Politechnika Świętokrzyska
Wydział Mechatroniki i Budowy Maszyn

ROZPRAWA DOKTORSKA

mgr inż. Maciej Salwa

ZASTOSOWANIE ALGORYTMU STEROWANIA W ROJU DLA BEZZAŁOGOWYCH OBIEKTÓW LATAJĄCYCH

Promotor:
dr hab. inż. Izabela Krzysztofik, prof. PŚk

Kielce 2025

Pragnę serdecznie podziękować:
Pani dr hab. inż. Izabeli Krzysztofik, prof. PŚk
za wszechstronną pomoc, opiekę naukową,
cenne rady,wszelką życzliwość
oraz zrozumienie

SPIS TREŚCI

Wykaz ważniejszych skrótów i terminów 9

WSTĘP 16

PRZEGLĄD KONCEPCJI STEROWANIA W ROJU 19

1.1. Literatura naukowa a sterowanie w roju 19

1.2. Przegląd literatury dla koncepcji sterowania w roju 26

1.3. Analiza implementacji na obiektach fizycznych 32

1.4. Podsumowanie analiz 40

CEL ORAZ ZAKRES BADAŃ 47

2.1. Modułowa koncepcja pracy 47

2.2. Cel główny badań 49

2.3. Cele szczegółowe 52

2.4. Zakres badań i struktura pracy 56

2.4.1. Planowane eksperymenty 57

2.4.2. Struktura pracy 58

JEDNOSTKA LATAJĄCA 60

3.1. Wymagania stawiane jednostce 60

3.1.1. Fizyczne oprzyrządowanie do komunikacji 61

3.1.2. Fizyczne oprzyrządowanie do nawigacji 65

3.1.3. Integracja danych czujników 67

3.1.4. Dodatkowe wyposażenie 72

3.1.5. Sterownik lotu 75
3.2. Model matematyczny 80
3.3. Podsumowanie rozdziału 85
ŚRODOWISKO GRAFICZNE 90
4.1. Programy i pakiety oprogramowania 90
4.2. Robot Operating System 91
4.3. Dotychczasowa koncepcja symulacji w środowisku graficznym Gazebo 95
4.3.1. Proponowana koncepcja walidacji 100
4.3.2. Przygotowanie eksperymentów 102
4.3.3. Wyniki eksperymentów walidacji 111
4.3.4. Analiza wyników walidacji 116
4.4. Przyjęta koncepcja środowiska graficznego do symulacji 120
4.5. Podsumowanie rozdziału 124
KOMUNIKACJA 128
5.1. Protokół komunikacji matka-operator a liderzy grup 128
5.1.1. Struktura wiadomości 129
5.1.2. Integracja protokołu 131
5.1.3. Bezpieczeństwo i potwierdzenie danych 131
5.1.4. Ograniczenia protokołu MAVLink 134
5.2. Protokół komunikacji wewnątrz grupy 135
5.2.1. Format struktury ramki 137
5.2.2. Podzielenie grupy 147

5.2.3. Mechanizm walidacji ramki 147
5.2.4. Zapisywanie buforu z wiadomością do pamięci jednostki 151
5.2.5. Komendy i informacje wysyłane jako informacje dodatkowe . 152
5.2.6. Schemat potwierdzenia odebrania danych ACK..... 153
5.3. Bezpieczeństwo i nawigacja 156
5.3.1. Autonomia jednostki 156
5.3.2. Ustalenie lokalnego układu współrzędnych 163
5.3.3. Kompensacja szyku 166
5.4. Walidacja opracowanej komunikacji wewnętrznej 167
5.4.1. Walidacja komunikacji wewnątrz roju z użyciem fizycznych sterowników lotu 168
5.4.2. Walidacja komunikacji wewnątrz roju z użyciem opracowanego środowiska graficznego 173
5.5. Podsumowanie rozdziału 175
MATKA-OPERATOR 177
6.1. Struktura programu sterującego..... 177
6.2. Struktura roju 184
6.3. Inteligencja roju 189
6.4. Przeprowadzone badania..... 196
6.4.1. Dobranie trajektorii jednostek roju przez matkę-operatora 196
6.4.2. Reakcja roju w szyku liniowym na wystąpienie nieznanej przeszkody 199
6.4.3. Szyk dwurzędowy..... 204
6.4.4. Uzupełnienie szyku 206

6.4.5. Konieczność wykonania manewru przed zakończeniem uzupełnienia szyku 209
6.5. Analiza wyników eksperymentów 211
6.6. Podsumowanie rozdziału 212
Zakończenie 215
LITERATURA 218
Załącznik nr 1 242
Załącznik nr 2 249
Załącznik nr 3 252
Załącznik nr 4 255

Wykaz ważniejszych skrótów i terminów

ACK (*Acknowledge*) – sygnał lub komunikat w protokołach komunikacyjnych potwierdzający, że dane zostały odebrane poprawnie. Gdy urządzenie odbierze dane, wysyła wiadomość ACK do nadawcy, informując go, że dane dotarły. Jeśli ACK nie zostanie odebrane, nadawca może ponownie przesłać dane, co pomaga w zapewnieniu niezawodnej komunikacji.

ArduPilot – wszechstronny i otwartoźródłowy system autopilota, który obsługuje różne typy pojazdów, takie jak drony, samoloty, łodzie i samochody. Jest znany ze swojej elastyczności i szerokiego wsparcia dla wielu różnych kontrolerów lotu, takich jak Pixhawk.

DARPA (*Defense Advanced Research Projects Agency*) – agencja rządowa USA odpowiedzialna za rozwój zaawansowanych technologii dla wojska. DARPA prowadzi wiele innowacyjnych projektów, w tym te związane z robotyką, sztuczną inteligencją, systemami autonomicznymi oraz bezzałogowymi statkami powietrznymi.

Debugowanie – proces identyfikacji i usuwania błędów w oprogramowaniu lub systemie. Polega na analizie kodu, śledzeniu działania programu oraz korygowaniu błędnych fragmentów, aby zapewnić poprawne działanie aplikacji.

Dron – na potrzeby pracy rozumiany w uściśleniu tylko do obiektów latających, traktowany jako synonim UAV.

Gazebo – zaawansowany program do symulacji robotów, który pozwala na testowanie i rozwijanie algorytmów oraz systemów wirtualnych środowisk. Gazebo oferuje realistyczne modelowanie fizyki, sensorów oraz interakcji między obiektami, używając silnika graficznego.

Inteligencja roju – zgodnie z definicją dostępną w literaturze naukowej: „Swarm intelligence refers to the study and simulation of the collective

behavior of gregarious creatures, such as insects, for the purpose of solving complex problems. It involves the exchange of information between decentralized and self-organized systems, leading to global system behavior" [90]. Tłumacząc: „Inteligencja roju odnosi się do badania i symulacji zbiorowego zachowania istot stadnych, takich jak owady, w celu rozwiązywania złożonych problemów. Obejmuje ona wymianę informacji między zdecentralizowanymi i samoorganizującymi się systemami, prowadząc do globalnego zachowania systemu".

Interfejs (w kontekście modułów) – zbiór reguł, standardów i wymagań definiujących sposób, w jaki moduły komunikują się i współdziałają ze sobą w ramach większego systemu. Interfejs określa, jakie dane moduł musi przyjmować, jakie dane zwracać oraz jakie operacje mogą być wykonywane. Dzięki temu, moduły mogą być wymieniane lub zastępowane bez konieczności modyfikacji całego systemu, pod warunkiem, że nowy moduł spełnia te same wymagania interfejsu. Interfejsy są kluczowe dla modularności i elastyczności systemów, umożliwiając niezależny rozwój i łatwą integrację różnych komponentów.

Internet Rzeczy (IoT, *Internet of Things*) – sieć połączonych ze sobą urządzeń, które mogą komunikować się i wymieniać dane przez internet bez udziału człowieka. Urządzenia te mogą obejmować czujniki, kamery, sterowniki oraz różnorodne obiekty codziennego użytku. W kontekście dronów, IoT umożliwia zdalne monitorowanie, zarządzanie i kontrolę dronów w czasie

Kotwica (*Anchor*) – w systemie UWB stacjonarne urządzenie o znanej pozycji, które odbiera i wysyła sygnały od urządzeń mobilnych. System UWB wykorzystuje sygnały z kilku kotwic do triangulacji położenia obiektu, co pozwala na dokładne określenie jego pozycji w przestrzeni 3D.

Lider – określany również jako lider podgrupy lub podroju. Jednostka roju, która ma specjalne uprawnienia do zarządzania swoją podgrupą i komunikowania się z matką lub liderami innych podgrup.

Matka-operator – określana również jako matka. Obiekt stojący najwyżej w hierarchii roju. Mający za zadania koordynować pracą całego roju. Określenie odnosi się do jednostki najczęściej nie biorącej aktywnie udziału w roju. Odpowiednik stacji naziemnej w postaci komputera, który realizuje algorytm roju. Może działać autonomicznie lub być obsługiwana przez operatora.

MAVLink (*Micro Air Vehicle Link*) – lekki, niskopoziomowy protokół komunikacyjny używany do wymiany danych między autopilotem a stacją naziemną, dronem lub innymi urządzeniami. Protokół zaprojektowany do pracy w systemach bezzałogowych, takich jak drony.

Moduł – odrębna, funkcjonalna jednostka w systemie, która realizuje określone zadania. Moduły mogą być zarówno sprzętowe (np. moduły GPS, kamery, czy komunikacyjne), jak i programowe (biblioteki, komponenty oprogramowania). W architekturze oprogramowania, modułowe podejście pozwala na łatwiejsze zarządzanie projektem, modyfikowanie jego części bez wpływu na całość oraz ponowne wykorzystanie innych modułów w razie wymiany jednego z nich. Moduły do zachowania spójności wymagają zdefiniowanych reguł oraz wymagań koegzystencji zwanych interfejsami.

NASA (*National Aeronautics and Space Administration*) – amerykańska agencja rządowa odpowiedzialna za program kosmiczny oraz badania aeronautyczne. NASA jest także zaangażowana w rozwój technologii związanych z autonomią lotów, nawigacją, komunikacją oraz zarządzaniem ruchem dronów (UTM – *Unmanned Aircraft System Traffic Management*).

Ogłoszenie (*advertisement*) – forma komunikacji rozpowszechniona głównie dzięki urządzeniom typu Beacon. Jako ogłoszenie należy rozumieć wysłany pakiet danych, nieadresowany do konkretnej jednostki. Każda jednostka może odebrać ten pakiet. Nie jest wymagane potwierdzenie, ani żadna reakcja. Ogłoszenia wysyłane są cyklicznie przez jednostkę ogłaszającą.

Quadrocopter – to wielowirnikowiec z czterema wirnikami, które są zazwyczaj rozmieszczone na ramionach w kształcie litery „X”.

Protokół komunikacyjny – zbiór zasad i reguł, które definiują sposób wymiany danych między urządzeniami w sieci. Protokoły określają, jak dane są formatowane, przesyłane i odbierane, zapewniając poprawność i spójność komunikacji.

Proxy – pośrednik między dwoma urządzeniami lub systemami, który przekazuje dane pomiędzy nimi. Proxy może pełnić różne funkcje, takie jak kontrola dostępu, filtrowanie ruchu, buforowanie danych lub ukrywanie tożsamości użytkownika. W systemach rozproszonych, takich jak rój UAV, proxy może być używane do zarządzania komunikacją pomiędzy dronami a centralnym serwerem, którego rolę pełni matka-operator.

PX4 – otwartoźródłowy system autopilota, stworzony z myślą o elastyczności i skalowalności. Obsługuje różne typy dronów i robotów, oferując zaawansowane funkcje, takie jak lot autonomiczny, stabilizacja i integracja z ROS.

Radio (w kontekście transmisji radiowej) – odnosi się do technologii komunikacji bezprzewodowej, która wykorzystuje fale radiowe do przesyłania danych między urządzeniami. W systemach dronów, radio jest używane do przesyłania sygnałów sterujących, telemetrii, a także strumieniowania wideo z kamer drona.

Ramka protokołu – struktura danych używana do transmisji informacji. Zawiera dane użytkowe oraz kontrolne, które umożliwiają odbiorcy rozpoznanie, zrozumienie i poprawne zinterpretowanie przesyłanych danych. Ramka protokołu składa się z różnych pól, takich jak np. nagłówek, ładunek danych, suma kontrolna oraz końcówka.

ROS/ROS2 (*Robot Operating System*) – otwartoźródłowy zestaw bibliotek używany w robotyce, który dostarcza narzędzia, moduły programistyczne i konwencje do tworzenia złożonych systemów robotycznych. ROS pozwala na modularne projektowanie oprogramowania, gdzie różne części systemu (węzły) mogą komunikować się ze sobą, wymieniając dane sensoryczne, polecenia oraz inne informacje.

Rój – grupa obiektów. Według definicji podanej w literaturze naukowej: „A swarm or fleet of Unmanned Aerial Vehicles (UAVs) is a set of aerial robots i.e., drones that work together to achieve a specific goal”[184]. Tłumacząc: „Rój lub flota bezzałogowych statków powietrznych (UAV) to zestaw robotów powietrznych, tj. dronów, które współpracują ze sobą, aby osiągnąć określony cel. W podrozdziale 1.1. przedstawiono wiele definicji tego pojęcia. Na potrzeby pracy przyjęto definicje roju jako zespołu wielu jednostek, działających w zorganizowany sposób, dążących do wspólnego celu, realizując go w kooperacji wzajemnej.

SITL (*Software In The Loop*) – technika symulacyjna stosowana w systemach sterowanych komputerowo, takich jak drony, pojazdy autonomiczne lub systemy robotyczne. Polega ona na uruchamianiu oprogramowania sterującego na rzeczywistym kontrolerze lub komputerze, podczas gdy reszta systemu (np. dynamika pojazdu, sensory, środowisko) jest symulowana. SITL umożliwia testowanie logiki sterowania, algorytmów i systemów decyzyjnych bez potrzeby fizycznych testów z rzeczywistymi urządzeniami, co redukuje koszty i ryzyko.

Sterownik lotu – elektroniczne urządzenie, które zarządza i kontroluje ruchy oraz stabilność drona. Otrzymuje dane z czujników (takich jak akcelerometry, żyroskopy i GPS). Poprzez oprogramowanie sterownika lotu wykonuje odpowiednie obliczenia, aby sterować silnikami i innymi elementami, zapewniając stabilny lot oraz realizację zaplanowanych manewrów.

Suma kontrolna (CRC, *Cyclic Redundancy Check*) – metoda detekcji błędów w danych przesyłanych przez sieć lub zapisanych na nośnikach danych. CRC jest generowana przez algorytm, który na podstawie przesyłanych danych tworzy sumę kontrolną w postaci dodatkowych informacji wysyłanych wraz z danymi. Odbiorca danych oblicza własną sumę kontrolną i porównuje ją z przesłaną wartością. Jeśli te wartości się zgadzają, dane uznawane są za przesłane poprawnie. W przeciwnym razie wiadomo, że wystąpił błąd w transmisji.

Sztuczna Inteligencja – dziedzina informatyki i nauk matematycznych, której celem jest tworzenie systemów zdolnych do wykonywania zadań, które normalnie wymagają ludzkiej inteligencji.

Środowisko symulacyjne – zaawansowane narzędzie programowe, które umożliwia modelowanie, testowanie i analizę zachowania systemów w wirtualnym, kontrolowanym otoczeniu. W kontekście dronów, środowisko symulacyjne pozwala na tworzenie realistycznych scenariuszy lotu, uwzględniając fizykę, warunki atmosferyczne oraz interakcje z otoczeniem. Może także wspierać testowanie i rozwój własnych protokołów komunikacyjnych oraz integrację różnych czujników.

TensorFlow – otwartoźródłowa biblioteka programistyczna do uczenia maszynowego i uczenia głębokiego, rozwinięta przez Google.

Test Driven Development (TDD) – metodologia programowania, w której pisanie testów poprzedza implementację kodu. W TDD programista najpierw tworzy testy jednostkowe określające, jak powinien działać dany fragment kodu, a następnie implementuje funkcjonalność, która spełnia te testy. Podejście to pozwala na wczesne wykrywanie błędów oraz zapewnia, że kod spełnia założone wymagania. Sumaryczny cel i jego walidacja są tożsame z poprawnym przejściem przez wszystkie testy jednostkowe. Podejście to jest często stosowane w modułowej realizacji złożonych systemów i pozwala niezależnie testować dane partie całego rozwiązania.

UAV (*Unmanned Aerial Vehicles*) – bezzałogowy obiekt (pojazd) latający. Definicja przytoczona w pracy naukowej podaje: „*Unmanned aerial vehicles (UAVs) are the robotic devices that can fly without the help of a pilot*” [89], co można tłumaczyć jako: „Bezzałogowe obiekty latające to zrobotyzowane urządzenia mogące latać bez pomocy pilota”.

Uczenie Maszynowe – poddziedzina sztucznej inteligencji, która polega na tworzeniu algorytmów zdolnych do uczenia się z danych. Algorytmy te analizują zbiory danych, rozpoznają wzorce i potrafią podejmować decyzje

lub przewidywać wyniki na ich podstawie. Możliwe jest uczenie z obserwatorem weryfikującym poprawność lub bez obserwatora.

Ultra-Wideband (UWB) – technologia komunikacji radiowej, która wykorzystuje bardzo szerokie pasmo częstotliwości (ponad 500 MHz) do transmisji danych. UWB charakteryzuje się bardzo niskim poziomem mocy sygnału, co minimalizuje zakłócenia z innymi technologiami bezprzewodowymi. Jest często stosowana do precyzyjnego pozycjonowania i lokalizacji w pomieszczeniach, ponieważ umożliwia pomiar odległości z dokładnością do kilku centymetrów dzięki precyzyjnemu pomiarowi czasu przelotu sygnału.

VTOL (*Vertical Take-Off and Landing*) – jednostka powietrzna zdolna do startu i lądowania w pionie, bez potrzeby użycia pasa startowego. Pojazdy VTOL mogą być różnego rodzaju, w tym samoloty, helikoptery oraz wielowirnikowce. Wiele współczesnych dronów klasyfikuje się jako VTOL, ponieważ mogą unosić się i lądować pionowo, co jest szczególnie przydatne w miejscach, gdzie brakuje przestrzeni na klasyczny start.

Warstwa fizyczna – najniższa warstwa w modelu OSI (*Open Systems Interconnection*), która odpowiada za bezpośrednią transmisję sygnałów pomiędzy urządzeniami w sieci. W tej warstwie definiowane są aspekty techniczne komunikacji, takie jak medium transmisyjne (np. przewody, fale radiowe), poziomy napięcia, modulacja, kodowanie sygnałów oraz szybkość transmisji danych. Warstwa fizyczna zapewnia mechanizmy do przesyłania bitów między urządzeniami oraz konwersję danych cyfrowych na sygnały analogowe lub cyfrowe, które mogą być przesyłane przez medium.

Wielowirnikowiec – rodzaj bezzałogowego statku powietrznego (UAV) lub załogowego pojazdu, który jest napędzany przez więcej niż dwa wirniki (śmigła). Wirniki te generują siłę nośną, umożliwiającą unoszenie się i manewrowanie pojazdu w powietrzu. Wielowirnikowce mogą mieć różną liczbę wirników, co wpływa na ich stabilność, udźwig i manewrowość. Najpopularniejsze konfiguracje to: quadrocoptery, tricoptery, hexacoptery.

WSTĘP

Rozwój technologii lotniczej w ostatnich dekadach doprowadził do powstania wielu innowacyjnych rozwiązań, w tym bezzałogowych obiektów latających (ang. *Unmanned Aerial Vehicles, UAVs*). Od początków ich zastosowania w wojskowości, poprzez misje ratunkowe, aż po cywilne zastosowania komercyjne, bezzałogowe obiekty latające powszechnie nazywane „dronami" stały się nieodłącznym elementem współczesnej rzeczywistości technologicznej. Jednym z najbardziej obiecujących i dynamicznie rozwijających się obszarów badań w tej dziedzinie jest sterowanie grupami dronów w szyku nazywanym „rojem".

Powstanie koncepcji „roju dronów" sięga inspiracją do natury. Obserwacje zachowania zbiorów owadów, ptaków i innych zwierząt, zainspirowały inżynierów do stworzenia systemów, w których wiele autonomicznych jednostek współpracuje w celu osiągnięcia konkretnego celu. Mrówki i pszczoły, działając w zorganizowanych koloniach, czy stada ptaków i ławice ryb, poruszające się w skoordynowany sposób, dostarczyły naukowcom cennych wzorców zachowań, które stały się podstawą dla algorytmów sterowania grupą dronów. Teorie o „inteligencji roju", które zaczęły pojawiać się już w latach 80. i 90. XX w., były kluczowe w rozwoju tej dziedziny.

Pierwszymi osobami, które formalnie przedstawiły koncepcję „sterowania w roju", byli Gerardo Beni i Jing Wang. W swojej pracy z 1989 r. zatytułowanej „*Swarm Intelligence in Cellular Robotic Systems*" [28] zaproponowali ideę zdecentralizowanego podejścia do sterowania wieloma robotami, działającymi w koordynacji. Inspiracją dla jej powstania było zachowanie kolonii owadów. Ich badania dały początek nowemu kierunkowi w robotyce i sztucznej inteligencji, który obecnie jest szeroko rozwijany na całym świecie.

Od momentu wprowadzenia koncepcji, badania nad sterowaniem w roju ewoluowały w znaczący sposób. W latach 90 oraz na początku XXI w., rozwój technologii komputerowych i komunikacyjnych umożliwił bardziej zaawansowane symulacje oraz testy rzeczywistych systemów rojów.

Przykładem przełomowych prac z tego okresu są badania dotyczące algorytmów określanych mianem *flockingu* prowadzonych przez m.in. Craiga Reynoldsa, które miały na celu modelowanie stadnych zachowań [162].

Sterowanie rojem dronów niesie ze sobą korzyści, które sprawiają, że jest to obszar o ogromnym potencjale. Przede wszystkim rój dronów może wykonywać zadania z większą efektywnością niż pojedynczy dron. Dzięki współpracy pomiędzy poszczególnymi jednostkami możliwe jest pokrycie większych obszarów, bardziej precyzyjne monitorowanie sytuacji oraz szybsza reakcja na zmieniające się warunki. Drony w roju mogą również wspierać się nawzajem w przypadku awarii jednej z jednostek, co zwiększa niezawodność całego systemu.

Pomimo licznych korzyści sterowanie rojem dronów wiąże się także z wieloma wyzwaniami. Kluczowym problemem jest opracowanie skutecznych algorytmów, które umożliwią autonomiczne działanie wielu dronów w dynamicznym i często nieprzewidywalnym środowisku. Algorytmy muszą uwzględniać zarówno aspekty związane z komunikacją między dronami, jak i z ich nawigacją oraz unikaniem kolizji. Ważnym elementem jest również zapewnienie bezpieczeństwa i niezawodności całego systemu, zwłaszcza w kontekście potencjalnych zagrożeń związanych z cyberatakami oraz zakłóceniami sygnałów GPS.

Technologia roju dronów jest wykorzystywana zarówno w sektorze wojskowym, jak i cywilnym. W sektorze wojskowym drony mogą być wykorzystywane do misji rozpoznawczych, obserwacji, a nawet ataków. Działając w sposób skoordynowany, rój może zwiększyć skuteczność operacji oraz zmniejszyć ryzyko strat dla załogowych jednostek. Przykładem może być tutaj program DARPA (ang. *Defense Advanced Research Projects Agency*) [48] dotyczący rojów dronów, który zakłada użycie wielu małych dronów do operacji w trudno dostępnych miejscach.

W związku z możliwymi zastosowaniami koncepcji sterowania obiektami w roju w sektorze militarnym, znaczna część prac dokonanych w tym zakresie jest utajniona i nie jest dostępna publicznie. Należy mieć na uwadze, że z racji na konflikty zbrojne na całym świecie, dysproporcja w zakresie technologii uzbrojenia może stanowić ogromną przewagę.

W związku z tym badania nad algorytmiką sterowania wszelakich obiektów latających stały się istotną gałęzią dochodów koncernów zbrojeniowych.

W sektorze cywilnym drony w roju mogą być używane do monitorowania obszarów rolniczych, zarządzania kryzysowego podczas katastrof naturalnych, inspekcji infrastruktury oraz dostarczania przesyłek. Przykładem tego ostatniego zastosowania jest firma Amazon, która testuje systemy dostarczania przesyłek za pomocą dronów działających w roju [12]. Innym przykładem jest projekt badawczy nad użyciem roju dronów do monitorowania i zarządzania obszarami leśnymi w celu zapobiegania pożarom [17].

Celem mojej rozprawy doktorskiej było opracowanie algorytmu sterowania w roju dla bezzałogowych obiektów latających. W poniższej pracy odniesiono się do zarówno prac naukowych, jak i analizy dostępnych materiałów przedstawiających wdrożenia koncepcji sterowania dronami w roju.

Rozdział 1

PRZEGLĄD KONCEPCJI STEROWANIA W ROJU

Analiza koncepcji sterowania w roju została skierowana zarówno w stronę literatury naukowej, jak i rozwiązań komercyjnych dedykowanych dla przemysłu, czy wojska. Podjęto próbę zdefiniowania pojęcia roju i kategoryzacji rozwiązań, jakie mogą zostać wykorzystane do sterowania rojem.

Rozdział otwiera przegląd literatury naukowej. Jako punkt wyjścia dla mechanizmów i siły roju zostały streszczone badania prowadzone nad jednostkami stanowiącymi naturalny archetyp roju, takimi jak mrówki i pszczoły. Przedstawiono również opracowane koncepcje zastosowań roju w kontekście sektorów, w których mogłyby znaleźć swoje zastosowanie.

1.1. Literatura naukowa a sterowanie w roju

Literatura naukowa stanowi bazę pracy poczynionej w zakresie optymalizacji sterowania wieloma obiektami latającymi, w tym dla specjalnej kategorii systemów znanych jako roje. Wzorce zachowania zaczerpnięte z natury, takie jak ruchy rojów ptaków, ryb czy owadów, stały się inspiracją dla inżynierów i naukowców w opracowywaniu algorytmów sterowania dla złożonych systemów wielojednostkowych.

Początki badań nad inteligencją rojów sięgają lat 90 XX wieku, kiedy to wprowadzono podstawowe założenia dla rozważanej koncepcji. J. Kennedy i R. Eberhart w artykule *Particle Swarm Optimization* [107] z 1995 roku zaprezentowali algorytm optymalizacji, inspirowany zachowaniem stad ptaków i ławic ryb. Był to jeden z pierwszych przykładów wykorzystania wzorców zachowań zwierząt w technikach obliczeniowych, który później stał się fundamentem dla wielu innych badań.

W 2001 roku L. Parker opublikowała przegląd *Current State of the Art in Distributed Autonomous Mobile Robotics* [149], gdzie omówiła stan badań nad rozproszonymi, autonomicznymi systemami mobilnymi, w tym

nad koncepcją sterowania rojami robotów. Natura była niezmiennie źródłem inspiracji dla badań nad sterowaniem w roju.

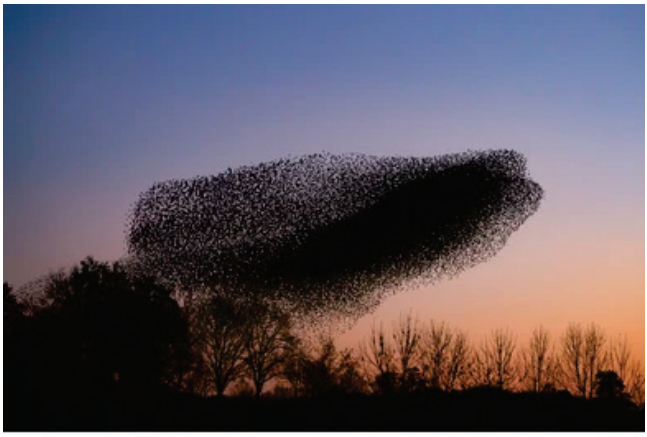
W 2004 roku Bonabeau Dorigo i Theraulaz w swojej książce *Swarm Intelligence: From Natural to Artificial Systems* [32] szczegółowo opisali, jak wzorce zachowań owadów społecznych, takich jak mrówki i pszczoły, mogą być stosowane do rozwiązywania problemów inżynierskich. Algorytmy inspirowane ruchem mrówek w poszukiwaniu pokarmu, znalazły zastosowanie w optymalizacji tras w sieciach komputerowych. Sam rój w języku polskim w kontekście natury najczęściej bywa kojarzony ze zorganizowanymi grupami pszczół lub mrówek.

Jeszcze zanim zaczęto analizować rój w kontekście badań nad przeniesieniem koncepcji zaczerpniętej z natury do świata techniki, badacze dogłębnie poszerzali wiedzę na temat złożoności organizacji na przykładzie między innymi wyżej wymienionych rojów mrówek. Jak zauważyła biolog z Uniwersytetu Stanford [56], istnieje znaczący wzrost możliwości grupy nawet prymitywnych jednostek, które potrafią kolektywnie współpracować. W wywiadzie dla National Geographic [139] dziennikarz pytał biolog o zależność siły roju oraz potencjału jednostek. Cytat:

– *Mrówki nie są małymi inteligentnymi konstruktorami, architektami, nawet wojownikami – przynajmniej nie w pojedynkę. Gdy przychodzi chwila, że trzeba podjąć decyzję, większość mrówek nie ma pojęcia, co robić dalej*. *Gdy się obserwuje pojedynczą mrówkę, jest się pod wrażeniem jej bezgranicznej nieudolności* – mówi Deborah M. Gordon, biolog z Uniwersytetu Stanforda. – *Jak w takim razie wyjaśnić sukces w kolonizacji naszej planety odniesiony przez blisko 12 tys. gatunków mrówek? W ciągu 140 mln lat musiały się czegoś nauczyć*. *Mrówki nie są inteligentne. To kolonia mrówek jest inteligentna* – zapewnia badaczka. *Mrowisko jest w stanie rozwiązywać problemy będące absolutnie poza zasięgiem indywidualnego osobnika, takie jak znalezienie najkrótszej drogi do źródła pokarmu, rozdzielenie różnych zadań między robotnice czy obrona terytorium przed sąsiadami. Jako pojedyncze osobniki mrówki to małe tępaki, jako kolonia – reagują szybko i skutecznie na wyzwania, jakie stawia im środowisko. Dokonują tego dzięki czemuś, co można by nazwać „inteligencją stadną”.*

Dzięki poczynionym spostrzeżeniom zaczęto mówić szeroko o inteligencji w kontekście roju, która jak zauważyli badacze, nie jest dziełem jednostki, czy sumą inteligencji jednostek, a mechanizmem synergicznym wypracowywanym przez kolektyw grupy.

Jak dowiodła Lauren A. Richardson [163], ten sam mechanizm stadny wykazują również pszczoły. Pojedyncze jednostki są słabe fizycznie. Niezdolne do wykonania niektórych zadań właściwych dla przetrwania gatunku oraz są podatne na zagrożenia ze strony silniejszych osobników innych gatunków. Siłą pszczół jest jednak ich zespołowość i organizacja, dzięki czemu pszczoły potrafią pokonać np. przeciwnika w postaci szerszenia. Jego pancerz jest odporny na ukąszenia ze strony pszczół i żadna w pojedynkę nie jest w stanie zagrozić szerszeniowi, który potrafi masowo zabijać pszczoły. Metodą walki pszczół jest otoczenie agresora możliwie dużą ilością, często ponad setką jednostek. Gdy pszczoły kumulują się wokół szerszenia, obsiadają go i przez przekazanie ciepła jakie generują, stykając się z szerszeniem, doprowadzają do jego śmierci poprzez przegrzanie. Finalne zwycięstwo roju jest okupione śmiercią kilku jednostek, które szerszeń zabije w czasie walki. Obrazuje to motywacje zachowań wśród pszczół, które działając w kolektywie, potrafią powierzyć i zaryzykować dobro jednostki dla całej grupy. Tylko dzięki temu są w stanie odeprzeć atak szerszenia. Oprócz omówionego mechanizmu walki, najczęściej kojarzonym z rojem w przyrodzie zjawiskiem jest przelot grupy pszczół.



Zdj. 1.1. Przelot roju pszczół

Na zdjęciu 1.1 przelotu roju pszczół można zaobserwować ogromną ilość owadów, które pojedynczo są zbyt małe, aby mogły zostać dostrzeżone. Dzięki przyjęciu formacji i trzymaniu szyku są one jednak widoczne z daleka, tworząc kolektyw nazywany rojem.

W 2005 roku Berman i współpracownicy w artykule *Algorithms for the Analysis and Synthesis of a Bio-inspired Swarm Robotic System* [29] zaprezentowali algorytmy inspirowane biologią do analizy i syntezy systemów robotycznych, działających w roju. Badania te ukazały, jak zasady samoorganizacji i rozproszonego podejmowania decyzji mogą być zaadaptowane do systemów technicznych.

W ostatnich latach badania nad sterowaniem w roju osiągnęły nowe poziomy złożoności i zastosowań. W 2013 roku, Rubenstein, Cornejo i Nagpal opublikowali artykuł *Programmable Self-Assembly in a Thousand-Robot Swarm* [165], w którym opisali eksperymenty z programowalnym, samoorganizowaniem się roju, składającego się z tysiąca jeżdżących robotów. Badania te demonstrują, jak zaawansowane techniki sterowania mogą prowadzić do tworzenia dużych, skoordynowanych systemów robotycznych.

Kolejne osiągnięcie naukowe to artykuł Anam Tahir *Formation Control of Swarms of Unmanned Aerial Vehicles* z 2023 roku [183]. Autorka skupiła się na stabilności i metodach sterowania dla rojów bezzałogowych pojazdów latających (UAV), co ma kluczowe znaczenie dla zastosowań, takich jak monitorowanie środowiskowe.

Na przekroju lat i badań naukowych widać tendencje do zmiany podejścia do problemy roju oraz adaptowania koncepcji do właściwych dla danych lat możliwości technologicznych.

Badania nad algorytmami sterowania w roju znalazły szerokie zastosowanie w różnych dziedzinach, od optymalizacji tras w sieciach komputerowych, po zarządzanie flotami pojazdów. Wykorzystując zasady samoorganizacji, adaptacji i rozproszonego lub hierarchicznego podejmowania decyzji, algorytmy oferują skalowalne i efektywne rozwiązania dla złożonych problemów.

Jednym z pierwszych i najbardziej znanych zastosowań algorytmów rojowych jest optymalizacja tras w sieciach komputerowych. Algorytmy

inspirowane zachowaniem mrówek, takie jak praca pod tytułem *Ant Colony Optimization* [173], zostały szeroko zaadaptowane do tego celu.

W transporcie algorytmy sterowania w roju są wykorzystywane do zarządzania grupami inteligentnych pojazdów, które muszą współpracować w celu osiągnięcia wspólnego zadania [164]. Przykłady takich zastosowań znajdziemy w logistyce, eksploracji terenu, czy ratownictwie.

Algorytmy sterowania w roju znalazły również zastosowanie w dziedzinie bezpieczeństwa i obronności, gdzie są używane głównie do zarządzania flotami bezzałogowych pojazdów powietrznych (UAV) w misjach rozpoznawczych [47] i operacyjnych [168].

W monitorowaniu [200] i zarządzaniu środowiskiem, algorytmy sterowania w roju są używane do zarządzania sieciami sensorów oraz flotami dronów monitorujących zmiany w środowisku naturalnym [41], takie jak zmiany klimatyczne, pożary lasów, czy zanieczyszczenia.

W przemyśle i logistyce, algorytmy sterowania w roju są stosowane do optymalizacji procesów produkcyjnych, logistycznych [110] oraz zarządzania magazynami [198], gdzie wiele autonomicznych pojazdów transportowych musi współpracować w celu efektywnego zarządzania zasobami i zamówieniami.

Obecnie wraz ze wzrostem popularności internetu rzeczy IoT (*Internet of Things*) oraz Przemysłu 4.0, coraz częstszym staje się zastosowanie dronów w kontekście inteligentnego monitoringu, szczególnie tam gdzie potrzebne jest zastosowanie czujników, których obszar pomiarów jest rozległy i wymaga przemieszczenia czujnika, celem dokonania pomiarów w różnych miejscach. Przykładem tutaj mogą być uprawy rolnicze, nad którymi rozważania stają się treścią prac naukowych. W 2023 badacze [140] pokusili się o porównanie dostępnych na rynku koncepcji roju dronów w kontekście monitorowania upraw. We wnioskach wyrazili opinię, że roje UAV są przyszłością upraw szeroko terytorialnych, które nazwali *unmanned farm,* co można przetłumaczyć jako autonomiczne (bezzałogowe) uprawy.

Ważnym z punktu widzenia nauki jest postawienie pytania, czym jest rój? Z angielskiego termin *Swarm* bywa używany także między innymi w naukach biologicznych. Istnieje również algorytm PSO – *Particle Swarm Optimization* (ang. optymalizacja rojem cząstek) [107]. Jest to zagadnienie

optymalizacji, jednak nie jest ono powiązane z algorytmem sterowania w roju w ujęciu robotyki mobilnej. W 2004 roku Gerardo Beni wygłosił pracę [27], w której wyszedł od zagadnienia ogólnej inteligencji roju do ujęciu roju w kontekście robotyki. Założonym efektem jego pracy miało być usystematyzowanie wiedzy i rozwianie wątpliwości na temat roju. Nie wyłonił on jednak definicji roju, a usystematyzował wiedzę na temat spektrum zagadnień, jak szeroko ten termin jest używany w zagadnieniach naukowych.

Rok później próbę zdefiniowania roju, we wspomnianym zakresie robotyki mobilnej, podjął Erol Sahin. W swojej pracy [167] zwrócił uwagę na niejednoznaczność tego słowa w kontekście mnogości obszarów i odmiennego znaczenia poszczególnych definicji. Definicja jaką podał w swojej pracy, odnosząca się do zagadnienia badań w zakresie robotyki mobilnej i roju to (cytat):

„Rój (robotyczny) to badanie tego, w jaki sposób duża liczba stosunkowo prostych fizycznie, osadzonych jednostek może zostać zaprojektowana w taki sposób, aby pożądane zachowanie zbiorowe wyłaniało się z lokalnych interakcji między jednostkami, oraz między jednostkami a środowiskiem”.

Definicja roju została również opracowana przez organizacje rządowe i militarne. Definicją roju pochodzącą z *Australian Defence Glosary* [18] brzmi – „*the large mass of autonomous systems interoperating collectively to act and respond in a coordinated effort to provide an overwhelming effect*”. Co można przetłumaczyć jako: Rój to (przyp. autor) „duża grupa autonomicznych systemów współpracujących ze sobą, aby działać i reagować w skoordynowany sposób w celu zapewnienia przeważającego efektu”. Innym przybliżeniem terminu może być definicja pochodząca z dokumentu NATO z 2022 roku [145]: „*From a military conflict viewpoint, swarming can be defined as a seemingly amorphous, but deliberately structured, coordinated, and strategic way to strike from all directions simultaneously, by means of a sustainable pulsing of force and/or fire, close-in as well as from stand-off positions*”. Co można tłumaczyć jako: „Z punktu widzenia konfliktu zbrojnego, rój można zdefiniować jako pozornie amorficzny, ale celowo zorganizowany, skoordynowany i strategiczny sposób uderzenia ze

wszystkich kierunków jednocześnie, za pomocą trwałego impulsu siły i/lub ognia, zarówno w pobliżu, jak i z pozycji dystansowych".
Jest do definicja odnosząca się wyłącznie do zastosowań militarnych, oryginalnie zaczerpnięta z pracy naukowej z 2005 roku [8].

Jak widać samo zdefiniowanie roju stało u podstaw prac nad tym zagadnieniom w kontekście tematyki badań naukowych. W słownikach instytucji narodowych najczęściej kategoryzowane jest w zestawieniu z terminami militarnymi. Na potrzeby pracy przyjęto **definicje roju jako zespołu wielu jednostek, działających w zorganizowany sposób, dążących do wspólnego celu, realizując go w kooperacji wzajemnej.**

Znając definicje roju, należy określić jeszcze warunki, jakie powinna spełniać grupa, aby mogła być uznana za rój. W pracach naukowych [167], [30], [153] powtarzającymi się kryteriami są:

- skalarność umożliwiająca koordynowanie niezdefiniowaną z góry ilością jednostek, jak i również dynamiczną zmianę liczby jednostek podczas pracy roju,
- niezawodność w zakresie możliwości kontynuowania pracy w przypadku awarii poszczególnych członków roju,
- elastyczność w postaci możliwości dostosowania roju do zadań o niejednorodnym charakterze w zakresie możliwym do realizacji przez poszczególne jednostki. Należy tu uwzględnić ograniczenia poszczególnych jednostek wynikające np. z dynamiki lub sprzętu,
- adaptowalność, umożliwiającą rojowi przeprowadzenie misji w zmiennych, niekoniecznie znanych warunkach,
- perceptywność w zakresie możliwości czerpania korzyści jednostki od innego uczestnika roju i jego możliwości np. w celu ustalania pozycji przeszkody,
- komunikacje w założonej topologii, umożliwiającą synchroniczne działanie,
- autonomiczność przejawiającą się w możliwości reakcji jednostki zgodnie z jej własnymi nadrzędnymi celami.

Są to ogólne kryteria, które nie poruszają aspektu hierarchii i wynikającej z niej topologii komunikacji. Następnym krokiem pracy jest ustalenie kryteriów podziału koncepcji roju w aspekcie hierarchii.

1.2. Przegląd literatury dla koncepcji sterowania w roju

W celu analizy badań nad rojem należy przyjąć możliwe wariacje koncepcji sterowania w roju oraz kryteria definiujące, które algorytmy na podstawie zachowań obiektów można uznać za algorytm roju. Należy mieć również na uwadze topologie komunikacji obiektów.

Koncepcje sterowań można bazowo podzielić na dwie kategorie:

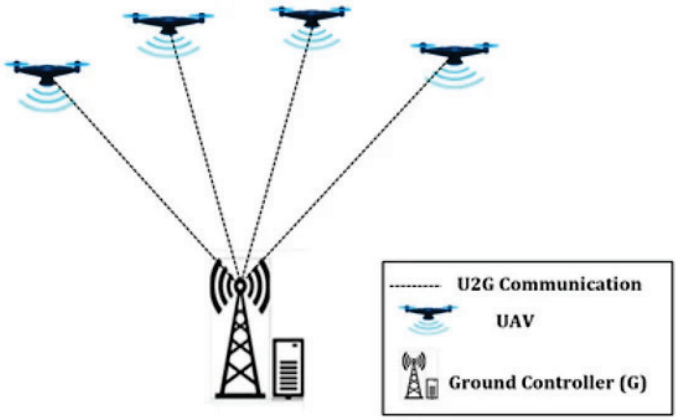
1. Sterowanie w roju wraz z obiektem typu matka roju,
2. Sterowanie w roju bez matki.

W celu rozdzielenia koncepcji ze względu na obecność matki w roju, należy zdefiniować, czym jest obiekt matki. Sama nazwa „matka" jest odniesieniem słownym do modelu roju w przyrodzie, szczególnie dla pszczół lub mrówek. Matka nazywana też bywa królową, zależnie od gromady w jakiej występuje. W stadach, które często rozważane są jako zbliżone do roju, jednostka pełniąca rolę nadrzędną nazywana jest najczęściej „liderem". Pod tą nazwą występuje również w pracach badawczych, które zapoczątkowały próby podjęcia skategoryzowania i podziału koncepcji sterowania w roju.

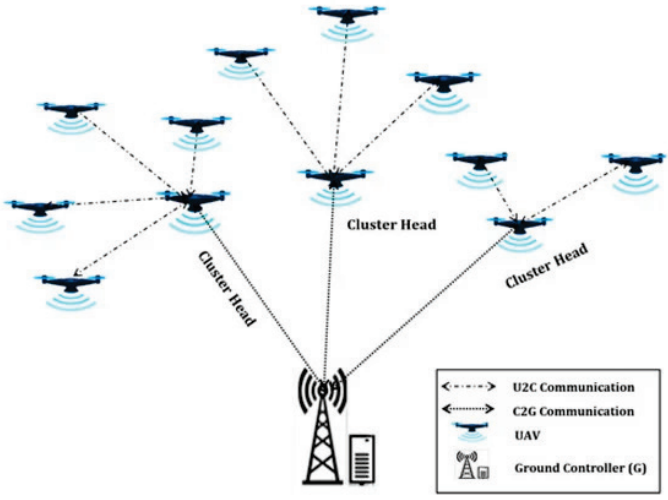
Hierarchię lidera wynikającą z zachowań stadnych opisano w artykule [217]. Autor skupił swoją uwagę na odwzorowaniu topologii, która na wzór naturalnego zachowania zwierząt stadnych, ale również ludzi, zawiera jednostki pośrednie. Przykład struktur społecznych jest z pewnością łatwiejszy do analizy, jednak warto mieć na uwadze, że i wśród zwierząt można wyróżnić hierarchie stadną, korelującą często z pełnionymi w stadzie obowiązkami.

W 2024 roku symulacji komputerowej dla roju dronów z naciskiem położonym na komunikacje i rolę lidera dokonali: Dariusz Marek, Marcin Paszkuta, Jakub Szyguła, Piotr Biernacki, Adam Domański, Marta Szczygieł, Marcel Król i Konrad Wojciechowski. W swojej pracy [136] użyli zapożyczenia z języka komputerowego i nazwali niektóre jednostki jako „hub", co można tłumaczyć w kontekście pracy tych obiektów jako koncentratorów. Ich rolą było, tak samo jak koncentratorów USB, zbieranie informacji i przekazywanie ich w postaci enkapsulowanej wiadomości do lidera. Innym terminem użytym w literaturze naukowej jest „*cluster head*", czyli „kierownik zespołu". Określenie to zostało zaproponowane w pracy

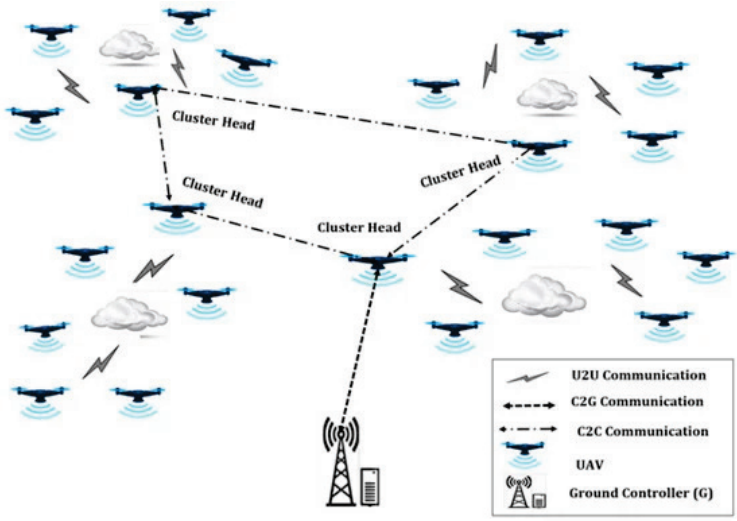
Drone Swarms as Networked Control Systems by Integration of Networking and Computing [15]. Badacze przedstawili trzy główne topologie roju wraz z wyróżnieniem hierarchii. Różnicę pokazują rysunki 1.1 – 1.3.



Rys. 1.1. Koncepcja roju bez interakcji między jego uczestnikami [15]



Rys. 1.2. Koncepcja roju z komunikacją wewnątrz grup bez interakcji między jednostkami przewodzącymi poszczególnym grupom [15]



Rys. 1.3. Koncepcja roju z komunikacją wewnątrz grup wraz z komunikacją między jednostkami przewodzącymi poszczególnym grupom [15]

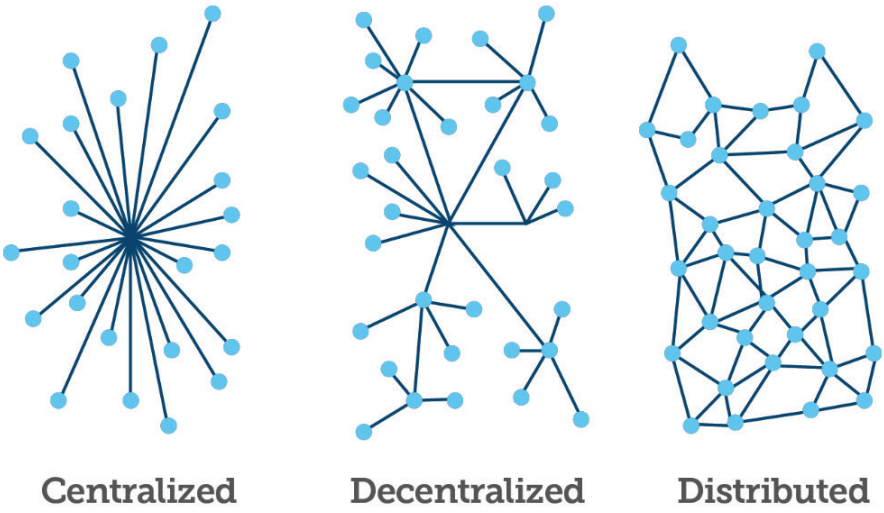
Pierwsza z przedstawionych koncepcji (rys. 1.1) posiada wady w postaci braku możliwości wymiany danych między obiektami w grupie. W przypadku utraty łączności z matką, reprezentowaną na zdjęciu jako stacja naziemna (*Ground Controller*) każda jednostka może liczyć tylko na swoją autonomię, bez możliwości kooperacji z innymi obiektami. Kolejną wadą jest konieczność wydawania poleceń nawigacyjnych przez matkę, bez możliwości weryfikacji przez wzajemnych uczestników roju. W przypadku niedokładności i rozbieżności w ustalaniu pozycji przez jednostkę możliwe jest nieprecyzyjne kierowanie nią przez matkę. W przypadku roju, który miałby pracować przy niewielkich dystansach wzajemnych między dronami, brak jest możliwości implementacji mechanizmów reagowania na potencjalne kolizje.

Koncepcja przedstawiona jako druga (rys. 1.2) obrazuje sieć określaną jako „sieć wielokrastrową". Analogicznie do topologii używanej w sieciach komputerowych, ta koncepcja zakłada wyznaczanie obiektów mogących pełnić rolę pośredników w przekazywaniu danych. Jest to struktura powszechnie używana do organizacji wielu społeczności. Różnica pomiędzy nią a przedstawioną na rys. 1.3. koncepcją sprowadza się do umożliwienia

komunikacji wzajemnej. Przy czym, jak wskazali badacze, ich założeniem jest, aby tylko jeden z „kierowników zespołu" przekazywał informacje do stacji naziemnej, przesyłając dane od wszystkich kierowników poszczególnych zespołów. Zaletą tego rozwiązania jest odporność na sytuacje, gdy jeden z kierowników zespołu nie będzie w stanie nawiązać łączności ze stacją naziemną, natomiast wciąż będzie utrzymywał komunikacje z którąś z innych jednostek zwaną kierowniczą. To rozwiązanie zakłada większe obciążenie łącza w stosunku do koncepcji przedstawionej na rys. 1.2. Wymusza również potwierdzenia wszystkich obiektów o przekazywaniu danych i znacząco dociąża jedną jednostkę, która musi komunikować się za wszystkie inne. Należy mieć na uwadze, że zaleta tego rozwiązania jest wymierna tylko w przypadku niepewności komunikacji na linii kierownik zespołu – stacja naziemna i zapewnieniu w danym momencie potrzebnej komunikacji na linii kierownik zespołu – kierownik zespołu. Innymi słowy w przypadku utraty zasięgu np. sieci komórkowej kierownik zespołu nie może komunikować się ze stacją naziemną, która ma zasięg niezmienny. Rozpatrywany kierownik zespołu z powodu braku własnego zasięgu nie nawiąże także łączności z innym kierownikiem. Stąd w przypadku komunikacji niebezpośredniej, jak wykorzystanie wspomnianego łącza komórkowego, rozwiązanie to nie wnosi pożytku. Ma ono zastosowanie wyłącznie w przypadku dodania fizycznie innej komunikacji tylko do wymiany danych między jednostkami kierującymi lub w przypadku komunikacji bezpośredniej między matką a jednostkami kierującymi i zbyt dalekiemu oddaleniu się jednej z nich. W przypadku przedstawionym na rys. 1.2. każda jednostka zarządzająca komunikuje się bezpośrednio, co zmniejsza opóźnienia i skracając drogę przepływu informacji usprawnia wymianę danych, co jest istotne szczególnie w przypadkach konieczności reagowania dynamicznego np. w nieznanych warunkach, w oparciu o decyzje podejmowane przez matkę roju. W obu przypadkach jest wymagane precyzyjne zdefiniowanie protokołów komunikacyjnych oraz warstw fizycznych używanych do komunikacji, co pozwoliłoby wybrać lepszą z nich.

W roku 2023 swoje koncepcje zaproponowali badacze, którzy rozważyli grupy bez lidera, kolejno bez podziału jednostek [37] i z podziałem

jednostek ze względu na różnice w parametrach [20]. Koncepcja pominięcia lidera rodzi następującą trudność: co lub kto ma podjąć decyzje, reagując na nieplanowaną sytuacje? Koncepcja rozwiązania tego dylematu występuje najczęściej w pracach naukowych pod nazwa „*decentralized brain*", czyli „rozproszony mózg". To również może być nieprecyzyjnym zdefiniowaniem, patrząc na nieodłączną analogie do natury. W kontekście nazewnictwa w języku angielskim można wyróżnić jeszcze trzeci układ określany jako „*distrubuted*" tzn. „rozproszony".



Rys. 1.4. Przedstawienie graficzne modeli topologii w naturze oraz ich anglojęzycznym nazewnictwie [37]

Jak widać tłumaczenie „rozproszony" wskazuje na inny model topologii niż określenie „*decentralized*". Może to budzić wątpliwości i być przyczyną niespójności w pracach badawczych opartej na literaturze anglojęzycznej. Stąd należy podkreślić, że określenie przyjęte w tej pracy jako „mózg rozproszony", dotyczy układów o topologii innej niż scentralizowana bez dalszego różnicowania. W przypadku roju bez jednostki

lidera podejmowanie decyzji może wymagać większej ilości przesłanych danych między obiektami. Sama złożoność procesu może mieć nieliniową strukturę algorytmiczną, w której obecność każdej dodatkowej jednostki znacząco zwiększa listę kroków do wykonania przez całą grupę w celu wypracowania wyniku. Pod znakiem zapytania kierowanym przez naukowców i inżynierów stoi też implementacja tego rozwiązania w warunkach rzeczywistych, gdzie opóźnienia w pracy systemów sterowników lotu mogą nawarstwiać proces podjęcia decyzji, kumulując czas reakcji, który w przypadku jednostek operujących w przestrzeni powietrznej jest priorytetowy [172], [152], [60]. Wzorcowym przykładem implementacji tej koncepcji jest utworzenie systemu generowania odpowiedzi na dylematy, które muszą być rozpatrywane w kontekście całej grupy, tak aby każda jednostka była zaangażowana w znalezienie rozwiązania. Rozważając strukturę w poszukiwaniu analogi, można przyjąć chociażby systemy demokratyczne, które są formą społeczną poszukiwania rozwiązań w charakterze rozproszonego mózgu. Niemniej jednak i w systemach demokratycznych większość decyzji, zarówno w charakterze codziennych oraz tych wymagających niezwłocznych reakcji, jest podejmowanych przez jednostki wybrane do reprezentowania ogółu społeczeństwa.

Z drugiej strony, szukając analogi w królestwie zwierząt, gdzie kolektyw w postaci ilości jednostek jakie mogą tworzyć rój, brak gromady oznacza brak wystąpienia lidera. Należy zauważyć, że nie należy ograniczać koncepcji lidera do jedynie jednostki, która zarządza wszystkimi innymi poszczególnymi samodzielnie. Koncepcje struktury liderów dla dronów przedstawili w swojej pracy Anam Tahir, Jari Böling, Mohammad-Hashem Haghbayan, Hannu T. Toivonen, oraz Juha Plosila [184], gdzie teoretycznym rozważaniom poddali możliwość dzielenia roju na klastry, z których każdy ma swojego lidera, który nie stoi na szczycie hierarchii.

Warto również nadmienić, że w najnowszych pracach badawczych [205], [53] często występuję stwierdzenie „wirtualny lider”, który pełni rolę jednostki lidera, jednak nie wyróżnia się spośród pozostałych obiektów roju. W pracy z 2024 [34] badacze zaproponowali użycie sztucznych sieci neuronowych jako mechanizmu wyboru wirtualnego lidera. Dwa lata wcześniej [215] pojawiła się propozycja algorytmu do wybrania lidera na

zasadzie głosowania przez poszczególne jednostki, której zasadność w kontekście homogenicznej grupy niewielu jednostek może budzić wątpliwości, co do pożyteczności w implementacji.

Jak zostało przedstawione kryterium obecności lidera nie jest podziałem do końca sztywnym. Pomimo, że należy przyjąć, że wszędzie, gdzie występuje jeden obiekt cechujący się prawami do zarządzania i koordynowania wszystkimi jednostkami roju, jest to koncepcja sterowania w roju z liderem lub inaczej mówiąc matką roju. Natomiast tam, gdzie każdy obiekt zarządza sobą i priorytetowo jego instrukcje odnoszące się zadań, które ma wykonywać, nie są wydawane przez inny obiekt, możemy przyjąć, że jest to koncepcja bez lidera. Zależnie czy lider jest traktowany jako wirtualny lider, czy nie można przyjąć następujący zakres zadań dla lidera:

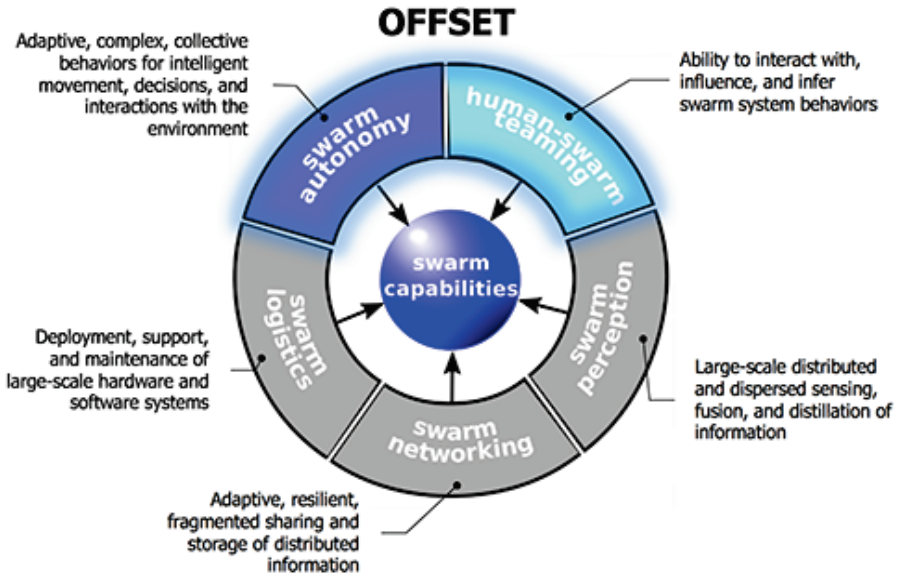
- wydawanie poleceń podrzędnym jednostkom,
- koordynowanie i walidacja wypełnienia misji.

W przypadku wirtualnego lidera nie ma konieczności ochrony lidera przez pozostałe jednostki, gdyż jako jednostka nieposiadająca specjalnych parametrów fizycznych lub sprzętowych może zostać zastąpiona każdą inną w przypadku awarii. W przypadku lidera niewirtualnego należy podkreślić, że nie może być wybrany spośród wszystkich członków roju, gdyż musi posiadać odpowiednie parametry, takie jak np. odpowiednie podzespoły do komunikacji lub nawigacji. W tym przypadku zadaniem, które często pełnią pozostałe jednostki jest ochrona lidera. Istnieją różne metody ochrony. Począwszy od umieszczenia lidera w centrum grupy, jak i również delegowanie, aby lider nie brał udziału w misji, jeśli ta stanowiłaby ewentualne zagrożenie, a niewzięcie udziału nie utrudniłoby komunikacji.

1.3. Analiza implementacji na obiektach fizycznych

Jak zostało wspomniane we wstępie, zagadnienie sterowania w roju bywa rozpatrywane przez niektóre instytucje jako strategicznie ważne i poufne. Analizę można w tym przypadku oprzeć na efektach zaprezentowanych podczas rzeczywistego lotu dronów. Pionierskim nazwać można organizacje DARPA (*Defense Advanced Research Projects Agency*), który została zapoczątkowana w 1958 roku. Organizacja ma charakter badawczy i została powołana przez amerykański departament obrony. Jako impuls do jej powołania przyjmuje się wystrzelenie przez armię sowiecką

Sputnika 1 w 1957 roku. Wielu ekspertów z różnych dziedzin wskazuje na organizacje DARPA jako prekursora, który ukształtował obecną technologię w zakresie komunikacji radiowej i lotnictwa. Od początku misja priorytetowa organizacji, którą jest wsparcie zasobów obrony państwa amerykańskiego została ukierunkowana na technologie lotnicze, poszerzając dziedziny badawcze o dyscypliny korelujące z udoskonalaniem obiektów lotniczych, czyniąc DARPA kompletną jednostką badawczo rozwojową. Wraz z dynamicznym rozwojem technologii bezzałogowych obiektów latających oraz postępem zainteresowań sterowaniem wieloma jednostkami w sposób zorganizowany, organizacja ta rozpoczęła badania nad wdrożeniem technologii sterowania w roju w zakresie możliwości użycia bezzałogowców jako potężnej, nowoczesnej broni. Rozpoczętych zostało wiele projektów. Publicznie głośnym stał się projekt "OFFSET" (*Offensive Swarm-Enabled Tactics*) rozpoczęty w roku 2017. Oficjalnie pracę nad projektem trwały 5 lat. Analizowano w nim różne koncepcje, gdzie użytych zostało 250 dronów. Sam projekt został opisany przez organizacje na ich stronach [46]. Poniżej przytoczony został diagram kołowy, nazwany *Swarm Sprint*, ilustrujący przyjętą strukturę projektu:



Rys. 1.5. Diagram Swarm Sprint obrazujący przyjętą koncepcję sterowania rojem dronów w projekcie OFFSET, na podstawie materiałów DARPA [46]

Przedstawiony diagram ilustruję koncepcję przyjętą w programie OFFSET. Szarymi polami oznaczono podstawowe koncepcje projektu roju takie jak: pozyskiwanie informacji przez jednostki, komunikowanie się wzajemne i przekazywanie informacji oraz założenia techniczne w kwestii utrzymania jednolitego oprogramowania i struktury sprzętowej. Jako wyższym, w hierarchii założeń projektu OFFSET jest utrzymanie autonomii jednostek przy jednoczesnym dodaniu interfejsu dla wielu użytkowników, których rolę można porównać do roli operatorów. Ostatnie założenie można traktować jako nowum, które mogło zainspirować badaczy w 2023 roku [108] do przeanalizowania symulacyjnej koncepcji roju, który byłby sterowany przez operatora. Eksperymenty w programie Matlab potwierdziły zasadność tego typu rozwiązania. Nie ujawniono ilu badaczy, inżynierów i innych specjalistów było zaangażowanych w projekt OFFSET. Podsumowania tego projektu podjęli się w 2023 badacze Timothy Chung i Roshan Daniel [40]. Na podstawie powszechnie dostępnych danych podsumowali całe przedsięwzięcie DAPRA. We wnioskach można przeczytać:
„Chociaż nadal istnieje wiele wyzwań technicznych, aby w pełni zrealizować przewidywane korzyści systemów roju w kontekście operacyjnym, program OFFSET zapewnił zarówno możliwości rozwoju, jak i demonstracji w nowatorski sposób, w tym ścisłe połączenie eksperymentów fizycznych i wirtualnych oraz wprowadzenie nowych technologii „stron trzecich" poprzez model integracji Swarm Sprint".

Oprócz programu OFFSET DAPRA prowadziła również wiele innych programów skupionych wokół tematyki UAV, a także sterowania wieloma jednostkami w tym:

- Projekt Gremlins [45] trwający od 2015 roku, mający na celu rozwój tanich UAV, wielokrotnego użytku, które mogą być wystrzeliwane i odzyskiwane w locie. Program ten koncentruje się na szybkim wdrażaniu rojów UAV z jednego samolotu.
- Program CODE (*Collaborative Operations in Denied Environment*) [44], realizowany w latach 2015-2019, ma na celu umożliwienie współpracy grupom UAV pod autonomiczną kontrolą, zwłaszcza w środowiskach, gdzie GPS i komunikacja są ograniczone.

Jako współpracę świata nauki i militarnych organizacji rządowych można uznać Projekt Perdix [26], aktywny w latach 2013-2018, który jest znany z udanego wdrożenia mikrodronów wystrzeliwanych z myśliwców. Program ten pokazuje praktyczne zastosowanie technologii rojowych w celach rozpoznawczych i nadzorczych. Inicjatywa przygotowania projektu powstała na uczelni MIT (*Massachusetts Institute of Technology*) w 2011 roku. Dwa lata później opieką finansową projekt objęło Ministerstwo Obrony Narodowej Ameryki w zamian za przekazanie wyników prac i utajnienie strategicznych rezultatów badawczych. W założeniu przyjęto, że wszystkie jednostki UAV są takie same, a sam prototyp został zaprojektowany na uczelni MIT. Nawiązanie współpracy miało też miejsce w przypadku innej amerykańskiej jednostki rządowej – Pentagonu, a największymi korporacjami technologicznymi, w tym firmy Google. Projektowi nadano nazwę *Maven*. Project Maven [196] rozpoczęty w 2017 roku, wykorzystuje sztuczną inteligencję i uczenie maszynowe do analizy materiałów wideo z dronów, znacznie zwiększając możliwości UAV w przetwarzaniu i rozumieniu danych zebranych przez roje. Jako biblioteki programowe sztucznej inteligencji zastosowano TensorFlow opracowany i utrzymywany przez firmę Google. Projekt objęto finansowaniem z budżetu sięgającego 7.4 biliony dolarów. Technologia TensorFlow jest powszechnie dostępna jako otwarty kod źródłowy i obecnie jest najczęściej wykorzystywanym narzędziem do wytwarzania oprogramowania sztucznej inteligencji. Jej wykorzystanie w zakresie priorytetowego programu zbrojeniowego wskazuje na jakość tego rozwiązania, ale też pokazuje możliwości zastosowania otwartoźródłowych technologii nawet w tak strategicznych sektorach jak uzbrojenie. Program został utajniony, natomiast użytą w nim technologią pochwalił się rzecznik Google [55] (tłumaczenie cytatu): *„Ten konkretny projekt jest pilotażowy z Departamentem Obrony, aby zaoferować API TensorFlow o otwartym kodzie źródłowym, które może pomóc w rozpoznawaniu obiektów na niesklasyfikowanych danych. Technologia oznacza obrazy do przeglądu przez człowieka i jest przeznaczona wyłącznie do zastosowań nieofensywnych”.*
Z czego można wysnuć wniosek, że sztuczna inteligencja ma pełnić rolę wsparcia dla operatorów, jednak nie zastąpi ich w pełni.

Technologie w zakresie bezzałogowych obiektów latających są najczęściej ściśle tajnymi informacjami, których dane nie są publicznie dostępne. Firmy działające w zakresie UAV oraz rojów są często wcielane do grup zbrojeniowych kraju, na którego terytorium są zarejestrowane. Technologie przygotowywane przez te firmy pomimo możliwości wykorzystania w sektorach cywilnych nie są wykorzystywane w branżach niemilitarnych. Jako przykład może posłużyć rosyjska firma Zala Aero Group działająca w ramach koncernu Kalashnikov, która rozwija technologie rojów dronów, mające teoretycznie zastosowanie zarówno cywilne, jak i wojskowe, jednak nie są stosowane w sektorach cywilnych. Nieco inaczej sytuacja ma się w Chinach, gdzie firmy z tego kraju przeganiają się w pobijaniu rekordów Guinnesa. W 2021 roku firma Genesis wzbiła w powietrze 3281 dronów.

Zdj. 1.2. Logo firmy Genesis utworzone przez 3281 dronów [135]

Zaprezentowane rozwiązanie (zdj. 1.2) nie było jednak rojem, czego dowiódł upadek pojedynczej jednostki. Inne obiekty pozostały na swoich miejscach, nie uzupełniając szyku. Wymienione rozwiązania nie są dostępne w sektorach cywilnych dla użytkowników końcowych. Jak zostało wspomniane, technologicznie wysoce zaawansowane rozwiązania są własnością instytucji rządowych, a technologia użycia dronów w roju w sektorze cywilnym jest wyhamowywana przez narastającą ilość konfliktów

zbrojeniowych i potrzebę adaptacji na warunki militarne. Rozwiązania militarne przekraczają możliwości finansowe użytkowników cywilnych. Nie są również dostarczane do wielu sektorów przemysłu, gdzie drony mogłyby przynieść wymierne korzyści ze względu na możliwość transakcji tych rozwiązań, jedynie na poziomie umożliwiającym ich kontrolowanie do rozpowszechniania i powielania dóbr technologicznych. W pracach badawczych naukowcy wskazują obszary zastosowań roju dronów, które mogłoby być wartością dodaną do następujących obszarów życia:

A. Rolnictwo

Drony mogą być szeroko stosowane w rolnictwie do monitorowania upraw [157] zarządzania wodą i precyzyjnego rolnictwa. Dzięki kamerom i sensorom, drony mogą dostarczać szczegółowych danych na temat stanu zdrowia roślin, poziomu wilgotności gleby oraz obecności szkodników [158]. Zastosowanie roju mogłoby umożliwić większą autonomie i zastosowanie na szerszą niż obecnie skalę.

B. Zarządzanie Kryzysowe

Roje dronów mogą posłużyć do monitorowania obszarów dotkniętych klęskami żywiołowymi, takimi jak trzęsienia ziemi, huragany czy pożary lasów [125]. Umożliwiają szybkie ocenienie sytuacji i identyfikację miejsc wymagających pomocy.

C. Dostawy i Logistyka

Drony obecnie są testowane przez niektóre firmy świadczące usługi dostarczania towarów handlowych do klienta. W zakresie roju mogą stanowić lepsze możliwości transportowe [19]. W 2023 roku badacze Abderahman Rejeb, Karim Rejeb, Steven J. Simske i Horst Treiblmaier pokusili się o przegląd rozwiązań bezzałogowych obiektów latających w kontekście logistyki wewnętrznej dla firm zajmujących duże obszary magazynowe [162]. We wnioskach zawarli tezę (tłumaczenie cytatu): *„W kontekście Logistyki 4.0 [21] istnieje potrzeba ciągłego dostosowywania, angażowania i wdrażania nowych technologii w celu poprawy wydajności systemów logistycznych. Aby to osiągnąć, organizacje mogą wykorzystać możliwości dronów w celu skrócenia czasu i obniżenia kosztów usług”.*

Badacze zwracają jednocześnie uwagę na potrzebę opracowania koncepcji zapewniającej pełne bezpieczeństwo, w kontekście możliwych wzajemnych kolizji, podczas pracy wielu jednostek wykonujących zadania transportowe.

D. Inspekcje i Monitoring

Drony są wykorzystywane do inspekcji infrastruktury krytycznej [100], takiej jak mosty, linie energetyczne, rurociągi i farmy wiatrowe. Mogą dostarczać szczegółowych zdjęć i danych, co pozwala na wczesne wykrywanie problemów i minimalizowanie ryzyka awarii. Zastosowanie algorytmu roju może pozwolić na zoptymalizowanie inspekcji przez synchronizacje danych miedzy jednostkami. Jak wskazują autorzy [100] należy położyć większy nacisk na aspekty operacyjne dla zastosowania wielu jednostek.

E. Ochrona Środowiska

Rój ma również zastosowanie w monitorowaniu stanu środowiska, ochrony dzikiej przyrody [166] i badaniach nad zmianami klimatycznymi. Drony mogą dostarczać danych na temat zanieczyszczeń, zdrowia lasów oraz migracji zwierząt. Kluczowym dla tego obszaru jest kooperacja wielu jednostek w czasie rzeczywistym, pozwalająca dokonywać wyborów, w jakie rejony należy skierować jednostki, aby zebrać tylko najważniejsze dane w celu oszczędności zasobów dronów.

F. Bezpieczeństwo Publiczne i Nadzór

Drony są wykorzystywane przez służby porządkowe do monitorowania tłumów [6], zarządzania ruchem oraz poszukiwania zaginionych osób. Umożliwiają szybkie i efektywne reagowanie na różnego rodzaju zagrożenia. Od roju dronów w tym segmencie wymagane jest autonomiczne działanie, umożliwiające odciążenie operatorów, zmniejszenie ich wymaganej ilości, a także działanie w systemie informowania odpowiednich jednostek w przypadku potencjalnych zdarzeń.

W zależności od sektora publicznego przed algorytmami sterowania w roju stawiane są różne wymagania. Można wyróżnić m.in. potrzebę autonomii, zmniejszenie ilości potrzebnych operatorów, inteligentne zarządzanie zasobami dronów, kooperacje wielu jednostek, możliwość współpracy z operatorami dzięki systemowi powiadomień i przekazywania informacji w trybie asynchronicznym. Ostatni punkt należy rozumieć jako

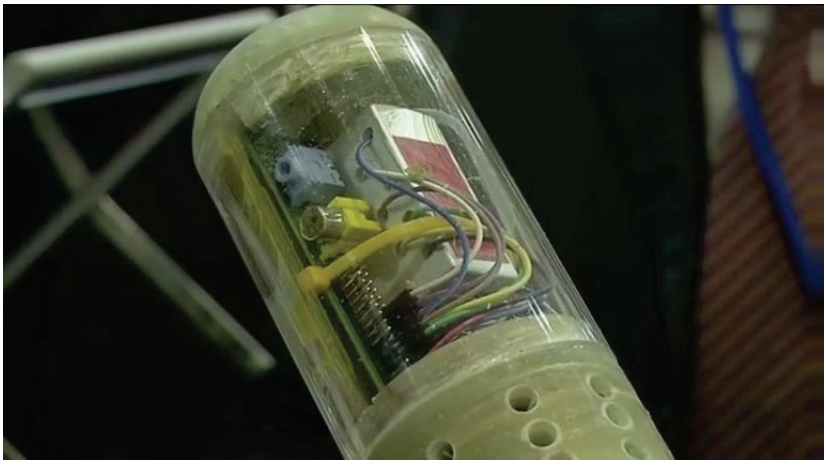
wymianę informacji i zadawanie misji niezależnie, czyli operator może w każdej chwili zadać misje wielu dronom, nie musi tego robić dla każdego z nich z osobna. Natomiast drony jako rój mogą poinformować operatora, gdy wykryją zdarzenie wymagające jego interwencji.

Obecnie na rynku jest bardzo dużo producentów dronów, głównie wielowirnikowców, w tym najpopularniejsze są quadrocoptery, które są do nabycia w przystępnych cenach dla cywilnych użytkowników. Często producenci tych rozwiązań wykorzystują otwartoźródłowe systemy sterowników lotów takie jak: ArduPilot, PX4, BetaFlight. Są one proste do zaimplementowania w dronach tworzonych po to, aby spełniać założenia budżetowe i być seryjnie produkowane. W wymienionych wyżej sektorach cywilnych, takie drony są najczęściej kupowane. Ich atutami są modularność i możliwość doposażenia w adekwatne do potrzeb użytkownika sensory. Natomiast wadą jest trudność wykorzystania wielu jednostek w sterowaniu grupowym, jak np. sterowanie w roju.

Wspomniane drony produkcji masowej znalazły również zastosowanie na froncie wojennym. Rozmiar potrzeb technologicznych w obliczu konfliktu zbrojnego i zapotrzebowanie na bezzałogowe obiekty latające były tak duże, że w Ukrainie w 2022 roku na froncie pojawiły się drony amatorskie. Rozpoczęto zbiórkę dronów znaną pod nazwami kampanii „Armia Dronów" lub jak została nazwana w mediach społecznościowych: „#DronyNaWschód". Celem akcji była zbiórka dronów prowadzona wśród polskiego społeczeństwa, głównie wśród miłośników amatorskiego latania. Informacje o wymaganiach technicznych jednostek, które mogłyby zostać przekazane, zostały podane do informacji publicznej w prasie [197] (cytat):
„Wymagany jest quadkopter z następującymi minimalnymi ustawieniami parametrów: bateria lipo 5000 mAh, zasięg lotu od 1 km, aparat 20 MP, czas ładowania 30 minut, rozdzielczość zdjęć 5280x3956 i rozdzielczość wideo 3840x2160, preferowane są joystick i pilot. Nie ma znaczenia, czy dron jest nowy czy stary, po prostu musi być sprawny".

Celem przekazania tych dronów miało być wykorzystanie ich do rozpoznania terenu i pozycji okupanta. Nie jest to pierwszy raz, gdy technologia, którą powszechnie nazwać można „amatorską", ze względu na jej powszechność, popularność i cenę, miała zostać wdrożona do zastosowań

militarnych. W 2016 roku pojawiły się zdjęcia wyrzutni rakietowej, której jednostką sterującą był minikomputer Raspberry Pi:



Zdj. 1.3. Głowica i moduł naprowadzający przedstawiony na ukraińskich targach broni w 2016 roku [141]

Komentatorzy [141] tego precedensu, jakim jest użycie technologii nazywanej „amatorską", zadają pytanie o odwrócenie trendu transferu technologii z pierwotnego kierunku, od systemów zbrojeniowych do wdrożeń w powszechnym użytku.

Należy zwrócić uwagę na cenę w skali produkcyjnej i ilościowe zapotrzebowanie danego państwa na niektóre technologie. Obecnie drony stanowią fundamentalne narzędzie do walki dystansowej, jednak z racji na budżet, jaki pochłania konflikt zbrojny, wykorzystywane są zarówno jednostki militarne, jak i te amatorskie. Stosowanie jednostek amatorskich wiąże się jednak z nieudogodnieniami w postaci obciążenia operatorów i brakiem możliwości synchronizacji wielu jednostek.

1.4. Podsumowanie analiz

Sterowanie w roju pozwala optymalizować pracę wielu jednostek UAV. Dzięki zastosowaniu algorytmów roju możliwe jest zmniejszenie nakładu pracy człowieka podczas sterowania jednostkami w czasie wykonywania powierzonego im zadania. Ponadto możliwe jest uzyskanie

zadawalających wyników w zakresie utrzymania szyku i reagowania na wydarzenia dynamiczne, a także w przypadku utraty komunikacji.

W przeciągu ostatnich 5 lat nastąpił dynamiczny przyrost liczby badań nad koncepcją sterowania w roju. W sektorze militarnym przeskok cenowy pomiędzy technologiami komercyjnymi, dedykowanymi dla wojska, a amatorskimi sprawił, że prawdziwym stało się zastosowanie technologii amatorskiej na realnym froncie. Nie można jednak mówić w tym zakresie o możliwości sterowania w lepszy sposób niż sterowanie ręczne każdej jednostki z osobna, z uwagi na brak możliwości synchronizacji komunikacji dronów. Stanowi to słaby punkt tego typu rozwiązań. Pojedynczo sterowane jednostki są łatwe do unieszkodliwienia, a także wymagają zaangażowania wielu operatorów.

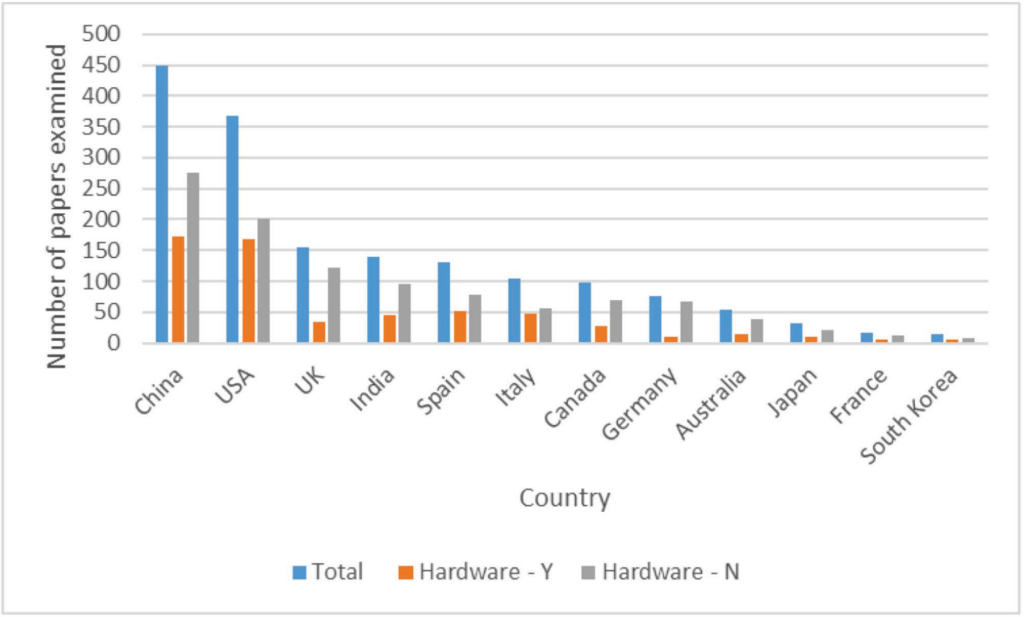
Drony stają się coraz częstszym zapleczem technologicznym wielu firm działającym w sektorze cywilnym. Dają przewagę w zakresie monitorowania, gromadzenia informacji, śledzenia, patrolowania i poszukiwania wybranych celów. Wdrażane są również do celów transportowych, gdzie mogą stanowić szybszą alternatywę od transportu lądowego. Brakuje prac poświęconym tym sektorom skierowanym na wskazanie drogi wdrożenia algorytmów sterowania wieloma jednostkami z pominięciem roli operatorów, a także gwarantujących bezpieczeństwo i bezkolizyjność grupy bezzałogowych obiektów latających.

Wymagania stawiane rojom dronów w sektorze cywilnym, pokrywają się z wymaganiami z sektora militarnego, gdzie następuje największy przyrost technologiczny. Z racji na konflikty zbrojne sektor cywilny jest pomijany. Rozwiązania komercyjne są nieprzystosowane do możliwości adaptacji dla obszaru cywilnego. Pomiędzy możliwościami technologicznymi rozwiązań bazujących na dronach nazywanymi amatorskimi i ich możliwościami w kontekście roju, a rozwiązaniami militarnymi z dedykowanymi dronami istnieje duża dysproporcja. Jest ona nieproporcjonalna do dostępności otwartej technologii i jej wykorzystaniu w rozwiązaniach klasy militarnej. Jako przykład można przywołać użycie bibliotek sztucznej inteligencji TensorFlow oficjalnie potwierdzonych w priorytetowym projekcie roju dronów Pentagonu, czy też użycie modułów Raspberry Pi w głowicach rakiet.

Dostępność dronów, ich popularność, cena, możliwości zachęcają do poszerzania obszarów wykorzystania. Badacze wielu sektorów [140], [6], [100], [161] wskazują na potrzebę użycia roju jako inteligentnego systemu sterującego wieloma jednostkami przy zachowaniu autonomii jednostki i możliwości pracy w trybie nadzorowanym przez pojedynczego operatora.

Zgłębienie tematu sterowania w roju dla bezzałogowych obiektów latających od strony podejścia naukowego jest zagadnieniem złożonym ze względu na interdyscyplinarność [2]. Holistyczne podejście do tematu wymaga zarówno uwzględnienia aspektu odpowiedniego doboru jednostek i ich parametrów, ustalenia hierarchii w roju, mechanizmów podejmowania decyzji [193] komunikacji pozwalającej na nawigacje wzajemną [156] i względem otoczenia [109] oraz wypracowania i przeniesie właściwej dla tego zagadnienia inteligencji do zastosowania w obranych obszarach pracy [140].

Ewolucje w pracach badawczych często pokrywają się z trendami technologicznymi, nieustanie czerpiąc inspiracje u źródeł obserwacji zachowań roju. Poza rozważnymi pszczołami i mrówkami naukowcy przyglądają się wciąż nie w pełni znanym wzorcom zachowań np. gołębi [126]. Mnogość zagadnień naukowych stojących u podstaw stertowania w roju sprawia, że rozpatrywane koncepcje roju nie uwzględniają aspektów sprzętowych [153], [127], [63]. W publikacji z 2024 roku *An analysis of trends in UAV swarm implementations in current research: simulation versus hardware* [154] badacze podjęli temat ilościowej oceny na temat prac dotyczących roju dronów. W swoich badaniach wskazali trendy zbieżne z obserwacjami poczynionymi podczas przygotowania niniejszej pracy.



Rys. 1.6. Zestawienie poszczególnych krajów przodujących w pracach badawczych nad rojem [154]

Materiały zebrane przez badaczy, zestawione w formie graficznej, wskazują jednoznacznie dwa kraje, które prowadzą najwięcej badań naukowych nad sterowaniem w roju. Stany Zjednoczone Ameryki pomimo drugiej lokaty, mają bardzo zbliżony wynik w kategorii możliwych do implementacji rozwiązań. Jako wyjaśnienie i uzupełnienie przekazanych badań należy zestawić rysunki 1.6 i 1.7. Rysunek 1.7 przedstawia ilość badań rok po roku i potencjał w realizacji badań na obiektach rzeczywistych:

Year	Hardware not used	Number of studies that outlined the possibility of hardware experiments
2013	18	0
2014	26	0
2015	33	2
2016	33	7
2017	74	14
2018	105	18
2019	165	11
2020	190	31
2021	210	23
2022	279	19
2023	251	21

Rys. 1.7. Ilościowe zestawienie badań prowadzonych nad rojem dronów [154]

Według przedstawionego zestawienia, jeśli trend ostatniego roku w pracach nad sterowaniem w roju utrzyma się, może nastąpić wyhamowanie wzrostu ilościowego prac w tej dziedzinie, choć i tak tematyka ta stała się niezwykle popularna i wyhamowanie jej wzrostu miałoby miejsce na bardzo wysokim ilościowo poziomie i nie oznaczałoby wyczerpania wiedzy w tym zakresie. Niewątpliwie analizowany w niniejszym rozdziale konflikt zbrojny na świecie i odsłonięcie zapleczy militarnych niektórych mocarstw korelują z przyrostem prac na badaniami sterowania algorytmów roju dla bezzałogowych obiektów latających. Jak podkreślają autorzy należy mieć na uwadze, że symulacja w przypadku badań nad wzajemnymi zachowaniami licznej grupy obiektów jest konieczna i może zapewnić wymagane warunki do przeprowadzenia badań naukowych. Również samo słowo symulacja nie jest spójnie definiowane, a eksperymenty symulacyjne przeprowadzane są z różnym stopniem odwzorowania warunków rzeczywistych.

We wnioskach badacze zawarli problem otwarty (tłumaczenie cytatu): *„Pozostaje jeszcze sprawdzić, czy podejście do walidacji poprzez symulację jest adekwatnym do tego, w jaki dzisiejsza społeczność badawcza postrzega i akceptuje wyniki wydajności roju”.*

Poza wspomnianymi różnicami w stopniu odwzorowań rzeczywistości i walidacjami proponowanych koncepcji istnieje druga potrzeba wynikająca z analiz przeprowadzonych przez twórców artykułu

[154] w postaci lepszego sposobu prowadzenia eksperymentów, aby proponowane koncepcje mogły być uznane za możliwe do realizacji na obiektach rzeczywistych. Jest to wniosek mogący potwierdzać tezę nieholistycznego podejścia do prac nad rojem, nie uwzględniając parametrów jednostki latającej oraz komunikacji w zakresie fizycznych i programowych możliwości w dronach.

Większość prac naukowych rozważających możliwości implementacji roju skupia swoją uwagę na modelowaniu zachowań roju i możliwościach zastosowania określonego typu misji. Naukowcy sami wskazują na potrzebę kontynuacji prac w zakresie zwiększenia bezpieczeństwa i elastyczności algorytmiki sterowania jednostkami UAV w roju, a także na opracowanie architektury komunikacji dla roju [30].

Niezależne grupy badaczy [35], [99] wskazują na potrzeby prac o potencjale wdrożeń z wykorzystaniem nowoczesnych technologii komunikacji. Cytując kolejno [35][99]: *„Chociaż technologia roju nie została jeszcze praktycznie wykorzystana w zastosowaniach komercyjnych, istnieje ogromny potencjał. Wykorzystanie mobilnej struktury komórkowej niweluje czynniki ograniczające tradycyjne podejścia do komunikacji roju UAV”* oraz *„Uczenie maszynowe, sieci 5G i inne technologie złagodzą czynniki ograniczające poprzednie badania i zwiększą wydajność roju oraz jego komercyjne wykorzystanie”*. Dodatkowo autorzy cytowanej pracy wyrażają przekonanie o potrzebie wykorzystania niedrogich systemów i jednostek w celu wdrażania opracowywanych koncepcji w dedykowane sektory: *„Oprócz wielu wspomnianych zalet roju i rozwoju technologicznego, istnieje wiele ważnych i interesujących ograniczeń, które mogą utrudniać działanie roju. Wśród tych ograniczeń, koszt produkcji roju na dużą skalę jest nadal wysoki. Istniejące obciążenia są ogromne, drogie i w większości przypadków nieodpowiednie do osiągnięcia wysokiej wydajności. Dlatego też lekkie i tanie modele ładunków i platform są niezbędne do tworzenia rojów”*.

W najnowszych pracach, w tym w pracy z 2023 roku poświęconej zastosowaniom w sektorze monitoringu publicznego i ratownictwa [102] wskazywana jest potrzeba odpowiedzi na potrzeby rynkowe. Jak określili to badacze Rakesh John, Amala Arokia Nathan, Indrajit Kurmi i Oliver Bimber: *„Wierzymy, że ciągły i szybki rozwój technologiczny sprawi, że duże roje*

dronów staną się wykonalne, przystępne cenowo i skuteczne w najbliższej przyszłości - nie tylko w zastosowaniach wojskowych, ale w szczególności także w zastosowaniach cywilnych, takich jak poszukiwania i ratownictwo".

Rozdział 2

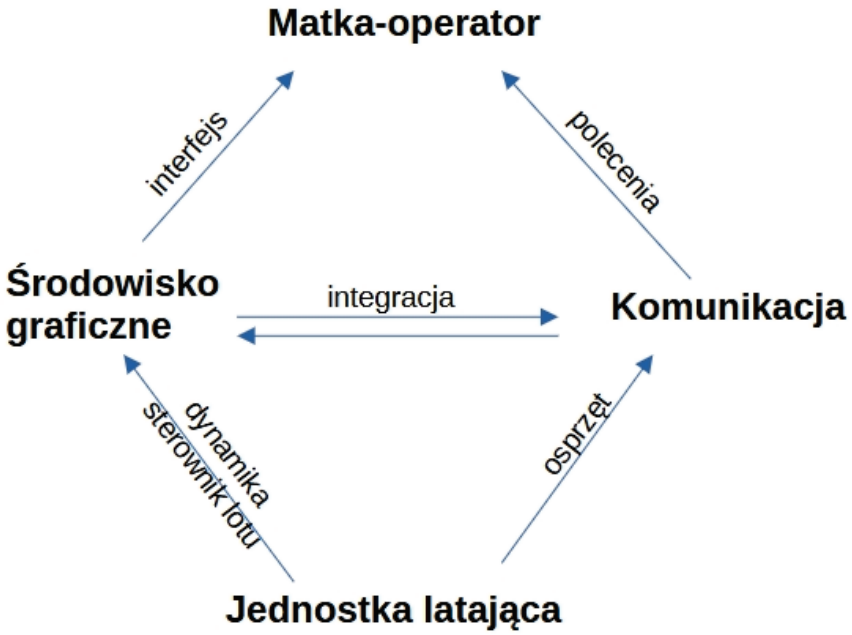
CEL ORAZ ZAKRES BADAŃ

Ze względu na interdyscyplinarność pracy przyjęto modułową strukturę pracy, którą opisano w podrozdziale 2.1. Rozwinięto przyjęte założenia pracy. Przedstawiono główny cel badań. Omówiono szerzej motywacje i idee zaprezentowane we wstępie, argumentując zasadność obrania celu. Opisano ewolucję koncepcji, jaka miała miejsce podczas prac nad rozwiązaniem postawionego celu. Przedstawiono również cele szczegółowe, nakreślając drogę do osiągnięcia celu głównego. Opisano metodykę prowadzenia badań, przyjęte założenia i konwencje służące odpowiedniemu zaprojektowaniu eksperymentów. Na końcu rozdziału przedstawiono strukturę pracy. Podział na rozdziały wraz z omówieniem zawartości koresponduje do argumentacji logiki podejścia do prowadzonych badań oraz sposobu ich prezentacji.

2.1. Modułowa koncepcja pracy

Przygotowanie kompleksowego, wdrażalnego algorytmu sterowania w roju dla bezzałogowych obiektów latających wymaga rozpatrzenia wielu aspektów. Aby móc sterować wieloma jednostkami należy przyjrzeć się dynamice i wyposażeniu danych obiektów. Również w zakresie oprogramowania wymagane jest zdefiniowanie komunikacji na poziomie kompatybilności protokołów umożliwiających wymianę danych. Koniecznym w przypadku zastosowania algorytmów uczenia maszynowego jest przygotowanie środowiska graficznego służącego do symulowania zachowań dronów w bezpiecznych dla jednostek warunkach. Kluczowym jest uzyskanie jak powtarzalności badań przy jednoczesnym precyzyjnym odzwierciedleniu zachowania obiektów w rzeczywistych warunkach.

Dopiero szczegółowa analiza i przystosowanie wymienionych obszarów: jednostki latającej, komunikacji, środowiska graficznego pozwala na wdrożenie algorytmiki inteligentnego sterowania w roju. Opisana zależność została przedstawiona na rysunku 2.1.



Rys. 2.1 Przedstawienie zależności modułów [źródło własne]

Modułem bazowym jest „Jednostka latająca”. Aby umożliwić implementacje komunikacji i dobranie protokołów tak, aby zapewnić bezpieczeństwo jednostkom niezbędnym jest określenie wymagań dotyczących osprzętu, z jakiego mógłby korzystać moduł „Komunikacja”. Na tym samym poziomie, na którym znajduje się moduł komunikacji jest moduł „Środowisko graficzne”. Wymaga on uwzględnienia dynamiki obiektu latającego, a także zachowania struktury sterownika lotu. Jako moduł nadrzędny zdefiniowano część pracy odnoszącą się bezpośrednio do algorytmu sterowania w roju. Nazwa „Matka-operator” zawiera drugi człon, który odnosi się do sytuacji w jakiej nadzorujący pracę roju operator mógłby przejąć kontrolę nad dronami. Jak przedstawia rysunek 2.1 bezpośrednio moduł ten nie bazuje na jednostce latającej, jednak opracowanie komunikacji i środowiska graficznego wymaga uprzedniego zdefiniowania osprzętu, a także dynamiki obiektu oraz sterownika lotu. Należy również zintegrować moduły środowiska graficznego i komunikacji, aby możliwa była symulacja

roju w warunkach oddających rzeczywistość wraz z użyciem przygotowanych protokołów komunikacji. Dzięki temu moduł nadrzędny matki-operatora może za pomocą interfejsu programowego realizować sterowanie w środowisku graficznym. Do przekazywania wiadomości i komend sterujących wykorzystywane są polecenia, będące funkcjami w oprogramowaniu, zdefiniowane w komunikacji.

Aby uporządkować prezentacje wyników badań w niniejszej pracy podział na rozdziały został ujednolicony z modułami zaprezentowanymi na rysunku 2.1. Każdy z rozdziałów stanowi odrębną część badań i realizuje cele szczegółowe pracy. Ich treść została opisana w podrozdziale 2.3

2.2. Cel główny badań

Pracę nad niniejszą rozprawą rozpoczęto przed wybuchem wojny na Ukrainie. Odbiła się ona echem na całym świecie, a szczególnie w Polsce. Założenia poczynione na początku prac, zarówno w zakresie technologicznym, jak i koncepcji zastosowań, adaptowano do bardzo dynamicznej odpowiedzi świata nauki na prowadzony konflikt zbrojny.

Obserwując dokonania badaczy wraz z danymi bibliometrycznymi[202], kierunki badań i trendy[43][99] a także postęp prac koncernów zbrojeniowych, których rozwiązania ujrzały światło dzienne, wyłonił się kierunek prac. Zgodnie z obranym tematem, zaktualizowanym przeglądem literatury oraz potrzebami współczesnych zagadnień sterowania dronami, wyłoniły się założenia techniczne, które w czasie prac były walidowane w odpowiedzi na bieżący stan wiedzy w zakresie sterowania rojem UAV. Jako założenia niniejszej pracy można wyróżnić:

- komunikacje jednostek zgodnie ze strukturą przyjętego roju, umożliwiającą syntezowanie i wymianę danych w całej grupie;
- autonomiczność jednostek i podgrup;
- zapewnienie bezpieczeństwa poszczególnym jednostkom ze szczególnym uwzględnieniem jednostek specjalnych;
- skalarność algorytmu, umożliwiającą dynamiczne przydzielanie jednostek i modyfikowanie podgrup;

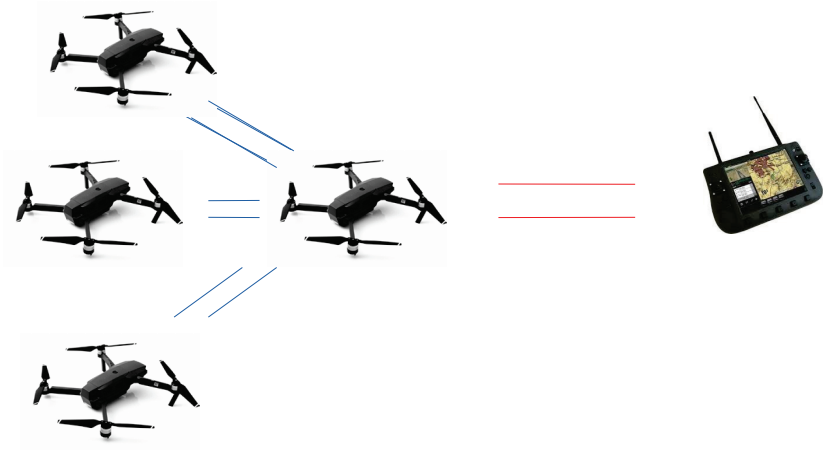
- wielofunkcyjność, tak aby rozwiązanie mogło zostać zaadoptowane w pierwszej kolejności do wymienionych w rozdziale pierwszym sektorów cywilnych;
- powszechność i elastyczność doboru komponentów, aby zastosowanie algorytmu nie zawężało się tylko do obiektów, na których prowadzone będą badania;
- modularność, dotyczącą składowych komponentów oraz modułów sterowania w roju. Dzięki nim istnieje możliwość wdrożenia autorskich oraz istniejących już koncepcji, a finalny efekt pracy może być rozwijany w przyszłości w zakresie poprawy określonych modułów przy pozostawieniu pozostałych, lub w formie wymiany poszczególnego modułu np. protokołu komunikacji na inny, bez konieczności wprowadzania zmian w strukturę algorytmu. Konsekwencją jest zostawanie możliwie wielu technologii otwartoźródłowych, aby umożliwić integracje między poszczególnymi modułami;
- możliwość symulacji, zakładająca, że zachowania roju będą mogły być badane również w symulatorze, który będzie odzwierciedlał warunki rzeczywiste. Symulator umożliwi testowanie inteligencji roju i poprawianie jakości algorytmu bez ryzyka szkód i strat w terenie. Ponadto, będzie służyć jako narzędzie do szkolenia operatorów w zarządzaniu rojem, zapewniając im praktyczne doświadczenie w kontrolowanych warunkach. Symulator umożliwi również przygotowanie i badanie rozwiązań na skalę wielu jednostek, co pozwoli na przetestowanie algorytmu w scenariuszach obejmujących dużą liczbę jednostek i podgrup;
- wdrażalność opracowanej koncepcji sterowania w roju. Zgodnie z przytoczoną w rozdziale 1 pracą oceniającą dotychczasowe postępy w nauce [154] uwagę skierowano, aby podejść do całego zagadnienia holistycznie. Począwszy od jednostek sterujących przez komunikacje do algorytmu. Do uzyskania wdrażalności wymagane było przetestowanie i walidacja środowiska graficznego. Daje to możliwość aby opracowany algorytm wraz z jego inteligencją zwalidować w środowisku symulacyjnym dla adekwatnie dużej

liczby bezzałogowych obiektów latających, przekraczających możliwość sprawdzenia na fizycznym sprzęcie.

Ramy przejętych założeń korespondują do wymagań stawianych rojom dronów, opisanych w rozdziale 1, a także są inspirowane najbardziej zaawansowanymi technologicznie programami, takimi jak przytoczony program OFFSET[46]. Nie są one jednak zbieżne z programami militarnymi ze względu na modułowość, elastyczność koncepcji i otwartość technologii. Założeniem pracy jest poszerzenie zastosowań roju, a nie próba podjęcia konkurencji z opisanymi programami militarnymi.

Celem niniejszej rozprawy doktorskiej było opracowanie algorytmu sterowania grupą bezzałogowych obiektów latających w szyku określanym mianem roju. Jak przedstawiono w rozdziale 1 jest to bardzo dynamicznie rozwijająca się dziedzina, o czym świadczą liczne publikacje naukowe. Innowacja w proponowanym w niniejszej pracy rozwiązaniu, opiera się na modularyzacji i enkapsulacji opracowanej koncepcji do poziomu niepodzielnych, możliwie atomowych rozwiązań, umożliwiających implementacje w różnych dziedzinach życia.

Inspiracją, a zarazem obrazowym przykładem stały się koncepcje wdrożeń technologii komputeryzacji. Analogicznie do struktury komputera, który jest złożony z podzespołów, mogących pochodzić od różnych producentów, przyjęto model umożliwiający zastąpienie poszczególnego komponentu innym. Zarówno w zakresie komponentów sprzętowych, jak i programowych. Obrana koncepcja roju jest wzorowana na programie OFFSET i można ją określić, jako wariant z matką/liderem, w którym rolę matki roju może pełnić stacja naziemna, obsługiwana i monitorowana przez operatora. Taką koncepcję przedstawia poniższa ilustracja (rys 2.2).



Rys. 2.2. Ilustracja koncepcji roju oraz wymuszonej struktury komunikacji [źródło własne]

Struktura komunikacji wymaga implementacji jej dwóch typów. Oznaczonej kolorem czerwonym – komunikacji daleko zasięgowej oraz niebieskim – komunikacji między dronami (rys 2.2). Przyjęta koncepcja roju pozwala na implementacje w sektorach cywilnych i militarnych. Jest uniwersalna i może spełniać założenia modułowości. Nowatorskim podejściem jest opracowanie koncepcji komunikacji, umożliwiającej elastyczny dobór algorytmów sterowania roju dla wymiennej grupy dronów.

2.3. Cele szczegółowe

Zagadnienie zastosowania roju zostało podzielone na poszczególne moduły, starając się zachować możliwie modularną i przenośną strukturę pracy. Ma to na celu wypracowanie koncepcji, która mogłaby w przyszłości zostać udoskonalona w obszarze tylko poszczególnego modułu. Dzięki takiemu podejściu nawet w przypadku dezaktualizacji poszczególnych koncepcji modułów, lub konieczności implementacji innej technologii całość pracy może zostać adaptowana do nowych warunków. Tak jak w przypadku modularyzacji powszechnie stosowanych obiektów, maszyn, czy pojazdów, gdzie możliwe jest dostarczenie podzespołów dla całego rozwiązania w postaci części zamiennych. Aby zachować koncepcje modułową przy jednoczesnym, maksymalnym skupieniem się na optymalizacji celu

głównego, zastosowano połączenie powszechnie dostępnych technologii wraz z autorskimi rozwiązaniami, *co czyni całość innowacyjną.*

Ze względu na modułowość np. w zakresie doboru fizycznej warstwy do komunikacji lider – matka roju przyjęto topologie zbliżoną do przedstawionej w rozdziale 1 koncepcji roju z komunikacją wewnątrz grup bez interakcji między jednostkami przewodzącymi poszczególnym grupom przedstawioną na rysunku 1.2. Ta koncepcja jest najbardziej zbieżna z założeniem elastyczności zarówno w adaptacji do charakteru misji, jak i możliwości zmiany topologii jedynie w zakresie opracowanej komunikacji, bez ingerencji w inne moduły pracy.

Poszczególne moduły stały się również celami podrzędnymi pracy, wśród których można wyróżnić:

1. Jednostka latająca – ten moduł dotyczy bezzałogowych obiektów latających. Nacisk pracy został położony na możliwość wymiany poszczególnych jednostek, ale również w zakresie wymiany komponentów na pokładzie konkretnej jednostki. Jest to odpowiedź na założoną elastyczność implementacji. **Celem szczegółowym tego modułu było:**

- **wypracowanie koncepcji jednostki będącej uczestnikiem roju,**
- **określenie zadań, jakie mają być realizowane po stronie obiektu,**
- **przygotowanie modelu fizycznego i matematycznego dla przykładowej jednostki,**
- **opracowanie sterownika lotu, weryfikacja powszechnie dostępnych rozwiązań,**
- **ustalenie minimalnych wymagań, aby jednostka mogła brać udział w grupie typu rój.**

Na potrzeby powyższych celów opracowano model fizyczny, korespondujący do bezzałogowego obiektu latającego użytego w dalszych krokach pracy. Podjęto również temat wykonania autorskiego prototypu sterownika lotu i walidowano to rozwiązanie na tle popularnie używanych jednostek sterujących. Wyznaczono poszczególne wymogi dotyczące jednostki z rozważeniem możliwości lidera podgrupy i jego specjalnych wymagań. Określono również autonomię jednostki i procesy, za jakie odpowiedzialny jest sterownik poszczególnego obiektu.

2. Środowisko graficzne – zagadnienia poruszane w tym module koncentrują się na realizacji dwóch potrzeb. Najpierw dostarczenia operatorowi możliwości testowania pracy roju i nauki operatora, a także możliwości nauki zachowań autonomicznych przez rój z wykorzystanie sztucznej inteligencji. Założeniem prac jest możliwość implementacji modelu CAD, odpowiadającemu fizycznemu dronowi.

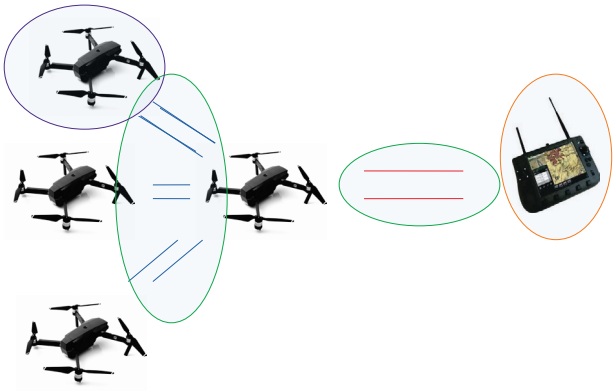
Celem szczegółowym tego modułu było wypracowanie rozwiązania pozwalającego na symulacje i wizualizacje obiektów fizycznych w warunkach odpowiadających rzeczywistym warunkom, aby móc prowadzić zaawansowane eksperymenty na roju dronów w środowisku komputerowym w maksymalnie odpowiadającym warunkom zastosowań w różnych sektorach. Na potrzebę walidacji tego rozwiązania przeprowadzono porównanie modelu matematycznego versus quadrocoptera w środowisku graficznym w zestawieniu z danymi zebranymi z latającego obiektu. Środowisko graficzne stanowi bazę prowadzenia eksperymentów na szeroką skalę, z użyciem parudziesięciu jednostek i implementacji algorytmów uczenia maszynowego w celu zastosowania inteligentnego roju.

3. Komunikacja – w celu zarządzania z poziomu roju wieloma jednostkami, które według założeń pierwszego modułu mogą być wymienne, należało dobrać metody komunikacji niezależne od modelu jednostek. Zgodnie z rysunkiem 2.2 to zagadnienie zostało podzielone na dwie komunikacje: pomiędzy matką-operatorem a liderem, oraz pomiędzy liderem a pozostałymi jednostkami.

Celem szczegółowym tego modułu było opracowanie komunikacji niezależnej od typu fizycznych obiektów, która pozwoli uniknąć wzajemnych kolizji przy jednoczesnym utrzymaniu i korygowaniu szyku roju. Również określono przepływ danych w roju. Zdefiniowano wzory zachowania dla badanych obiektów w przypadku zakłóceń komunikacji. Podano przykłady komunikacji w grupie i zachowania autonomii na poziomie poszczególnej grupy. Omówiono mechanizm podziału obiektów na grupy. Zwrócono uwagę na sytuacje awarii poszczególnej jednostki i mechanizm reakcji grupy.

4. Matka-operator – jako warstwę nadrzędną sterowania rojem ustanowiono stację roboczą nazwaną matką- operatorem lub matką roju. Zgodnie z kierunkiem badań wskazywanych jako przyszłościowe, w badaniach nad sterowaniem rojem, użyto algorytmów sztucznej inteligencji. Ponadto, ukierunkowano pracę na potrzeby współczesnego przemysłu, który wykazuje zapotrzebowanie na koncepcje sterowania wieloma obiektami przez pojedynczego operatora.

Celem szczegółowym tego modułu było przygotowanie oprogramowania pełniącego rolę matki roju, tak aby mogło zostać ono wykonywane przez komputer pełniący rolę stacji naziemnej. Dodatkowo uwaga skierowana jest na to, aby umożliwić obsługę i diagnostykę roju pojedynczemu operatorowi z pomocą sztucznej inteligencji. Na potrzeby roju dobrano algorytmy uczenia maszynowego, które mogą być wdrażane przez wielokrotne testowanie w środowisku graficznym, optymalizując zachowania roju.



Rys 2.3. Oznaczenie modułowości roju [źródło własne]

Próbując zobrazować podział na moduły, można oznaczyć poszczególne punktu na Rysunku 2.3. Poszczególne kolory odpowiadają:

- fioletowy – moduł jednostki latającej,
- zielony – moduł komunikacji,
- pomarańczowy – moduł matki-operatora.

Moduł środowiska graficznego jest trudny do zobrazowania, ponieważ jest podłożem do prowadzenia eksperymentów. Założeniem wszystkich modułów jest pełna wymienność, przez co należy rozumieć, że w przypadku dobrania innego modułu np. innej jednostki latającej, możliwe jest zachowanie całej struktury roju. Rolę łączników między modułami reprezentowanymi przez fizyczne obiekty, takimi jak matka i jednostki latające, pełnią moduły komunikacji i środowiska graficznego. Na potrzeby integracji dostępnych rozwiązań sprzętowych w postaci komputerów PC mogących pełnić rolę stacji naziemnej i gotowych jednostek latających, większy nacisk pracy został położony na pracę łączników, aby była w pełni zoptymalizowana na potrzeby sterowania w roju. Poszczególną zależność modułów przedstawiona została na rysunku 2.1.

2.4. Zakres badań i struktura pracy

Modułowa koncepcja pracy stwarza również potrzebę walidacji każdego modułu na miarę możliwości, niezależnie. Należy przez to rozumieć, że chcąc dokonać wymiany lub udoskonalenia pojedynczego modułu, nie jest wymagana zmiana w innych modułach. Na przykład podczas prac nad modułem matki nie ma konieczności zmian w metodyce komunikacji przy zastosowaniu innej algorytmiki sterowania w roju. Wiąże się to z określeniem wymagań każdego modułu oraz jego ograniczeń. Nowy algorytm matki nie mógłby zadać komendy, która nie zostałaby dostarczona do jednostki lidera grupy przez warstwę komunikacji. Warstwa komunikacji musi być możliwa do zaimplementowania w środowisku graficznym, aby umożliwiła pracę z wieloobiektowym rojem.

W celu zachowania i walidacji modułowości przyjęto koncepcje pracy podobną do koncepcji Test-Driven Development (TDD). Polega ona na założeniu przygotowania dużej ilość małych eksperymentów i walidacji efektu końcowego przez sprawdzenie wszystkich poszczególnych eksperymentów pełniących rolę testów. Przy czym jako „test" w programowaniu należy rozumieć test jednostkowy, który sprawdza poszczególną funkcjonalność w kodzie podzielonym na moduły oraz komunikacje między modułami. Tak jak w przytoczonym przykładzie modyfikacja w module matki, która sama w sobie byłaby poprawna,

natomiast nie byłaby możliwa do przekazania do lidera grupy, mogłaby prowadzić do błędu całego roju. Przyjęta koncepcja umożliwia wychwycenie tego rodzaju błędu i rozbudowaną diagnostykę oprogramowania.

2.4.1. Planowane eksperymenty

W kontekście pracy przyjęto walidacje każdego modułu przyrostowo. Oznacza to, że dla każdego modułu zaplanowane zostały eksperymenty, których poprawne przejście odpowiada za uznanie sukcesu w prowadzeniu badań. Poniżej przedstawiono plan badań dla każdego rozdziału.

Dla modułu jednostki latającej przewidziano:

- sprawdzenie opracowanego modelu fizycznego i odpowiadającego mu modelu matematycznego UAV w oprogramowaniu Matlab/Simulink w symulacji wzlotu drona na zadaną wysokość przy użyciu nastaw regulatorów dobranych za pomocą narzędzi optymalizujących oprogramowanie symulacyjne. Wykonanie bardziej zaawansowanego scenariusza w celu weryfikacji symulowanego sterownika lotu w innych warunkach niż te, do których został zoptymalizowany.

Dla modułu środowiska graficznego zaplanowane zostały eksperymenty:

- walidacja wzlotu jednej jednostki latającej reprezentowanej przez modele matematyczne, programy do symulacji, fizyczną jednostkę i porównanie danych zebranych z eksperymentów z danymi arbitralnymi zebranymi z niezależnego zewnętrznego czujnika, odczytującego wysokość jednostki fizycznej podczas wzlotu,
- ponowienie eksperymentów w celu bardziej kompleksowej walidacji wyników.

Dla modułu komunikacji przewidziano:

- test komunikacji wzajemnej, zaimplementowanej do oprogramowania sterowników lotu. Eksperyment zakłada awarię jednej z 4 jednostek i zebranie odpowiedzi od pozostałych z wykorzystaniem fizycznych sterowników lotu,
- przelot szyku liniowego 4 dronów z awarią jednej jednostki podczas misji. Sprawdzenie odpowiedzi obiektów w środowisku graficznym.

Dla modułu matki-operatora przewidziano:

- ponowienie eksperymentu z poprzedniego rozdziału z użyciem inteligencji programu sterującego matki-operatora w celu optymalnego wyznaczenia trasy dla jednostki, która musi wykonać dynamiczną korektę podczas pracy roju,
- przelot dronów w trakcie którego drony natrafiają na niezaplanowaną przeszkodę wymuszającą podzielenie roju,
- ponowienie poprzedniego eksperymentu w innym szyku grupy,
- kompensacje szyku roju przez powrót do pierwotnej formacji,
- wymuszenie dynamicznej reakcji jednostek spowodowaną natrafieniem na nową przeszkodę w momencie kompensowania szyku.

2.4.2. Struktura pracy

Struktura pracy odzwierciedla podział na moduły. Dla lepszego zrozumienia pracy nazwy rozdziałów są jednobrzmiące z nazwami modułów.

Rozdział 3 zawiera model matematyczny bezzałogowego obiektu latającego w postaci quadrocoptera, który został obrany jako model UAV do badań w niniejszej pracy. W rozdziale został opisany sterownik lotu pod kątem sprzętowym z omówieniem minimalnych wymagań dotyczących sensoryki drona, a także pod kątem oprogramowania, które umożliwiałoby jego implementacje.

Rozdział 4 zawiera opis wybranych pakietów i programów do przygotowania środowiska graficznego. Omówione zostały metody implementacji obiektów latających, a także przeniesienie oryginalnej komunikacji do środowiska graficznego. Przygotowany został model CAD rzeczywistego drona, służący do porównania wyników ze środowiska graficznego z fizycznym quadrocopterem oraz modelami matematycznymi.

Rozdział 5 zawiera opis protokołów komunikacji. W celu zapewnienia maksymalnej jakości i możliwości sterowania w roju jednostek uznawanych za amatorskie, utworzono *autorski protokół komunikacyjny* do zastosowań wewnętrznych roju. Przedstawiono metodykę komunikacji oraz omówiono wzajemne pozycjonowanie wewnątrz roju i związane z tym kwestie bezpieczeństwa.

Rozdział 6 zawiera diagram oprogramowania służącego do kontroli nad rojem. Przedstawiono w nim implementacje algorytmów uczenia maszynowego wykorzystanego do kontrolowania roju z poziomu stacji matka-operator. Omówiono inteligencje roju i wykorzystanie podległych warstw: komunikacji i jednostki latającej do celów realizowania zadanych misji.

Rozdział 7 stanowi podsumowanie i syntezę prac ze wskazaniem możliwości rozwoju koncepcji zastosowania opracowanego algorytmu sterowania w roju dla dronów.

Rozdział 3

JEDNOSTKA LATAJĄCA

Zgodnie z modułową koncepcją, przyjęto, aby jednostka latająca mogła być wymienna w zależności od potrzeb, jakie niesie ze sobą wykonanie zadanej misji. Aby zachować możliwość zmiany obiektów, bez potrzeby zmian w innych modułach, wymagane jest wyznaczenie wymagań sprzętowych i programowych dla dronów.

3.1. Wymagania stawiane jednostce

W związku z modułowością przyjęto założenie, że obrany obiekt latający może zostać wymieniony na inny. Wiąże się to z dwoma wymaganiami: dynamicznymi i sprzętowymi. Dynamika obiektu dla jednostek typu wielowirnikowców może być uznana jako zbliżona niezależnie od ilości silników [150], [170], [14]. Jednak różni się od innych jednostek latających takich jak np. VTOL[207] – *Vertical Take-Off and Landing,* czyli (obiekt) poziomego wzlotu i lądowania, helikopter [143]. Rozpatrywanym dronem na potrzeby pracy jest quadrocopter, z racji na powszechność tego typu obiektu w wielu sektorach. Również atuty tego typu jednostki w postaci niezawodności [4] oraz możliwości dynamicznego kierunku zmiany lotu, czynią quadrocopter znakomitym wyborem dla wdrażania koncepcji roju. Jednak należy mieć na uwadze, że dla niektórych typów misji, takich jak szybkie patrolowanie, obiekty typu VTOL mogą być najlepszym wyborem i koncepcja modułowa umożliwia realizacje zmiany typu jednostki.

Quadrocopter jest również najczęstszym obiektem badań naukowych i jest doskonale znany [97], [7]. Naukowcy przyjmują, że najczęściej występującym modelem quadrocoptera jest obiekt, którego silniki rozpięte są na rogach kwadratu [189]. W pracy założono również, że cały rój będzie jednorodny, to znaczy wszystkie jednostki są takie same pod kątem budowy i dynamiki. Dopuszczalne różnice mogą wynikać z odrębnego osprzętu, który może wpływać nieznacznie na parametry dynamiczne i masę obiektu.

Aby skompensować ewentualne dysproporcje przyjęto, że bazowy obiekt może poruszać się z prędkością równą 90% maksymalnej prędkości. Daje to bufor dla obiektów, które muszą utrzymywać większą masę np. przez dodatkowe czujniki. Masa dodatkowych czujników nie powinna przekraczać 3% masy drona, aby była uznana za pomijalną. Większy bufor (w postaci 10% dla prędkości) jest przewidziany dla ogólnego bezpieczeństwa wszystkich jednostek i w przypadku zagrożenia jednostki, może on zostać przez nią wykorzystany.

W przyjętej koncepcji wśród jednostek roju należy wyróżnić jednostki podrzędne i jednostki mogące pełnić rolę lidera podgrupy. Lider pozostaje w ramach homogeniczności roju tą samą jednostką pod kątem konstrukcji, gabarytu i elementów wykonawczych. Wymagane jest lepsze oprzyrządowanie komunikacyjne, aby wymieniać informacje z matką-operatorem i adekwatny do rozszerzonej komunikacji sterownik lotu. Poniżej przedstawiono wymagania dla oprzyrządowania komunikacyjnego, oprzyrządowania nawigacyjnego, potencjalnych dodatkowych czujników a także dla sterownika lotu w zakresie oprogramowania i fizycznego sterownika. Pod każdym z nich przedstawiono również adekwatne poszerzone wymagania dla jednostek dedykowanych do pełnienia roli liderów.

Analiza dynamiki została zawarta w podrozdziale 3.2. W pracy przyjęto koncepcje, aby model matematyczny oddawał przyjęte założenia sprzętowe odnośnie baterii układu silników i śmigieł. Pozostałe wymagania można nazwać założeniami oprzyrządowania. Są one niezależne od modelu. Zalicza się do nich: sterownik lotu, oprzyrządowanie nawigacyjne, oprzyrządowanie komunikacyjne, dodatkowe czujniki.

3.1.1. Fizyczne oprzyrządowanie do komunikacji

Fundamentem pracy wielu jednostek w roju jest ich wzajemna komunikacja. Wymagane jest, aby przepływ informacji był szybki, jak i również obsługa komunikacji nie obciążała mocy obliczeniowej sterowników lotu. Dodatkowo wymagane jest, aby lider mógł komunikować się z matką-operatorem.

Standardowo w wyposażeniu sprzedawanych dronów znajduje się moduł do komunikacji radiowej z operatorem, aby ten mógł sterować dronem. W zależności od jednostki dodatkowym standardem jest przesyłanie obrazu FPV – *First Person View*, czyli „widok pierwszoosobowy". Obie formy wymiany danych opierają się na komunikacji radiowej.

Do przesyłania obrazu z kamery drona (w systemach FPV) również używa się fal radiowych, najczęściej w paśmie 5.8 GHz, które oferuje większą przepustowość, choć na krótszym dystansie. W zależności od częstotliwości i mocy nadajnika, komunikacja radiowa może obejmować zasięg od kilku metrów do wielu kilometrów. Fale radiowe mogą być tłumione przez przeszkody, takie jak ściany, drzewa lub budynki, co może ograniczać zasięg.

Obecnym trendem w kierunku badań nad komunikacją w bezzałogowych obiektach latających jest zastosowanie modemów komórkowych [175], [10], [208], [176]. Technologie Internetu Rzeczy oparte o rozwiązania sieci komórkowych, wskazywane są przez naukowców jako przyszłość rozwoju dronów i ich komunikacji.

Komunikacja IoT oparta na sieciach komórkowych (*IoT cellular*) to technologia, która wykorzystuje sieci komórkowe, takie jak m.in. LTE, 4G, 5G, do łączenia urządzeń Internetu Rzeczy IoT z Internetem i z sobą nawzajem. W kontekście dronów, szczególnie tych operujących na dużych odległościach, komunikacja komórkowa oferuje szereg zalet, które czynią ją idealnym rozwiązaniem do dalekiego przesyłania informacji. Dron wyposażony w moduł komórkowy łączy się z najbliższą stacją bazową operatora sieci komórkowej. Stamtąd dane są przesyłane przez sieć operatora do chmury, Internetu lub bezpośrednio do użytkownika, który może znajdować się w dowolnym miejscu na świecie.

Dzięki sieciom komórkowym dron może przesyłać dane telemetryczne, wideo na żywo, zdjęcia oraz inne informacje do operatora lub do serwerów w chmurze praktycznie bez ograniczeń odległościowych, dopóki znajduje się w zasięgu sieci komórkowej. Jednym z największych atutów sieci komórkowych jest ich zasięg. W przeciwieństwie do tradycyjnych systemów radiowych, które mają ograniczony zasięg, komunikacja komórkowa pozwala na operacje dronów na bardzo dużych

odległościach, nawet setkach kilometrów, bez utraty łączności. Sieci komórkowe są zbudowane z myślą o niezawodności, co jest szczególnie ważne w krytycznych aplikacjach, gdzie przerwanie komunikacji mogłoby prowadzić do utraty drona lub zakończenia jego misji. Dzięki redundancji i zaawansowanym mechanizmom zarządzania siecią, połączenia komórkowe są stabilne i mogą dostosowywać się do warunków środowiskowych. Nowoczesne sieci komórkowe, zwłaszcza 4G LTE i 5G, oferują wysoką przepustowość, co umożliwia przesyłanie dużych ilości danych w czasie rzeczywistym, takich jak wideo w jakości HD lub 4K. Jest to istotne w dronach stosowanych do monitoringu, inspekcji lub transmisji na żywo. Drony wyposażone w moduły *IoT cellular* mogą łatwo integrować się z usługami chmurowymi. Dane mogą być na bieżąco przesyłane do chmury, gdzie są analizowane, przechowywane lub udostępniane innym systemom. To umożliwia zaawansowane analizy w czasie rzeczywistym i szybkie podejmowanie decyzji, ale także archiwizację i monitoring całego roju.

Chociaż sieci komórkowe mają duży zasięg, są miejsca, takie jak odległe obszary górskie, czy środowiska morskie, gdzie zasięg może być ograniczony lub nieobecny. W takich sytuacjach konieczne mogą być alternatywne metody komunikacji.

Komunikacja IoT oparta na sieciach komórkowych jest idealnym rozwiązaniem dla dronów operujących na dużych odległościach, oferując szeroki zasięg, stabilne i niezawodne połączenia oraz wysoką przepustowość danych. Jest to szczególnie korzystne dla zastosowań komercyjnych, przemysłowych i rządowych, gdzie ciągła łączność i możliwość przesyłania dużych ilości danych w czasie rzeczywistym są kluczowe. Pomimo pewnych wyzwań, takich jak koszty i opóźnienia, komunikacja *IoT cellular* staje się coraz bardziej popularnym wyborem w zaawansowanych systemach dronów.

Poza komunikacją z matką-operatorem, która jest realizowana tylko przez drona będącego liderem, wymagana jest również komunikacja wewnętrzna dronów w roju. W komercyjnie sprzedawanych dronach brakuje domyślnie zastosowanych modułów przystosowanych do tego celu. Komunikacja IoT oparta na sieciach komórkowych między poszczególnymi dronami jest złym rozwiązaniem ze względu na niepotrzebne skomplikowanie drogi przepływu wiadomości. W tym rodzaju komunikacji,

jak zostało wyżej przedstawione, dane przesyłane są przez szereg punktów pośrednich, z którymi musi skomunikować się moduł. Wydłuża to drogę wiadomości oraz tworzy niebezpieczeństwo braku zasięgu. Należy wykorzystać komunikacje bezpośrednią. Przedstawione wyżej rozwiązanie oparte na komunikacji radiowej jest rozwiązaniem bezpośrednim, jednak stosowanym powszechnie w komunikacji jeden do jednego. W przypadku modelu roju topologia komunikacji to wiele do wielu. Jest to bardzo obciążające dla sygnału, który musi być modulowany. Duża ilość danych przesyłana w małych, ale bardzo częstych pakietach, przez wiele jednostek może wpływać na wzajemne zakłócenia transmisji wewnątrz roju.

W związku z powyższym należy użyć innej technologii. Obecnie dynamicznie rozwijana jest technologia UWB – *Ultra-Wideband*. To technologia komunikacji radiowej, która wykorzystuje bardzo szerokie pasmo częstotliwości do przesyłania danych. Jest szczególnie ceniona za swoją zdolność do precyzyjnego pomiaru odległości oraz niskiego poziomu interferencji z innymi technologiami radiowymi. W kontekście komunikacji między dronami wewnątrz roju, UWB oferuje unikalne zalety, które mogą znacząco poprawić synchronizację, koordynację i autonomię dronów. Naukowcy wskazują na potencjalne korzyści zastosowania UWB jako technologii cechującej się niskimi zakłóceniami [195]. Dodatkowo technologia ta może służyć zarówno do komunikacji i nawigacji [36].

UWB operuje w paśmie o szerokości od 500 MHz do 8 GHz, co pozwala na przesyłanie krótkich impulsów radiowych. Dzięki szerokości pasma, sygnał UWB może być bardzo precyzyjnie lokalizowany w czasie, co jest kluczowe dla dokładnych pomiarów odległości.

Jednym z głównych zastosowań UWB jest precyzyjny pomiar odległości między urządzeniami. Dla dronów UWB może być używane do mierzenia odległości między jednostkami w roju z dokładnością do kilku centymetrów, co pozwala na precyzyjne utrzymywanie formacji i unikanie kolizji. UWB działa przy bardzo niskiej mocy i szerokim paśmie, co sprawia, że jego sygnały są mniej podatne na zakłócenia od innych systemów radiowych. W środowisku, gdzie może być wiele różnych źródeł sygnału, jak np. Wi-Fi, czy Bluetooth, UWB minimalizuje ryzyko interferencji, co jest kluczowe dla stabilnej i niezawodnej komunikacji w roju. Chociaż UWB nie

jest zaprojektowane z myślą o przesyłaniu dużej ilości danych [118] doskonale sprawdza się w wymianie krótkich komunikatów między dronami. Tego typu komunikacja jest wystarczająca do synchronizacji i koordynacji ruchów dronów w roju, a także do przekazywania istotnych informacji dotyczących stanu systemu czy warunków otoczenia.

UWB operuje przy niskim zużyciu energii, co jest istotne dla dronów, które muszą oszczędzać baterię na lot i inne krytyczne funkcje. Dzięki temu technologia UWB nie obciąża znacząco systemów zasilania dronów, co pozwala na dłuższe misje i operacje. UWB radzi sobie dobrze w środowiskach o dużej liczbie przeszkód, takich jak ściany czy inne konstrukcje. Dzięki temu jest idealne do operacji wewnątrz budynków i hal, lub w zatłoczonych przestrzeniach, gdzie drony muszą poruszać się z dużą precyzją. Dzięki dokładnym pomiarom odległości, drony mogą szybko reagować na zmiany w położeniu innych jednostek w roju, co minimalizuje ryzyko kolizji. Jest to szczególnie ważne w dynamicznych środowiskach, gdzie drony muszą często zmieniać kurs lub szybko dostosowywać swoją pozycję.

Wymogiem stawianym każdemu obiektowi, aby mógł uczestniczyć w roju jest posiadanie modułu UWB. Umożliwia to wzajemną komunikację. Jednostka aby mogła zostać liderem musi posiadać moduł do komunikacji z matką operatorem. W przypadku miejsc o dobrym zasięgu komórkowym, dla wybranego rodzaju sieci komórkowej, można wyposażyć jednostkę w modem komórkowy. Atutem tego rozwiązania jest zasięg sieci komórkowej i możliwość akwizycji oraz monitorowania danych. **Jeśli miejsce wykonywania operacji przez drony jest niewystarczająco dobrze pokryte zasięgiem sieci komórkowej lub z góry zdefiniowane są warunki nie wymagające dużych dystansów i pracy w jednym miejscu, najlepszym rozwiązaniem jest użycie komunikacji radiowej o dobranej do warunków długości fali.**

3.1.2. Fizyczne oprzyrządowanie do nawigacji

Wymaganymi elementami niezbędnymi do stabilnego funkcjonowania dronów w zakresie działania poszczególnych jednostek, ale i też całego roju, są komponenty nawigacyjne. Należy przez nie rozumieć oprzyrządowanie, które pozwala sterownikowi lotu odczytywać pozycje

drona. Standardem dla większości jednostek latających, niezależnie czy załogowych, czy bezzałogowych jest obecność barometru, żyroskopu, akcelerometru oraz magnetometru na pokładzie płytki elektronicznej sterownika lotu [204]. W celu zapewnienia bezpieczeństwa przez redundancje danych, stosuje się powielanie niektórych czujników. Pozwala to zabezpieczyć jednostkę przed uszkodzeniem jednego czujnika i wynikającego z tego paraliżu układu sterowania. Daje to również możliwość walidacji danych sterownikowi lotu i wykrycie dryfu czujników. Kolejnym czynnikiem poprawy pracy układu nawigacyjnego jest dodanie podkładki amortyzującej pod czujniki, aby ograniczyć wpływ wibracji na dane.

Należy mieć na uwadze, że pomiar z większości czujników nie jest pomiarem bezpośrednim. Akcelerometr jest używany do pomiaru sił przyspieszenia, które działają na drona. Na podstawie tych danych można określić kąt nachylenia drona (*pitch i roll*), co jest kluczowe dla jego stabilności i orientacji.

W połączeniu z innymi czujnikami, akcelerometr pomaga określić położenie drona w przestrzeni, na przykład poprzez detekcję przechyłów, które mogą sugerować zmianę kierunku ruchu. Przyspieszenie mierzone przez akcelerometr jest dwukrotnie całkowane w czasie, aby uzyskać położenie. Jednakże, ze względu na błędy i dryft, akcelerometr nie jest używany samodzielnie do wyznaczania dokładnej pozycji drona, lecz jest zintegrowany z danymi z innych czujników, takich jak np. GPS i żyroskop. Magnetometr mierzy pole magnetyczne Ziemi w trzech osiach (X, Y, Z). Jest to podobne do działania kompasu, który określa kierunek północny. Magnetometr jest używany do określania orientacji drona względem północy magnetycznej, czyli do wyznaczania azymutu (*heading*). Pomaga to dronowi w utrzymaniu kursu i orientacji w przestrzeni. Dane z magnetometru są przekształcane w informacje o kierunku (azymut), które są następnie używane do korekcji orientacji drona w osi Z.

Barometr jest używany do pomiaru wysokości drona nad poziomem morza. Mierzy ciśnienie atmosferyczne, a zmiany w ciśnieniu są bezpośrednio związane z wysokością nad poziomem morza. Na podstawie wzoru barometrycznego, pomiary ciśnienia są przeliczane na wysokość nad poziomem morza.

$$P(h) = P_0 \cdot exp\left(\frac{-Mgh}{RT}\right) \tag{3.1}$$

gdzie:
- $P(h)$ – ciśnienie atmosferyczne na wysokości h,
- P_0 – ciśnienie atmosferyczne na poziomie morza,
- M – molowa masa powietrza (dla suchego powietrza $M \approx 0{,}029$ kg/mol),
- g – przyspieszenie ziemskie ($g \approx 9{,}81$ m/s2),
- R – uniwersalna stała gazowa ($R \approx 8{,}314$ J/molK),
- T – temperatura powietrza w kelwinach,
- h – wysokość nad poziomem morza.

Barometr zapewnia względną dokładność wysokościową i jest często używany w połączeniu z GPS i akcelerometrem, aby poprawić precyzję wyznaczania wysokości drona. Żyroskop mierzy prędkość kątową, czyli tempo zmiany orientacji drona wokół jego osi X, Y, Z (*roll, pitch, yaw*). Jest ona wyrażana w stopniach na sekundę (°/s) lub radianach na sekundę (rad/s). Żyroskop jest kluczowym czujnikiem dla stabilizacji drona, dostarczając danych o jego ruchach obrotowych. Pomaga w utrzymaniu stałej orientacji i reaguje na zmiany kątowe w czasie rzeczywistym, co przekłada się na stabilność lotu.

Dane z żyroskopu są całkowane w czasie, aby określić zmianę kąta orientacji drona. Podobnie jak akcelerometr, żyroskop jest częścią algorytmu fuzji sensorycznej, który łączy dane z różnych czujników w celu dokładnego śledzenia położenia i orientacji drona w przestrzeni.

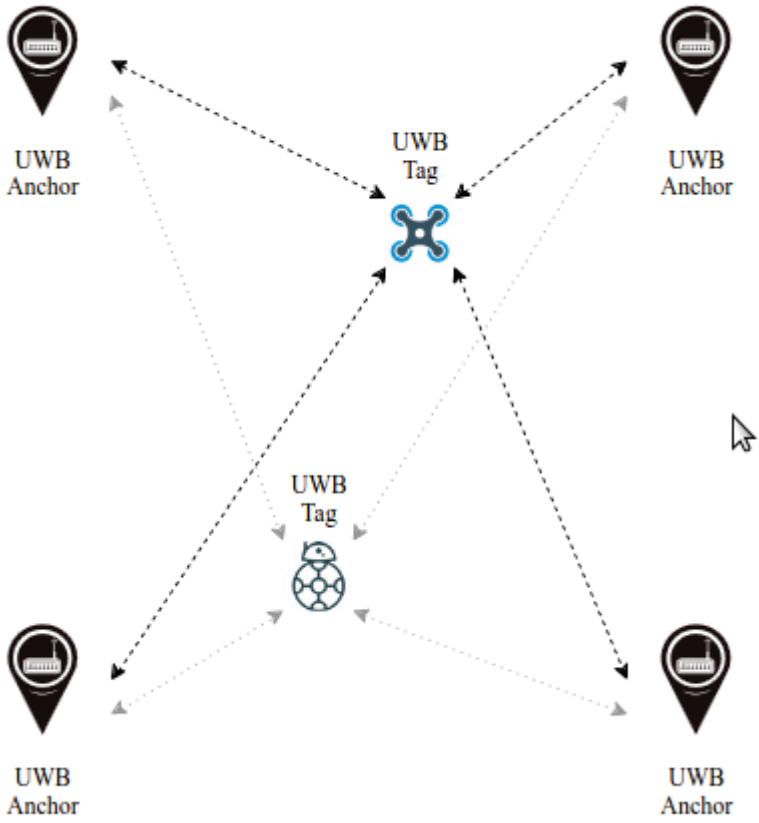
3.1.3. Integracja danych czujników

Dane z powyższych czujników są łączone w czasie rzeczywistym za pomocą algorytmów, takich jak filtr Kalmana lub filtry komplementarne. Te algorytmy uwzględniają błędy i niepewności pomiarów, aby dostarczyć precyzyjne informacje o położeniu i orientacji drona. Porównanie jakości filtrów oraz ich kalibracja stały się tematem osobnej pracy [169]. Rezultaty badań wskazały na zastosowanie rozszerzonego filtru Kalmana jako najlepszego rozwiązania do fuzji danych w powyższych sensorów.

Czujniki takie jak akcelerometr, żyroskop, magnetometr i barometr doskonale nadają się do stabilizacji i kontroli orientacji drona, ale mają trudności z precyzyjnym ustaleniem pozycji w płaszczyznach X i Y. Ze względu na dryft i błędy w pomiarach, same czujniki nie wystarczają do dokładnego określenia pozycji. Dlatego drony często korzystają z GPS, który dostarcza precyzyjnych współrzędnych geograficznych, lub z optycznych czujników przepływu, które analizują ruch względem powierzchni podłoża, aby dokładnie ustalić pozycję i prędkość w płaszczyznach X i Y.

Optyczne czujniki przepływu są używane do ustalania pozycji i prędkości drona względem powierzchni, ale mają kilka istotnych wad w kontekście pracy w roju. Czujniki te polegają na analizie obrazu powierzchni pod dronem. W warunkach słabego oświetlenia, takich jak noc, mogą mieć trudności z rozpoznawaniem wzorców, co prowadzi do błędów w pomiarach. Jeśli dron będzie latać nad powierzchnią, która jest jednolita i pozbawiona wzorców np. pustynia, woda, śnieg, czujnik będzie mieć trudności z wykrywaniem ruchu, co utrudnia precyzyjne określenie pozycji. Sensory optyczne działają najlepiej na małych wysokościach, ponieważ zasięg ich działania jest ograniczony. Na większych wysokościach mogą tracić zdolność do precyzyjnego śledzenia ruchu. Drony mogą generować drgania podczas lotu, co może zakłócać pracę czujników optycznych i prowadzić do błędnych odczytów. Optyczne czujniki przepływu, jeśli nie mają własnego procesora, wymagają intensywnego przetwarzania obrazu, co obciąża sterownik lotu drona i może wpływać na jego wydajność, szczególnie w przypadku mniejszych dronów z ograniczoną mocą obliczeniową.

Powyższe powody sprawiają, że należy użyć innych rozwiązań. Rozpatrując aspekt nawigacji, należy mieć na uwadze kontekst pracy poszczególnej jednostki w grupie roju. Opracowanie naukowe z 2020 roku [177] przedstawia koncepcje, która została zaadoptowana do wzajemnej nawigacji w rozważanym roju dronów. Podjęta oryginalnie przez autorów koncepcja opierała się na użyciu obiektów *Anchor*, czyli „kotwica”. Jako kotwicę w nawigacji UWB definiuje się obiekty o znanej pozycji, względem których obliczany jest dystans. W konsekwencji możliwa do uzyskania jest pozycja obiektu od układu kotwic przedstawiona na rys. 3.1.

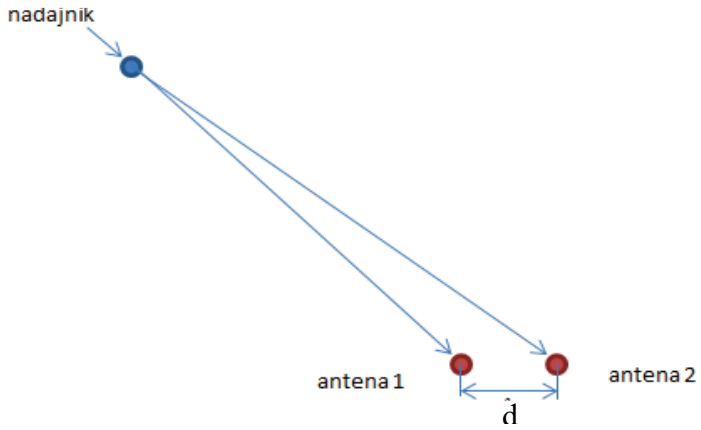


Rys. 3.1. Kotwice w nawigacji UWB [177]

Rysunek 3.1 przedstawia pozycje kotwic w nawigacji UWB. Badacze w swojej pracy skupili się na lokalizacji UAV w płaszczyznach X i Y przy użyciu czterech stałych kotwic. Jest to świetne rozwiązanie do bardzo precyzyjnej nawigacji pojedynczej jednostki. Wyzwaniem postawionym do realizacji podczas tej pracy było opracowanie nawigacji wewnątrz danej grupy nazywanej podrojem tak, aby wszystkie jednostki w grupie mogły określić swoją pozycje w lokalnym dla grupy układzie współrzędnych. Umożliwia to modułowe podejście do tematu sterowania w roju. Dzięki niemu matka-operator odpowiada za nawigacje całej grupy podroju, co przekłada się na zadawanie i kontrolowanie wykonania misji. Natomiast lider

wraz z poszczególnymi jednostkami odpowiadają za nawigacje wewnątrz grupy.

Problemem sterowania w roju jest niemożliwość korzystania ze stałych kotwic. W związku z tym należy rozważyć zagadnienie wykorzystania niektórych z jednostek jako kotwic ruchomych. W przypadku przedstawionej koncepcji [177], należy mieć na uwadze wyposażenie dronów i jako kotwice wykorzystać drony wyposażone w czujniki pozycji liniowej, jak np. odbiornik GPS. Oczywiście zasadnym rozwiązaniem jest wyposażenie każdego drona w odbiornik GPS. Należy jednak rozważyć rozwiązania, gdy odbiorniki GPS w przypadku dość bliskiego i ciasnego rozłożenia obiektów w roju nie są wystarczające bez zastosowania mechanizmu wzajemnego obliczania odległości. Mechanizm działania odbiorników GPS opiera się na nawiązaniu połączenia z minimum 3 satelitami [106]. Następnie na podstawie triangulacji kąta obliczana jest pozycja odbiornika GPS, czego schematyczny widok przedstawiono poniżej.



Rys. 3.2. Poglądowy rysunek triangulacji kąta na płaszczyźnie dwuwymiarowej [źródło własne]

Rysunek 3.2 obrazuje minimalną ilość anten wymaganych w płaszczyźnie dwuwymiarowej. Pierwszym czynnikiem warunkującym jakość nawigacji wzajemnej w roju jest dokładność wynikająca z dużych

dystansów. Jak wynika z opracowania naukowego z 2016 roku [121] błąd GPS w przypadku standardowych odbiorników może wynosić 3-10 metrów. W przypadku bardziej zaawansowanych technologicznie rozwiązań jakimi jest różnicowy GPS - DGPS (*Differential GPS*), możliwe jest uzyskanie dokładności pozycjonowania rzędu kilku centymetrów. To wystarczająca dokładność dla roju dronów. Jednak należy mieć na uwadze fakt, że otoczenie może przesłaniać sygnał GPS. W przypadku utraty nawigacji zewnętrznej, której źródłem byłby GPS, wszystkie drony pozostałyby bez informacji o wzajemnej pozycji, co doprowadziłoby do licznych kolizji w roju. Mogą również pojawić się sytuacje, w których do obliczenia triangulacji kąta tylko niektóre z odbiorników przełączą się między satelitami. W takiej sytuacji lecące obok siebie dwa drony będą używały różnych satelit. Pierwszy z nich użyje satelit startowych, natomiast odbiornik GPS drugiego drona przełączy się z jednej satelity na inną. Oprogramowania odbiorników GPS w przypadkach zastosowań cywilnych nie posiada wbudowanego systemu synchronizacji przełączania satelit między wieloma odbiornikami.

W przypadku roju, który musi wykonać misje, w których część satelit mogłaby być przesłonięta np. przez blok, obok którego drony miałyby przelecieć, rozwiązanie oparte tylko na odbiornikach GPS jest zbyt ryzykowne. Aby uzyskać pożądany efekt sterowania podroju, należy zastosować fuzje nawigacji GPS oraz wykorzystania UWB według algorytmów takich jak np. AoA (*Angle of Arrival*) i TDOA (*Time Difference of Arrival*), tak jak zrobili to m.in. naukowcy NASA [146]. Algorytm TDOA pozwala na wyznaczenie odległości od obiektu nadającego do obiektu odbierającego sygnał.

Istnieje korelacja pomiędzy ilością wymiarów, a ich ilością potrzebną do zebrania sygnałów. W przypadku obiektów latających można jednak rozpatrzyć układ odmiennie dzięki temu, że każdy obiekt latający zawiera barometr i akcelerator na pokładzie swojego sterownika lotu. Każdy obiekt samodzielnie może wyznaczyć swoją wysokość. Do tego celu oprogramowania sterowników lotu wykorzystują filtr AHRS (*Attitude and Heading Reference System*) oparty na rozszerzonym filtrze Kalmana wraz z odczytami z barometru. Pozwala to uprościć nawigacje do płaszczyzny dwuwymiarowej X i Y.

Tak jak zostało rozważone powyżej, **wymagane jest, aby minimum dwie jednostki w grupie posiadały odbiornik GPS**. Daje to możliwość użycia ich jako ruchomych kotwic. Nie jest wymagane, aby to lider był jednostką posiadającą odbiornik GPS. Pozostałe wymagania sprzętowe, czyli obecność modułu UWB zostały zaznaczone w podrozdziale 3.1.1. Z punktu widzenia nawigacji należy dodać, że **moduły UWB muszą mieć zaimplementowane sprzętowo możliwości pracy w algorytmach TDOA i ToA** – *Time of Arrival*. Szczegółowa analiza algorytmów znajduje się w rozdziale 5 dotyczącym komunikacji. Gdy w grupie podroju nie ma wystarczającej ilości odbiorników GPS, czyli minimum 4 odbiorniki, aby podzielić podrój na 2, grupa nie może zostać podzielona. Stawia to problem przed matką-operatorem i **w przypadku zadań obarczonych ryzykiem wystąpienia niepożądanych zdarzeń zalecanym jest, aby każda jednostka wyposażona była w odbiornik GPS. Wymagane jest, aby na pokładzie drona znalazły się wszystkie z rozpatrywanych czujników: akcelerometr, żyroskop, barometr, magnetometr.**

3.1.4. Dodatkowe wyposażenie

Oprócz wyróżnionych czujników i modułów komunikacji drony mogą zawierać dodatkowy osprzęt potrzebny do realizacji danej misji. W założeniu całym rojem steruje operator-matka. Omijanie przeszkód jest zagadnieniem złożonym, wymagającym analizy ukierunkowanej na typy przeszkód adekwatne do zadań wykonywanych przez drony. W przypadku roju dronów nie w każdym rozpatrywanym, wymienionym w podrozdziale 1.3, przypadku istnieje możliwość wystąpienia przeszkód. Mając to na uwadze wyposażenie potrzebne do omijania przeszkód nie znalazło się na liście niezbędnego wyposażenia. W przypadku braku informacji o ewentualnych przeszkodach oraz w razie potrzeby zmapowania otoczenia przez rój, osprzęt do wykrywania przeszkód jest najważniejszym z zestawu dodatkowego wyposażenia. Temat wykrywania przeszkód jest często rozważany przez naukowców w pracach badawczych [192], [122], [24], [187], [42], [38]. Autorzy prac wskazali cztery najczęściej używane czujniki do wykrywania przeszkód. Część z nich poza zadaniami detekcji przeszkód posiada także spektrum dodatkowych możliwości.

1. Kamera – drony mogą być wyposażone w różne typy kamer, w tym kamery HD, 4K, a także kamery termograficzne i kamery z zoomem optycznym. Kamery są używane do nagrywania wideo, robienia zdjęć, inspekcji, monitorowania oraz do zastosowań w systemach FPV (*First-Person View*). Kamery termograficzne są stosowane do monitorowania temperatury i inspekcji obiektów w ciemności lub w trudnych warunkach.

2. Lidar (*Light Detection and Ranging*) – czujnik „detekcji fal świetlnych i zakresu" (odbicia fal). Lidar wykorzystuje pulsacyjne światło laserowe do mapowania powierzchni i pomiaru odległości. Używany jest głównie do tworzenia precyzyjnych map 3D, modelowania terenu i inspekcji.

3. Czujnik ultradźwiękowy – mierzy odległość do przeszkód za pomocą fal dźwiękowych o wysokiej częstotliwości, które pomagają w unikaniu przeszkód i stabilizacji lotu na małych wysokościach. Są używane głównie do pomiarów odległości w bliskim zasięgu.

4. Radar – wykorzystuje fale radiowe do detekcji obiektów, mierzenia ich odległości oraz prędkości. Używany w bardziej zaawansowanych dronach do monitorowania dużych obszarów, wykrywania przeszkód i nawigacji w trudnych warunkach atmosferycznych.

Poza czujnikami do wykrywania przeszkód na pokładzie drona może znaleźć się każdy czujnik potrzebny do wykonania misji. Naukowcy w pracach [211], [22], [13], [3] wskazują inne niewymienione wcześniej czujniki:

- czujnik jakości powietrza – mierzy różne zanieczyszczenia powietrza, takie jak dwutlenek węgla, tlenki azotu czy cząstki stałe. Jest pomocny w monitorowaniu środowiska, na przykład do badania zanieczyszczenia powietrza w miastach, czy w obszarach przemysłowych.
- czujnik temperatury i wilgotności – jest przydatny w monitorowaniu warunków atmosferycznych oraz w badaniach środowiskowych, takich jak monitorowanie zmian klimatycznych, czy w rolnictwie precyzyjnym.

Prym w popularności zarówno w zakresie osprzętu do wykrywania przeszkód, jak i w zastosowaniu w wielu rodzajach misji, wiodą kamery. Mogą one posłużyć w wykrywaniu przeszkód przy użyciu algorytmów wizyjnych [5].

Kolejną ważna cechą jest możliwość wykorzystaniu obrazu z kamer „mobilnych", czyli umieszczonych na pokładzie bezzałogowych obiektów latających, do synchronizacji i uzupełnienia obrazu z kamer statycznych [131]. Autorzy badań z 2023 [215] na zakończenie wniosków swojej pracy wskazali kierunek rozwoju tej technologii:

„*Due to the interdisciplinary nature of the research, extensive effort will be made to realize the synergistic advancement of the integral components and then synthesize them seamlessly with emerging technologies, such as the Internet of Things (IoT), cloud computing, collaborative UAVs, etc., for technology evolution*".

Co można przetłumaczyć:

„Ze względu na interdyscyplinarny charakter badań, podjęte zostaną szeroko zakrojone wysiłki, aby zrealizować synergiczny postęp zintegrowanych komponentów, a następnie płynnie zsyntetyzować je z nowymi technologiami, takimi jak Internet Rzeczy (IoT), przetwarzanie w chmurze, współpracujące UAV itp. w celu ewolucji technologii".

Warstwa sprzętowa komunikacji jest przystosowana do tego, aby przesyłać obraz oraz umożliwić mapowanie terenu przy użyciu czujnika LIDAR, który jest wskazywany przez autorów prac jako najlepszy do tego typu zadań [23], [214], [33], może zostać zrealizowane według założeń Internetu Rzeczy przy dobranej metodzie komunikacji. Do zadań roju należy zebranie informacji w czasie rzeczywistym, lider ma zebrać i przekazać dane, a matka-operator wykonać algorytm mapowania. Do dodatkowego osprzętu mogą też należeć elementy wykonawcze, takie jak np. lampy błyskowe, czy brzęczyki do odstraszania zwierząt.

Dodatkowy osprzęt dla wszystkich jednostek w roju powinien zostać dobrany na potrzeby wykonywanej misji. Najważniejszym jest ustalenie, czy drony mają pracować na znanym terenie, którego mapa może zostać odczytana przez matkę-operatora tak, aby sterowała rojem bez ryzyka kolizji z obiektami zewnętrznymi. **W przypadku potrzeby dynamicznego mapowania terenu należy wyposażyć część jednostek w czujniki typu: kamery, LIDAR, czujniki ultradźwiękowe, radary.** Nie wszystkie bezzałogowe obiekty latające muszą zostać wyposażone w te czujniki. Dzięki komunikacji UWB między jednostkami, lepiej doposażone obiekty mogą

przekazywać liderowi informacje o wykrytych przeszkodach. Oprócz tego w zależności od charakteru misji należy adekwatnie wyposażyć drony, aby mogły wykonać postawione im zadanie.

3.1.5. Sterownik lotu

Zgodnie z przyjętą koncepcją elastyczności rozwiązania i możliwości wymiany komunikacji w przyszłych badaniach na inną przyjęto, że sterownik lotu musi być dostępny na licencji otwartego oprogramowania lub przynajmniej mieć możliwość ingerencji w komunikacje przez API – *Application Programming Interface* (interfejs programowania aplikacji). Należy przez to rozumieć, że wymagana jest możliwość dopisania własnych komend, umożliwiających realizacje potrzeb oprogramowania, zachowań roju i jego komunikacji.

Kolejnym wymaganiem odnośnie sterownika lotu wynikającym z założenia, aby praca mogła posłużyć w sektorach, gdzie dostęp cenowy do technologii odgrywa znaczącą rolę jest zastosowanie sterownika lotu, który umożliwiałby fizyczne dodanie i programowe obsłużenie czujników dostępnych na rynku, nie ograniczając zakresu do czujników dedykowanych do jednostek powietrznych. Celem tego jest umożliwienie wykorzystania czujników np. dedykowanych do Internetu rzeczy, takich jak detektor ognia etc. Istotnym, w aspekcie wszechstronności, jest również możliwość dodania elementów wykonawczych takich jak np. brzęczek odganiający zwierzęta, który byłby umieszczony na pokładzie drona.

Początek prac skierowany był na przygotowanie autorskiego rozwiązania w postaci taniego sterownika lotu. Został zaprojektowany sterownik lotu oparty na minikomputerze Raspberry Pi Zero, który posłużył do pierwszych testów. Napisano własne oprogramowanie dla sterownika lotu. W czasie prac uwagę skierowano na pracę naukową z 2009 r. [31], gdzie badacze wykorzystali popularne oprogramowanie ArduPilot do przygotowania własnej jednostki latającej. Kierunek badań w niniejszej rozprawie został skierowany, aby możliwe było wykorzystanie zarówno własnego sterownika lotu, jak i gotowych rozwiązań. Praca badaczy z Danii [49] porównała ponad 20 rozwiązań technologicznych dostępnych na licencji

otwartej. Wśród oprogramowania sterowników lotu rozpatrywane były modele oprogramowania:

- Hack flight,
- Cleanflight,
- Betaflight,
- LibrePilot,
- Ronin,
- ArduPilot,
- PX4,
- Paparazzi.

Wybór oprogramowania jest bardzo duży. Nie wszystkie z wymienionych oprogramowań pozwalają na implementacje własnych komunikacji wymaganych do komunikowania wzajemnego wewnątrz roju, a także niektóre z wymienionych mogą nie zapewniać adekwatnej jakości w sterowaniu autonomicznym. Szczególnie w kontekście możliwości użycia innych typów UAV niż quadrocopter. W 2023 roku badacze [185] wskazali kierunki rozwoju otwartożródłowych oprogramowań sterowników lotu. Ich wnioski pokryły się z badaniami prowadzonymi w niniejszej pracy. Dwa modele oprogramowań ArduPilot i PX4 nie tylko pozwalają na implementacje potrzebnych dodatków na potrzeby sterowania w roju, ale także są najlepiej zarządzane przez społeczność przygotowującą oprogramowanie. Oba oferują zaawansowane możliwości konfiguracji i są używane w szerokim zakresie zastosowań, od hobbystycznych projektów po profesjonalne misje wojskowe i przemysłowe.

ArduPilot to otwartoźródłowy sterownik lotu, który jest jednym z najbardziej dojrzałych i wszechstronnych rozwiązań na rynku. Obsługuje szeroką gamę platform, w tym drony, samoloty, łodzie i pojazdy lądowe. ArduPilot jest niezwykle modułowy, co pozwala na łatwą integrację różnych czujników i modułów komunikacyjnych. Można go skonfigurować do współpracy z różnymi systemami GPS, LIDAR, kamerami oraz systemami komunikacji, co jest istotne dla roju dronów, gdzie różne jednostki mogą korzystać z różnych technologii. ArduPilot oferuje zaawansowane algorytmy autonomii, w tym zaawansowane planowanie trasy. ArduPilot może być

skonfigurowany do pracy z różnymi technologiami komunikacyjnymi. Oprogramowanie ma dużą i aktywną społeczność, co przekłada się na szeroką dokumentację, wsparcie i dostępność różnych dodatkowych modułów oraz rozszerzeń.

PX4 to również otwartoźródłowy sterownik lotu, który jest znany ze swojej elastyczności i wysokiej wydajności. Jest to obecnie najczęściej używany sterownik na otwartej licencji w profesjonalnych i badawczych projektach. Podobnie jak ArduPilot, PX4 obsługuje wiele platform, w tym drony, samoloty i pojazdy lądowe. PX4 oferuje wsparcie dla szerokiej gamy czujników, co umożliwia łatwe dopasowanie systemu do specyficznych potrzeb projektu. Obejmuje to wsparcie dla różnych typów kamer, radarów, lidarów oraz innych zaawansowanych czujników. PX4, podobnie jak ArduPilot jest elastyczny pod względem integracji z różnymi technologiami komunikacyjnymi. Umożliwia korzystanie z takich technologii jak: UWB, LTE, Wi-Fi i innych. PX4 działa na systemie operacyjnym czasu rzeczywistego lub w przypadku minikomputerów na systemie Linux, co zapewnia wysoką wydajność i deterministyczne zachowanie systemu. Jest to szczególnie istotne w krytycznych zastosowaniach, takich jak loty w roju. PX4 jest rozwijany przez dużą społeczność oraz wspierany przez Dronecode Foundation, co oznacza stałe aktualizacje, poprawki oraz rozwój nowych funkcji.

ArduPilot może być lepszym wyborem dla projektów, które wymagają szerokiej dokumentacji, wsparcia społeczności i elastyczności w zakresie obsługi różnych typów pojazdów. PX4 jest często preferowany w profesjonalnych i badawczych projektach, gdzie kluczowa jest wysoka wydajność, wsparcie dla systemów czasu rzeczywistego oraz elastyczność w integracji z zaawansowanymi czujnikami i systemami.

Za wyborem oprogramowania ArduPilot przemawia dojrzałość projektu, który od lat przoduje jako wybór wśród użytkowników amatorskich i przez nich jest rozwijany bardzo stabilnie. Aktualizacje oprogramowania nie wnoszą zasadniczych nowości, ale również są bardziej stabilne i lepiej przetestowane przez szerokie grono użytkowników niż aktualizacje dla PX4. Projekt PX4 został zapoczątkowany na Politechnice Federalnej w Zurychu przez Lorenz Meier, który jest szwajcarskim inżynierem i badaczem. Lorenz

Meier jest również założycielem Dronecode Foundation, organizacji wspierającej rozwój otwartoźródłowego oprogramowania dla dronów, w tym PX4.

ArduPilot i PX4 wspierają szeroki zakres fizycznych kontrolerów lotu. Oba systemy są bardzo elastyczne, jeśli chodzi o kompatybilność z różnymi modelami kontrolerów lotu. Jednym z najbardziej znanych sterowników zgodnych z ArduPilotem jest rodzina kontrolerów Pixhawk, na których przeprowadzano początkowe testy i porównania obu oprogramowań.

Zdj. 3.1. Kontroler Pixhawk w wersji FMUv2 [źródło własne]

Sterowniki lotu Pixhawk, przedstawione na rysunku powyżej, są szeroko stosowane zarówno w projektach hobbystycznych, jak i komercyjnych. Oferują zaawansowane możliwości i wsparcie dla różnych typów pojazdów, takich jak drony, samoloty, łodzie i pojazdy lądowe. Mogą występować w dużych jednostkach z uniwersalnymi złączami, tak jak na powyższym rysunku, ale również w postaci bardzo małych jednostek zoptymalizowanych pod kątem wielkości i masy.

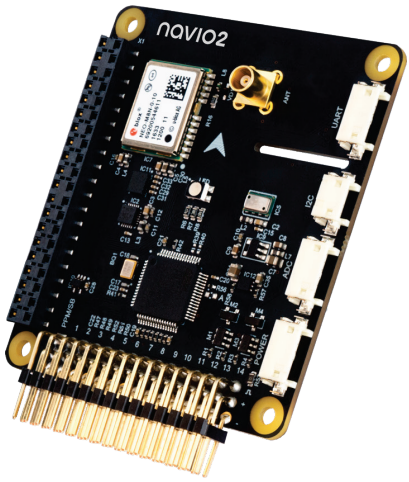
ArduPilot wspiera również sterowniki od wielu innych producentów, takich jak Matek, Holybro i Radiolink. Każdy z tych sterowników oferuje różne poziomy zaawansowania i funkcjonalności – od prostych, tanich rozwiązań, po zaawansowane systemy do zastosowań profesjonalnych.

Obecnie wszystkie kontrolery lotu Pixhawk są utrzymywane przez Dronecode Foundation i posiadają pełne wsparcie dla projektu PX4. Oba rodzaje oprogramowania pozwalają na dodawanie własnych fizycznych sterowników lotu o architekturze procesorów zgodnej z systemami czasu rzeczywistego, wykorzystywanych przez poszczególne oprogramowania.

Zarówno PX4 jak i ArduPilot mogą być wykorzystywane opracowanej koncepcji w kontekście integracji z modułem środowiska graficznego. Posiadają dobrą dokumentacje i wsparcie dla wielu popularnych środowisk. Z porównania wynika, że oba rodzaje oprogramowania są bardzo zbliżone. Są też na tyle powszechne, że używają ich naukowcy NASA [16] dla swoich badań.

Jako sterownik lotu należy uwzględnić przede wszystkim oprogramowanie tego sterownika. W zakresie sprzętowym sterownik musi zawierać wymienione w podrozdziałach 3.1.1 i 3.1.2 komponenty na pokładzie lub uzupełnić je przez podpięcie do sterownika. **Rekomendowane są dwa rodzaje oprogramowania – PX4 i ArduPilot**. Są na tyle zbieżne, że ciężko wskazać lepszą opcję, a jednocześnie oba są podobnie kompatybilne, więc można je wymieniać. W dalszej części prac posługiwano się oprogramowaniem ArduPilot ze względu na bardziej przystępną dokumentację i mniej dynamicznych zmian w oprogramowaniu. Jednak przyszłościowo PX4 może okazać się lepszym wyborem, gdy wszystkie dodatkowe funkcjonalności zostaną w pełni wdrożone i przetestowane.

Jednocześnie należy dodać, że można użyć innych oprogramowań, które tylko pozwolą na obsługę komponentów wymienionych w podrozdziałach 3.1.1 i 3.1.2, a także zagwarantują stabilny lot. **Rekomendowanym typem sterownika lotu jest Pixhawk**. Możliwym jest również przygotowanie własnego sterownika, zarówno z układem mikrokontrolerowym kontrolowanym przez system czasu rzeczywistego, jak i układem minikomputerowym kontrolowanym przez system operacyjny Linux. Przygotowany na wczesnym etapie prac sterownik oparty na minikomputerze Raspberry Pi Zero nie wniósł różnicy w jakości w stosunku do przetestowanej nakładki sterownika lotu Navio 2 przedstawionej na zdjęciu 3.2.



Zdj. 3.2. Nakładka sterownika lotu Navio 2, która jest rozszerzeniem dla minikomputera Raspberry Pi modele 2, 3, 4 [źródło własne]

Zaprezentowany układ Navio 2 jest gotowym rozwiązaniem przystosowanym do tego, aby użyć minikomputer Raspberry Pi jako sterownik lotu. Posiada dodatkowy procesor wykonawczy, który steruje pinami wejścia i wyjścia, zapewniając odpowiedź na krytyczne dla stabilności lotu sygnały w czasie rzeczywistym. **Wymaganiem koniecznym dla dronów pełniących role lidera jest większą moc obliczeniowa**, niż pozostałych jednostek ze względu na dużo bardziej zaawansowane potrzeby w zakresie komunikacji. Może to zostać zagwarantowane przez wykorzystanie najnowszych sterowników lotu, w tym Pixhawk, o większej mocy obliczeniowej. **Rozwiązanie z minikomputerem jako sterownikiem lotu jest rekomendowane dla jednostek mogących pełnić rolę lidera.** Pozostawia to duży bufor możliwości obliczeniowych, a także możliwości lepszej diagnostyki i zarządzania grupą w przypadku utraty łączności z matką-operatorem i dodanie własnych algorytmów uruchamianych z poziomu systemu Linux.

3.2. Model matematyczny

Zagadnienie modelowania obiektów latających jest popularnym tematem prac naukowych. W niniejszej pracy obrano cel zamodelowania

układu w sposób odpowiadający wymogom fizycznym najbardziej zbliżonym do symulacji w środowisku graficznym. Należy przez to rozumieć potrzebę zamodelowania zjawisk zachodzących w procesie przekształcania sygnałów sterujących ze sterownika lotu w sygnały elektryczne odbierane przez silniki. W przypadku silników bezszczotkowych, które dużo częściej używane są w dronach wykonujących misje na zewnątrz budynków, sygnały te odbierane są za pomocą regulatorów napięcia ESC – *Electronic Speed Controller*, czyli „elektroniczny regulator prędkości". Następnie silnik zamienia sygnał elektryczny na ruch obrotowy, który napędza śmigło generujące ciąg i moment skręcający.

W licznych pracach badawczych [52], [188], [98], [132] ten ciąg następstw i ich konsekwencji jest pomijany lub skracany do jednego wzoru modelującego całą zależność, a nacisk pracy jest kładziony na dobranie najlepszych sygnałów sterujących i nastaw dla regulatorów. **Innowacyjnym w tym podejściu do modelowania jest położenie nacisku na realizacje sygnału sterującego przez sterownik lotu wraz z układem wykonawczym**. Pomimo licznie prowadzonych badań nad modelem najpopularniejszego bezzałogowego obiektu latającego jakim jest quadrocopter, wciąż pojawiają się nowe badania, jak praca z 2024 roku [148]. Nie jest to obszar badań, który można uznać za zamknięty i w pełni rozpoznany. Poniższy model jest przygotowany na potrzeby ewaluacji modułu środowiska graficznego.

Model matematyczny został przygotowany dla przyjętego na początku rozdziału typu jednostki latającej, czyli quadrocoptera o silnikach rozmieszczonych na kwadracie. Aby zamodelować nieliniowość współczynnika ciągu i momentu skręcającego od prędkości obrotowej silnika, wybrano metodę krzywej regresji. Na etapie modelowania przyjęto dane dla quadrocoptera Crazyflie 2.1. Parametry drona i wyprowadzenia modelu zostały zamieszczone w załączniku nr 1.

Równania ruchu przedstawiają się następująco:

$$\ddot{\phi} = \frac{d}{2 \cdot I_{xx}} \cdot [(T_3 + T_4) - (T_1 + T_2)]$$

$$\ddot{\theta} = \frac{d}{2 \cdot I_{yy}} \cdot [(T_2 + T_3) - (T_1 + T_4)]$$

$$\ddot{\psi} = \frac{Q_2 + Q_4 - (Q_1 + Q_3)}{I_{zz}} \tag{3.2}$$

$$\ddot{x} = \frac{cos(\phi) \cdot sin(\theta) \cdot (T_1 + T_2 + T_3 + T_4) \pm D_x}{m}$$

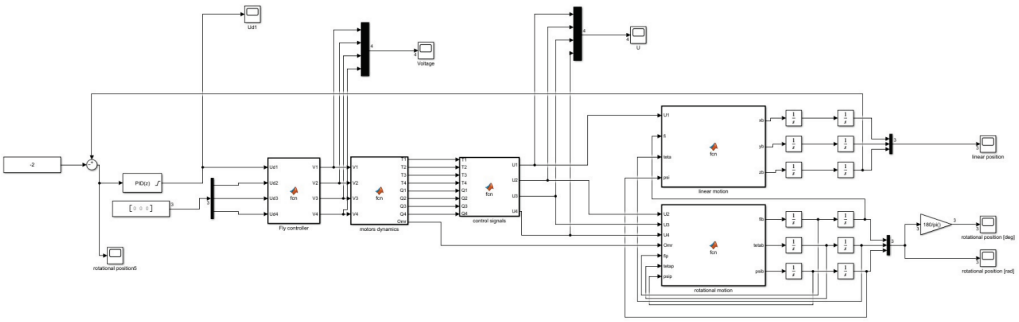
$$\ddot{y} = \frac{sin(\phi) \cdot cos(\theta) \cdot (T_1 + T_2 + T_3 + T_4) \pm D_y}{m}$$

$$\ddot{z} = \frac{cos(\phi) \cdot cos(\theta) \cdot (T_1 + T_2 + T_3 + T_4) \pm D_z}{m} - g$$

Gdzie:

$\ddot{\phi}, \ddot{\theta}, \ddot{\psi}$ – przyśpieszenie kątowe działające wokół danej osi,
I_{xx}, I_{yy}, I_{zz} – momenty bezwładności działające wokół danej osi,
d – szerokość oraz wysokość mierzona od osi silników,
Q - momenty skręcające,
$\ddot{x}$, $\ddot{y}$, $\ddot{z}$ – przyśpieszenie układu w danej osi,
D_x, D_y, D_z – zakłócenia działające w danej osi powodowane siłą wiejącego wiatru,
m – masa drona,
g – przyspieszenie ziemskie,
T – ciąg wygenerowany przez pojedynczy silnik.

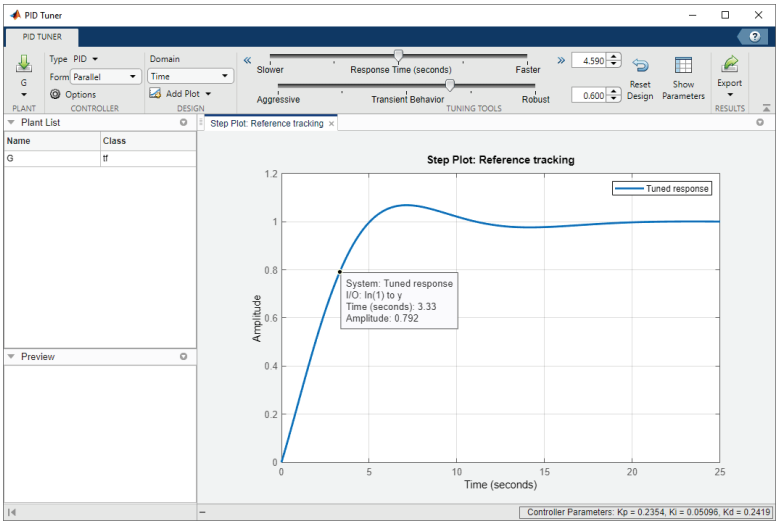
Model został zaimplementowany w programie Matlab/Simulink w wersji 2021b. Widok graficzny w programie Simulink przedstawia rys. 3.3.



Rys. 3.3. Implementacja modelu w programie Simulink [źródło własne]

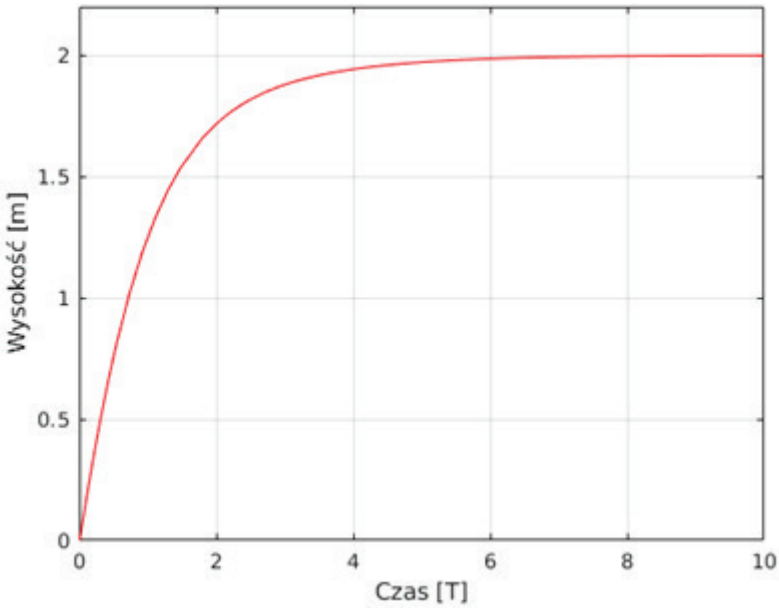
Gdzie bloki w programie odpowiadają składowym elementom drona. Całkowanie przeprowadzono metodą ode45. Jako test modelu wykonano eksperyment polegający na wzlocie drona na 2 metry powyżej ziemi i pozostaniu w zawisie. Drugim testem weryfikującym responsywność nastaw na dynamiczne wymuszenia był test wzlotu na 2 metry, następnie wzlot na wyższą wysokość wynoszącą 5 metrów. Następnie powrót do wysokości 2 metrów.

W celu dobrania nastaw dla regulatora wysokości PID wykorzystano moduł PID Tuner programu Matlab [83]. Wygląd aplikacji graficznej przedstawia rys 3.4



Rys. 3.4. Widok bloku Pid Tuner programu Matlab [źródło własne]

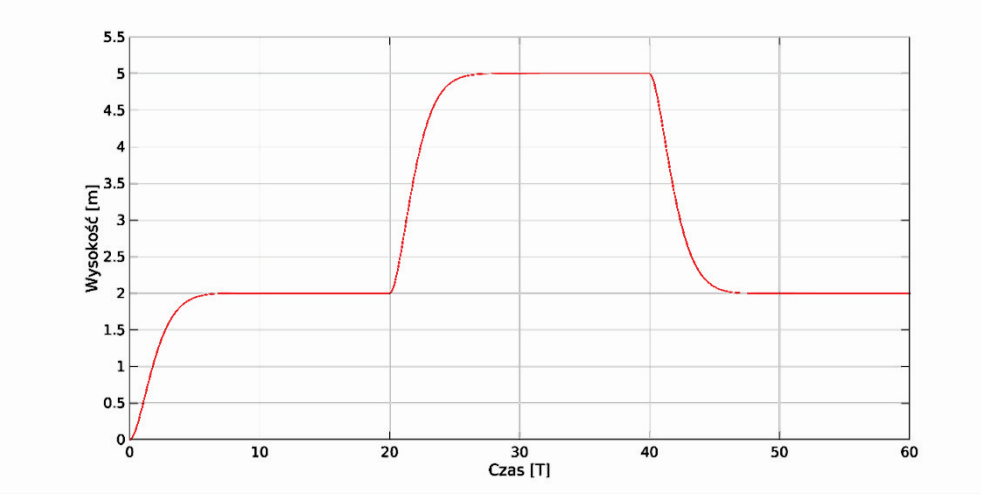
Przedstawiony na rysunku 3.4 blok dostępny jest w pakiecie Control Toolbox i umożliwia zlinearyzowanie modelu w symulacji i dobranie nastaw dla regulatora PID. Dla zamodelowanego drona dobrane zostały nastawy Kp = 104.53, Ki = 5.42, Kd = 187.34. Blok PID Tuner dobiera nastawy precyzyjnie dla danego eksperymentu. Rezultaty symulacji przedstawiają się następująco:



Rys. 3.5. Zmiana wysokości drona w funkcji czasu przy zawisie na wysokość 2 metrów [źródło własne]

Na rysunku 3.5. przedstawiono odpowiedź drona na wymuszenie skokowe i zadanie wzlotu na dwa metry powyżej ziemi. Dron osiąga zadaną wysokość w czasie około 6 sekund.

Drugim wspomnianym scenariuszem testowym był wzlot na 2 metry. Następnie wzlot na wyższą wysokość wynoszącą 5 metrów. Finalnie powrót do wysokości 2 metrów. Aby uzyskać stabilny lot wymagane było ponowne wyznaczenie nastaw regulatora wysokości przy użyciu bloku PID Tuner. Wyniosły one kolejno: Kp = 70.34, Ki = 7.32, Kd = 201.64.



Rys. 3.6. Zmiana wysokości drona w funkcji czasu dla drugiego scenariusza lotu [źródło własne]

Po zmianie nastaw regulatora dron osiągnął zadaną wysokość w czasie około 7 sekund. Na rysunku 3.6 można zauważyć mniej dynamiczny start w przypadku gdy dron startował z pozycji wyjściowej. Różnica wynika z nowych nastaw regulatora PID dobranych pod scenariusz eksperymentu przedstawionego na rysunku 3.6.

3.3. Podsumowanie rozdziału

Celem rozdziału było przedstawienie modułu jednostki latającej z opisem możliwych typów jednostki oraz wymagań sprzętowych umożliwiających implementacje komunikacji na wyższych poziomach. Wykonano przegląd sprzętowych możliwości komunikacji bezprzewodowych. Dokonano rozróżnienia zgodnego z przyjętą koncepcją na komunikację wewnątrz grup roju i komunikację liderów grup z matką-operatorem. **Dla komunikacji z matką-operatorem zaproponowano dwie możliwe komunikacje: komunikacje IoT po sieci komórkowej, a także komunikacje radiową.** Obie komunikacje mogą mieć różne warianty. Dla łączności po sieci komórkowej warianty wyboru zależne są od dostępnej sieci i wymaganej przepustowości przepływu danych. Dla komunikacji radiowej różnorodność wynika z częstotliwości fal. Również należy uwzględnić

wymaganą przepustowość danych, a także dystans między liderami a matką-operatorem. Należy mieć na uwadze, że jest to komunikacja bezpośrednia, co czyni ją dobrym wyborem, gdy z góry znany jest dystans pomiędzy matką-operatorem rojem i dystans ten nie wykracza poza możliwości fal radiowych. Z kolei komunikacja IoT po sieci komórkowej jest dobrym wyborem, gdy miejsce wykonywania zadania dronów jest dobrze pokryte zasięgiem wybranej sieci komórkowej. Dystans do matki-operatora nie ma wtedy znaczenia. **Wymogiem kompatybilności z modułem komunikacji jest dobranie protokołu komunikacyjnego do wymiany danych między liderami a matką-operatorem, który mógłby funkcjonować na obu wybranych warstwach sprzętowych wymiennie.** Jest to zadanie, którego sposób realizacji jest treścią rozdziału 5. Obustronna kompatybilność ma umożliwić szybką wymianę modułu do komunikacji w jednostkach mogących pełnić rolę lidera, aby uczynić rozwiązanie elastycznym zgodnie z przyjętymi dla pracy założeniami.

Komunikacja wewnątrz roju została dobrana tak, aby była szybka, niezawodna, przepustowa dla wielu danych, wysyłanych w małych pakietach. Dodatkowo ma ona posłużyć do pozycjonowania dronów **wewnątrz roju. Wybrana została komunikacja UWB.** Wykorzystuje ona fale elektromagnetyczne do transferu danych. Na poziomie warstwy sprzętowej spełnia postawione wymagania. Wymagane jest, aby każdy bezzałogowy obiekt latający w roju był wyposażony w moduł UWB. Innowacyjnym i autorskim jest wykorzystanie UWB zarówno do wymiany danych między jednostkami, ale również użyciu tej komunikacji do wyznaczania pozycji wzajemnej jednostek w grupach roju. Wymaga to niestandardowego podejścia do zrealizowania komunikacji po stronie sterowników lotu. **Od strony sprzętowej wymaga obsługi algorytmów TDOA i ToA w modułach komunikacyjnych UWB.**

Poza wymaganymi modułami do komunikacji zdefiniowane zostały pozostałe wymagania sprzętowe. **Zalecanym jest, aby każdy obiekt posiadał odbiornik GPS, ale nie jest to warunkiem koniecznym. Wymagane jest, aby każdy obiekt zawierał czujniki akcelerometr, żyroskop, magnetometr, barometr.** Dane z tych czujników są niezbędne do stabilizacji UAV. Są też istotne w kontekście wyznaczania pozycji obiektu.

Dane odczytywane bezpośrednio z tych czujników nie reprezentują wymaganych informacji o pozycji obiektu. Należy użyć specjalnych filtrów takich jak filtr Kalmana.

Uzupełnienie osprzętu stanowią opcjonalne komponenty, które mogą być potrzebne zgodnie ze specyfiką danego zadania. Szczególną uwagę zwrócono na rozwiązania stosowane w bezzałogowych jednostkach latających dedykowanych do wykrywania przeszkód. Dobranie odpowiedniego oprzyrządowania może być kluczowe dla właściwego zmapowania terenu i może wymagać korekty dobranych modułów komunikacyjnych, aby umożliwić odpowiednią przepustowość danych. Jest to powód dlaczego komunikacja liderów z matką-operatorem pozostaje do doboru modułowego.

Wytyczone wymagania odnośnie niezbędnego osprzętu czujników są realizowane przez sterowniki lotu Pixhawk. Są one polecane od strony sprzętowej sterownika lotu. Część z nich jest również udostępniana wraz ze schematami elektronicznymi, dzięki czemu można z nich skorzystać, prototypując własny sterownik. Kolejnym rozwiązaniem jest użycie płytki rozszerzającej, jak np. przytoczony przykład płytki Navio 2. Pozwala ona zamienić minikomputer Raspberry Pi w sterownik lotu działający pod kontrolą systemu operacyjnego Linux. **Wykorzystanie sterownika lotu z systemem Linux jest polecane dla obiektów, które mogą pełnić rolę lidera w grupie. Dla pozostałych jednostek system czasu rzeczywistego wykorzystywany w oprogramowania sterownika lotu jest wystarczającym rozwiązaniem. Wytypowano dwa optymalne otwartoźródłowe oprogramowania sterowników lotu: ArduPilot i PX4.** Praca przewiduje zastosowanie dowolnego z nich. Eksperymenty przeprowadzono na oprogramowaniu ArduPilot, ale PX4 również nadaje się do wykorzystania dronów w roju i nie wymaga zmian na poziomie innych modułów pracy.

Wymogi zostały określone tak, aby zachować maksymalną elastyczność w wymianie typu jednostki, oprogramowania i komponentów, aby móc dobrać je do wykonywanych zadań. Jednocześnie zostały uściślone kluczowe elementy wymagane do poprawnej i niezawodnej pracy takie jak obowiązkowe czujniki, moduły komunikacyjne. W trakcie prac testowano

rozwiązania na różnych komponentach od różnych producentów. Ze względu na otwarte źródłowe oprogramowanie sterowników lotu nie ma konieczności zawężania komponentów tylko do sprecyzowanych modeli. Tak jak w przypadku modemu komunikacji UWB, wszystkie moduły, które mają wbudowaną obsługę sprzętową algorytmów TDOA i ToA oraz posiadają możliwość komunikacji ze sterownikiem lotu, mogą zostać wykorzystane. Pozostaje również duża dowolność w zakresie wyboru fizycznej jednostki sterownika lotu, gdyż zarówno ArduPilot i PX4 obsługują wiele gotowych rozwiązań, a także pozwalają zaprojektować własny sterownik.

Oczekiwanym efektem prac na tym etapie było wypracowanie maksymalnie elastycznych, a jednocześnie precyzyjnych wymagań sprzętowych umożliwiających implementacje roju na wyższych poziomach, co zostało zrealizowane. Wyższe poziomy oznaczają prowadzenie eksperymentów, integracje komunikacji i nawigacji oraz wypracowanie inteligencji roju, co jest kolejno treścią następnych rozdziałów.

W tym rozdziale poruszona została jeszcze kwestia modelowania obiektu. Jak zostało wspomniane w rozdziale 2, koncepcja prac na rojem opiera się o użycie środowiska graficznego do prac nad inteligencją roju. Opracowany został autorski model matematyczny o konwencji zbliżonej do rzeczywistego realizowania sygnałów sterujących przez sterownik lotu. Przeprowadzono dwa eksperymenty symulacyjne, dobrane tak, aby przedstawić problematykę zachowań symulacji numerycznych i ich adekwatność do fizycznego lotu, gdzie sterownik lotu nie wyłącza silników podczas obniżania pułapu, nawet kosztem dłuższej realizacji zadania. Warunkiem nadrzędnym jest bezpieczeństwo, co przekłada się na to, że opadanie obiektu jest bardziej kontrolowane i w przypadku znalezienia się np. nieplanowego podłoża pod obiektem, kontakt jest bardziej kontrolowany i zmniejsza ryzyko zniszczenia jednostki. Optymalny zestaw nastaw dla pierwszego eksperymentu nie sprawdził się w inaczej zaplanowanej próbie. Istnieje potrzeba opracowania środowiska dla prowadzenia badań nad rojem. Zgodnie z założeniami dla pracy poczynionej w rozdziale 2 moduły środowiska graficznego oraz moduł komunikacji weryfikowane są przez eksperymenty na rzeczywistych jednostkach. Moduł inteligencji roju wymaga możliwości wielokrotnego odtwarzania eksperymentu w celu

wypracowania tytułowej inteligencji. Również adekwatnie potrzeba dużej ilości jednostek, aby móc wykorzystać mechanizmu roju, co wymusza prowadzenie badań w przygotowanym środowisku.

Rozdział 4

ŚRODOWISKO GRAFICZNE

W niniejszym rozdziale skupiono się na wypracowaniu optymalnej koncepcji pracy z liczną grupą bezzałogowych obiektów latających. Dokonano przeglądu istniejących rozwiązań. Zaprezentowano opracowaną koncepcję w postaci algorytmicznej listy kroków, jaką należy podjąć, aby skorzystać z proponowanego rozwiązania. Zestawiono również to rozwiązanie z koncepcjami dotychczas przyjętymi do symulacji. Walidacji rozwiązań dokonano na fizycznej jednostce latającej. Jej zarejestrowany lot posłużył jako dane arbitralne.

Opracowanie koncepcji symulacji opartej na silniku graficznym z możliwością implementacji modeli CAD (*Computer Aided Design*) rzeczywistych UAV z możliwością testowania komunikacji i sensoryki, jest odpowiedzią na potrzebę środowiska pracy nad rojem w aspekcie wielu jednostek oraz wdrażania inteligencji roju. Z kolei walidacja z obiektem rzeczywistym jest odpowiedzią na potrzebę sformułowaną przez badaczy o skutecznej walidacji rozwiązań i dążenia do wdrażalności tych rozwiązań [154].

4.1. Programy i pakiety oprogramowania

Symulatory dronów, zwłaszcza te przystosowane do pracy z rojami, są kluczowe w badaniach nad złożonymi systemami bezzałogowych statków powietrznych (UAV). W porównaniu do numerycznych symulacji, oferują one bardziej realistyczne środowisko, które uwzględnia fizyczne właściwości, dynamikę lotu, interakcje z otoczeniem, a także efekty zakłóceń i warunki atmosferyczne. Dzięki wizualizacji 3D i zaawansowanej grafice, symulatory pozwalają na przeprowadzenie testów, które wierniej oddają rzeczywistość niż modele matematyczne. Dodatkowo, symulatory umożliwiają testowanie algorytmów kontroli i koordynacji w czasie rzeczywistym, co jest trudne do osiągnięcia w czystych symulacjach numerycznych. Mnogość argumentów przemawiających za symulatorami

graficznymi, a także możliwości technologiczne przyczyniły się do powstania wielu programów do symulacji graficznej. Początkowo były one dedykowane do robotów mobilnych, uwzględniając głównie pojazdy jezdne. Wraz z rozwojem technologii obiektów latających, zaczęły pojawiać się dedykowane symulatory dla jednostek latających, zarówno w zakresie nauki pilotów, jak i obiektów bezzałogowych. Adoptowano również symulatory robotyczne do możliwości obsługi obiektów latających. Zarówno w pracach badawczych dotyczących symulatorów lotu [134], [155], [104], [62], jak i poszczególnych pracach, w których wykorzystano symulatory graficzne do badań nad dronami [190], [133], [210], [130], [181], [138], [101] zarysowuje się użycie siedmiu programów: Gazebo, AirSim, PX4-SITL, FlightGear, Xplane, OpenUAV, jMAVSim. Porównanie symulatorów bezzałogowych obiektów latających znajduje się w załączniku nr 2.

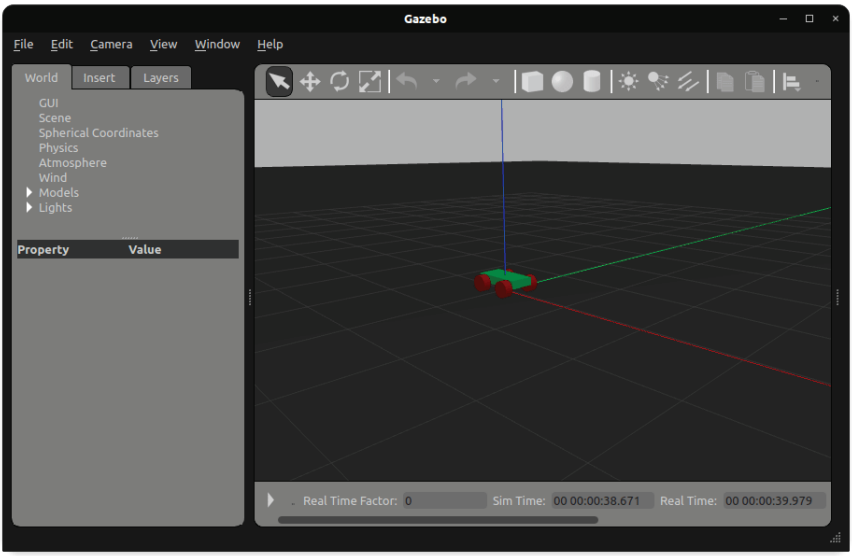
W pracy badawczej na temat luki między symulacjami a fizycznymi lotami w zastosowaniu przemysłowym, badacze w 2023 roku wskazali na wysoki potencjał programu Gazebo [151]. Dzięki swojej powszechności, elastyczności w doborze komunikacji między symulatorem a zewnętrznym oprogramowaniem, a także możliwości wykorzystania tego programu do napisania dodatkowej liczby pakietów oprogramowania, wskazany jest jako najlepszy program do symulacji. Autorzy wskazali także większą ilość nakładek na program Gazebo, przygotowanych do pracy z dronami. Należy dodać, że Gazebo spośród wszystkich programów oferuje najlepsze wsparcie do testowania dronów wraz z ich oryginalnym oprogramowaniem, a zarazem umożliwia testowanie inteligencji roju w warunkach najbardziej zbliżonych do rzeczywistych. Komunikacja z Gazebo możliwa jest przy użyciu ROS oraz protokołu MAVLink, który to protokół został szerzej opisany w kolejnym rozdziale.

4.2. Robot Operating System

ROS (*Robot Operating System*) to otwarte oprogramowanie służące do tworzenia aplikacji robotycznych. Dzięki możliwościom w zakresie badawczym i inżynieryjnym ROS stał się narzędziem w wielu badaniach [180], [186], głównie skierowanych na autonomiczne pojazdy [213], [54], [51], a także został samodzielnym tematem prac naukowych [160], [159], [9].

Jest również główną tematyką monografii naukowych opracowywanych przez Anisa Koubaa [111-117].

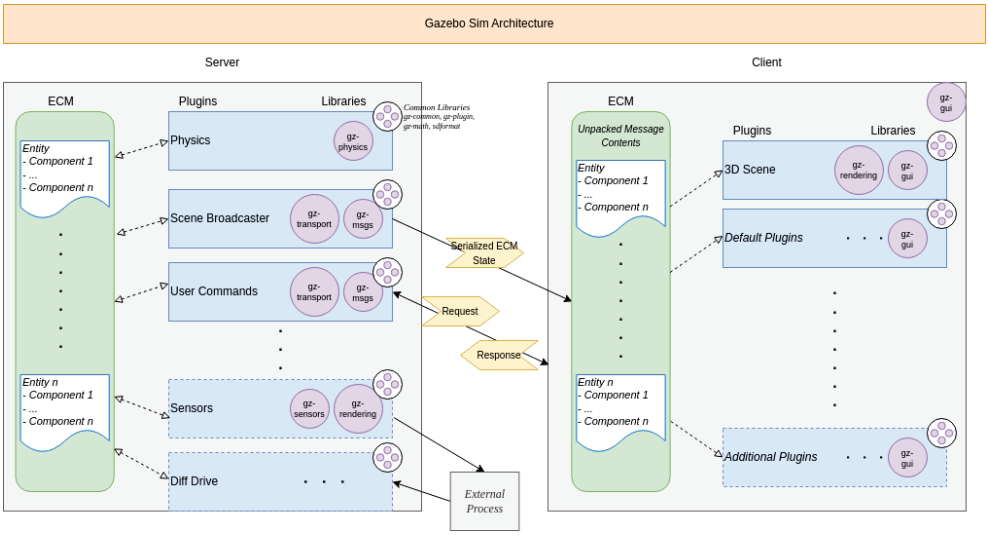
Gazebo jest programem, który powstał przed metasystemem ROS. Od samego początku ROS program Gazebo był rozwijany jako narzędzie, które można zintegrować z ROS, co umożliwia symulowanie robotów korzystających z tej platformy. W 2007 projekt ROS został rozszerzony o wersje ROS2, która jest rozwijana równolegle do ROS1. W 2012 roku projekt ROS i Gazebo zostały połączone pod wspólnym zarządem fundacji Open Source Robotics Foundation (OSRF). Fundacja ta miała na celu rozwijanie otwartego oprogramowania dla robotyki, a integracja ROS z Gazebo stała się kluczową częścią tej strategii. Wraz z rozwojem ROS 2, Gazebo zostało również rozwinięte w nowym kierunku jako część projektu Ignition Robotics, który wprowadził ulepszenia w zakresie fizyki, renderowania oraz architektury oprogramowania. Ignition Gazebo to bardziej modułowa i elastyczna wersja symulatora, która jest lepiej dostosowana do potrzeb nowoczesnych systemów robotyki. Choć Gazebo może funkcjonować samodzielnie, zauważalnym jest podobieństwo topologii komunikacji do ROS. Program Gazebo posiada tę samą strukturę wysyłania i odbierania danych. Głównym elementem jest silnik symulacji. Obecnie domyślnym jest silnik OGRE (*Object-Oriented Graphics Rendering Engine*) dostępny na licencji otwartoźródłowej. Silnik analizuje zdarzenia i z każdą iteracją przesyła obliczone dane o pozycji danych komponentów w symulacji. W przypadku, gdy symulacja jest uruchomiona w trybie graficznym, wykorzystywane jest Gazebo GUI (Graphical User Interface) wyświetlające aktualizowany widok obiektów. Przedstawia to rys. 4.1.



Rys. 4.1. Widok symulatora Gazebo [źródło własne]

Zależnie od możliwości obliczeniowych komputera symulator Gazebo może wykonywać symulacje w czasie rzeczywistym lub spowalniać ją, o czym informuje współczynnik „*Real Time*" (z ang. czas rzeczywisty) widoczny w prawym dolny rogu rys. 4.1. Obszar symulacji wraz ze wszystkimi występującymi na nim obiektami zapisywany jest w pliku o rozszerzeniu „.word". Jest to tekstowy zapis opary o format XML (*Extensible Markup Language* – rozszerzalny język znaczników). Zawierają się w nim odniesienia do modeli CAD. W celu zastosowania realistycznych modeli 3D w symulacjach, modele CAD muszą zostać przekształcone do formatów kompatybilnych z Gazebo, takich jak STL czy OBJ. Proces ten obejmuje kilka kroków, począwszy od konwersji plików CAD przy użyciu odpowiednich narzędzi, aż po tworzenie plików SDF (*Simulation Description Format*) lub URDF (*Unified Robot Description Format*), które są następnie importowane do Gazebo. Modele te mogą być dalej dostosowywane pod kątem właściwości fizycznych oraz parametrów symulacji, co umożliwia ich pełne wykorzystanie w analizach. Takie podejście pozwala na precyzyjne odwzorowanie fizyki i geometrii modeli w wirtualnym środowisku. Silnik graficzny nie ma możliwości modyfikacji obiektów poza zmianami

w aspekcie połączeń ruchomych, nazywanych *joints* (stawy). W przykładzie na rysunku 4.1 podwozie, oznaczone kolorem zielonym, jest połączone przez obrotowe stawy z kołami, oznaczonymi kolorem czerwonym. Ruch odbywa się przez obrót kół. W przypadku kolizji ze statyczną przeszkodą podczas jazdy robota, symulator nie rozpatruje efektów takich, jak np. wgniecenia czy inne odkształcenia, jednak nie pozwala obiektowi przejechać. Czujniki w programie Gazebo traktowane są jako kolejny moduł, który odbiera dane od silnika graficznego. Następnie wprowadza symulowane błędy pomiarowe i przekazuje wartości adekwatne dla danego typu czujnika do innego tematu, z którego może odbierać dane np. kontroler robota. Istnieje możliwość dodawania własnych modułów, w tym np. konkretnych czujników, co czyni symulator bardzo elastycznym w obsłudze i rozszerzalnym dla wielu zadań. Jak zostało wcześniej wspomniane, komunikacja wewnątrz symulatora jest podobna do komunikacji w ROS i opiera się na tematach, do których poszczególne moduły mogą publikować lub odbierać dane. Pełny przebieg komunikacji został zilustrowany przez twórców Gazebo na rysunku 4.2.



Rys. 4.2. Graf komunikacji wewnętrznej w symulatorze Gazebo [72]

Jako server oznaczony na lewej części rysunku 4.2 autorzy Gazebo wskazują na silnik graficzny, który zawiera integracje z poszczególnymi modułami:

- fizyki (*Physics*),
- oddziaływania wzajemnego w polu symulacji (*Scene Broadcaster*), przekazywane do GUI (*Graphical User Interface* – Graficzny Interfejs Użytkownika) symulacji, aby tam móc wyświetlić podgląd na żywo całej symulacji,
- ingerencji z GUI przez komendy zadawane w czasie symulacji (*User Commands*),
- czujników (*Sensors*), których dane mogą być publikowane do zewnętrznego procesu, jak np. do modułu ROS, będącego kontrolerem robota,
- sygnałów wymuszeń (*Diff Drive*), będących odbieranymi od zewnętrznego procesu, sterującego robotem.

Klient (*Client*) to wymienione wcześniej GUI programu Gazebo, które zostało przedstawione na rysunku rys. 4.1. Moduły zawarte w tej sekcji to:

- widok 3D (*3D scene*), w którym znajdują się wszystkie obiekty w widoku trójwymiarowym,
- pluginy (wtyczki/moduły) domyślne (*Default plugins*), na które składa się zestaw przycisków pozwalających manipulować symulacją,
- dodatkowe moduły (*Additional plugins*), które mogą być dopisane przez użytkownika, aby np. strumieniować obraz do chmury lub nagrywać obraz z zadanymi parametrami.

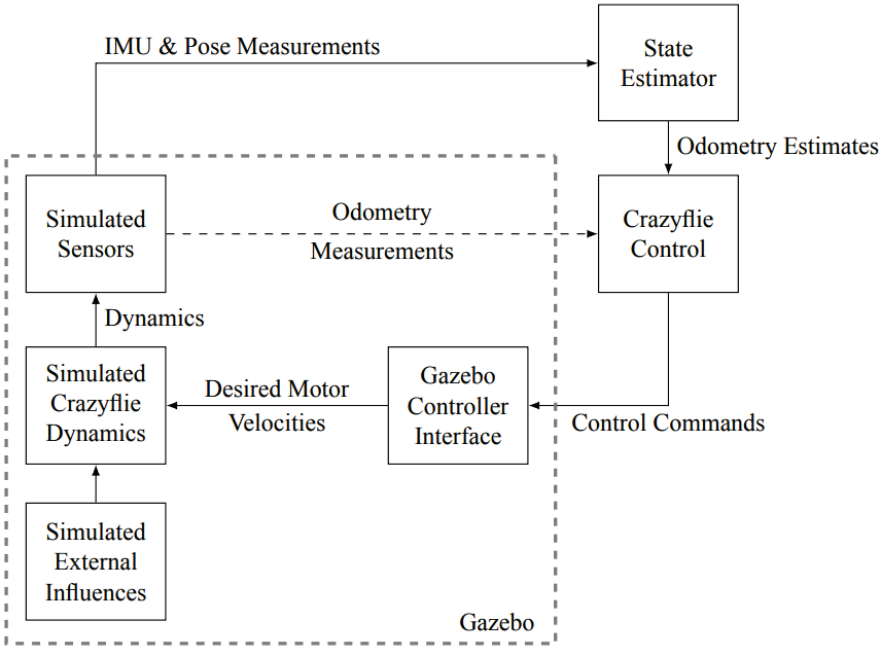
Integracja Gazebo i ROS, a także możliwość użycia ich do symulacji pojedynczych bezzałogowych obiektów latających, stała się już także tematem prac badawczych [137], [174], [199], [178]. W kolejnym podrozdziale podjęto analizę jednego z dotychczasowych rozwiązań i jego walidacje na obiekcie rzeczywistym.

4.3. Dotychczasowa koncepcja symulacji w środowisku graficznym Gazebo

Wykonanie modułu symulatora opartego o Gazebo z wykorzystaniem metasystemu ROS, podjął się badacz Giuseppe Silano [74]. Opracowany moduł umożliwia symulacje drona CrazyFlie 2.x. Przez oznaczenie „x”

należy rozumieć iteracje tego quadrocoptera, gdyż niedługo po pierwszym wypuszczeniu wersji 2.0, dokonano poprawek w zakresie elektroniki i wydano wersje 2.1. Obie wersje nie różnią się w zakresie parametrów dynamicznych tej jednostki. CrazyS jest rozwijany jako symulator, który pozwala na realistyczne testowanie i rozwój algorytmów kontroli lotu, i nawigacji. Sam moduł zawiera system plików, które automatyzują symulacje poprzez umieszczenie obiektu Crazyflie w programie Gazebo wraz z uruchomieniem metasystemu ROS. Całość należy traktować w kategorii nakładki programowej lub skryptu automatyzującego. Poza przygotowaniem środowiska dodano kontrolery dla obiektu, które odpowiedzialne są za pracę obiektu w środowisku symulacyjnym. Symulator należy do typu symulacji SITL (*Software In The Loop*). W tego typu symulacjach program jest uruchamiany wewnątrz środowiska testowego, które dostarcza symulacji realnych wartości wejściowych, sprawdza odpowiedzi oprogramowania i jego skuteczność.

Projekt symulatora stał się tłem pracy badawczej Giuseppe Silano, która ukazała się we wspomnianej monografii dotyczącej systemu ROS [114]. Wraz z Luigim Iannellim w 2020 roku przedstawili oni efekty prac nad utworzeniem modelu fizycznego obiektu Crazyflie i porównanie wyników eksperymentu wzlotu oraz zawisu pomiędzy symulacją numeryczną, a symulacją graficzną przy użyciu CrazyS [178]. Badacze przedstawili topologie swojej symulacji jak na rys 4.3.



Rys. 4.3. Schemat symulacji CrazyS [178]

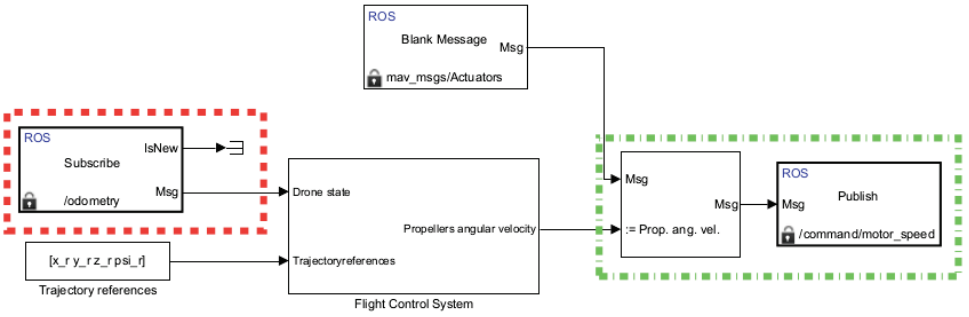
Na schemacie (rys 4.3) przerywaną linią zakreślono obszary symulacji realizowane przez program Gazebo. Składają się na nie:

- interfejs kontrolera Gazebo (*Gazebo Controller Interface*), dzięki któremu możliwe jest odbieranie komend sterujących (*Control Commands*) w czasie trwania symulacji spoza programu Gazebo i konwertowanie ich do sygnałów wykonawczych dla silników (*Desired Motor Velocities*),
- symulowane warunki środowiska (*Simulated External Influeces*), takie jak np. wpływ wiatru, które oddziałują na drona,
- symulowana dynamika Crazyflie (*Simulated Crazyflie Dynamics*), która na podstawie modelu CAD i silnika graficznego programu Gazebo generuje siły, zachowanie, dynamikę (*Dynamics*) obiektu, aktualizując przy tym widok symulacji graficznej,
- symulowane sensory (*Simulated Sensors*), które zgodnie z modułami programu Gazebo, na podstawie przekazanych danych o dynamice w wewnętrznym przepływie danych symulatora, dodają symulowany bias

oraz błąd pomiarowy czujników. Następnie udostępniane są dane o pozycji uzyskanej z symulowanych czujników inercyjnych Crazyflie (*IMU & Pose Measurements*). Istnieje również możliwość wysłania danych wprost z wewnętrznej komunikacji systemu, odzwierciedlających pozycje drona (*Odometry Measurements*), bez nałożenia błędu symulowanych czujników.

Pozostałą częścią symulacji jest przygotowanie oprogramowania sterownika lotu (*Crazyflie Control*), który będzie komunikował się z interfejsem kontrolera Gazebo, przekazując wyznaczone sygnały sterujące. Również w przypadku odczytywania danych z symulowanych czujników, wymagane jest przygotowanie programowego estymatora stanu (*State Estimator*), który będzie odbierał dane z czujników i inicjalizował sterownik lotu do wykonania obliczeń aktualnych sygnałów sterujących. Sterownik lotu został przez autorów zaprojektowany w oprogramowaniu Simulink. Badacze podają, że do opracowanego przez nich sterownika wykorzystali implementacje kontrolera wysokości (*altitude controller*) oraz kontrolera orientacji. Typy kontrolerów to PID, PI, P, przy czym jako kontroler wysokości został użyty został kontroler PID. Jako metodę obliczania pozycji kątowej obiektu autorzy wskazali możliwość użycia filtru komplementarnego.

Do wymiany danych między symulowanym dronem a sterownikiem, został użyty pakiet ROS, którego wsparcie dostępne jest w *Toolboxie* programu Matlab: ROS Toolbox. Model komunikacji przedstawia rysunek 4.4.

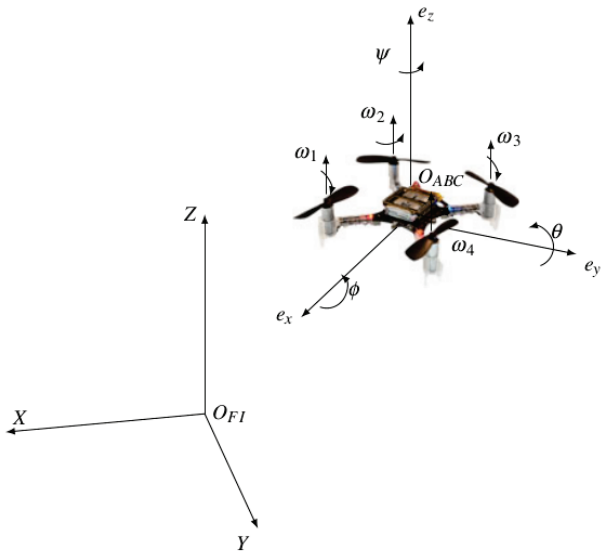


Rys. 4.4. Zamodelowany sterownik lotu w programie Simulink [178]

Kolor czerwony oznacza dane subskrybowane z metasystemu ROS, pochodzące z symulatora Gazebo. W momencie gdy dostępny jest nowy zestaw danych odomentrycznych, uruchamiana jest iteracja programu Simulink. W tym przypadku komunikacji symulacja nie ma wyznaczonego kroku całkowania, ilości iteracji, ani czasu trwania. Każda kolejna iteracja wykonania bloku sterownika lotu (*Flight Control System*) aktywowana jest na zasadzie przerwania wywoływanego przez blok subskrybenta ROS oznaczonego na czerwono. Po wyznaczeniu sygnałów sterujących przez blok sterownika lotu, sygnały sterujące w postaci prędkości każdego z czterech silników są publikowane do metasystemu ROS, z którego Gazebo subskrybuje i aktualizuje prędkości silników.

Uwagę należy zwrócić na to, że sterownik lotu został w pełni zaprojektowany w oprogramowaniu Simulink. Oznacza to, że regulatory, filtry oraz mechanizmy systemu sterownika lotu, nie pokrywają się z rzeczywistym oprogramowaniem drona Crazyflie.

Badacze w swojej pracy przygotowali również model matematyczny w celu zestawienia go z symulacją CrazyS. Przyjęty przez nich układ współrzędnych przedstawia rys. 4.5.



Rys. 4.5. Widok drona Crazyflie 2.x wraz z przyjętym układem współrzędnych [178]

W przyjętym przez badaczy rozwiązaniu występują dwa układy współrzędnych:
O_{FI} – układ globalny,
O_{ABC} – układ lokalny jednostki Crazyflie.

Do wyznaczenia równań ruchu badacze użyli kątów Eulera φ , θ , ψ. Prędkości obrotowe silników oznaczone są jako $\omega_{1\text{-}4}$.

W przypadku badań prowadzonych przy użyciu CrazyS, badacze dostarczyli implementacje sterownika lotu przygotowaną w programie Simulink. Ta sama implementacja sterownika lotu posłużyła im zarówno do symulacji w programie Gazebo, jak i symulacji numerycznej w oprogramowaniu Simulink. Różnica pomiędzy wynikami eksperymentów jest efektem jakości zamodelowania drona i dobrania współczynników. Rozwiązanie to nie rozpatruje jednak porównania i walidacji obu koncepcji z fizyczną jednostką, której funkcjonowanie jest bardziej złożone ze względu na sterownik lotu, który pełni więcej funkcji niż projektowane w oprogramowaniu Simulink kontrolery wysokości i kątów oraz filtr komplementarny.

4.3.1. Proponowana koncepcja walidacji

Badania omówione w poprzednim podrozdziale zostały odwzorowane tak, aby zestawić je z proponowanymi konwencjami: symulacji w środowisku graficznym oraz modelu matematycznym, przedstawionym w rozdziale 3. Różnicą w niniejszej rozprawie w podejściu do modelowania, opisaną w poprzednim rozdziale, jest dodanie wpływu układu silnik-śmigło i uwzględnienie go w modelu przy użyciu wielomianów czwartego rzędu. Dodatkowo wprowadzoną zmianą w podejściu do symulacji z użyciem silnika graficznego, w stosunku do wcześniej przedstawionej koncepcji, jest nie tylko wprowadzenie dynamiki jednostki przez posłużenie się modelem CAD bezzałogowego obiektu latającego, ale także użycie oprogramowania sterownika lotu jakie jest używane w latających jednostkach. Zgodnie z przyjętym kierunkiem prac rozpatrywanym sterownikiem lotu jest ArduPilot.

Jako walidacje wszystkich rozpatrywanych koncepcji posłużono się fizycznym dronem Crazyflie 2.1. Dane z eksperymentu na tej jednostce

wykorzystano jako dane arbitralne do oceny wszystkich koncepcji. Jako eksperymenty do walidacji wybrano rozpatrywany wzlot przy identycznych warunkach. Rozważony został także, przedstawiony na koniec rozdziału 3, eksperyment z dwukrotnym wzlotem UAV, a następnie powrotem do poprzedniej pozycji. Do wykazania poprawności wyników względem fizycznego wzlotu drona użyto kryterium IAE – *Integral Absolute Error* (z ang. całka wartości bezwzględnej błędu), która w wielu pracach wybierana jest jako najlepsze kryterium błędu w sterowaniu obiektów [142], [103], [171].

Celem badania było przeanalizowanie 6 eksperymentów dla tych samych scenariuszy. Rozpatrywane scenariusze to: pierwszy: wzlot i zawis drona, oraz drugi: wzlot, ponowny wzlot, powrót do poprzedniej pozycji. Wyniki obu scenariuszy zostały zaprezentowane na końcu rozdziału. Przeprowadzono sześć eksperymentów w dwóch zadanych scenariuszach. Kolejno:

1. Model matematyczny drona Crazyflie opracowany przez G. Silano oraz L. Iannelliego, zaimplementowany w oprogramowaniu Simulink, wraz ze sterownikiem lotu odwzorowującym pracę badaczy, uwzględniającym podane przez nich nastawy regulatorów i ich typy.
2. Model matematyczny quadrocoptera opracowany w poprzednim rozdziale tej pracy z parametrami dobranymi dla bezzałogowego obiektu latającego Crazyflie, zaimplementowany w oprogramowaniu Simulink. Dla weryfikacji tylko między modelami, do tego modelu użyty został ten sam zamodelowany sterownik lotu, jak opisany w punkcie pierwszym.
3. Model CAD Crazyflie z użyciem symulatora CrazyS. Dzięki obszerności pracy G. Silano oraz L. Iannelliego, a także publicznej dostępności repozytorium CrazyS i prawidłowo napisanej dokumentacji, możliwym było odwzorowanie pracy badaczy. Jako sterownik lotu został wykorzystany zamodelowany w oprogramowaniu Simulink sterownik z punktu pierwszego. Został wykorzystany do sterowania dzięki pakietowi ROS Toolbox, umożliwiającego komunikacje ROS w środowisku Matlab/Simulink. Ideowy schemat tej komunikacji znajduje się na rysunku 4.6.
4. Model CAD zaimplementowany do programu Gazebo przez Exporter ROS dla programu Solidworks. Jako sterownik lotu wykorzystane zostało

tutaj oprogramowanie ArduPilot. Umożliwia ono połączenie z Gazebo jako zewnętrzny proces (widok na rysunku 4.4). Wartościami zadanymi dla sterownika lotu steruje skrypt ROS napisany w języku Python, zadający komendę wzlotu na żądaną wysokość. Skrypt komunikuje się z oprogramowaniem ArduPilot, uruchomionym obok programu Gazebo przez MAVROS, czyli wysyłając komendy zgodne z protokołem MAVLink.

5. Fizyczna jednostka Crazyflie 2.1. W rozpatrywanym UAV wgrane zostało oprogramowanie ArduPilot w stabilnej wersji 4.4.4. dla drona Crazyflie. Jako skrypt zadający wartości został wykorzystany skrypt ROS z punktu 4. Tak samo, jak dla jednostki w graficznym środowisku symulacyjnym zadaje wysokość wzlotu, jaką ma wypracować sterownik. Odczyt pozycji odbywa się przez osobny skrypt ROS, który zbiera wysyłane przez oprogramowanie ArduPilot dane o wysokości i archiwizuje je do porównania z wynikami symulacyjnymi.
6. Fizyczna jednostka Crazyflie 2.1. wraz z dalmierzem laserowym umieszczonym pod nią. Aby uzyskać dane arbitralne, niezależne od komunikacji z obiektem i jego wbudowanymi czujnikami, jako ostatnie źródło danych wybrano laserowy czujnik odległości, który zebrał dane podczas wzlotu opisanego w punkcie 5.

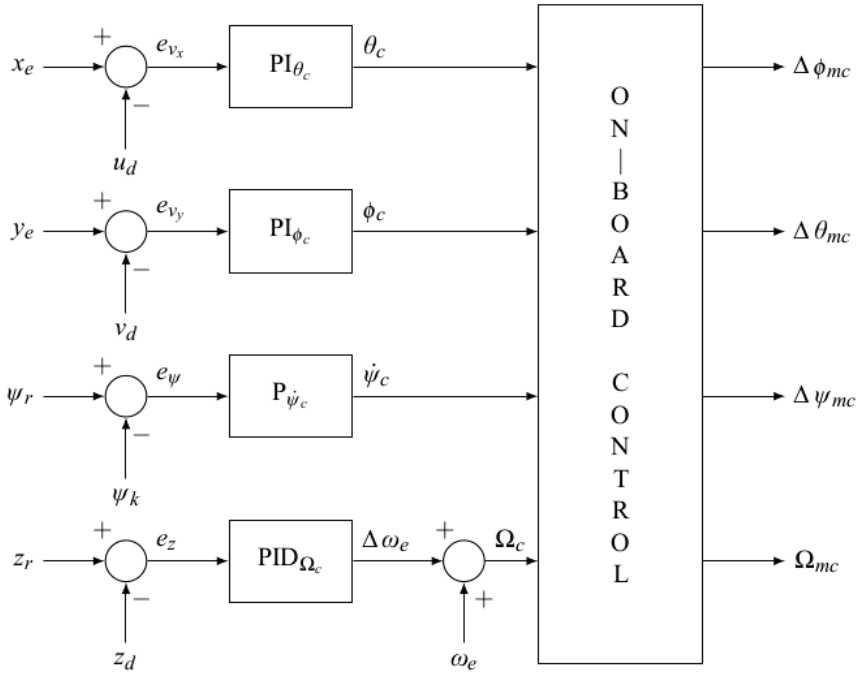
Tabelaryczne podsumowanie przygotowania 6 eksperymentów znajduje się w tabeli 4.2, na końcu kolejnego podrozdziału.

4.3.2. Przygotowanie eksperymentów

Jako model CAD bezzałogowego obiektu latającego Crazyflie posłużył model przygotowany przez producenta Bitcraze [91] przedstawiony na rys. 4.6.

Rys. 4.6. Model CAD Crazyflie 2.X [91]

Parametry obiektu przedstawionego na rysunku 4.6 są zamieszczone w załączniku nr 1. Dla eksperymentów 1-3 odtworzono sterownik lotu zamodelowany w programie Simulink. Użyte kontrolery to: kontroler wysokości oraz kontroler pozycji kątowej. Przedstawione zostały na rysunku 4.7.



Rys. 4.7. Struktura kontrolerów przyjęta przez G. Silano and L. Iannelliego [178]

Nastawy powyższych regulatorów, a także wartości ograniczające przedstawiono w tabeli 4.1.

Tab. 4.1. Nastawy regulatorów i adekwatne wartości ograniczeń zgodne z pracą G. Silano and L. Iannelliego [178]

Kontroler	**Nastawy**	**Ograniczenia**
PI_{θ}	$K_P = 3.59$	$[-\pi/6, \pi/6]$
	$K_I = 5.73$	
PI_{φ}	$K_P = -3.59$	$[-\pi/6, \pi/6]$
	$K_I = -5.73$	
$P_{\dot{\psi}}$	$K_P = 0.0914$	$[-1.11\pi, 1.11\pi]$
PID_{Ω}	$K_P = 70$	[5156, 8163]
	$K_I = 3.15$	
	$K_D = 373$	

Dla regulacji pozycji kątowej badacze wskazali na użycie filtru komplementarnego. Nie podali jednak wykonanej przez nich implementacji. Aby odwzorować ten filtr wykorzystano gotowy blok dla oprogramowania Simulink [81], dostępny w pakiecie Sensor Fusion and Tracking Toolbox dla programu Matlab od wersji R2023a.

Do połączenia pomiędzy sterownikiem lotu zamodelowanym w oprogramowaniu Simulink a dronem Crazyflie, odwzorowując pracę G. Silano and L. Iannelliego, wykorzystano pakiet ROS Toolbox dla oprogramowania Simulink. Proponowaną przez badaczy nakładkę na program Gazebo pod nazwą CrazyS pobrano z repozytorium G. Silano i zainstalowano w systemie Ubuntu 18.04 LTS (*Long Time Support*) z metasystem ROS Melodic Morenia LTS. Komunikacje przetestowano z oprogramowaniem Matlab/Simulink 2023a, zainstalowanym na tym samym systemie.

Dla eksperymentu czwartego użyto tego samego systemu i oprogramowań. Model CAD otworzony w programie SolidWorks pod systemem Windows, został wyeksportowany dla Gazebo dzięki wtyczce programu SolidWorks o nazwie: SolidWorks to URDF Exporter [93]. Wtyczka umożliwia wygenerowanie nieruchomych części, a także pliku

„.urdf", w którym zawarte są informacje na temat połączeń (*joints*) między poszczególnymi częściami.

Należy również w pliku .urdf zadeklarować, jakie czujniki ma zawierać obiekt, w którym miejscu mają się one znajdować na obiekcie i jak powinien być ustawiony dany czujnik. Standardem programu Gazebo są moduły czujników uniwersalnych. Należy przez to rozumieć, że nie ma rozróżnienia na modele czujników, a zestaw czujników pokładowych drona zapisany jest jako IMU (*Inertial Measure Unit*) i jest identyczny dla wszystkich jednostek, niezależnie od czujników, jakie znajdują się na pokładzie konkretnej jednostki. Możliwym jest napisanie modułu czujnika z rozdzielczością i błędem odpowiadającym konkretnej katalogowej jednostce pomiarowej.

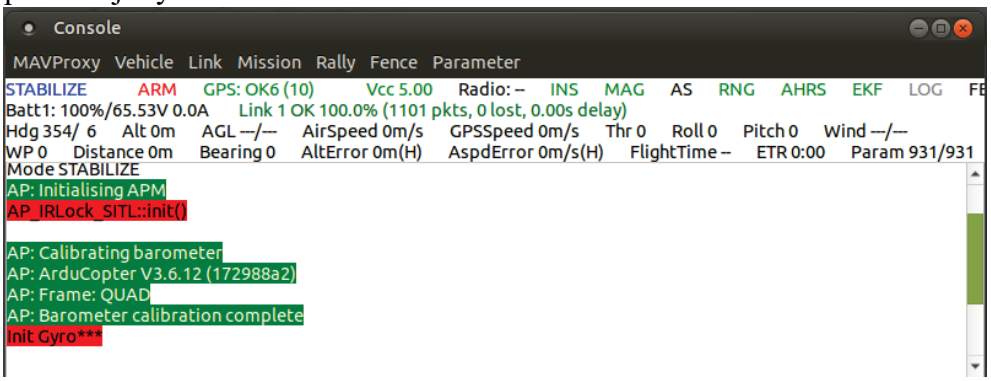
Dron został wstawiony do programu Gazebo przez plik o rozszerzeniu .word. Przygotowanie symulacji wraz z uruchomieniem wszystkich programów, zautomatyzowane zostało skryptem bash powłoki systemu Linux. Uruchamia on kolejno:

1. Moduł Roscore w systemie Linux,
2. Oprogramowanie sterownika lotu ArduPilot w systemie Linux,
3. Plik *launch* metasystemu ROS, który uruchamia:
   A. Gazebo wraz z plikiem .word, dzięki któremu symulator ładuje potrzebne obiektu i moduły programu Gazebo,
   B. Moduł MAVROS,
   C. Skrypt ROS napisany w języku programowania Python, który ma zadanie zadać żądanie wzlotu na ustaloną wysokość.

Oprogramowanie sterownika lotu ArduPilot może być uruchamiane zarówno w systemach czasu rzeczywistego, jak i systemach operacyjnych Linux. Dzięki temu może służyć jako oprogramowanie dla minikomputera Raspberry Pi wraz z płytką Navio, wspomnianą w rozdziale 3. Może też również być uruchomione na komputerze osobistym PC z systemem Linux. Należy uruchomić je przed modułem MAVROS. W nim należy podać adres, a także port pod jakim oprogramowanie ArduPilot może komunikować się wewnątrz systemu operacyjnego. Jest to opcja dostępna tylko dla systemu Linux [65]. Sterownik inicjalizuje się z dronem w programie Gazebo.

Wymagane jest, aby ten dron zawierał czujniki, określone w rozdziale 3 jako minimalne wymagania sprzętowe. Przy czym czujniki te, jak wspomniano, deklaruje się jako IMU dla każdej jednostki latającej.

Uruchomione w systemie Linux oprogramowanie ArduPilot prezentuje rys. 4.8.



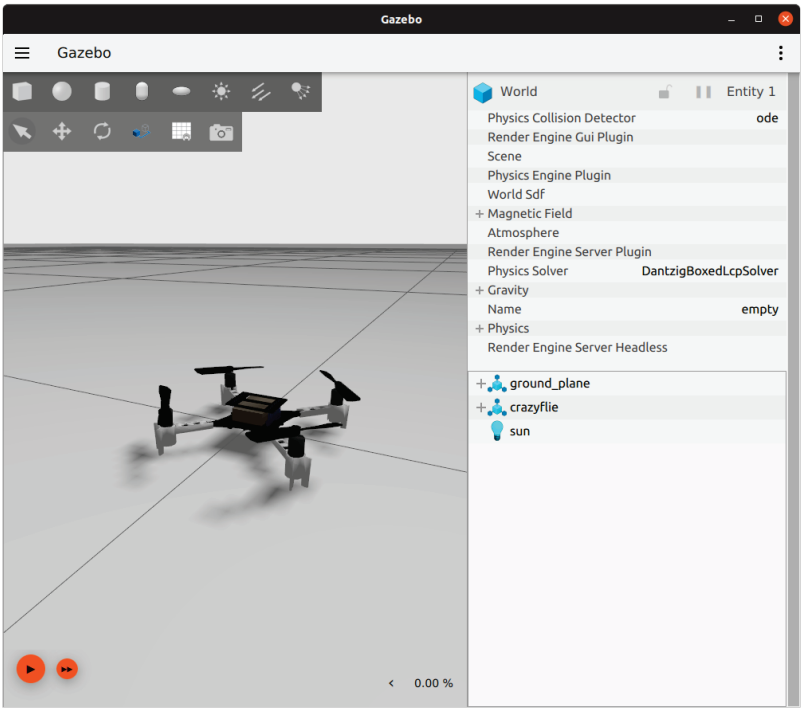
Rys. 4.8. Widok konsoli oprogramowania sterownika lotu ArduPilot dla jednostki typu copter [źródło własne]

Konsola sterownika lotu jest połączona w relacji jeden do jednego z każdym dronem, dla którego wywołane zostało uruchomienie oprogramowania ArduPilot w systemie Linux. Przedstawia takie informacje, jak między innymi: tryb lotu, w jakim aktualnie jest dany obiekt, status uzbrojenia obiektu, poziom baterii etc.

Wywołany w systemie operacyjnym Linux sterownik lotu ArduPilot jest tym samym oprogramowaniem, jakie jest wgrywane na jednostki fizyczne. Każda instancja uruchomionego oprogramowania zawiera plik konfiguracyjny, który jest unikalny dla poszczególnej jednostki latającej. Zawiera on dane o procesie, jaki ma obsłużyć oprogramowanie ArduPilot, a także o parametrach sprzętowych danej jednostki. W przypadku uruchomienia go w systemie Linux na potrzeby symulatora, należy użyć pliku z listy dostępnych do pobrania na stronie producenta oprogramowania sterownika lotu dla uniwersalnego obiektu typu quadrocopter, uruchamianego w systemie Linux. Po wywołaniu konsoli dokonuje się automatyczna kalibracja czujników wraz z oprogramowaniem sterownika lotu. W tym czasie nie można uzbroić drona, dlatego wymagane jest, aby

poczekać do momentu, kiedy konsola pokaże, że quadrocopter może zostać uzbrojony. Można to również sprawdzić w skrypcie ROS.

Widok drona Crazyflie w programie Gazebo przedstawia rys. 4.9.



Rys. 4.9. Widok Crazyflie w programie Gazebo [źródło własne]

Na potrzeby zebrania danych z lotu w symulacji do skryptu powłoki systemu Linux uruchamiającego właściwe dla eksperymentu skrypty, dodany został program pobierający dane o wysokości z modułu MAVROS i archiwizujący je do porównania z pozostałymi wynikami.

Dron Crazyflie posiada dwa mikrokontrolery STM32F405, które są głównymi kontrolerami sterującymi dronem, oraz nRF51822, który odpowiada za komunikacje oraz zarządzanie zasobami energii [68]. Dzięki układowi nRF51822 możliwe jest zdalne sterowanie drona z poziomu aplikacji mobilnej z wykorzystaniem Bluetooth lub poprzez odpowiednie moduły radiowe i wykorzystujące technologie ANT+, również wspieraną przez mikrokontroler komunikacyjny. ANT+ jest bezprzewodową

technologią, która pozwala przesyłać dane na inne urządzenia z większym zasięgiem niż Bluetooth. Jednostka Crazyflie z racji na kompaktowy kształt i gabaryt nie zawiera dedykowanej anteny dla technologii ANT+. Jest to dron przeznaczony do latania w zamkniętych przestrzeniach, znakomicie nadający się do powtarzalnych prac, w tym badawczo rozwojowych, jednak nieadekwatny do misji terenowych. Został on wybrany do walidacji graficznego środowiska symulacyjnego dzięki możliwości użycia w warunkach kontrolowanych. Należy przez to rozumieć testy wewnątrz budynku, niezakłócone przez warunki zewnętrzne. Umożliwia to maksymalną powtarzalność prób istotnych przy walidacji. Dodatkowo był on tematem analizowanej pracy [178] i, **aby porównać wyniki środowiska symulacyjnego konieczne było odwzorowanie pracy na tej samej jednostce**.

Do zbierania danych o wysokości z obiektu został użyty program z eksperymentu nr 4, gdzie za pośrednictwem modułu MAVROS następuje cykliczne zbieranie danych i zapisywanie informacji o pozycji liniowej drona. W przypadku eksperymentu nr 5 różnica polega na wywołaniu modułu MAVROS z innymi parametrami startowymi, aby zamiast połączenia z UAV w symulatorze, Gazebo nawiązał łączność z fizycznym dronem.

Do uruchomienia potrzebnych programów został użyty skrypt automatyzujący powłoki systemu Linux. Poza programem do archiwizacji danych, oraz modułem MAVROS, został także uruchomiony moduł Roscore. Moduł MAVROS został zainicjowany do komunikacji z dronem Crazyflie przy pomocy Crazyradio PA, określanym też Crazy PA [67]. Crazy PA to moduł radiowy USB o dużym zasięgu oparty na układzie nRF24LU1+ firmy Nordic Semiconductor. Obsługuje protokół ANT+. Pozwala sterować dronem Crazyflie z poziomu aplikacji komputerowej lub dedykowanego oprogramowania stacji naziemnej. Aby wyeliminować różnice wynikające z warunków losowych, wpływające na ocenę jakościową danych z czujników pokładowych drona, zdecydowano, aby zbierać dane jednocześnie podczas jednego lotu UAV. Do obiektu Crazyflie wgrano oprogramowanie ArduPilot Copter 4.4.4. Skonfigurowano je dla zadanego typu jednostki.

Jako czujnika arbitralnego użyto dalmierza laserowego o wysokiej dokładności. Dzięki temu, że scenariusze obu eksperymentów zakładają

jedynie ruch drona w osi Z, możliwym jest wykorzystanie dalmierza laserowego, który mierzy odległość do płytki drona, będącej jego podwoziem. Jest to bardzo precyzyjny pomiar, który może być uznany za arbitralny dzięki temu, że użyty dalmierz mierzy bezpośrednio odległość między płytką sterującą a brzegiem czujnika. Aby obliczyć wysokość drona, należy użyć prostego wzoru:

$$h = d\text{-}d_0 \tag{4.1}$$

gdzie:
h – wysokość drona,
d – dystans pomiędzy płytką elektroniczną drona a brzegiem dalmierza,
d_0 – początkowy dystans między płytką elektroniczną drona a brzegiem dalmierza zmierzony w pozycji wyjściowej, czyli przed startem UAV.

Dalmierz konwertuje dane do sygnału analogowego 0-24 V. Napięcie jest odczytywane przez kartę pomiarową z przetwornikiem ADC o wysokiej dokładności i częstotliwości pomiarów. Dane zapisywane są do pliku o formacie „.csv" wraz z oznaczeniem czasu (*timestamp*), aby mogły zostać zestawione z wynikami pochodzącymi z pozostałych eksperymentów.

W Tab. 4.2. zestawiono podsumowanie informacji o eksperymentach, służących do walidacji rozwiązań symulacyjnych względem danych z jednostki fizycznej. Jako dane arbitralne wykorzystano dane zebrane podczas eksperymentu nr 6.

Tab. 4.2. Podsumowanie metodyki przygotowania eksperymentów [źródło własne]

ID	**Metoda symulacji**	**Odwzorowanie**	**Oprogramowanie**	**Sterownik lotu**	**Dane wejściowe**
1	Numeryczna	Tak, pełne	Simulink	Symulacyjny: Regulatory kątów i wysokości	Artykuł G. Silano i L. Iannelli [178]

2	Symulacja w środowisku graficznym	Większość tak, własny filtr komplementarny	CrazyS (Gazebo oraz ROS) i Simulink	Symulacyjny: Regulatory kątów i wysokości + filtr komplementarny	Artykuł G. Silano i L. Iannelli [178]
3	Numeryczna	Większość nie, własny model i dane, nastawy regulatorów odwzorowane	Matlab/Simulink	Symulacyjny: Regulatory kątów i wysokości	Pomiary fizyczne, analiza modelu CAD
4	Symulacja w środowisku graficznym	Nie, własne środowisko symulacyjne i niezależny model CAD	Gazebo, ROS i skrypty systemu Linux	Oprogramowanie ArduPilot dla quadrocoptera	Pomiary fizyczne, analiza modelu CAD,
5	Eksperyment fizyczny	Nie	Dron Crazyflie, ROS i skrypty systemu Linux	Oprogramowanie ArduPilot dla quadrocoptera	Pomiary fizyczne, model CAD
6	Eksperyment fizyczny - **arbitralny**	NIE	Dron Crazyflie, ROS i skrypty systemu Linux	Oprogramowanie ArduPilot dla quadrocoptera	Pomiary fizyczne, model CAD

4.3.3. Wyniki eksperymentów walidacji

Zgodnie z zarysowanym w tabeli 4.2. przebiegiem eksperymentów przygotowano dwa scenariusze walidacji:

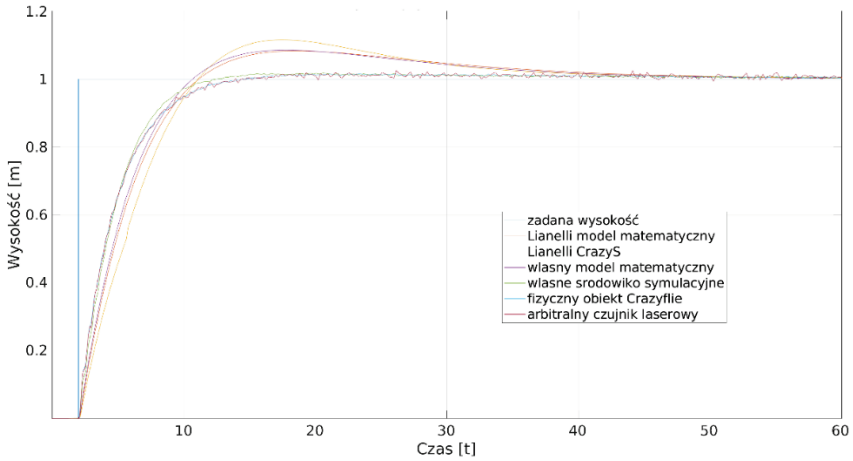
1. Wzlot drona na wysokość jednego metra i zawis w powietrzu (*hover test*). Żądaną wysokość ustalono na pułapie 1 metra zgodnie z pracą Iannelliego. Zadaniem tego scenariusza było weryfikacja odwzorowania pracy włoskiego badacza i porównanie jego wyników z obiektem rzeczywistym.
2. Wzlot drona na wysokość jednego metra i utrzymanie pozycji. Następnie wzlot na wysokość 2 metrów. Obniżenie pozycji i powrót do poziomu wysokości 1 m. Osiągnięcie zadanej wysokości oraz obniżenie pozycji ma na celu sprawdzenie wpływu dodania oprogramowania sterownika lotu ArduPilot.

Do walidacji wyników eksperymentów względem arbitralnego lotu quadrocoptera Crazyflie przyjęto wspomniane kryterium wartości bezwzględnej całki błędu IAE. Eksperymenty wykonywano w kolejności zgodnej z ich numeracją. Dane z eksperymentu 5. i 6. są wynikami zebranymi z dwóch różnych systemów pomiarowych, zarejestrowanymi podczas jednego lotu. Do przeprowadzenia eksperymentów użyto komputera z systemem Linux Ubuntu 18.4 LTS wraz z zainstalowanymi ROS Melodic Morenia LTS i oprogramowaniem Matlab/Simulink 2023a. Do całkowania użyto metody ode45 [84].

Dane z wszystkich eksperymentów nakładano na jeden wykres w celu możliwości wizualnej analizy wyników. W przypadku opóźnień związanych z koniecznością uzbrojenia sterownika lotu przyjęto, że wszystkie wyniki zostały przeskalowane, aby wykres dla każdego eksperymentu zaczynał się 2 sekundy przed rozpoczęciem lotu.

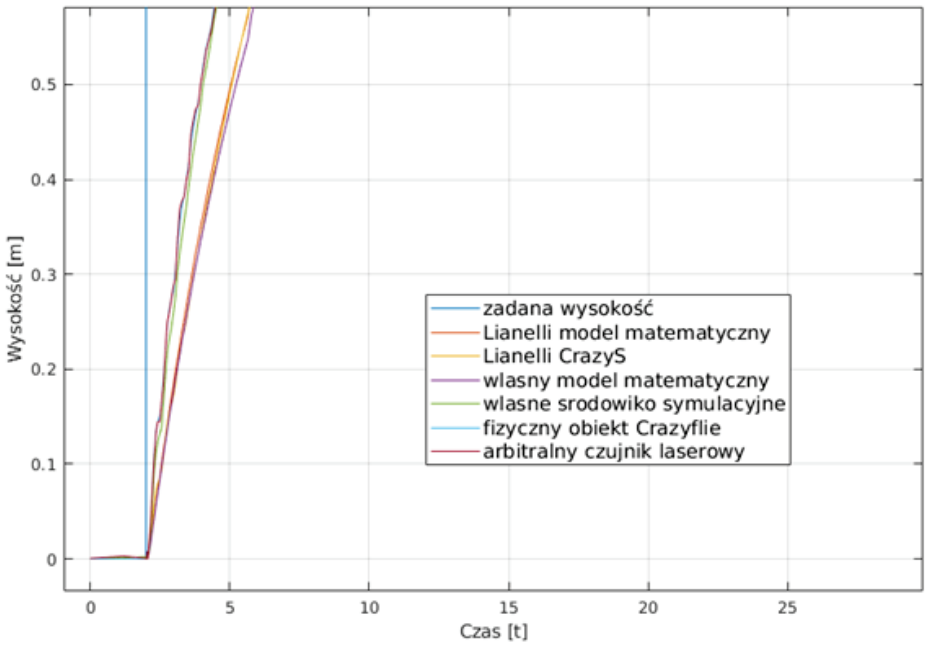
Nastawy dla regulatorów przyjęto zgodnie z przedstawionymi w tabeli 4.6. Dla symulacji z użyciem symulatora CrazyS odczytano dane z czujników z programu Gazebo.

4.3.3.1. Scenariusz 1 – Wzlot drona



Rys. 4.10. Graficzne przedstawienie rezultatu scenariusza nr 1 dla 6 eksperymentów wraz z oznaczeniem zadanej wysokości [źródło własne]

Na rys. 4.10. zaprezentowano eksperymenty dla scenariusza zbieżnego z porównywaną pracą [178]. Można zauważyć, że dla eksperymentów uwzględniających oprogramowanie sterownika lotu ArduPilot linie wykresu mają kształt nie będący linią prostą. Różnice można także zauważyć na początku lotu podczas wzbijania drona.



Rys. 4.11. Początkowa faza lotu dla scenariusza nr 1 [źródło własne]

Na rys. 4.11. dla eksperymentów 2, 4, 5, 6, czyli tych wykonanych w programie Gazebo lub przy użyciu fizycznej jednostki Crazyflie, można zauważyć „efekt ziemi" (ang. *ground effect*). Jest to zjawisko aerodynamiczne, które występuje, gdy dron, helikopter lub inne statki powietrzne znajdują się blisko powierzchni ziemi (lub innej powierzchni, np. wody). W efekcie ziemi pojawia się wzrost siły nośnej i zmniejszenie oporu aerodynamicznego, co sprawia, że maszyna może unosić się w powietrzu z mniejszym zużyciem energii, niż gdyby znajdowała się wyżej. Efekt ten jest szczególnie odczuwalny podczas startu i lądowania, gdy odległość od ziemi jest mała.

Aby porównać wyniki eksperymentów przygotowano tabelaryczne zestawienie wyników porównania eksperymentów 1-5 z eksperymentem nr 6, który został przyjęty jako arbitralny.

Tab. 4.3. Zestawienie wyników regulacji zgodnie z kryterium całki wartości bezwzględnej błędu (IAE) dla scenariusza nr 1 [źródło własne]

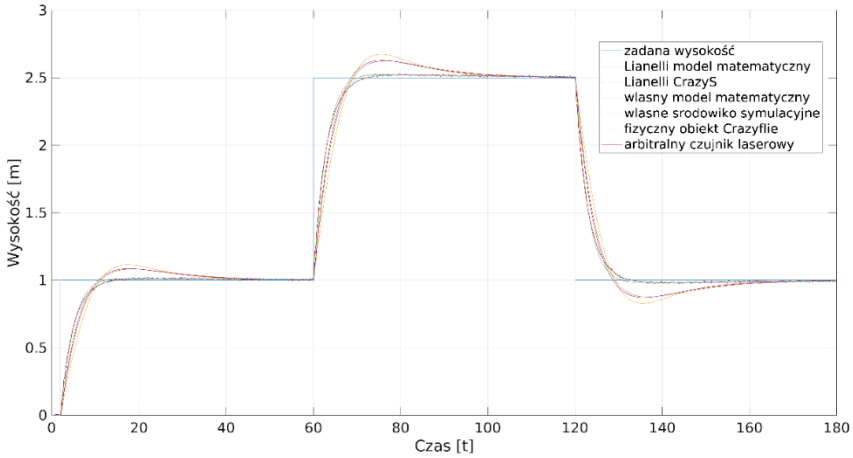
ID	Nazwa eksperymentu	IAE
1	Odwzorowanie symulacji numerycznej z artykułu Iannelliego [178]	13.565
2	Odwzorowanie symulacji CrazyS z artykułu Iannelliego [178]	18.762
3	Opracowany autorski model matematyczny wraz z kontrolerami użytymi przez Ianellego [178]	11.326
4	Symulacja w środowisku graficznym wraz ze sterownikiem lotu ArduPilot	0.672
5	Fizyczny lot drona Crazyflie wraz z wgranym oprogramowaniem sterownika lotu ArduPilot	0.434

Tabela 4.3. przedstawia wyniki liczbowego porównania zgodnie z przyjętym kryterium całki wartości bezwzględnej błędu. Potwierdza ona podział eksperymentów zgodnie z jakością na dwie grupy. Pierwsza z nich zawierająca eksperymenty 1-3 cechuje się znacząco gorszą jakością odwzorowania zachowania rzeczywistego drona.

4.3.3.2. Scenariusz 2 – Wzlot drona, ponowny wzlot i powrót do pierwotnego pułapu

Drugi scenariusz został zrealizowany dla wysokości 1 oraz 2,5 metra. Na początku wykonano wzlot na wysokość jednego metra. W 60-ej sekundzie

eksperymentu zadano wzlot na wysokość 2,5 metra. Po czym w 120-ej sekundzie wysłano wymuszenie powrotu do wysokości 1 metra. Wyniki scenariusza przedstawia rys. 4.12.



Rys. 4.12. Graficzne przedstawienie rezultatu scenariusza nr 2 dla 6 eksperymentów wraz z oznaczeniem zadanej wysokości [źródło własne]

Na rysunku 4.12 widać podobne zależności jak w pierwszym scenariuszu. Należy zauważyć, że odpowiedź układu w każdym eksperymencie osiąga żądaną wartość. Dla eksperymentów z użyciem oprogramowania sterownika lotu ArduPilot 4-6 występuje minimalne opóźnienie w wykonaniu zadanej komendy względem pozostałych eksperymentów o numerach 1-3. Wynika to bezpośrednio z oprogramowania sterownika lotu i nie jest efektem synchronizacji startu eksperymentów w punkcie czasowym równym 2s.

Wyliczone wartości IAE przedstawia tabela 4.4.

Tab. 4.4. Zestawienie wyników regulacji zgodnie z kryterium całki wartości bezwzględnej błędu (IAE) dla scenariusza nr 2 [źródło własne]

ID	Nazwa eksperymentu	IAE
1	Odwzorowanie symulacji numerycznej z artykułu Iannelliego [178]	41.253
2	Odwzorowanie symulacji CrazyS z artykułu Iannelliego [178]	59.534
3	Opracowany autorski model matematyczny wraz z kontrolerami użytymi przez Ianellego [178]	35.423
4	Symulacja w środowisku graficznym wraz ze sterownikiem lotu ArduPilot	1.634
5	Fizyczny lot drona Crazyflie wraz z wgranym oprogramowaniem sterownika lotu ArduPilot	1.146

Tab. 4.4. odwzorowuje dysproporcje między eksperymentami zawierającymi używane w fizycznych dronach oprogramowanie ArduPilot, a eksperymentami opartymi na kontrolerach zamodelowanych w programie Matlab/Simulink.

4.3.4. Analiza wyników walidacji

Przeprowadzone zostały dwa scenariusze zadań do walidacji jakości metod badań prowadzonych na bezzałogowych obiektach latających. Jednym możliwym kierunkiem ruchu drona, z racji na porównanie z pracą Iannelliego [178], gdzie nie zawarto nastaw dla kontrolerów X i Y był ruch w płaszczyźnie Z. Umożliwiło to umiejscowienie czujnika laserowego pod dronem. Dzięki temu było możliwe zbieranie danych o wysokości UAV z niezależnego zewnętrznego urządzenia pomiarowego, mierzącego bezpośrednio dystans z dużą precyzją. Dane z tego eksperymentu zostały potraktowane jako arbitralne i do nich zostały odniesione wyniki pozostałych eksperymentów. Należy zauważyć, że wygląd danych arbitralnych przypomina kształtem zaszumiony sygnał. Wygląd danych z czujnika laserowego nie jest kwestią szumu. Odzwierciedla niewielkie oscylacje drona Crazyflie w locie. Należy mieć na uwadze, że są one spowodowane korekcją

pozycji kątowej w osiach X i Y i dynamicznych odpowiedziach sterownika lotu, który utrzymuje obiekt w zadanej pozycji kątowej.

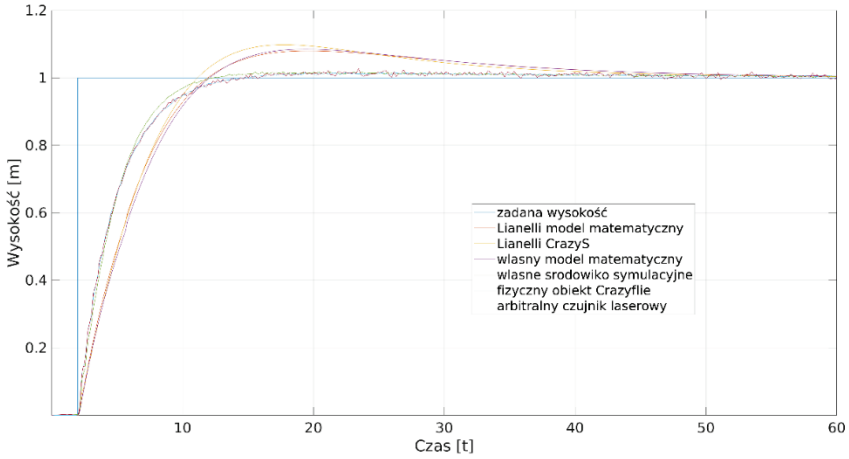
Zestawienia wyników dla kryterium IAE przedstawionych w tabelach 4.3. i 4.4. są zbieżne i wskazują na zależność, według której **eksperymenty z uwzględnieniem oprogramowania sterownika lotu ArduPilot, czyli fizyczny wzlot drona oraz symulacja w przygotowanym środowisku graficznym, są wzajemnie spójne**. Nieco odmienne zachowania wykazują eksperymenty, gdzie do kontroli wysokości zastosowano sterownik oparty na kontrolerach zamodelowanych w programie Matlab/Simulink. Należy zwrócić uwagę, że dla obu tych grup zadane zostały różne nastawy regulatorów. Inna okazała się również sama struktura regulatorów, a także typy filtrów użytych do regulacji pozycji kątowej. Nastawy dla regulatorów w eksperymentach 1-3 zostały dobrane zgodnie z tabelą 4.3. Nastawy w oprogramowaniu ArduPilot są kalibrowane automatycznie. Samo oprogramowanie wymaga określenia typu obiektu, następnie podczas lotu początkowe wartości nastaw są korygowane i zapisywane do pamięci nieulotnej sterownika lotu. Wartości nastaw odczytanych z drona Crazyflie z oprogramowaniem ArduPilot przedstawiają tabela 4.5.

Tab. 4.5. Nastawy ArduPilot odczytane z drona Crazyflie [źródło własne]

Kontroler	**Nastawy**
PID_{θ}	$K_P = 3.23$
	$K_I = 7.08$
	$K_D = 0.45$
PID_{φ}	$K_P = -3.40$
	$K_I = -6.89$
	$K_D = 0.44$
PID_{ψ}	$K_P = 0.60$
	$K_I = 0.01$
	$K_D = 0$

PID_{Ω}	$K_P = 38.40$
	$K_I = 1.25$
	$K_D = 3.32$

W oprogramowaniu ArduPilot struktura wszystkich regulatorów jest zbieżna i ma postać regulatora PID. Możliwe są wartości zerowe w poszczególnych polach, co zmienia strukturę regulatora. Nastawy wymienione w tabeli 4.5 użyte były dla eksperymentów 4, 5, a także 6 (arbitralny). Powtórzono eksperyment zadając te nastawy dla pozostałych eksperymentów. Wyniki powtórzonego scenariusza nr 1 z nastawami z tabeli 4.5 dla każdego eksperymentu przedstawia rys. 4.13.



Rys. 4.13. Wyniki graficzne ponownie przeprowadzonych eksperymentów ze scenariusza nr 1 z nastawami regulatorów z oprogramowania ArduPilot [źródło własne]

Oprócz przedstawienia graficznego na rys. 4.13, analogiczne do poprzednich eksperymentów, przygotowano tabelaryczne porównanie przy użyciu kryterium IAE:

Tab. 4.6. zestawienie wyników regulacji zgodnie z kryterium całki wartości bezwzględnej błędu (IAE) dla scenariusza nr 1 przy użyciu nastaw regulatorów odczytanych z drona Crazyflie [źródło własne]

ID	**Nazwa eksperymentu**	**IAE**
1	Odwzorowanie symulacji numerycznej z artykułu Ianellego [178]	12.320
2	Odwzorowanie symulacji CrazyS z artykułu Ianellego [178]	19.043
3	Opracowany autorski model matematyczny wraz z kontrolerami użytymi przez Ianellego [178]	11.091
4	Symulacja w środowisku graficznym wraz ze sterownikiem lotu ArduPilot	0.631
5	Fizyczny lot drona Crazyflie wraz z wgranym oprogramowaniem sterownika lotu ArduPilot	0.403

Wyniki przedstawione w tabelach 4.3. i 4.6. są zbliżone. Wskazuje to na różnicę pomiędzy eksperymentami 1-3 oraz 4-6, wynikającą w użycia oprogramowania ArduPilot zamiast kontrolerów opracowanych w środowisku obliczeniowym Matlab/Simulink. Różnica w nastawach dobranych przez Iannelliego zaprezentowanych w tabeli 4.1, a także nastawach odczytanych z fizycznego drona w tabeli 4.5, nie jest istotna pod kątem rozbieżności wyników próby wzlotu według scenariusza nr 1.

Na podstawie przeprowadzonych eksperymentów można stwierdzić, że eksperyment nr 5 – wzlotu fizycznego drona z zebraniem danych z obiektu, nie odbiega znacząco od wyników arbitralnych pochodzących z czujnika laserowego. Jako w pełni poprawne można uznać dane zbierane z drona Crazyflie podczas lotu zarówno pod kątem poprawności filtracji danych z czujników na pokładzie obiektu, jak i transferze danych przez komunikacje metasystemu ROS.

Przygotowane środowisko graficzne oparte o program Gazebo i komunikacje ROS, w którym wykonano eksperyment nr 4, sprawdziło się

najlepiej z grupy eksperymentów nieprowadzonych na obiekcie fizycznym. Dzięki zastosowaniu oprogramowania sterownika lotu ArduPilot, oddane zostały poszczególne zachowania obiektu, co przełożyło się na niską wartość całki wartości bezwzględnej błędu. Zachowane zostały również niewielkie opóźnienia w wykonywaniu poleceń widoczne na rysunku 4.12, które odpowiadają opóźnieniom zarejestrowanym na fizycznym dronie.

Eksperyment nr 3 wykonany przy użyciu symulatora CrazyS, okazał się zbliżony do eksperymentów wykonanych przy użyciu modeli matematycznych. Rozpatrując wyniki uzyskane w tym eksperymencie pod kątem analizy kryterium IAE, należy uznać go za najmniej wiernie odzwierciedlający dane arbitralne. Jako zaletę niemierzalną przez kryteria porównawcze, należy uznać zachowanie na początku lotu, gdzie oddany została efekt ziemi.

Eksperymenty wykonane na modelach matematycznych 1 i 3 uzyskały nieco lepsze wyniki w analizie porównawczej do eksperymentu arbitralnego, przy czym cała grupa eksperymentów 1-3 uzyskała znacząco gorszy punktowo wynik od eksperymentu nr 4, w którym wykorzystano oprogramowanie ArduPilot.

4.4. Przyjęta koncepcja środowiska graficznego do symulacji

Na podstawie analizy wyników poczynionej w poprzednim podrozdziale, wyłoniona została potrzeba uwzględnienia oprogramowania sterownika lotu w środowisku graficznym. Wyniki kryterium IAE jednoznacznie dzielą eksperymenty na dwie grupy według jakości odwzorowania zachowań drona rejestrowanych przez czujnik arbitralny. Koncepcja eksperymentu nr 4 jest bardzo bliska wynikom uzyskanym z arbitralnego czujnika laserowego oraz danym zebranym podczas lotu drona. Należy mieć na uwadze, że dzięki uruchomieniu oprogramowania sterownika lotu możliwe jest również testowanie zmian w kodzie źródłowym sterownika lotu w zakresie między innymi komunikacji wzajemnej dronów. Stanowi to treść kolejnego rozdziału, a praca nad poruszonym w nim zagadnieniu komunikacji i możliwość pracy z nim w środowisku graficznym, została umożliwiona dzięki wyborze metody, w której uruchamiane jest oprogramowanie sterownika lotu podczas symulacji.

Rozpatrywany program Gazebo okazał się dobrym wyborem dzięki możliwości symulacji obiektu głównie na podstawie jego modelu CAD i danych o silnikach. Jego atutem jest komunikacja z metasystemem ROS, która umożliwia komunikacje przez zewnętrzne programy różnych języków programowania, pozwalając na rozproszoną wymianę danych, co jest wymagane do implementacji modułu matki-operatora i możliwości pracy w środowisku graficznym.

Przygotowana koncepcja zakłada uruchomienie oprogramowania sterownika lotu ArduPilot przez jego wcześniejszą kompilacje z kodu źródłowego. W celu pracy na stabilnej wersji oprogramowania użyto wersji ArduPilot Copter 4.4.4. Kompilacja kodu źródłowego daje możliwość wdrożenia własnych funkcjonalności. Po skompilowaniu oprogramowania jest ono uruchamiane dla każdego bezzałogowego obiektu latającego. W pliku programu Gazebo zawierającym dane o początku symulacji, zawarte są informacje o adresach sieciowych i portach, przez które można komunikować się z obiektami symulacyjnymi. Adresy i porty muszą być zgodne z zadanymi przy uruchomieniu skomplikowanego oprogramowania ArduPilot na komputerze lokalnym. Dla każdego UAV należy osobno uruchomić oprogramowanie sterownika lotu o unikalnym porcie. Dodatkowo należy również zainicjować komunikacje ROS. Wszystkie te zadania zostały zautomatyzowane przez użycie skryptu systemu Linux napisanego w języku Bash. Moduły uruchamiane przez ROS zostały zawarte w pliku o rozszerzeniu „.launch" uruchamianym z poziomu wymienionego wcześniej skryptu.

Dodatkową korzyścią wynikająca z tego rozwiązania jest możliwość użycia oprogramowania stacji naziemnej. Stacja naziemna to kompleksowy system służący do zdalnej kontroli, monitorowania oraz zarządzania misjami bezzałogowych obiektów. W jej skład wchodzi specjalistyczne oprogramowanie do nadzorowania parametrów lotu, planowania tras oraz analizy danych telemetrycznych. Stacja naziemna pełni kluczową rolę w zapewnianiu bezpiecznego i efektywnego sterowania obiektami bezzałogowymi, umożliwiając przejęcie kontroli operatorowi nad dronem oraz szybkie reagowanie na zmieniające się warunki operacyjne i widok bieżących parametrów drona.

Jednym z najbardziej zaawansowanych i powszechnie stosowanych narzędzi do zarządzania lotem jest QGroundControl (QGC) [88]. QGC to otwartoźródłowe oprogramowanie stacji naziemnej, które oferuje szeroki zakres funkcjonalności, w tym:

- Zaawansowane planowanie misji: Umożliwia tworzenie skomplikowanych tras lotu z wieloma punktami nawigacyjnymi, z uwzględnieniem parametrów takich jak wysokość, prędkość, czy czasy przelotu.
- Monitorowanie parametrów lotu w czasie rzeczywistym: QGC wyświetla dane telemetryczne, takie jak pozycja GPS, wysokość, prędkość, stan baterii, a także umożliwia podgląd obrazu z kamer na pokładzie drona.
- Konfiguracja i diagnostyka: Umożliwia pełne zarządzanie ustawieniami autopilota, w tym kalibrację sensorów, aktualizację oprogramowania i diagnostykę systemu.
- Integracja z protokołem MAVLink: QGC wspiera protokół MAVLink, który jest standardem komunikacyjnym dla wielu popularnych systemów autopilotów, takich jak ArduPilot czy PX4.

Moduł MAVROS umożliwia komunikacje z dronami przez proxy, przygotowanym dla protokołu MAVLink. Proxy MAVLink umożliwia równoczesne korzystanie z ROS i QGC poprzez rozdzielanie strumienia danych z kontrolera lotu do MAVROS i QGC. Dzięki temu w przypadku potrzeby przejęcia kontroli nad danym dronem, operator może za pomocą oprogramowania QGC połączyć się z danym UAV i sterować nim ręcznie. Oprogramowanie może też dostarczyć danych diagnostycznych lub przestrzennych wraz z obrazem z kamer w przejrzystej graficznej postaci.

Rys. 4.14. Widok programu QGC w trybie monitorowania obiektu [źródło własne]

Na rysunku 4.14 przedstawiono widok programu QGC wykorzystanego do monitorowania pojedynczego drona z oprogramowaniem ArduPilot. QGC jest tworzone i rozwijane przez Drone Foundation, a także jest kompatybilne ze sterownikami lotu PX4 i ArduPilot.

Użycie oprogramowania QGC nie jest fundamentalną częścią środowiska graficznego w kontekście roju dronów. Jest niezależną częścią, mogącą posłużyć przy pracy w środowisku graficznym oraz w tej samej konfiguracji przy kontrolowaniu roju fizycznych dronów. Daje bogate możliwości, szczególnie w zakresie sektorów, gdzie wymagane jest monitorowanie lub reagowanie przez operatora na warunki. Użycie tego oprogramowania daje też sposobność szkolenia operatorów do zarządzania rojami dronów, przy użyciu go wraz z przygotowanym środowiskiem graficznym.

Ważnym ograniczeniem z technicznego punktu widzenia jest konieczność angażowania znacznych zasobów obliczeniowych do symulacji wielu obiektów. Program Gazebo może wykorzystywać różne silniki graficzne, co czyni go elastycznym w tym zakresie. Jego kolejnym atutem jest wykorzystanie systemu operacyjnego Linux, którego wymagania są

niewielkie i pozwalają wykonywać zadania bez obciążeń systemowych właściwych dla innych systemów operacyjnych. Silnik graficzny programu Gazebo wykorzystuje głównie zasoby karty graficznej komputera i to ona jest najważniejszym podzespołem potrzebnym do symulacji. Dzięki rozproszonej komunikacji ROS możliwym jest uruchomienie oprogramowań sterowników lotu, a także modułu matki-operatora na innym komputerze. Kolejnym sposobem odciążenia komputera może być również uruchomienie programu Gazebo bez widoku graficznego, zgodnie ze strukturą działania tego programu przedstawioną na rysunku 4.2. Istnieje także możliwość skalowania czasu symulacji względem czasu rzeczywistego, dzięki czemu nawet przy niespełnionych pełnych wymaganiach, symulacja może przebiegać z koniecznymi opóźnieniami.

4.5. Podsumowanie rozdziału

W rozdziale przedstawiono koncepcje prowadzenia badań nad rojem w zakresie prac w programach symulacyjnych. Zgodnie z założeniami poczynionymi w rozdziale 2 niezbędnym było przygotowanie środowiska graficznego, aby dać podłoże do bezpiecznego, wielokrotnego i powtarzalnego testowania koncepcji roju. Dokonano przeglądu najczęściej używanych programów wykorzystywanych przez badaczy dronów. Uwzględniono również możliwości komunikacyjne, aby uczynić przyjęte rozwiązanie modułowym i wymiennym, zgodnie z założeniem przyjętym na początku prac.

Dużą uwagę poświęcono istniejącym już koncepcją symulatorów. Jako odniesienie potraktowano pracę Silano, Aucone, Iannelliego [178] ze względu na możliwość weryfikacji rezultatów tej pracy z fizycznym UAV. Dron Crazyflie z racji na to, że jest modelem powszechnie dostępnym, a także wyprodukowanym w zamyśle uczynienia go kompatybilnym z powszechnymi oprogramowaniami sterowników lotu był najlepszym obiektem do prac walidacyjnych. Po przeanalizowaniu koncepcji pracy badawczej [178], odtworzono ją, dodając eksperymenty na fizycznym obiekcie. Dzięki temu możliwe było porównanie autorskiej koncepcji symulatora z tą zaproponowaną przez badaczy [178]. Aby uniknąć ewentualnej możliwości wpływu przyjętej komunikacji lub błędów

w obliczaniu pozycji przez oprogramowanie sterownika lotu, dodano arbitralne źródło pomiarowe, mierzące wysokość drona bezpośrednio z dużą precyzją. Wyniki przedstawiono w formie graficznej oraz tabelarycznej, posługując się kryterium całki wartości bezwzględnej błędu IAE.

Zrealizowano dwa scenariusze. Pierwszy z nich był tożsamy z badaniami prowadzonymi w porównywanej pracy [178]. Drugi miał na celu sprawdzenie odpowiedzi układów na więcej niż jedno wymuszenie zmiany pozycji. Przeprowadzenie drugiego scenariusza, uwarunkowane było potrzebą sprawdzenia zachowania modeli w innych warunkach. Dzięki temu zminimalizowana zostaje możliwość przypadkowej zbieżności w korzystnych warunkach właściwych tylko jednemu scenariuszowi. Adekwatnie do rozdziału 3, gdzie przy powtórzeniu eksperymentu na modelu matematycznym blok PID Tuner programu Simulink dobrał inne nastawy, właściwe do konkretnego przypadku. Dzięki sprawdzeniu odpowiedzi na więcej niż jedno wymuszenie, przy tych samych nastawach, eliminowana jest potencjalna możliwość uzyskania zbieżnej odpowiedzi na pojedyncze wymuszenie oraz walidowana jest jakość kontroli lotu, nie tylko w zakresie pojedynczego wzlotu. Do analizy zebrano 6 serii danych. Ostania z nich pochodząca z czujnika laserowego została uznana jako arbitralna i wartości IAE zostały obliczone względem tejże próby. Eksperymenty oznaczone numerycznie jako 1, 2, 3 okazały się rozbieżne od wartości arbitralnej. Zostały one wykonane bez uwzględnienia oprogramowania sterownika lotu ArduPilot, który był użyty w fizycznym dronie. Pod względem kryterium IAE najmniej dokładny okazał się eksperyment przeprowadzony przy pomocy symulatora CrazyS. Kolejne, dość zbliżone wyniki uzyskano dla prób przeprowadzonych przy użyciu modeli matematycznych: autorskiego, opracowanego w załączniku nr 1, a także modelu z analizowanego badania [178]. Nieznacznie lepszym w ujęciu porównania przez kryterium IAE okazał się model autorski.

Znacząco lepszą próbą okazał się eksperyment przeprowadzony przy użyciu przygotowanego środowiska graficznego, w którym wykorzystano oprogramowanie ArduPilot do kontroli drona w programie Gazebo. Dane uzyskane z tej próby można uznać za zbliżone do danych odczytanych z lotu, a także zebranych przez

arbitralny czujnik laserowy. Wyniki walidacji metod prowadzenia eksperymentów były zbieżne dla obu scenariuszy. Drugi scenariusz, zobrazowany na rysunku 4.18, dowiódł także, że proponowana koncepcja symulacji w środowisku graficznym, oddaje zachowania obiektu w zakresie minimalnych opóźnień w wykonywaniu zadanych komend.

W celu sprawdzenia wpływu nastaw regulatorów na wyniki przeprowadzonych eksperymentów i wykluczyć ich wpływ jako czynnika różnicującego próby 1-3 od prób 4-6, powtórzono eksperymenty zgodnie z pierwszym scenariuszem, przy nastawach odczytanych z oprogramowania ArduPilot z drona Crazyflie. Ponownie wykonane badanie nie wykazało zmian. Przez to potwierdzona została teza, że czynnikiem różnicującym w jakości prowadzonych eksperymentów jest uwzględnienie oprogramowania sterownika lotu.

Opracowana koncepcja środowiska graficznego uwzględnia oprogramowanie sterownika lotu. Dla każdej jednostki w środowisku graficznym uruchamiana jest niezależna instancja oprogramowania ArduPilot z niezależnym i unikalnym adresem oraz portem. Dzięki kompilacji oprogramowania z kodu źródłowego możliwe jest również wprowadzania własnych zmian w kodzie sterownika. Zgodnie z przedstawionym porównaniem jako najlepszy program do symulacji uznano Gazebo, między innymi dzięki możliwości komunikacji przez metasystem ROS. Dodatkowym atutem tego rozwiązania jest możliwość użycia programu stacji naziemnej QGC w celu np. ręcznego sterowania poszczególną jednostką roju przez operatora. Całe środowisko graficzne, jego uruchomienie i komunikacja są zarządzane pod systemem Linux. Stanowi to nieznaczne ograniczenie, gdyż dzięki integracji dokonanej przez firmę Microsoft, możliwe jest uruchomienie systemu Linux wewnątrz systemu Windows od wersji 10 do nowszych [77]. Korzyścią wynikająca z wykorzystania systemu Linux jest pełna zgodność z wszystkimi poszczególnymi programami, pełna stabilność, a także możliwość automatyzacji uruchamiania przez skrypty powłoki bash. Ograniczenie w postaci zasobów sprzętowych również nie stawia istotnych problemów i może być niwelowane na wiele sposobów omówionych w podrozdziale 4.4. Do zalet przygotowanego środowiska graficznego należy zaliczyć jego

łatwość obsługi, możliwość integracji w wieloma programami i językami programowania, użycie oprogramowania otwartoźródłowego niewymagającego opłat, powszechność poszczególnych programów i bardzo dobre wsparcie ze strony fundacji i środowisk, angażujących się w zagadnienia związane z robotyką mobilną i dronami. **Przygotowane rozwiązanie spełnia wszystkie postawione założenia**, pozwalające utrzymać modułowy koncept pracy. Uwzględnia dynamikę jednostek latających, pozwala implementować własne mechanizmy komunikacyjne tożsame dla oprogramowania jednostek fizycznych, a także dostarcza interfejs do komunikacji z matką-operatorem. Interfejs umożliwia przełączenie pomiędzy graficznym środowiskiem a fizycznymi jednostkami, bez potrzeby zmian w modułu matki-operator.

Rozdział 5

KOMUNIKACJA

W poniższym rozdziale przedstawiono wybrane protokoły do dwóch typów komunikacji mających miejsce w obranej topologii roju. Wybrano protokół najpopularniejszy dla bezzałogowych jednostek oraz rozważono jego możliwości i ograniczenia. Do celów komunikacji wzajemnej, która ma miejsce tylko w grupach dronów, które muszą posiadać inteligentne mechanizmy zachowań, przygotowano autorski protokół. Przedstawiono źródła inspiracji, a także potrzeby i sposób realizacji komunikacji wzajemnej w roju. Dokonano walidacji opracowanego protokołu na fizycznych jednostkach sterujących oraz w opracowanym środowisku graficznym.

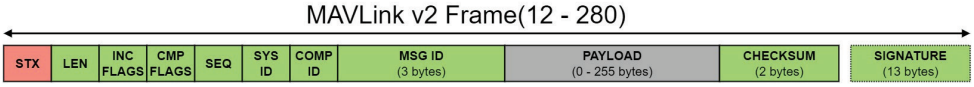
5.1. Protokół komunikacji matka-operator a liderzy grup

Zgodnie z przyjętą topologią komunikacji i założeniami opisanymi w rozdziale nr 2, komunikacja została podzielona na komunikację między matką-operatorem a liderami grup oraz na komunikację między poszczególnymi jednostkami w roju. Jako pierwszy dobrano protokół do komunikacji pomiędzy matką-operatorem a liderami grup. Powszechnie używanym, w kontekście bezzałogowych obiektów latających, jest protokół MAVLink (*Micro Air Vehicle Link*) [82]. Jego obsługa jest domyślnie zaimplementowana w popularnych sterownikach lotu. Został on też obiektem badań naukowców pracujących nad komunikacją obiektów latających [119]. MAVLink to protokół komunikacji, który został zaprojektowany specjalnie do wymiany danych między systemami autopilotów, komputerami naziemnymi oraz innymi elementami systemów bezzałogowych. Jest szeroko stosowany nie tylko w dronach, ale również w innych pojazdach autonomicznych. MAVLink to minimalistyczny protokół komunikacyjny, który przesyła dane w formie małych, uporządkowanych wiadomości. Został zaprojektowany tak, aby być bardzo wydajnym, nawet przy niskiej przepustowości łącza (np. komunikacja radiowa). MAVLink jest obecnie dostępny w dwóch wersjach:

- MAVLink 1.0: wersja szeroko stosowana w istniejących systemach.
- MAVLink 2.0: wersja, która wprowadza m.in. lepsze zabezpieczenia, możliwą rozszerzalność ilości danych i większą liczbę typów wiadomości.

5.1.1. Struktura wiadomości

Wiadomości MAVLink mają ściśle określoną strukturę, która pozwala na łatwą interpretację przez różne urządzenia. Na rysunku 5.1. znajduje się struktura typowej wiadomości MAVLink 2.0.



Rys. 5.1. Struktura ramki MAVLink [83]

Struktura ramki jest oddzielona od warstwy sprzętowej, co oznacza, że jest niezmienna, niezależnie od tego, czy komunikacja następuje z wykorzystaniem przewidzianych w założeniach z rozdziału 3 modułów radiowych lub modułów internetu rzeczy. W zależności od użytej warstwy sprzętowej mogą różnić się parametry komunikacji takie jak: maksymalny dystans między modułami, prędkość i niezawodność przesyłania. Czyni to protokół bardzo uniwersalnym. Gwarantowane jest zachowanie stałej struktury ramki, której poszczególne pola oznaczają:

- **STX – Nagłówek** (1 bajt): Bajt startowy, który oznacza początek wiadomości (0xFE dla MAVLink 1.0, 0xFD dla MAVLink 2.0).
- **LEN – Długość ładunku** (*Payload Length*, 1 bajt): Określa długość danych w wiadomości PAYLOAD. Reszta pól protokołu ma stałą wielkość i nie jest wliczana do tego pola.
- **INC FLAGS – Kompatybilna wersja wiadomości** (*Incompatibility Flag*, 1 bajt, używane tylko w MAVLink 2.0): Wskazuje, czy wiadomość jest kompatybilna z daną wersją MAVLink.
- **CMP FLAGS – Kompatybilna wersja wiadomości** (*Compatibility Flag*, 1 bajt, używane tylko w MAVLink 2.0): Używane do rozpoznania, jakie rozszerzenia są wspierane.

- **SEQ – Sekwencja** (*Sequence*, 1 bajt): Numer sekwencyjny wiadomości, służy do wykrywania zgubionych pakietów.
- **SYS ID – System ID (**1 bajt**):** Identyfikator systemu (np. drona).
- **COMP ID – Component ID** (1 bajt): Identyfikator komponentu w systemie (np. autopilota).
- **MSG ID – Message ID** (1-3 bajty): Unikalny identyfikator typu wiadomości.
- **PAYLOAD – zawartość przekazana w ramce** (0 – 255 bajtów): Zawiera rzeczywiste dane wiadomości. Struktura danych wewnątrz tego pola zależy od typu wiadomości.
- **CHECKSUM – Suma kontrolna** (2 bajty)**:** Suma kontrolna służy do weryfikacji integralności danych. W MAVLink 2.0 dodatkowo używana jest funkcja XOR do poprawy bezpieczeństwa.
- **SIGNATURE – Podpis** (0-13 bajtów, opcjonalne pole, dostępne tylko w MAVLink 2.0**):** Dodatkowy mechanizm zabezpieczeń, którego implementacja może być dobrana dla danego rozwiązania. Najczęściej zabezpiecza wiadomość przed fałszowaniem poprzez dodanie podpisu kryptograficznego.

5.1.1.1. Rodzaje wiadomości zawartych w buforze danych (*payload)*

MAVLink posiada zdefiniowane setki typów wiadomości [82], które są wykorzystywane do różnych zadań. Przykładowe typy wiadomości to:

- **HEARTBEAT:** Wiadomość wysyłana okresowo przez system, sygnalizująca, że jest on aktywny i gotowy do pracy.
- **ATTITUDE:** Informacje o położeniu i orientacji jednostki.
- **GPS_RAW_INT:** Surowe dane z GPS.
- **COMMAND_LONG:** Wiadomość służąca do przesyłania poleceń do wykonania przez odbiornik, np. zmiana trybu lotu. Ta „długa” komenda dzielona jest na wiele innych, każda o unikatowym binarnym ID komendy. Istnieje możliwość dodania własnych komend na tym etapie. Warto jednocześnie zwrócić uwagę, że wysyłanie komend tego typu najbardziej obciąża łącze transmisji danych przez użycie największej ilości pól.

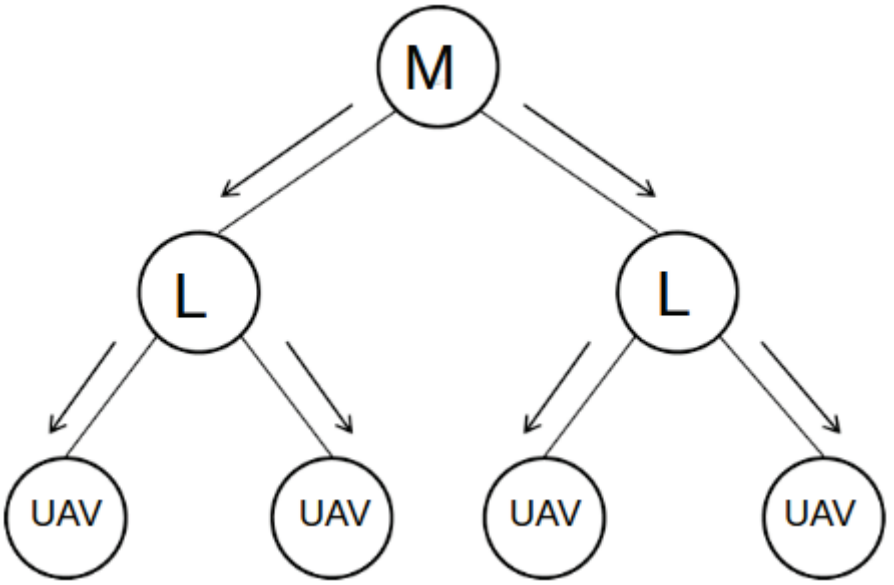
5.1.2. Integracja protokołu

MAVLink jest łatwy do integracji z różnymi platformami dzięki licencji otwartego oprogramowania i wsparciu dla wielu języków programowania, takich jak między innymi: C, C++, Python, Matlab. Po stronie sterowników lotu, w tym rozważanych – PX4 i ArduPilot, ale także innych, wsparcie dla tego protokołu jest domyślnie zaimplementowane. Fizyczna jednostka obliczeniowa, realizująca algorytm sterowania w roju, mogąca być jednostką latającą lub jednostką stacjonarną obsługiwaną przez operatora, również może zostać przystosowana do pracy z protokołem MAVLink. Zarówno komunikacja z systemem ROS, ale też pakietami dla języka Python, wykorzystuje MAVLink. Dzięki temu można korzystać z wielu wyższych warstw oprogramowania, które między sobą utrzymają synchronizację komunikacji w jednym standardzie. Do tego ten protokół może być debugowany, czyli uruchamiany w trybie wykrywania błędów, dzięki oprogramowaniu Wireshark. Wireshark jest narzędziem służącym do śledzenia i podsłuchiwania komunikacji dla celów diagnostycznych. Powszechność i otwartość standardu MAVLink sprawia, że jest najlepszym wyborem dla elastycznego utrzymania oprogramowania, które może wykorzystywać różnorodne pakiety technologiczne i być rozwijane w zależności od potrzeb.

5.1.3. Bezpieczeństwo i potwierdzenie danych

Wersja MAVLink 2.0 wprowadza dodatkowe funkcje bezpieczeństwa, dodane jako opcjonalne na końcu ramki. Podpisywanie wiadomości umożliwia zabezpieczenie jej przed fałszerstwem, co jest szczególnie ważne w zastosowaniach, gdzie niezbędne jest bezpieczeństwo. Nie jest to ochroną przed przechwyceniem komunikacji, ale przed przyjęciem wiadomości z nieautoryzowanego źródła. Kolejnym zabezpieczeniem wewnątrz protokołu jest zwiększona suma kontrolna. Zmniejsza ryzyko błędów w transmisji przez walidacje przysłanych danych. Dodatkowo w przypadku komunikacji za pośrednictwem łączy globalnie dostępnych, czyli będących częścią większej sieci, gdzie dane przepływają z różnych źródeł, istnieją dodatkowe możliwości zabezpieczenia tego protokołu. W przypadku rozpatrywanej komunikacji z wykorzystaniem modemów sieci internetowej

wąskiego pasma, wszystkie dane stają się częścią globalnej sieci internetowej. Aby je zabezpieczyć, wymagana jest enkapsulacja protokołu wewnątrz innej szyfrowanej informacji. Pełni to rolę obudowy dla całej ramki protokołu, w której zawiera się informacja, gdzie ma trafić dana ramka MAVLink. Podczas transmisji danych, zawartość jest szyfrowana przed przechwyceniem. Protokół nie przewiduje komunikacji z potwierdzeniem ACK (*acknowledge*, z ang. potwierdzać). Dane wysyłane są bez otrzymania informacji zwrotnej o odebraniu. Rozwiązaniem przyjętym w komunikacji z użyciem tego protokołu dla pojedynczych dronów jest wysyłanie sygnału *hearbeat* (z ang. uderzenie serca, co należy rozumieć jako puls, który informuje o obecności). Rozważając strukturę przyjętej topologii przedstawioną poniżej, należy zwrócić uwagę, że matka roju nie komunikuje się bezpośrednio z każdą z jednostek co zobrazowano na rysunku 5.2.



Rys. 5.2. Topologia komunikacji w przyjętym modelu roju [źródło własne]

Na rysunku 5.2 przedstawiono przyjętą topologię komunikacji, gdzie:
M – matka-operator roju,
L – liderzy roju,
UAV – podrzędne jednostki roju.

Komunikacja pomiędzy liderami a podrzędnymi jednostkami jest treścią podrozdziału 5.2. W kontekście komunikacji matka-operator a liderzy należy mieć na uwadze, że matka-operator musi mieć możliwość wysyłania informacji do poszczególnych jednostek. W tym celu wysyła komendę do lidera danej podgrupy, aby ten przedłożył informacje do wyznaczonej jednostki. Na rysunku obrazuje to kierunek strzałek. Matka roju wysyła informacje pośrednio do każdej jednostki. Istnieją różne możliwości weryfikacji takiego scenariusza. Poniżej przedstawiono 3 z nich.

Pierwszą z nich jest weryfikacja po efektach. Matka roju zadaje komendę każdej jednostce i po czasie sprawdza efekty wykonania tych komend. Niesie to za sobą możliwe opóźnienia w realizacji zadań. Gdy dane nie dotrą do lidera, ten nie rozgłosi ich dalej, a cała grupa nie wykona powierzonego zadania i matka będzie musiała ponowić wezwanie do wykonania zadania.

Drugim sposobem implementacji, jaki został przewidziany między innymi w pakiecie ROS, jest wysyłanie danych komend przez matkę ciągle, w zadanym interwale. Dzięki temu każdy lider otrzymuje informacje o wykonywanym zadaniu. Podczas zadań o dłuższym czasie wykonywania, każdy lider dostaje ponowienia zadania wcześniej rozpoczętego, co z jednej strony obciąża jego łącze komunikacji, gdyż musi każdą daną odbierać i odrzucać dane powtarzane do innych liderów. Z drugiej jednak strony dostarcza informacje, że jest w kontakcie z matką-operatorem. Po stronie matki istnieje konieczność wysyłania dużej ilości danych przez cały czas. W przypadku procesorów wielordzeniowych może to być dobrze zrealizowane, poprzez przeznaczenie mocy obliczeniowej jednego rdzenia do zarządzania wysyłaniem danych, przez co ponawiająca się komunikacja nie ma wpływu na inne procesy.

Trzecim rozwiązaniem, które jest zastępstwem braku potwierdzenia odebrania danych ACK w protokole, jest wysyłanie własnej komendy ACK wbudowanej wewnątrz „COMMAND_LONG”. Należy samodzielnie dopisać to rozwiązanie do listy komend i wgrać aktualizacje do każdej jednostki lidera oraz matki. Jest to dłuższe rozwiązanie niż standardowa implementacja ACK w innych protokołach. Poza tym należy jeszcze ustalić,

czy matka odpowiada każdemu liderowi, czy dostała ACK i jak długo lider próbuje wysyłać ACK do matki.

W pracy przyjęto koncepcję potwierdzeń opierającą się na drugim modelu polegającym na cyklicznym wysyłaniu informacji o realizowanym zadaniu przez matkę-operatora dla każdego lidera. Jest to rozwiązanie kompatybilne z pakietem ROS. Wiąże się to z większym obciążeniem łącza i jednostki matki. Należy jednak mieć na uwadze, że jednostka matki jest jedna dla całego roju, który może liczyć wiele obiektów i koszt mocniejszej jednostki matki, którego rolę co do założenia pełni komputer personalny obsługiwany przez operatora, jest pomijalnie mały. W tym rozwiązaniu liderzy są informowani przez cykliczne komunikaty od matki o trwającym połączeniu. Z kolei matka otrzymuje sygnał *heartbeat* od każdego lidera, dzięki czemu wie, czy ma komunikacje z każdą grupą. Zadania dedykowane dla poszczególnej jednostki wysyłane są z żądaniem wysyłania potwierdzenia.

Należy mieć na uwadze, że komendy wysyłane przez matkę do poszczególnej jednostki danej grupy, specyficzne dla jednostki i trudne do weryfikacji, jak np. wyzerowanie czujnika na pokładzie danej jednostki, należy wysłać tylko raz, ponieważ każda kolejna iteracja tej komendy zakłóciła by pracę roju. Z kolei pojedyncze wysłanie takiej wiadomości bez wymaganego potwierdzenia, może prowadzić do niewykonania zadania. Niepożądane efekty mogą być bardzo trudne do zweryfikowania.

5.1.4. Ograniczenia protokołu MAVLink

Do ograniczeń protokołu MAVLink należą:

- W przypadku komunikacji z użyciem modemu protokół wymaga modułu z procesorem do obsługi komunikacji. W przypadku komunikacji radiowej obciąża zasoby obliczeniowe sterownika lotu jednostki latającej.
- Protokół poprzez swoją uniwersalność zawiera wiele dodatkowych pól, które są stałe dla standardu i w przypadku przekazywania wielu danych, obciąża łącze transmisji danych.
- Pomimo niewielkich rozmiarów struktur danych protokołu jego implementacja jest złożona i niekiedy wymaga własnych mechanizmów potwierdzania danych.

- Dodanie dodatkowych komend wymaga zmian po stronie zarówno oprogramowania liderów, jak i algorytmu matki. Może wpływać na proces algorytmiki, jeśli ten oparty byłby o pakiety, które utrudniałyby dodanie potwierdzeń danych komend. W tym przypadku rośnie złożoność zadań na poziomie utrzymania kodu podczas rozwoju oprogramowania.

5.2. Protokół komunikacji wewnątrz grupy

Na potrzeby komunikacji wzajemnej, przy wykorzystaniu radia UWB opracowany został autorski protokół komunikacji oparty o technologie radiowego przesyłania danych między poszczególnymi jednostkami roju. Opracowanie autorskiego rozwiązania podyktowane było brakiem zadowalającego protokołu do rozpatrywanego celu. Innowacyjnym podejściem było wykorzystanie technologii UWB, zarówno do przesyłania danych, jak i również ustalaniu wzajemnej pozycji między poszczególnymi bezzałogowymi obiektami latającymi.

Przyjęto następujące cele i założenia dla komunikacji wzajemnej:

1. Wszystkie jednostki komunikują się wzajemnie w celu weryfikacji dystansu wzajemnego, aby uniknąć kolizji.
2. Wszystkie jednostki otrzymują dane z sensorów, które znajdują się na pokładzie pozostałych obiektów. Każda jednostka ma ten sam dostęp do danych, dzięki czemu może syntetyzować je z własnymi danymi.
3. Lider podgrupy może żądać informacji lub akcji od każdej jednostki w grupie, przez co może wysyłać żądania do każdej jednostki.

Konieczność użycia radia UWB w celu określenia wzajemnego dystansu między modułami radiowymi jest atutem komunikacji wzajemnej z racji na możliwości poprawy bezpieczeństwa roju w zakresie unikania kolizji wzajemnych wewnątrz grupy. Ograniczeniem jest konieczność cyklicznego wysyłania danych przez każdą jednostkę w celu ostrzeżenia innych jednostek o swojej pozycji. Aby nie obciążać wszystkich jednostek dodaniem pojedynczego obiektu do grupy, koniecznym było skoncentrowanie się tylko na rozwiązaniach, które umożliwiają komunikację jednostronną bez potrzeby potwierdzenia odebrania danych. Dzięki temu każda jednostka mogłaby wybierać wiadomości tylko od obiektów, od których potrzebuje informacji. Konsekwencją tego typu rozwiązania jest zredukowanie przyrostu

przetwarzania danych z komunikacji wzajemnej przy dodaniu kolejnych jednostek do grupy. Z drugiej strony w przypadku komunikacji lidera wymagane jest, aby komunikacja zachodziła dwustronnie w potwierdzeniem przesyłania danych. Wymaga to elastyczności od protokołu, aby móc dobierać model ramki protokołu w zależności od potrzebnego zastosowania. Również należało uwzględnić, że pomimo elastyczności protokół musi być lekki, czyli nie obciążający możliwości obliczeniowych sterownika lotu. Oznacza to, że wymagane jest, aby ramki protokołu były krótkie, bez wielokrotnego zagnieżdżenia, które wymagałoby większego obciążenia radia, jak i również dłuższego odbioru niezbędnych informacji z ramki protokołu podczas nasłuchiwania danych.

Jako, że jest to protokół autorski, a jego innowacja polega na minimalistycznym wypełnieniu potrzeb komunikacji wewnątrz grupy, mechanizmy komunikacji zostały wbudowane w obsługę protokołu. Opis topologii komunikacji zawiera również algorytmikę i mechanizmy selekcji jednostek będących liderami. Są one nierozłączne z protokołem, dzięki temu stanowią bezpieczną i szybką warstwę transportu informacji, lokalizacji oraz poleceń wymaganych do pracy całego zespołu obiektów. Mechanizmy komunikacji wbudowane w protokół zostały opisane wraz ze strukturą danych protokołu w tym podrozdziale. Metodyka synchronizacji pozycji i wyznaczania pozycji bazowej grupy jest wyższą warstwą oprogramowania, przez co należy rozumieć, że jest nadbudową protokołu. Dzięki temu może zostać modyfikowana bez zmian protokołu oraz podstawowych mechanizmów komunikacji.

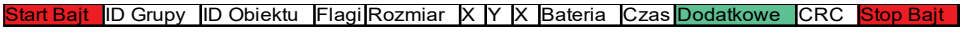
Podczas prac nad protokołem rozważano różne metody komunikacji między obiektami. Z racji na potrzebę zastosowania rozwiązania, które umożliwiłoby precyzyjny pomiar odległości między jednostkami, szybkość komunikacji, bezpieczeństwo sygnałów w warunkach pracy roju dronów, wybrano moduły UWB. Są one popularne i łatwe do implementacji w popularnych sterownikach lotu. Inspiracją dla mechanizmów protokołu jest komunikacje BLE (*Bluetooth Low Energy*) [69]. Jest to standard najwyższej jakości, nad którym pracowały największe obecnie firmy zajmujące się komunikacją bezprzewodową. Z racji na przedstawiony problem dystansów pomiędzy obiektami w powietrzu, standard BLE nie mógł zostać

zaimplementowany w poniższej pracy. Wzorowano się jednak na niektórych mechanizmach tego protokołu, takich jak – metoda potwierdzenia odebrania danych oraz rozgłaszanie informacji do wszystkich bez potrzeby uzyskania informacji zwrotnej. W wielu miejscach ograniczono elastyczność protokołu w zamian za oszczędzenie niezbędnej pamięci do obsługi komunikacji oraz ujednolicenie mechanizmów komunikacji, czego podsumowania dokonano na końcu podrozdziału.

5.2.1. Format struktury ramki

Struktura ramki protokołu została zaprojektowana w taki sposób, aby zapewnić niezawodną wymianę danych oraz uwzględniać niezbędne informacje, takie jak pozycja, stan baterii i dodatkowe dane. Ramka jest podzielona na kilka pól, z których każde ma określone przeznaczenie. Tak jak większość protokołów, również i ten jest autoryzowany po odbiorze przez obliczanie sumy kontrolnej. Aby nie obciążać sterowników lotu, zaimplementowano rozwiązania zapewniające minimalne obciążenie warstwy fizycznej przesyłu, ograniczając ilość wysyłanych bajtów do minimum. Kosztem tego rozwiązania jest rezygnacja z zabezpieczeń bezpieczeństwa danych przed przechwyceniem lub próbą sfałszowania danych. Jest to autorski protokół przygotowany na potrzeby zadań cywilnych, gdzie potrzeba szyfrowania nie jest wymagana do działania całej grupy obiektów. Dane przesyłane w tym protokole mają też ograniczony zasięg, stąd przechwycenie ich przez zewnętrzne radio, mogłoby być trudne w realizacji, ze względu na przemieszczanie się grupy.

Format ramki protokołu jest zorganizowany w następujący sposób:



Rys 5.3. Ramka autorskiego protokołu do komunikacji wewnątrz roju [źródło własne]

Gdzie:

- **Bajt startowy (1 bajt):** unikalne pole informujące o rozpoczęciu transmisji ramki protokołu o stałej wartości 0xFF,

- **ID Grupy (1 bajt):** zawiera informacje o numerze grupy inaczej zwanej podrojem obiektu, który wysłał ramkę lub do którego ramka jest skierowana,
- **ID Obiektu (1 bajt):** numer identyfikujący obiekt, który wysłał ramkę lub do którego ramka jest skierowana,
- **Flagi bitowe (2 bajty)**
    - Bit 0: kierunek transmisji (0: wysłana z obiektu o zadanym ID, 1: wysłana do obiektu o zadanym ID),
    - Bit 1: status alarmu (0: brak alarmu, 1: alarm wyzwolony),
    - Bit 2: stan baterii (0: normalny, 1: krytyczny),
    - Bit 3: GPS (0: brak odbiornika na pokładzie jednostki, 1: odbiornik na pokładzie),
    - Bit 4: stan GPS – uwzględniane tylko jeśli poprzedni bit zawierał wartość 1 (0: odbiornik GPS niedziałający, 1: odbiornik GPS działający),
    - Bit 5: znacznik czasu (*timestamp*), (0: brak znacznika czasu w ramce, 1: znacznik czasu obecny w ramce na jej końcu przed informacjami dodatkowymi i sumą kontrolną),
    - Bit 6: wymagane potwierdzenie odebrania danych (ACK, skrót od „*acknowledgement*"), (0: ACK niewymagane, 1: ACK wymagane). W przypadku adresowania ramki do konkretnego odbiorcy – nadająca jednostka,
    - Bit 7: odpowiedź na wymagane potwierdzenie danych (0: brak potwierdzenia, 1: potwierdzenie odebrania ramki wymagającej potwierdzenia, lub potwierdzenie otrzymania potwierdzenia przez pierwotnego nadawcę ramki wymagającej potwierdzenia),
    - Bit 8: oznaczenie lidera (0: jednostka wysyłająca inna niż lider grupy, 1: jednostka wysyłająca jest liderem grupy),
    - Bity 9-10: dodatkowe dane dotyczące trajektorii loty grupy, możliwe tylko do wysłania przez lidera,
    - Bity 11-14: struktura grupy, możliwa do wysłania tylko przez lidera grupy,
    - Bit 15: zarezerwowany.

- **Rozmiar danych** (1 bajt): rozmiar danych, jakie zostaną jeszcze wysłane w danej ramce protokołu podany w bajtach,
- **Współrzędna X (4 bajty):** pozycja X obiektu w lokalnym układzie lidera. Jeśli jest to jednostka zawierająca odbiornik GPS, wysłane dane dotyczące pozycji liniowej będą odczytane z odbiornika GPS, uwzględniając współczynniki korekcji zapisane do pamięci nieulotnej sterownika lotu,
- **Współrzędna Y (4 bajty):** pozycja Y obiektu, o analogicznym znaczeniu jak pozycja X,
- **Współrzędna Z (4 bajty):** pozycja Z obiektu, o analogicznym znaczeniu jak pozycja X,
- **Stan baterii (1 bajt):** procentowy stan naładowania baterii (1-100%),
- **Znacznik czasu (4 bajty, wysłanie opcjonalne):** znacznik wysyłany w przypadku zadeklarowania wysyłania go we fladze bitowej. Domyślnym jest przesyłanie znacznika czasu przez obiekty posiadające odbiornik GPS,
- **Informacje dodatkowe (N bajtów, wysłanie opcjonalne):** pole w którym mogą zawierać się dane właściwe danym jednostką, np. odczyty z czujników,
- **CRC (2 bajty):** suma kontrolna służąca do walidacji poprawności ramki,
- **Bajt Stopu (1 bajt) –** informuje o zakończeniu ramki. Jest to pole o stałej wartości 0x00.

Jako współrzędne X, Y, Z poszczególne jednostki w grupie, za wyjątkiem lidera, wysyłają swoją aktualną pozycje. Lider, którego pozycja zawsze jest tożsama z początkiem lokalnego układu współrzędnych grupy, przesyła jako współrzędne pozycję, do której dąży on sam. Oznacza to, że jeśli lider przesyła koordynaty X=0, Y=0, Z=0, to osiągnął pozycje określoną dla ostatniego zadania. Dla grupy wielowirnikowców możliwe jest, aby lider zatrzymał się w miejscu. Pozostałe jednostki zobowiązane są również do zatrzymania, jednakże zachowując szyk grupy. Przykład ten pokazuje strukturę autonomii, gdzie najważniejsza jest autonomia jednostki, która szczególną uwagę zwraca na to, aby chronić lidera. Następnie rozpatrywana jest autonomia grupy, kierowanej przez lidera. Protokół został

zaprojektowany tak, aby na poziomie warstwy aplikacji umożliwiać realizacje rozważanych autonomii.

Aby zapewnić nieobciążający dla łącza radiowego i możliwości obliczeniowych standard komunikacji wszystkim polom poza „informacje dodatkowe", nadano stały rozmiar. Jedynie informacje dodatkowe mają zmienny rozmiar, który odbierająca jednostka może przewidzieć, odczytując flagi bitowe oraz rozmiar danych. Mechanizm ten enkapsuluje wewnątrz protokołu możliwość implementacji bardziej zaawansowanych danych diagnostycznych oraz komend przygotowanych dla zadanego zadania.

Z założenia nie jest wymagane potwierdzenie każdej ramki. Większość ramek przeznaczona jest do publikowania jedynie pozycji jednostki, która rozgłasza(wysyła) ramkę z zadanym interwałem czasowym. Jedynie lider grupy może wysyłać ramki adresowane do poszczególnych uczestników. Każdy z uczestników wysyła swoją ramkę adresowaną do wszystkich, dzięki czemu możliwe jest, aby inne obiekty wyznaczały swoją pozycje, a także w razie możliwości zderzenia korygowały swoją pozycje, nie dopuszczając do kolizji. Warto zaznaczyć, że poza danymi przekazanymi wprost w protokole, obliczany jest jeszcze na poziomie sprzętowym dystans między nadawcą a odbierającym, przez moduł UWB. Ta wartość zostaje zapisana wraz z ramką, dzięki czemu możliwe jest zidentyfikowanie jednostki, która znajduje się w obliczonym dystansie oraz rozpoznanie, na podstawie przesłanych danych pozycji liniowej, z którego kierunku przyszedł sygnał i jaką korektę lotu wykonać w przypadku prawdopodobieństwa kolizji.

W formacie protokołu „flagi bitowe" są elementem o możliwym do zmodyfikowania rozmiarze. Należy jednak przez to rozumieć modyfikacje statyczną, czyli w przypadku konieczności rozszerzenia funkcjonalności o dodatkowe flagi, należy wgrać do wszystkich jednostek informacje o nowym formacie ramki. Wymusza to wgrywanie korekt do oprogramowania każdej poszczególnej jednostki, jednak zapewnia bezpieczeństwo struktury protokołu, przy jednoczesnym priorytecie pozostawienia go możliwie minimalistycznym w implementacji. Łatwość w zmianie protokołu czyni go elastycznym i możliwym do przyszłych modyfikacji.

Wewnątrz pola zawierającego „informacje dodatkowe”, przyjęty został następujący schemat protokołu:

- **Bajt informacyjny** – zawierający dwie informacje:
  - bity 0-3 – zawierają numer polecenia, jakie zostanie przekazane w części właściwej,
  - bity 4-7 – zawierają ilość bajtów, jaką zajmie polecenie w części właściwej,
- **Część właściwa** – zawiera parametry dla komendy, które sterownik lotu może zidentyfikować dzięki numerowi komendy przekątnej w poprzednim bajcie. Ilość bajtów jaką zajmie ta cześć, jest zdefiniowana w drugiej części bajtu informacyjnego. Aby dokonać poprawnego przydzielenia parametrów, sterownik odczytuje typy argumentów dla danej funkcji reprezentującej komendę i zgodnie z tym przydziela odpowiednie typy i ilość parametrów.

Ilość komend wysłanych jednorazowo nie jest możliwa do obliczenia po odebraniu bajtu, określającego ile danych ma zostać przekazanych w ramce. Jest to spowodowane możliwymi rozbieżnościami w parametrach przekazywanych w części właściwej. W strukturze języka C++, reprezentującej pełną otrzymaną ramkę danych, informacje dodatkowe są zawarte wewnątrz alokowanej tablicy. Aby nie powodować obciążeń systemu spowodowanych dynamiczną alokacją miejsca w pamięci, podczas odbierania ramki zaimplementowano następującą koncepcję. Tablica jest alokowana do jej maksymalnego rozmiaru wynoszącego 104 bajty danych. Po odebraniu pełnej ramki i przejściu wszystkich procesów walidacji, gdy ramka zostanie zakwalifikowana jako poprawna, weryfikowana jest zajętość pamięci. Następuje realokacja pamięci do minimalnej wymaganej ilości. W tym samym procesie alokowana jest pamięć na kolejną ramkę protokołu, również z założeniem, że sekcja protokołu zawierająca informacje dodatkowe może zająć maksymalną ilość pamięci. Po rozpatrzeniu pełnej ramki komunikacji i wszystkich zawartych w niej komend, specjalnych ramka jest kasowana z buforu kołowego, na zasadzie dynamicznej realokacji pamięci. Następuje zwolnienie wskaźnika do części zawierającej informacje specjalne.

Umownie dokonany został podział na przekazywane komendy oraz informacje. Jako komendy należy rozumieć informacje jaka procedura, która musi być zaimplementowana w sterowniku, ma zostać wykonana z żądanymi parametrami. Jako informacje definiowane są wartości przekazywane przez obiekt np. odczyty z czujników właściwego dla danego urządzenia. Komendy są inicjowane najczęściej przez lidera. W większości przypadków komendy są adresowane do danej jednostki i wymagają potwierdzenia ACK. Tylko komendy traktowane jako komendy najniższego priorytetu nie wymagają ACK. Priorytet komend jest parametrem wgrywanym do sterownika lotu podczas kompilacji i nie może zostać zmieniony podczas pracy obiektów. Jedynym przypadkiem kiedy komenda nie jest adresowana do danej jednostki, jest adresacja komendy do wszystkich jednostek w grupie, wysłana przez lidera. Jest to mechanizm przewidziany do sterowania grupą w przypadku potrzeby szybkiego reagowania, gdzie czas komunikacji nie pozwala na adresacje komend do poszczególnych jednostek i na oczekiwanie potwierdzenia od każdej z nich, przed wysłaniem tej samej komendy do kolejnej jednostki. Po komendzie wysłanej do wszystkich jednostek zwracane są potwierdzenia ACK do lidera. Jednak następuje to w formie asynchronicznej, ponieważ każda jednostka z właściwym dla siebie czasem reakcji wysyła potwierdzenie ACK liderowi.

Komenda adresowana do wszystkich jednostek musi zawierać następującą kombinacje:

- ID jednostki musi wskazywać na lidera,
- w polach bitowych kierunek transmisji wskazuje na wiadomość wysłaną od jednostki,
- w bicie identyfikacji lidera znajduje się logiczna jedynka.

Aby przechowywać dane zbierane i ustawiane podczas pracy grupy, do obsługi protokołu została zaimplementowana struktura pomocnicza. Zawiera pola uniwersalne dla wszystkich jednostek oraz pola właściwe dla szczególnych jednostek. Jako szczególne jednostki należy rozumieć jednostki zawierające dodatkowy osprzęt np. odbiornik GPS. W przypadku potrzeby poszerzenia możliwych zadań roju wraz z dobraniem ukierunkowanego osprzętu dla bezzałogowych obiektów latających, procedura implementacji

opiera się na dopisaniu do struktury nowych parametrów, a także dopisaniu komend lub informacji dla użyteczności osprzętu jednostek przez grupę.

Struktura domyślna obsługi protokołu zawiera następujące pola:

- ID własne obiektu,
- ID grupy w której się znajduje,
- ID lidera,
- statyczny bufor kołowy, dla komend i informacji odczytanych podczas analizowania ramek,
- tablicę z dynamicznie aktualizowanymi wartościami dystansu do najbliższych sąsiadujących jednostek,
- punkt docelowy do zrealizowania dla aktualnie wykonywanego zadania,
- aktualną strukturę szyku grupy,
- częstotliwość z jaką obiekt ma rozgłaszać, czyli wysyłać bez oczekiwania potwierdzenia, swoje dane,
- strukturę danych w postaci mapy przechowującej informacje o wszystkich jednostkach w grupie. Struktura zawiera pary: ID obiekt oraz typ obiektu. Dzięki temu każdy obiekt wie, który obiekt ma specjalne uprawnienia i możliwe jest zadanie, aby obiekty podrzędne, które traktują, na poziomie autonomii własnej, bezpieczeństwo lidera ponad swoje, traktowały tak również inne obiekty specjalne, lub specjalnie reagowały na dane od tych obiektów.

Pola właściwe jednostkom specjalnym zawierającym GPS na pokładzie:

- współczynniki korekcji dla GPS.

Jak zostało wcześniej wspomniane protokół zakłada wyróżnienie lidera grupy. Mechanizm wyboru lidera uzależniony może być od wielu warunków. Domyślnie wybierany jest obiekt, spośród jednostek których osprzęt pozwala na komunikacje z matką-operatorem i jest zlokalizowany najbardziej centralnie względem wszystkich dronów w grupie. Należy przyjąć hierarchię, według której lider jest jednostką, która komunikuje się ze wszystkimi obiektami w grupie oraz może żądać od nich konkretnej akcji. Oprócz tego lider komunikuje się z matką roju. Komunikacja z matką jest realizowana przez protokół MAVLink.

Do zadań lidera wewnątrz grupy należy:

- zainicjowanie komunikacji, zgłoszenie się wszystkim jednostkom jako lider, nadanie numeru identyfikacyjnego grupy,
- zidentyfikowanie obiektów z odbiornikami GPS, zainicjowanie synchronizacji odbiorników GPS i obsłużenie korekcji swojego odbiornika,
- zapoczątkowanie rozgłaszania pozycji przez grupę obiektów posiadających odbiorniki GPS,
- zbieranie informacji o pozycjach poszczególnych jednostek, ich stanie baterii oraz o możliwych awariach,
- sterowanie prędkości przelotu całej grupy,
- odbieranie danych zawierających trajektorie oraz formacje grupy, a także rozgłaszanie cykliczne tych informacji wszystkim jednostkom w grupie,
- czuwanie nad własnym bezpieczeństwem, poprzez odbieranie danych o dystansie do sąsiadujących dronów i tylko w skrajnych przypadkach korygowanie swojego położenia.

Do zadań pozostałych obiektów należy:

- odbieranie komend od lidera, wykonywanie ich i jeśli potrzeba, to wysyłanie potwierdzeń liderowi,
- odbieranie zadanej trajektorii i kształtu roju, a także utrzymanie szyku,
- obliczanie własnej pozycji w lokalnym dla grupy układzie współrzędnych. Jeśli jednostka zawiera moduł GPS, odczytuje dane z tego odbiornika, po uprzedniej jednorazowej korekcji od lidera. Jeśli jednostka nie zawiera odbiornika GPS, jej zadaniem jest obliczenie swojej pozycji, na podstawie położenia wysyłanego przez obiekty zawierające odbiorniki GPS,
- rozgłaszanie swojej pozycji poprzez wysyłanie cyklicznych wiadomości adresowanych do wszystkich jednostek w grupie,
- czuwanie nad własnym bezpieczeństwem, poprzez odbieranie danych o dystansie do sąsiadujących dronów i w razie potrzeby korygowanie swojego położenia, ze szczególnym uwzględnieniem bezpieczeństwa lidera.

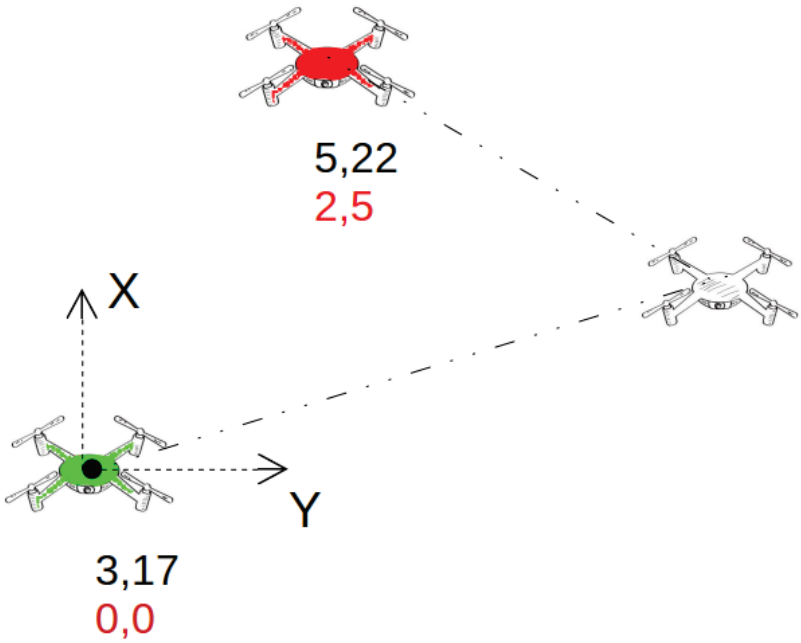
Przebieg inicjalizacji wymaganej do synchronizacji grupy jest następujący:

1. Stanem wyjściowym jest cisza komunikacyjna w grupie, każdy obiekt jedynie nasłuchuje.
2. Lider zgłasza się wszystkim jednostkom, identyfikując się jako lider. Każda z nich ma potwierdzić odebranie tego zgłoszenia przez wysłanie ACK do lidera. Jeśli jakaś jednostka nie otrzyma zgłoszenia lidera, ustala lidera nasłuchując pozostałych, zsyłanych potwierdzeń adresowanych do lidera przez inne obiekty. To jedyny przypadek kiedy mechanizm walidacji pozwala na wyłapanie ramki nieadresowanej do danego urządzenia poprzez pominięcie trzeciego punktu walidacji. Mechanizmy walidacji poruszone zostały w podrozdziale 5.2.3.
3. Po zakończeniu zbierania potwierdzeń od wszystkich jednostek lider identyfikuje jednostki z działającymi odbiornikami GPS, zapisując ich ID do swojej pamięci. Następnie rozpoczyna synchronizacje z każdym z obiektów z odbiornikiem GPS. Jeśli potrzeba, dokonuje korekty swojego odbiornika.
4. Po zakończeniu synchronizacji i nadaniu odpowiednich współczynników korekcji wybranym jednostkom, lider rozpoczyna nadawanie swoich ramek.

Jak zostało przedstawione w strukturze protokołu, ramka lidera jest różna od pozostałych, ponieważ nie zawiera pozycji lidera. Ta traktowana jest zawsze jako początek lokalnego dla grupy układu współrzędnych. W zamian lider zawiera punkt w przestrzeni, zlokalizowany w lokalnym dla grupy układzie współrzędnych, do którego dąży lider, a także kształt formacji, jaką mają przyjąć pozostałe obiekty. Lider nie decyduje o formacji grupy. Początkowo jest ona definiowana w pliku zawierającym parametry misji. W trakcie trwania misji może zostać zmodyfikowana przez matkę-operatora. Możliwe formacje zostały opisane w rozdziale 6.

Droga, którą ma przebyć lider do punktu docelowego, jest tożsama z przesunięciem całego lokalnego układu współrzędnych i każdy z obiektów po skorygowaniu szyku, gdy nie występują niepożądane zdarzenia, podąża tą samą trajektorią co lider. Wraz z liderem swoje ramki zaczynają nadawać

jednostki z odbiornikiem GPS. Poszczególne jednostki nie zawierające odbiornika GPS ustalają swoje pozycje na podstawie danych od jednostek z odbiornikami GPS. Ten mechanizm został opisany w kolejnym podrozdziale 5.3 poświęconym autonomii grupy podroju. Po ustaleniu swoich pozycji wszystkie jednostki formują, utrzymują szyk i podążają wyznaczoną trajektorią. Każda jednostka ustala swoją pozycje liniową. Należy zaznaczyć, że pozycja kątowa poszczególnych jednostek nie jest istotna z racji na to, że każdy obiekt może wykonywać ruchy według własnej autonomii, aby chronić się przed kolizjami. Układ współrzędnych grupy, który ma swój punkt w układzie współrzędnym lidera, nie jest obracany według żadnej z osi, również przy manewrach lidera. Układ lokalny grupy jest zwrócony osią Z w dół, czyli wychodzącą prostopadle do Ziemi. Orientacja wokół osi Z jest zbieżna z orientacją otrzymywaną z odbiorników GPS.



Rys. 5.4. Zależność lokalnego układu współrzędnych od odczytów GPS po przeprowadzonej kalibracji [źródło własne]

Na rysunku 5.4 widać odczyty z odbiorników GPS dla obiektu lidera i drona oznaczonego czerwonym kolorem. Poniżej pod odczytami właściwymi z odbiorników, wpisanymi czarnym kolorem, znajdują się koordynaty w lokalnym układzie współrzędnych grupy, kolor czerwony. Wraz z rotacją lidera, czy też innych dronów z odbiornikami GPS, nie zmienia się orientacja lokalnego układu grupy. Dron oznaczony na zielono reprezentuje lidera, dron oznaczony na czerwono to obiekt z odbiornikiem GPS, a dron bez markera koloru to jednostka niewyposażona w odbiornik GPS. Współrzędne oznaczone czarnym kolorem pochodzą z odbiorników GPS, a oznaczone kolorem czerwonym reprezentują współrzędne w układzie lokalnym, obliczone po kalibracji.

5.2.2. Podzielenie grupy

W przypadku konieczności podzielenia grupy, obecny lider wysyła komendy wybrania nowego lidera nowej podgrupy do każdej jednostki, która ma zostać przeniesiona do nowej podgrupy, również do lidera nowej podgrupy. Każda jednostka nowej podgrupy wysyła potwierdzenie ACK do poprzedniego lidera, identyfikując się numerem grupy, do której poprzednio należała. Lider poprzedniej grupy wysyła potwierdzenie odebrania ACK zgodnie z mechanizmem komunikacji, wymagającej potwierdzenia opisanym poniżej. Niezależnie od poprzedniego lidera, nowy lider grupy wysyła do wszystkich podległych mu jednostek zgłoszenie siebie jako lidera. Następnie ponawiany jest proces inicjalizacji dla nowo utworzonej grupy. Poprzednia grupa pozostaje we wcześniej zorganizowanej hierarchii, z tą różnicą, że zawiera mniej obiektów. ID poszczególnych jednostek nie zmienia się, natomiast wszystkim obiektom, które opuściły uprzednią grupę na rzecz nowo utworzonej, przydzielane jest nowe ID grupy. Dzięki temu matka nawet w wypadku chwilowego braku komunikacji z liderem jest w stanie precyzyjnie zidentyfikować każdą jednostkę po podziale i utworzeniu nowej grupy dzięki ID obiektu.

5.2.3. Mechanizm walidacji ramki

Omawiany protokół został zaprojektowany do obsługi bardzo dużej ilości małych ramek. Konsekwencją takiego typu komunikacji jest wysoki

stopień fałszowania ramek. Wiele z nich może przekazywać fałszywe dane lub mogłoby zostać niepoprawnie rozpoznanych. W związku z tym zaprojektowane zostały mechanizmy wielopoziomowej walidacji ramki, tak aby móc odrzucać potencjalne ramki uznane za błędne jeszcze przed ich pełnym zakończeniem. Jednocześnie autorska implementacja ukierunkowana jest na nieobciążenie systemu i minimalne opóźnienia w przerwaniach od modułu UWB, aby zapewnić bezpieczne działanie systemu sterowania obiektem. Walidacja dzieli się na następujące etapy:

1. Sprawdzenie bajtu startowego – po skompletowaniu pełnego bajtu danych następuje sprawdzenie, czy zawierał on bajt startowy wynoszący 0xFF, czyli wypełniony 8 bitami, każdy o wartości logicznej 1. Jeśli tak, rozpoczyna się zapis do buforu UWB dla poszczególnej wiadomości. Ta walidacja wykonywana jest w przerwaniu sprzętowym.
2. Sprawdzenie ID grupy – jeśli ID grupy w wysłanej ramce jest inne niż ID grupy, w której znajduje się dana jednostka, wiadomość jest odrzucana, a bufor gdzie się znajdowała, jest zerowany. Radio wraca do nasłuchiwania w oczekiwaniu na wystąpienie warunku z punktu nr 1. Ta walidacja wykonywana jest w przerwaniu sprzętowym. Jeśli na tym etapie ramka nie została odrzucona, oznacza to, że w całości zostanie zapisana do buforu kołowego, przechowującego wiadomości i kolejne walidacje będą realizowane jako planowe zadania w przypadku wolnej mocy przerobowej sterownika lotu, z racji na większą złożoność obliczeniową.
3. Sprawdzenie ID obiektu oraz flagi kierunku transmisji i lidera – jeśli flaga kierunku transmisji wskazuje na ramkę kierowaną do danego obiektu i jest to ramka identyfikowana jako pochodząca od lidera, który jako jedyna jednostka może wywoływać komunikację bezpośrednią z innymi obiektami, ramka najprawdopodobniej zostanie zakwalifikowana jako priorytetowa do rozpatrzenia. Należy przez to rozumieć, że zostanie ona ustawiona w buforze przed ramkami, które wcześniej zostały odebrane. Priorytet ramki nie jest tożsamy z priorytetem komendy, jaką w tej ramce mógł przekazać lider. Jeśli priorytet komendy będzie wysoki, poszczególne ramki będące kolejne w kolejce bufora, zostaną skasowane na etapie rozpatrywania komend.

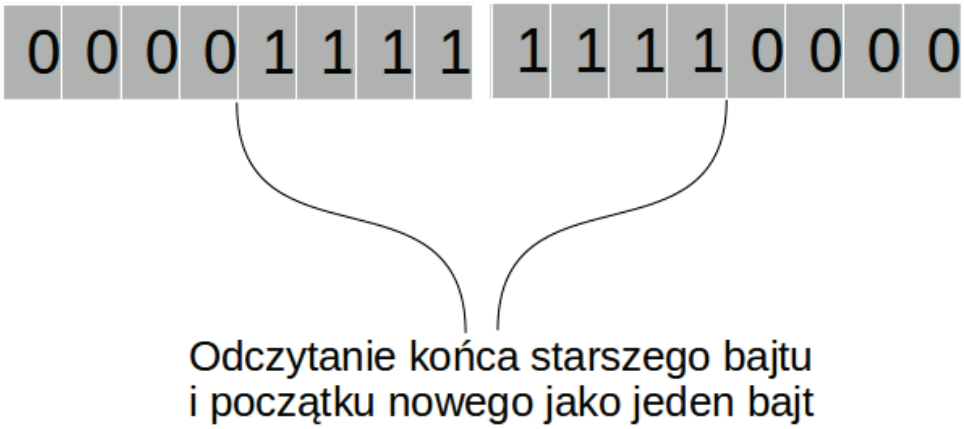
4. Sprawdzenie bajtu rozmiaru danych i porównanie z długością ramki – jeśli na tym etapie dojdzie do rozbieżności w odległości, ramka zostanie w pełni usunięta z pamięci buforu.
5. Sprawdzenie mechanizmu obliczania sumy kontrolnej CRC (Cyclic Redundancy Check) – CRC to metoda polegająca na traktowaniu danych jako wielomianu i dzieleniu ich przez ustalony wielomian zwany generatorem. Wynik dzielenia, zwany resztą lub CRC, jest dołączany do danych.

Sprawdzenie sumy kontrolnej opiera się na wykonaniu tej samej operacji i porównaniu wyniku. Jeśli wyniki sumy kontrolnej nie pokrywają się, cała ramka jest odrzucana. Jest to metoda sprawdzająca każde pole i przekłamanie na jednym bajcie skutkuje odrzuceniem ramki. Jest to zarazem ostatni etap walidacji.
W protokole zastosowano sumę kontrolną CRC-16-CCITT o wzorze generatora:

$$G(x) = x^{16} + x^{12} + x^5 + 1 \tag{5.1}$$

gdzie:
G – wielomian generatora, który jest sprawdzany,
x – zmienna formalna, czyli symbol, który reprezentuje przesunięcia bitów w rejestrze przesuwnym.

Należy zwrócić uwagę na stopniowość walidacji i złożoność jej poszczególnych kroków. Protokół został zoptymalizowany, aby kroki walidacji pokrywały się z kolejnością wysyłanych bajtów. Jako pierwszy sprawdzany jest bajt startu zadany jako 0xFF. Jest to jedyny bajt, który w przypadku niezaszumionej ramki może wynosić 0xFF. Istnieje jednak prawdopodobieństwo odebrania i złożenia wartości bitów z sąsiadujących bajtów. Przedstawione zostało to na rysunku 5.5.



Rys. 5.5. Możliwe przesunięcie bitowe prowadzące do błędnego rozpoznania bajtu startowego [źródło własne]

Jak widać na rysunku 5.5, możliwe jest, aby dwa sąsiadujące bajty zawierały ciąg 8 jedynek logicznych, które odbiorca mógłby omylnie zinterpretować, jako jeden pełny bajt i zwalidować go poprawnie. Należy zaznaczyć, że koncepcja autorskiego protokołu przygotowana jest do pracy z modułami UWB różnych producentów i nie opiera się o wbudowane mechanizmy walidacji, które mogłyby zabezpieczyć przed taką interpretacją danych. W przypadku opisanego powyżej błędu walidacji, pozostają kolejne kroki, w których ramka z przesunięciem w odczycie zostanie rozpoznana jako błędna i usunięta.

Początkowe dwa kroki walidacji mają stałą złożoność algorytmiczną, a sama walidacja opiera się na porównaniu przez pojedynczą instrukcję warunkową wykonywaną atomowo, czyli niemogącej być niedokończoną przez przerwanie. Gwarantuje to szybką diagnostykę danych odporną na wypadek przypadkowego scalenia sekwencyjnych fragmentów ramek w fałszywy bajt startowy. Kolejna weryfikacja polegająca na sprawdzeniu ID grupy ma za zadanie odrzucić ramkę z fałszywym bajtem startowym. Jednocześnie wartości zarówno bajtu startu, jak i stopu, są tak dobrane, aby maksymalnie rzadkie było ich przypadkowe wystąpienie.

5.2.4. Zapisywanie buforu z wiadomością do pamięci jednostki

W przygotowanym protokole zastosowano autorskie podejście do implementacji buforowania danych. Jest ono zoptymalizowane pod względem zużycia pamięci w sterowniku. Bufor dla odebranych danych jest buforem kołowym typu FIFO (*First In First Out*) – gdzie pierwsza walidowana ramka zapisywana jest w formie struktury do buforu. Sam bufor w zależności od parametrów sterownika może być alokowany jako tablica statyczna lub tablica dynamiczna. Dla sterowników o wystarczająco dużej ilości pamięci RAM lepszą implementacją jest tablica statyczna ze względu na mniejsze obciążenie procesora podczas pracy. Natomiast dla sterowników o mniejszej ilości pamięci RAM można zastosować bufor z zaimplementowaną tablicą dynamiczną. Domyślnie zastosowany jest pierwszy wariant, który dla sterowników typu np. Pixhawk (omówionych w rozdziale 3) jest optymalny.

W buforze jako pojedynczy element zapisywana jest struktura danych, odpowiadająca formatowi ramki przedstawionemu na początku rozdziału. Do struktury zostało dodane dodatkowe pole niebędące bezpośrednio wysłane przez nadający obiekt. Jest to dystans w linii prostej, dzielący obiekty, rozpoznany i wyliczony dzięki możliwością wyznaczania dystansu przy użyciu modułów UWB. Dystans jest zapisywany podczas odbierania ramki i obliczany, gdy weryfikacja dwóch pierwszych mechanizmów walidacji przebiegnie pomyślnie.

Po zapisaniu ramki, co ma miejsce w przerwaniu sprzętowym, następuje dodanie do kolejki dwóch następujących zadań: „walidacja ramki” oraz „analiza ramki”. Zadanie walidacji opiera się na wykonaniu kroków do 3 do 5 z listy kroków walidacji (patrz 5.2.3). Jeśli ramka zostanie zakwalifikowana jako poprawna, może zostać wyzwolone analizowanie ramki, gdzie wykonywane są następujące czynności:

1. Weryfikacja bezpieczeństwa jednostki przez rozpoznanie dystansu od obiektu, który wysłał ramkę. Jeśli jest on zbyt blisko, następuje natychmiastowe skorygowanie lotu, przy czym sprawdzana jest odległość od lidera, tak aby nie zagrozić bezpieczeństwu lidera. W przypadku luki w postaci większej ilości wolnego miejsca po przeciwstawnej stronie

obiektu, który odebrał dane, następuje delikatne przesunięcie tego obiektu celem bardziej równomiernego wypełnienia przestrzeni grupy.
2. Jeśli ramka pochodzi od obiektu posiadającego GPS, następuje weryfikacja pozycji własnej obiektu, który odebrał ramkę.
3. W każdej ramce pochodzącej od lidera następuje weryfikacja kursu oraz pozycji własnej względem przyjętego kształtu grupy.
4. Jeśli ramka od lidera zawierała komendę, następuje dodanie funkcji komendy do kolejki zadań zgodnie z priorytetem, jaki powinna mieć ta komenda oraz odesłanie potwierdzenia liderowi. Ustawiony jest również semafor czekający na weryfikacje odebrania potwierdzenia od lidera. Jeśli nie nadejdzie ono w wyznaczonym czasie, jednokrotnie ponawiane jest wysłanie potwierdzenia, zgodnie ze schematem komunikacji z potwierdzeniem.

5.2.5. Komendy i informacje wysyłane jako informacje dodatkowe

Informacje dodatkowe stanowią rozszerzenie protokołu o komendy, jakie mogą być przesyłane między jednostkami. Może to być użyteczne np. podczas przydziału jednostek do podgrup podczas wykonywania zadania. Przykład komendy zawartej w informacjach dodatkowych przedstawia tabela 5.1.

Tab. 5.1. Przykładowe komendy autorskiego protokołu zawarte wewnątrz pola dodatkowych informacji [źródło własne]

ID	**Typ**	**Priorytet**	**Korespondująca funkcja**	**Argumenty przekazane**
1	Komenda	Wysoki	Wybranie lidera	Numer ID nowej grupy
2	Komenda	Wysoki	Żądanie surowego odczytu GPS	brak
3	Informacja	Wysoki	Wysłanie surowej pozycji GPS	Pozycja X, Y, Z odczytana z GPS
4	Komenda	Wysoki	Żądanie synchronizacji GPS	Współczynniki korekcji dla odbiornika GPS

5	Informacja	Niski	Odczyt czujnika	Wartość odczytana z czujnika

5.2.6. Schemat potwierdzenia odebrania danych ACK

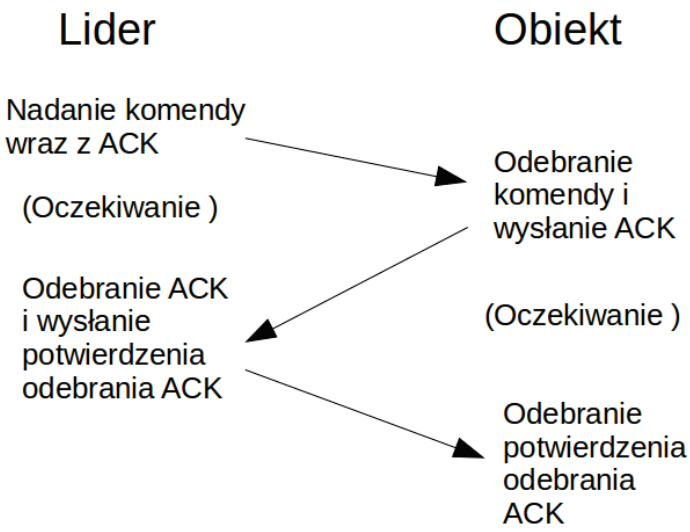
Procedura potwierdzenia ACK zapewnienia, że dane zostały poprawnie odebrane przez odbiorcę. Dodatkowo, nadawca potwierdza odbiór ACK od odbiorcy, aby zapewnić pełną dwukierunkową komunikację. Jeśli odbiorca nie otrzyma tego potwierdzenia, ponownie wysyła ACK jeden raz.

W celu skrócenia procedury komunikacji opracowany protokół umożliwia wysłanie minimalistycznej ramki służącej tylko do potwierdzenia ACK i potwierdzenia odebrania ACK przez pierwotnego nadawcę.
Struktura uproszczonej ramki zawsze wynosi 8 bajtów. Przedstawia się ona następująco:

- Bajt Startowy (1 bajt),
- ID Grupy (1 bajt),
- ID Obiektu (1 bajt),
- Flagi bitowe (2 bajty),
- CRC (2 bajty),
- Bajt Stopu (1 bajt).

Pomijane są kroki 3. i 4. walidacji ramki. Należy mieć na uwadze, że ramka ACK jest skróconą formą zawierającą tylko wybrane pola, opisane podczas analizy pełnej ramki. W przypadku potrzeby przesłania dodatkowych danych wymagana ramka jest w pełnej formie.

Na rysunku 5.6. przestawiono algorytm procedury potwierdzenia ACK.



Rys. 5.6. Algorytm komunikacji wraz z potwierdzeniami ACK [źródło własne]

Wyróżniamy tutaj 6 stanów:

1. **Nadanie ramki danych przez lidera:**
   - Nadawca wysyła ramkę danych do odbiorcy.
   - Ramka danych zawiera unikalne ID, które identyfikuje tę ramkę.

2. **Oczekiwanie na ACK przez lidera:**
   - Nadawca przechodzi w tryb oczekiwania na potwierdzenie (ACK) od odbiorcy. Oczekiwanie jest realizowane przez mechanizm semafora systemu czasu rzeczywistego obsługiwanego przez oprogramowanie sterownika lotu. Dzięki temu wątek nie jest blokowany, pozostała komunikacja i obsługa priorytetowych zadań sterownika lotu odbywa się bez zakłóceń.
   - Jeśli nadawca nie otrzyma ACK w określonym czasie (*timeout*), ponownie wysyła ramkę danych.
3. **Odbiór ramki danych i wysłanie ACK przez obiekt:**
   - Odbiorca odbiera ramkę danych.

- Odbiorca sprawdza poprawność ramki,
- Jeśli ramka danych jest poprawna, odbiorca wysyła potwierdzenie (ACK) do nadawcy.

4. **Oczekiwanie na potwierdzenie ACK przez obiekt:**
    - Odbiorca oczekuje na potwierdzenie odebrania ACK, wysłane od nadawcy. Tak jak w przypadku oczekiwania w punkcie nr 2, oczekiwanie nie blokuje wątku oraz pozostałych procesów wymiany danych.
5. **Odbiór ACK i wysłanie potwierdzenia odbioru ACK przez lidera:**
    - Nadawca odbiera ACK.
    - Nadawca sprawdza, czy ID ramki w ACK odpowiada ID wysłanej ramki danych.
    - Nadawca wysyła potwierdzenie odbioru ACK do odbiorcy. Jest to wiadomość najniższego priorytetu. W przypadku konieczności obsługi wielu wiadomości może nie zostać nadana żadna nowa ramka z racji na brak zasobów obliczeniowych.
    - Jeśli ID się zgadza, proces jest zakończony. Jeśli nie, nadawca ponownie wysyła ramkę danych.
    - Nadawca wysyła potwierdzenie odbioru ACK do odbiorcy.
6. **Odebranie potwierdzenia przez obiekt**:
    - Jeśli odbiorca nie otrzyma potwierdzenia w określonym czasie, ponownie wysyła ACK jeden raz.
    - Po ponownym wysłaniu potwierdzenia, niezależnie czy otrzymane zostanie potwierdzenie, odbiorca uznaje transmisje za udaną.

W przypadku komend, które żądają przysłania informacji zwrotnych, których odczyt można uznać za nieopóźniający w komunikacji, nie ma zastosowania skrócona ramka potwierdzenia przez obiekt nie będący liderem. W zamian flaga potwierdzenia odebrania wysyłana jest w pełnej ramce wraz z informacjami żądanymi.

5.3. Bezpieczeństwo i nawigacja

Kluczowe w aspekcie stabilnej pracy roju jest bezpieczeństwo wynikające z mechanizmów zapobiegania kolizjom. Należy wyróżnić dwa typy kolizji: kolizje wzajemne obiektów i kolizje z przeszkodami. Na potrzeby założonej modułowości systemu sterowania zastosowano następujący podział. Kolizje wzajemne rozpatrywane są wewnątrz grupy na poziomie autonomii poszczególnych jednostek. Kolizje wynikające z obecności przeszkód zarządzane są na wyższym poziomie. W tym przypadku lider otrzymuje odpowiedz od matki, która zbierając informacje od liderów, mapuje teren. Na poziomie matki-operatora spoczywa odpowiedzialność za zadanie trajektorii dla grupy tak, aby ta uniknęła przeszkody. W przypadku, gdy ostatnia zadana trajektoria dla grupy podroju zostaje rozpoznana przez lidera jako kolizyjna, może on podjąć decyzje o bezpiecznym wycofaniu podroju. Najwyższy priorytet ma autonomia jednostki, której celem jest uniknięcie kolizji wzajemnej. Następnie, jeśli nie jest to zagrożone, realizowana jest autonomia grupy, która w przypadku możliwej kolizji w przeszkodą jest wycofywana z misji przez lidera. Jeśli ten poziom jest niezagrożony, wykonywana jest misja narzucona przez matkę-operatora. Należy pamiętać, że to do zadań matki należy dobieranie trajektorii tak, aby żaden z wyższych priorytetów nie był zagrożony. Istnieją jednak sytuacje, kiedy z powodu nieprzewidzianych zdarzeń, bezpieczeństwo jednostek może być zagrożone. Priorytet reakcji w tej sytuacji zakłada najszybsze możliwe działanie, czyli każda jednostka troszczy się o swoje bezpieczeństwo. Z racji na założenie, że nie każda jednostka posiada czujniki pozwalające wykryć przeszkodę, obowiązek reakcji na potencjalną kolizje ciąży na matce roju. W ramach modułu komunikacji zawarte są mechanizmy autonomii jednostki i autonomii grupy, które odpowiadają kolejno za wzajemną bezkolizyjność.

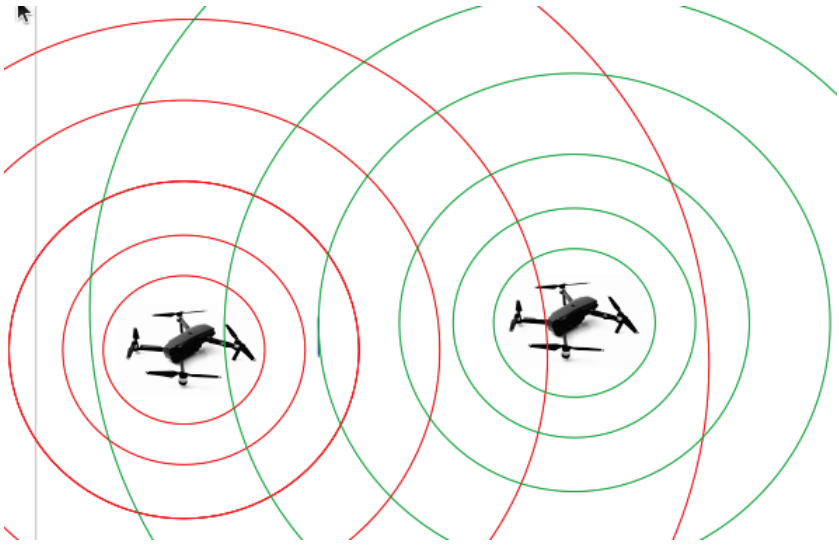
5.3.1. Autonomia jednostki

Bezkolizyjność jednostek jest zagadnieniem często rozważanym przez naukowców. Zasadniczym podejściem do tego tematu jest rozpoznanie kąta i odległości pomiędzy jednostkami [203]. Innym podejściem jest poleganie na wyznaczeniu pozycji jednostek w jednym układzie współrzędnych

i wyznaczenie bezpiecznego pułapu dystansu między jednostkami [59]. Kolejną propozycją było modelowanie fizycznych ograniczeń w postaci klatki, która odpychałaby inne jednostki [123]. Opracowania te wielokrotnie były rozważane symulacyjnie, bez implementacji na platformach sprzętowych i uwzględnienia systemów komunikacji oraz fizycznej realizacji przedstawionej koncepcji [212], [194], [96].

Zaproponowane rozwiązanie opiera się na autorskim protokole, który został zaprojektowany, aby umożliwić pomiar dystansu między jednostkami i dynamiczne wyznaczanie najbliższych jednostek w grupie wraz z jednoczesnym identyfikowaniem ich ID. Zgodnie z założeniami skalarności, elastyczności, a także przystępności cenowej, aby móc zastosować sterowanie w roju w sektorach cywilnych, przedstawione rozwiązanie może funkcjonować bez czujników wykrywania przeszkód na każdym UAV, tak jak w pracy badawczej z 2020 [203]. Ma to też zaletę w postaci zachowania stopniowej autonomii bezkolizyjności. W przytoczonej pracy każdy obiekt, który znalazłby się w dystansie grożącym kolizją, byłby traktowany tak samo, niezależnie czy byłby to dron z podgrupy, czy też przeszkoda statyczna. Dzięki rozróżnieniu tych obiektów, czego możliwość daje koncepcja oparta o autorski protokół, w wyższym priorytecie traktowane jest ominięcie kolizji z innym obiektem, która to kolizja zakończyłaby się utratą więcej niż jednej jednostki. Do tego możliwe jest wzajemne, uczone i synchroniczne reagowanie wielu jednostek, co w przypadku konieczności zagęszczenia szyku roju jest kluczowym czynnikiem. Innowacja podejścia do unikania kolizji polega na odbieraniu pozycji sąsiadującej jednostki wraz z jej pozycją w spójnym dla wszystkich jednostek w grupie układzie współrzędnych.

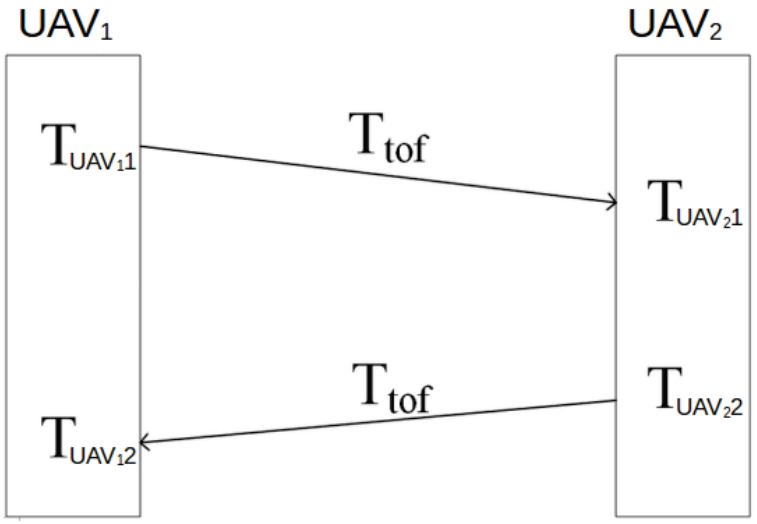
Wszystkie drony komunikują się ze sobą za pośrednictwem UWB, wyznaczając wzajemną odległość. Każdy cyklicznie rozgłasza swoją ramkę ogłoszeniową, co przedstawiono na rysunku 5.7.



Rys. 5.7. Rozgłaszanie ramek wiadomości o pozycji własnej [źródło własne]

Istnieją publikacje naukowe, w których badacze analizowali popularne algorytmy do ustalania pozycji wzajemnej, takie jak ToF (*Time of Flight*) [61], [94], TWR (*Two Way Ranging*) [120], TDOA (*Time Difference of Arrival*) [95] czy AoA (*Angle of Arrival*) [144] do ustalania odległości między jednostkami.

Wszystkie te algorytmy korzystają z założenia, że fala UWB jest falą elektromagnetyczną, a co za tym idzie prędkość propagacji sygnału w próżni jest równa prędkości światła. Przyjmuje się jako przybliżenie, że również w warunkach powietrznych prędkość fali elektromagnetycznej jest równa prędkości światła. Według założeń wyznaczania odległości według ToF czasy rozgłaszania przedstawia rys. 5.8.



Rys. 5.8. Obliczanie dystansu między dwoma dronami na podstawie *Time of Flight* [źródło własne]

Na rysunku 5.8 zaznaczono czasy poszczególnych jednostek, gdzie:
$T_{UAV_1 1}$ – punkt czasowy wysłania ogłoszenia o pozycji własnej przez pierwszego drona,
$T_{UAV_2 1}$ – punkt czasowy odebrania ogłoszenia o pozycji pierwszego drona przez drugiego drona,
$T_{UAV_2 2}$ – punkt czasowy wysłania ogłoszenia o pozycji własnej przez drugiego drona,
$T_{UAV_1 2}$ – punkt czasowy odebrania ogłoszenia o pozycji pierwszego drona przez drugiego drona,
T_{tof} – czas pomiędzy wysłaniem ogłoszenia przez jednego z dronów, a odebraniem sygnału zwrotnego przez tego samego drona.

Dystans między obiektami możliwy jest do wyliczenia zgodnie ze wzorem:

$$d = \frac{c \cdot T_{\text{tof}}}{2} \tag{5.2}$$

gdzie:
d – dystans,
c – prędkość światła.

Założeniem często stawianym algorytmowi ToF jest natychmiastowa odpowiedź na otrzymany sygnał w postaci wysłania własnego ogłoszenia, zsumowanie czasu przepływu danych w obie strony i podzielenie rezultatu przez dwa. Podobnym algorytmem jest TWR, który wprowadza dodatkowe opóźnienie od odbioru pierwszego sygnału przez drugiego drona do wysłania informacji zwrotnej. Opóźnienie musi mieć znaną wartość.

Są to dwa najprostsze algorytmy do ustalania wzajemnej pozycji. Nie wymagają synchronizacji zegarów z racji na odpowiedź drugiego obiektu. Metody te są implementowane na poziomie sprzętowym w modułach UWB. Doskonale nadają się do wyznaczania dystansu w relacji 1 do 1. Nie są możliwe do zastosowania w pracy z racji na relacje wielu do wielu, a także w przyjętej autorskiej koncepcji, która zakłada wykorzystanie radia UWB również do przesyłania danych. Użycie wspomnianych algorytmów, spowodowałoby niemożliwość natychmiastowej odpowiedzi na wysłany pierwszy sygnał, a także niemożliwość ustalenia stałego opóźnienia.

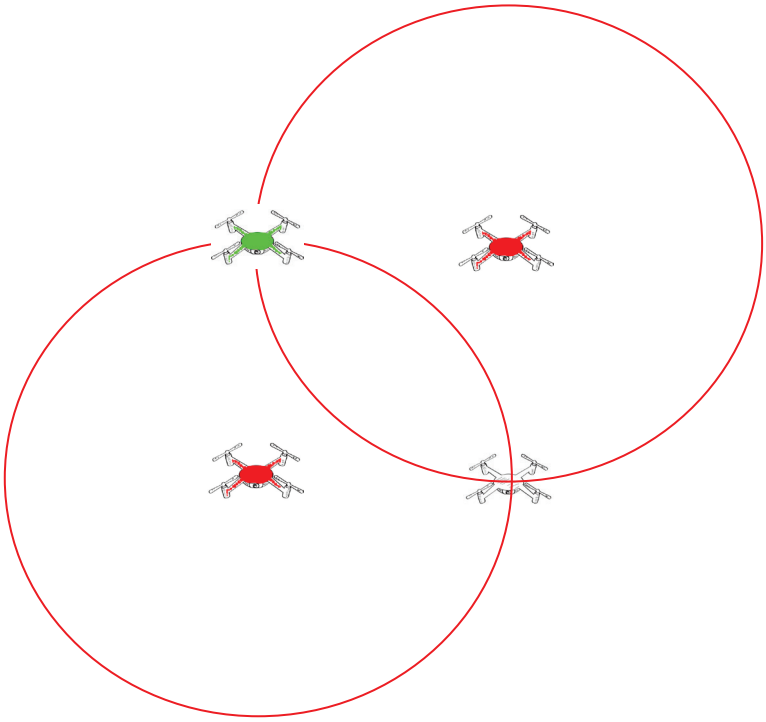
W celu wykorzystania potencjału możliwości komunikacji UWB wykorzystano algorytm TDOA (*Time Difference of Arrival*). Wymaga on synchronizacji zegarów modułów sprzętowych. Dzięki ustaleniu zsynchronizowanego czasu możliwe jest zmierzenie dystansu. Jest on obliczany na podstawie różnicy czasu przybycia sygnału od nadajnika do kilku odbiorników. Sygnał wysyłany przez nadajnik dociera do odbiorników w różnych momentach, a różnice czasowe są przekształcane na różnice dystansów. Wzór na różnicę odległości w systemie TdoA przedstawia się następująco:

$$\Delta d_{ij} = c \cdot \Delta t_{ij} \tag{5.3}$$

gdzie:
Δd_{ij} – różnica odległości między dronami
c – prędkość światła w próżni c ≈ 3×10^8 m/s,

$\Delta t_{ij} = t_j - t_i$ – różnica czasów przybycia sygnału do odbiorników UWB dronów i oraz j.

W celu obliczenia pozycji jednostki względem obiektów o ustalonych pozycjach, stosuje się różnice czasowe przybycia sygnałów do wielu odbiorników. Każda różnica czasów zarejestrowanych w modułach UWB może posłużyć do wyznaczenia okręgów na płaszczyźnie, na której znajduje się nadajnik. W miejscu przecięcia okręgów dla różnych par odbiorników można określić pozycję nadajnika (rys. 5.9).



Rys. 5.9. Ustalenie pozycji drona bez modułu GPS (kolor zielony) przy pomocy dwóch dronów z odbiornikami (kolor czerwony) [źródło własne]

W przypadku systemu z więcej niż dwoma drona z odbiornikiem GPS należy utworzyć układ równań, gdzie dla każdej pary UAV obliczamy różnicę odległości posługując się poniższym wzorem:

$$\Delta d_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{5.4}$$

gdzie:
x_i, y_i– współrzędne jednostki, która ma za zadanie obliczyć swoją pozycje,
$x_j y_j$, – współrzędne drona z odbiornikiem GPS.

Poprzez rozwiązanie układu równań, który należy utworzyć przez przygotowanie ilości równań (5.4) odpowiadającej ilości par dronów, można wyznaczyć pozycję nadajnika. W przypadku tylko dwóch bezzałogowych obiektów latających z odbiornikiem GPS brakuje jednej danej, umożliwiającej precyzyjne rozpoznanie pozycji. Na podstawie wzoru 5.3 możliwego do wykorzystania dzięki technologii UWB każda jednostka może wyznaczyć swoją odległość od obiektu, który wysyła dane ze swoją pewną pozycją, ustalaną dzięki odbiornikowi GPS. Następnie jednostka oblicza swoją pozycje rozwiązując układ równań oparty na wzorze 5.4, w którym zna odległość od obiektu, który przesłał ramkę danych ze swoją pozycją. Przy stałej wysokości lotu możliwe jest zebranie wystarczających informacji od dwóch dronów z odbiornikami GPS co ilustruje rysunek 5.9. Dron oznaczony kolorem zielonym, ustala swoją pozycję na podstawie danych pozyskanych z modułu UWB. Dwa drony, oznaczone kolorem czerwonym, służą jako punkty odniesienia. Często w terminologii technologii UWB nazywane są kotwicami (z ang. *anchor*). Standardowo przyjętym jest, że kotwica jest elementem nieruchomym względem którego, na podstawie zmierzonego dystansu, obliczana jest pozycja obiektu ruchomego obiektu. Innowacyjnym w opracowanej koncepcji jest użycie dronów z odbiornikami GPS jako punktów odniesienia, umownie nazwanych ruchomymi kotwicami. Jest to możliwe dzięki wcześniejszej korekcie odbiorników GPS i ustaleniu lokalnego układu współrzędnych, z liderem w centrum.

Dron bez własnego odbiornika GPS (oznaczony jako dron bez wypełnienia kolorem na rysunku 5.9) wykonuje zadanie triangulacji kąta. Nie jest rekomendowane, aby w roju były tylko dwie jednostki z odbiornikiem GPS. Natomiast w przypadku kiedy musi dojść do podziału roju na dwie grupy, możliwym jest, aby minimalnie dwie jednostki zawierały odbiornik GPS. W tej sytuacji dron, który ustala swoją pozycję, na podstawie

poprzednich danych o własnej pozycji odrzuca jeden z dwóch wariantów możliwej pozycji, który jest niepoprawny.

Kolejnym założeniem, które pozwala uprościć do niezbędnego minimum możliwości jednostek z odbiornikami GPS w podroju, jest rozpatrywanie pozycji, tak jakby była dwuwymiarowa. Z racji na to, że każda jednostka jest wyposażona w czujniki inercyjne, na podstawie których może odczytać swoją wysokość, ta dana jest traktowana jako znana. W ramce odbieranej od bezzałogowych obiektów latających z odbiornikami GPS zawarte są informacje o pozycji w osi Z tych obiektów. Dron, który wyznacza swoją pozycję w lokalnym dla lidera układzie współrzędnych, oblicza różnicę w swojej wysokości względem wysokości jednostki, od której otrzymał dane i uwzględnia to w wyznaczeniu swojej pozycji na podstawie dystansu zmierzonego przez moduł UWB algorytmem TDOA.

5.3.2. Ustalenie lokalnego układu współrzędnych

Dzięki wzajemnej komunikacji każdy dron przekazuje swoją wysokość przez co każdy inny uczestnik podroju może obliczyć różnicę wysokości pomiędzy poszczególnymi dronami. Pozycja lidera jest pozycją bazową układu przyjętego dla całego podroju. Względem niej obliczane będą pozycje wszystkich uczestników podroju. W pierwszej kolejności należy ustalić wzajemną pozycje dla wszystkich obiektów mających odbiorniki GPS. Jest to faza, w której każdy z dronów z nadajnikiem GPS identyfikuje się wraz ze swoją pozycją do lidera, który na podstawie otrzymanych danych oraz pomiaru odległości między nim a jednostką, która wysłała dane, ustala czy uznać pozycje za prawdziwą. Zapobiega to sytuacji, w której odczyty GPS wykazują błędy względem odczytu wzajemnej odległości. Obiekt będący liderem po odebraniu danych z nadajnika GPS przesłanych przez inną jednostkę, oblicza wzajemną odległość i porównuje ją z dystansem arbitralnym, ustalonym dzięki modułowi UWB. Poprzez moduł UWB dystans arbitralny wyznaczany jest dzięki algorytmowi TDOA.

Następnie lider porównuje obliczoną odległość wzajemną z dystansem zmierzonym z komunikacji UWB. Jeśli różnice przekraczają możliwy błąd UWB wynoszący 10 cm, należy wprowadzić korektę. Optymalnym rozwiązaniem jest odpytanie w pierwszej kolejności wszystkich obiektów posiadających odbiorniki GPS o pozycje, aby korektę wprowadzić

całościowo, bez potrzeby iteracji dla każdego odpytywanego obiektu. W tym celu proponowanym rozwiązaniem jest algorytm:

1. Odpytanie wszystkich obiektów z nadajnikami GPS o ich pozycje.
2. Jeśli większość wyliczonych dystansów będzie mieścić się w zakresie błędu UWB, należy przyjąć, że pozycja lidera jest poprawną pozycją i może być uznana jako środek układu współrzędnych podroju. Do obiektów których dystans został uznany jako rozbieżny od zmierzonego przez UWB, należy wprowadzić współczynniki korekcji pozycji według następujących wzorów:

$$X_{\text{koryg}} = X_{\text{popr}} + k_x \cdot \frac{D_{GPS} - D_{UWB}}{D_{GPS}} \qquad (5.5)$$
$$Y_{\text{koryg}} = Y_{\text{popr}} + k_y \cdot \frac{D_{GPS} - D_{UWB}}{D_{GPS}}$$

gdzie:
X_{koryg} i Y_{koryg} – skorygowane współrzędne rozpatrywanego obiektu,
X_{popr} i Y_{popr} – współrzędne odebrane z nadajnika GPS na pokładzie rozpatrywanego obiektu,
D_{GPS} – dystans między dwoma dronami wyliczony na podstawie danych z GPS obu dronów, obliczany na podstawie wzoru 5.4,
D_{UWB} – dystans zmierzony przez moduł UWB między dwoma dronami,
k_x i k_y – współczynniki korekcji dla osi X i Y.

3. Jeśli większość wyliczonych dystansów będzie rozbieżnych od możliwego błędu UWB, należy przyjąć, że pozycja lidera jest niepoprawna, to znaczy jego odczyt GPS może być niearbitralny dla całego podroju. Należy wtedy wprowadzić modyfikacje, dodając współczynniki korekty dla lidera. Każdy odczyt odbiornika traktowany jest z równym priorytetem. Odczyt pozycji wysokości przyjęty jest jako arbitralny. Pozostałe koordynaty, czyli X i Y należy

wyliczyć, wykorzystując wzory 7 i 8, przy czym w tym wypadku należy użyć go do obliczenia pozycji i korekt dla lidera, traktując pozycje zgłaszającego się drona z odbiornikiem GPS jako arbitralną. Procedurę należy powtórzyć dla każdego drona, wyznaczając *n* pozycji dla lidera, gdzie *n* to liczba dronów z GPS. W *n* należy również zawrzeć pozycje pierwotnie wskazaną przez odbiornik GPS na pokładzie lidera. Następnie lider oblicza średnią arytmetyczną ze wszystkich *n* pozycji. Po wyliczeniu nowej pozycji ustala współczynniki korekty.

4. Kolejno należy dokonać korekty reszty jednostek z GPS, odnosząc je do pozycji lidera i wprowadzając, jeśli potrzeba, współczynniki korekcji dla pozostałych jednostek z odbiornikiem GPS.

Przedstawiony wyżej algorytm został maksymalnie uproszczony, aby mógł zostać obliczony w jednostkach UAV w czasie rzeczywistym. Opóźnienie wynikające z przesyłu pozycji drona z GPS do lidera należy uznać za nieistotne. Przykładowo dla odległości między dronami wynoszącej 20 m opóźnienie w przesłaniu danych wyniesie około 67 nanosekund. Dla systemów czasu rzeczywistego jest to pomijalna wartość. Dla każdej informacji przychodzącej z drona wyposażonego w GPS lider wykonuje obliczenia, odnosząc do pozycji z własnego odbiornika GPS, pobranej w momencie otrzymania pakietu z informacją od innego drona. W przypadku gdy wymagana jest korekta pozycji lidera z punktu 3 przedstawionego algorytmu korekcji, każda iteracja wyznaczenia *n* pozycji, z których następnie należy będzie wyliczyć średnią, jest zapisywana do pamięci nieulotnej sterownika i obliczana dla aktualnej w danym czasie pozycji. Zgodnie z możliwościami obliczeniowymi, które mogą nie być stałe ze względu na priorytetowość zadań w kolejce systemu czasu rzeczywistego i ograniczone przez konieczność otrzymaniu kursu oraz pozycji drona, należy przyjąć koncepcję reagowania na przychodzące wiadomości bez buforowania ich. Oznacza to, że nie można oczekiwać od sterowników wszystkich jednostek z GPS, aby w tym samym czasie wysłały wiadomość o swojej pozycji do lidera. W związku z czym lider musi użyć koncepcji mutexu. Jest to programowy mechanizm zabezpieczający przed uruchamianiem tego

samego kodu przez różne wątki oprogramowania w tym samym czasie. Mutex blokuje liderowi możliwość odczytania informacji odebranej od drona z GPS, podczas gdy wcześniej została odebrana informacja pochodząca z innego źródła.

Omawiane mechanizmy mają na celu umożliwienie pracy roju przy minimalnym możliwym nakładzie pieniężnym. Jak zostało ustalone w rozdziale 3, zasadnym jest, aby każdy obiekt wyposażyć w odbiornik GPS. Należy również mieć na uwadze, że w przypadku, gdy nie ma gwarancji, że na drodze roju podczas wykonywania pracy nie pojawi się przeszkoda, należy wyposażyć część jednostek w oprzyrządowanie do wykrywania przeszkód. Zostało ono omówione wraz z wymaganiami stawianymi poszczególnym jednostkom w rozdziale 3. W kontekście roju jako grupy nie ma konieczności, aby każda jednostka posiadała wspomniane oprzyrządowanie, ponieważ omijanie przeszkód realizowane jest na wyższym poziomie autonomii przez matkę-operatora. Dane od jednostki, która posiada osprzęt do wykrywania przeszkód, przekazywane są za pomocą protokołu komunikacji wewnątrz roju do lidera. Pozycja przeszkody ustalana jest w lokalnym układzie współrzędnych. Lider przekazuje informacje o rozpoznanych przeszkodach do programu sterującego matki-operatora. Ominięcie przeszkód i udział matki-operatora jest treścią rozdziału 6.

5.3.3. Kompensacja szyku

Wykorzystując omówione w podrozdziale 5.3.1 mechanizmy do pomiaru odległości bezzałogowego obiektu latającego od sąsiadujących jednostek, priorytetowym zadaniem uczestników roju jest uniknięcie kolizji wzajemnej. Ma to miejsce, gdy jednostka przekroczy zadany próg bezpiecznego dystansu, który powinien być zdefiniowany zgodnie z gabarytem i dynamiką obiektów przewidzianych do konkretnego zadania. W przypadku gdy bezpieczeństwo jednostki nie jest zagrożone, odczyty wzajemnych pozycji mogą być również wykorzystane. Jeśli przewidziany jest szyk, w którym odległość od jednostek powinna być w rzędzie taka sama, zasadnym jest, aby drony korygowały swoją pozycję. Aby nie obciążać mocy sterownika lotu, został zaprojektowany prosty mechanizm kompensacji, polegający na tym, że obiekty mają za zadanie przybliżać lub oddalać się od

sąsiadujących jednostek, tak aby dążyć do zniwelowania różnic w dystansach od sąsiadujących dronów w rzędzie. Mechanizm ten ma zastosowanie szczególnie w przypadku utraty komunikacji z matką-operatorem, która zarządza planowaniem trajektorii. W rozpatrywanym przypadku braku łączności, jednostki same korygują swój szyk, dążąc do symetrii formacji, wykonując korekty podczas przelotu. Wszystkie korekty dokonywane są dynamicznie, proporcjonalnie do dystansu jaki ma zostać zredukowany lub powiększony.

5.4. Walidacja opracowanej komunikacji wewnętrznej

Zgodnie z przedstawionym zakresem badań w rozdziale 2 wykonano eksperymenty, mające sprawdzić komunikacje przy użyciu autorskiego protokołu do komunikacji wzajemnej opartej o łączność UWB. Opracowane środowisko graficzne opiera się na programie Gazebo, który nie posiada natywnego wsparcia dla technologii UWB. Umożliwia dodanie własnych czujników i wykorzystanie pakietów metasystemu ROS. Aby wprowadzić komunikacje wewnątrz roju, do środowiska graficznego wykonano następujące kroki:

1. Przygotowano modyfikację funkcji zapisującej do pola struktury ramki danych protokołu z informacją o wzajemnym dystansie. Zostało to zrealizowane w oprogramowaniu ArduPilot, korzystając z flag kompilacji sterownika lotu. Dzięki temu podczas kompilowania oprogramowania dla fizycznego sterownika lotu, używana jest funkcja bazująca na odczycie z fizycznego modułu UWB. W przypadku kompilacji oprogramowania dla środowiska graficznego używana jest zmodyfikowana funkcja, która korzysta z symulowanego w programie Gazebo czujnika.
2. Przygotowano opis czujnika w pliku SDF (*Simulation Description Format*) właściwego dla programu Gazebo.
3. Opracowano kod programu w języku C++, który na podstawie danych wejściowych o poszczególnych dronach, możliwych do uzyskania z programu Gazebo, symuluje pracę modułu UWB, wykonując algorytm TDOA. Autorska implementacja modułu jest traktowana w programie Gazebo tak samo jak implementacje pozostałych czujników, przygotowanych przez programistów tworzących to oprogramowanie. Kod jest kompilowany

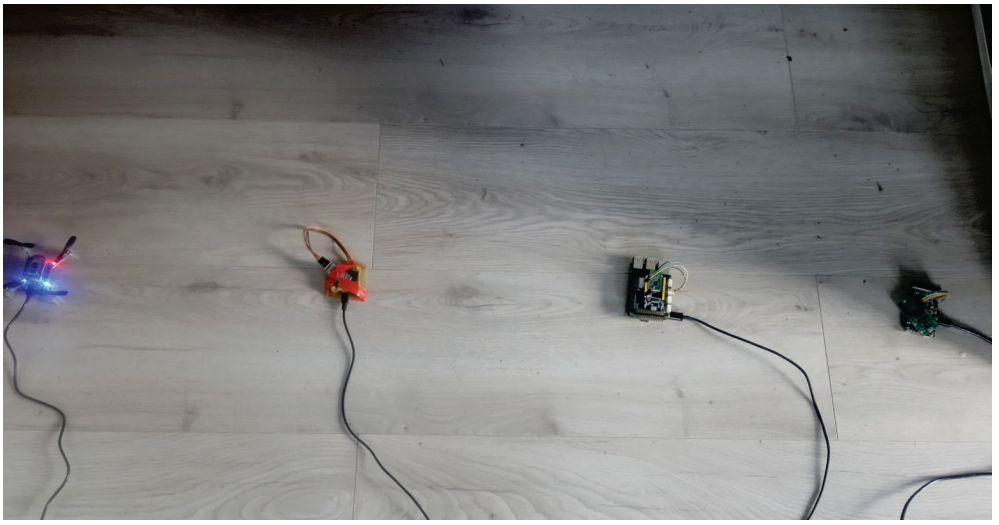
do poziomu biblioteki, która po umieszczeniu we właściwym folderze jest traktowana tak samo jak biblioteka natywna.

5.4.1. Walidacja komunikacji wewnątrz roju z użyciem fizycznych sterowników lotu

Jako pierwszy eksperyment przewidziany dla tego modułu przygotowano prosty test dla 4 fizycznych sterowników lotu. Wykorzystano 4 różne sterowniki lotu o wymaganiach zgodnych ze zdefiniowanych w rozdziale 3. Wszystkie ułożono w linii w równych wzajemnych odległościach. Do dwóch losowych sterowników dołączono odbiornik GPS. Scenariusz tego badania zakłada włączenie wszystkich sterowników, z wgranym zmodyfikowanym oprogramowaniem ArduPilot, obsługującym komunikację wewnątrz roju. Po uruchomieniu sterowników nastąpi przeprowadzanie procesu inicjalizacji roju i określenie pozycji wszystkich sterowników w lokalnym układzie współrzędnych. Na tym etapie walidowana jest jakość ustalenia pozycji, wykorzystując wiedzę o rozmieszczeniu sterowników lotu jako arbitralny punkt odniesienia. Następnie od jednego ze sterowników odcinane jest zasilanie. Jest to wymuszenie odpowiadające dynamicznej awarii jednostki. Walidowane są zachowania na poziomie jednostek, które w danej sytuacji powinny dążyć do odwzorowania szyku liniowego i równomiernego wypełnienia formacji liniowej. Do sprawdzenia badanych zachowań wykorzystano komunikacje przewodem USB ze sterowników, umożliwiającą zbieranie logów z każdej z jednostek. Zebrane informacje zostały zarchiwizowane, aby posłużyć do prezentacji w tabeli 5.2. na koniec walidacji. Oczekiwanym jest, aby po wyłączeniu zasilania jednego ze sterowników, dwa sterowniki po lewej i prawej stronie zgłosiły „brak" obiektu, który wcześniej znajdował się między nimi, a sterownik lotu niebędący na brzegu formacji, starał się przywrócić szyk formacji, korygując dystanse między jednostkami. Jako wskaźnik korekty dodano log do oprogramowania ArduPilot, który zostanie wywołany, gdy któraś z jednostek chciałaby skorygować swoją pozycje.

Aby móc przetestować precyzyjnie badane zachowania, koniecznym było użycie 4 sterowników lotu. Rolę brzegową pełnią 2 z nich, określone zgodnie z przyjętą formacją liniową. Z pozostałych 2 sterowników lotu jeden

ulega awarii, natomiast drugi ma za zadanie dążyć do skorygowania szyku tylko na podstawie własnego mechanizmu autonomii. Formacje roju i sposób ich definiowania należą do zadań matki-operatora i zostały omówione w rozdziale 6. Na tym etapie lider nie ma łączności z matka-operatorem, aby zbadane zostały wyłącznie zachowania autonomiczne dla jednostek w grupie. Aby zwalidować eksperyment, przyjęto jako najlepsze, aby przetestować tylko sterowniki lotów bez osprzętu umożliwiającego im ruch w powietrzu. Dzięki temu możliwy jest precyzyjny pomiar dystansów między jednostkami, również w dużych odległościach wzajemnych. Z racji na to, że każdy sterownik lotu jest nieruchomy, możliwe było odniesienie do wartości dystansów wzajemnych, mierzonych co do centymetra. Dodatkowo możliwe było kontrolowane przeprowadzenie awarii sprzętowej jednej z jednostek oraz zebranie danych ze sterowników z wykorzystaniem logowania po USB, dodając własne logi niebędące częścią komunikacji po protokole wewnętrznym roju lub protokole MAVLink. Dzięki temu możliwe było wyeliminowanie możliwych opóźnień w logowaniu informacji służących do walidacji badania.



Zdj. 5.1. Widok 4 sterowników lotu przygotowanych do logowania danych [źródło własne]

Eksperyment został przeprowadzony dwukrotnie dla różnych odległości między poszczególnymi jednostkami. Wszystkie sterowniki lotu są w zakresie fizycznego sprzętu inne, natomiast do każdego z nich wgrano oprogramowanie ArduPilot w wersji Copter 4.4.4. wraz z dodaną implementacją autorskiego protokołu do komunikacji wewnętrznej. Dla pierwszej próby przewidziano odległości wzajemne między sterownikami lotu w postaci 50 cm, tak jak przedstawiono na zdjęciu 5.1. Kolejną próbę wykonano dla odległości między poszczególnymi jednostkami wynoszącej 10 m. Do logowania danych użyto 4 komputerów, każdy do zbierania logów z jednej jednostki. Dane zbierano wraz z oznaczeniem czasu, kiedy została zapisana kolejna dana przy użyciu programu HTerm 0.8.5. Następnie zebrano dane ze wszystkich komputerów, zestawiając logi zgodnie z czasem w jakim zostały odebrane. Wyniki przedstawiono w tabelach poniżej (5.2 i 5.3). Czas odebrania logów mierzono od momentu rozpoczęcia eksperymentu. Pozycja lokalna wyznaczona przez rój została przekazana w logach w metrach.

W tabeli 5.2. przedstawiono logi zebrane podczas eksperymentu przeprowadzonego przy zmierzonym rozstawie dronów wynoszącym 50 cm. Różnica między poszczególnymi czasami logów w jednym wierszu nie przekroczyła 0.050 sekundy. Treść logów została wgrana do sterownika lotu, tak aby była krótka i jednoznacznie informowała o wydarzeniach, które mają miejsce w sterowniku. Jako obiekt lidera został zainicjowany sterownik nr 2, ponieważ był w środku formacji i zawierał odbiornik GPS. Jego pozycja lokalna zgodnie z założeniem ma zawsze wynosić zero, ponieważ jest oznaczeniem początku lokalnego układu współrzędnych. W logach przyjęto, aby przekazywać pozycje w osiach X, Y. Pierwsza część eksperymentu przebiegła zgodnie z oczekiwaniami.

Tab. 5.2. Zestawienie logów zbieranych ze sterowników lotu w przypadku próby z dystansem wzajemnym jednostek wynoszącym 50 cm [źródło własne]

Czas odebrania logów [s]	**Sterownik lotu nr 1**	**Sterownik lotu nr 2**	**Sterownik lotu nr 3**	**Sterownik lotu nr 4**
0.021	Włączenie UAV	Włączenie UAV	Włączenie UAV	Włączenie UAV
1.894	Inicjalizacja lidera i układu lokalnego	Inicjalizacja lidera i układu lokalnego	Inicjalizacja lidera i układu lokalnego	Inicjalizacja lidera i układu lokalnego
4.534	Inicjalizacja zakończona	Inicjalizacja zakończona – obiekt lidera	Inicjalizacja zakończona	Inicjalizacja zakończona
4.593	Pozycja lokalna [0.00, -0.51]	Pozycja lokalna [0.00, 0.00]	Pozycja lokalna [0.00, 0.50]	Pozycja lokalna [0.05, 1.00]
Wyłączenie zasilania sterownika lotu nr 3				
24.453	Brak logu	Dysonans odległości wzajemnych, wymagana korekta do prawej		Dysonans odległości wzajemnych, dron brzegowy, korekta niemożliwa

Druga część eksperymentu, polegająca na sprawdzeniu reakcji jednostek w roju w przypadku wyłączenia zasilania dla sterownika lotu nr 3, również zakończyła się sukcesem. Sterownik lotu nr 2 wskazał na konieczność korekty. Musiałby ją zrealizować przez ruch w kierunku

sterownika lotu nr 4, który nie może podjąć się korekty pozycji, ze względu na to, że jest jednostką brzegową i jego zadanie utrzymania szyku sprowadza się do utrzymania zadanej szerokości formacji.

W tabeli 5.3. przedstawiono logi zebrane podczas eksperymentu wykonane dla rozstawu dronów wynoszącego 10m. Podczas ponowienia eksperymentu dla dalszych odległości odebrano niemal identyczne logi, świadczące o tym, że jednostki w roju na poziomie własnej autonomii ponownie zainicjowały korekcje odbiorników GPS i wyznaczyły lokalny układ współrzędnych oraz poprawnie zareagowały na brak zasilania symulujący awarie jednej z jednostek.

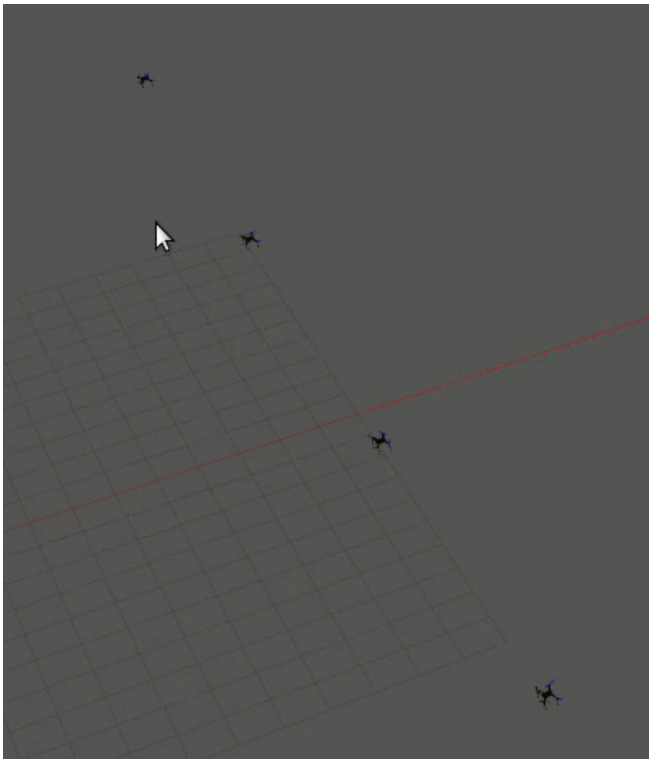
Tab. 5.3. Zestawienie logów zbieranych ze sterowników lotu w przypadku próby z dystansem wzajemnym jednostek wynoszącym 10 m [źródło własne]

Czas odebrania logów [s]	**Sterownik lotu nr 1**	**Sterownik lotu nr 2**	**Sterownik lotu nr 3**	**Sterownik lotu nr 4**
0.065	Włączenie UAV	Włączenie UAV	Włączenie UAV	Włączenie UAV
2.464	Inicjalizacja lidera i układu lokalnego	Inicjalizacja lidera i układu lokalnego	Inicjalizacja lidera i układu lokalnego	Inicjalizacja lidera i układu lokalnego
6.252	Inicjalizacja zakończona	Inicjalizacja zakończona – obiekt lidera	Inicjalizacja zakończona	Inicjalizacja zakończona
6.865	Pozycja lokalna [0.00, -10.05]	Pozycja lokalna [0.00, 0.00]	Pozycja lokalna [0.05, 9.95]	Pozycja lokalna [-0.05, 20.20]
Wyłączenie zasilania sterownika lotu nr 3				

22.534	Brak logu	Dysonans odległości wzajemnych, wymagana korekta do prawej		Dysonans odległości wzajemnych, dron brzegowy, korekta niemożliwa

5.4.2. Walidacja komunikacji wewnątrz roju z użyciem opracowanego środowiska graficznego

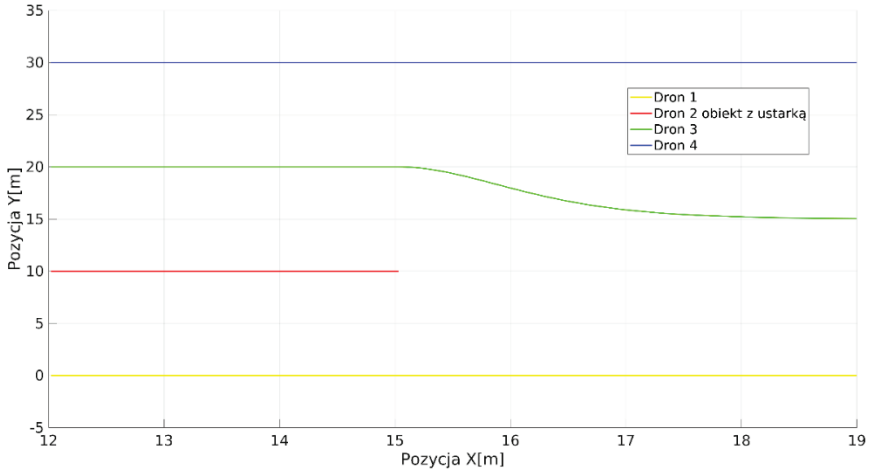
Zgodnie z krokami przedstawiony na początku podrozdziału 5.4. umożliwiającymi użycie komunikacji wewnętrznej roju w programie Gazebo, przygotowano eksperyment przelotu 4 dronów w środowisku graficznym.



Rys. 5.10. Pozycja startowa dronów przed rozpoczęciem przelotu w środowisku graficznym [źródło własne]

Na rysunku 5.10. przedstawiono rozmieszczenie dronów w szyku liniowym, który został zadany jako formacja startowa. Analogicznie do wcześniej przeprowadzonego eksperymentu przyjęto rozstaw dronów 10 m. Na początku symulacji następuje inicjalizacja układu lokalnego oraz korekcja odbiorników GPS, tak samo jak w przypadku eksperymentu przeprowadzonego na fizycznych sterownikach lotu. W losowym momencie czasu jedna z jednostek ulega awarii. Zadaniem stojącym przed pozostałymi jednostkami jest odwzorowanie szyku na podstawie mechanizmu opisanego w podrozdziale 5.3.3.

Na rysunku 5.11. przedstawiono trajektorie otrzymane z przelotu bezzałogowych obiektów latających wraz z kompensacją szyku. Podczas przelotu zasymulowane zostało wyłączenie sterownika lotu drona nr 2. UAV nr 3 poprawnie skorygował szyk, dokonując proporcjonalnego przybliżania się w kierunku drona nr 1.



Rys. 5.11. Trajektorie dronów podczas eksperymentu awarii jednej z jednostek podczas przelotu [źródło własne]

5.5. Podsumowanie rozdziału

W rozdziale omówiona została przygotowana koncepcja komunikacji. Dla wymiany danych z matką roju wykorzystano popularny dla bezzałogowych obiektów protokół MAVLink. Dla komunikacji wzajemnej wewnątrz roju przygotowano autorski protokół. Na najniższej warstwie komunikacji w warstwie fizycznej wykorzystywana została technologia UWB do wymiany danych o pozycji oraz o danych z obiektów i potencjalnych awariach z użyciem autorskiego, nieszyfrowanego i nieobciążającego protokołu. Dzięki innowacyjnej implementacji możliwe jest zachowanie autonomii poszczególnych jednostek, co może być szczególnie przydatne w pracy w warunkach, gdzie mogą wystąpić przesłonięcia sygnałów dalekiego zasięgu i utrata komunikacji z matką roju. Gwarantuje to bezkolizyjność bezzałogowych obiektów latających, jak i również rozmieszczenie dronów w równych wzajemnych odległościach, co jest najczęściej pożądanym układem podczas wykonywania różnych typów misji jak np. patrolowanie, monitorowanie, eksploracja, mapowanie przestrzeni, zwiad terenu. Przeanalizowane zostało użycie odbiorników GPS w kontekście pracy roju. Omówiono wady koncepcji opartych jedynie na zastosowaniu tego typu nawigacji. Przedstawiona została autorska koncepcja wykorzystania inercyjnych czujników, znajdujących się na pokładzie każdego sterownika lotu, modułów UWB oraz minimalnej ilości odbiorników GPS.

Poza autonomicznością poszczególnych jednostek, omówiono wymagania, aby podrój mógł funkcjonować autonomicznie w przypadku braku komunikacji z matką. Na tym poziomie ustalany jest lokalny układ odniesienia dla podroju, którego punktem centralnym jest jedna z jednostek będąca liderem. Pozycje pozostałych jednostek są oznaczane w tym układzie. Wzajemne informacje o pozycjach poszczególnych jednostek przekazywane są przez dedykowany autorki protokół, używany na warstwie fizycznej UWB. Dzięki temu komplet danych odnośnie każdego drona w podroju wraz z jego poszczególnym ID jest przekazywany w pakiecie danych, a poszczególne dane są przechowywane przez lidera. Wyznaczono również minimalną ilość jednostek w podroju, które stanowią kotwice ruchome dla nawigacji UWB. Omówiono jak jednostki podroju opracowują jeden układ odniesienia,

korzystając z dostępnych czujników. Dzięki temu możliwe jest również zlokalizowanie jednostki w tym układzie i zorientowanie względem całości roju na późniejszym etapie przez matkę.

Przeprowadzono doświadczenie z różnymi sterownikami lotu, które dowiodło skuteczności koncepcji komunikacji i nawigacji obiektów w podroju. Następnie zaimplementowano opracowaną komunikację wraz z autorskim protokołem do środowiska symulacyjnego, gdzie przeprowadzono korespondujący eksperyment, obrazujący wzajemną komunikację, nawigację i zachowanie w przypadku awarii jednego z dronów. Wyniki potwierdzają zasadność zastosowania komunikacji wzajemnej między poszczególnymi obiektami podroju. Dzięki opracowanym koncepcjom możliwe jest sterowanie całym rojem lub podrojem jak pojedynczą jednostką. W tym kontekście sterowanym jest lider, a pozostałe jednostki mogą, dzięki mechanizmom komunikacyjnym opracowanym w tym rozdziale, przemieszczać się wraz z liderem, wzajemnie korygując swój szyk.

Wszystkie eksperymenty zostały przeprowadzone pomyślnie. Co potwierdza możliwość implementacji autorskiego protokołu zarówno przy użyciu obiektów rzeczywistych, jak i w opracowanym w poprzednim rozdziale środowisku symulacyjnym. Przygotowana komunikacja stanowi podstawę do możliwości wdrożenia bardziej zaawansowanych mechanizmów sterowania rojem przez matkę-operatora.

Rozdział 6

MATKA-OPERATOR

W niniejszym rozdziale skupiono się na opracowaniu koncepcji oraz metodyce realizacji sterowania wieloma bezzałogowymi obiektami latającymi przez przygotowany program. Jako program należy rozumieć algorytm uruchamiany na komputerze lub obiekcie pełniącym rolę matki roju. Zgodnie z założeniem ma on umożliwiać autonomiczną pracę roju, a także umożliwiać operatorowi ingerencje w pracę jednostek. Jak zostało ustalone w celach pracy, algorytm nie jest dedykowany do pojedynczego sektora, jest elastyczny, modyfikowalny oraz wdrażalny dla wielu obszarów gdzie wykorzystywane są drony. W rozdziale przedstawiono strukturę formacji roju, koncepcje autorskiego oprogramowania sterującego. Uwagę poświęcono zagadnieniom związanym ze sztuczną inteligencją użytą, w zakresie uczenia maszynowego, do optymalizacji pracy roju. Moduł ten bazuje na wcześniej rozpatrywanych zagadnieniach środowiska graficznego oraz komunikacji.

Pracę kończą eksperymenty przeprowadzane dla grup 12 oraz 20 dronów. Zaimplementowano w nich typowe scenariusze misji wykonywanych przez drony, a także wskazano na zachowania właściwe rojowi i ich realizacje w opracowanym autorskim programie sterującym.

6.1. Struktura programu sterującego

Program sterujący został napisany w języku Python. Wybór tego języka został podyktowanym natywnym wsparciem dla biblioteki TensorFlow, która została wykorzystana do implementacji uczenia maszynowego dla roju. Użyto również bliotek do komunikacji poprzez ROS oraz MAVLink, a także bibliotek do komunikacji z bazą danych SQLite. Na drodze przygotowań pracy jako najlepsze rozwiązanie uznano możliwość użycia proxy protokołu MAVLink, który służy do komunikacji programu z jednostki posiadającymi moduły do komunikacji dalekodystansowej. W proponowanej koncepcji do wizualizacji pracy UAV służy program QGC.

Jego możliwości i użycie zostały opisane szczegółowo w podrozdziale 4.4. Należy podkreślić, że program ten umożliwia kompleksową diagnostykę bezzałogowych obiektów, a także może służyć do przejęcia kontroli nad jednostką w przypadku gdyby operator uznał za konieczne sterowanie manualne poszczególnych dronem. Przygotowana koncepcja może służyć do pracy z dronami będącymi pod kontrolą rozważanych w rozdziale 3 sterowników lotu ArduPilot i PX4. Oba te oprogramowania używają rozróżnienia trybów pracy drona i część trybów jest tożsama między nimi. Dla obiektów latającym można wyróżnić następujące tryby:

- Stabilize – Tryb ręczny, w którym dron stabilizuje swoją pozycję i wykonuje manewry na polecenia wydane z aparatury sterującej lub pilota. Użytkownik steruje mocą silników, co oznacza pełną kontrolę nad pozycją. Jest to tryb podstawowy, używany do ręcznego latania.
- AltHold – W tym trybie dron automatycznie utrzymuje swoją wysokość, ale pozycja w płaszczyźnie poziomej jest kontrolowana ręcznie. Wysokość drona jest utrzymywana na podstawie danych z barometru, co pozwala na bardziej precyzyjne utrzymywanie wysokości.
- Auto – Tryb autonomiczny, w którym dron wykonuje wcześniej zaprogramowaną misję opartą na punktach nawigacyjnych. Dron może automatycznie startować, lądować i przemieszczać się między punktami zgodnie z planem misji.
- RTL (*Return to Launch*) – Tryb powrotu do punktu startowego. Dron automatycznie wraca do miejsca, z którego wystartował, przy zachowaniu zadanej wysokości. Jest często używany w sytuacjach awaryjnych lub po zakończeniu misji.
- Land – Tryb automatycznego lądowania. Dron powoli opada i ląduje w miejscu, w którym został uruchomiony tryb. Używany jest na koniec misji lub w sytuacjach awaryjnych.
- Guided – Tryb półautonomiczny, w którym dron jest sterowany zdalnie z poziomu zewnętrznego programu lub skryptu komputerowego. Jest to tryb, w którym polecenia są wydawane poprzez MAVLink, czyli protokół komunikacyjny używany przez ArduPilot do komunikacji z komputerami zewnętrznymi.

Przygotowany program używa sterowania dronami w trybie Guided. Wymaga to stabilnej łączności. Wybór łącza tak aby zapewnić najlepsze warunki komunikacji został opisany w podrozdziale 3.1.1. Jeśli rój znajduje się w zasięgu, operator może wymusić ręcznie zmianę trybu pracy poszczególnych dronów i w razie potrzeby wymusić powrót do bazy, lądowanie awaryjne lub przejść do trybu Stabilize i przejąć kontrolę na dronem sterując nim z poziomu radia lub joysticku połączonego z komputerem i wysyłającego sygnały sterujące przez protokół MAVLink niezależnie od pracy programu sterowania w roju. Program sterujący traktuje drona, któremu tryb został zmieniony ręcznie przez operatora, jako drona przejętego – nie próbuję ponownie wymusić objęcia kontroli przez przejście w tryb Guided. Sterowane są pozostałe jednostki, które na poziomie własnej autonomii unikają zderzenia ze sterowanym ręcznie dronem poprzez mechanizm odczytywania odległości wzajemnej od drona niebędącego w roju, który wciąż wysyła swoje ramki informujące o jego pozycji, dzięki czemu jest identyfikowany przez pozostałe jednostki oraz mapowany przez matkę-operatora. Mechanizm wysyłania ramek informujących o pozycji drona, z użyciem autorskiego protokołu, przez który drony rozgłaszają swoją pozycje został opisany w 5 rozdziale. Implementacja tego mechanizmu leży w priorytetowej warstwie autonomii, która zakłada że każda jednostka, niezależnie od komend otrzymanych od programu sterującego dba aby nie doprowadzić do kolizji z innymi jednostkami. W przypadku przejęcia, któregoś z dronów przez operatora pozostałe jednostki pozostające w trybie Guided i będące sterowane przez program będą odsuwać się od przejętego drona gdyby groziła im kolizja. W przypadku przejęcia przez operatora jednostki będącej liderem roju lub podroju wybierany jest następny lider, który pełni rolę łącznika komunikującego drony w podroju z programem sterującym.

Opisana koncepcja programu sterującego nosi nazwę matki-operatora, a drugi człon dodany jest ze względu na możliwość ingerencji operatora, na wzór koncepcji opracowanej przez organizacje DARPA opisanej w rozdziale 1.

Wydawanie poleceń z poziomu programu sterującego dla roju odbywa, zgodnie warunkiem kompatybilności modułów, za pośrednictwem

protokołu MAVLink. Istnieją 3 implementacje bibliotek umożliwiające wysyłanie wiadomości w tym protokole dla języka Python. Są to: ROS, Pymavlink [84] i DroneKit [70]. Pakiet ROS, nazywany również metasystemem ze względu na mnogość operacji jakie wykonuje i możliwości jakie daje, został opisany w rozdziale 4.2. Mostem połączeniowym dla matki-operatora jest moduł metasystemu ROS o nazwie MAVROS. Jego implementacja opiera się na użyciu biblioteki Pymavlink. Pymavlink jest nisko poziomową implementacją, która zawiera podstawowe komendy i operacje dostępne w protokole MAVLink. Nie obejmuje żadnej algorytmiki, a jedynie enkapsulacje i odczytywanie ramek protokołu. W przypadku chęci modyfikacji tego modułu należy też zaznaczyć, że wspomniane ograniczenie kompatybilności z pakietem ROS dotyczy również użycia środowiska graficznego.
Z racji na to, że moduł MAVROS implementuje wewnątrz własnego kodu bibliotekę Pymavlink, wspomniane ograniczenie nie dotyczy tej biblioteki, a jej funkcjonalności można wykorzystać w programie sterującym. Niesie to ze sobą korzyść w postaci możliwości uproszczenia pewnych komend i wysyłania ich bezpośrednio, co przyśpiesza proces komunikacji i upraszcza implementacje algorytmu sterowania.

Oprogramowania sterowników lotu rozważane w pracy, ArduPilot i PX4, odbierając dane z protokołu MAVLink, obsługują dwie komendy za pomocą których można zadać UAV żądaną trajektorię. Komendy to:

- SET_POSITION_TARGET_GLOBAL_INT
- MISSION_ITEM_INT

Pierwsza z wymienionych komend zawiera współrzędne punktu, który ma osiągnąć dron. Druga zawiera serie punktów, na podstawie których sterownik lotu wyznacza trajektorie opartą o lot po łuku. Zgodnie z założeniami i koncepcją zarządzania rojem na poziomie autonomii roju, omówionymi w rozdziale 5, trajektoria może być wysyłana albo tylko dla jednostki będącej liderem albo dla wszystkich jednostek. W pierwszym przypadku lider jest kierowany, pozostałe drony utrzymują zadany szyk i podążają za liderem. Trajektoria lidera, odbierana od matki-operatora jest przez niego rozgłaszana dla pozostałych dronów w podroju, i traktowana jest

przez sterowniki lotu na tych samych warunkach jak gdyby otrzymały tę komendę przez protokół MAVLink.

Istnieją przypadki, w których komendy w protokole MAVLink [85] okazały się niewystarczające do przekazania wszystkich informacji od programu sterującego do roju. Jest to podyktowane tym, że protokół MAVLink został przygotowany, aby być nieobciążającym w obsłudze protokołem do sterowania i zarządzania pojedynczymi jednostkami. Aby przystosować go do pracy z rojem wymagane było dodanie kilku komend między innymi komend umożliwiających adresowanie informacji pośrednio do jednostek nie będących liderami. Dzięki temu, w specjalnych przypadkach, można zadać trajektorię dla poszczególnego obiektu. Przez dodanie komendy należy rozumieć implementacje własnego polecenia po stronie zarówno biblioteki Pymavlink, a także po stronie oprogramowania ArduPilot. Proces dodania własnej komendy, po obu stronach, został szczegółowo rozpisany w dokumentacji oprogramowania ArduPilot [66].
Program sterujący został napisany przy użyciu paradygmatu programowania obiektowego. Można wyróżnić 4 obiekty:

1. Środowisko
2. Matka
3. Rój
4. UAV

Środowisko jest obiektem nadrzędnym i możliwe jest utworzenie tylko jednej instancji tego obiektu. Może reprezentować rzeczywiste drony jak i również może odwoływać się do przygotowanego środowiska graficznego. Struktura komend, poszczególnych oprogramowań i metod komunikacji jest taka sama dla środowiska graficznego jak i fizycznych dronów kontrolowanych przez sterowniki lotu z wgraną przygotowaną wersją oprogramowania uwzględniającą dodany protokół do komunikacji wewnętrznej oraz dodane komendy do protokołu MAVLink. Na etapie utworzenia instancji obiektu środowiska wymagane jest zdefiniowanie adresów poszczególnych UAV, a także przekazanie obiektu roju z którym zainicjowane ma zostać środowisko. W tej klasie przechowywane są informacje zbierane od poszczególnych jednostek służące do mapowania

terenu. Również wszystkie pozostałe dane mogące posłużyć do sprecyzowania warunków w jakich znajdują się drony. Przechowywana jest również informacja o ilości podrojów, czyli egzemplarzach klasy rój. Lista zadań zdefiniowana przez użytkownika przechowywana jest na tym poziomie.

Do danych zawartych w środowisku ma dostęp jedynie instancja klasy matki. Możliwa do utworzenia jest tylko jedna instancja i jest ona tworzona zawsze na początku uruchamiania programu. Obiekt klasy matki zawiera szereg komend, którymi komunikuje się z liderami roju. Posiada również szereg mechanizmów mających na celu optymalizacje pracy. Uruchamiane są w nim procesy odpowiadające za inteligencje roju, wykorzystujące uczenie maszynowe. Zostało to omówione precyzyjniej w podrozdziale 6.3.

Na początku pracy programu tworzony jest jeden obiekt roju. W trakcie pracy roju istnieje możliwość utworzenia większej ilości instancji np. przez wymuszenie podziału roju przy napotkaniu przeszkody. Umownie w pracy przyjęto nazewnictwo, że jeśli jest więcej niż jedna instancja roju, każda z nich nazywana jest podrojem. Każdy egzemplarz klasy rój zawiera informacje odnośnie każdej poszczególnej jednostki, ze szczególnym uwzględnieniem jednostki lidera. Na tym poziome zawarte są także informacje o pozycji każdej jednostki należącej do grupy, zadanie powierzone grupie, planowana trajektoria dla lidera. Dodatkowo przechowywane są informacje o strukturze roju. Należy przez to rozumieć między innymi zadany kształt roju, wymuszone odstępy między jednostkami. Dostępne struktury roju zostały omówione w podrozdziale 6.2.

Klasa UAV reprezentuje poszczególne bezzałogowe obiekty latające. Jej struktura jest w większości zbieżna ze strukturą oprogramowania sterownika ArduPIlot. Większość pól i metod jest taka sama, a ich wystąpienie tej klasie służy uporządkowaniu kodu. Jako poszczególne metody np. zadanie żądanej pozycji wykorzystywana jest koncepcja wzorca programowego adaptera. Wzorzec ten opiera się na dopasowaniu metod używanej biblioteki do potrzeb interfejsu pisanego programu. Zgodnie z przytoczonym przykładem zadania żądanej pozycji wywołanie metody obiektu UAV opiera się na wywołaniu funkcji z biblioteki Pymavlink, która za pomocą wybranej komendy przesyła informacje do drona. Jeśli dron jest

liderem przekazanie informacji odbywa się bezpośrednio przez użycie komendy protokołu MAVLink. Jeśli jednostka nie jest liderem używana jest inna, dodana autorska komenda do protokołu MAVLink, która zawiera informacje do jakiej jednostki lider ma przekazać wiadomość. Polami różniącymi autorską klasę UAV oraz klasę Copter, będącą nadrzędną klasą w oprogramowaniu ArduPilot, jest np. znacznik czy dany obiekt może być liderem, informacje o posiadanych czujnikach dodatkowych, a także poszczególny adres pod jaki można nawiązać komunikację z fizycznym dronem. To czy dany obiekt jest liderem czy nie, weryfikowane jest na poziomie klasy roju, ponieważ to nie dany obiekt decyduje samodzielnie o tym czy zostanie liderem. Również parametry określające dynamikę obiektu, jak np. maksymalna prędkość, kąty przechyły jak i typ jednostki są zawarte na tym poziomie.

Przepływ danych i procesu wydania komendy przedstawia się następująco:

1. Matka odczytując misje rozpoznaje teren na podstawie danych przechowywanych w obiekcie środowisko.
2. Po wyznaczaniu optymalnych trajektorii dla roju oraz komend specjalnych dla poszczególnych jednostek przekazuje dane do obiektu roju.
3. W każdym obiekcie roju przechowywana jest informacja odnośnie lidera. Przekazywana jest informacja do konkretnego obiektu, który jest liderem.
4. Obiekt klasy UAV będący liderem uruchamia funkcję do spełnienia żądanego polecenia. Zgodnie z tym czy dana lub komenda będzie adresowana do niego czy też do innej jednostki dopasowywane jest ciało funkcji, aby dobrać poszczególne komendy protokołu MAVLink.
5. Po wysłaniu komendy MAVLink ta trafia do jednostki lidera roju, który zarządza nią przekazując innej jednostce lub wykonując treść zawartą w ramce protokołu MAVLink.

Dane o pozycji poszczególnych jednostek, a także nieustanne odpytywanie jednostek o sygnał *heartbeat* (uderzenie serca), które identyfikuje obecność jednostki, są realizowane przez moduł MAVROS, który po nawiązaniu połączenia z dronem subskrybuje, czyli inaczej mówiąc

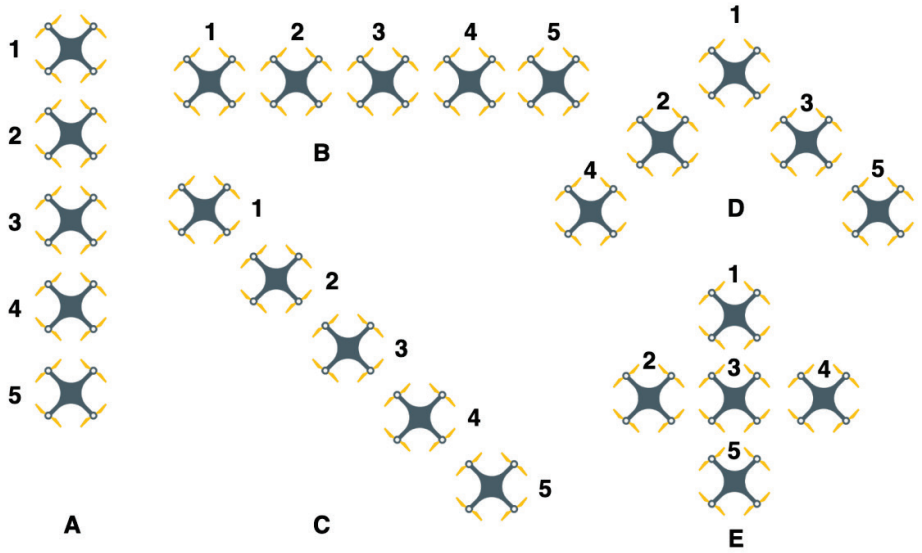
nieustannie odczytuje, wszystkie dane o jednostce. Dane te są mapowane przez poszczególne modułu pakietu ROS, a także są zbierane w obiekcie klasy środowisko. W każdej chwili może do nich odnieść się obiekt matki. W szczególnych przypadkach matka wykorzystując podany proces przesłania komendy może zapytać o żądaną informację i jeśli ta miała by tyczyć się środowiska nadpisać ją w obiekcie klasy środowisko.

Dzięki przyjętemu schematowi obiektów, możliwe jest precyzyjne diagnozowanie programu, z podglądem wszystkich obiektów. Zadania komunikacji na poziomie abstrakcji programowania takie jak żądanie wykonania ruchu przez matkę obejmuje abstrakcyjną warstwę oprogramowania, gdzie żądanie przekazywane jest do obiektu roju a następnie do odpowiedniego obiektu lidera. Nie zabiera to zasobów systemowych w istotnym stopniu a daje możliwość prześledzenia komunikacji na poziomie programu. Również zmiany parametrów poszczególnych obiektów dają szerokie możliwości diagnostyczne. Procesy wykonywane w bezzałogowych obiektach latających można śledzić na poziomie programu.

6.2. Struktura roju

Jednym z omówionych w rozdziale 1 warunków niezbędnych aby uznać wiele jednostek latających za rój jest utrzymywanie zadanej im formacji. Jak przedstawiono, we wspomnianym rozdziale badacze analizują mechanizmy oparte o zachowania zwierząt, wzorowane na naturze. Nie można wytypować jednej, najlepszej dla każdego zadania lub specyfiki uczestników roju formacji. Tym bardziej w kontekście koncepcji kierowanej do wielu sektorów należało przyjąć, że struktura roju może być zmienna, dobrana do potrzeb danego zadania i możliwości dynamicznych obiektów. Rozpatrywane w pracy jako przykładowe obiekty typu quadrocopter są bardzo dobrze sterowalne i posiadają możliwość zawisu w powietrzu co czyni je uniwersalnymi. Ich konstrukcja pozwala przyjmować różne formacje i reagować na zmiany w otoczeniu przez dynamiczną zmianę trajektorii w dowolnym kierunku.

Badacze rojów dronów przedstawili w swojej pracy [11] niektóre możliwe koncepcje szyku roju (rys. 6.1.).



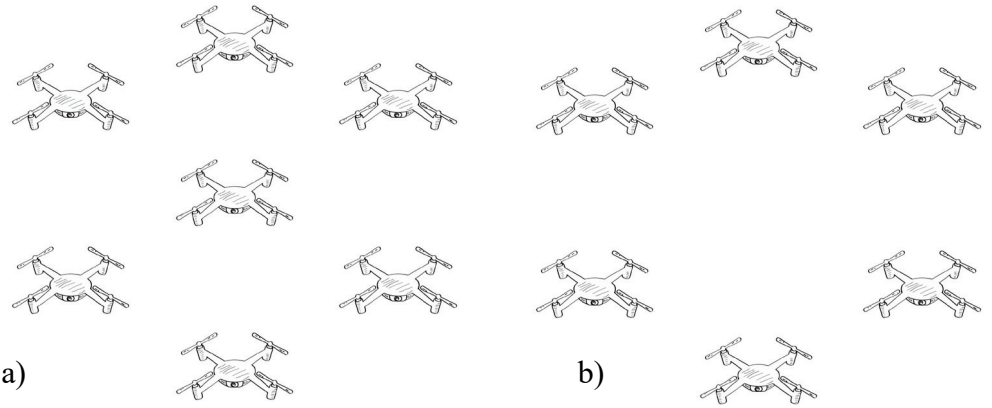
Rys 6.1. Kształty formacji roju, źródło [11]

Można wyróżnić następujące formacje:
A - Kolumna,
B - Szyk liniowy,
C – Formacja eszelon znana z wojska,
D - V-kształt na wzór klucza ptaków,
E - Ułożenie w diament.

Przedstawione koncepcje formacji roju, rozważane były pod kątem teoretycznym, a badacze nie wskazali możliwych aplikacji danych szyków w sektorach gdzie używane są drony. Najbardziej uniwersalną i wdrażalną koncepcją z wyżej przedstawionych, jest koncepcja szyku liniowego. Może być ona używana do monitorowania, skanowania, lub też wykonywania operacji jak np. gaszenia danego terenu. Jest też również dobrze skalowalna dla większej ilości jednostek. Pozostałe koncepcje mają znacząco mniejsze możliwością wdrożenia tak jak szyk kolumnowy i eszelon. Formacja o kształcie zbliżonym do litery V, w innych pracach nazywana również formacją trójkątną [201] nie wykazuje większego potencjału niż szyk liniowy [201]. Również formacja diamentu, może zostać zastąpiona przez inny kształt, być bardziej skalowalną i wdrażalną.

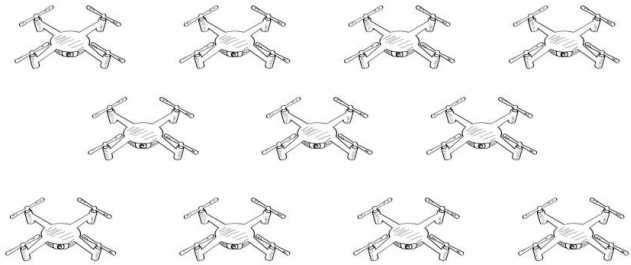
Opracowana koncepcja sterowania w roju jest elastyczna i nie zawęża liczby dronów jakie mogą zostać użyte w roju, choć koncentruje się koncepcjach dla większej ilości bezzałogowych obiektów latających. Dodatkowo na etapie prac rozważono wiele potencjalnych formacji o różnych źródłach inspiracji, w tym również kierując się naturą [128], ale przede wszystkim skupiając się na możliwości aplikacji w różnych sektorach.

Formacją, od której zaczęto badania, była formacja plastra miodu przestawiona na rysunku 6.2.



Rys 6.2. Formacja plastra miodu z obiektem centralnym oraz bez niego [źródło własne]

W wariancie oznaczonym literą „a” w środku umownego plastra znajduje się kolejny bezzałogowy obiekt latający, natomiast w wariancie „b” drony tworzą sześciokąt równoramienny bez jednostki znajdującej się w środku. Umieszczenie dodatkowego obiektu w środku jest zasadne, szczególnie w przypadku koncepcji topologii komunikacji z liderem grupy. Warto zauważyć, że koncepcja przedstawiona na rysunku 6.2.a może być uznana za formacje wielorzędową szyku liniowego.

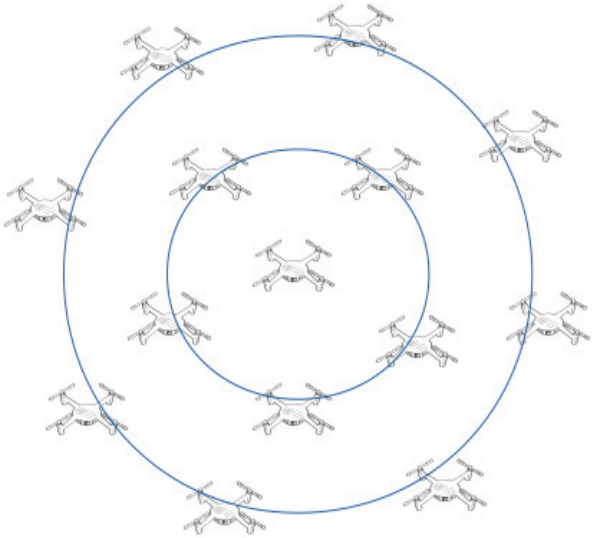


Rys 6.3. Szyk wielorzędowy [źródło własne]

Na rysunku 6.3 przedstawiono szyku wielorzędowy. Uzyskano go przez dodanie 4 jednostek na rogach rysunku. Jest to uniwersalny, skalowalny szyk, który jest łatwy do zadania, prosty i jednoznaczny w ewaluacji a także skuteczny dla zadań w wielu sektorach. Jest on przedstawiony w postaci dwuwymiarowej, natomiast może też być z powodzeniem implementowany do formacji w przestrzeni trójwymiarowej.

Zależnie od przyjętych parametrów formacja wielorzędowa, może przypominać szyk liniowy będąc formacją jednorzędową lub np. dwurzędową. Są to najczęściej stosowane formacje, o najszerszym spektrum aplikacji [129], [209]. W przypadku konieczności podziału roju na podgrupy daje najlepsze możliwości do przeprowadzenia szybkiego i skutecznego procesu przegrupowania. W przypadku awarii lub strącenia jednej z jednostek wymusza małe przesunięcia wśród niemal wszystkich obiektów.

Wadą tego szyku może być konieczność przesuwania jednostki lidera, co w pewnych misjach, szczególnie statycznych o niewielkim wymaganym przelocie roju, może być newralgiczne lub może narażać lidera na strącenie. Proponowanym autorskim szykiem jest szyk nazwany „orbitującym”, gdzie drony ustawiają się wokół lidera tworząc pierścienie (rys. 6.4.).



Rys 6.4. Szyk orbitujący [źródło własne]

Założeniem szyku orbitującego jest, aby jednostka umiejscowiona w centrum była traktowana priorytetowo i zawsze otaczana. W przypadku awarii jednego z obiektów na poszczególnym pierścieniu (orbicie) pozostałe drony, znajdujące się na tym pierścieniu, mają odwzorować szyk uzupełniając dystanse.

Ta formacja wymaga zarządzania przez matkę-operatora ponieważ informacje pochodzące z komunikacji wzajemnej między bezzałogowymi obiektami latającymi są niewystarczające, aby w przypadku wystąpienia zdarzenia które zakłóciłoby szyk drony mogły same go odwzorować na podstawie danych o tym który obiekt jest w pobliżu. Autonomia jednostki pozwala korygować pozycje względem sąsiadujących dronów. Jest to zbieżne jedynie z korekcjami w szyku wielorzędowym lub jednorzędowym. Uzupełnienie formacji może odbyć się bez komunikacji z matką-operatorem i daje najlepsze możliwości do bezkolizyjnego i optymalnego wykonania misji. Ze względu na przedstawione argumenty związane z bezpieczeństwem, autonomiczną strukturą sterowania oraz wspomnianymi wcześniej możliwościami aplikacji szyk wielorzędowy, a szczególnie jego podtyp, szyk

jednorzędowy, jest najlepszym wyborem i to dla tego szyku przygotowane zostały badania w dalszej części tego rozdziału.

6.3. Inteligencja roju

Inteligencja roju w opracowanej koncepcji jest zgodna z poziomami autonomii opisanymi w poprzednim rozdziale. Zadaniem matki roju jest przede wszystkim inteligentne planowanie trajektorii roju lub podrojów, tak aby wykonać zlecone misje. Misje, jak zostało napisane w podrozdziale 6.1, przechowywane są w obiekcie środowiska. Zadanie szczegółowych parametrów misji dokonuje się przez wypełnienie przygotowanego na potrzeby pracy szablonu, który ma strukturę pliku o formacie XML. Szablon znajduje się w załączniku nr 3. Zawiera parametry misji oraz bazowe informacje na temat roju. Użytkownik programu sterującego musi wypełnić zadany plik przed uruchomieniem modułu matki-operatora. Autorski parser, czyli fragment programu odpowiedzialny za odczytanie danych z pliku XML i umiejscowienie ich we właściwe miejsce w programie, wykonuje się wraz z uruchomieniem programu sterującego inicjalizując obiekty tworzone na początku pracy modułu.

Do wdrożenia inteligencji roju wykorzystano bibliotekę TensorFlow [92]. TensorFlow to otwarto-źródłowa biblioteka do uczenia maszynowego opracowana przez Google, której pierwsza wersja została udostępniona w 2015 roku. Architektura TensorFlow umożliwia uruchamianie modeli na różnych platformach, takich jak procesory CPU, karty graficzne GPU czy specjalistyczne jednostki TPU, co zapewnia elastyczność i skalowalność. Kluczową cechą TensorFlow jest wykorzystanie tzw. grafów obliczeniowych, w których węzły reprezentują operacje matematyczne, a krawędzie – przepływ danych w postaci tensorycznej. To podejście umożliwia optymalizację wydajności, szczególnie w obliczeniach rozproszonych. TensorFlow obsługuje szeroką gamę algorytmów, w tym sieci neuronowe, regresję liniową, analizę skupień, co czyni go uniwersalnym narzędziem dla różnych problemów analitycznych i predykcyjnych. TensorFlow stał się jednym z najpopularniejszych narzędzi do rozwoju systemów opartych na sztucznej inteligencji. Stał się również narzędziem wykorzystywanym w wielu dziedzinach nauki, głównie medycznych [1],

[147], ze względu na znaczące podobieństwa procesów uczących implementowanych w bibliotece w stosunku do pracy mózgu [50]. Wśród prac badawczych nad bezzałogowymi obiektami latającymi nie jest jeszcze szeroko wykorzystywany, choć pojawiają się pojedyncze pracę z użyciem tej biblioteki [191], [57], to jednak nie przedstawiają jej wykorzystania w kontekście roju.

Zastosowanie uczenia maszynowego w kontekście roju może wnieść znaczące rezultaty w zakresie poprawy jakości dobierania trajektorii oraz zarządzania rojem i procesami jakie mają miejsce wewnątrz niego. Algorytmy uczenia maszynowego zostały dokładnie opisane w wielu monografiach [179], [105]. Można podzielić je na trzy główne kategorie: uczenie nadzorowane, uczenie nienadzorowane oraz uczenie ze wzmocnieniem.

Uczenie nadzorowane polega na trenowaniu modelu na zbiorze danych, który zawiera zarówno wejścia, jak i przypisane do nich wyniki. Popularne algorytmy to: regresja liniowa, drzewa decyzyjne i las losowy, maszyny wektorów nośnych, sieci neuronowe.

Uczenie nienadzorowane działa na danych bez możliwości walidacji przez zewnętrzny proces. Przykładowe algorytmy to: algorytm K-średnich, analiza głównych składowych, algorytmy asocjacyjne.

Uczenie ze wzmocnieniem (reinforcement learning, RL) to metoda, w której agent uczy się poprzez interakcję z otoczeniem, otrzymując nagrody za poprawne działania i kary za błędne. Algorytmy RL są szczególnie skuteczne w problemach dynamicznych, takich jak gry czy sterowanie robotami. Popularnym algorytmem jest Q-learning, który umożliwia agentowi uczenie się optymalnej strategii postępowania.

Uczenie ze wzmocnieniem ma duży potencjał w zastosowaniach dla roju dronów, szczególnie w misjach, w których autonomiczne drony muszą współpracować, podejmować decyzje w dynamicznym środowisku i dostosowywać się do zmieniających się warunków. W przypadku roju dronów, RL może być użyteczne, aby trenować drony w optymalnym wykonywaniu zadań takich jak monitorowanie terenu, śledzenie obiektów, patrolowanie, czy inne misje zespołowe, w których ważna jest efektywność całego roju, a nie tylko pojedynczych jednostek.

Aby wykorzystać wspomniany algorytm należy zaadaptować warunki pracy roju dronów do wymogów algorytmu. Standardowo w RL stosowane są pojęcia takie jak: Agent, Środowisko, Stany, Akcje, Nagroda, Polityka, Funkcja wartości.

Jak można wnioskować z opisu terminologii związanej z uczeniem przez wzmacnianie, efekt finalny wykazuje podobieństwa do optymalizacji. Określenie „optymalny" używane w odniesieniu do roju i jego pracy na dalszych etapach pracy może zostać użyte w kontekście efektu uczenia, ale nie należy traktować go jako synonim słowa „wyuczony". Różnica w procesie, który klasycznie określa się mianem optymalizacji, a procesem uczenia przez wzmacnianie polega na tym, że:

1. Uczenie przez wzmacnianie skupia się na uczeniu agenta sekwencyjnych decyzji w dynamicznym środowisku, gdzie podejmowane akcje wpływają na przyszłe stany, a agent uczy się na podstawie nagród i interakcji z otoczeniem.
2. Optymalizacja odnosi się do znajdowania najlepszego rozwiązania dla określonego problemu, często w jednym kroku, bez konieczności interakcji z dynamicznym środowiskiem.

W uczeniu przez wzmocnienie optymalizowana jest polityka w kontekście sekwencji decyzji, natomiast klasyczna optymalizacja opiera się na statycznym problemie. Kroki wykorzystywane w uczeniu przez wzmocnienie zostały przedstawione w załączniku nr 4.

Uczenie przez wzmocnienie może odbywać się za pomocą różnych algorytmów, których dobór powinien być podyktowany możliwościami dostosowania algorytmów do uzyskania optymalnych wyników. Implementacja i wybór algorytmów dzięki użyciu biblioteki TensorFlow jest sprowadzana do wyboru parametru funkcji definiującego jaki algorytm powinien zostać użyty. W pracy przyjęto, że moduł matki-operatora może pracować z dwoma algorytmami, które w początkowym etapie prac dały najlepsze wyniki i dalsze prace były kontynuowane ich przy użyciu.

1. Q-Learning to jeden z popularnych algorytmów RL, w którym agent uczy się wartości akcji bez potrzeby znajomości modelu środowiska. Aktualizacja funkcji wartości akcji jest dana przez:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a) + \gamma \max(a') Q(s',a') - Q(s,a)] \quad (6.1)$$

gdzie:

α - współczynnik uczenia (learning rate).

γ - współczynnik dyskontowania.

$R(s,a)$ - nagroda za akcję a w stanie s.

$\max(a') Q(s',a')$ - maksymalna oczekiwana wartość przyszłych nagród w stanie s′.

2. Policy Gradient Methods czyli algorytmy gradientu polityki. Uczą bezpośrednio optymalnej polityki poprzez maksymalizację oczekiwanej sumy nagród. W tych algorytmach, polityka $\pi_\theta(a \mid s)$jest parametryzowana przez θ, a optymalizacja odbywa się przy pomocy gradientu polityki:

$$\nabla J(\theta) = E_\pi[\nabla_\theta \log \pi_\theta(a \mid s) Q_\pi(s,a)] \quad (6.2)$$

gdzie:

$J(\theta)$ - oczekiwana zdyskontowana suma nagród przy polityce πθ.

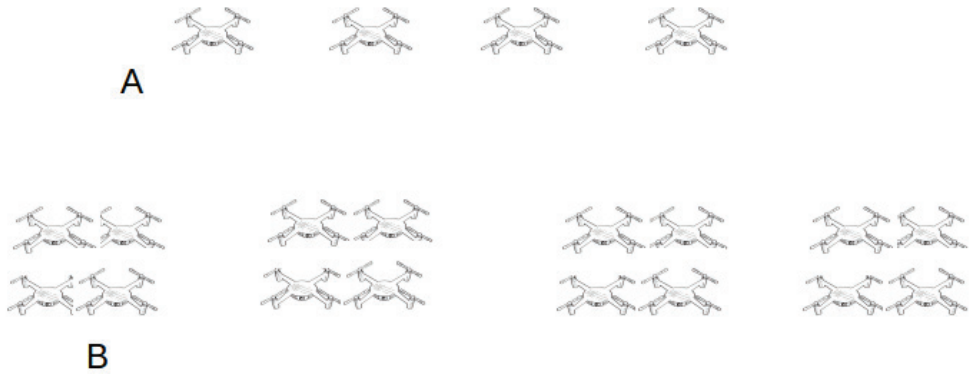
Przyjęto, że domyślnym algorytmem jest algorytm Q-Learning, ponieważ zakładane jest zastosowanie roju w środowiskach, które nie są z góry zdefiniowane i mogą wymagać dynamicznej reakcji roju na panujące warunki. Ważnym w przypadku tego algortymu jest uwzględnienie zjawiska nadmiernego dopasowania nazywanego w literaturze naukowej overfitting [206], [124]. Overfitting to zjawisko w uczeniu maszynowym, w którym model osiąga bardzo wysoką skuteczność na danych treningowych, ale jego zdolność do reakcji na nowe, nieanalizowane wcześniej dane jest znacząco ograniczona. Dzieje się tak, gdy model zbyt dokładnie dopasowuje się do specyficznych cech lub szumu w danych treningowych, przez co nie odzwierciedla rzeczywistych, ogólnych zależności w danych.

W kontekście uczenia wzmacniającego z użyciem algorytmów takich jak Q-learning, overfitting może prowadzić do sytuacji, w której agent uczy się strategii optymalnych tylko dla ściśle określonych warunków, ignorując bardziej uniwersalne zasady. Taki agent będzie podejmował właściwe decyzje w środowisku treningowym, ale w nowych, nieznacznie zmienionych sytuacjach może wykazywać słabe wyniki. Przyjętym, autorskim sposobem, wypracowanym podczas badań jest podział misji na podtypy, czego deklaracja ma miejsce w pliku misji. Wypracowane strategie są zapisywane do pliku bazodanowego, przy użyciu bazy danych SQLite. W sytuacji gdy misja ma ten sam typ jak wcześniej wykonywana, używane są dane wczytywane z bazy danych. Jeśli żądany typ misji ma inną specyfikę, wczytywany jest adekwatny zestaw informacji z poprzednich prób z bazy danych. W rezultacie możliwe jest dobieranie wyuczonych strategii precyzyjnie do typu misji, a także minimalizowana jest możliwość wystąpienia zjawiska nadmiernego dopasowania.

Koncepcja została zaprojektowana tak, aby można było uczyć rój w przygotowanym środowisku graficznym. Dzięki precyzyjnemu dopasowaniu środowiska, którego walidacje przeprowadzono w rozdziale 4, możliwe jest wielokrotne uczenie roju w bezpiecznych, bez kosztowych, w pełni monitorowanych warunkach. Jak wykazano w dalszej części rozdziału, zawierającej wyniki eksperymentów, możliwa jest poprawa uczenia strategii przez wykonywanie misji tego samego typu, z losowo zmieniającymi się warunkami. Losowa zmiana warunków, jest wymagana aby uniknąć wyżej omawianej sytuacji overfittingu. Zmiana warunków zewnętrznych, takich jak np. losowe rozmieszczenie przeszkód, w przypadku misji gdzie wymagane jest od dronów w roju utrzymanie szyku w terenie gdzie mogą wystąpić przeszkody, pozwala nauczyć odpowiednich mechanizmów zachowań, które będą sprawdzały się w dynamicznych, nieznanych warunkach.

Uczenie maszynowe w postaci uczenia przez wzmocnienie ma zastosowanie w przygotowanej koncepcji na dwóch etapach. W przypadku roju, który nie jest podzielony na podroje, trajektoria misji wraz z wymogami tyczącymi się roju są adresowane do lidera i zadania w postaci np. ominięcia przeszkody są realizowane na poziomie autonomii roju, lub na poziomie

wyznaczenia trajektorii przez matkę-operatora. W tym przypadku rozważana w niniejszym podrozdziale inteligencja roju sprowadza się do wyuczonego korygowania lotu poszczególnych jednostek. W przypadku minimalistycznego roju zawierającego 4 bezzałogowe obiekty latające formacja przedstawiona została na rysunku 6.5. A.



Rys 6.5. Enkapsulacja roju w przypadku podziału roju na podroje [źródło własne]

Jedyną interakcją jest interakcja między uczestnikami roju – rys 6.5. A. Jako agenta należy traktować każdą jednostkę.

W przypadku bardziej zaawansowanego zadania wymagającego podziału na podroje rys 6.5. B, wewnątrz każdego roju następuje interakcja między poszczególnymi jednostkami. Kolejnym poziomem jest interakcja między podrojami. Również na tym poziomie wykorzystywane jest uczenie przez wzmocnienie. Jeśli przyjąć założenie, że każdy podrój można potraktować jako jedną autonomiczną jednostkę, grupę z rysunku 6.5.B można uznać jak rój na rysunku 6.5.A. Ma to miejsce w rozpatrywanej koncepcji. Wciąż aktywne są priorytetowe mechanizmy autonomii zapewniające każdej jednostce bezpieczeństwo przez unikanie wzajemnych kolizji, również między poszczególnymi podrojami. Możliwy jest podział podroi, tak aby wykonywały poszczególne, składowe misje, które sumarycznie stanowią główną misję całego podroju. Każdy podrój może być sterowany niezależnie. Na tym poziomie uczenie ma zastosowanie

w kontekście współpracy podroi. Jako agenta należy traktować każdy podrój, kierowany przez swojego lidera.

Opracowana koncepcja dopuszcza jeden stopień enkapsulacji. Oznacza to, że nie można dokonać kolejnego podzielenia i zdefiniować grupy podroi jako kolejnego agenta.

Środowisko, w obu przypadkach użycia uczenia maszynowego, jest tożsame ze środowiskiem graficznym lub środowiskiem fizycznym w jakim misje wykonują drony. Stan każdego drona będzie składał się z parametrów takich jak między innymi:

- pozycja X, Y, Z w układzie odniesienia lidera,
- odległość od innych dronów w roju,
- oraz inne parametry zgodne z dostępnymi dla danego egzemplarza UAV.

Każdy dron może podejmować różne akcje, takie jak:

- zmiana pozycji,
- miana prędkości,
- inne akcje zgodne z dostępnymi dla danego egzemplarza.

Nagroda dla dronów może zostać przyznana na podstawie różnych kryteriów, w zależności od celu misji, np.:

- za ukończenie misji w określonym czasie,
- za zachowanie optymalnej odległości od innych dronów,
- za precyzyjne osiąganie punktów trajektorii,
- za oszczędność energii.

Polityka: Każdy dron w roju będzie uczył się optymalnej polityki, aby wykonywać swoje zadania. Polityka może dotyczyć takich aspektów jak:

- jak dostosować swoją prędkość, aby utrzymać formacje,
- jak reagować na zmiany w otoczeniu np. nowe przeszkody,
- jak koordynować ruch z innymi dronami, aby cała formacja osiągnęła cel efektywnie.

Wypracowaną koncepcją jest ustalenie relacji jeden do jednego między rodzajem misji, definiowanym w pliku wsadowym dla programu (załącznik nr 3), a nagrodą jaka zostaje przyznana w tej misji. Dla przykładu dla najpopularniejszej misji monitorowania terenu w przelocie nagroda

przyznawana jest za efektywne wypracowanie tras przelotu w kontekście minimalizowania zużycia energii. Na politykę składają się wszystkie 3 wymienione wyżej aspekty.

6.4. Przeprowadzone badania

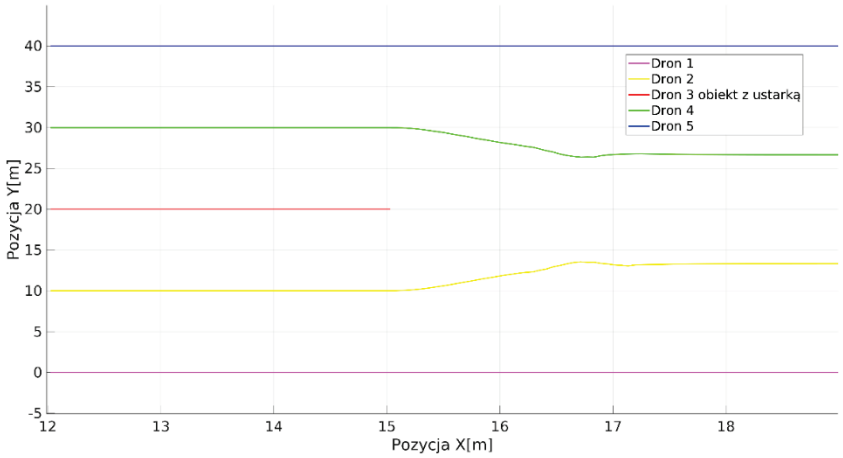
Zgodnie z założeniami z rozdziału 2 przeprowadzono szereg eksperymentów mających na celu zbadanie przede wszystkim:

1. Wpływu udziału programu matki-operatora wraz z zastosowanymi mechanizmami inteligencji.
2. Dzielenia na podroje i utrzymywania szyku w przypadku zaistnienia dynamicznych wydarzeń zmuszających do przeformowania roju.
3. Zdolności uczenia roju, w sposób mający na celu zapobiec zjawisku overfitting'u.

W pracy przyjęto problematykę, z którą najczęściej mierzą się bezzałogowe obiekty latające w postaci omijania przeszkód i wykonywania zadań z gatunku monitorowania, przeszukiwania, mapowania etc. Badania wykonywano przyrostowo, aby przedstawić wpływ poszczególnych zmiennych. Z racji na to, że eksperymenty wykonywano w warunkach losowo zmiennych, część badań zaprezentowano z ponowieniem aby przedstawić zachowania roju w różnych warunkach. Przeszkody symulowane były w postaci bloku mieszkalnego. Jego umieszczenie było wybierane losowo, a w oznaczonych przypadkach celowo powtarzano pozycje np. dla innej ilości dronów w roju, aby pokazać różnicę między przelotami w identycznych warunkach.

6.4.1. Dobranie trajektorii jednostek roju przez matkę-operatora

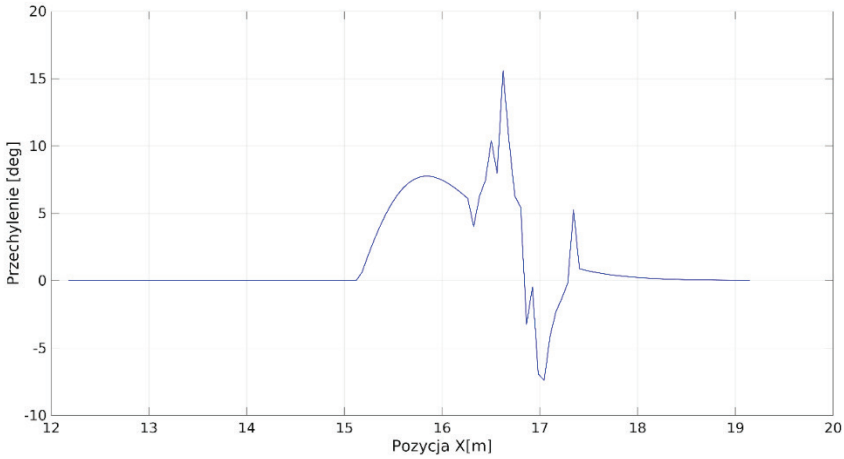
Powtórzono eksperyment z rysunku 5.11 dla 5 dronów z wymuszeniem awarii jednostki w środku szyku. Początkowo przeprowadzono badanie bez udziału programu sterującego. Następnie wykonano tą samą symulację wraz z aktywnym modułem matki operatora. Badanie miało na celu sprawdzenie poprawy trajektorii jednostek przy udziale matki-operatora.



Rys 6.6. Trajektoria roju 5 dronów korygowana przez obiekty podczas usterki jednostki w środku grupy [źródło własne]

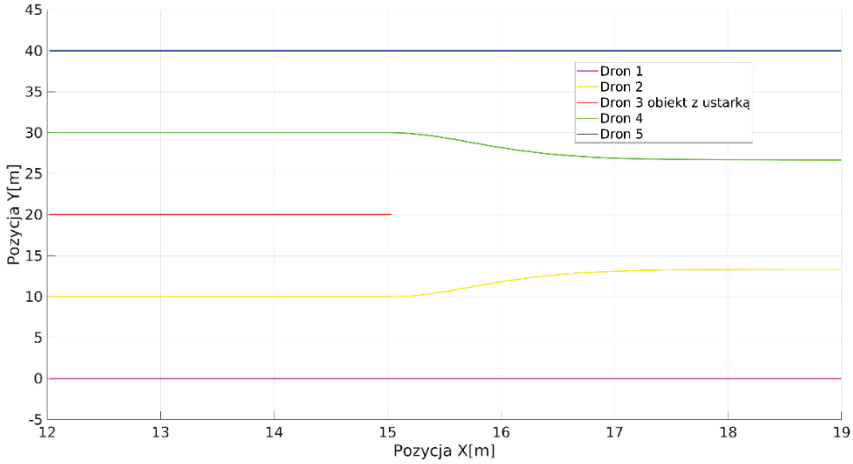
Na rysunku 6.6 przedstawiono trajektorie zebraną podczas przelotu dronów w przygotowanym środowisku graficznym. Dron nr 3 został wybrany jako jednostka, która ma ulec awarii, natomiast czas awarii był losowo dobrany. Poprzez konieczność odwzorowania szyku przez drony 2 i 4 wymuszoną intencjonalną usterką UAV pomiędzy nimi, można łatwo zauważyć niespójność w reakcji dronów. W idealnych warunkach symulacyjnych trajektoria dronów 2, 4 powinna być lustrzanym odbiciem dążącym do płynnego uzupełnienia szyku. Tak ja w przypadku gdy tylko jeden dron miał uzupełnić szyk (rys 5.11). W powyższym przypadku doszło do zjawiska „nadkompensacji" spowodowanej wzajemnym przybliżaniem się do siebie dwóch jednostek. Spowodowało to zbytnie zbliżenie i każda z jednostek zmuszona była odsunąć się od środka, co również wygenerowało, tym razem mniejsze, przesterowanie. Nieregularność trajektorii spowodowana jest wpływem zewnętrznych czynników uwzględnionych w środowisku symulacyjnym, a także możliwym niedokładnościom przy odczycie pozycji wzajemnej.

Zmiana pozycji w osi Y realizowana jest przez przechylenie poszczególnych jednostek. Wykres przechylenia drona nr 2 w czasie przedstawia rys. 6.7.



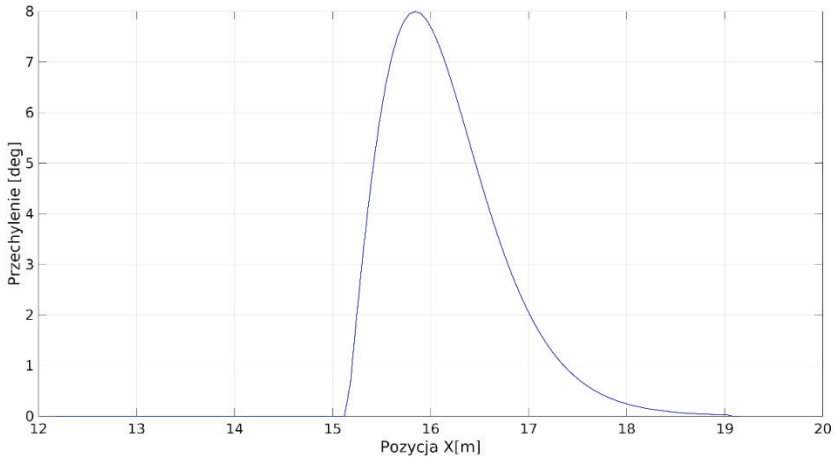
Rys 6.7. Przechylenie drona nr 2 podczas korekty wzajemnej dronów [źródło własne]

Można zauważyć jak bardzo nieoptymalne jest zachowanie jednostki. Musi ona wykonać ruch w jednym kierunku, a następnie skorygować wykonując ruch w drugim kierunku, aby ustalić właściwą pozycje.
Eksperyment został powtórzony, przy udziale modułu matki-operatora, przy zapisie momentu awarii drona nr 3, aby można było lepiej porównać wyniki.



Rys 6.8. Trajektoria roju 5 dronów korygowana przez matkę-operatora podczas usterki jednostki w środku grupy [źródło własne]

Na rysunku 6.8. przedstawiono trajektorie zebraną podczas przelotu roju 5 dronów kontrolowanego przez matkę-operatora. Po rozpoznaniu upadku drona nr 3 lider przesyła informacje do programu sterującego (matki). Obliczone zostają punkty trajektorii, które następnie matka adresuje, za pośrednictwem lidera, do dronów 2 i 4. Wykres przechylenia dla drona nr 2 przedstawiono na rys 6.9.



Rys 6.9. Przechylenie drona nr 2 podczas korekty trajektorii przez matkę-operatora [źródło własne]

Przechylenie zostało zoptymalizowane, dron wykonał ruch płynnie, bez drgań, ustawiając się w docelowej pozycji na osi Y.

6.4.2. Reakcja roju w szyku liniowym na wystąpienie nieznanej przeszkody

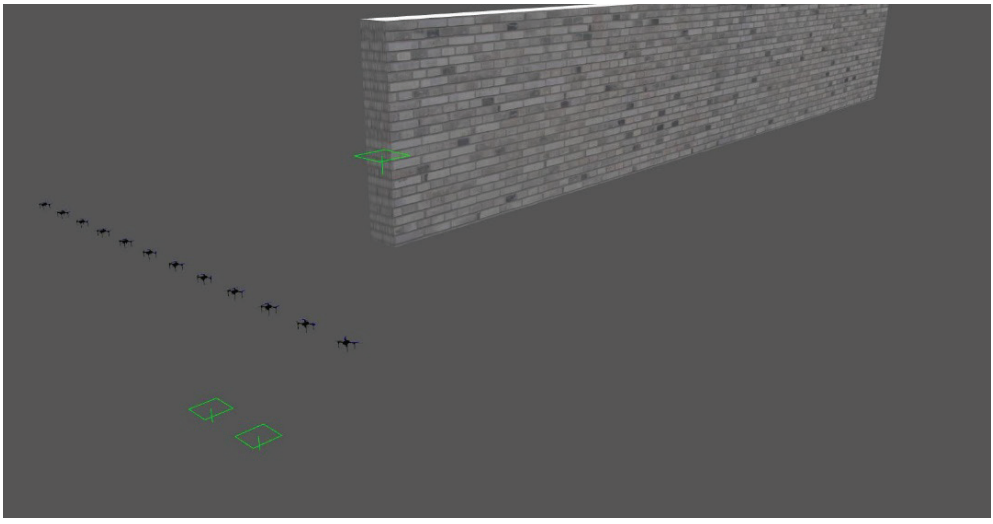
W tym eksperymencie wykonano przelot dronów w linii prostej w szyku liniowym. W losowym punkcie czasu drony napotykają przeszkodę w postaci bloku mieszkalnego. Może on zostać zauważony z odległości 15 metrów. Przy użyciu matki-operatora wyznaczana jest trajektoria dla wszystkich jednostek. Następuje podział roju na dwa podroje, a także wyznaczony jest nowy lider dla grupy do której nie trafił wcześniej wybrany lider. Eksperyment został wykonany dwu krotnie dla różnych losowych pozycji przeszkody dla roju składającego się z 12 dronów. Pozycje

przeszkody zostały zapamiętane w pamięci środowiska graficznego, a następnie eksperyment powtórzono dla 20 dronów.
Na rysunkach przyjęto oznaczenia:

- Kolorem niebieskim oznaczono pierwotny, całkowity rój.
- Kolorem czerwony, oznaczono podrój nr 1.
- Kolorem żółtym, oznaczono podrój nr 2.
- Kolorem zielonym oznaczono przeszkodę.
- Pogrubioną linią oznaczono drona lub drony pełniące rolę lidera.

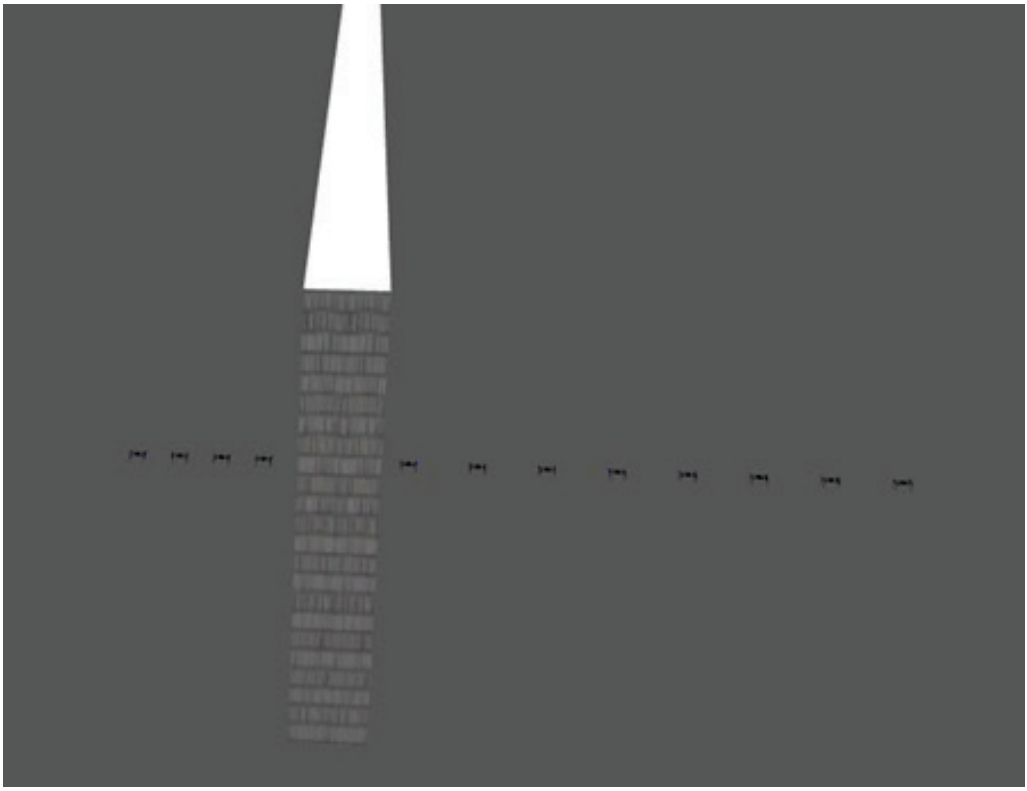
6.4.2.1. Przelot roju 12 dronów i reakcja na losową przeszkodę

Wszystkie eksperymenty dla wykonano dla dwóch rojów. Pierwszy z nich liczył 12 jednostek, drugi 20. Eksperymenty prowadzono w różnych warunkach, aby uniknąć sytuacji w której odpowiedz roju będzie przygotowana tylko dla jednego przypadku. Do przeprowadzenie eksperymentów posłużyło przygotowane środowisko graficzne. Widok roju 12 dronów w szyku liniowym podczas przelotu przedstawia rys 6.10:



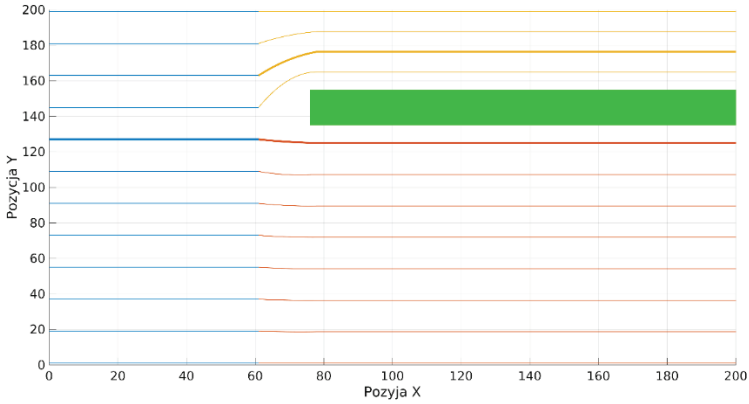
Rys 6.10. Reakcja roju 12 dronów na losową przeszkodę. Widok w środowisku graficznym przed napotkaniem przeszkody [źródło własne]

W trakcie przelotu roju zaprezentowanego na rysunku 6.10 włączony został moduł graficzny programu Gazebo umożliwiający podgląd trajektorii poszczególnych dronów. Został on opisany w podrozdziale 4.4.
W odpowiedzi na napotkaną przeszkodę drony reagują dzieląc się na dwa podroje.



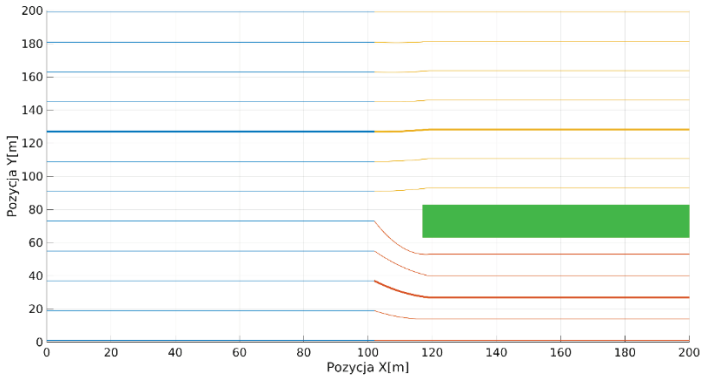
6.11. Reakcja roju 12 dronów na losową przeszkodę. Podział roju na dwa podroje [źródło własne]

Jak zostało zaprezentowane na rysunku 6.11 rój dwunastu dronów podzielił się na dwa podroje. Jeden z nich liczy 4 jednostki, drugi 8. Szczegółowe trajektorie poszczególnych obiektów zostały przedstawione na rysunku 6.12.



Rys 6.12. Reakcja roju 12 dronów na losową przeszkodę. Pierwsza pozycja przeszkody [źródło własne]

Podczas eksperymentu przeprowadzonego w środowisku graficznym, dane o pozycji każdego obiektu zostały zarchiwizowane. Na ich podstawie wygenerowane został rysunek 6.12 ilustrujący trajektorie każdej jednostki oraz odpowiedz roju na przeszkodę. Analogicznie powtórzono eksperyment dla innej pozycji przeszkody.

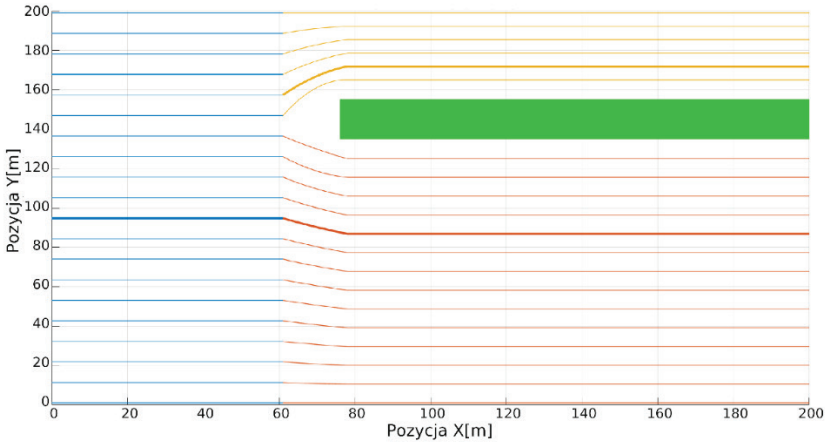


Rys 6.13. Reakcja roju 12 dronów na losową przeszkodę. Druga pozycja przeszkody [źródło własne]

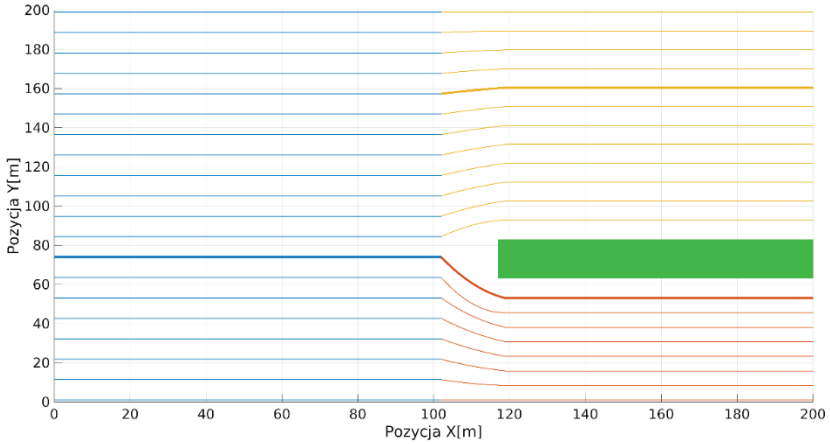
Jak widać na runkach 6.12 i 6.13 zmiana warunków determinuje zmianę zachowania obiektów w tym wybranie innych liderów podrojów. Kolejne eksperymenty prowadzone były analogicznie do rozpatrywanego

przypadku 6.4.2.1. Przedstawiono trajektorie dronów zarejestrowane podczas badań w środowisku graficznym.

6.4.2.2. Przelot roju 20 dronów i reakcja na losową przeszkodę



Rys 6.14. Reakcja roju 20 dronów na losową przeszkodę. Pierwsza pozycja przeszkody [źródło własne]

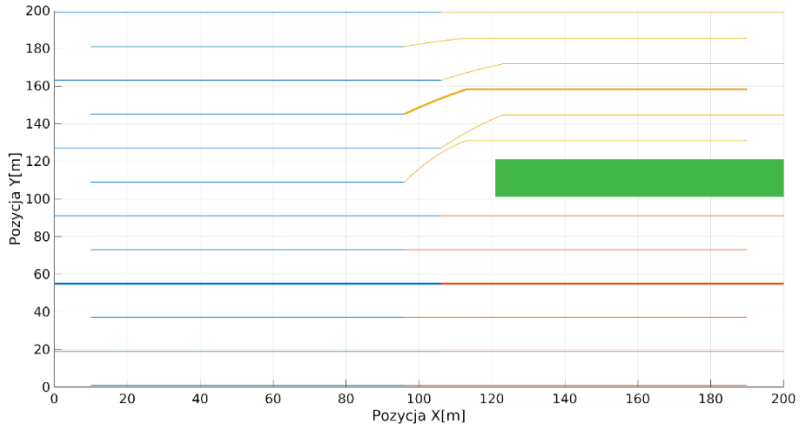


Rys 6.15. Reakcja roju 20 dronów na losową przeszkodę. Druga pozycja przeszkody [źródło własne]

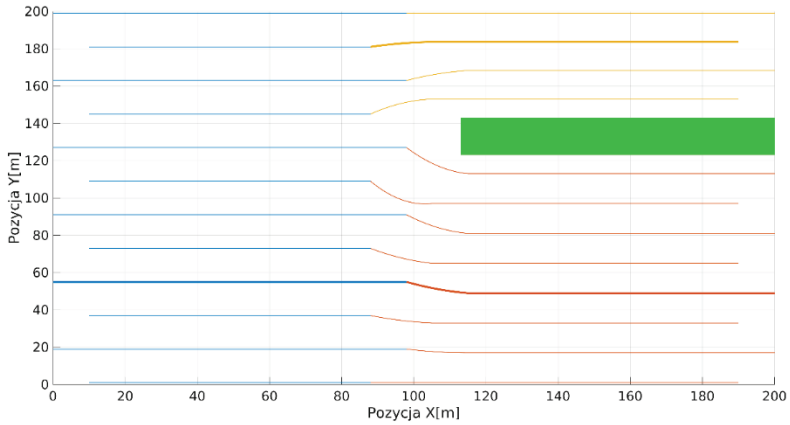
Na rysunkach 6.14-15 przedstawiono reakcje roju liczącego 20 jednostek sterowanego przez matkę-operatora. Scenariusz eksperymentu jest analogiczny do zaprezentowanego na rysunkach 6.12, 6.14. Mechanizm komunikacji został omówiony podczas opisu eksperymentu analizującego wpływ programu matki-operatora w podrozdziale 6.6.1. W przypadku 4 powyższych badań dodatkowo program sterujący był odpowiedzialny za wybranie nowego lidera, oraz przydzielenie dronów do podroi.

6.4.3. Szyk dwurzędowy

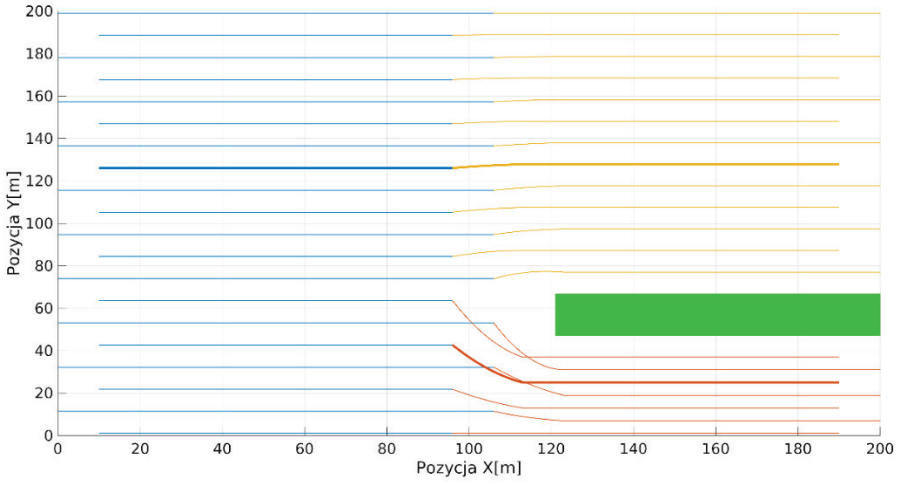
Przeprowadzono symulacje dla szyku dwurzędowego, opisanego w podrozdziale 6.2 i przedstawionego na rysunku 6.3. Pierwsza linia dronów ustawiona jest 10 metrów przed drugą. Dzięki temu drony tworzą dwa rzędy. Odległość między rzędami jest zachowywana podczas lotu. Obiekty będące na czele roju rozpoznając przeszkodę przekazują informację wewnątrz całej grupy, przez co jednostki będące z tyłu mają więcej czasu na reakcje. Należy zwrócić uwagę, że na rysunkach obrazujących przelot roju dwurzędowego, 6.16-19 linie reprezentujące trajektorie są wzajemnej przesunięte w pozycji X, przez co mogą się przecinać. Nie oznacza to kolizji dronów. Jednostki z drugiego rzędu mogą przeciąć ślad pozostawiony przez obiekty z pierwszego rzędu, ponieważ te są cały czas 10m przed nimi.



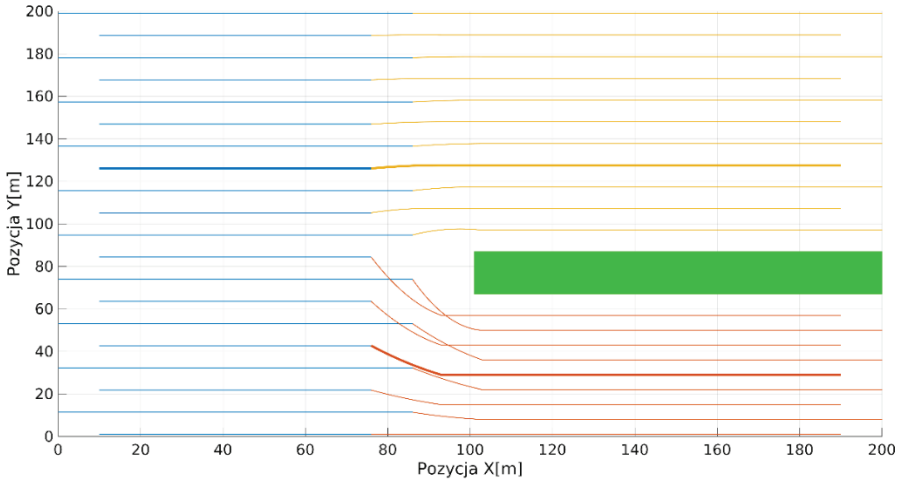
Rys 6.16. Trajektoria lotu roju dwurzędowego składającego się z 12 jednostek. Pierwsza pozycja przeszkody [źródło własne]



Rys 6.17. Trajektoria lotu roju dwurzędowego składającego się z 12 jednostek. Druga pozycja przeszkody [źródło własne]



Rys 6.18. Trajektoria lotu roju dwurzędowego składającego się z 20 jednostek. Pierwsza pozycja przeszkody [źródło własne]

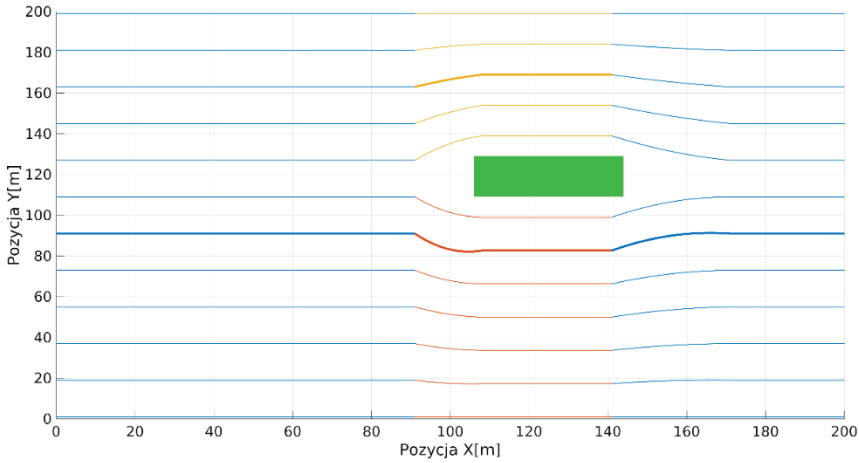


Rys 6.19. Trajektoria lotu roju dwurzędowego składającego się z 20 jednostek. Druga pozycja przeszkody [źródło własne]

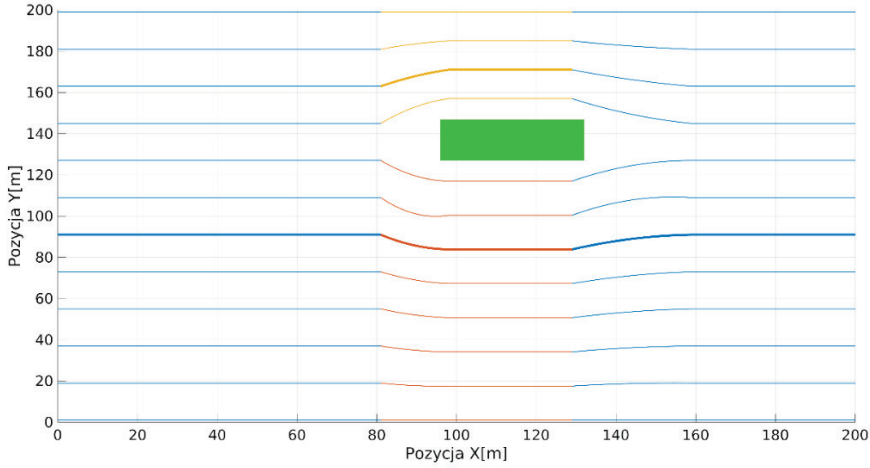
Na rysunkach 6.16-6.19 zaprezentowano reakcje roju dwurzędowego na scenariusz analogiczny do eksperymentów 6.12-6.15. W przypadku szyku dwurzędowego możliwe jest ustalenie trajektorii dla dronów znajdujących się z tyłu (w drugim rzędzie) z wyprzedzeniem czasowym. Umożliwia to dokonanie większych korekt widocznych na rysunkach 6.16-6.19.

6.4.4. Uzupełnienie szyku

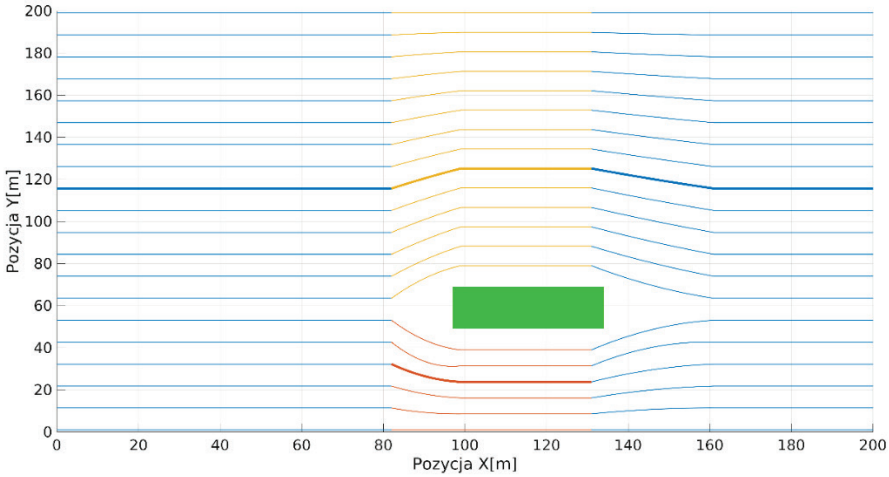
W eksperymencie założono warunki analogiczne do badań z podrozdziału 6.6.2. z tą różnicą, że przeszkoda kończy się w losowym miejscu. Drony mają za zadanie wrócić do wcześniejszego, wyjściowego szyku. Podroje łączą się w jeden rój. Anulowana zostaje również funkcja lidera, którą otrzymał jeden z obiektów podczas podziału na podroje. Opisane zachowanie zostało zilustrowane na rysunkach 6.20-23.



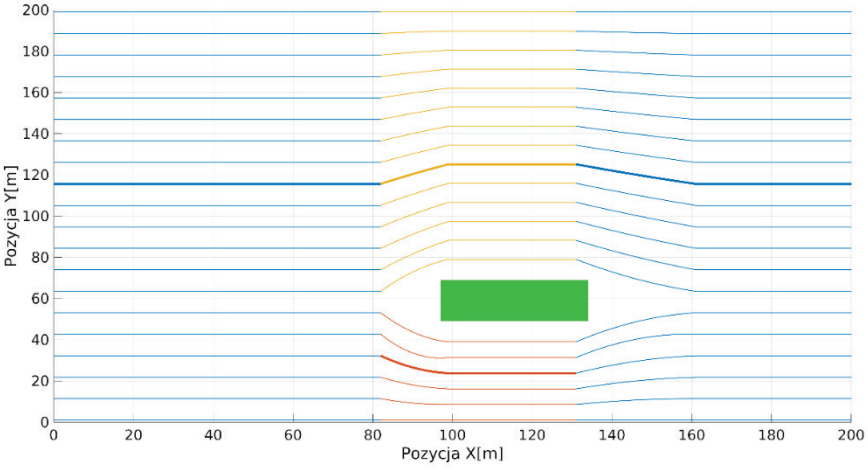
Rys 6.20 Trajektoria roju 12 dronów podczas przelotu z ominięciem przeszkody i powrotem do pierwotnego szyku. Pierwsza pozycja przeszkody [źródło własne]



Rys 6.21 Trajektoria roju 12 dronów podczas przelotu z ominięciem przeszkody i powrotem do pierwotnego szyku. Druga pozycja przeszkody [źródło własne]



Rys 6.22 Trajektoria roju 20 dronów podczas przelotu z ominięciem przeszkody i powrotem do pierwotnego szyku. Pierwsza pozycja przeszkody [źródło własne]



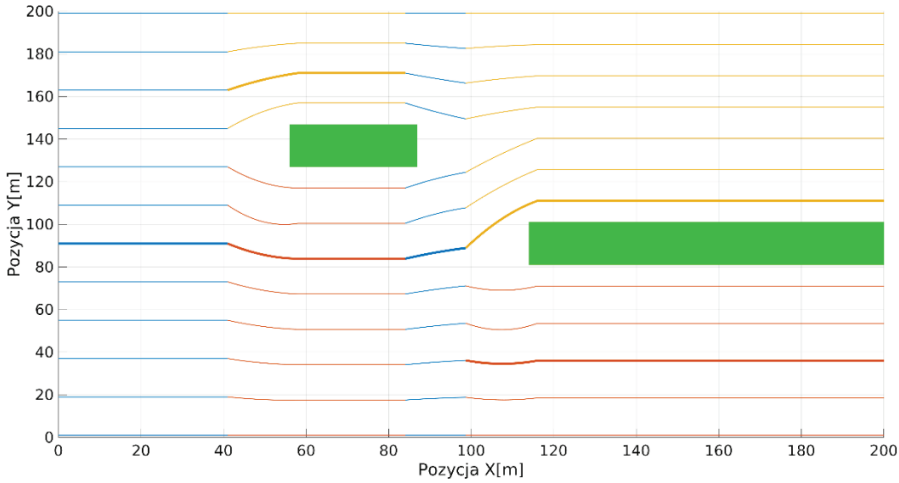
Rys 6.23 Trajektoria roju 20 dronów podczas przelotu z ominięciem przeszkody i powrotem do pierwotnego szyku. Druga pozycja przeszkody [źródło własne]

W eksperymentach zilustrowanych na rysunkach 6.20-6.23 można zauważyć, że drony zaczynają powracać do scalenia podroji w jeden rój jeszcze przed

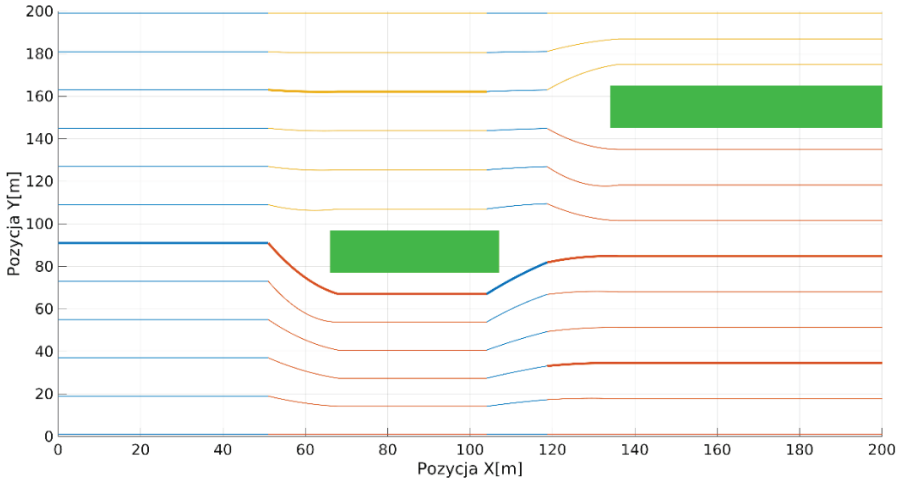
przelotem za zakończenie przeszkody. Ruch ten jest bardziej dynamiczny na początku. Następuje również przywrócenie początkowego lidera.

6.4.5. Konieczność wykonania manewru przed zakończeniem uzupełnienia szyku

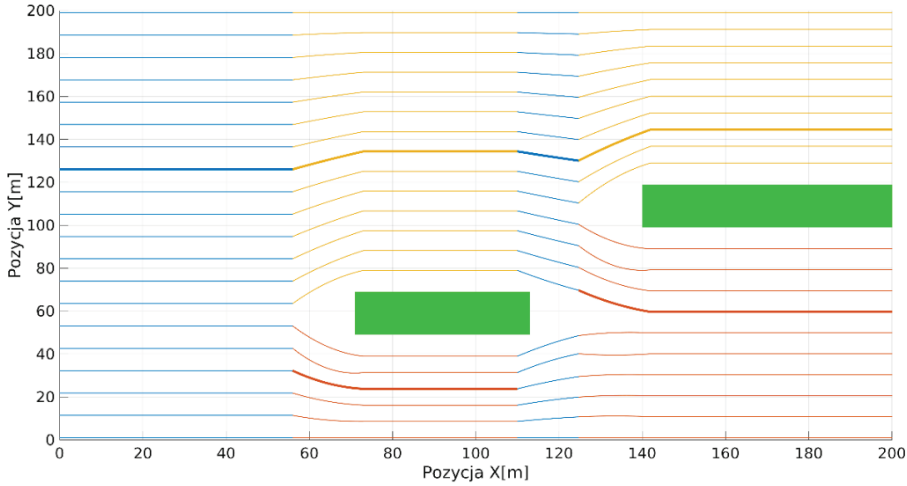
Aby wymusić konieczność wykonania manewru przed zakończeniem powrotu do pierwotnego szyku zaprojektowano scenariusz eksperymentu, w którym kolejna przeszkoda rozpoczyna się w odległości 20-35 metrów po zakończeniu pierwszej przeszkody. Wymusza to na bezzałogowych obiektach latających dynamiczną korekcje pozycji przez ponowny podział na podroje, zanim jednostki osiągną równe wzajemne rozłożenie



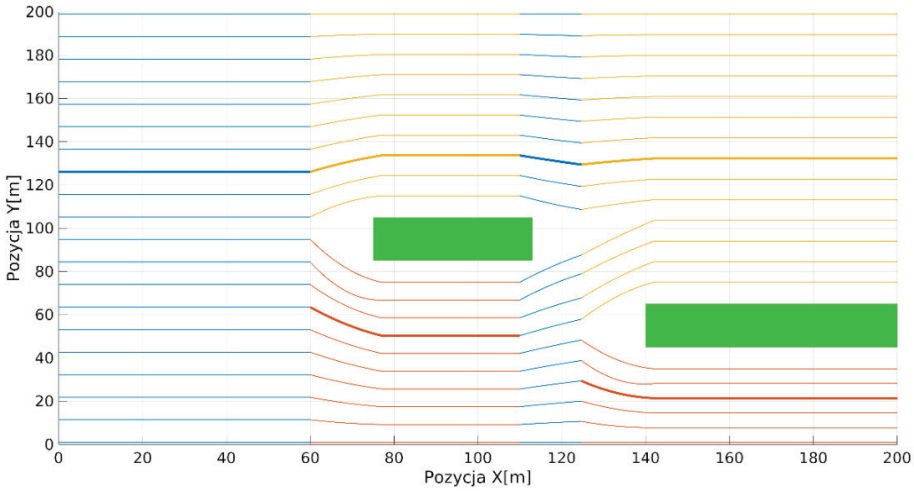
.Rys 6.24. Przelot roju 12 dronów z ominięciem dwóch przeszkód. Pierwszy układ przeszkód [źródło własne]



Rys 6.25. Przelot roju 12 dronów z ominięciem dwóch przeszkód. Drugi układ przeszkód [źródło własne]



Rys 6.26. Przelot roju 20 dronów z ominięciem dwóch przeszkód. Pierwszy układ przeszkód [źródło własne]



Rys 6.27. Przelot roju 20 dronów z ominięciem dwóch przeszkód. Drugi układ przeszkód [źródło własne]

W przypadku scenariusza z dwiema przeszkodami przedstawionego na rysunkach 6.24-6.27 można zauważyć wiele manewrów wykonywanych przez drony. Dzięki temu, że po zakończeniu pierwszej przeszkody obiekty wykonują dynamiczniejszy ruch możliwe jest lepsze przysposobienie do ewentualnego manewru ominięcia kolejnej przeszkody. Przez zadanie dynamicznych manewrów matka roju zmusza obiekty do szybkiego powrotu do szyku o możliwie jednakowych odległościach wzajemnych między jednostkami co przekłada się na możliwość reakcji na kolejną przeszkodę.

6.5. Analiza wyników eksperymentów

Przez umyślne dobranie scenariusza z awarią drona znajdującego się w środku szyku uwydatniona została słabość odwzorowania szyku na podstawie mechanizmu wzajemnej kompensacji pozycji. W przypadku braku łączności z matką-operatorem jest to funkcjonujące rozwiązanie. Dodatkowe wykorzystanie nadrzędnego programu sterującego niesie za sobą poprawę jakości sterowania. W bardziej zaawansowanych przypadkach, wymagających szybkiej reakcji, problem nadmiernego skompensowania, mógłby nieść większe przesterowania.

Nauka dobierania właściwych trajektorii dla poszczególnych jednostek roju jest możliwa do realizacji dzięki opracowanemu środowisku graficznemu. Umożliwia ono wykonywanie prób niezbędnych do nauczenia algorytmu sztucznej inteligencji. Należy w tym celu zadawać losowo dobrane warunki, a program uczy jak dobierać strategie, aby optymalizować zużycie baterii.

Mechanizm podziału na podroje został dobrany tak aby jednostki przydzielane były, o ile jest taka możliwość, w ilościowo równe podroje. Widać to na rysunkach 6.12-19 oraz 6.24-27.

W każdym z przywołanych rysunków następuje decyzja obiektu, który mógłby przelecieć poniżej lub powyżej przeszkody. Najlepsze w tym zakresie możliwości daje szyk wielorzędowy. W przypadkach badań nad szykiem dwurzędowym obiekty będące w drugiej linii miały większy czas i program sterujący matki-operatora mógł zadać im wyznaczone trajektorie z większym wyprzedzeniem, bardziej korygując rozkład podroi. Wyjątkami w kierowaniu jednostek do mniej licznych grup są przypadki zawarte na rysunkach: 6.25 oraz 6.27. Dla bezpieczeństwa poszczególnych jednostek program sterujący zezwolił aby jednostki, przed którymi znajdowała się przeszkoda utrzymały swoją trajektorię.

6.6. Podsumowanie rozdziału

W rozdziale przedstawiono strukturę programu. Omówiony został przepływ danych i koncepcja obiektowej reprezentacji bezzałogowego obiektu latającego, roju, matki a także środowiska. Abstrakcja oprogramowania została przygotowana w sposób umożliwiający śledzenie komunikacji realizowanej fizycznie na poziomie programowym, co czyni program elastycznym, przygotowanym do możliwej rozbudowy oraz łatwym do śledzenia i naprawiania ewentualnych błędów. Każdemu obiektowi odpowiadają parametry i metody zgodne z fizycznie reprezentowanymi cechami obiektu. Zdefiniowana została struktura misji, która może zostać łatwo modyfikowalna przez zmiany w pliku, który służy jako dane wejściowe dla programu matki-operatora. Omówiono możliwą ingerencje operatora, jego możliwości w zakresie przejęcia poszczególnej jednostki, a także

monitorowania bezzałogowych obiektów latających przy użyciu proxy protokołu MAVLink w programie QGC.

Rozważono poszczególne formacje jakie mogą przyjąć drony w roju. Przyjęto zuniwersalizowaną formacje, dającą największe możliwości w zakresie prostoty obsługi, skalowalności, funkcjonalności, a także możliwości zadania szyku przez operatora w pliku misji. Przeanalizowano zastosowanie uczenia maszynowego w kontekście sterowania w roju i przedstawiono autorską koncepcje dwustopniowego użycia uczenia przez wzmocnienie. Dzięki zastosowaniu jednych z najnowszych i najlepiej dopracowanych technologii, w postaci bibliotek ROS i TensorFlow, które zintegrowano w tym module otrzymano innowacyjny program sterujący dla roju dronów. Pozwala on wykorzystać możliwości sztucznej inteligencji, a dzięki prostocie implementacji i warstwie abstrakcji oprogramowania wykonanej tak, aby była zbieżna z fizycznymi obiektami całość jest prosta w obsłudze i debugowaniu. W pracy położono nacisk na praktyczne możliwości wykorzystania algorytmów uczenia maszynowego, w tym głównie Q-learning, który okazał się najlepszym wyborem. Wykorzystanie uczenia przez wzmacnianie w kontekście zastosowania do sterowania bezzałogowymi obiektami latającymi wraz z uwzględnieniem wdrażalności przygotowanej koncepcji stanowi innowacyjny wkład pracy. Wykorzystanie gotowych rozwiązań, takich jak biblioteka sztucznej inteligencji TensorFlow, która została użyta przez Pentagon[55] w projekcie dotyczącym sterowania dronami sprawia, że przygotowany program matki-operatora jest modułowy a kod mógł zostać przygotowany z użyciem gotowych rozwiązań. Gwarantują one wysoką jakość inżynierką aplikacji, a także możliwość późniejszych prac i modyfikacji kodu z zachowaniem przejrzystości oprogramowania.

Scenariusze badań przedstawiają przelot jednostek w szyku, a także warunkach które są możliwe dla wykonania dla różnego typu jednostek latających, nie tylko wielowirnikówców. Zostały dobrane tak, aby odzwierciedlać typowe do wykonania misje dla wielu dronów w sektorach omówione w rozdziale 2. Wszystkie założone badania potwierdzające żądaną jakoś opracowanego modułu zostały zrealizowane z sukcesem. Korzyścią z użycia, w warunkach pełnej komunikacji, modułu matki-roju jest osiągnięcie nauczonych reakcji na niezaplanowane wydarzenia mające

miejsce w trakcie wykonywania zadanej misji. Wykazano w formie graficznej poprawę realizacji trajektorii przez poszczególne jednostki. Scenariusze wykonano w warunkach losowych, badając dynamiczną odpowiedz roju. W celu poprawnej walidacji, większość eksperymentów powtórzono, aby móc zestawić dane o zachowaniu jednostek w różnych warunkach, minimalizując możliwość losowego natrafienia na mniej lub bardziej korzystne warunki misji i wpływu losowych warunków na wyniki walidacji koncepcji.

Zastosowanie algorytmu Q-learning może poprawić jakość pracy dronów, a możliwość wykonania misji w środowisku graficznym, w którym można przeprowadzić większą ilość prób bez konieczności angażowania fizycznego sprzętu pozwala skutecznie wytrenować drony do wykonywania zadanych misji. Dzięki użyciu bazy danych i zapisie wytrenowanych rezultatów, a także możliwości randomizacji warunków dla poszczególnych misji możliwe jest osiągnięcie bardzo dobrych rezultatów bez ryzyka wystąpienia nadmiernego dopasowania. Uczenie dronów sterowanych w roju w przygotowanym środowisku graficznym jest nowatorskim pomysłem, mogącym stanowić bazę do wdrażania sztucznej inteligencji dla dronów w szerokim spektrum zastosowań, bez potrzeby udziału człowieka, ryzyka zniszczeń sprzętu, a także kosztów eksploatacji podczas procesu uczenia.

Zakończenie

W pracy przedstawiono koncepcję sterowania w roju dla bezzałogowych obiektów latających. Zgodnie z dokonanym przeglądem literatury oraz postawionym celem podjęto starania aby rozważyć rój w ujęciu holistycznym, uwzględniając jednostkę latającą, środowisko umożliwiające pracę w kontrolowanych warunkach, komunikacje wewnątrz roju, oraz program sterujący pełniący rolę matki roju. Podział na moduły został dokonany, aby uczynić pracę wdrażalną do wielu sektorów, w których wykorzystywane są drony. Podjęty został trud, aby wykorzystać najnowsze badania naukowe. Uwaga została również skierowana na projekty rządowe wdrażane do fazy realizacji. Na ich podstawie wielokrotnie ewaluowano koncepcje, aby mogła sprostać postawionym wymogom, być użyteczna i nowatorska.

Użyte technologie zostały dobrane aby móc być wdrożone do obszarów przemysłu, gdzie cena jednostkowa zakupu często stanowi barierę przed implementacją nowoczesnych rozwiązań. Obranym kierunkiem było przedstawienie zastosowań sterowania w roju dla bezzałogowych obiektów latających dla różnych sektorów, bez koncentracji nad konkretnym dla obszaru typem misji. Realizując badania na poszczególnych fazach pracy wielokrotnie porównywano własne rozwiązania z istniejącymi już na rynku. Dla większości przypadków obrano implementacje polegającą na przystosowaniu obecnej już technologii, choć jak w przypadku przygotowanego własnego sterownika lotu nie wykluczono funkcjonowania obu rozwiązań. Użycie powszechnych technologii użytych w pracy niesie za sobą wiele korzyści, w tym możliwość dalszych prac nad sterowaniem w roju w większych zespołach badawczo inżynierskich i zachowanie skalarności oraz modułowości koncepcji.

Rozprawa zawiera autorskie rozwiązania, takie jak m.in. protokół do komunikacji wewnątrz grupy roju. Wykorzystano również powszechnienie stosowane rozwiązania jak np. protokół MAVLink wykorzystywany do wymiany danych z matką roju. Własny protokół jest w pełni wdrażalny w środowisku graficznym oraz osprzęcie fizycznym i możliwy do

implementacji w oprogramowaniu sterownika lotu ArduPilot. Ze względu na brak gotowych protokołów do komunikacji pomiędzy jednostkami umożliwiającymi minimalistyczną wymianę danych wraz z wykryciem dystansu pomiędzy obiektami zaimplementowano autorskie rozwiązanie. Wykorzystanie powszechnych oraz autorskich rozwiązań daje możliwość wymiany w zakresie poszczególnych modułów jak m. in. zmiana oprogramowania sterownika lotu na rozważany podczas prac PX4, lub użycie innego niż quadrocopter typu drona.

Synteza autorskich oraz gotowych technologii jest innowacyjnym wkładem pracy w zakresie sterowania bezzałogowymi obiektami latającymi. Rozprawa porusza zagadnienia związane z wieloma dyscyplinami wiedzy technicznej. Przeanalizowane zostały aspekty umożliwiające sterowanie w roju, a także samo sterowanie z uwzględnieniem bardzo szybko rozwijającej się gałęzi sztucznej inteligencji. Osiągnięte zostały wszystkie cele szczegółowe zawarte w rozdziale 2.

Praca porusza wiele aspektów, które mogą stać się podstawą do przyszłych badań lub wdrożeń do ściśle określonego sektora. Niektóre kierunki to np. rozszerzenie integracji z oprogramowaniem stacji naziemnej QGC w celu zadania misji rojowi UAV z użyciem przygotowanego w programie interfejsu, który obecnie pozwala zadawać misje pojedynczym jednostkom, wdrożenie większej ilości mechanizmów sztucznej inteligencji w celu adaptacji roju do wyznaczonych misji w określonym terenie.

Podsumowując, praca spełniła wszystkie założone cele szczegółowe, przedstawiając innowacyjne i interdyscyplinarne podejście do sterowania w roju bezzałogowych obiektów latających. Algorytm został sprawdzony w wielu scenariuszach. Zaprezentowane rozwiązania, zarówno autorskie, jak i wykorzystujące powszechne technologie, stanowią znaczący wkład w rozwój technologii UAV i mogą być bezpośrednio zastosowane w dalszych badaniach lub praktycznych wdrożeniach. Możliwość integracji przygotowanych modułów z istniejącymi systemami oraz ich adaptacji do różnych scenariuszy misji czyni przedstawioną koncepcję elastycznym narzędziem dla wielu sektorów przemysłu i nauki. Wykorzystanie uczenia maszynowego i przygotowanego środowiska graficznego pozwala

optymalizować pracę roju w bezpiecznych, powtarzalnych warunkach. Dzięki swojej modułowości i zgodności z obecnymi standardami, zaproponowany system otwiera perspektywy dalszego rozwoju technologii sterowania w roju.

LITERATURA

[1] Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., Devin M., Ghemawat S., Irving G., Isard M., Kudlur M., Levenberg J., Monga R., Moore S., Murray D.G., Steiner B., Tucker P., Vasudevan V., Warden P., Wicke M., Yu Y., Zhang X..: *TensorFlow: A system for large-scale machine learning*, 2016, https://www.researchgate.net/publication/303657108_TensorFlow_A_system_for_large-scale_machine_learning, dostęp 30.09.2024.

[2] Abdelkader M., Güler S., Jaleel H., Shamma J. S.: *Aerial Swarms: Recent Applications and Challenges*, 2021, https://link.springer.com/article/10.1007/s43154-021-00063-4, dostęp 29.08.2024.

[3] Agha S., Mohsan H., Othman N. Q. H., Li Y., Alsharif M. H., Khan M. A.: *Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends*, w *Intell Serv Robot*, 2023, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9841964/, dostęp 06.09.2024.

[4] Ahmed F., Yeole S. N.: *Design and Analysis of a Quadcopter*, 2016, https://www.researchgate.net/publication/357832831_Design_and_Analysis_of_a_Quadcopter, dostęp 01.09.2024.

[5] Akbari Y., Almaadeed N., Al-ma'adeed S., Elharrouss O.: *Applications, databases and open computer vision research from drone videos and images: a survey*, https://www.researchgate.net/publication/349504594_Applications_databases_and_open_computer_vision_research_from_drone_videos_and_images_a_survey, dostęp 06.09.2024.

[6] AL-Dosari K., Hunaiti Z., Balachandran W.: *Systematic Review on Civilian Drones in Safety and Security Applications*, 2023, https://www.researchgate.net/publication/369396771_Systematic_Review_on_Civilian_Drones_in_Safety_and_Security_Applications, dostęp 29.08.2024.

[7] Al-Haddad L. A., Jaber A. A., Giernacki W., Khan Z. H., Ali K. M., Tawafik M. A., Humaidi A. J.: Quadcopter Unmanned Aerial

Vehicle Structural Design Using an Integrated Approach of Topology Optimization and Additive Manufacturing, w Designs, 2024, https://www.mdpi.com/2411-9660/8/3/58, dostęp 01.09.2024.

[8] Alberts D.S., Haye R.E., Power to the Edge: Command and Control in the Information Age, Waszyngton, 2005.

[9] Albonico M., Đorđević M., Hamer E., Malavolta I.: Software engineering research on the Robot Operating System: A systematic Mapping study, https://www.sciencedirect.com/science/article/pii/S0164121222002503, dostęp 10.09.2024.

[10] Aljumah A.: UAV-Based Secure Data Communication: Multilevel Authentication Perspective, w Sensors, 2024, https://www.mdpi.com/1424-8220/24/3/996, dostęp 01.09.2024.

[11] Alkouz B., Bouguettaya A.: Formation-based Selection of Drone Swarm Services, 2020, https://www.researchgate.net/publication/353798257_Formation-based_Selection_of_Drone_Swarm_Services, dostęp 30.09.2024.

[12] Amazon, How Amazon is building its drone delivery system, https://www.aboutamazon.com/news/transportation/how-amazon-is-building-its-drone-delivery-system, dostęp 15.08.2024.

[13] Arfaoui A., Unmanned Aerial Vehicle: Review of Onboard Sensors, Application Fields, Open Problems and Research Issues, 2017, https://www.researchgate.net/publication/315076314_Unmanned_Aerial_Vehicle_Review_of_Onboard_Sensors_Application_Fields_Open
_Problems_and_Research_Issues, dostęp 06.09.2024.

[14] Artale V., Ricciardello A., Milazzo C.: Mathematical modeling of hexacopter, https://www.researchgate.net/publication/269073664_Mathematical_modeling_of_hexacopter,
dostęp 01.09.2024.

[15] Asaamoning G., Mendes P., Rosário D., Cerqueira E.: Drone Swarms as Networked Control Systems by Integration of Networking and Computing, w Sensors, 2021, https://www.mdpi.com/1424-8220/21/8/2642, dostęp 27.08.2024.

[16] Aspe G. D., Simmons B. M.: Rapid Flight Control Law Deployment and Testing Framework for Subscale VTOL Aircraft, 2022, https://ntrs.nasa.gov/api/citations/20220011570/downloads/NASA-TM-20220011570.pdf, dostęp 07.09.2024.

[17] Ausonio E., Bagnerini P., Ghio M.: Drone Swarms in Fire Suppression Activities: A Conceptual Framework, Genua, 2021.

[18] Australian Army Research Centre: Swarming and Counterswarming. Report on Applied Research Directions and Future Opportunities for Swarm Systems in Defence, https://researchcentre.army.gov.au/library/occasional-papers/swarming-and-counterswarming, dostęp 27.08.2024.

[19] Awasth S., Gramse N., Reining C., Roidl M.: UAVs for Industries and Supply Chain Management, https://arxiv.org/pdf/2212.03346, dostęp 29.08.2024.

[20] Azzam R., Boiko I., Zweiri Y.: Swarm Cooperative Navigation Using Centralized Training and Decentralized Execution, w Drones, 2023, https://www.mdpi.com/2504-446X/7/3/193, dostęp 28.08.2024.

[21] Bag S., Gupta S., Luo Z.: Examining the role of logistics 4.0 enabled dynamic capabilities on firm performance, 2020, https://www.emerald.com/insight/content/doi/10.1108/IJLM-11-2019-0311/full/html, dostęp 29.08.2024.

[22] Balestrieri E., Daponte P., Vito L. De, Lamonaca F.: Sensors and Measurements for Unmanned Systems: An Overview, w Sensors, 2021, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7926674/, dostęp 06.09.2024.

[23] Bartmiński P., Siłuch M., Kociuba W.: *The Effectiveness of a UAV-Based LiDAR Survey to Develop Digital Terrain Models and Topographic* Texture Analyses, w Sensors, 2023, https://www.mdpi.com/1424-8220/23/14/6415, dostęp 06.09.2024.

[24] Bassolillo S.R., D'Amato E., Notaro I., Blasi L., Mattei M.: Decentralized Mesh-Based Model Predictive Control for Swarms of UAVs,

https://www.researchgate.net/publication/343401414_Decentralized_Mesh-Based_Model_Predictive_Control_for_Swarms_of_UAVs, w Sensors, 2020, dostęp 06.09.2024.
[25] Bastourous M.: Decentralized Control of Groups of Autonomous UAVs, https://theses.hal.science/tel-03937831/file/BASTOUROUS.pdf, 2021, dostęp 28.08.2024.
[26] Beaver Works: Project Perdix, https://beaverworks.ll.mit.edu/CMS/bw/projectperdixcapstone, dostęp 29.08.2024.
[27] Beni G.: From Swarm Intelligence to Swarm Robotics, konferencja Swarm Robotics, Santa Monica, 2004.
[28] Beni G., Wang J.: Swarm Intelligence in Cellular Robotic Systems, Berlin, 1993.
[29] Berman S., Halasz A., Kumar V., Pratt S.: Algorithms for the Analysis and Synthesis of a Bio-Inspired Swarm Robotic System, https://labs.engineering.asu.edu/acs/wp-content/uploads/sites/33/2013/01/Berman_SAB2006.pdf, dostęp 27.08.2024.
[30] Bęben K., Głębocki R.: Architecture of the UAV Swarm Command and Control Systems – an Overvie, 2021, https://psaa.meil.pw.edu.pl/AEC2021/Papers/AEC_2021_136_Beben_Glebocki.pdf, dostęp 27.08.2024.
[31] Bin H., Amahah J.: The Design of an Unmanned Aerial Vehicle Based on the ArduPilot, https://www.researchgate.net/publication/289318923_The_Design_of_an_Unmanned_Aerial_Vehicle_Based_on_the_ArduPilot, dostęp 06.09.2024.
[32] Bonabeau E., Dorigo M., Theraulaz G.: Swarm Intelligence: From Natural to Artificial Systems, Nowy York, 2004.
[33] Brede B., Bartholomeus H.M., Barbier N., Pimont F., Vincent G., Herold M.: Peering through the thicket: Effects of UAV LiDAR scanner settings and flight planning on canopy volume discovery, https://www.sciencedirect.com/science/article/pii/S1569843222002448, dostęp 06.09.2024.
[34] Bu Y., Yan Y., Yang Y.: Advancement Challenges in UAV Swarm Formation Control: A Comprehensive Review, https://www.mdpi.com/2504-446X/8/7/320, dostęp 28.08.2024.

[35] Campion M., Ranganathan P., Faruque S.: A Review and Future Directions of UAV Swarm Communication Architectures, https://und.edu/research/rias/_files/docs/swarm_ieee.pdf, dostęp 29.08.2024.
[36] Cao S., Zhou Y., Yin D., Lai J.: UWB Based Integrated Communication and Positioning System for Multi-UAVs Close Formation, 2018, https://www.atlantis-press.com/proceedings/mecae-18/25893731, dostęp 02.09.2024.
[37] Chen R., Li J., Peng T.: Decentralized UAV Swarm Scheduling with Constrained Task Exploration Balance, w Drones, 2023, https://www.mdpi.com/2504-446X/7/4/267, dostęp 28.08.2024.
[38] Chen Y., Guo M.: Multi-UAV Deployment in Obstacle-Cluttered
[39] Environments with LOS Connectivity, https://arxiv.org/pdf/2308.12117, dostęp 06.09.2024.
[40] Chung T., Daniel R.: DARPA OFFSET: A Vision for Advanced Swarm Systems through Agile Technology Development and ExperimentationDARPA OFFSET: A Vision for Advanced Swarm Systems through Agile Technology Development and Experimentation, 2023, https://www.researchgate.net/publication/367321998_DARPA_OFFSET_A_Vision_for_Advanced_Swarm_Systems_through_Agile_Technology_Development_and_ExperimentationDARPA_OFFSET_A_Vision_for_Advanced_Swarm_Systems_through_Agile_Technology_Development_and_Exper, dostęp 28.08.2024.
[41] Corke P., Wark T., Jurdak R., Hu W., Valencia P., Moore D.: Environmental Wireless Sensor Networks, w Proceedings of the IEEE, 2010.
[42] Corsel C. W.: YOLO-based Obstacle Avoidance for Drones, 2020, https://theses.liacs.nl/pdf/2019-2020-CorselCW.pdf, dostęp 06.09.2024.
[43] Cybulski P.: A Framework for Autonomous UAV Swarm Behavior Simulation, https://annals-csis.org/Volume_18/drp/pdf/278.pdf, dostęp 30.08.2024.

[44] DAPRA: Collaborative Operations in Denied Environment (CODE) (Archived), https://www.darpa.mil/program/collaborative-operations-in-denied-environment, dostęp 29.08.2024.
[45] DAPRA: Gremlins (Archived), https://www.darpa.mil/program/gremlins, dostęp 29.08.2024.
[46] DARPA: OFFensive Swarm-Enabled Tactics (OFFSET) (Archived), https://www.darpa.mil/work-with-us/offensive-swarm-enabled-tactics, dostęp 28.08.2024.
[47] De Gennaro M. C., Jadbabaie A., Swarm Stability and Control for Unmanned Aerial Vehicles, w IEEE Transactions on Control Systems Technology, 2021.
[48] Defense Advanced Research Projects Agency (DARPA): DARPA's Offensive Swarm-Enabled Tactics (OFFSET) Program, https://www.darpa.mil/, dostęp 15.08.2024.
[49] Ebeid E. S. M., Skriver M., Laursen K. H., Jensen K.: A Survey of Open-Source UAV Flight Controllers and Flight Simulators, 2018, https://www.researchgate.net/publication/325134452_A_Survey_of_Open-Source_UAV_Flight_Controllers_and_Flight_Simulators, dostęp 06.09.2024.
[50] Ertam F., Aydın G.: Data classification with deep learning using Tensorflow, https://ieeexplore.ieee.org/document/8093521/, dostęp 30.09.2024.
[51] Fabro J. A., Guimarães R. L., Oliveira A. S., Becker T.: ROS Navigation: Concepts and Tutorial, 2016, https://www.researchgate.net/publication/302986850_ROS_Navigation_Concepts_and_Tutorial, dostęp 10.09.2024.
[52] Faiyaz A., Shivraj N. Y.: Design and Analysis of a Quadcopter, 2016, https://www.researchgate.net/publication/357832831_Design_and_Analysis_of_a_Quadcopter, dostęp 07.09.2024.
[53] Gao H., Li W., Cai H.: Fully Distributed Robust Formation Flying Control of Drones Swarm Based on Minimal Virtual Leader Information, w Drones, 2022, https://www.mdpi.com/2504-446X/6/10/266, dostęp 28.08.2024.

[54] Gatesichapakorn S., Takamatsu J., Ruchanurucks M.: ROS based Autonomous Mobile Robot Navigation using 2D LiDAR and RGB-D Camera, 2019, https://ieeexplore.ieee.org/document/8645984, dostęp 09.09.2024.

[55] Gibbs S.: Google's AI is being used by US military drone programme, w The Guardian, 2018, https://www.theguardian.com/technology/2018/mar/07/google-ai-us-department-of-defense-military-drone-project-maven-tensorflow, dostęp 29.08.2024.

[56] Gordon D., Ants At Work: How An Insect Society Is Organized, Cambridge, 2001.

[57] Grác Š., Beňo P., Duchoň F., Dekan M., Tölgyessy M.: Automated Detection of Multi-Rotor UAVs Using a Machine-Learning Approach, https://www.researchgate.net/publication/342942023_Automated_Detection_of_Multi-Rotor_UAVs_Using_a_Machine-Learning_Approach, dostęp 30.09.2024.

[58] Greiff M.: Modelling and Control of the Crazyflie Quadrotor for Aggressive and Autonomous Flight by Optical Flow Driven State Estimation, 2017, https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8905295&fileOId=8905299, dostęp 13.09.2024.

[59] Guan X., Lyu R., Shi H., Chen J.: A survey of safety separation management and collision avoidance approaches of civil UAS operating in integration national airspace system, w Chinese Journal of Aeronautics, 2020, https://www.sciencedirect.com/science/article/pii/S100093612030220X, dostęp 21.09.2024.

[60] Gui J., Yu T., Deng B., Zhu X., Yao W.: Decentralized Multi-UAV Cooperative Exploration Using Dynamic Centroid-Based Area Partition, https://www.mdpi.com/2504-446X/7/6/337, dostęp 28.08.2024.

[61] Han Y., Zhang X., Lai Z., Geng Y.: TOF-Based Fast Self-Positioning Algorithm for UWB Mobile Base Stations, w Sensors, 2021, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8512438, dostęp 21.09.2024.

[62] Hentati A. I., Krichen L., Mohamed F., Fourati L. C.: Simulation Tools, Environments and Frameworks for UAV Systems Performance Analysis, 2018, https://www.researchgate.net/publication/327323613_Simulation_Tools_Environments_and_Frameworks_for_UAV_Systems_Performance_Analysis, dostęp 09.09.2024.
[63] Herich D., Vaščák J.: Advancing Drone Swarm Simulation: A Comprehensive System for Real-Time Control and Dynamic Interaction, 2024, https://www.researchgate.net/publication/381037595_Advancing_Drone_Swarm_Simulation_A_Comprehensive_System_for_Real-Time_Control_and_Dynamic_Interaction, dostęp 29.08.2024.
[64] https://apxp.info/list/PL/, dostęp 21.09.2024.
[65] https://ardupilot.org/dev/docs/ardupilot-on-linux.html, dostęp 21.09.2024.
[66] https://ardupilot.org/dev/docs/code-overview-adding-a-new-mavlink-message.html, dostęp 21.09.2024.
[67] https://www.bitcraze.io/products/crazyradio-pa/, dostęp 21.09.2024.
[68] https://www.bitcraze.io/products/old-products/crazyflie-2-1/, dostęp 21.09.2024.
[69] https://www.bluetooth.com/learn-about-bluetooth/tech-overview/, dostęp 21.09.2024.
[70] https://dronekit.io/, dostęp 21.09.2024.
[71] https://www.flightgear.org/, dostęp 21.09.2024.
[72] https://gazebosim.org/docs/latest/architecture/, dostęp 21.09.2024.
[73] https://gazebosim.org/home, dostęp 21.09.2024.
[74] https://github.com/gsilano/CrazyS, dostęp 21.09.2024.
[75] https://github.com/PX4/jMAVSim, dostęp 21.09.2024.
[76] https://github.com/PX4/PX4-SITL_gazebo-classic, dostęp 21.09.2024.
[77] https://learn.microsoft.com/en-us/windows/wsl/install, dostęp 21.09.2024.
[78] https://index.ros.org/p/mavros/, dostęp 21.09.2024.

[79] https://www.mathworks.com/help/control/ref/pidtuner-app.html, dostęp 21.09.2024.
[80] https://www.mathworks.com/help/matlab/ref/ode45.html, dostęp 21.09.2024.
[81] https://www.mathworks.com/help/nav/ref/complementaryfilter.html, dostęp 21.09.2024.
[82] https://mavlink.io/en/, dostęp 21.09.2024.
[83] https://mavlink.io/en/guide/serialization.html, dostęp 21.09.2024.
[84] https://mavlink.io/en/mavgen_python, dostęp 21.09.2024.
[85] https://mavlink.io/en/messages/common.html, dostęp 21.09.2024.
[86] https://microsoft.github.io/AirSim/, dostęp 21.09.2024.
[87] https://openuav.eu/, dostęp 21.09.2024.
[88] http://qgroundcontrol.com/, dostęp 21.09.2024.
[89] https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/unmanned-aerial-vehicle, dostęp 21.09.2024.
[90] https://www.sciencedirect.com/topics/computer-science/swarm-intelligence, dostęp 21.09.2024.
[91] https://sketchfab.com/3d-models/crazyflie-2x-7b3ef11ebced4bfdb91bd756bfae46a5, dostęp 21.09.2024.
[92] https://www.tensorflow.org/, dostęp 21.09.2024.
[93] https://wiki.ros.org/sw_urdf_exporter, dostęp 21.09.2024.
[94] Hu X., Luo Z., Jiang W.: *AGV Localization System Based on Ultra-Wideband and Vision Guidance*, 2020, https://www.researchgate.net/publication/339790358_AGV_Localization_System_Based_on_Ultra-Wideband_and_Vision_Guidance, dostęp 21.09.2024.
[95] Huang S., Chen C., Wei T., Tsai W., Liou C., Mao Y., Sheng W., Mao S.: *Range-Extension Algorithms and Strategies for TDOA Ultra-Wideband Positioning System*, w *Sensors*, 2023, https://www.mdpi.com/1424-8220/23/6/3088, dostęp 21.09.2024.
[96] Huang Y., Tang J., Lao S.: *UAV Group Formation Collision Avoidance Method Based on Second-Order Consensus Algorithm and Improved Artificial Potential Field*, w *Symmetry*, 2019, https://www.mdpi.com/2073-8994/11/9/1162, dostęp 01.09.2024.

[97] Idrissi M., Salami M. Annaz F.: *A Review of Quadrotor Unmanned Aerial Vehicles: Applications, Architectural Design and Control Algorithms*, 2021, https://link.springer.com/content/pdf/10.1007/s10846-021-01527-7.pdf, dostęp 01.09.2024.

[98] Islam M., Okasha M., Idres M. M.: *Dynamics and control of quadcopter using linear model predictive control approach*, 2017, https://iopscience.iop.org/article/10.1088/1757-899X/270/1/012007/pdf, dostęp 07.09.2024.

[99] Iqbal M. M., Ali Z. A., Khan R., Shafiq M.: *Motion Planning of UAV Swarm: Recent Challenges and Approaches*, 2022, https://www.intechopen.com/chapters/82985, dostęp 29.08.2024.

[100] Jacobsen R. H., Matlekovic L., Shi L., Malle N., Ayoub N., Hageman K., Hansen S., Nyboe F. F., Ebeid E.: *Design of an Autonomous Cooperative Drone Swarm for Inspections of Safety Critical Infrastructure*, https://www.mdpi.com/2076-3417/13/3/1256, dostęp 29.08.2024.

[101] Jing Y., Wang X., Heredia-Juesas J., Fortner C.*: PX4 Simulation Results of a Quadcopter with a Disturbance-Observer-Based and PSO-Optimized Sliding Mode Surface Controller*, 2022, https://www.researchgate.net/publication/363734361_PX4_Simulation_Results_of_a_Quadcopter_with_a_Disturbance-Observer-Based_and_PSO-Optimized_Sliding_Mode_Surface_Controller, dostęp 09.09.2024.

[102] John R., Nathan A. A., Kurmi I., Bimber O.: *Drone swarm strategy for the detection and tracking of occluded targets in complex environments*, 2023, https://www.nature.com/articles/s44172-023-00104-0#auth-Rakesh_John-Amala_Arokia_Nathan-Aff1, dostęp 29.08.2024

[103] Jones K., Mercangoez M., Cortinovis A., Ferreau J.: *Distributed Model Predictive Control of Centrifugal Compressor Systems*, 2017, https://www.researchgate.net/publication/320497738_Distributed_Model_Predictive_Control_of_Centrifugal_Compressor_Systems, dostęp 11.09.2024.

[104] Jovanovic M., Starčević D. B., Obrenovic Z.: *UAV-based Simulation Environment: Experience Report*, 2007, https://www.researchgate.net/publication/224296956_UAV-based_Simulation_Environment_Experience_Report, dostęp 09.09.2024.
[105] Jung A.: *Machine Learning: The Basics*, Singapur, 2022.
[106] Kawamura K., Tanaka T.: *Improvement of Accuracy in Changing the Number of GPS Satellites*, https://dl.acm.org/doi/abs/10.1093/ietfec/e89-a.7.2092, dostęp 02.09.2024.
[107] Kennedy J., Eberhart R.: *Particle Swarm Optimization*, https://ieeexplore.ieee.org/document/488968, dostęp 17.08.2024.
[108] Khayat G., Mavromoustakis C., Pitsillides A., Batalla J. M.: *On the Weighted Cluster S-UAV Scheme Using Latency-Oriented Trust*, 2023, https://www.researchgate.net/publication/371260862_On_the_Weighted_Cluster_S-UAV_Scheme_Using_Latency-Oriented_Trust, dostęp 28.08.2024.
[109] Khelifi M., Butun I.: *Swarm Unmanned Aerial Vehicles (SUAVs): A Comprehensive Analysis of Localization, Recent Aspects, and Future Trends*, 2022, https://onlinelibrary.wiley.com/doi/10.1155/2022/8600674, dostęp 29.08.2024.
[110] Khoufi I., Laouiti A., Rachid A.: *Multi-Robot Coordination for Logistics Applications: A Review*, w *IEEE Transactions on Automation Science and Engineering*, 2019.
[111] Koubaa A.: *Robot Operating System (ROS). The Complete Reference (Volume 1)*, 2016, https://link.springer.com/book/10.1007/978-3-319-26054-9, dostęp 10.09.2024.
[112] Koubaa A.: Robot Operating System (ROS). The Complete Reference (Volume 2), 2017, https://link.springer.com/book/10.1007/978-3-319-54927-9, dostęp 10.09.2024.
[113] Koubaa A.: Robot Operating System (ROS). The Complete Reference (Volume 3), 2019, https://link.springer.com/book/10.1007/978-3-319-91590-6, dostęp 10.09.2024.
[114] Koubaa A.: Robot Operating System (ROS). The Complete Reference (Volume 4), 2020, https://link.springer.com/book/10.1007/978-3-030-20190-6, dostęp 10.09.2024.

[115] Koubaa A.: Robot Operating System (ROS). The Complete Reference (Volume 5), 2021,https://link.springer.com/book/10.1007/978-3-030-45956-7, dostęp 10.09.2024.

[116] Koubaa A.: Robot Operating System (ROS). The Complete Reference (Volume 6), 2021, https://link.springer.com/book/10.1007/978-3-030-75472-3, dostęp 10.09.2024.

[117] Koubaa A.: Robot Operating System (ROS). The Complete Reference (Volume 7), 2023, https://link.springer.com/book/10.1007/978-3-031-09062-2, dostęp 10.09.2024.

[118] Kshetrimayum R. S.: An introduction to UWB communication systems, 2009, https://ieeexplore.ieee.org/document/4803808, dostęp 02.09.2024.

[119] Kunze S, Weinberger A., Pöschl R.: A Software Defined Radio Based Implementation for the Radio Frequency Analysis of Signals from Unmanned Aerial Systems, 2019, https://www.researchgate.net/publication/344738029_A_Software_Defined_Radio_Based_Implementation_for_the_Radio_Frequency_Analysis_of_Signals_from_Unmanned_Aerial_Systems, dostęp 16.09.2024.

[120] Kwon S.G., Kwon O., Kwon K., Lee S.: UWB and MEMS IMU Integrated Positioning Algorithm for a Work-Tool Tracking System, 2021, https://www.researchgate.net/publication/354796664_UWB_and_MEMS_IMU_Integrated_Positioning_Algorithm_for_a_Work-Tool_Tracking_System, dostęp 21.09.2024.

[121] Ladcenko A.: Accuracy of static GPS-derived relative position vector, https://www.researchgate.net/profile/Andrej-Ladcenko/publication/309375769_Accuracy_of_static_GPS-derived_relative_position_vector/links/580bef5908ae2cb3a5da6f64/Accuracy-of-static-GPS-derived-relative-position-vector.pdf, dostęp 02.09.2024.

[122] Lee H.-I., Shin H.-S., Tsourdos A.: A Probabilistic–Geometric Approach for UAV Detection and Avoidance Systems, w Sensors, 2022, https://www.mdpi.com/1424-8220/22/23/9230, dostęp 06.09.2024.

[123] Li G., Zuo H., Xu J.: Research on the Influence of UAV Anti-collision Device on Aerodynamic Shape, w Journal of Physics Conference Series, 2023, https://www.researchgate.net/publication/370116269_Research_on_the_Influence_of_UAV_Anti-collision_Device_on_Aerodynamic_Shape, dostęp 21.09.2024.

[124] Li H., Li J., Guan X., Liang B., Lai Y., Luo X.: Research on Overfitting of Deep Learning, 2019, https://ieeexplore.ieee.org/document/9023664, dostęp 30.09.2024.

[125] Li J., Yue Q., Huang Z., Xie X.: Vulnerability Analysis of UAV Swarm Network with Emergency Tasks, w Electronics, 2024, https://www.mdpi.com/2079-9292/13/11/2005, dostęp 29.08.2024.

[126] Liao J., Cheng J., Xin B., Luo D., Zheng L., Kang Y., Zhou S.: UAV swarm formation reconfiguration control based on variable-stepsize MPC-APCMPIO algorithm, 2023, https://link.springer.com/article/10.1007/s11432-022-3735-5, dostęp 29.08.2024.

[127] Ling H., Luo H., Chen H., Bai L., Zhu T., Wang Y.: Modelling and Simulation of Distributed UAV Swarm Cooperative Planning and Perception, 2021, https://onlinelibrary.wiley.com/doi/10.1155/2021/9977262, dostęp 29.08.2024.

[128] Lissaman P. B. S., Shollenberger C. A.: Formation Flight of Birds, w Science, 1970, https://www.science.org/doi/10.1126/science.168.3934.1003, dostęp 30.09.2024.

[129] Liu X., Zheng S.: Research on Flight Inspection Using Unmanned Aircraf, 2019, https://www.researchgate.net/publication/334427025_Research_on_Flight_Inspection_Using_Unmanned_Aircraft, dostęp 30.09.2024.

[130] Lukina A., Kumar A., Schmittle M., Singh A., Das J., Rees S., Buskirk C. P., Sztipanovits J., Grosu R., Kumar V.: Formation Control and Persistent Monitoring in the OpenUAV Swarm Simulator on the NSF CPS-VO, 2018, https://publik.tuwien.ac.at/files/publik_275460.pdf, dostęp 09.09.2024.

[131] Luo B., Wang X., Zhang Z.: Application of Computer Vision Technology in UAV, https://iopscience.iop.org/article/10.1088/1742-6596/1881/4/042052, dostęp 06.09.2024.

[132] Luukkonen T.: Modelling and control of quadcopter, 2011, https://sal.aalto.fi/publications/pdf-files/eluu11_public.pdf, dostęp 07.09.2024.
[133] Madaan R., Gyde N., Vemprala S., Brown M., Nagami K., Taubner T., Cristofalo E., Scaramuzza D., Schwager M., Kapoor A.: AirSim Drone Racing Lab, 2020, https://arxiv.org/abs/2003.05654, dostęp 09.09.2024.
[134] Mairaj A., Baba A. I., Javaid A. Y.: Application specific drone simulators: Recent advances and challenges, 2019, https://www.sciencedirect.com/science/article/abs/pii/S1569190X19300048, dostęp 09.09.2024.
[135] Majchrzyk Ł.: Genesis wykorzystało w pokazie rekordowe 3281 dronów, 2021, https://mobirank.pl/2021/04/05/genesis-wykorzystalo-w-pokazie-rekordowe-3281-dronow/, dostęp 29.08.2024.
[136] Marek D., Paszkuta M., Szyguła J., Biernacki P., Domański A., Szczygieł M., Król M., Wojciechowski K.: Swarm of Drones in a Simulation Environment – Efficiency and Adaptation, w Appl. Sci., 2024, https://www.mdpi.com/2076-3417/14/9/3703, dostęp 27.08.2024.
[137] Megalingam R. K., Prithvi D. V., Kumar N. C. S., Egumadiri V.: Drone Stability Simulation Using ROS and Gazebo, 2021, https://www.researchgate.net/publication/353374509_Drone_Stability_Simulation_Using_ROS_and_Gazebo, dostęp 10.09.2024.
[138] Meier L., Honegger D., Pollefeys M.: PX4: A Node-Based Multithreaded Open Source Robotics Framework for Deeply Embedded Platforms, https://people.inf.ethz.ch/marc.pollefeys/pubs/MeierICRA15.pdf, dostęp 09.09.2024.
[139] Miller P.: Teoria Stada, https://www.national-geographic.pl/traveler/artykul/teoria-stada, dostęp 27.08.2024.
[140] Ming R., Jiang R. Luo H., Lai T., Guo E., Zhou Z.: Comparative Analysis of Different UAV Swarm Control Methods on Unmanned Farms, w Agronomy, 2023, https://www.mdpi.com/2073-4395/13/10/2499, dostęp 29.08.2024.

[141] Mizokami K.: New Ukrainian Rocket Launcher Appears to Use Raspberry Pi, 2016, https://www.popularmechanics.com/military/weapons/a23774/raspberry-pi-ukrainian-weapon-system/, dostęp 29.08.2024.

[142] Mousakazemi S. M. H.: Comparison of the error-integral performance indexes in a GA-tuned PID controlling system of a PWR-type nuclear reactor point-kinetics model, 2021, https://www.sciencedirect.com/
science/article/abs/pii/S0149197020303504, dostęp 11.09.2024.

[143] Naidoo Y. Stopforth R., Bright G.: Quad-Rotor Unmanned Aerial Vehicle Helicopter Modelling & Control, https://www.researchgate.net/publication/221915443_Quad-Rotor_Unmanned_Aerial_Vehicle_Helicopter_Modelling_Control, dostęp 01.09.2024.

[144] Naseri M., Shahid A., Poorter E.: Adapting UWB AoA estimation towards unseen environments using transfer learning and data augmentation, https://www.sciencedirect.com/science/article/abs/pii/S2542660524002397, dostęp 21.09.2024.

[145] NATO (The North Atlantic Treaty Organization): Swarm System for Intelligence Surveillance and Reconnaissance, https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.sto.nato.int/publications/STO%2520Technical%2520Reports/STO-TR-SET-263/%24%24TR-SET-263-ALL.pdf&ved=2ahUKEwjouK7VtbiHAxVyGhAIHaYzOhoQFnoECBcQAQ&usg=AOvVaw3mbbarZN0m_1rB1lIFhMks, dostęp 27.08.2024.

[146] Ni J. D., Refford M.: UWB Tracking Algorithms AOA and TDOA, https://ntrs.nasa.gov/api/citations/20060021507/downloads/20060021507.pdf, dostęp 04.09.2024.

[147] Nur H.: Research on Tensor Flow with a system for large-scale machine learning, 2022, https://www.researchgate.net/publication/363937209_Research_on_Tensor_Flow_with_a_system_for_large-scale_machine_learning, dostęp 30.09.2024.

[148] Novotňák J., Szőke Z., Kašper P., Šmelko M.: Quadcopter Modeling Using a System for UAV Parameters Measurement, w Drones, 2024, https://www.mdpi.com/2504-446X/8/7/280, dostęp 07.09.2024.

[149] Parker L. E.: Current State of the Art in Distributed Autnomous Mobile Robotics, https://www.researchgate.net/profile/Lynne-Parker/publication/221230010_Current_State_of_the_Art_in_Distributed_Autnomous_Mobile_Robotics/links/0deec52b1bf084ef14000000/Current-State-of-the-Art-in-Distributed-Autnomous-Mobile-Robotics.pdf, dostęp 17.08.2024.

[150] Peksa J., Mamchur D.: A Review on the State of the Art in Copter Drones and Flight Control Systems, w Sensors, 2024, https://www.mdpi.com/1424-8220/24/11/3349, dostęp 01.09.2024.

[151] Perez-Segui R., Arias-Perez P., Melero-Deza J., Fernandez-Cortizas M., Perez-Saura D., Campoy P.: Bridging the Gap between Simulation and Real Autonomous UAV Flights in Industrial Applications, w Aerospace, 2023, https://www.mdpi.com/2226-4310/10/9/814, dostęp 09.09.2024.

[152] Petitprez E., Guérin F., Guinand F., Germain F., Kerthe N.: Decentralized Coordination of a Multi-UAV System for Spatial Planar Shape Formations, w Sensors, 2023, https://www.mdpi.com/1424-8220/23/23/9553, dostęp 28.08.2024.

[153] Phadke A., Medrano F. A.: Towards Resilient UAV Swarms – A Breakdown of Resiliency Requirements in UAV Swarms, 2022, https://www.mdpi.com/2504-446X/6/11/340, dostęp 27.08.2024.

[154] Phadke A., Medrano F. A., Sekharan C. N., Chu T.: An analysis of trends in UAV swarm implementations in current research: simulation versus hardware, 2023, https://cdnsciencepub.com/doi/full/10.1139/dsa-2023-0099, dostęp 29.08.2024.

[155] Phadke A., Medrano F. A., Sekharan C. N., Chu T.: Designing UAV Swarm Experiments: A Simulator Selection and Experiment Design Process, w Sensors, 2023, https://www.mdpi.com/1424-8220/23/17/7359, dostęp 09.09.2024.

[156] Qamar S., Khan S. H., Arshad M. A., Qamar M., Khan A.: Autonomous Drone Swarm Navigation and Multi-target Tracking in 3D

Environments with Dynamic Obstacles, https://arxiv.org/pdf/2202.06253, dostęp 29.08.2024.
[157] Qu C., Boublin J., Gafurov D., Zhou J.: UAV Swarms in Smart Agriculture: Experiences and Opportunities, 2022, https://www.researchgate.net/publication/363195151_UAV_Swarms_in_Smart_Agriculture_Experiences_and_Opportunities, dostęp 29.08.2024.
[158] Qu C., Boubin J., Gafurov D., Zhou J., Aloysius N., Nguyen H., Calyam P.: UAV Swarms in Smart Agriculture: Experiences and Opportunities, 2022, https://ieeexplore.ieee.org/document/9973697, dostęp 29.08.2024.
[159] Quigley M.: ROS: an open-source Robot Operating System, https://www.semanticscholar.org/paper/ROS%3A-an-open-source-Robot-Operating-System-Quigley/d45eaee8b2e047306329e5dbfc954e6dd318ca1, dostęp 10.09.2024.
[160] Quigley M., Gerkey B., Conley K., Faust J., Foote T., Leibs J., Berger E., Wheeler R., Ng A.: ROS: an open-source Robot Operating System, http://www.robotics.stanford.edu/~ang/papers/icraoss09-ROS.pdf, dostęp 10.09.2024.
[161] Rejeb A., Rejeb K., Simske S. J., Treiblmaier H.: Drones for supply chain management and logistics: a review and research agenda, https://www.researchgate.net/publication/354832698_Drones_for_supply_chain_management_and_logistics_a_review_and_research_agenda, dostęp 29.08.2024.
[162] Reynolds C.: Flocks, Herds, and Schools: A Distributed Behavioral Model, California 1987, s. 25-34.
[163] Richardson L. A.: A Swarm of Bee Research, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5221816/, dostęp 27.08.2024.
[164] Rubenstein M., Cornejo A., Nagpal R.: Programmable Self-Assembly in a Thousand-Robot Swarm, w Science, 2013.
[165] Rubenstein M., Cornejo A., Nagpal R.: Robotics. Programmable self-assembly in a thousand-robot swarm, https://pubmed.ncbi.nlm.nih.gov/25124435/, dostęp 27.08.2024.

[166] Saffre F., Karvonen H., Hildmann H.: Wild Swarms: Autonomous Drones for Environmental Monitoring and Protection, 2024, https://cris.vtt.fi/en/publications/wild-swarms-autonomous-drones-for-environmental-monitoring-and-pr, dostęp 29.08.2024.

[167] Sahin E.: Swarm Robotics: From Sources of Inspiration to Domains of Application, 2005, https://www.researchgate.net/publication/225703852_Swarm_Robotics_From_Sources_of_Inspiration_to_Domains_of_Application, dostęp 27.08.2024.

[168] Sahin E., Spears W. M.: Swarm Robotics: SAB 2004 International Workshop, Santa Monica, 2005.

[169] Salwa M., Krzysztofik I.: Application of Filters to Improve Flight Stability of Rotary Unmanned Aerial Objects, 2022, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8875604/, dostęp 02.09.2024.

[170] Salwa M., Krzysztofik I.: Optimal Control for a Three-Rotor Unmanned Aerial Vehicle in Programmed Flights, w Appl. Sci., https://www.mdpi.com/2076-3417/13/24/13118, dostęp 01.09.2024.

[171] Sato T., Hayashi I., Horibe Y., Vilanova R.: Optimal Robust PID Control for First- and Second-Order Plus Dead-Time Processes, 2019, https://www.researchgate.net/publication/333046622_Optimal_Robust_PID_Control_for_First-_and_Second-Order_Plus_Dead-Time_Processes, dostęp 11.09.2024.

[172] Schilling M., Melnik A., Ohl F. W., Ritter H. J., Hammer B.: Decentralized control and local information for robust and adaptive decentralized Deep Reinforcement Learning, 2021, https://www.sciencedirect.com/science/article/pii/S0893608021003671, dostęp 28.08.2024.

[173] Schoonderwoerd R., Holland O. E., Rothkrantz L. J. M.: Ant-Based Load Balancing in Telecommunications Networks, https://journals.sagepub.com/doi/abs/10.1177/105971239700500203, dostęp 27.08.2024.

[174] Sciortino C., Fagiolini A.: ROS/Gazebo-Based Simulation of Quadcopter Aircrafts, 2018, https://ieeexplore.ieee.org/document/8548411, dostęp 10.09.2024.
[175] Sharma J., Mehra P. S.: Secure communication in IOT-based UAV networks: A systematic survey, https://www.sciencedirect.com/science/article/abs/pii/S2542660523002068, dostęp 01.09.2024.
[176] Shivakoti S.: UAVs communication over the Internet Of Things (IoT), 2021, https://openarchive.usn.no/usn-xmlui/handle/11250/2993313, 2021, dostęp 01.09.2024.
[177] Shule W., Almansa C. M., Queralta J. P., Zou Z., Westerlund T.: UWB-Based Localization for Multi-UAV Systems and Collaborative Heterogeneous Multi-Robot Systems, 2020, https://doi.org/10.1016/j.procs.2020.07.051, dostęp 02.09.2024.
[178] Silano G., Aucone E., Iannelli L.: CrazyS: A Software-In-The-Loop Platform for the Crazyflie 2.0 Nano-Quadcopter, 2018, https://ieeexplore.ieee.org/document/8442759, dostęp 10.09.2024.
[179] Simeone O.: A Brief Introduction to Machine Learning for Engineers, Foundations and Trends in Signal Processing, Londyn, 2018.
[180] Siwek M.: Modelowanie ruchu i synchroniczne sterowanie grupą robotów mobilnych o strukturze rozproszonej, Rozprawa doktorska na Wojskowej Akademii Technicznej im. Jarosława Dąbrowskiego, 2023, dostęp 10.09.2024.
[181] Sorton E. F., Hammaker S.: Simulated Flight Testing of an Autonomous Unmanned Aerial Vehicle Using FlightGear, https://web.archive.org/web/20060106015225id_/http://www.softwareresearch.org:80/pdfs/publishedpapers/Simulated%20Flight%20Testing%20of%20a%20UAV%20Using%20FlightGear.pdf, dostęp 09.09.2024.
[182] Spakovszky Z. S.: 16. Unified: Thermodynamics and Propulsion, http://web.mit.edu/16.unified/www/FALL/thermodynamics/notes/notes.html, dostęp 08.09.2024.
[183] Tahir A.: Formation Control of Swarms of Unmanned Aerial Vehicles,

https://www.researchgate.net/publication/374022555_Formation_Control_of_Swarms_of_Unmanned_Aerial_Vehicles, dostęp 27.08.2024.

[184] Tahir A., Böling J., Haghbayan M.-H., Toivonen H. T., Plosila J.: Swarms of Unmanned Aerial Vehicles – A Survey, https://www.sciencedirect.com/science/article/pii/S2452414X18300086, dostęp 28.08.2024.

[185] Telli K., Kraa O., Himeur Y., Ouamane A., Boumehraz M., Atalla S., Mansoor W.: A Comprehensive Review of Recent Research Trends on Unmanned Aerial Vehicles (UAVs), w Systems, 2023, https://www.mdpi.com/2079-8954/11/8/400, dostęp 11.09.2024.

[186] Thale S. P., Prabhu M. M., Thakur P., Kadam P.: ROS based SLAM implementation for Autonomous navigation using Turtlebot, 2020, https://www.researchgate.net/publication/343284410_ROS_based_SLAM_implementation_for_Autonomous_navigation_using_Turtlebot, dostęp 10.09.2024.

[187] Thoma A., Gardi A., Fisher A., Braun C.: Obstacle encounter probability dependent local path planner for UAV operation in urban environments, 2024, https://link.springer.com/article/10.1007/s13272-024-00746-6, dostęp 06.09.2024.

[188] Thu K. M., Gavrilova A. I.: Designing and Modeling of Quadcopter Control System Using L1 Adaptive Control, 2017, https://www.researchgate.net/publication/313464119_Designing_and_Modeling_of_Quadcopter_Control_System_Using_L1_Adaptive_Control, dostęp 07.09.2024.

[189] Thua K. M., Gavrilova A. I.: Designing and Modeling of Quadcopter Control System Using L1 Adaptive Control, 2017, https://www.researchgate.net/publication/313464119_Designing_and_Modeling_of_Quadcopter_Control_System_Using_L1_Adaptive_Control, dostęp 01.09.2024.

[190] Thuan P. N. , Queralta J. P. , Westerlund T.: Simulation Analysis of Exploration Strategies and UAV Planning for Search and Rescue, 2023, https://arxiv.org/abs/2304.05107, dostęp 09.09.2024.

[191] Topalli M. T., Yilmaz M., Corapsiz M. F.: Real Time Implementation of Drone Detection using TensorFlow and MobileNetV2-SSD, 2021, https://www.researchgate.net/publication/356670923_Real_Time_Implementation_of_Drone_Detection_using_TensorFlow_and_MobileNetV2-SSD, dostęp 30.09.2024.

[192] Wang N., Dai J., Ying J.: UAV Formation Obstacle Avoidance Control Algorithm Based on Improved Artificial Potential Field and Consensus, https://link.springer.com/article/10.1007/s42405-021-00407-6, dostęp 06.09.2024.

[193] Wang Z., Li J., Li J., Liu C.: A decentralized decision-making algorithm of UAV swarm with information fusion strategy, https://www.sciencedirect.com/science/article/abs/pii/S0957417423019462, dostęp 29.08.2024.

[194] Wei Z., Meng Z., Lai M., Wu H., Han J., Feng Z.: Anti-collision Technologies for Unmanned Aerial Vehicles: Recent Advances and Future Trends, 2022, https://arxiv.org/pdf/2109.12832, dostęp 21.09.2024.

[195] Wen T., Song Q., Gao Y.: The Analysis about UWB's EMI Cause to UAV System Based on Matlab/Simulink, 2006, https://ieeexplore.ieee.org/document/4027404, dostęp 02.09.2024.

[196] Wikipedia: Projekt Maven, https://en.wikipedia.org/wiki/Project_Maven, dostęp 29.08.2024.

[197] Wronowska K.: Ukraina zbiera drony. Każdy może przekazać swojego drona na potrzeby ukraińskiego wojska, w PAP, 2022, https://www.pap.pl/aktualnosci/news%2C1272717%2Cukraina-zbiera-drony-kazdy-moze-przekazac-swojego-drona-na-potrzeby, dostęp 29.08.2024.

[198] Wurman P. R., D'Andrea R., Mountz M.: Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses, w AI Magazine, 2008.

[199] Xiao K., Tan S., Wang G., An X., Wang X., Wang X.: XTDrone: A Customizable Multi-Rotor UAVs Simulation Platform, https://arxiv.org/pdf/2003.09700, dostęp 10.09.2024.

[200] Xu Y., Sengupta R. Hierarchical Configuration Management for a Multi-Robot System, w Autonomous Robots, 2003.

[201] Yang Z., Fang Z., Li P.: Decentralized 4D Trajectory Generation for UAVs Based on Improved Intrinsic Tau Guidance Strategy, w International Journal of Advanced Robotic Systems, 2016, https://www.researchgate.net/publication/303478330_Decentralized_4D_Trajectory_Generation_for_UAVs_Based_on_Improved_Intrinsic_Tau_Guidance_Strategy, dostęp 30.09.2024.

[202] Yangyang J., Yan G., Wenqi S., Yue L., Quan Q.: Bibliometric analysis of UAV swarms, 2022, https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9775048, dostęp 30.08.2024.

[203] Yasin J., Mohamed S.A.S., Haghbayan H., Heikkonen J.: Unmanned Aerial Vehicles (UAVs): Collision Avoidance Systems and Approaches, 2020, https://www.researchgate.net/publication/341918402_Unmanned_Aerial_Vehicles_UAVs_Collision_Avoidance_Systems_and_Approaches, dostęp 21.09.2024.

[204] Ye X., Song F., Zhang Z., Zeng Q.: A Review of Small UAV Navigation System Based on Multisource Sensor Fusion, 2023, https://ieeexplore.ieee.org/document/10179240, dostęp 01.09.2024.

[205] Yildiz B., Durdu A., Kayabasi A.: Consensus-based virtual leader tracking algorithm for flight formation control of swarm UAVs, w Turkish Journal of Electrical Engineering and Computer Sciences, 2024, https://www.researchgate.net/publication/379168336_Consensus-based_virtual_leader_tracking_algorithm_for_flight_formation_control_of_swarm_UAVs, dostęp 28.08.2024.

[206] Ying X.: An Overview of Overfitting and its Solutions, 2019, https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022/pdf, dostęp 30.09.2024.

[207] Yu S., Heo J., Jeong S., Kwon Y.: Technical Analysis of VTOL UAV, 2016, https://www.researchgate.net/publication/311159328_Technical_Analysis_of_VTOL_UAV, dostęp 01.09.2024.

[208] Yu Z., Wu Q., Wang J.: A Wireless Communication System of IoT Based on UAV, https://ieeexplore.ieee.org/document/9720479, dostęp 01.09.2024.

[209] Zaitseva E., Levashenko V., Brinzei N., Kovalenko A.: Reliability Assessment of UAV Fleets, 2023, https://www.researchgate.net/publication/369409989_Reliability_Assessment_of_UAV_Fleets, dostęp 30.09.2024.

[210] Zhang J., Zhang Y., Cong X., Zhao B., Zhang W.: Motion Control Method of UAV Buzzer Based on AirSim Platform, 2022, https://ieeexplore.ieee.org/document/10019949, dostęp 09.09.2024.

[211] Zhang Z., Zhu L.: A Review on Unmanned Aerial Vehicle Remote Sensing: Platforms, Sensors, Data Processing Methods, and Applications, w Drones, https://www.mdpi.com/2504-446X/7/6/398#:~:text=Among%20the%20sensors%20commonly%20used,sensors%2C%20small%20radars%2C%20etc, dostęp 06.09.2024.

[212] Zhao F., Zeng Y., Han B., Fang H., Zhao Z.: Nature-inspired self-organizing collision avoidance for drone swarm based on reward-modulated spiking neural network, https://www.sciencedirect.com/science/article/pii/S2666389922002367, dostęp 21.09.2024.

[213] Zhao J., Liu S., Li J.: Research and Implementation of Autonomous Navigation for Mobile Robots Based on SLAM Algorithm under ROS, w Sensors, 2022, https://www.mdpi.com/1424-8220/22/11/4172, dostęp 09.09.2024.

[214] Zhao X., Su Y., Hu T., Cao M., Liu X., Yang Q., Guan H., Liu L., Guo Q.: Analysis of UAV lidar information loss and its influence on the estimation accuracy of structural and functional traits in a meadow steppe, https://www.sciencedirect.com/science/article/pii/S1470160X21011808, dostęp 06.09.2024.

[215] Zhou K., Wang Z., Ni Y.-Q., Zhang Y., Tang J.: Unmanned aerial vehicle-based computer vision for structural vibration measurement and condition assessment: A concise survey, https://www.sciencedirect.com/science/article/pii/S2772991523000063, dostęp 06.09.2024.

[216] Zuo Y., Yao W., Chang Q., Zhu X., Gui J., Qin J.: Voting-Based Scheme for Leader Election in Lead-Follow UAV Swarm with

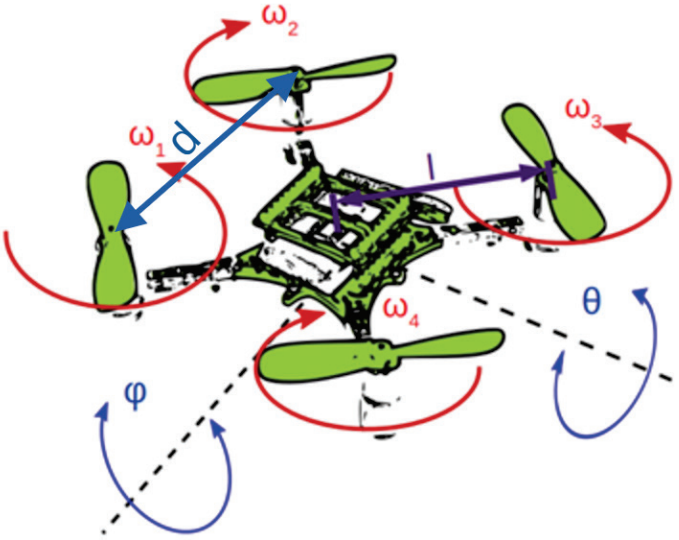
Constrained Communication, w Electronics, 2022, https://www.mdpi.com/2079-9292/11/14/2143, dostęp 28.08.2024.
[217] Zweibelson B.: Let Me Tell You about the Birds and the Bees: Swarm Theory and Military Decision-Making, 2015, https://css.ethz.ch/en/services/digital-library/articles/article.html/192366, dostęp 27.08.2024.

Załącznik nr 1

Parametry i model matematyczny drona Crazyflie 2.X

Parametry drona zostały pobrane ze strony producenta[68] oraz zmierzone. Poniżej przedstawiono wybrane dane:
m = 0.027 [kg] – masa drona,
l = 0.0046 [m] - odległość osi silnika od środka drona,
d = 0.0065 [m] - szerokość oraz wysokość mierzona od osi silników,
C = 250 [mAh] - pojemność baterii,
U = 3.7 [V] - napięcie znamionowe baterii,
x = 0.009 [m] - długość śmigła,
ω_{max} = 1750 [rad/s] - maksymalna prędkość silników,
k_V = 3800 [rpm/V] - stała silników.

Przyjęto układ współrzędnych zwrócony do góry, tzn. od ziemi. Prędkości obrotowe poszczególnych silników oznaczone zostały jako $\omega_{1\text{-}4}$. Kąty Eulera ϕ, θ, ψ reprezentują przechylenie, pochylenie i odchylenie. Oznaczenia zostały umieszczone rysunku poniżej:



Rys 1.1. Widok drona Crazyflie 2.X [źródło własne]

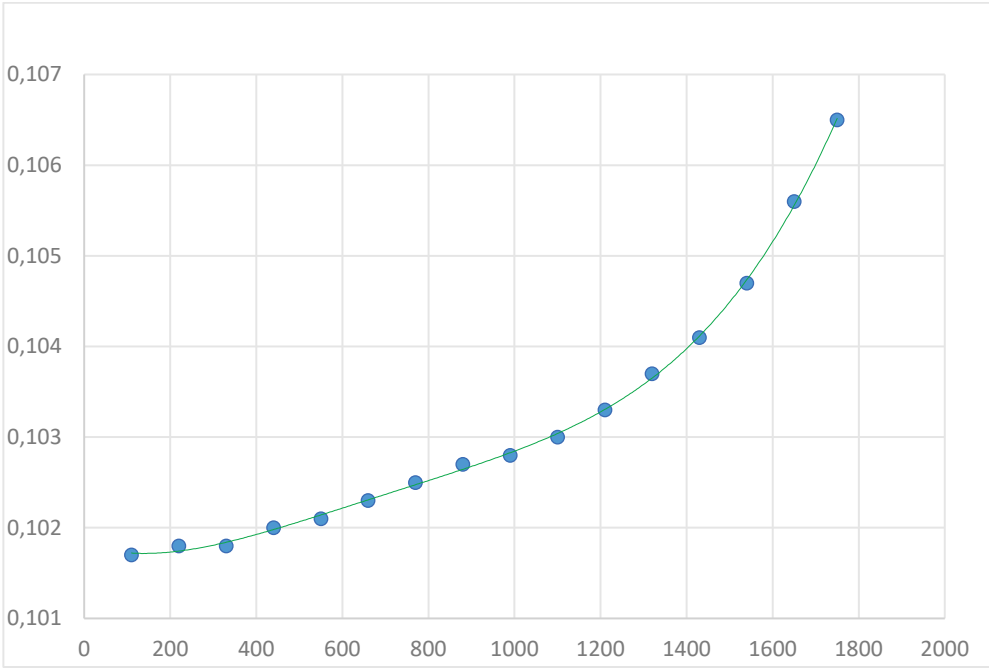
Momenty bezwładności zostały wyznaczone przy użyciu programu SolidWorks i wynoszą kolejno:
I_{xx} = 1,657171 · 10^{-5} kg·m²,
I_{yy} = 1,657171· 10^{-5} kg·m²,
I_{zz} = 2,9261652· 10^{-5} kg·m².

Obecnie najczęściej używanymi silnikami do budowy dronów są silniki bezszczotkowe (BLDC). Z racji na gabaryty Crazyflie w tym quadrocopterze użyto silników szczotkowych, które nie wymagają dodatkowych sterowników przez co cały układ jest lżejszy. Producenci podają stałą k_V określającą z jaką prędkością kręci się silnik w zależności od podanego napięcia. Charakterystyka pracy silników bez obciążenia jest zbliżona do liniowej. Założenie, że ciąg i moment skręcający generowane przez śmigła zmieniają się liniowo wraz z przyłożonym napięciem, jest dużym uproszczeniem modelu. Najlepszym sposobem wyznaczenia charakterystyk jest przeprowadzenie badań na obiekcie fizycznym. Dzięki nim można uzyskać potrzebne współczynniki. Zarówno współczynnik ciągu, jak i momentu skręcającego, zależne są od prędkości obrotowej silnika, jak i prędkości obiektu. W dalszej części obliczono korelacje współczynnika ciągu w stosunku do prędkości obrotowej silników. Wpływ prędkości drona został pominięty.
Prof. Z. S. Spakovsky w swojej pracy[182] wyprowadził wzór na siłę ciągu generowaną przez śmigła. Jego zapis przedstawia się następująco:

$$T = k_T \cdot \rho \cdot n^2 \cdot x^4 \tag{1.1}$$

gdzie:
k_T – współczynnik ciągu,
ρ – gęstość powietrza,
n – prędkość obrotowa,

Do aproksymacji k_T posłużono się metodą krzywej regresji. Wybrano rząd wielomianu równy 5. Na rysunku 1.2 przedstawiono oryginalne dane, oznaczone punktami o kolorze niebieskim, oraz aproksymowaną funkcje, oznaczoną kolorem zielonym.



Rys. 1.2. Charakterystyka współczynnika ciągu od prędkości obrotowej silnika [źródło własne]

Na rysunku 1.2 na osi poziomej przedstawiono prędkości obrotowe wyrażone w radianach na sekundę[rad/s]. Dla każdej z tych prędkości zbadano współczynnik ciągu (przedstawiony na osi pionowej) wyrażony w niutonach[N], zgodnie ze wzorem:

$$k_T = \frac{T}{\rho n^2 x^4} \tag{1.2}$$

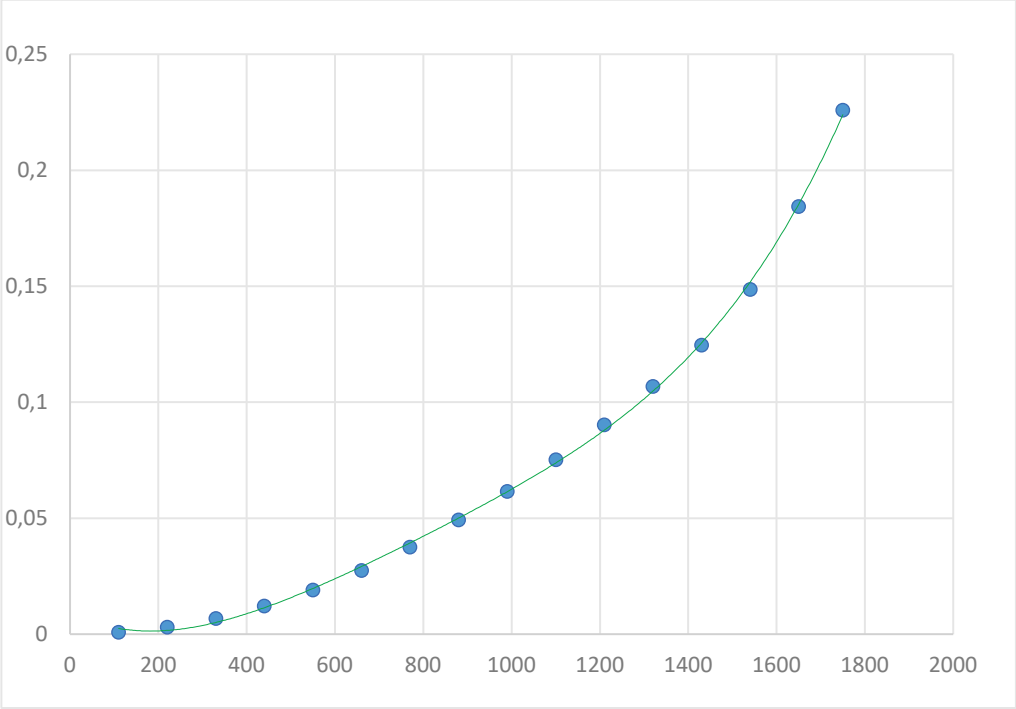
gdzie:

T – ciąg wygenerowany przez pojedynczy silnik.

Aproksymując wielomianem 5 rzędu, uzyskano funkcję wynoszącą:

$$k_T(n) = -3 \cdot 10^{-19} n^5 + 3 \cdot 10^{-15} n^4 - 7 \cdot 10^{-12} n^3 + 7 \cdot 10^{-9} n^2 - 2 \cdot 10^{-6} n + 0.1018 \tag{1.3}$$

Podobnie na podstawie zebranych danych wyznaczono funkcje dla momentu skręcającego:



Rys. 1.3. Moment skręcający od prędkości obrotowej silnika [źródło własne]

Na rysunku 1.3 na osi X przedstawiono prędkości obrotowe wyrażone w radianach na sekundę[rad/s]. Dla każdej z tych prędkości zbadano moment skręcający (przedstawiony na osi Y) wyrażony w niutonometrach[Nm].

Aproksymując wielomianem 4 rzędu, uzyskano funkcję wynoszącą:

$$Q(n) = 8 \cdot 10^{-14}n^4 - 3 \cdot 10^{-10}n^3 + 3 \cdot 10^{-7}n^2 - 9 \cdot 10^{-5}n + 0.0092 \quad (1.4)$$

gdzie:
Q - moment skręcający.

W silnikach prądu stałego moment skręcający oraz natężenie są proporcjonalne według następującej formuły:

$$Q = K_I \cdot I \tag{1.5}$$

gdzie:
K_I - stała momentu wyrażona w $\mathrm{Nm/A}$, będącą odwrotnością współczynnika k_V,
I – natężenie prądu pobieranego przez silnik.

Analogicznie do 1.4 wzór na pobierany prąd wynosi:

$$I = k_V \cdot Q \tag{1.6}$$

Zgodnie z drugą zasadą dynamiki Newtona dla ruchu obrotowego można zapisać trzy równania ruchu kulistego:

$$\begin{aligned} M_x &= I_{xx} \cdot \ddot{\phi} \\ M_y &= I_{yy} \cdot \ddot{\theta} \\ M_z &= I_{zz} \cdot \ddot{\psi} \end{aligned} \tag{1.7}$$

gdzie:
M_x, M_y, M_z – moment obrotowy działający wokół danej osi,
$\ddot{\phi}, \ddot{\theta}, \ddot{\psi}$ – przyśpieszenie kątowe działające wokół danej osi.

Kąt pochylenia uzyskiwany jest w quadrocopterze przez różnicę w ciągu między silnikami pierwszym oraz drugim, a silnikami trzecim oraz czwartym. Kąt przechylenia realizowany jest przez różnicę w ciągu między silnikami czwartym oraz pierwszym, a silnikami drugim oraz trzecim. Kąt odchylenia jest realizowany przez różnicę w momentach skręcających między silnikami czwartym oraz drugim, a silnikami pierwszym i trzecim. Siła grawitacji przyłożona jest w środku masy, przez co nie wpływa na momenty siły. Można zapisać je następująco zapisać następująco:

$$M_x = (T_3 + T_4) \cdot \left(\frac{d}{2}\right) - (T_1 + T_2) \cdot \left(\frac{d}{2}\right)$$

$$M_y = (T_2 + T_3) \cdot \left(\frac{d}{2}\right) - (T_1 + T_4) \cdot \left(\frac{d}{2}\right) \tag{1.8}$$

$$M_z = Q_2 + Q_4 - (Q_1 + Q_3)$$

Na podstawie wzorów 1.7 i 1.8 możemy wyznaczyć równania ruchu kulistego:

$$\ddot{\phi} = \frac{d}{2 \cdot I_{xx}} \cdot [(T_3 + T_4) - (T_1 + T_2)]$$

$$\ddot{\theta} = \frac{d}{2 \cdot I_{yy}} \cdot [(T_2 + T_3) - (T_1 + T_4)] \tag{1.9}$$

$$\ddot{\psi} = \frac{Q_2 + Q_4 - (Q_1 + Q_3)}{I_{zz}}$$

Zgodnie z drugą zasadą dynamiki Newtona dla ruchu obrotowego można zapisać trzy równania ruchu liniowego:

$$F_x = m \cdot \ddot{x}$$

$$F_y = m \cdot \ddot{y} \tag{1.10}$$

$$F_z = m \cdot \ddot{z}$$

gdzie:
F_x, F_y, F_z – wypadkowa sił działających w danej osi,
$\ddot{x}$, $\ddot{y}$, $\ddot{z}$ – przyśpieszenie układu w danej osi.

Zgodnie z przyjętym układem współrzędnych siły działające w poszczególnych osiach możemy rozpisać jako:

$$F_x = F_{T_x} \pm D_x$$

$$F_y = F_{T_y} \pm D_y \tag{1.11}$$

$$F_z = F_{T_z} \pm D_z - m \cdot g$$

gdzie:
$F_{T_x}, F_{T_y}, F_{T_z}$ – siła generowana przez ciąg silnika działająca w danej osi,
D_x, D_y, D_z – zakłócenia działające w danej osi powodowane siłą wiejącego wiatru.
Zakłócenia zamodelowane zostały według następującego wzoru:

$$D = \frac{1}{2} \cdot \rho \cdot (V_{wiatru})^2 \cdot A \cdot k_d \tag{1.12}$$

gdzie:
V_{wiatru} – prędkość wiatru wiejącego w danej osi,
A – powierzchnia oporu,
k_d – współczynnik oporu powierzchni.

Siły generowane przez ciąg, działające w danych osiach, są rozkładem wektora siły ciągu na poszczególne osie układu. Znając kąty pochylenia i przechylenia można zapisać poniższe równania:

$$\begin{aligned} F_{T_x} &= cos(\phi) \cdot sin(\theta) \cdot (T_1 + T_2 + T_3 + T_4) \\ F_{T_y} &= sin(\phi) \cdot cos(\theta) \cdot (T_1 + T_2 + T_3 + T_4) \\ F_{T_z} &= cos(\phi) \cdot cos(\theta) \cdot (T_1 + T_2 + T_3 + T_4) \end{aligned} \tag{1.13}$$

Na podstawie 1.10, 1.11 i 1.13 można sformułować równania ruchu liniowego:

$$\begin{aligned} \ddot{x} &= \frac{cos(\phi) \cdot sin(\theta) \cdot (T_1 + T_2 + T_3 + T_4) \pm D_x}{m} \\ \ddot{y} &= \frac{sin(\phi) \cdot cos(\theta) \cdot (T_1 + T_2 + T_3 + T_4) \pm D_y}{m} \\ \ddot{z} &= \frac{cos(\phi) \cdot cos(\theta) \cdot (T_1 + T_2 + T_3 + T_4) \pm D_z}{m} - g \end{aligned} \tag{1.14}$$

Załącznik nr 2

Porównanie symulatorów bezzałogowych obiektów latających

Gazebo [73] to jeden z najbardziej znanych symulatorów robotyki, który jest często używany do symulacji dronów. Oferuje realistyczne modelowanie fizyki, rendering 3D, a także możliwość integrowania czujników. Gazebo jest kompatybilny z systemem ROS (*Robot Operating System*), co czyni go wszechstronnym narzędziem do badań nad autonomicznymi systemami UAV. W kontekście roju dronów, Gazebo umożliwia symulację wielu jednostek w czasie rzeczywistym oraz badanie ich interakcji.

AirSim [86] to symulator stworzony przez Microsoft Research, specjalnie zaprojektowany do testowania i trenowania autonomicznych pojazdów, w tym dronów. Jest zbudowany na silniku Unreal Engine, co zapewnia wysoką jakość grafiki i realistyczne środowiska. AirSim oferuje możliwość symulacji różnych warunków pogodowych, co jest kluczowe dla testowania systemów UAV w różnych scenariuszach. Chociaż AirSim jest głównie wykorzystywany do indywidualnych symulacji dronów, istnieje możliwość rozszerzenia go na symulacje roju.

FlightGear [71] to wieloplatformowy, otwartoźródłowy symulator lotów, który został stworzony głównie do symulacji samolotów, ale jest również używany w kontekście dronów. Jego wszechstronność, dokładność i możliwość rozbudowy sprawiają, że jest popularny wśród entuzjastów lotnictwa, badaczy oraz w edukacji.

PX4-SITL [76] to platforma symulacyjna dla autopilota PX4, używana do testowania i rozwoju oprogramowania dronów. W połączeniu z Gazebo, PX4 SITL pozwala na realistyczne symulacje lotów dronów, w tym też dla roju UAV. SITL umożliwia uruchomienie tego samego kodu, który działa na rzeczywistych dronach, co pozwala na testowanie algorytmów bez potrzeby posiadania fizycznych jednostek.

X-Plane [64] to zaawansowany symulator robotyki, który wspiera szeroką gamę robotów, w tym drony. Dzięki elastycznemu systemowi skryptów i wsparciu dla różnych języków programowania, V-REP jest często używany

do symulacji wielorobotycznych, w tym rojów dronów. Symulator ten oferuje zaawansowane funkcje, takie jak symulacje czasu rzeczywistego oraz złożone modele dynamiki.

OpenUAV [87] to platforma symulacyjna do badania i rozwijania systemów bezzałogowych pojazdów latających (UAV) w środowisku chmurowym. Umożliwia symulację dronów z wykorzystaniem infrastruktury chmurowej, co pozwala na skalowanie symulacji i uruchamianie wielu instancji jednocześnie.

jMAVSim [75] to lekki symulator stworzony z myślą o testowaniu i rozwijaniu oprogramowania kontrolerów lotu, w szczególności w ekosystemie PX4. Dzięki integracji z trybem SITL (*Software-in-the-Loop*). jMAVSim używa Java OpenGL (JOGL) do renderowania grafiki, co czyni go prostym, ale wystarczającym programem dla podstawowych symulacji dronów. Nie jest tak zaawansowany graficznie jak inne symulatory, ale jest dobrze zoptymalizowany pod kątem szybkiego testowania.

Podstawowe dane porównawcze zebrano w tabeli 2.1.

Tab. 2.1. Porównanie 7 programów używanych do symulacji UAV w kolejności przedstawionej powyżej [źródło własne]

Open Source	**Silnik Graficzny**	**Sposób Komunikacji**	**Opóźnienia**	**Importowanie Modeli CAD**	**Obsługa PX4/ArduPilot**
Tak	Gazebo Engine	ROS, MAVLink	Niskie	URDF, SDF, Collada	Tak
Tak	Unreal Engine	RPC, ROS	Średnie	FBX, OBJ	Tak
Tak	OSG (OpenSceneGraph)	Socket, HTTP	Średnie	AC3D, STL, OBJ	Tak (z ograniczeniami)
Tak	Gazebo Engine	MAVLink, ROS	Niskie	SDF, URDF	Tak

Nie	Proprietary	Socket, UDP	Niskie	OBJ, WED	Tak (z ograniczeniami)
Tak	Gazebo Engine	MAVLink, ROS	Niskie	SDF, URDF	Tak
Tak	Java OpenGL (JOGL)	MAVLink	Niskie	Brak wsparcia dla zaawansowanych modeli CAD	Tylko PX4

Oprócz Gazebo, będącego podstawowym programem, w tabeli przedstawiono, że dwa inne symulatory są formą nakładki na program Gazebo: PX4-SITL oraz OpenUAV. Oferują one dodatkowe możliwości i są dedykowane dla bezzałogowych obiektów latających. Jednak mogą stanowić też niepotrzebną nadbudowę oprogramowania, utrudniającą korzystanie z pełnego potencjału programu Gazebo.

Załącznik nr 3

Szablon misji wykonywanej przez rój

```
<?xml version="1.0" encoding="UTF-8"?>
<Mission>
  <Swarm>
    <InitialDroneCount>5</InitialDroneCount>
    <Formation>
      <Shape>2D</Shape> <!-- Domyślnie 2D -->
      <RequestedFormation>Multi-Row</RequestedFormation> <!-- Domyślnie
wielorzędowa -->
      <RowCount>1</RowCount> <!-- Domyślnie jeden rząd -->
      <Width>10</Width> <!-- Szerokość formacji -->
      <Depth>5</Depth> <!-- Głębokość formacji -->
      <RowSpacing>2</RowSpacing> <!-- Odstęp między rzędami -->
      <UnitSpacing>1.5</UnitSpacing> <!-- Odstęp między jednostkami w
rzędzie -->
      <LeaderDrone>1</LeaderDrone> <!-- Numer drona będącego liderem -->
    </Formation>
    <MissionType>Monitoring</MissionType> <!-- Typ misji - domyślnie
monitorowanie -->
    <CommunicationType>Radio</CommunicationType> <!-- Typ komunikacji
z matką - domyślnie radiowo -->
  </Swarm>

  <Trajectory>
    <Waypoints>
      <Waypoint>
        <Position>
          <X>0</X>
          <Y>0</Y>
          <Z>10</Z>
        </Position>
```

```
            <Movement>Linear</Movement> <!-- Sposób przemieszczania się -
domyślnie po linii -->
            <Speed>5</Speed> <!-- Prędkość do osiągnięcia między punktami -->
          </Waypoint>
          <Waypoint>
            <Position>
              <X>50</X>
              <Y>0</Y>
              <Z>10</Z>
            </Position>
            <Movement>Linear</Movement>
            <Speed>5</Speed>
          </Waypoint>
          <!-- Dodaj kolejne punkty jeśli potrzebne -->
        </Waypoints>
    </Trajectory>

    <MissionControl>
        <RepeatMission>False</RepeatMission> <!-- Powtarzanie misji lub
zakończenie po wykonaniu -->
        <RepeatInterval>0</RepeatInterval> <!-- Interwał powtarzania misji (jeśli
powtarzana) -->
    </MissionControl>

    <DataCollection>
        <CollectedData>SensorData</CollectedData> <!-- Dane zbierane podczas
misji - domyślnie czujniki -->
    </DataCollection>

    <ThresholdConditions>
        <Condition>
            <ThresholdPoint>50,0,10</ThresholdPoint> <!-- Punkt treshold (X,Y,Z) --
>
            <ConditionType>BatteryLow</ConditionType> <!-- Warunek, np. niski
poziom baterii -->
            <Action>ReturnToBase</Action> <!-- Akcja po wystąpieniu warunku -->
        </Condition>
```

```
    <!-- Dodaj kolejne warunki jeśli potrzebne -->
  </ThresholdConditions>
</Mission>
```

W podanym szablonie zadawania misji dodano komentarze na poziomie kodu informujące o przeznaczeniu poszczególnych pól. Dodatkowo poniżej przedstawiono uproszczoną analizę struktury pliku:

1. Swarm:
InitialDroneCount: liczba dronów w roju.
Formation: parametry formacji: kształt, liczba rzędów, szerokość, głębokość itp.
LeaderDrone: ID lidera roju.

2. Trajectory:
Waypoints: lista punktów na trasie.
Movement: sposób przemieszczania się między punktami. Liniowy(linear) lub inny typ ruchu.
Speed: prędkość między punktami.

3. MissionControl:
Opcje powtarzania misji oraz czas powtarzania.

4. DataCollection:
Dane, które są zbierane w trakcie misji np. dane z czujników.

5.ThresholdConditions:
Warunki, które wywołają określone akcje np. niski poziom baterii lub odebranie żądanej informacji przez jednego z dronów.

Załącznik nr 4

Algorytmika procesu uczenia przez wzmocnienie

Aby optymalizować politykę w uczeniu przez wzmacnianie wykorzystywane są następujące metody i kroki:

1. Zdefiniowanie problemu jako Markov Decision Process (MDP)
Podstawą RL jest formalizacja problemu jako Markov Decision Process (MDP). Kluczowe elementy z których się składa to:

- Zbiór stanów, **s** - stany w których może znajdować się agent właściwe rozpatrywanemu przypadkowi,
- Zbiór akcji, **a** – działania które agent może podjąć,
- Funkcja przejścia stanów, **p (s' | s, a)** - opisuje prawdopodobieństwo przejścia do stanu s′ po podjęciu akcji a w stanie s,
- Funkcja nagrody, **R (s, a)** - definiuje wartość nagrody za podjęcie akcji a w stanie s,
- Współczynnik dyskontowania, **γ** - określa, jak bardzo przyszłe nagrody są ważne w porównaniu do natychmiastowych nagród.

2. Ustalenie funkcji wartości stanu
Funkcja wartości stanu ocenia, jak dobra jest polityka agenta w danym stanie. Jest to oczekiwana suma zdyskontowanych nagród, które agent może uzyskać, startując ze stanu s i działając zgodnie z polityką określaną jako **π**.

$$V_\pi(s) = E_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s] \qquad (4.1)$$

gdzie:
$V_\pi(s)$: Funkcja wartości stanu dla polityki π,
γ∈[0,1]: Współczynnik dyskontowania (bliski wartości 1 premiuje długoterminowe korzyści).

3. Ustalenie funkcji wartości akcji

Jest to podobne do funkcji wartości stanu, ale zamiast stanu oceniamy akcję w danym stanie. Oczekiwana zdyskontowana suma nagród, jeśli agent zaczyna w stanie s, podejmuje akcję a i postępuje zgodnie z polityką π.

$$Q_\pi(s,a) = E_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a] \quad (4.2)$$

gdzie:

$Q_\pi(s,a)$ - funkcja wartości stanu i akcji dla polityki π.

4. Ustalenie optymalnej funkcji wartości

W optymalnej polityce określanej jako **π*,** gdzie „*" jest oznaczeniem optymalności, agent wybiera akcje, które maksymalizują długoterminową sumę nagród. Funkcja wartości dla stanu lub akcji przy optymalnej polityce jest określona wzorami:

Dla stanu:

$$V^*(s) = \max(\pi) * V_\pi(s) \quad (4.3)$$

Dla akcji:

$$Q^*(s,a) = \max(\pi)\, Q_\pi(s,a) \quad (4.4)$$

5. Dobór równania Bellmana

Równanie Bellmana rozkłada problem oceny wartości stanu na sumę nagrody natychmiastowej i oczekiwanej wartości przyszłego stanu.

Dla funkcji wartości stanu Vπ(s):

$$V\pi(s) = E\pi[R(s,a) + \gamma V\pi(s') \mid s] \quad (4.5)$$

Dla funkcji wartości akcji Qπ(s,a):

$$Q\pi(s,a) = R(s,a) + \gamma Es'[V\pi(s')] \quad (4.6)$$