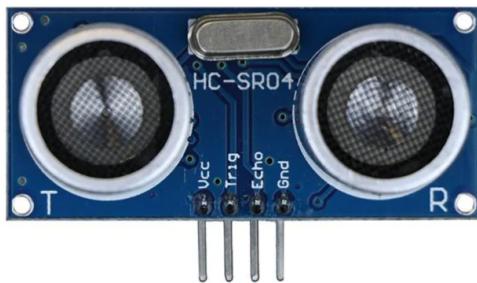
4.4 Czujniki zamontowane w robocie

W urządzeniu zamontowano szereg sensorów do mierzenia różnych wartości, które zostały założone na etapie planowania robota. Poniżej opisano osobno każdy z czujników, do czego służy i jak przebiega proces odczytywania jego wartości. Do każdego sensora dołączono również zdjęcie, przedstawiające jak dany układ wygląda.

4.4.1 Czujniki odległości HC-SR04

Czujnik dystansowy HC-SR04 (rys. 13) wykorzystuje ultradźwięki do pomiaru odległości sensora od obiektu. Zasada jego działania opiera się na emisji fali ultradźwiękowej oraz pomiarze czasu, jaki jest potrzebny na powrót odbitej fali do czujnika. Układ składa się z dwóch głównych elementów, nadajnika i odbiornika ultradźwięków. Cały proces pomiaru przebiega następująco:

1. Wysyłanie impulsu: Nadajnik emituje krótkie脉冲 ultradźwiękowe.
2. Czekanie na odbicie: Odbiornik nasłuchuje odbitych fal ultradźwiękowych.
3. Pomiar czasu: Mierzony jest czas od momentu wysłania impulsu do momentu jego powrotu.
4. Obliczenie odległości: Na podstawie zmierzonego czasu, znając prędkość poruszania się dźwięku w powietrzu, która wynosi ok. 340 m/s , obliczana jest odległość od obiektu, dzieląc wyżej wymienioną liczbę przez czas odbicia w sekundach.



Rys. 13: Czujnik odległości HC-SR04, źródło: [26]

4.4.2 Czujniki płomieni Waveshare 9521

Czujnik ognia i płomieni Waveshare 9521 (rys. 14) wykrywa obecność płomienia na podstawie promieniowania w zakresie podczerwieni IR (ang. Infrared), które jest emitowane przez ogień. Jest on również wrażliwy na bardzo silne źródła światła generowane przez płomienie. Główne elementy tego sensora to fotodioda IR oraz układ elektroniczny przetwarzający sygnał. Cały proces detekcji ognia lub płomieni wygląda następująco:

1. Detekcja promieniowania: Fotodioda IR rejestruje promieniowanie podczerwone emitowane przez płomienie (a także silne źródła światła takie jak ogień).
2. Przetwarzanie sygnału: Układ elektroniczny wzmacnia sygnał z fotodiody.
3. Wynik detekcji: Sygnał przetworzony jest przekazywany do mikrokontrolera, który interpretuje go jako obecność lub brak płomieni czy ognia.

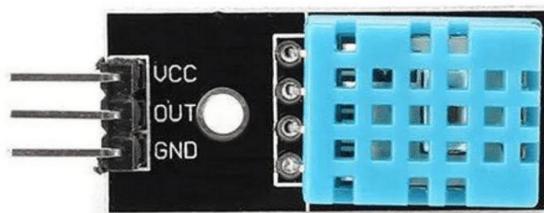


Rys. 14: Czujnik płomieni Waveshare 9521, źródło: [27]

4.4.3 Czujnik temperatury i wilgotności DHT11

Czujnik DHT11 (rys. 15) służy do pomiaru temperatury i wilgotności powietrza. Składa się z dwóch głównych elementów, termistora do pomiaru temperatury oraz rezystora pomiarowego do detekcji wilgotności. Cały przebieg pracy tego modułu wygląda następująco:

1. Wysłanie zapytania: Zewnętrzne urządzenie wysyła sygnał do mikrokontrolera, z prośbą o zmierzenie wartości temperatury i wilgotności oraz o przesłanie tych danych.
2. Pomiar temperatury: Termistor zmienia swoją rezystancję w zależności od temperatury, a mikrokontroler wbudowany w czujnik przetwarza te zmiany na wartość temperatury.
3. Pomiar wilgotności: Pomiar wilgotności odbywa się poprzez rezystor pomiarowy, którego właściwości elektryczne zmieniają się wraz z wilgotnością powietrza.
4. Przesyłanie danych: Mikrokontroler przetwarza sygnały z obu elementów, wykonuje konwersje uzyskanych wartości do odpowiednio stopni Celsjusza dla temperatury i wartości procentowych dla wilgotności, a następnie przesyła je w odpowiedzi jako cyfrowe dane do zewnętrznego urządzenia.



Rys. 15: Czujnik temperatury i wilgotności DHT11, źródło: [28]

4.4.4 Zestaw czujników powietrza MQ

Czujniki z serii MQ (rys. 16) są przeznaczone do wykrywania różnych gazów. Zasada działania tych czujników opiera się na zmianie przewodności warstwy półprzewodnikowej, która jest wrażliwa na obecność specyficznych gazów. Każdy czujnik MQ posiada element detekcyjny z warstwą półprzewodnikową (np. dwutlenku cyny), której przewodność zmienia się pod wpływem gazów. Cały przebieg działania każdego z tych układów jest podobny i wygląda następująco:

1. Ogrzewanie elementu: Warstwa półprzewodnikowa sensora jest podgrzewana, co umożliwia reakcję chemiczną z gazem, odpowiednim dla danego modelu czujnika.
2. Zmiana przewodności: W obecności określonego gazu, przewodność warstwy zmienia się.
3. Przetwarzanie sygnału: Zmiany przewodności są przetwarzane na sygnał elektryczny, który jest następnie wzmacniany i przesyłany do mikrokontrolera.



Rys. 16: Czujnik MQ2, jeden z ośmiu czujników powietrza, źródło: [29]

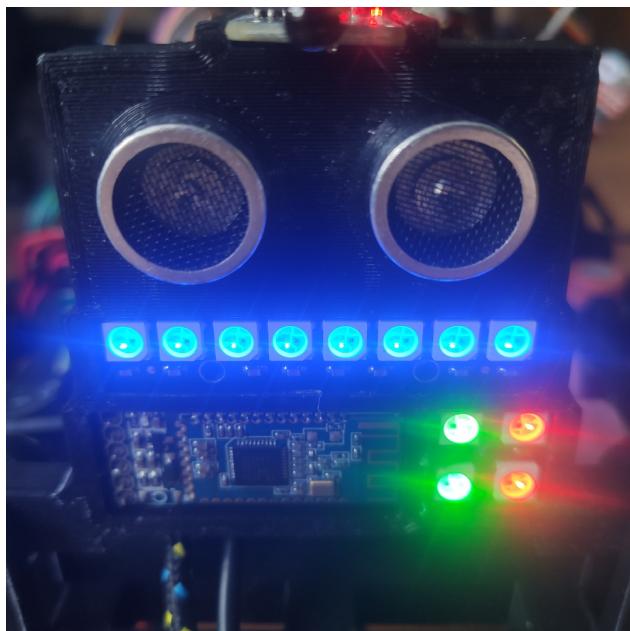
Każdy czujnik z serii MQ jest skalibrowany na inne gazy, co przedstawiono poniżej (tab. 3):

Czujnik	Wykrywane gazy
MQ2	Metan, Butan, LPG, Dym
MQ3	Alkohol, Etanol, Dym
MQ4	Metan, Gazy CNG
MQ5	LPG, Gazy Naturalne
MQ6	LPG, Butan
MQ7	Tlenek Węgla (CO)
MQ8	Wodór (H ₂)
MQ9	Gazy Łatwopalne

Tab. 3: Czujniki MQ i wykrywane przez nie gazy, źródło: opracowanie własne

4.4.5 Diody LED Adafruit Neopixel

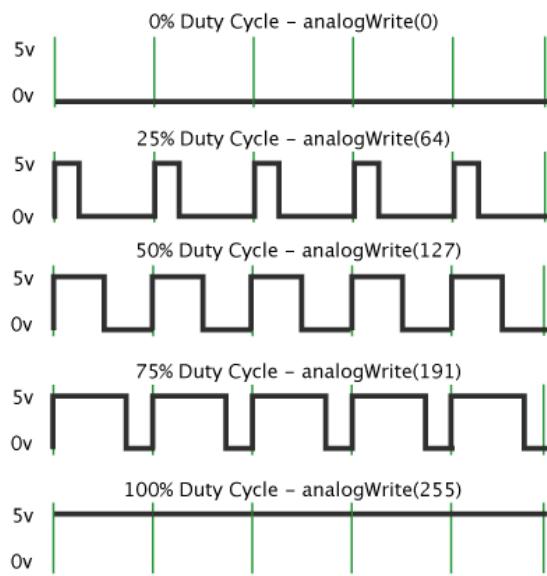
Do wyświetlania informacji o czujnikach (poza aplikacją) wykorzystano moduł LED Adafruit Neopixel (rys. 17) - jest to układ składający się z adresowalnych diod LED, które mogą emitować światło w pełnym spektrum RGB. Każda dioda posiada wbudowany sterownik, który umożliwia niezależne kontrolowanie koloru i jasności każdej z nich. Układ wykorzystuje jednokierunkowy protokół komunikacyjny (one-wire), co upraszcza sterowanie dużą ilością diod za pomocą jednego pinu GPIO. W projekcie użyto tego modułu do wyświetlania wartości z czujników MQ (8 diod LED w górnej części), oraz danych z czujników odległości i płomieni (4 diody LED w dolnej części). Każda z diod świeci się w kolorach od zielonego, poprzez żółty, do czerwonego, w zależności od procentowego wskazania czujnika. Dodatkowo, w przypadku sensorów MQ, które muszą być nagrzane przed użyciem, zastosowano opóźnienie 5 minut połączeniu zasilania. Jest to sygnalizowane niebieskim światłem diody przez kilka minut po uruchomieniu. Następnie gotowość użycia czujnika jest sygnalizowana poprzez zmianę koloru diody na kolor zgodny z wartością podawaną przez sensor w skali kolorów od zielonego do czerwonego - czym bardziej zielony kolor, tym mniejsza wartość czujnika (ok. 0%), kolor żółty oznacza średni zakres (ok. 50%) i odpowiednio czym bardziej czerwony kolor, tym większa wartość czujnika (ok. 100%). Opóźnienie wynika z konstrukcji układów, gdzie wg. danych producenta czujnik nie używany przez długi czas (ok. miesiąca lub więcej) powinien być wygrzewany ok. 24 godzin, natomiast przy używaniu go regularnie wystarczy rozgrzać go przez kilka minut. Po przeprowadzonych testach ustalono, że wartości sensorów stabilizują się już po ok. 5 minutach, dlatego też przyjęto takie opóźnienie.



Rys. 17: Adresowalne diody LED Adafruit Neopixel, źródło: opracowanie własne

4.5 Poruszanie się robota

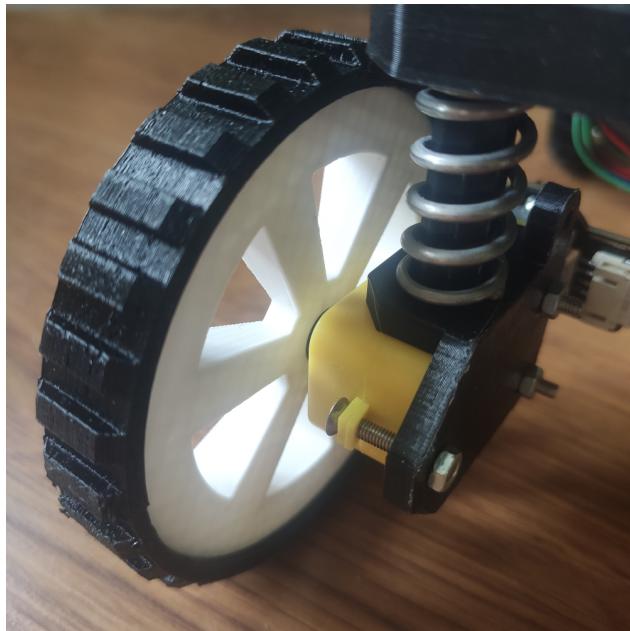
Poruszanie się robotem zrealizowano wykorzystując cztery silniki SJ01 z przekładnią 120 : 1, o prędkości maksymalnej 160 obr / min. W celu kontroli kierunku oraz prędkości silników, zostały one podłączone do dwukanałowych sterowników silników TB6612, znanych również jako mostki typu H. Do ustalenia kierunku poruszania się silnika wykorzystano dwa piny GPIO znajdujące się w układzie mostka H. Sterując odpowiednio wartościami, tj. ustawiając stan wysoki na pierwszym pinie i stan niski na drugim pinie silnik porusza się do przodu, a ustawiając stan niski na pierwszym pinie i stan wysoki na drugim pinie silnik porusza się do tyłu. W celu określenia prędkości poruszania się silnika wykorzystano kolejny pin GPIO znajdujący się na mostku H oznaczony jako PWM (ang. Pulse Width Modulation). Jest to metoda regulacji sygnału prądowego lub napięciowego, o stałej amplitudzie i częstotliwości, polegająca na zmianie wypełnienia sygnału, używana w zasilaczach impulsowych, wzmacniaczach impulsowych i układach sterujących pracą silników elektrycznych [30]. Przykład sygnału 8-bitowego (wartości od 0 do 255) wygenerowanego za pomocą PWM przedstawiono na poniższym rysunku (rys. 18):



Rys. 18: Przykładowe sygnały wygenerowane za pomocą PWM, źródło: [31]

Do silników zaprojektowano, wydrukowano i zamontowano koła, a także opony z wypustkami, w celu zapewnienia dużej przyczepności na nierównym i trudnym terenie. Całość umieszczono na specjalnie przygotowanym uchwycie, do którego dodatkowo zamontowano sprężyny, w celu zwiększenia amortyzacji podczas poruszania się, a następnie zamocowano do głównej ramy robota, przy pomocy śrub (rys. 19). Po uruchomieniu wszystkich silników w tym samym kierunku, robot porusza się odpowiednio do przodu, lub do tyłu w przypadku uruchomiania silników w kierunku

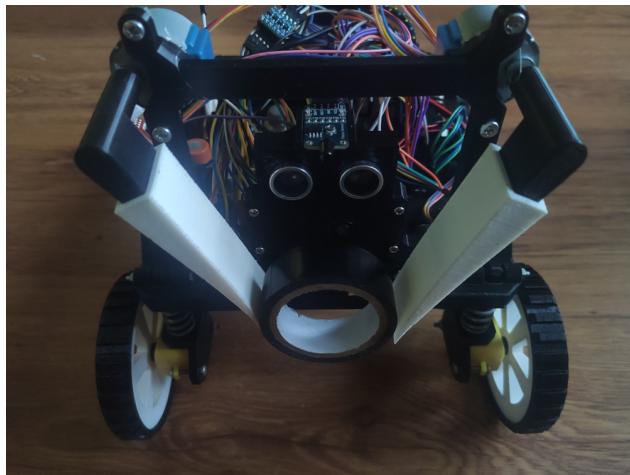
przeciwnym. Dzięki niezależnemu silnikowi na każdej z osi, możliwe było zrezygnowanie z osi skrętnej. Zamiast tego do poruszania się robotem w lewo lub w prawo, silniki po bokach robota poruszają się w przeciwnym kierunku. Dzięki temu, gdy prawa strona porusza się do przodu, a lewa do tyłu, robot skręca w lewo, a w przeciwnym przypadku, robot skręca w prawo. Taki napęd, znany również jako różnicowy, pozwala robotowi na skręcanie w miejscu, co zwiększa jego manewrowość i elastyczność w ograniczonych przestrzeniach. Układ ten eliminuje potrzebę osi skrętnej, co upraszcza konstrukcję i redukuje liczbę ruchomych części, co może zwiększyć niezawodność. Pozwala to również na bardziej precyzyjne kontrolowanie ruchów robota, co jest szczególnie ważne w zadaniach wymagających dużej dokładności.



Rys. 19: Koło z PLA i opona z TPU zamontowane w robocie, źródło: opracowanie własne

4.6 Przenoszenie przedmiotów

Robota rozszerzono o dodatkową funkcjonalność przenoszenia drobnych przedmiotów. W tym celu na specjalnie przygotowanej ramie zamontowano dwa silniki krokowe, znane również jako skokowe (rys. 20). Jest to to rodzaj silnika elektrycznego, który porusza się w dyskretnych krokach, a każdy impuls prądu dostarczany do silnika powoduje, że wirnik przemieszcza się o określoną liczbę stopni, co pozwala na precyzyjne kontrolowanie pozycji. Zastosowany silnik krokowy z przekładnią 28BYJ-48 ze sterownikiem ULN2003A posiada przekładnię 1 : 64, co pozwala na precyzyjne podnoszenie i upuszczanie przedmiotów. Ze względu na kamerę oraz czujniki odległości zamontowane z przodu robota, zamiast klasycznego jednego ramienia na środku, zamontowano dwa po bokach robota, tak aby nie zakłócać pracy innych modułów.



Rys. 20: Ramiona robota trzymające taśmę izolacyjną, źródło: opracowanie własne

4.7 Zasilanie całego układu

Do działania wszystkich wyżej wymienionych sensorów, a także modułów głównych potrzebne jest zasilanie. Ze względu na ogólną specyfikę robota, czyli mobilność, zamiast zasilania stałego z sieci postanowiono zamontować baterię. Dzięki temu robot jest nie jest ograniczony przez długość przewodu. W tym rozdziale opisano dokładnie, co zastosowano do zasilenia układu. Po dokonaniu niezbędnych obliczeń przedstawiono też pojemność akumulatorów i szacunkowy czas pracy na baterii całego układu.

4.7.1 Wykorzystane elementy

Za zasilanie całego układu odpowiada bateria składająca się z dwóch ogniw litowo-jonowych typu 18650 o napięciu 3,7V oraz pojemności 2600mAh każde (rys. 21). Zostały one umieszczone w specjalnym module, który zapewnia ich równomierne ładowanie i rozładowywanie, a także chroni ogniska przed przeładowaniem oraz nadmiernym rozładowaniem. Układ posiada dwie diody LED, sygnalizujące stan zasilania, dioda czerwona świeci się w momencie rozładowywania, a niebieska podczas ładowania baterii. Moduł posiada również wejście USB-C pozwalające na ładowanie akumulatora bez wyciągania ogniw z modułu za pomocą kabla USB i dowolnej ładowarki, takiej jak np. standardowa ładowarka do telefonu czy port USB w komputerze. Na płytce obecny jest także konwerter DC-DC (ang. Direct Current - prąd stały), konwertujący napięcie ok. 3.7V pochodzące z baterii na napięcie 5V pozwalające zasilić wszystkie elementy w układzie.



Rys. 21: Bateria typu 18650 wykorzystana do zasilania układu, źródło: [32]

4.7.2 Czas działania robota

Zużycie prądu przez cały układ po zmierzeniu oszacowano na ok. 2A w stanie spoczynku przy uruchomionych wszystkich czujnikach oraz modułach sterujących, oraz ok. 3A podczas poruszania się robotem. Całość zasilana jest napięciem 5V. Posiadając te dane obliczono szacunkowy czas działania robota na baterii:

Pojemność akumulatora i zgromadzona energia:

$$\text{Pojemność akumulatora: } 2 \times 2600 \text{ mAh} = 2 \times 2.6 \text{ Ah} = 5.2 \text{ Ah}$$

$$\text{Energia zgromadzona: } 5.2 \text{ Ah} \times 3.7 \text{ V} = 19.24 \text{ Wh}$$

Pobór prądu i czas działania w stanie spoczynku:

$$\text{Pobór prądu: } 2 \text{ A} \times 5 \text{ V} = 10 \text{ W}$$

$$\text{Czas działania: } \frac{19.24 \text{ Wh}}{10 \text{ W}} \approx 1.9 \text{ h}$$

Pobór prądu i czas działania w trakcie poruszania się:

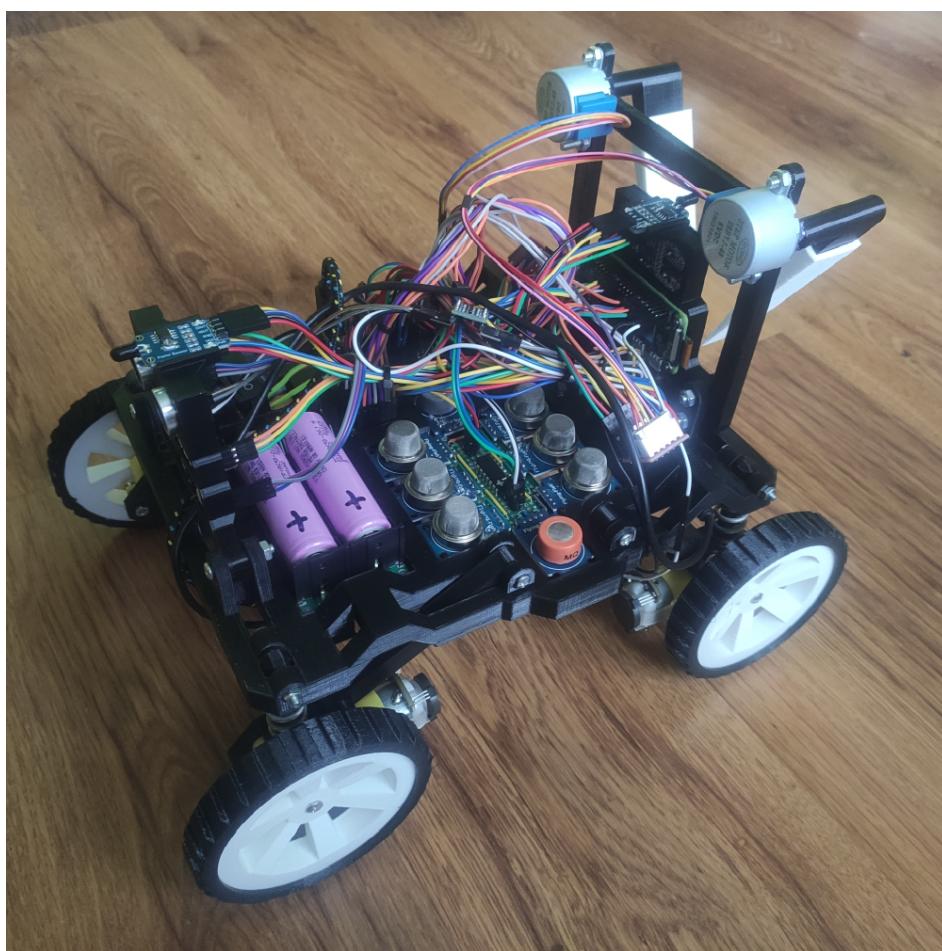
$$\text{Pobór prądu: } 3 \text{ A} \times 5 \text{ V} = 15 \text{ W}$$

$$\text{Czas działania: } \frac{19.24 \text{ Wh}}{15 \text{ W}} \approx 1.3 \text{ h}$$

Energię akumulatora obliczono mnożąc pojemność i napięcie ogniw, a pobór prądu obliczono mnożąc napięcie i prąd wszystkich modułów. Następnie otrzymaną pojemność podzielono przez pobór prądu i otrzymano czas działania na baterii. Z otrzymanych powyżej wyników oszacowano czas działania robota na ponad 1 godzinę w trakcie intensywnego użytkowania i niecałe 2 godziny w stanie spoczynku.

4.8 Pokaz robota

Wszystkie wymienione wcześniej elementy, czujniki, moduły i oprogramowanie pozwoliły na złożenie robota w całość, tak jak pokazano to na zdjęciu poniżej. Elementy plastikowe takie jak rama czy uchwyty zostały połączone ze sobą używając najczęściej śrub i nakrętek o odpowiednio dobranej średnicy i długości. W celu połączenia modułów wykorzystano popularne przewody prototypowe, tak aby umożliwić łatwą przebudowę robota w przyszłości. Następnie spięto je opaskami zaciskowymi, by nie płatały się ze sobą, szczególnie w trakcie pracy robota. Finalnie projekt po ukończeniu prezentuje się jak na zdjęciu poniżej (rys. 22):



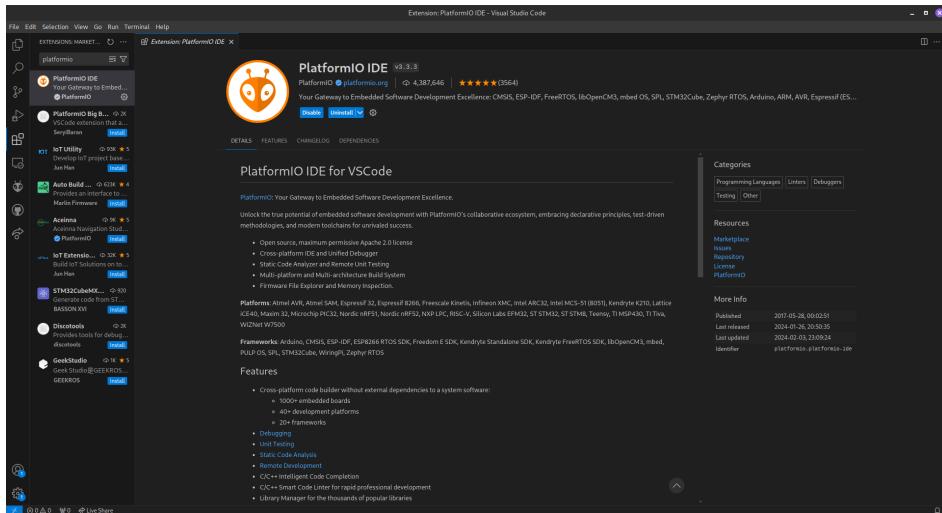
Rys. 22: Robot po złożeniu w całość, źródło: opracowanie własne

5 Oprogramowanie

Po złożeniu robota w całość, kolejnym krokiem do tego, aby urządzenie działało jest stworzenie oprogramowania. W tym celu napisano kod zarówno na mikrokontroler jak i na mikrokomputer. W celu zapewnienia możliwości sterowania robotem z telefonu utworzono aplikację mobilną.

5.1 Program na arduino

Pierwszym krokiem jaki wykonano w celu zaprogramowania robota było napisanie programu w języku C++ na mikrokontroler Arduino Nano. Do tego celu wykorzystano program Visual Studio Code z dodatkiem PlatformIO (rys. 23), pozwalającym na tworzenie oprogramowania na wiele popularnych platform ze świata IoT, w tym także Arduino.



Rys. 23: Program Visual Studio Code z dodatkiem Platform IO, źródło: opracowanie własne

5.1.1 Zasada działania kodu

Program posiada dwie główne funkcje, pierwszą z nich jest funkcja `setup`, która wykonuje się od razu po uruchomieniu mikrokontrolera. W tej funkcji zawarto takie instrukcje jak przypisanie poszczególnym pinom ich roli (wejście lub wyjście), rozpoczęcie komunikacji na protokołach takich jak SPI czy IIC, a także nawiązanie komunikacji z innymi urządzeniami. Drugą główną funkcją jest funkcja `loop` wykonująca kod w pętli, która uruchamia się zaraz po zakończeniu wykonywania funkcji startowej. Zaprogramowano w niej takie rzeczy jak komunikacja z modułem bluetooth, sterowanie silnikami odpowiadającymi za napęd czy odczytywanie wartości z czujników i wyświetlanie ich na diodach LED. Komendy wysyłane podczas komunikacji podzielono na sześć kategorii, w zależności od pierwszej litery wiadomości:

- 'A' - wiadomości przychodzące, dotyczą poruszania się robotem
- 'B' - wiadomości przychodzące, dotyczą poruszania ramieniem
- 'C' - wiadomości wychodzące, dotyczą czujników powietrza
- 'D' - wiadomości wychodzące, dotyczą czujnika odległości
- 'E' - wiadomości wychodzące, dotyczą temperatury i wilgotności
- 'F' - wiadomości wychodzące, dotyczą czujnika płomieni

Wszystkie komunikaty są tego samego priorytetu, jednak ze względu na to, że przesyłane są z różną częstotliwością (któłą dokładniej opisano w późniejszym rozdziale o numerze 5.1.3), postanowiono rozbić wartości każdego z sensorów na osobną wiadomość. Przykładowe przesyłane wiadomości wyglądają następująco:

- 'A F 1' - dane przekazane do napędu robota, oznaczają one odpowiednio kierunek do przodu (front) oraz standardową prędkość poruszania się (1).
- 'C 2 3 5 7 8 5 3 2' - dane uzyskane z czujników powietrza, są to wartości procentowe ośmiu czujników powietrza, kolejno od czujnika MQ2 do czujnika MQ9.
- 'E 23 45' - dane uzyskane z czujników temperatury i wilgotności, w tym przypadku oznacza to temperaturę 23°C oraz wilgotność powietrza na poziomie 45%.

5.1.2 Sterowanie silnikami

Sterowanie silnikami w programie odbywa się na zasadzie odczytywania w pętli poleceń przychodzących z modułu Bluetooth. Następnie określany jest kierunek wszystkich kół ustawiając wartość binarną 0 lub 1 na odpowiednich pinach, sprawdzany jest także warunek, czy można poruszać się w danym kierunku (pochodzący z czujnika odległości), a finalnie ustawiana jest prędkość za pomocą sygnału PWM generowanego na odpowiednich pinach. Omówione wyżej działania przedstawiono w poniższym fragmencie kodu programu (prog. 1):

```

1 void setMovingDirection(char direction)
2 {
3     // [ ... ]
4
5     if(direction == 'F' && CanMove_Forward)
6     {
7         ExpanderDigital.digitalWrite(PIN_MOT_FRDA, HIGH);
8         ExpanderDigital.digitalWrite(PIN_MOT_FRDB, LOW);
9         ExpanderDigital.digitalWrite(PIN_MOT_FLDA, LOW);
10        ExpanderDigital.digitalWrite(PIN_MOT_FLDB, HIGH);
11    }

```

```

12         ExpanderDigital.digitalWrite(PIN_MOT_RRDA, HIGH);
13         ExpanderDigital.digitalWrite(PIN_MOT_RRDB, LOW);
14         ExpanderDigital.digitalWrite(PIN_MOT_RLDA, LOW);
15         ExpanderDigital.digitalWrite(PIN_MOT_RLDB, HIGH);
16     }
17 }
18
19 // [ ... ]
20 }
21
22 void setMovingSpeed(int value)
23 {
24     analogWrite(PIN_MOT_FRS, value * 50);
25     analogWrite(PIN_MOT_FLS, value * 50);
26     analogWrite(PIN_MOT_RRS, value * 50);
27     analogWrite(PIN_MOT_RLS, value * 50);
28 }
29
30 void loop()
31 {
32     // [ ... ]
33
34     if(Serial.available())
35     {
36         String command = Serial.readStringUntil('\n');
37
38         if(command.charAt(0) == 'A')
39         {
40             setMovingDirection(command.charAt(2));
41             setMovingSpeed(command.charAt(4) - '0');
42             MotorTimeout = 25;
43         }
44
45         // [ ... ]
46     }
47
48     // [ ... ]
49 }
```

Prog. 1: Odczytywanie poleceń i poruszanie robotem

5.1.3 Odczytywanie wartości sensorów

Odczytywanie wartości czujników odbywa się również w głównej pętli programu, na zasadzie odczytania wartości sensora, a następnie wysłania jej poprzez moduł Bluetooth. Każdy z czujników odczytywany jest osobno, z różną częstotliwością, wynoszącą odpowiednio 0.1 sekundy dla sensorów płomieni i odległości, 1 sekundy dla czujników powietrza oraz 10 sekund dla sensora temperatury i wilgotności. Zastosowano taką częstotliwość, ze względu na priorytet czujników, tak aby nie generować zbyt dużo informacji i tym samym nie zapełniać przepustowości łącza. Dlatego też dane z czujników odległości czy płomieni wysyłane są często, aby umożliwić szybką reakcję na zmiany i zapobiec np. zderzeniu się. Z kolei wartości z sensorów powietrza wysyłane są ze średnią częstotliwością, a dane z temperatury wysyłane są rzadko, ponieważ zmieniają się wolniej. Dodatkowo dla czujników powietrza zastosowano 5 minutowy limit czasu ze względu na specyfikację tych sensorów, które muszą być rozgrzane, zanim zwrócią wiarygodny wynik. Dane w postaci wartości od 0 do 1023 przekształcane są na wartość procentową, oraz na kolor ze skali RGB, od zielonego, poprzez żółty, aż do czerwonego. Ostatecznie, dane z wszystkich czujników po odczytaniu przesyłane są do aplikacji oraz wyświetlane na diodach LED. Całość pokazana została na poniższym fragmencie kodu programu (prog. 2):

```
1 void refreshSensorValues()
2 {
3     long time = millis();
4
5     // [ ... ]
6
7     if(SensorLastUpdate_RefreshRateMedium + 1000 < time)
8     {
9         SensorLastUpdate_RefreshRateMedium = time;
10
11        if(time > 600 * 1000)
12        {
13            SensorValue_MQ2 = convertToPercent(ExpanderAnalog.readADC(0));
14            SensorValue_MQ3 = convertToPercent(ExpanderAnalog.readADC(1));
15            SensorValue_MQ4 = convertToPercent(ExpanderAnalog.readADC(2));
16            SensorValue_MQ5 = convertToPercent(ExpanderAnalog.readADC(3));
17            SensorValue_MQ6 = convertToPercent(ExpanderAnalog.readADC(4));
18            SensorValue_MQ7 = convertToPercent(ExpanderAnalog.readADC(5));
19            SensorValue_MQ8 = convertToPercent(ExpanderAnalog.readADC(6));
20            SensorValue_MQ9 = convertToPercent(ExpanderAnalog.readADC(7));
21        }
22    }
```

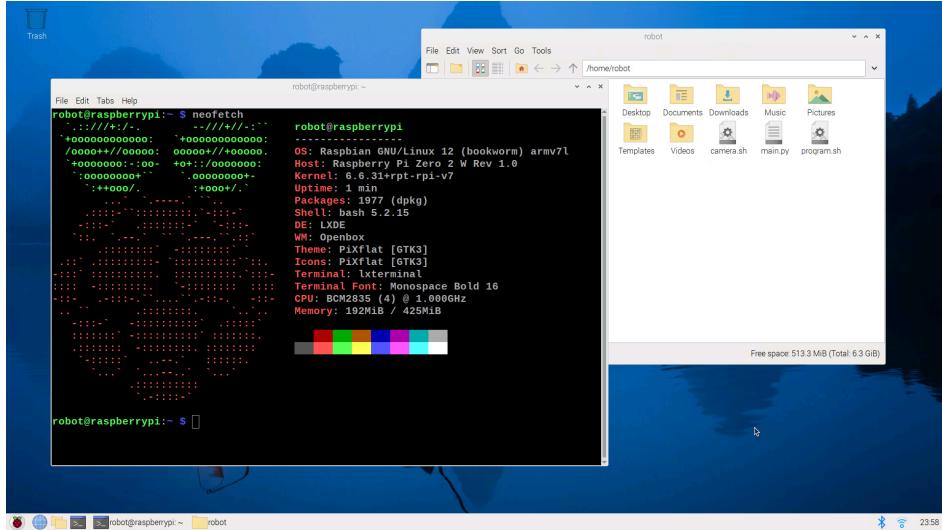
```

23
24     Serial.print('C');
25     Serial.print(' '); Serial.print(SensorValue_MQ2);
26     Serial.print(' '); Serial.print(SensorValue_MQ3);
27     Serial.print(' '); Serial.print(SensorValue_MQ4);
28     Serial.print(' '); Serial.print(SensorValue_MQ5);
29     Serial.print(' '); Serial.print(SensorValue_MQ6);
30     Serial.print(' '); Serial.print(SensorValue_MQ7);
31     Serial.print(' '); Serial.print(SensorValue_MQ8);
32     Serial.print(' '); Serial.print(SensorValue_MQ9);
33     Serial.println();
34 }
35
36 // [ ... ]
37 }
38
39 void displaySensorValues()
40 {
41     Leds.setPixelColor(0, convertPercentToColor(SensorValue_MQ2));
42     Leds.setPixelColor(1, convertPercentToColor(SensorValue_MQ3));
43     Leds.setPixelColor(2, convertPercentToColor(SensorValue_MQ4));
44     Leds.setPixelColor(3, convertPercentToColor(SensorValue_MQ5));
45     Leds.setPixelColor(4, convertPercentToColor(SensorValue_MQ6));
46     Leds.setPixelColor(5, convertPercentToColor(SensorValue_MQ7));
47     Leds.setPixelColor(6, convertPercentToColor(SensorValue_MQ8));
48     Leds.setPixelColor(7, convertPercentToColor(SensorValue_MQ9));
49
50     Leds.setPixelColor(8, convertPercentToColor(
51         SensorValue_DistanceFront, true));
52     Leds.setPixelColor(9, convertPercentToColor(
53         SensorValue_DistanceBack, true));
54
55     Leds.setPixelColor(10, convertPercentToColor(
56         SensorValue_FireFront));
57     Leds.setPixelColor(11, convertPercentToColor(
58         SensorValue_FireBack));
59
60     Leds.show();
61 }
```

Prog. 2: Odczytywanie wartości sensorów i przesyłanie wartości do aplikacji

5.2 Konfiguracja i oprogramowanie mikrokomputera

Kolejnym krokiem wykonanym do poprawnego oprogramowania robota była konfiguracja systemu na mikrokomputerze. W tym celu na karcie microSD zainstalowano system Linux, a dokładniej Raspberry Pi OS (znany również jako Raspbian) i ustawił go odpowiednio pod wymagania robota (rys. 24). Następnie zainstalowano wymagane programy, a także napisano program do obsługi podłączonych modułów.



Rys. 24: System Raspberry Pi OS w trybie graficznym, źródło: opracowanie własne

5.2.1 Konfiguracja systemu

Na początku przeprowadzono aktualizację wszystkich pakietów polecienniem `apt-get`, zainstalowano wymagane programy i skonfigurowano podstawowe parametry za pomocą wbudowanego narzędzia `raspi-config`. Moduł WiFi ustawił tak, aby po uruchomianiu systemu tworzył Hotspot WiFi, do którego można podłączyć się z telefonu, co pozwoliło na zastosowanie statycznego adresu IP, a to z kolei usprawniło proces pisania aplikacji mobilnej. Ze względu na wcześniejsze wykorzystanie zewnętrznego modułu Bluetooth podłączonego bezpośrednio do Arduino w celu minimalizowania opóźnień, ten znajdujący się w RapsberryPi został wyłączony. Następnie stworzono skrypt, wykonujący się przy starcie systemu, który uruchamia transmisję z kamery na żywo za pomocą programu `libcamera-vid`, zainstalowanego domyślnie w systemie. Ustawiono też w konfiguracji systemu piny odpowiedzialne za bezpieczne wyłączenie systemu, tak aby można było wyłączyć RaspberryPi za pomocą fizycznego przycisku znajdującego się na ramie robota. Po wykonaniu wszystkich prac konfiguracyjnych zmieniono tryb uruchamiania systemu z trybu pulpitu (desktop) na tryb konsoli (console), by zmniejszyć zużycie zasobów.

5.2.2 Sterowanie ramieniem

Na mikrokomputer RaspberryPi Zero 2W napisano program odczytujący wiadomości przychodzące z Arduino Nano, które następnie są przetwarzane podobnie jak to miało miejsce w przypadku mikrokontrolera. Program zawiera główną metodę `main` z nieskończoną pętlą `while`, w której odbywa się komunikacja i poruszanie uchwytnami służącymi do podnoszenia i upuszczania przedmiotów. Do tego celu zdefiniowano tabelę ośmiu kroków, przez które silnik kolejno musi przejść, aby wykonać ruch. W zależności czy chcemy poruszyć tylko prawym, tylko lewym, czy obydwoma ramionami, ustawiany jest odpowiednio kierunek obrotu, a następnie silnik wykonuje zadany ruch, przechodząc przez kolejne osiem kroków. Wszystkie te operacje pokazano w poniższym fragmencie kodu (prog. 3):

```
1 step_sequence = [[1,0,0,1], [1,0,0,0], [1,1,0,0], [0,1,0,0],
                   [0,1,1,0], [0,0,1,0], [0,0,1,1], [0,0,0,1]]
2
3 while True:
4     if ser.in_waiting > 0:
5         command = ser.readline().decode().strip()
6         if command[0] == 'B':
7             if command[2] == 'L':
8                 if command[4] == 'C':
9                     move_a = 1
10                move_b = 0
11            if command[4] == 'O':
12                move_a = -1
13                move_b = 0
14            if command[2] == 'R':
15                if command[4] == 'C':
16                    move_a = 0
17                    move_b = -1
18                if command[4] == 'O':
19                    move_a = 0
20                    move_b = 1
21            if command[2] == 'B':
22                if command[4] == 'C':
23                    move_a = 1
24                    move_b = -1
25                if command[4] == 'O':
26                    move_a = -1
27                    move_b = 1
28
```

```

29
30     for i in range(8):
31         motor_step_counter_a = (motor_step_counter_a + move_a) % 8
32         motor_step_counter_b = (motor_step_counter_b + move_b) % 8
33         time.sleep(0.005)
34         for pin in range(4):
35             GPIO.output(motor_pins_a[pin], step_sequence[
36                             motor_step_counter_a][pin])
37             GPIO.output(motor_pins_b[pin], step_sequence[
38                             motor_step_counter_b][pin])

```

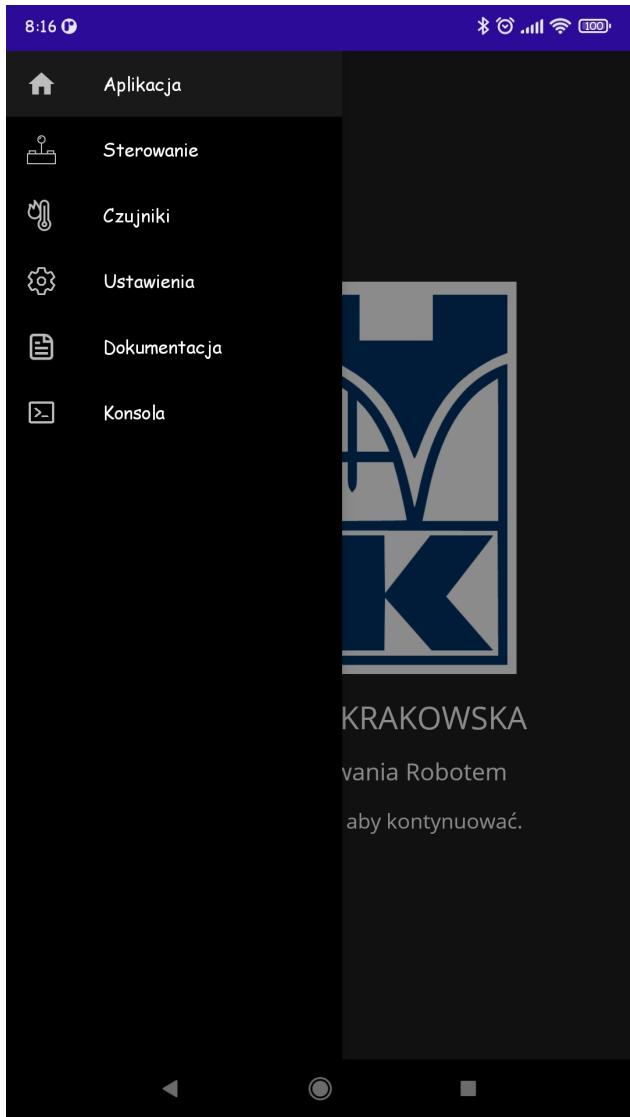
Prog. 3: Sterowanie uchytwami ramienia

5.3 Aplikacja mobilna

Ostatnim krokiem w celu oprogramowania robota było zaprojektowanie aplikacji mobilnej. W tym celu wybrano język C# oraz technologię MAUI (ang. Multi Platform App UI). Jest to zestaw narzędzi pozwalających na budowanie aplikacji wielo-platformowych dla systemów takich jak Android, iOS, Windows czy MacOS, będący ewolucją platformy Xamarin [33], który pozwolił na stworzenie aplikacji na smartfon z systemem Android.

5.3.1 Ogólny opis aplikacji

Aplikacja zawiera stronę główną wyświetlającą się zaraz po uruchomieniu oraz pięć podstron: Sterowanie, Czujniki, Ustawienia, Dokumentacja i Konsola (rys. 25). W celu szybkiego przełączania pomiędzy stronami dodano rozwijane menu z lewej strony ekranu, a przy każdej stronie umieszczono ikonkę, w celu zwiększenia intuicyjności aplikacji. Wygląd każdej ze stron zdefiniowany został w plikach z rozszerzeniem `xaml`, który jest rozszerzeniem formatu `xml`, a kod dla każdej ze stron został umieszczony w plikach o takich samych nazwach z rozszerzeniem `cs`. Dodatkowo wykorzystane zostały pakiety NuGet, czyli biblioteki rozszerzające funkcjonalność aplikacji. Jednym z pakietów jest plugin Bluetooth LE [34] służący do zarządzania połączeniami z urządzeniami typu Bluetooth Low Energy oraz transmisją danych pomiędzy nimi. Kolejnym pakietem jest biblioteka LibVLCSharp [35] pozwalająca na odtwarzanie na żywo transmisji wideo, wykorzystana w projekcie do wyświetlania obrazu z kamery. Całość została zaprojektowana z użyciem popularnego obecnie ciemnego motywu.



Rys. 25: Główne menu aplikacji, źródło: opracowanie własne

5.3.2 Ustawienia aplikacji

Aby ułatwić pracę aplikacji z robotem utworzono stronę z ustawieniami (rys. 26), gdzie można zobaczyć, do jakiej sieci WiFi jesteśmy obecnie podłączeni, a po przeskanowaniu wybrać z listy urządzenie Bluetooth, z którym połączy się smartphone. Można tutaj także ustawić parametry takie jak adres IP urządzenia oraz port, na którym transmitowany jest obraz z kamery (domyślnie jest to adres 192.168.0.1 oraz port 5000). Wybrać można również metodę transmisji obrazu z kamery, do wyboru są protokoły UDP (ang. User Datagram Protocol - protokół bezpołączeniowy), TCP (ang. Transmission Control Protocol - protokół połączeniowy - domyślnie wybrany) oraz RTSP (ang. Real-Time Streaming Protocol). Ostatnią wartością możliwą do skonfigurowania jest prędkość poruszania się robota (wolno, normalnie (wybrane domyślnie) lub szybko). W zależności

od wybranego ustawienia wartość wypełnienia sygnału PWM wynosi odpowiednio 50, 100 lub 150 (gdzie skala sygnału PWM wynosi od 0 do 255). Po poprawnym skonfigurowaniu aplikacji (rys. 26), tj. wybraniu odpowiedniego urządzenia (oraz opcjonalnie zmiany adresu IP oraz portu) można rozpocząć pracę z robotem przechodząc na stronę sterowania lub danych z czujników. Cały wygląd strony zaprojektowany jest w dokumencie widocznym poniżej (prog. 4):

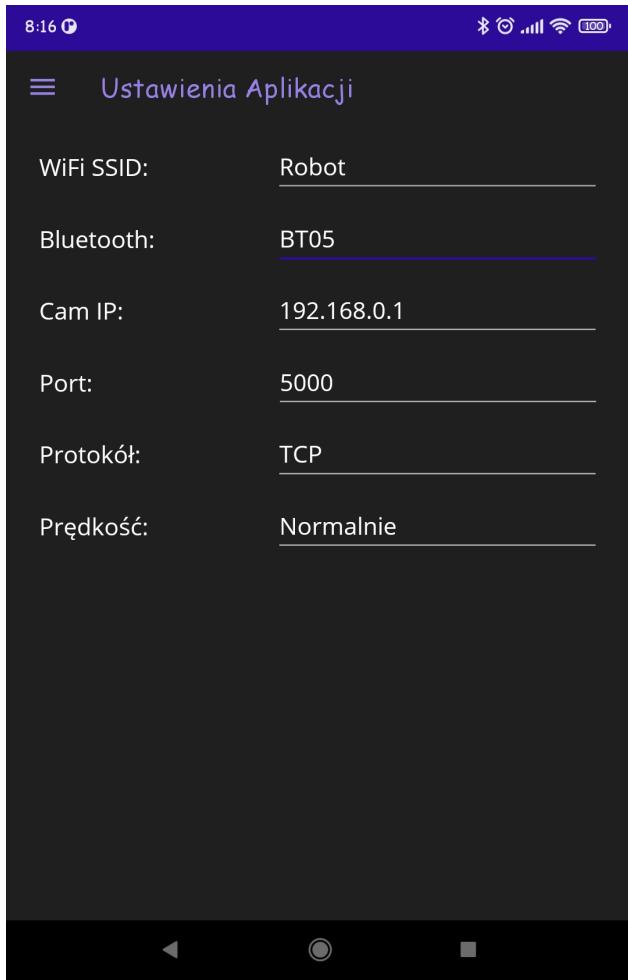
```
1  <?xml version="1.0" encoding="utf-8" ?>
2
3  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
4      xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
5      x:Class="Mobile.SettingsPage"
6      Title="Ustawienia Aplikacji">
7      <StackLayout Margin="25,10,25,10" Spacing="10">
8
9          <StackLayout Orientation="Horizontal">
10             <Label FontSize="Medium" x:Name="WifiText" Text="WiFi SSID:"/>
11             <Picker FontSize="Medium" x:Name="WifiValue" WidthRequest="250"
12                 SelectedIndexChanged="OptionChanged" SelectedItem="Robot"/>
13                 SelectedIndex="0"/>
14         </StackLayout>
15
16         <StackLayout Orientation="Horizontal">
17             <Label FontSize="Medium" x:Name="BlueText" Text="Bluetooth:"/>
18             <Picker FontSize="Medium" x:Name="BlueValue" WidthRequest="250"
19                 SelectedIndexChanged="OptionChanged" SelectedItem="BT05"/>
20                 SelectedIndex="0"/>
21         </StackLayout>
22
23         <StackLayout Orientation="Horizontal">
24             <Label FontSize="Medium" x:Name="AddrText" Text="Cam IP:"/>
25             <Entry FontSize="Medium" x:Name="AddrValue" WidthRequest="250"
26                 Text="192.168.1.10" HorizontalOptions="End" Keyboard="texttt"
27                 TextChanged="OptionChanged"/>
28         </StackLayout>
29
30         <StackLayout Orientation="Horizontal">
31             <Label FontSize="Medium" x:Name="PortText" Text="Port:"/>
32             <Entry FontSize="Medium" x:Name="PortValue" WidthRequest="250"
33                 Text="8080" HorizontalOptions="End" Keyboard="Text"/>
34         </StackLayout>
35     </StackLayout>
36 
```

```

        Text="5000" HorizontalOptions="End" Keyboard="Numeric"
        TextChanged="OptionChanged"/>
27    </StackLayout>
28
29    <StackLayout Orientation="Horizontal">
30        <Label FontSize="Medium" x:Name="ProtoText" Text="Protokol:"
            HorizontalOptions="StartAndExpand" Margin="0,10,0,0"/>
31        <Picker FontSize="Medium" x:Name="ProtoValue" WidthRequest="
                250" SelectedIndexChanged="OptionChanged" SelectedItem="TCP
                " SelectedIndex="1">
32            <Picker.ItemsSource>
33                <x:Array Type="{x:Type x:String}">
34                    <x:String>UDP</x:String>
35                    <x:String>TCP</x:String>
36                    <x:String>RTSP</x:String>
37                </x:Array>
38            </Picker.ItemsSource>
39        </Picker>
40    </StackLayout>
41
42    <StackLayout Orientation="Horizontal">
43        <Label FontSize="Medium" x:Name="SpeedText" Text="Predkosc:"
            HorizontalOptions="StartAndExpand" Margin="0,10,0,0"/>
44        <Picker FontSize="Medium" x:Name="SpeedValue" WidthRequest="
                250" SelectedIndexChanged="OptionChanged" SelectedItem="
                Normalnie" SelectedIndex="1">
45            <Picker.ItemsSource>
46                <x:Array Type="{x:Type x:String}">
47                    <x:String>Wolno</x:String>
48                    <x:String>Normalnie</x:String>
49                    <x:String>Szybko</x:String>
50                </x:Array>
51            </Picker.ItemsSource>
52        </Picker>
53    </StackLayout>
54
55    </StackLayout>
56 </ContentPage>

```

Prog. 4: Strona z ustawieniami aplikacji



Rys. 26: Strona z ustawieniami aplikacji, źródło: opracowanie własne

5.3.3 Sterowanie robotem

Do sterowania robotem utworzono stronę z kontrolerem, w jej górnej części wyświetla się obraz z kamery, a poniżej umieszczono przyciski do sterowania silnikami robota (rys. 27). Przyciski umieszczone na środku ekranu, tworzące tzw. D-PAD (ang. Directional Pad), czyli kontroler z przyciskami ułożonymi w kształt plusa (+) służą do przemieszczania się do przodu i do tyłu, oraz do skręcania bądź zawracania w miejscu. Aby zapobiec niekontrolowanemu poruszaniu się robota w przypadku np. utraty połączenia zastosowano limit czasu, tzw. timeout, a wartości przycisków przesyłane są w sposób ciągły, co kilkanaście milisekund, dzięki czemu można wykryć utratę transmisji. Na dole ekranu (rys. 27) umieszczono przyciski służące do poruszania ramieniem, które zgodnie z ikonami zamkają lub otwierają uchwyty robota. Używając środkowych przycisków można sterować obydwooma ramionami jednocześnie, a gdyby była taka potrzeba można również używając przycisków z lewej lub prawej strony sterować odpowiednio tylko lewym lub prawym ramieniem. Przesyłanie danych zobrazowano na poniższym programie (prog. 5):

```

1 public async static void SerialLoop(IDevice device)
2 {
3     // [ ... ]
4
5     bool nowMoving = false;
6     bool wasMoving = false;
7     int wasTime = -1;
8     int nowTime = -1;
9
10    while (loop)
11    {
12        // [ ... ]
13
14        if (wasTime != nowTime)
15        {
16            wasTime = nowTime;
17            nowMoving = false;
18
19            if (ControlPage.ButtonState_MoveForward)
20            {
21                nowMoving = true;
22                wasMoving = true;
23                await charact.WriteAsync(Encoding.ASCII.GetBytes("A F " +
24                                         SettingsPage.Speeds[SettingsPage.Speed] + '\n'));
25            }
26
27            // [ ... ]
28
29            if (!nowMoving && wasMoving)
30            {
31                wasMoving = false;
32                await charact.WriteAsync(Encoding.ASCII.GetBytes
33                                         ("A O 0" + '\n'));
34            }
35        }
36    }
37 }

```

Prog. 5: Przesyłanie instrukcji do robota



Rys. 27: Strona z podglądem kamery oraz sterowaniem, źródło: opracowanie własne

5.3.4 Dane z czujników

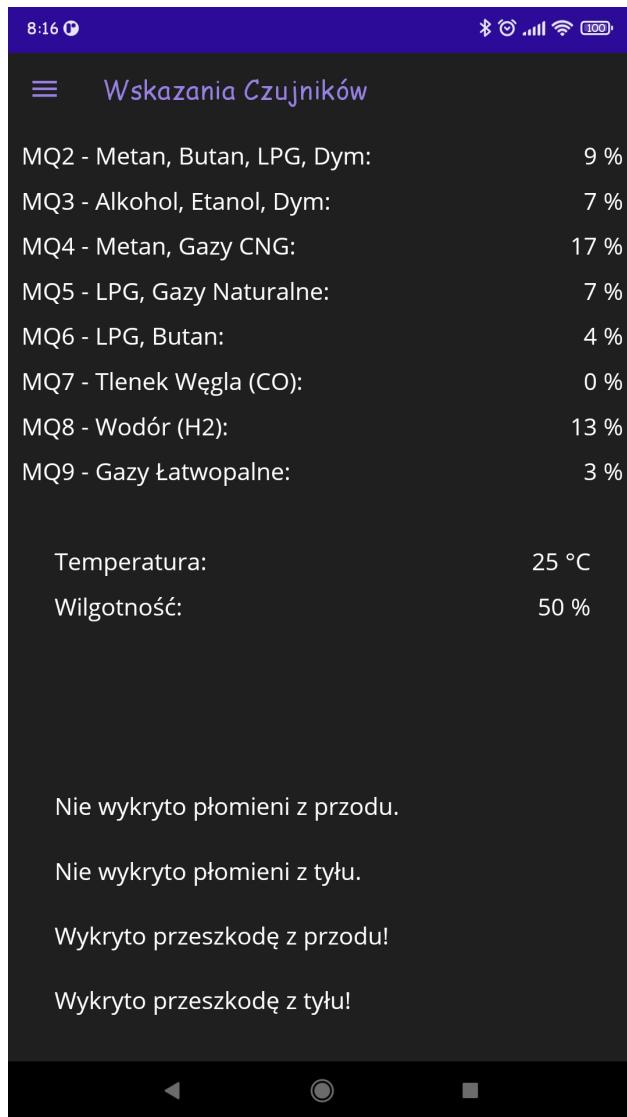
Do odczytywania stanu czujników zamontowanych w robocie utworzono kolejną stronę w aplikacji (rys. 28). Na górze strony znajduje się lista wszystkich ośmiu czujników, wraz z wykrywanymi substancjami. Przy każdym sensorze znajduje się informacja, ile wykrył danej substancji w powietrzu w wartościach procentowych. Ważną informacją jest tutaj to, że wartość nie oznacza wartości procentowej gazu w powietrzu, lecz wartością procentową zakresu działania czujnika. Przykładowo, dla sensora z zakresem od 0 do 500 PPM (ang. Parts Per Milion), wartość procentowa dla 250 PPM wynosi 50%, co przekłada się na 0.025% gazu w powietrzu. Można to przeliczyć wg. wzoru, który mówi że $1\text{PPM} = 0.0001\%$, co dla wartości podanych w przykładzie daje następujący wynik: $250 * 0.0001 = 0.025$. Pod informacjami z czujników powietrza

umieszczone dane z czujnika temperatury i wilgotności, dla którego wartości wyświetla się odpowiednio w stopniach Celsjusza dla temperatury oraz procentowej wartości dla wilgotności. Na dole ekranu (rys. 28) umieszczone informacje o wykrytych przeszkodach i płomieniach. W przypadku sensora odległości, informacja o wykrytej przeszkodzie pojawia się w przypadku wykrycia obiektu w odległości mniejszej niż 50 cm, a w przypadku czujnika płomieni informacja o wykrytym zagrożeniu pojawia się w przypadku, gdy sensor zwróci wartość powyżej 50%. Przesyłanie danych z czujników przedstawiono w poniższym programie (prog. 6):

```
1 public async static void SerialLoop(IDevice device)
2 {
3     var ble = CrossBluetoothLE.Current;
4     var adapter = CrossBluetoothLE.Current.Adapter;
5     await adapter.ConnectToDeviceAsync(device);
6     var service = await device.GetServiceAsync(Guid.Parse(guidS));
7     var charact = await service.GetCharacteristicAsync(Guid.Parse(guidC));
8
9     // [ ... ]
10
11    while (loop)
12    {
13        var stream = (await charact.ReadAsync()).data;
14        var commands = Encoding.ASCII.GetString(stream).Split('\n');
15
16        foreach (string command in commands)
17        {
18            if(command.Length > 1)
19            {
20                ConsolePage.LogMessage(command);
21                string[] data = command.Split(' ');
22
23                if (command[0] == 'C' && data.Length > 8)
24                {
25                    int.TryParse(data[1], out SensorsPage.SensorValue_MQ2);
26                    int.TryParse(data[2], out SensorsPage.SensorValue_MQ3);
27                    int.TryParse(data[3], out SensorsPage.SensorValue_MQ4);
28                    int.TryParse(data[4], out SensorsPage.SensorValue_MQ5);
29                    int.TryParse(data[5], out SensorsPage.SensorValue_MQ6);
30                    int.TryParse(data[6], out SensorsPage.SensorValue_MQ7);
31                    int.TryParse(data[7], out SensorsPage.SensorValue_MQ8);
32                    int.TryParse(data[8], out SensorsPage.SensorValue_MQ9);
33                }
34            }
35        }
36    }
37}
```

```
34
35         }
36
37         // [ ... ]
38     }
39 }
40
41     // [ ... ]
42 }
43 }
```

Prog. 6: Odczytywanie danych z czujników



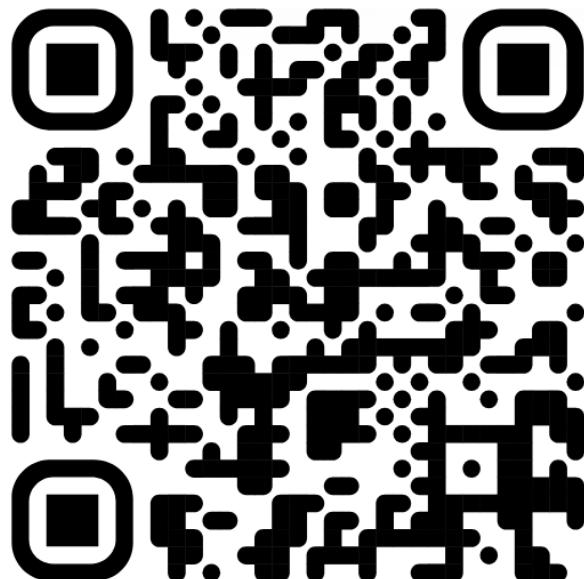
Rys. 28: Strona z danymi ze wszystkich czujników, źródło: opracowanie własne

5.3.5 Pozostałe strony

W aplikacji dodano też strony takie jak Konsola czy Dokumentacja. Zostały one dodane w celu ułatwienia pisania aplikacji i programów, a także stanowiły one dużą pomoc przy testowaniu poprawności działania robota. Strona z konsolą zawiera wszystkie wiadomości przychodzące i wychodzące podczas komunikacji pomiędzy smartfonem a robotem, razem z datą i godziną wysłania lub odebrania wiadomości. Z kolei strona z dokumentacją zawiera informacje techniczne dotyczące robota, a także użytych sensorów i modułów. Do wszystkich stron, włącznie ze stroną główną aplikacji można przejść bezpośrednio w dowolnym momencie wybierając odpowiedni przycisk z menu bocznego.

5.4 Kod źródłowy

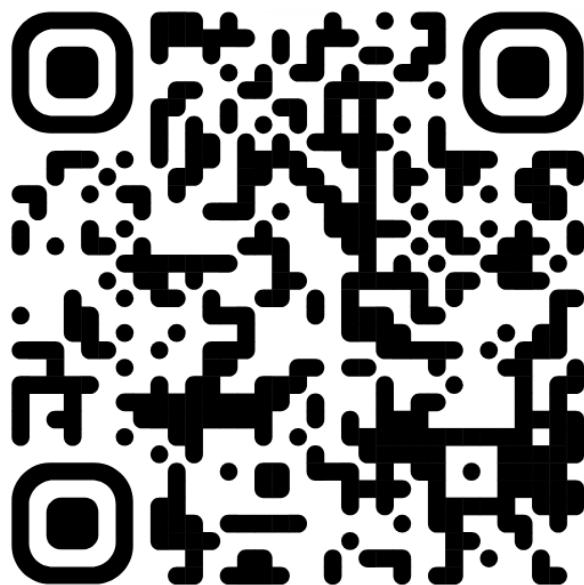
Cały kod źródłowy, zarówno programu na mikrokontroler napisanego w języku C++ jak i programu na mikrokomputer napisanego w języku Python umieszczono w repozytorium w serwisie Github. Znajduje się tam również kod źródłowy całej aplikacji mobilnej napisanej w języku C#, a także pliki XAML definiujące wygląd aplikacji. Dodatkowo umieszczono tam również schematy połączeń głównych modułów oraz wykorzystanych czujników i sensorów, a także wszystkie zaprojektowane modele obiektów. Repozytorium dostępne jest pod adresem <https://github.com/TheQffel/Robot> lub po zeskanowaniu poniższego kodu QR (rys. 29):



Rys. 29: Kod QR z linkiem do repozytorium, źródło: opracowanie własne

5.5 Działanie robota

Ostatnim wykonanym w projekcie krokiem było przetestowanie wszystkich funkcji robota. Testowi poddany został układ napędowy oraz ramię robota, a także wszystkie zamontowane sensory. Do przetestowania czujników gazu użyto płynu do czyszczenia na bazie alkoholu, co spowodowało wykrycie tej substancji poprzez niektóre z sensorów. Z kolei czujniki odległości i płomieni przetestowano używając powerbank'a z podłączonym mocnym panelem LED. Zbliżając go na odpowiednią odległość i włączając lub wyłączać zasilanie w wyżej wymienionym źródle światła, sensor odległości raportował wykrycie przeszkode, a czujnik ognia informował o wykryciu płomieni. Wszystkie przeprowadzone testy zostały udokumentowane w postaci filmu, który można obejrzeć w serwisie YouTube pod adresem <https://youtu.be/u5CSH7bqYWo> lub po zeskanowaniu poniższego kodu QR (rys. 30):



Rys. 30: Kod QR z linkiem do filmu, źródło: opracowanie własne

6 Podsumowanie

W dobie dynamicznego rozwoju technologii komputerowych i robotyki, stworzenie robota do zadań specjalnych stało się możliwe. Opisany robot, zaprojektowany z myślą o bezpieczeństwie, posiada szeroki wachlarz zastosowań dzięki zastosowaniu licznych modułów i czujników.

6.1 Robot na tle innych rozwiązań

W porównaniu do istniejących na rynku rozwiązań, robot wyróżnia się elastycznością i możliwością łatwej adaptacji do specyficznych wymagań użytkownika. Przykładowo, wiele komercyjnie dostępnych urządzeń skupia się na jednej konkretnej funkcji, podczas gdy ten projekt oferuje szerokie spektrum zastosowań. Jest to możliwe głównie dzięki jego modułowej budowie i różnorodności zastosowanych czujników. Robot posiada bardzo szeroki wachlarz scenariuszy użytkowania. Po odpowiednim dostosowaniu i certyfikowaniu, urządzenie może być wykorzystywane w różnorodnych środowiskach, takich jak straż pożarna, służby ratunkowe, laboratoria badawcze, czy przemysł. W przyszłości, dzięki możliwości łatwej modyfikacji i rozbudowy, robot może być dostosowywany do nowych, nieprzewidzianych obecnie zastosowań.

6.2 Zrealizowane założone cele

Główym zadaniem urządzenia jest monitorowanie pomieszczeń, wykrywanie płomieni i niebezpiecznych substancji, a aplikacja umożliwia sterowanie robotem i podgląd obrazu z kamery. Robot znajduje zastosowanie w wielu sytuacjach, a jego zaawansowana technologia i modułowa konstrukcja czynią go wszechstronnym narzędziem o dużym potencjale adaptacyjnym. Warto podkreślić, że realizacja projektu przebiegła zgodnie z założonymi celami. Kluczowe było komplementarne podejście do projektu, obejmujące zebranie wymagań, analizę istniejących rozwiązań oraz wybór odpowiednich metod i technologii. Następnie zaprojektowane zostały odpowiednie elementy konstrukcyjne, a także napisane zostało dedykowane specjalistyczne oprogramowanie. Dzięki temu, wszystkie zaplanowane funkcjonalności zostały zrealizowane, a robot spełnia wszystkie określone wcześniej oczekiwania. Podsumowując, cały opisany projekt zakończył się sukcesem, a wszystkie jego cele zostały pomyślnie zrealizowane. Dzięki szerokiemu wachlarzowi zastosowań oraz modułowej konstrukcji, robot jest dobrą bazą do dalszych prac i rozwoju w wielu kierunkach, takich jak robotyka, automatyka, czy szeroko pojęta inżynieria.

Spis rysunków

1	Wymagania funkcjonalne przedstawione w SysML, źródło: opracowanie własne	8
2	Wymagania niefunkcjonalne przedstawione w SysML, źródło: opracowanie własne	8
3	Shakey, jeden z pierwszych robotów mobilnych, źródło: [9]	9
4	Sojourner, pierwszy łazik marsjański opracowany przez NASA, źródło: [11]	10
5	Spot, robot stworzony przez Boston Dynamics, źródło: [13]	11
6	Dron firmy Dji, sterowany pilotem, źródło: [14]	12
7	Kerfuś, robot do zabawiania klientów, maskotka sklepu, źródło: [17]	14
8	Okno programu Blender z zaprojektowaną ramą robota, źródło: opracowanie własne	16
9	Okno programu CURA z elementem do druku, źródło: opracowanie własne	16
10	Diagram połączeń pomiędzy poszczególnymi elementami, źródło: opracowanie własne	17
11	Opis wyprowadzeń GPIO w Arduino Nano, źródło: [21]	18
12	Opis wyprowadzeń GPIO w Raspberry Pi Zero 2W, źródło: [23]	19
13	Czujnik odległości HC-SR04, źródło: [26]	21
14	Czujnik płomieni Waveshare 9521, źródło: [27]	22
15	Czujnik temperatury i wilgotności DHT11, źródło: [28]	22
16	Czujnik MQ2, jeden z ośmiu czujników powietrza, źródło: [29]	23
17	Adresowalne diody LED Adafruit Neopixel, źródło: opracowanie własne	24
18	Przykładowe sygnały wygenerowane za pomocą PWM, źródło: [31]	25
19	Koło z PLA i opona z TPU zamontowane w robocie, źródło: opracowanie własne	26
20	Ramiona robota trzymające taśmę izolacyjną, źródło: opracowanie własne	27
21	Bateria typu 18650 wykorzystana do zasilania układu, źródło: [32]	28
22	Robot po złożeniu w całość, źródło: opracowanie własne	29
23	Program Visual Studio Code z dodatkiem Platform IO, źródło: opracowanie własne	30
24	System Raspberry Pi OS w trybie graficznym, źródło: opracowanie własne	35
25	Główne menu aplikacji, źródło: opracowanie własne	38
26	Strona z ustawieniami aplikacji, źródło: opracowanie własne	41
27	Strona z podaniem kamery oraz sterowaniem, źródło: opracowanie własne	43
28	Strona z danymi ze wszystkich czujników, źródło: opracowanie własne	45
29	Kod QR z linkiem do repozytorium, źródło: opracowanie własne	46
30	Kod QR z linkiem do filmu, źródło: opracowanie własne	47

Spis tabel

1	Specyfikacja mikrokontrolera, źródło: [20]	18
2	Specyfikacja mikrokomputera i kamery, źródło: [22]	19
3	Czujniki MQ i wykrywane przez nie gazy, źródło: opracowanie własne	23

Spis kodów programów

1	Odczytywanie poleceń i poruszanie robotem	31
2	Odczytywanie wartości sensorów i przesyłanie wartości do aplikacji	33
3	Sterowanie uchytwami ramienia	36
4	Strona z ustawieniami aplikacji	39
5	Przesyłanie instrukcji do robota	42
6	Odczytywanie danych z czujników	44

Bibliografia

- [1] Roland Siegwart, Illah R. Nourbakhsh, Davide Scaramuzza,
"Introduction to Autonomous Mobile Robots", The MIT Press, 2011
- [2] Hod Lipson, Melba Kurman, "Fabricated: The New World of 3D Printing",
John Wiley & Sons, 2013
- [3] Arshdeep Bahga, Vijay Madisetti, "Internet of Things: A Hands-On Approach",
VPT (Vijay Madisetti Publishing), 2014
- [4] Maja J Mataric, "The Robotics Primer", The MIT Press, 2007
- [5] Roger Wiens, "Red Rover: Inside the Story of Robotic Space Exploration,
from Genesis to the Mars Rover Curiosity", Basic Books, 2013
- [6] Sanford Friedenthal, Alan Moore, and Rick Steiner, "A Practical Guide to SysML:
The Systems Modeling Language", Morgan Kaufmann, 2008
- [7] "What is the Systems Modeling Language (SysML)?", Data dostępu: 1 lipiec 2024
<https://sysml.org/sysml-faq/what-is-sysml.html>
- [8] Aleksandra Szczepaniak, "What was the world's first mobile intelligent robot?", 9 maj 2023
<https://www.leorover.tech/post/what-was-the-worlds-first-mobile-intelligent-robot>
- [9] "Shakey the robot", Data dostępu: 1 lipiec 2024
<https://www.researchgate.net/profile/Konstantinos-Zikidis/publication/259503287/figure/fig2/AS:667639546388486@1536189080881/Shakey-the-robot-Probably-the-first-mobile-robot-It-was-developed-at-the-Artificial.png>
- [10] Stuart Atkinson, "Sojourner: NASA's first Mars rover", 6 lipiec 2023
<https://www.astronomy.com/space-exploration/sojourner-nasas-first-mars-rover/>
- [11] "Sojourner rover", Data dostępu: 1 lipiec 2024
https://www.astronomy.com/wp-content/uploads/sites/2/2023/07/ASY-SA0723_07-Sojourner-rover.png
- [12] "Spot - Boston Dynamics", Data dostępu: 1 lipiec 2024
<https://bostondynamics.com/products/spot/>

- [13] "Spot Explorer", Data dostępu: 1 lipiec 2024
<https://bostondynamics.com/wp-content/uploads/2023/05/spot-explorer-web-2.jpg>
- [14] "Dji Mini 3 Pro Smart Control", Data dostępu: 1 lipiec 2024
https://m.media-amazon.com/images/I/61Y1P6uIRFL._AC_UF894,1000_QL80_.jpg
- [15] "Real Time Operating System (RTOS)", Data dostępu: 1 lipiec 2024
<https://www.geeksforgeeks.org/real-time-operating-system-rtos/>
- [16] "ROS: Why ROS?", Data dostępu: 1 lipiec 2024
<https://www.ros.org/blog/why-ros/>
- [17] "Zdjęcie Kerfusia - Wikimedia", Data dostępu: 1 lipiec 2024
https://upload.wikimedia.org/wikipedia/commons/thumb/c/cd/Zdj%C4%99cie_Kerfusia_%28crop%29.jpg/800px-Zdj%C4%99cie_Kerfusia_%28crop%29.jpg
- [18] Sandra Marcinkowska, "PLA – Co to jest? Właściwości, zastosowanie, materiał", 13 maj 2023
<https://botland.com.pl/blog/pla-co-to-jest-wlasciwosci-zastosowanie-material/>
- [19] Sandra Marcinkowska, "Filament TPU – Czym jest i do czego służy?", 26 czerwiec 2024
<https://botland.com.pl/blog/filament-tpu-czym-jest-i-do-czego-sluzy/>
- [20] "Nano - Arduino Documentation", Data dostępu: 1 lipiec 2024
<https://docs.arduino.cc/hardware/nano/#tech-specs>
- [21] "Arduino Nano Pinout", Data dostępu: 1 lipiec 2024
<https://mischianti.org/wp-content/uploads/2023/11/Pinout-Arduino-Nano-low-resolution-1024x834.jpg>
- [22] "Raspberry Pi Zero 2 W - Raspberry Pi", Data dostępu: 1 lipiec 2024
<https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/>
- [23] "Raspberry Pi Zero Pinout", Data dostępu: 1 lipiec 2024
<https://zbotic.in/wp-content/uploads/2024/02/rq9a6ohs.png>

- [24] "SPI - SparkFun Learn", Data dostępu: 1 lipiec 2024
<https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all>
- [25] "I2C - SparkFun Learn", Data dostępu: 1 lipiec 2024
<https://learn.sparkfun.com/tutorials/i2c/all>
- [26] "HC-SR04", Data dostępu: 1 lipiec 2024
<https://res.cloudinary.com/rsc/image/upload/Y2153181-01>
- [27] "Waveshare 9521", Data dostępu: 1 lipiec 2024
https://www.waveshare.com/media/catalog/product/cache/1/image/800x800/9df78eab33525d08d6e5fb8d27136e95/f/1/flame-sensor-3_2.jpg
- [28] "DHT11 Sensor", Data dostępu: 1 lipiec 2024
https://miro.medium.com/v2/resize:fit:1400/1*SSssR7f2xngjWAqZhF0RyQ.png
- [29] "MQ2 Gas Sensor", Data dostępu: 1 lipiec 2024
<https://components101.com/sites/default/files/components/MQ2-gas-sensor.jpg>
- [30] "PWM - SparkFun Learn", Data dostępu: 1 lipiec 2024
<https://learn.sparkfun.com/tutorials/pulse-width-modulation/all>
- [31] "Pulse Width Modulation", Data dostępu: 1 lipiec 2024
<https://docs.arduino.cc/54ef6da144b4531dd9ada686a7e67c56/pwm.gif>
- [32] "Akumulator Samsung INR 18650 20A", Data dostępu: 1 lipiec 2024
https://alltronix.pl/environment/cache/images/640_640_productGfx_4b1d1d2ad151af1281f2732632301324.webp
- [33] ".NET Multi-Platform App UI", Data dostępu: 1 lipiec 2024
<https://dotnet.microsoft.com/en-us/apps/maui>
- [34] "Bluetooth LE Plugin", Data dostępu: 1 lipiec 2024
<https://github.com/dotnet-bluetooth-le/dotnet-bluetooth-le>
- [35] "LibVLCSharp", Data dostępu: 1 lipiec 2024
<https://github.com/videolan/libvlcssharp>