POLITECHNIKA GDAŃSKA

WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

Dokumentacja Projektu grupowego

Dokumentacja techniczna projektu

Wydział Elektroniki, Telekomunikacji i Informatyki
Politechnika Gdańska

{wersja dokumentu wzorcowego: wersja 2/2023}

Nazwa i akronim projektu: *Autonomiczny dron analizująco-mapujacy (ADAM)*	**Zleceniodawca:** *dr inż. Krzysztof Gierłowski (opiekun PG)*	
Numer zlecenia: *2@KTIN'2024*	**Kierownik projektu:** *Kamil Ratajczyk*	**Opiekun projektu:** *dr inż. Krzysztof Gierłowski (opiekun PG)*

Nazwa / kod dokumentu: **Dokumentacja techniczna produktu – DTP**	**Nr wersji:** *2.00*
Odpowiedzialny za dokument: *Paweł Dolak Bartosz Twardowski*	**Data pierwszego sporządzenia:** *30.01.2025*
	Data ostatniej aktualizacji: *12.09.2025*
	Semestr realizacji Projektu grupowego: *2*

Historia dokumentu

Wersja	Opis modyfikacji	Rozdział / strona	Autor modyfikacji	Data
1.00	*Pierwsza wersja*	*całość*	*Paweł Dolak* *Bartosz Twardowski*	*30.01.2025*
2.00	*Aktualizacja po 2 semestrze*	*całość*	*Kamil Ratajczyk*	*12.09.2025*

Spis treści

1 Wprowadzenie - o dokumencie

1.1 Cel dokumentu

Celem dokumentu jest udokumentowanie informacji dotyczących produktu, jego cech funkcjonalnych, parametrów technicznych, schematów blokowych, oprogramowania, wyników działania, zdjęć produktu, pomiarów, testów oraz innych elementów wymaganych przez opiekuna i klienta.

1.2 Zakres dokumentu

Przegląd używanych komponentów, sposób połączenia ze sobą modułów elektronicznych, opis działania systemu, oprogramowania urządzeń i aplikacji.

1.3 Odbiorcy

Opiekun projektu: dr inż. Krzysztof Gierłowski

Członkowie zespołu projektowego:
- Kamil Ratajczyk
- Jakub Sawicki
- Paweł Dolak
- Bartosz Twardowski
- Jan Landecki

1.4 Terminologia

● ROS2 (Robot Operating System 2) – framework służący do projektowania i implementacji oprogramowania dla systemów robotycznych. Wspiera komunikację między komponentami robota oraz umożliwia tworzenie rozproszonych aplikacji.

● Nav2 – pakiet w środowisku ROS2, służący do planowania trasy i nawigacji autonomicznych robotów. Obejmuje funkcje takie jak wyznaczanie ścieżki, omijanie przeszkód oraz kontrolę ruchu.

● Biblioteka Cartographer – oprogramowanie do mapowania i lokalizacji w czasie rzeczywistym (SLAM). Umożliwia budowanie map środowiska oraz lokalizowanie pozycji robota na podstawie danych z czujników, takich jak LIDAR.

● SLAM (Simultaneous Localization and Mapping) – jednoczesna lokalizacja i mapowanie. Proces, w którym robot jednocześnie tworzy mapę swojego otoczenia i określa swoje położenie w tej mapie.

● Napęd różnicowy – rodzaj napędu, w którym każde koło pojazdu może być niezależnie sterowane, co pozwala na manewrowanie przez różnicowanie prędkości obrotowej kół. Jest to powszechnie stosowane w robotyce mobilnej.

● Projektowanie CAD – proces projektowania obiektów trójwymiarowych przy użyciu komputerowych narzędzi wspomagających projektowanie, takich jak AutoCAD, Fusion 360 czy SolidWorks.

● Druk 3D – proces tworzenia fizycznych obiektów na podstawie projektów CAD poprzez nakładanie kolejnych warstw materiału, co pozwala na szybkie prototypowanie i produkcję niestandardowych części.

● LIDAR (Light Detection and Ranging) – czujnik wykorzystujący wiązkę światła laserowego do określania odległości do obiektów i tworzenia map otoczenia w czasie rzeczywistym.

● Raspberry Pi - komputer jednopłytkowy o niewielkich rozmiarach, zaprojektowany w celu wspierania edukacji informatycznej i różnorodnych projektów DIY (“zrób to sam”).

● Nakładki do Raspberry Pi - Nakładki (z ang. "Hats" - Hardware Attached on Top) to specjalne płytki rozszerzeń, które można montować bezpośrednio na GPIO (General Purpose In/Out) na płycie Raspberry Pi. Nakładki te dodają różne funkcje i możliwości do standardowego zestawu, zawierają między innymi: sensory, moduły radiowe lub inne elementy.

● Silnik DC - silnik prądu stałego przetwarzający energię elektryczną na ruch obrotowy, wykorzystywany do napędzania kół drona naziemnego lub innych ruchomych elementów. Umożliwia precyzyjną kontrolę prędkości i kierunku obrotu.

● Platforma - mobilna platforma kołowa do poruszania się po powierzchniach lądowych. Stanowi bazę robota, na której montowane są kluczowe komponenty, takie jak silniki, czujniki i jednostka sterująca.

2 Dokumentacja techniczna projektu

2.1 Napęd różnicowy platformy

Do napędu platformy wykorzystano cztery silniki szczotkowe prądu stałego. Każdy z silników napędza jedno z kół platformy poprzez odpowiedni wał. Silniki są sterowane przez ESP32 z wykorzystaniem dwóch dwukanałowych sterowników do silników DC. Całość zasilana jest odpowiednią baterią LiPo.

Podzespoły systemu napędowego:
● Silniki: 4 × FIT0186 DFROBOT – silniki prądu stałego z przekładnią zapewniające odpowiedni moment obrotowy do napędu pojazdu. Datasheet w folderze ./Datasheets na repozytorium

● Sterowniki silników: 2 × DRI0041 DFROBOT – moduły odpowiedzialne za sterowanie pracą silników, umożliwiające regulację prędkości oraz kierunku obrotów. Datasheet w folderze ./Datasheets na repozytorium.

● Kontroler: ESP32 – jednostka centralna zarządzająca kontrolująca napęd i zbierająca dane z enkoderów.

● Zasilanie: Akumulator LiPo 6500 mAh 11.1V 60C – źródło energii zapewniające odpowiednią moc do zasilania całego systemu napędowego.

Zasada działania:

Napęd różnicowy pozwala na sterowanie kierunkiem ruchu pojazdu poprzez różnicowanie prędkości obrotowej kół po obu stronach. Przy skręcie jedno koło obraca się szybciej lub w przeciwnym kierunku niż drugie, co umożliwia zmianę kierunku jazdy bez konieczności stosowania mechanizmu skrętu osi.

Sterowniki silników odbierają sygnały sterujące z kontrolera ESP32 i dostosowują prędkość oraz kierunek obrotów poszczególnych silników, co umożliwia precyzyjne manewrowanie. Zastosowanie czterech silników zapewnia lepszą trakcję i stabilność pojazdu.

2.2 ROS2

W framework ROS2 różne elementy programu są podzielone na tak zwane węzły, które się ze sobą komunikują poprzez wysyłanie i odbieranie danych na tak zwane tematy.

Dzięki takiemu mechanizmowi można podzielić elementy programu takie jak np:
- węzeł odbierający dane z lidaru i wysyłający je na dany temat
- węzeł, który tworzy mapę z danych z lidaru
- węzeł od wysyłania danych sterujących do silników
- węzeł odbierający te dane i wysyłające je przez interfejs szeregowy do ESP32
i wiele innych

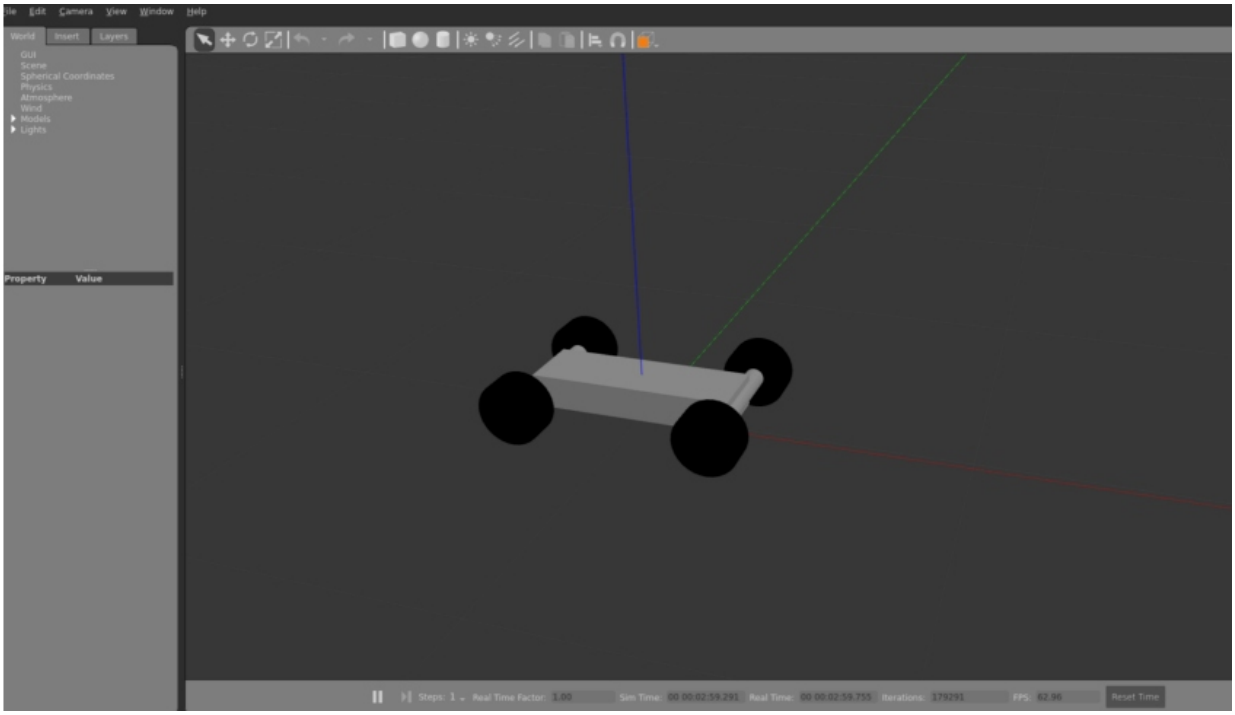
Dzięki takiemu podziałowi można odseparować od siebie różne elementy i z łatwością podmieniać jeden element na drugi, wystarczy tylko że wysyła dane na ten sam temat który zastępuje.

Stąd wziął się pomysł, aby odseparować algorytmy mapujące, nawigujące i sterujące od fizycznej budowli robota. Robot, jego silniki, czujniki i jego otoczenie może być symulowane, co pozwala na łatwiejsze pisanie i testowanie algorytmów.

2.3 Model robota

Kluczowym założeniem projektu było stworzenie symulacji robota, aby móc pisać program oraz testować algorytmy bez gotowej platformy.

Zostało zdecydowane, że do symulacji zostanie wykorzystany program Gazebo, ze względu na to, że dobrze współgra z frameworkiem ROS2. Aby symulować robota w Gazebo, potrzebny jest jego opis w formacie URDF (Unified Robot Description Format), który jest formatem XML. W formacie tym robot jest opisywany za pomocą elementów “link” i “joint”.

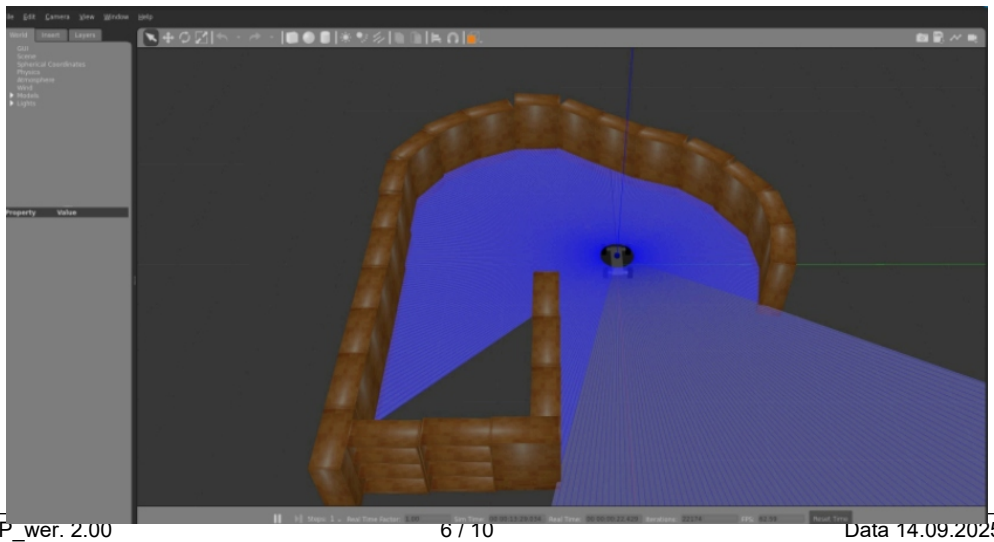


Końcowy wynik tworzenia modelu wygląda w następujący sposób

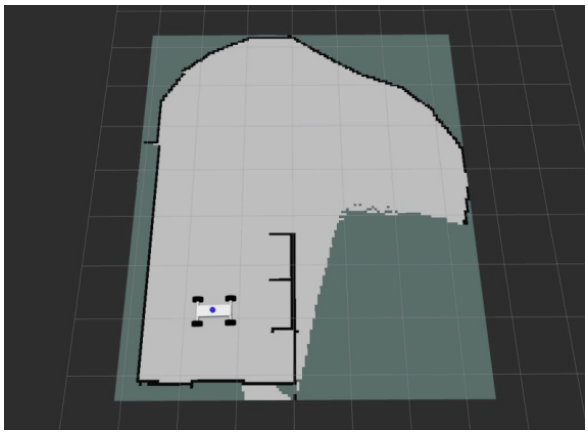
2.4 Łączenie programu w ROS2 z symulacją Gazebo

Do połączenia symulacji z ROS2 została użyta nakładka, zmieniająca tematy ROS2 na tematy Gazebo i vice versa. Dzięki temu dane z czujnika symulowanego w Gazebo może odebrać węzeł ROS2 i przekazywać go dalej do węzłów odpowiedzialnych za algorytmy sterowania, mapowania i nawigacji.

Symulacja po dodaniu LIDARu i przeszkód wygląd w następujący sposób. Wizualizowane są również promienie lasera.



Do wizualizacji tego co “widzi” robot wykorzystany został program RVIZ. Można w nim wyświetlić np. informacje odbierane na tematach i wizualizować na różne sposoby. Poniżej przedstawione jest to jak robot “widzi” swoją sytuację.

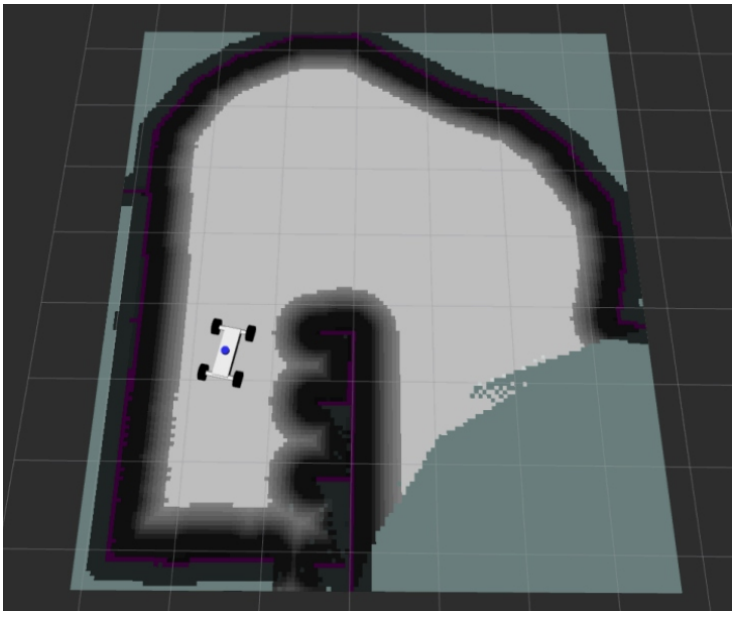


Jak widać informacja z lidaru jest interpretowana w zadowalający sposób. Widać również model robota, oraz między innymi orientację jego kół. Zawdzięczane jest to węzłowi “state publisher”, który na podstawie modelu i danych z symulowanych czujników położenia koła wysyła takie dane na odpowiedni temat ROS2.

2.4 Mapowanie i nawigacja

Do mapowania została wykorzystana paczka slam_toolbox, dzięki której tworzony jest temat /map, który publikuje dane dotyczące generowanej mapy. Dane te są oparte na pomiarach z lidaru publikowanych na temacie /scan. Rezultat tworzenia tej mapy można zobaczyć na poprzednim obrazie.

Następnie do nawigacji została wykorzystana paczka Nav2, która generuje mapę kosztów, widoczną na kolejnym rysunku. Mapa ta określa koszt każdego z możliwych położeń. Najciemniejsze piksele oznaczają obszary, w które robot nie może wjechać, szare piksele reprezentują miejsca, które robot powinien unikać, o ile nie prowadzi przez nie jedyna droga do celu, a jasne piksele symbolizują obszary dostępne do poruszania się. Za pomocą Nav2 można wyznaczyć cel podróży, a robot autonomicznie dotrze do wskazanego miejsca. System działa najlepiej, gdy dostępna jest wcześniej stworzona mapa kosztów, choć nie jest ona wymagana.



Aby poruszać się w nieznanym środowisku, konieczne jest ciągłe działanie paczki slam_toolbox, która w czasie rzeczywistym generuje mapę na podstawie danych z lidaru. Na tej podstawie Nav2 tworzy w czasie rzeczywistym mapę kosztów i nawigację robota. Dzięki temu robot jest w stanie poruszać się w dynamicznie zmieniających się lub nieznanych środowiskach, korzystając wyłącznie z danych z lidaru.

2.5 Raspberry PI

Komunikacja z tym komputerem jednopłytkowym odbywa się poprzez wifi. Gdy Raspberry Pi jest w tej samej sieci co komputer, można podglądać wiadomości krążące w programie na robocie zdalnie. Dzięki temu możemy także wysyłać polecenia do robota, np. goal pose. Instrukcja do odpalenia Rviza, przez który możemy się komunikować z programem na Raspberry Pi znajduje się w punkcie 2.7.

2.6 Symulacja

Używamy ROS Humble.

Aby uruchomić symulację potrzebujemy:
- Docker
- VS Code
- pliki z repozytorium
- instancja Ubuntu 22.04 (można próbować odpalić na Windowsie co powinno zadziałać ponieważ wszystko jest w kontenerze, ale interfejs graficzny czasami lubi się psuć. Na Linuxie powinno być ok)

Wykonujemy kroki:
1. Odpalamy Dockera (musimy być zalogowani)
2. Odpalamy VS Code i nawigujemy do folderu ./simulation w repozytorium ground-station-pg
3. Klikamy w lewym dolnym rogu programu przycisk „Remote Connection” i wybieramy reopen in container.
4. Następnie postępujemy zgodnie z krokami opisanymi w README.md w repozytorium
5. Powinien odpalić się program Gazebo i przykładowa scena.
6. Z potencjalnych problemów na Windowsie, może pojawić się że jeden z modułów gazebo nie odpalił się, prawdopodobnie jest to moduł który zajmuje się dźwiękiem i obrazem. Nie udało się nam znaleźć na niego rozwiązania na Windowsie, najlepszą opcją jest przeniesienie się na Linuxa, może być to maszyna wirtualna.

2.7 Rviz2

Rviz2 to narzędzie pozwalające na podejrzenie topiców ROS2 w bardziej przystępny sposób niż wypisywanie ich do terminala. Będziemy chcieli go odpalić gdy bawimy się w symulacji oraz gdy odpalamy prawdziwego robota i chcemy go kontrolować.

Aby go odpalić na komputerze przechodzimy kroki:
1. Spełnić wymagania do uruchomienia symulacji
2. Zrobić kroki od 1 do 3 z odpalania symulacji
3. Rviz nie instaluje się automatycznie przy odpaleniu kontenera więc wpisujemy komendę, do instalacji. Nie zostaje ona tu wklejona bo potencjalnie może zostać zmieniona w przyszłości, najlepiej znaleźć aktualną w internecie lub zapytać o nią chat bota.
4. Po tym odpalamy Rviz2 komendą w terminalu
5. Pojawi się okno Rviza2, jest ono dosyć intuicyjne, po lewo możemy dodawać topici do podglądania, a na górze jest opcja „goal pose”, możemy ją ustawić aby zadać miejsce w które chcemy żeby robot przejechał.

2.8 Program na ESP32

Model dev boarda to: ESP-WROOM-32

Program na ESP32 znajduje się w folderze esp32-encoders/esp32-encoders. Aby móc go edytować i wgrywać na płytkę należy wykonać następujące kroki:

Wymagania:
- VS Code
- ESP32 podłączony do któregoś z portów USB komputera

1. Pobieramy rozszerzenie ESP-IDF
2. Przechodzimy przez konfigurację rozszerzenia zgodnie z krokami które się wyświetlają, wybieramy opcję „simple configuration”. Instalacja może potrwać kilkanaście minut.
3. Po instalacji restartujemy VS Code.
4. Otwieramy folder esp32-encoders.
5. W folderze main mam plik main.c, w którym znajduje się całość kodu.
6. Aby wgrać program na płytkę wybieramy opcję ESP-IDF: Build, Flash and Monitor na dolnym pasku zadań w VS Code.
7. Jeśli rozszerzenie nie widzi płytki to klikamy „Select COM port to use” po lewej stronie przycisku do wgrywania i wybieramy port na którym mamy podłączonego ESP’a.

W kodzie można zmieniać parametry takie jak wymiary platformy czy prędkości kół.

2.9 Pliki 3D

W celu zmiany przenoszenia napędu platformy na skid steering zaprojektowane zostało kilka części. Wszystkie z nich są w folderze 3D-models na repozytorium. Pliki były tworzone w programie Autodesk Inventor, licencja na niego jest darmowa dla studentów.

lacznik2.ipt – łącznik do utrzymywania stałej odległości pomiędzy silnikami z przodu platformy
motor_shaft.ipt – wejście na wał od strony koła
nasilnikprzod1_new.ipt oraz nasilnikprzod2_new.ipt – holdery do silników z przodu
wal.ipt – wał Cardana

Wszystkie z plików mogą być otworzone w programie Autodesk Inventor i dowolnie modyfikowane, a następnie wyeksportowane do np. plik STEP i wydrukowane.

2.10 Program na drona

Program na drona, który odpala się na Raspberry Pi znajduje się w folderze raspberry-pg, jest to dodatkowe repozytorium, więc gdy je zaciągamy musimy dodać rekursywne pobieranie repozytoriów.

Aby je odpalić musimy połączyć się do Raspberry Pi po SSH. Aby to zrobić wchodzimy do tej samej sieci do której podłączone jest Raspberry Pi i wpisujemy komendę „ssh raspberry.local” jeśli to nie zadziała musimy podłączyć monitor do Raspberry Pi i sprawdzić jakie dostało IP w sieci, wtedy wpisujemy „ssh kamil@{ip_raspberry}” jeśli będzie wymagane hasło, wpisujemy „projekt123”, login to „kamil”.

Następnie odpalamy kontener Docker tak jak w punkcie z odpalaniem symulacji do momentu gdy odpalamy plik launch. W tym przypadku główna paczka robota nazywa się my_robot, a plik launch znajduje się w folderze my_robot/launch, komenda wygląda tak:

ros2 launch my_robot full_slam_launch.py

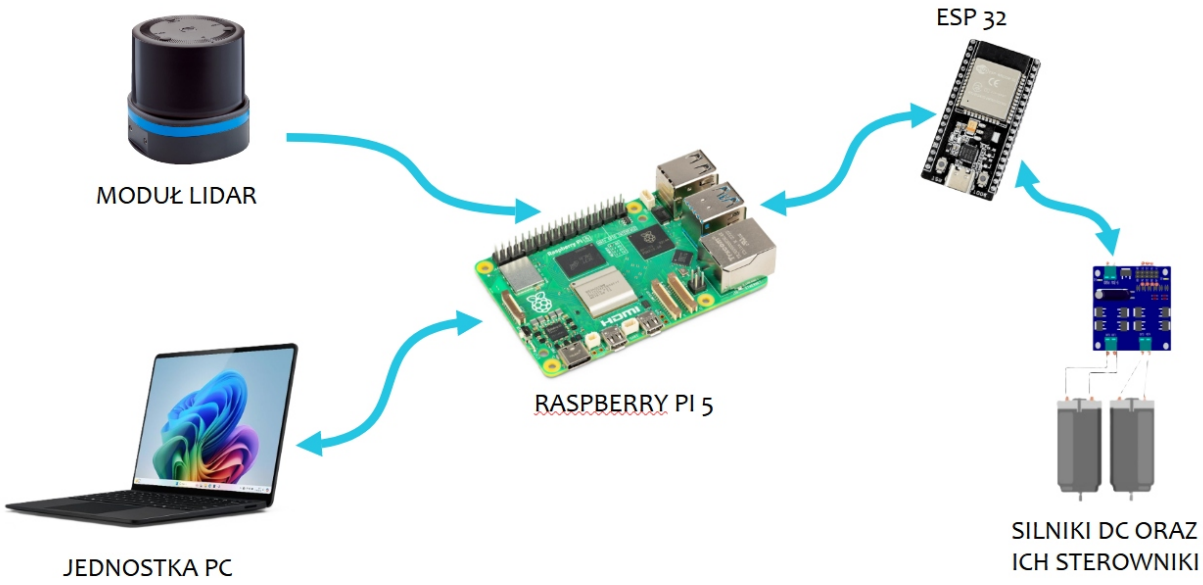
To powinno odpalić program. Po tym możemy kontrolować jego działanie w programie Rviz2 na zdalnym komputerze.

Paczki z których korzystamy to:

- slam_toolbox: tworzenie mapy i lokalizacja w przestrzeni
- nav2: nawigacja w terenie do zadanej pozycji goal_pose
- ldlidar_stl_ros2: paczka producenta dla lidaru zamontowanego na dronie
- uart_publisher: customowa paczka do komunikacji przy użyciu interfejsu UART

Pliki config do kontroli parametrów paczek znajdują się w folderze ros2_ws/src/my_robot/config.

2.11 Schemat ideowy działania drona



- Moduł Raspberry Pi jest sercem układu, to on odpowiada za to co robot robi i nim steruje przy użyciu paczki nav2.
- Lidar wysyła dane do programu ROS2 na Raspberry Pi na temat /scan
- Jednostka PC komunikuje się dwustronnie z Raspberry Pi, odbiera dane z topic'ów aby móc je podglądać zdalnie i wysyła dane na temat „goal pose" aby zadać pozycję do której chcemy się dostać.
- Esp32 odbiera z Raspberry Pi komendy ruchu w postaci prędkości liniowej i kątowej, a sam przelicza je na odpowiednie prędkości silników które ustawia i wysyła do sterowników. Odbiera natomiast dane z enkoderów i wysyła je do Raspberry Pi.
- Kontrolery silników wysterowują napięcia na silnikach.

3 Załączniki

Tabela 3.1. Lista załączników

L.p.	Nazwa dokumentu	Nazwa pliku
1.	**Repozytorium zdalne**	https://github.com/xKMx1/Projekt_grupowy_ID-44