

WOJSKOWA AKADEMIA TECHNICZNA
im. Jarosława Dąbrowskiego

WYDZIAŁ CYBERNETYKI



PRACA DYPLOMOWA
STUDIA II^O

Temat pracy: **MOBILNY SYSTEM ZARZĄDZANIA I STEROWANIA BEZPILOTOWYM STATKIEM LATAJĄCYM**

INFORMATYKA

(kierunek studiów)

SYSTEMY INFORMATYCZNE

(specjalność)

Diplomant:

Norbert WASZKOWIAK

Promotor:

dr inż. Michał DYK

Warszawa 2022

OŚWIADCZENIE

*Wyrażam zgodę / nie wyrażam zgody *
na udostępnianie mojej pracy przez Archiwum WAT*

Dnia

(podpis)

* Niepotrzebne skreślić

Spis treści

Wstęp	5
Rozdział I. Prezentacja zagadnienia bezpilotowych statków latających oraz koncepcji ich wykorzystania	6
I.1. Definicja BSP	6
I.2. Historia BSP	6
I.3. Technologia i producenci BSP	11
I.4. Zastosowania BSP	13
I.5. Dostosowywanie BSP do wymagań użytkownika	16
Rozdział II. Przegląd i prezentacja technologii mobilnych z uwzględnieniem aspektów tworzenia aplikacji i komunikacji M2M	18
II.1. Komunikacja M2M	18
II.2. Technologie komunikacji M2M	19
II.3. Komunikacja bezprzewodowa w dronach konsumenckich	22
II.4. Kontrolowanie BSP za pomocą API dostarczonego od producenta	27
Rozdział III. Projekt mobilnego systemu zarządzania i sterowania BSP	29
III.1. Wymagania funkcjonalne	29
III.2. Wymagania pozafunkcjonalne	29
III.3. Stos technologiczny	29
III.4. Wysokopoziomowy diagram systemu	29
III.5. Diagram komponentów	30
III.6. Diagram klas i sekwencji	31
III.7. Wykorzystane urządzenia	37
Rozdział IV. Implementacja systemu	38
IV.1. Klasa komend i jej wykonywanie	38
IV.2. Obsługa MQTT	40
Rozdział V. Testy systemu oraz prezentacja użycia na wybranym case study	43
V.1. Testy jednostkowe TODO	43
V.2. Testy autonomiczne	46
V.3. Coś o sprawdzeniu statusu statku powietznego	48
Podsumowanie	49
Bibliografia	50
Spis rysunków	53

Spis tabel	53
Załączniki	54

Wstęp

Rozdział I. Prezentacja zagadnienia bezpilotowych statków latających oraz koncepcji ich wykorzystania

I.1. Definicja BSP

W nomenklaturze związanej z domeną bezpilotowych statków latających można znaleźć wiele tożsamy terminów na określanie bezpilotowych statków latających, są to m.in.:

- Bezzałogowy statek powietrzny, BSP (ang. *unmanned aerial vehicle*, UAV);
- Bezzałogowy system powietrzny (ang. *unmanned aerial system*, UAS);
- Samolot zdalnie sterowany (ang. *remotely piloted aircraft*, RPA);
- Dron (ang. *drone*),

Każdy z tych terminów kładzie nacisk na inną cechę, ale wszystkie nadal odnoszą się do jednego obiektu i będą w tej pracy używane zamiennie.

Amerykański pisarz zajmujący się zagadnieniami systemów bezzałogowych i technologii obronnych, Kelsey Artheon na łamach czasopisma *Popular Science* definiuje to pojęcie następująco: "dron oznacza każdy bezzałogowy zdalnie sterowany pojazd latający, bez względu na to, czy jest to malutki, sterowany radiem helikopter-zabawka, czy też ważący 14,5 tony Global Hawk, wart 104 mln dolarów. Jeżeli coś lata i jest sterowane przez pilota z ziemi, to pasuje do potocznej definicji drona".¹ Biorąc to pod uwagę, można zdefiniować następujące warunki do zakwalifikowania obiektu jako bezzałogowy statek powietrzny:

- **bezpilotowość** - na swoim pokładzie nie posiada pilota;
- **dwukierunkowość** - musi mieć możliwość powrotu/wylądowania (Jest to podstawowa cecha odróżniająca drony od pocisków manewrujących);
- **sterowalność** - możliwość zmiany kierunku lotu w trakcie jego wykonywania.[1][2]

I.2. Historia BSP

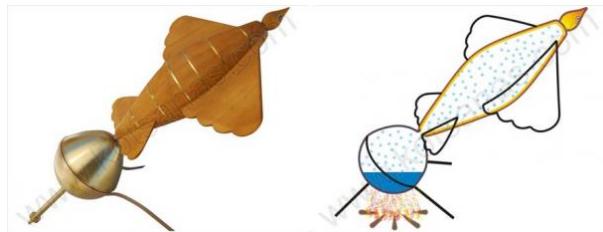
Po przedstawieniu definicji BSP można przystąpić do przedstawienia historii całej tej domeny, wraz ze wskazaniem jej początku w poprawny sposób.

I.2.1. Błędnie klasyfikowane obiekty

Po zdefiniowaniu czym jest dron, można się zastanowić, co było pierwszym elementem spełniającym tę definicję. Autor tej pracy uważa, że kluczowym elementem umożliwiającym zakwalifikowanie obiektu jako dron jest możliwość zmiany trajektorii lotu w trakcie jego działania. W literaturze często wskazywane są dwa obiekty jako prekursory dronów, tzn. gołąb Archytasa z Tarentu i balony zawierające ładunki wybuchowe

¹A. Kelsey, *Flying Robots 101: Everything You Need to Know about Drones*, Popular Science[2]

wykorzystane w konflikcie między Austrią i Wenecją w 1849 r. Pierwszy rzekomy prekursor nie umożliwia sterowania obiektem po jego wystartowaniu, więc tym samym nie jest to zgodne z przytoczonymi definicjami. Ten wynalazek można uznać za pierwszą rakietę lub robota, ale nie drona. Drugi przykład, czyli balony na gorące powietrze, również nie mogą zostać uznane za bezzałogowy statek powietrzny z tego samego powodu. Jako ciekawostkę można dodać, że pomysł Austriaków zakończył się niepowodzeniem, ponieważ wiatr związał balony na ich własne pozycje.[1].



Rys. 1. Latający gołąb Archytasa z Tarentu

Źródło: <https://input.niezalezna.pl/259fef5fd.jpg>

I.2.2. Pierwszy pełnoprawny dron

Podczas I wojny światowej podjęto liczne próby skonstruowania bezpilotowych statków latających, ale żaden z nich nie został ukończony przed skończeniem wojny. Przykładowo *Kettering Bug*, był w stanie dolecieć na odpowiednią odległość, ale jego sterowanie polegało na wyliczeniu przez operatora dokładną liczbę obrotów silnika. Mając to na uwadze, takiemu samolotowi bliżej do torpedy niż do drona.

W 1931 r. Królewskie Siły Powietrzne (ang. Royal Air Force) na podstawie samolotu szkolnego *De Havilland DH-60T "Tiger Moth"* opracowywały pierwszy bezpilotowy statek powietrzny *DH-82B "Queen Bee"*. Samolot ten, sterowany przez pilota za pomocą fal radiowych, miał służyć jako ruchomy cel do ćwiczeń dla obsługi działa przeciwlotniczych. Jego oficjalna prezentacja została jednak przerwana, ponieważ ówczesne systemy obrony powietrznej były tak mało skuteczne, że strzelającym skończyła się amunicja, zanim zestrzelili oni bezpilotowy samolot. Obiekt ten też spełnia wszystkie wymagania określone wcześniej przez autora, więc uznaje on go za pierwszego drona.

Równolegle w tym samym okresie, a konkretnie w 1935 r. powstał identyczny samolot dla amerykańskich odbiorców, *Radioplane OQ-2*. Powstał on jako pierwotnie jako bezzałogowiec, a nie przez modyfikacje tak jak dron brytyjski, więc jego budowa bardziej odstawała od klasycznych samolotów.[10][1]

I.2.3. Pierwsze drony rozpoznawcze

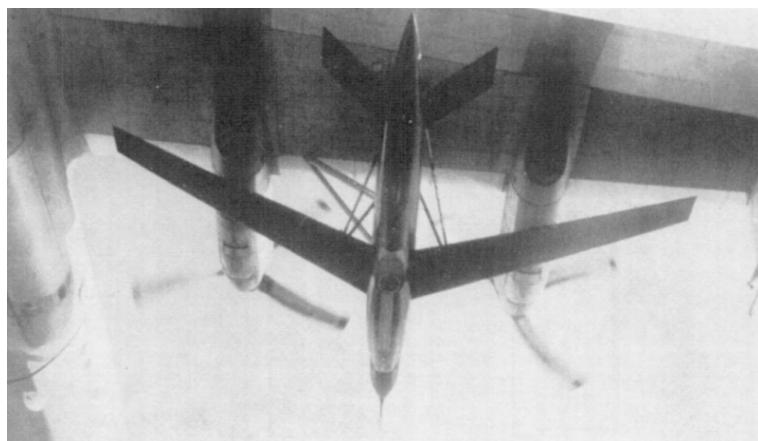
W bezpośrednim okresie po II wojnie światowej USA kontynuowało prace nad dronami, poprzez firmę *Ryan Aeronautical Company* i ich serii dronów *Firebee*, produkowanych



Rys. 2. De Havilland Queen Bee i premier Wielkiej Brytanii Winston Churchill

Źródło: <https://www.iwm.org.uk/collections/item/object/205195356>

od 1951 r. Efektem tej serii był m.in. opracowany w 1962 r. *Ryan Model 147 Lightning Bug*. Był to dron rozpoznawczy, napędzany był za pomocą silnika rakietowego. Nie posiadał on wyposażenie do lądowania i startowania z ziemi, tak więc odbywała się to za pomocą spadochronu, w który był wyposażony, i jego przechwyceniu w locie przez helikopter. Sam start odbywał się z pokładu samolotu. Tak samo, jak pociski rakietowe, dron ten był umieszczany pod skrzydłem samolotu.



Rys. 3. Ryan Model 147 Lightning Bug umieszczony pod skrzydłem samolotu transportowego

Źródło: https://en.wikipedia.org/wiki/File:Model_147_RPV_pictured_in_flight_under_wing_pylon_of_a_carrier_aircraft.png

I.2.4. Ikona wśród BSP

Zdecydowanie do najpopularniejszego BSP na świecie należy zaliczyć, produkowanego przez amerykańskiego producenta, *General Atomics MQ-1 Predator*. Pierwsze jego wersje nie posiadały na swoim pokładzie żadnych pocisków, ponieważ rząd USA nie byli pewni czy jest to zgodne z obowiązującym układem dotyczącym całkowitej likwidacji pocisków rakietowych średniego zasięgu (ang. *Intermediate-range Nuclear Forces (INF) Traty*). Jednak wydarzenia z 11 września 2001 r. były impulsem do podjęcia decyzja o uzbrojeniu Predatorów w pociski rakietowe i skierowania ich do akcji. Umożliwiło to prowadzenie operacji militarnych bez ponoszenia strat w żołnierzach. Drony te były wykorzystywane w trakcie konfliktów w Afganistanie, Iraku czy Pakistanie. Na przestrzeni lat 2009-2021 Zjednoczona Ameryka stała się światowym liderem w używaniu dronów bojowych.[1][21]



Rys. 4. *MQ-1 Predator*, wyposażony w rakiety AGM-114 Hellfire

Źródło: https://en.wikipedia.org/wiki/File:MQ-1_Predator,_armed_with_AGM-114_Hellfire_missiles.jpg

I.2.5. Drony cywilne

Trudno zaprzeczyć stwierdzeniu, że wojna przyczynia się do szybszego rozwoju, bo to właśnie rozwiązania opracowane dla armii przenoszone są często do życia codziennego. Było tak z herbatą ekspresową, jak jest i teraz z bezzałogowymi statkami powietrznym. Zmieniła się tylko ich rola, za ich pomocą nie prowadzi się działań wojennych, a nagrywa sceny do filmów, prowadzi transmisje skoków narciarskich z ciekawszej perspektywy i ratuje ludzi zagubionych w górach. Okolice obecnego roku można wskazywać jako okres największego zainteresowania tą technologią na rynku cywilnym. Początku tego okresu można próbować określić na 2013 r., czyli datę premiery pierwszej wersji, prawdopodobnie najpopularniejszej serii dronów *Phantom* od obecnie najpopularniejszego producenta dronów konsumenckich *DJI*.

I.2.6. Konflikt na Ukrainie

W kontekście bezzałogowych statków powietrznych nie można pominąć aktualnego konfliktu zbrojnego na Ukrainie. Należy go rozpatrywać w dwóch aspektach: przewagi, która armia ukraińska uzyskuje dzięki tureckim dronom *Bayraktar TB2* i wykorzystaniu dronów konsumenckich od ludności cywilnej do przeprowadzenia rozpoznania powietrznego.

Rząd ukraiński zwrócił się z prośbą do swoich obywateli o przekazanie swoich dronów na potrzeby armii. Są one wykorzystywane do bezpiecznego prowadzenie rozpoznania przez wojska ukraińskie. Dostarczają one obraz na żywo, wraz ze swoimi współrzędnymi geograficznymi. Pozwala to budować przewagę informacyjną na polu bitwy, a ten konflikt szczególnie uświadomił, jak ważna jest dzisiaj informacja na polu bitwy.[12]

Czytając artykuły poświęcone dronom *Bayraktar TB2* w kontekście konfliktu, można odnieść wrażenie, jakby było to jedyny element budujący ich siłę. Autor nie może się z tym zgodzić, ale nie da się zaprzeczyć, że ich rola jest znacząca. Szczególnie po obejrzeniu licznych nagrań dostępnych w internecie z działań tych samolotów na wojnie. Głównym celem tej maszyny jest jednak prowadzenie rozpoznania, ale mogą być one doposażone w cztery pociski kierowane o zasięgu 8 km. Sam dron jest jedną z tańszych opcji na rynku, bo jego cena waha się między 2-6 mln dolarów, a drony z najwyższej półki sięgają 100 mln dolarów. Rozpiętość skrzydeł tego drona to 12 metrów, a długość to 6,5 metra, co przekłada się na możliwość szybowania przez 27 godzin lub przelecania 150 km. W dodatku wzbija się na pułap 8200 m i rozwija prędkość do 220 km/h, a wszystko to za sprawą silnika *Rotax 912 iS* o mocy 100 koni mechanicznych.[13][14]



Rys. 5. Bayraktar TB2

Źródło: https://www.instalki.pl/images/newsy/03-2022/Bayraktar_TB2_ukraina.jpg

I.3. Technologia i producenci BSP

Wymagania stawiane dronom na rynku cywilnym i militarnym znacznie się od siebie różnią, tak samo, jak konstrukcje które je spełniają. Największą różnicą jest właśnie rozmiar. Militarne BSP do realizacji swoich zadań muszą pokonywać dużo większe dystanse, przewożąc przy tym ciężkie wyposażenie. Te dwa środowiska można też uznać za hermetyczne względem siebie, rzadko następuje przejście z jedno w drugie. Większość producentów dronów wojskowych wcześniej produkowała inny sprzęt wojskowy, a cywilnych kamery czy elektryczne szybowce.

I.3.1. Shenzhen DJI Sciences and Technologies Ltd.

Shenzhen DJI Sciences and Technologies Ltd., znany powszechnie pod nazwą handlową DJI, jest obecnie największym producentem dronów konsumentckich. Z siedzibą w chińskiej "dolinie krzemowej" Shenzhen. Firma została założona w 2006 r., a swój pierwszy sukces odniosła w 2013 r. kiedy wpuściła na rynek pierwszy model serii dronów *Phantom*. Był to BSP przeznaczony dla początkujących operatorów, a na tle konkurencji wyróżniała go łatwość obsługi. W kolejnych latach firma kontynuowała swój rozwój, a w 2015 r. wraz z wypuszczeniem trzeciej wersji jego wersji stała się ona największym producentem na świecie. W 2016 r. firma ta posiadała 50% udziałów w światowym rynku, a rok później już 72%. W 2020 r. było około 74%, podczas gdy żadna inna firma w tym samym czasie nie posiadała więcej niż 5% udziału w światowym rynku.

W 2020 r. BSP firmy DJI były wykorzystane przez Chiny do przypominania ludziom o obowiązku noszenia maseczek na twarzy w celu ograniczenia rozprzestrzeniania się wirusa COVID-19. Z tego samego powodu w takich krajach jak Maroko czy Arabia Szaryjska drony mierzyły temperatury przemieszczającej się populacji w terenach silnie zurbanizowanych.[9][11]

I.3.2. Yuneec International

Yuneec International to drugi co do wielkości producent dronów konsumentckich, pomimo że w rynku międzynarodowym posiada zaledwie 5% udziałów. Firma ta została założona w 1999 r. i pierwotnie zajmowała się produkcją samolotów elektrycznych i szybowców. Obecnie jej siedziba znajduje się w Jiangsu w Chinach. Do swojej oferty wprowadziła pierwszego drona w 2015 r. Był on przeznaczony do fotografii.

W 2014 r. firma została członkiem założycielem *Dronecode*, organizacji non-profit prowadzonej przez *Linux Foundation*, której celem jest dostarczanie otwartego oprogramowania do dronów opartego na jądrze systemu Linux. Firma *Intel Corp* w sierpniu 2015 r. zainwestowała 60 mln dolarów w Yuneec w zamian za 15% udziałów. Celem inwestycji była wspólna realizacja przyszłych projektów. W tym samym miesiącu Yuneec wprowadził na rynek drona *Breeze*, zdolnego do rejestrowania filmów i zdjęć w rozdzielcości UltraHD 4K.

Również w 2015 r. firma nawiązała współpracę z *Ocean Alliance*, organizacją zajmującą się ochroną wielorybów. Celem było stworzenie bezpieczniejszego sposobu na zbieranie danych o stanie zdrowia wielorybów. Organizacja *Ocean Alliance*, zamiast

korzystać, tak jak dotychczas z rzutek biopsjnych zaczęła używać do tego celów specjalnie wyposażonych dronów Yuneec.[7]

I.3.3. Baykar

W tym zestawieniu nie mogło zabraknąć producenta najpopularniejszego drona militarnego w kontekście aktualnego konfliktu zbrojnego na Ukrainie, czyli: *Baykar*. Jest to prywatna firma Turecka specjalizująca się w obszarach sztucznej inteligencji, BSP i systemach *Command And Control* (pl. dowodzenie i kontrola). Została ona założona w 1984 r. przez Özdemir Bayraktar i pierwotnie dostarczała części samochodowe takie jak pompy, silniki i inne. W latach dwutysięcznych firma zainteresowała się obszarem BSP, czego efektem było wyprodukowanie *Bayraktar Mini UAV*. Był to pierwszy dron w całości wyprodukowany z kapitału krajowego Turcji i w 2007 r. znalazł się na wyposażeniu Tureckich Sił Zbrojnych. Rozpoczęcie prac badawczo-rozwojowych przyczyniło się do produkcji pionierskich i zaawansowanych systemów. Portfolio firmy obejmuje również latający samochód, nazwany *Cezeri*, który w trakcie testów w Stambule we wrześniu 2020 r. wzniósł się na wysokość 10 m.[22]



Rys. 6. Dron osobowy Cezarei w trakcie testów w Stambule

Źródło: <https://www.savunmahaber.com/en/wp-content/uploads/2020/09/CEZERI-UCAN-ARABA-26.jpg>

I.3.4. General Atomics

General Dynamics Corporation to amerykańska korporacja założona w 1952 r., z siedzibą w Reston, w stanie Wirginia. Do 1990 r. dostarczała on czołgi, rakiety, pociski, łodzie podwodne, okręty wojenne, myśliwce i elektronikę dla wszystkich rodzajów

wojsk. Na początku lat 90tych sprzedała całe swoje portfolio, z wyjątkiem działalności związanej z pojazdami wojskowymi i okrętami podwodnymi.

Korporacja ta jest właścicielem *General Atomics Aeronautical Systems*, która zajmuje się produkcją systemów radiowych i bezzałogowych statków powietrznych, w tym rewolucyjnego kiedyś drona wojskowego *Predator*. Seria tych dronów jest nadal rozwijana, a poszczególne konstrukcje są do siebie bardzo zbliżone.[28][27]

I.4. Zastosowania BSP

Bezpilotowe statki latające znalazły szereg zastosowań, pierwotnie były wykorzystywane głównie w obszarze militarnym, dopiero później dostrzeżono w nich potencjał również w środowisku cywilnym. Ponieważ kontekst militarny został już dość szczegółowo przedstawiony, w poniższym tekście przyłożono większą uwagę do ich cywilnych zastosowań.

I.4.1. Militarne zastosowania BSP

Bezzałogowe statki powietrzne w kontekście militarnym można dokonać podziału na następujące kategorie:

- **bojowe** - przenoszące i używające środki bojowe/ środki rażenia, np. *Bayraktar TB2*;
- **amunicja krążąca** - umożliwiające wykrywanie, rozpoznanie oraz atak na wyznaczony cel poprzez autodestrukcję, np. *WB Electronics Warmate*;
- **operacyjno-rozpoznawcze** - realizujące rozpoznawanie oraz śledzenie obiektu/celu, a także monitorowanie i kontrolę obszaru zainteresowania, np. granic lub strefy przybrzeżnej. Przykładem takiego drona jest *Lockheed Martin RQ-170 Sentinel*;
- **wsparcia**: umożliwiające ewakuacje lub dostawę amunicji, wyposażenia, środków medycznych i żywności do wysuniętych stanowisk wojsk własnych, np. *Kaman KARGO UAV* [26]

I.4.2. Cywilne zastosowania BSP

Wymienienie wszystkich cywilnych rozwiązań jest trudne, ponieważ rynek ten znajduje co chwilę kolejne zastosowania dronów. Poniżej przedstawiono parę wyselekcjonowanych interesujących rozwiązań.

Transport medyczny

W czerwcu 2022 r. Polska Agencja Żeglugi Powietrznej wydała zgodę na wykonywanie regularnych długodystansowych lotów BSP. Realizować będzie je firma transportowa na rzecz systemu opieki zdrowotnej. Połączenie obejmie dwie trasy Warszawa-Pułtusk i Warszawa-Sochaczew. Przewidywana częstotliwość lotów to ok. 7 w tą i z powrotem w ciągu dnia. Obie te trasy mają długość 60 km i odbywają się na wysokości 100 m.

Sam samolot posiada systemy wizyjne, które będą korygować lot w przypadku wystąpienia przeszkód na jego trasie przelotu. Takie połączenie zapewni szybki transport np. narzędzi do przeszczepu co może przyczynić się do uratowania komuś życia. [29]



Rys. 7. Dron Farada G1, za pomocą którego będzie obywał się transport medyczny w okolicach Warszawy

Źródło: <https://www.pansa.pl/wp-content/uploads/2022/02/IMG-20220213-WA0001-1024x577.jpg>

Ratownictwo

Drony powietrzne pomagają również w ratowaniu ludzkiego życia, szczególnie w górzystych terenach. W styczniu 2022 r. ze względu na trudne warunki atmosferyczne ratownicy TOPR nie mogli udzielić pomocy dwóm turystom, którzy nie byli w stanie zejść z góry. Za pomocą drona dostarczono im koce i ogrzewacze, co pozwoliło im przetrwać noc. Kolejnego dnia, gdy pogoda uległa poprawie ratownicy dotarli do poszkodowanych i ich sprowadzili w bezpieczny sposób.[30]

Również na dalekim zachodzie można znaleźć przykłady ratowania życia z użyciem BSP, a konkretnie w Północnej Kalifornii. Właśnie tam, w lesie, zgubił się młody myśliwy. Służby za pomocą drona zlokalizowali jego lokalizację, a następnie wysłali tam strażników, którzy z jego pomocą wyprowadzili zagubionego.[32]

Są cztery powody, dla których BSP dobrze odnajdują się w ratownictwie.

- **Czas reakcji** - są opcją bardzo szybkiego reagowania, czas potrzebny na przygotowanie do startu jest minimalny;
- **Szybkie przeszukiwania** - mogą przeszukać trudny teren w dużo szybszym czasie niż człowiek pieszo;
- **Komunikacja** - mogą komunikować się z poszkodowanymi za pomocą głośnika i mikrofonu;
- **Lokalizacja** - są w stanie przekazywać na żywo do operatora swoją aktualną lokalizację, co w przypadku ratowania i poszukiwania osób może znacznie minimizować czas poszukiwania i ewakuacji. [31]

Kinematografia i produkcja filmowa

Jednym z pierwszych filmów, które przywoływanie jako przykład dobrego wykorzystania dronów jest *Skyfall* z 2012 r., a konkretnie scena pościgu motorem przez Jamesa Bonda złoczyńców po dachach w Istambule. BSP dały kinematografii wyjątkową przewagę nad tradycyjnymi metodami filmowania. Mają większy zasięg niż żuraw i są też bardziej zwinne niż helikopter. Reżyserzy dzięki temu mogą wykonywać bardziej ryzykowne, prawdziwie akrobatyczne ujęcia, które gdyby nie drony musiałyby być wytworzone na komputerze. Nie można zapomnieć, że BSP są przede wszystkim tańsze w zakupie i utrzymaniu niż np. helikoptery.[34]

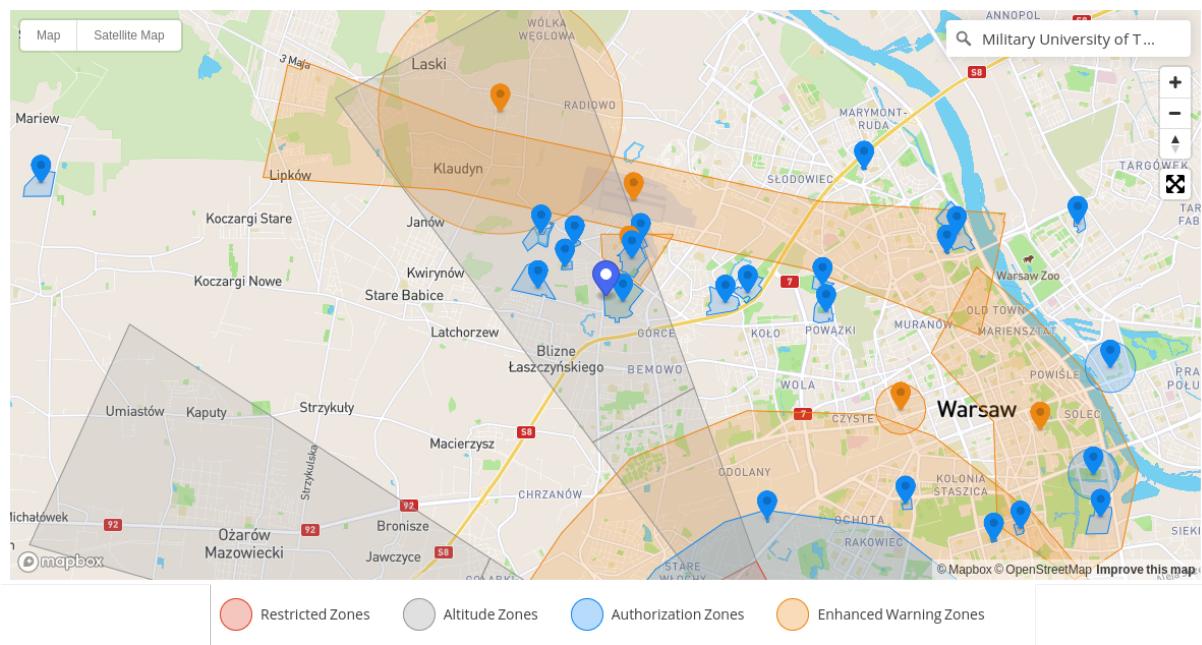
Przykładem gdzie wcześniej nie było możliwe nagrywanie ujęć filmowych, są erupcje wulkanów. Takie loty wiążąły się z dużym bezpieczeństwem, a drony są dużo tańsze i nie posiadają na swoim pokładzie załogi, tak więc operatorzy mogą pokusić się o takie ryzyko. Takie nagranie wykonał znany youtuber Joey Helms, który postanowił uwiecznić z bliska rzekę lawy wyrzucaną przez wulkan Fagradalsfjall na Islandii. W trakcie nagrywania erupcja lewy w pewnym momencie sięgnęła jego statku. Youtuber bezpowrotnie stracił swój statek. Ujęcie, które zostało zachowane, jest bardzo emocjonujące, więc prawdopodobnie było to warter swojej ceny.[33]

Transport towarów

W 2013 r. Jeff Bezos ogłosił koncepcje, która miała zrewolucjonizować transport towarów. Autonomiczne drony powietrzne miały dostarczać paczki Amazonu w 30 minut pod drzwi odbiorcy. Mogły one zaoferować konsumentom dostawę żywności, leków czy innych lżejszych przedmiotów, bez spalania paliw kopalnianych i czekania. Niestety jednemu z najbogatszych ludzi na świecie nie udało się tego osiągnąć. Tak samo, jak firmie *Zipline*, która miała dostarczać leki w Rwandzie i projektowi Google *Wing*, który miał zapewniać burito głodnym studentom. Na świecie istnieje duża presja na wprowadzenie takiego rozwiązania. W 2021 r. Amazon zwolnił większość pracowników odpowiedzialnych za rozwój wspominanego projektu, uzasadnił to błędem w zarządzaniu i panującym chaosem. W tym samym roku DHL również ogłosił zakończenie swojego identycznego projektu. Było to 8 lat po tym, gdy ich dron pierwszy raz wbił się w powietrze. Koszty transportu towaru między centrum dystrybucji a klientem końcowym to 40% całkowitego kosztu, a konsumenci oczekują szybkiej taniej i ekologicznej dostawy. Wszystkie te potrzeby mogłyby zaspokoić dostawy dronami.

Jest parę czynników, które mogły wpływać na wstrzymanie masowych dostaw towarów dronami. Głównym z nich jest prawdopodobnie ograniczenie stref lotów BSP. Osoby zarządzające państwami i miastami na bieżąco wprowadzają regulacje w tej dynamicznie rozwijającej się domenie. Wprowadzone zostały m.in. strefy wyłączone dla dronów, które mają zapewnić bezpieczeństwo wokół miejsc wymagających specjalnej ochrony. Są to najczęściej okolice lotnisk i jednostek wojskowych. Takie strefy mocno ograniczają dostawy towarów dronami.[36]

Mimo wszystko taki transport jest jednak nadal możliwy, ale nie na masową skalę. W Sosnowcu od 2021 r. realizowany jest projekt, w którym drony będą świadczyć usługi transportu m.in. pilnych przesyłek medycznych. Odbiór i nadawanie będzie odbywało się w stacjach dokujących, które będą automatycznie ładowały i zdejmowały



Rys. 8. Strefy wyłączone dla dronów w okolicach Warszawy

Źródło: <https://www.dji.com/pl/flysafe/geo-map>

ładunek z drona. Wykorzystanie takich punktów umożliwia ustawienie stałej trasy lotów, co zdejmuje dużą część odpowiedzialności z uczestników projektu, dlatego też w przyszłości można spodziewać się większej ilości tego typu rozwiązań.[37]

I.5. Dostosowywanie BSP do wymagań użytkownika

Na rynku znajduje się duża ilość dostępnych komponentów, z których można łatwo zbudować własne bezzałogowe statki powietrzne. Najwięcej odpowiedzialności z konstruktora zdejmują gotowe kontrolery lotu, np. *pixhawk*, na którym można zainstalować otwarte oprogramowanie *ArduPilot*. Pozwala to zachować dużą ilość zasobów, bo cała logika lotu jest praktycznie *out-of-the-box*. Przykładowy zestaw elementów potrzebnych do zbudowania własnego drona to::

- rama,
- silnik,
- elektroniczny kontroler prędkości,
- śmigła,
- załączka,
- rozdzielnia zasilająca,
- baterie,
- monitoring baterii,
- mata montażowa,

- kontroler,
- odbiornik RC,
- kamera,
- kartka pamięci SD.[35]



Rys. 9. Kontroler lotu Pixhawk

Źródło: https://ardupilot.org/plane/_images/Pixhawk_with_legend.jpg

Budowa własnego drona jest jak najbardziej możliwa, bez wymagania specjalistycznej wiedzy, ale nadal to potrzeba na to dużo ilości zasobów, szczególnie czasu, dlatego warto przyjrzeć się gotowym rozwiązaniom dostarczonym przez producentów i to jak bardzo można je dostosowywać. Firma DJI udostępnia do swoich produktów SDK, czyli bibliotekę, za pomocą której można zaprogramować działanie naszego drona. Dodatkowo producent dla wybranych statków przygotował szereg rozszerzeń, które pomogą dostosować wyposażenie do naszych wymagań. Przykładowo dla *Mavic 2 Enterprise Advanced* dostępne są:

- **głośnik** - który umożliwia komunikację z dronem, np. w czasie sytuacji alarmowych do ludności znajdującej się na ziemi
- **dodatkowe oświetlenie** - w przypadku wystąpienie niekorzystnych warunków atmosferycznych lub lotów w nocy
- **moduł RTK** - który umożliwia osiągnięcie dokładności lokalizacji na poziomie jednego centymetra

Dodatkowo dla innych modeli dostępne są wyspecjalizowane kamery np. do skanu obiektów w 3D czy podglądu w podczerwieni.

Rozdział II. Przegląd i prezentacja technologii mobilnych z uwzględnieniem aspektów tworzenia aplikacji i komunikacji M2M

W tym zdefiniowano cel i wymagania stawiane technologią M2M. Następnie przedstawiono przykładowe technologie ze szczególnym uwzględnieniem komunikacji pomiędzy bezzałogowym statkiem powietrznym a kontrolerem. Rozdział zakończono analizą interfejsu API umożliwiającego kontrolowanie drona.

II.1. Komunikacja M2M

Komunikacja M2M (machine-to-machine) to kategoria technologii, która umożliwia wymianę informacje pomiędzy urządzeniami w sieci bez jakikolwiek ingerencji ludzi. Obiekty pracujące w tej sieci charakteryzują się mniejszą lub większą autonomicią. Wspierana jest ona często szeroko pojętą sztuczną inteligencją, ze szczególnym uwzględnieniem technik uczenia maszynowego. Ta komunikacja jest podstawą istnienia IoT.[4]

II.1.1. Cel

Głównym celem M2M jest autonomiczna komunikacji pomiędzy maszynami, a jej obecnie najczęstszym zastosowaniem jest przenoszenie danych z sensorów do sieci. Obecnie operatorzy coraz częściej rozbudowują swoją infrastrukturę o węzły zgodnie z tą kategorią technologii. Dzięki temu koszty utrzymania całego systemu są znacznie zredukowane. Składają się na niego następujące elementy:

- łącze do przesyłu danych, np. WiFi, GSM
- sensory, np. czujnik temperatury, kamera
- oprogramowanie, które automatyzuje procesy komunikacyjne, np. przeszukiwanie ścieżki routingu

Celem telemetrii jest automatyczny pomiar wielkości fizycznej przez odpowiednie sensory. Wartość pomiaru jest przesyłana do miejsca, zwykle odległego, w którym jest dalej przetwarzana. Na początku do tego celu były wykorzystywane linie telefoniczne, a następnie radiowe. Rozwój technologii, a w tym łączności bezprzewodowej sprawił, że poszerzyła się rolą wykorzystywania telemetrii w nauce, inżynierii i produkcji. Dzisiaj jest ona używana także w życiu codziennym, w jednostkach grzewczych, miernikach elektrycznych i wszelkich urządzeń podłączonych do internetu. Jej rozwój jest ściśle powiązany z komunikacją M2M.[4]

II.1.2. Wymagania

Według Europejskiego Instytutu Norm Telekomunikacyjnych (ETSI) komunikacja M2M musi spełniać następujące wymagania:

- **Skalowalność:** w miarę dołączania kolejnych urządzeń do systemu system nadal musi funkcjonować;
- **Anonimowość:** w związku z wymaganiami prawnymi, na każde żądanie system musi umożliwiać ukrywanie tożsamości urządzenia;
- **Logowanie:** ważne wydarzenia w systemie, takie jak: pojawienie się błędnych informacji czy nieudane próby instalacji, muszą być zarejestrowane, a rejesty te muszą być dostępne na żądanie;
- **Zasady komunikacji między aplikacjami:** aplikacje w systemie powinny mieć możliwość komunikowania się. W szczególności bramki i urządzenia końcowe komunikujące się za pomocą technologii SMS czy Ethernet powinny komunikować się za pomocą połączenia P2P (peer-to-peer);
- **Metody dystrybucji:** w ramach systemu powinny być dostarczane metody dystrybucji takie jak: *unicast*, *multicast*, *broadcast* i *annycast*, a wszędzie gdzie to możliwe metoda *broadcast* powinna być zastąpiona za pomocą *multicast*, tak aby zminimalizować obciążenie sieci;
- **Harmonogram przesyłania komunikatów:** dostęp do sieci powinien być kontrolowany, tak samo, jak harmonogram przesyłania komunikatów. Sam system powinien również uwzględniać obciążenia aplikacji M2M w harmonogramie przesyłania wiadomości;
- **Wybór ścieżek komunikacyjnych:** ścieżki w systemie powinny zapewniać optymalizacje bazującą na: awariach transmisji, kosztu i opóźnieniach, w momencie, gdy istnieją inne ścieżki do punktu docelowego. [4]

II.2. Technologie komunikacji M2M

Poprawnym stwierdzeniem będzie, że obecnie znajdujemy się w epoce połączonych ze sobą obiektów. IoT (Internet of Things) zdobywa aktualnie coraz więcej uwagi nie mal w każdej domenie, a szczególnie w takich jak biznes, elektronika konsumencka, przemysł czy transport. Niemal każdy obiekt elektryczny w dzisiejszym świecie jest ze sobą połączony w ten czy inny sposób. Siedząc w biurze, za pomocą dostarczanych aplikacji, możemy kontrolować drzwi, bramę garażową, czajnik elektryczny czy rolety okienne w naszym domu. Z kolei w mieście kontrolujemy kamery i oświetlenie, a wszystko to z odległych lokacji. IoT odgrywa w tym ważną rolę, ponieważ to ono umożliwia łączenie przeróżnych obiektów, za pomocą sieci połączeń i wymianę danych między nimi.[6]

II.2.1. Ogólna klasyfikacja technologii M2M

Poniżej przedstawiono ogólne porównanie technologii komunikacyjnych M2M.

	Local Area Network Komunikacja krótko dystansowa	Low Power Wide Area Internet Of Things	Celluar Network Tradycyjne M2M
Użycie	40%	45%	15%
Zalety	- Dobrze ugruntowana norma - W budynkach	- Niskie zużycie energii - Niskie koszty - Pozycjonowanie	- Istniejące pokrycie znacznego obszaru - Duża prędkość transmisji
Wady	- Wysokie zużycie energii elektrycznej - Duży koszt sieci i zależności	- Niska prędkość transmisji - Wschodzący standard	- Wysoki koszt posiadania - Mała autonomia
Technologia	Bluetooth, WiFi	LoRa	GSM, 3G, 4G, 5G

Tab. 1. Porównanie rodzajów technologii M2M.[6]

II.2.2. LoRa

Komunikacja w aplikacjach IoT jest dzisiaj wykonywana w przeróżnych technologiach, a każda z nich ma swoje zalety, funkcje, a przez to też przeznaczenie. Żadna z tych technologii nie może pokryć całego zapotrzebowania świata IoT, ponieważ wszystkie one posiadają cechy, które czynią je odpowiednie dla postawionego konkretnego zadania.

LoRa (Long Range) to nowa technologia połączeń bezprzewodowych w świecie IoT. Ostatnio znacznie ewoluowała i zyskała szczególną popularność w urządzeniach z ograniczoną pojemnością elektryczną, umożliwiając systemom wbudowanym przesyłanie małej ilości danych na dużych dystansach w krótkich interwałach czasowych.

WiFi to najpopularniejsza technologia komunikacji bezprzewodowej, która jest już rozwijana przez wiele lat. W kontekście IoT wykorzystywana jest przede wszystkim do komunikacji na dużych odległościach. Na krótkie dystanse lepiej pasują do tego takie protokoły jak Bluetooth czy ZigBee. We wszystkich z nich największą wadą jest duże zużycie energii elektrycznej. Technologia Lora zapewnia bezpieczne, mobilne dwukierunkowe połączenie o niskim koszcie elektrycznym. Wykorzystywane jest ono w IoT, szczególnie w domenie smart city, czy nawet ogólnej komunikacji M2M. LoRa zalicza się do LPWA(Low Power Wide Area), czyli rodzaju bezprzewodowej rozległej sieci telekomunikacyjnej, stworzonej w celu umożliwienia komunikacji na duże odległości przy niskiej przepływności i niskim poborze energii. [5] W tego typu komunikacji wyróżnia się LoRa ze względu na jej:

- długodystansowość;
- dwukierunkowość;
- wysoką pojemność węzłów w sieci;

- długość życia na baterii;
- odporność interfejsów;
- bezpieczeństwo i efektywność sieci. [6]

Cechy

Technologie tą wyróżniają następujące cechy:

- Pojedyncza bramka może pokryć obszar aż $100km^2$;
- Oferuje ona podwójne szyfrowanie AES;
- Bazuje na technologii CSS (widmo rozproszone Chrip), które umożliwia śledzenie obiektów i jest odporne na zanikanie sygnałów;
- Topologia gwiazdy eliminuje zanikanie danych przez urządzenia pośrednie, co przyczynia się do zmniejszenia poboru mocy. [6]

Ograniczenie przepustowości

W sieci LoRa wszystkie klasy ramek wymagają potwierdzenia. Wiąże się z tym, że po każdym potwierdzeniu ramki przez urządzenie końcowe w dowolnym oknie czasowym następuje okres wyłączenia, w celu zachowanie zgodności z przepisami dotyczącymi cyklu pracy. W związku z tym, aby uniknąć wyczerpania limitu pojemności przez sieć i urządzenia końcowe, muszą one ograniczyć liczbę potwierdzeń. Również w podsieciach LoRa po przesłaniu danych następuje okres wyłączenia, w którym na danym kanale nie są wysyłane żadne dane. Te dwa okresy, tzn. okres wysyłania danych i wstrzymania transmisji stanowią cykl pracy. Cały ten mechanizm przyczynia się do ograniczenia przepustowości sieci.[6]

II.2.3. Narowband IoT

Wraz z rozwojem świata IoT zyskała również technologia Narowband IoT (NB-IoT). Wykorzystywana jest ona przede wszystkim przy komunikacji komórkowej dla zdalnych pomiarów w całej Europie. Jest to technologia dostępu radiowego. Używa ona ponownie komponentów stworzonych przez jej poprzednika LTE, aby umożliwić jej działanie na licencjonowanej częstotliwości. Może ona również działać w trybie autonomicznym. Tak jak sama nazwa wskazuje, cały system działa w wąskim spektrum częstotliwości, bo tylko w 200kHz, co wprowadza elastyczność zastosowań dzięki minimalnym wymaganiom częstotliwości, w porównaniu do jej poprzednika LTE. Cała szerokość 200kHz została podzielona na kanały po 3.75 kHz lub 15 kHz, co umożliwia połączenie w bardzo wysoką prędkość nadawania, a także daleki zasięg połącznia. [8]

II.2.4. NB-IoT vs Lora

Zarówno LoRa, jak i NB-IoT należą do wspomnianej wcześniej technologii LPWAN. Podstawowe różnice pomiędzy tymi dwoma technologiami można dostrzec w zużyciu baterii, prędkości transmisji i opóźnień.

Parametr	LoRa	NB-IoT
Pasmo	125 kHz	180 kHz
Pokrycie	165 dB	164 dB
Żywotność baterii	15+ lat	10+ lat
Maksymalne natężenie elektryczne	32 mA	120 mA
Spoczynkowe natężenie elektryczne	1 µA	5 µA
Przepustowość	50 Kbps	60 Kbps
Opóźnienie	Zależne od klasy urządzenia	10 s
Bezpieczeństwo	AES 128 bit	3GPP (128 to 256 bit)
Geolokalizacja	Tak (TDOA)	Tak (In 3GPP Rel 14)
Jakość/cena	Wysoka	Średnia

Tab. 2. Porównanie technologii LoRa i NB-IoT [3]

II.3. Komunikacja bezprzewodowa w dronach konsumenckich

Przeglądając katalog największego producenta dronów konsumenckich DJI, można wyróżnić tylko 3 technologie komunikacji bezprzewodowej: wzmacnione WiFi (ang. enhanced WiFi), Lightbridge, OcuSync.

II.3.1. WiFi

WiFi nie zostało wprowadzone ściśle do komunikacji bezprzewodowej statków powietrznych, ale odnajduje się w tym całkiem dobrze. Jest ona wykorzystywana głównie w bardziej budżetowych wersjach dronów, ze względu na możliwość skorzystanie przez producenta z posiadanej przez użytkownika infrastruktury (smartfonów), czy niskiej ceny komponentów.

Przykładowo dron *DJI Tello*, który jest najtańszą opcją dostępną od producenta DJI, przeznaczoną głównie do nauki latania i programowania, również przez najmłodszych pasjonatów. Nie posiada on w zestawie dedykowanego kontrolera, ponieważ odbywa się ono za pomocą aplikacji na smartfona, która łączy się z dronem za pomocą WiFi tak jak do punktu dostępowego z internetem. Zasięg takiego połączenia według producenta to 100 m.[16]

W swojej ofercie DJI ma również dostępnego drona *DJI Mini SE*, który również korzysta z technologii WiFi, ale w swoim wyposażaniu posiada dedykowany do niego kontroler. Taka konfiguracja pozwala na uzyskanie zasięgu do 2 km. [15]

WiFi jest także bardzo podatne na wszelkie zakłócenia, wynikające z ukształtowania terenu czy zaszumienia sieci pochodzącego z istniejących sieci domowych. Wyprodukowanie drona w tej technologii jest najtańszą dostępną opcją, która umożliwia transmisje obrazu, jednak należy pamiętać, aby nie stawiać jej przy tym za dużych wymagań. Stanowi ono dobry punkt startowy w komunikacji bezprzewodowej bezzałogowych statków powietrznych.



Rys. 10. DJI Tello

Źródło: <https://store.dji.com>

II.3.2. Lightbridge

TODO

Lightbridge to technologia od DJI, która doczekała się jej dwóch wydań. Pierwszych wzmianek o niej można doszukiwać się w 2014 r., a drugiego wydania już w 2015 roku. Jest ona zbliżona do technologii WiFi, przed wszystkim transmisja ta odbywa się na tej samej częstotliwości 2,4GHz.

Była ona kierowana głównie do dronów z wyższego pułpu cenowego, dlatego że jej produkcja była bardzo kosztowna, a koszt wynikał z tego, że producent opracował to rozwiązanie na swoim autorskim układzie scalonym i oprogramowaniu. Umożliwiło to osiągnąć duże lepsze wyniki niż transmisja po WiFi. Zasięg lotu według producenta to odległość do 5 km.

Obecnie Lightbridge nie jest już rozwijany, a producent skupił się na jego następcu, technologii OcuSync. [17][18]

II.3.3. OcuSync

OcuSync został po raz pierwszy zademonstrowany przez producenta wraz z wydaniem drona *Mavic Mini Pro*. Pierwsze wydanie tej technologii pozwalało na transmisje do 7 km na częstotliwości 2,4 GHz. Obraz mógł być przesyłany w rozdzielczości 720p i 1080p. Jakość fullHD była dostępna tylko na krótszych odległościach. Na większych dystansach gdy dostępna prędkość transmisji spadała, dron przechodził automatycznie na transmisje w 720 p. Opóźnienie było rzędu 160-170ms. A największą cechą wyróżniającą tę technologię była możliwość podłączenia jednocześnie dwóch kontrolerów i do 4 urządzeń odbiorczych.

Kolejnym krokiem było wydane wersji oznaczonej jako OcuSync 1.5, w której dodano transmisję również na częstotliwości 5 Ghz. Zmniejszono także opóźnienia w transmisji. Dodatkowo technologia umożliwiała automatyczną zmianę kanałów komunikacyjnych w trakcie lotu na te najmniej obciążone. W pierwszej wersji kanał transmisji można było wybrać tylko przed startem bezzałogowego statku powietrznego.[24]

Wraz z wydaniem nowej wersji zaprezentowano gogle *DJI* przeznaczone do transmisji obrazu w trybie FPV (ang. first person view, widok pierwszo-osobowy) i również



Rys. 11. Pierwsza wersja gogli do FPV od DJI

Źródło: https://u.cyfrowe.pl/600x0/2/7/2_732250420.png



Rys. 12. Dji OcuSync Air Unit

Źródło: <https://store.dji.com>

OcuSync Aircraft System, czyli zintegrowanego systemu umożliwiającego sterowanie i transmisji obrazu z wykorzystaniem tej technologii w dronach i pojazdach DIY.

Producent w trakcie swojej historii doprowadził do pewnych nieścisłości. Pomimo że dron *Phantom 4 pro v 2.0* korzystał teoretycznie z najnowszej wersji OcuSync, nie posiadał on możliwości zmiany kanałów transmisji w trakcie lotu, a opóźnienie zależało też od tego, czy korzystano z kontrolera dołączonego do zestawu, czy jego droższej, lepiej wyposażonej wersji: *DJI RC Plus*.

Wersja 2.0 wprowadziła dalsze ulepszenia, m.in. kontorlowanie dronów na jeszcze większe dystanse i z jeszcze mniejszymi opóźnieniami, a także kompatybilność wsteczną po aktualizacji oprogramowania.



Rys. 13. DJI Phantom v2.0

Źródło: <https://store.dji.com>

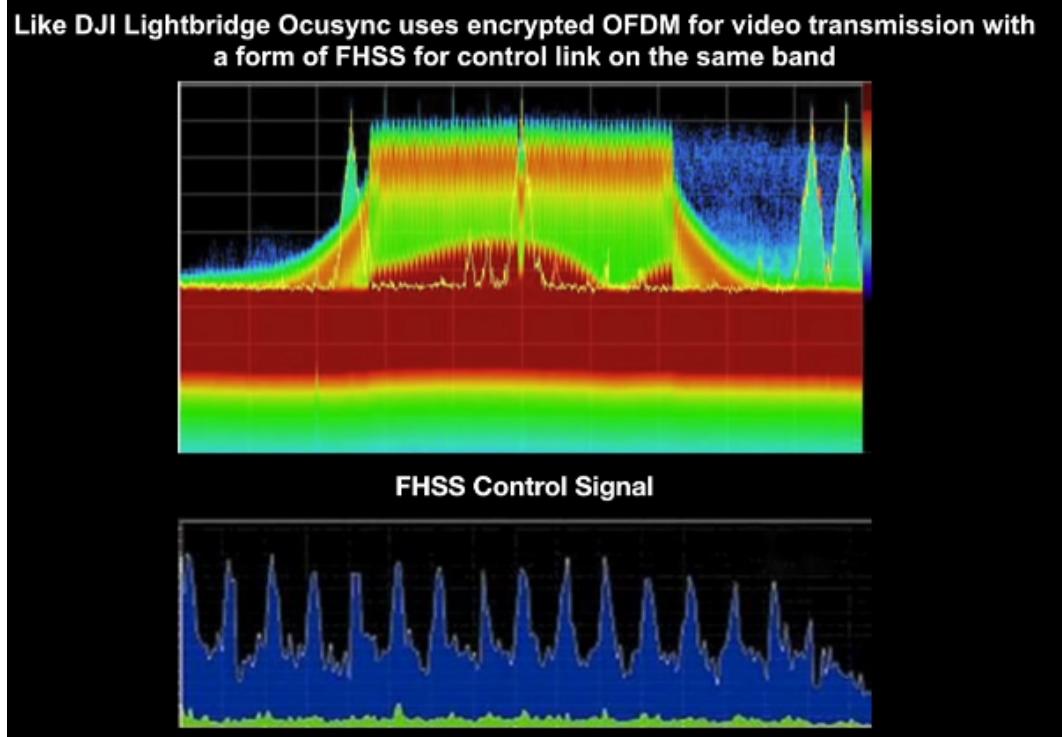


Rys. 14. DJI RC plus

Źródło: <https://store.dji.com>

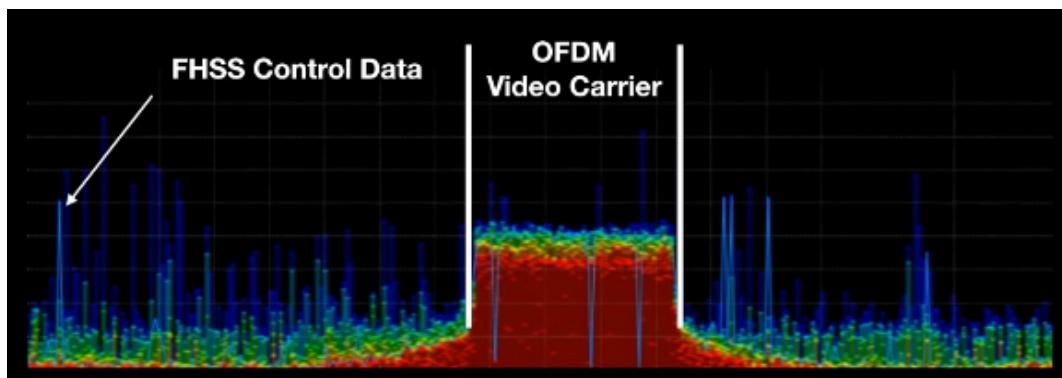
II.3.4. FHSS i OFDM

Zarówno Lightbridge, jak i OcuSync używają szyfrowanej modulacji OFDM (ang. Orthogonal Frequency-Division Multiplexing, zwielokrotnianie z ortogonalnym podziałem częstotliwości) dla transmisji obrazu i z formy FHSS (ang. Frequency-Hopping Spread Spectrum) dla transmisji sygnałów sterowania. Kanał dla transmisji obrazu nie zmienia się w trakcie całego lotu, pod warunkiem, że nie następują zakłócenia, albo użytkownik nie ustawi ręcznie innej częstotliwości. Z kolei metoda FHSS śkacze "po częstotliwościach w całym dostępnym widmie, w tym nawet w pasmie przeznaczonym do transmisji obrazu.[23] [19]



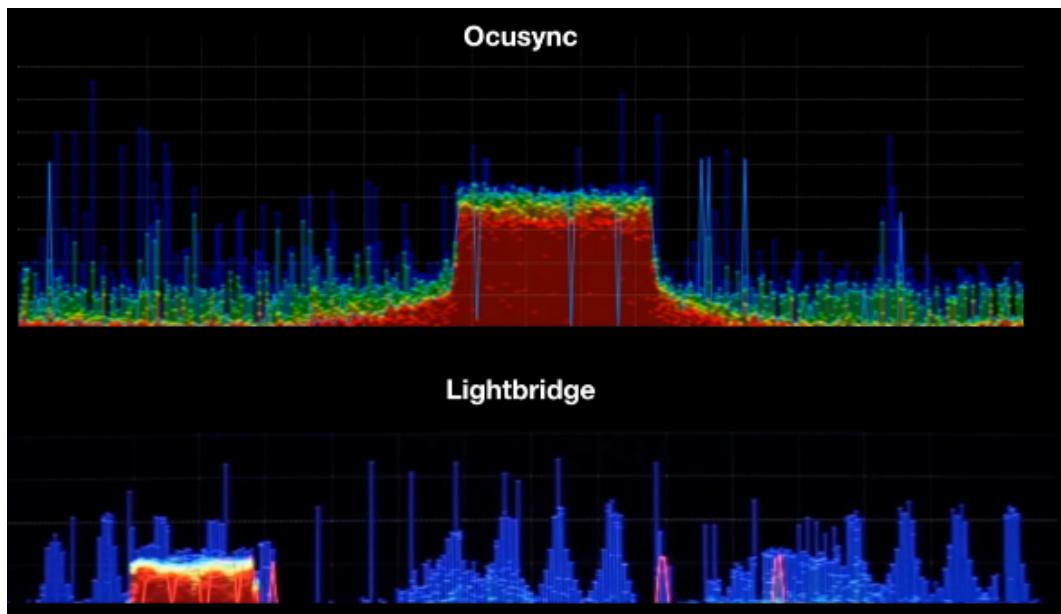
Rys. 15. Widmo OFDM i FHSS

Źródło: <https://www.youtube.com/watch?v=gfqcSv9sR0A>



Rys. 16. Widmo OcuSync z zaznaczoną modulacją FHSS i OFDM

Źródło: <https://www.youtube.com/watch?v=gfqcSv9sR0A>



Rys. 17. Porównanie widma OcuSync i Lightbridge

Źródło: <https://www.youtube.com/watch?v=gfqcSv9sR0AV>

II.3.5. Przewaga OcuSync nad Lightbridge

OcuSync stało się główną technologią rozwijaną przez DJI. Producent wykorzystuje układy scalone przeznaczone do komunikacji WiFi. Wytwarza na nie swoje oprogramowanie, która można bez problemu aktualizować. Lightbridge nie miał takiej możliwości, a w dodatku komunikacji opierała się wyłącznie na paśmie 2,4GHz. Nowe procesory w układach WiFi dzięki coraz większej częstotliwości pracy zapewniły osiąganie tych samych efektów, co DJI uzyskiwał za pomocą autorskich układów scalonych, bez dodatkowego kosztu wynikającego z produkcji.

II.4. Kontrolowanie BSP za pomocą API dostarczającego od producenta

Przeszukując internet w poszukiwaniu bezzałogowych statków powietrznych umożliwiających ich sterowanie za pomocą API od producenta, można natknąć głównie rozwiązania od DJI. Wszystkie pozostałe rozwiązania nie działają na gotowych dronach, a na oprogramowaniu przeznaczonym do wgrania na wybranych jednostkach do sterowania modelami RC.

Najpopularniejszym tego rozwiązaniem jest ArduPilot, czyli pakiet oprogramowania nawigacyjnego działającego w pojeździe wraz z oprogramowaniem sterującym stacją naziemną.

II.4.1. DJI SDK

DJI dostarcza do swoich produktów następujące interfejsy API:

- **App Dev.** - interfejsy API przeznaczone do sterowania dronem z poziomu stacji bazowej, kontroler stanowi interfejs pośredniczący między aplikacją wykorzystującą SDK a dronem powietrznym:

1. **Mobile SDK** - SDK przeznaczona na platformę iOS i Android. Aplikacja na smartfon za pomocą kabla USB podłączonego do kontrolera statku powietrznego realizuje zaprogramowaną logikę działania.
 2. **UX SDK** - to Mobile SDK rozszerzony o elementy interfejsu użytkownika, co przyspiesza znacznie proces tworzenia oprogramowania.
 3. **Windows SDK** - SDK umożliwiające wydawanie aplikacji na systemach operacyjnych Windows.
- **Payload Dev.** - interfejsy API przeznaczone do nadawania logiki działania drona na poziomie samego drona, dzięki temu po utracie zasięgu może ona dalej funkcjonować. Opcja dostępna dla najdroższych wersji dronów DJI, które można dostosowywać do swoich wymagań za pomocą odpowiednich rozszerzeń, np. kamery termowizyjnej
 1. **Payload SDK** - zestaw narzędzi programistycznych umożliwiających tworzenie oprogramowania do rozszerzeń, które mogą być montowane na dronach DJI.
 2. **Onboard SDK** - otwarte-źródłowe API umożliwiające bezpośrednią komunikację z wybranymi dronami i kontrolerami za pomocą interfejsu szeregowego.

Rozdział III. Projekt mobilnego systemu zarządzania i sterowania BSP

III.1. Wymagania funkcjonalne

TODO

III.2. Wymagania pozafunkcjonalne

TODO

III.3. Stos technologiczny

TODO

III.3.1. DJI Mobile SDK

TODO

III.3.2. Android

TODO

III.3.3. Kotlin

TODO

III.3.4. Java

TODO

III.3.5. Gradle

TODO

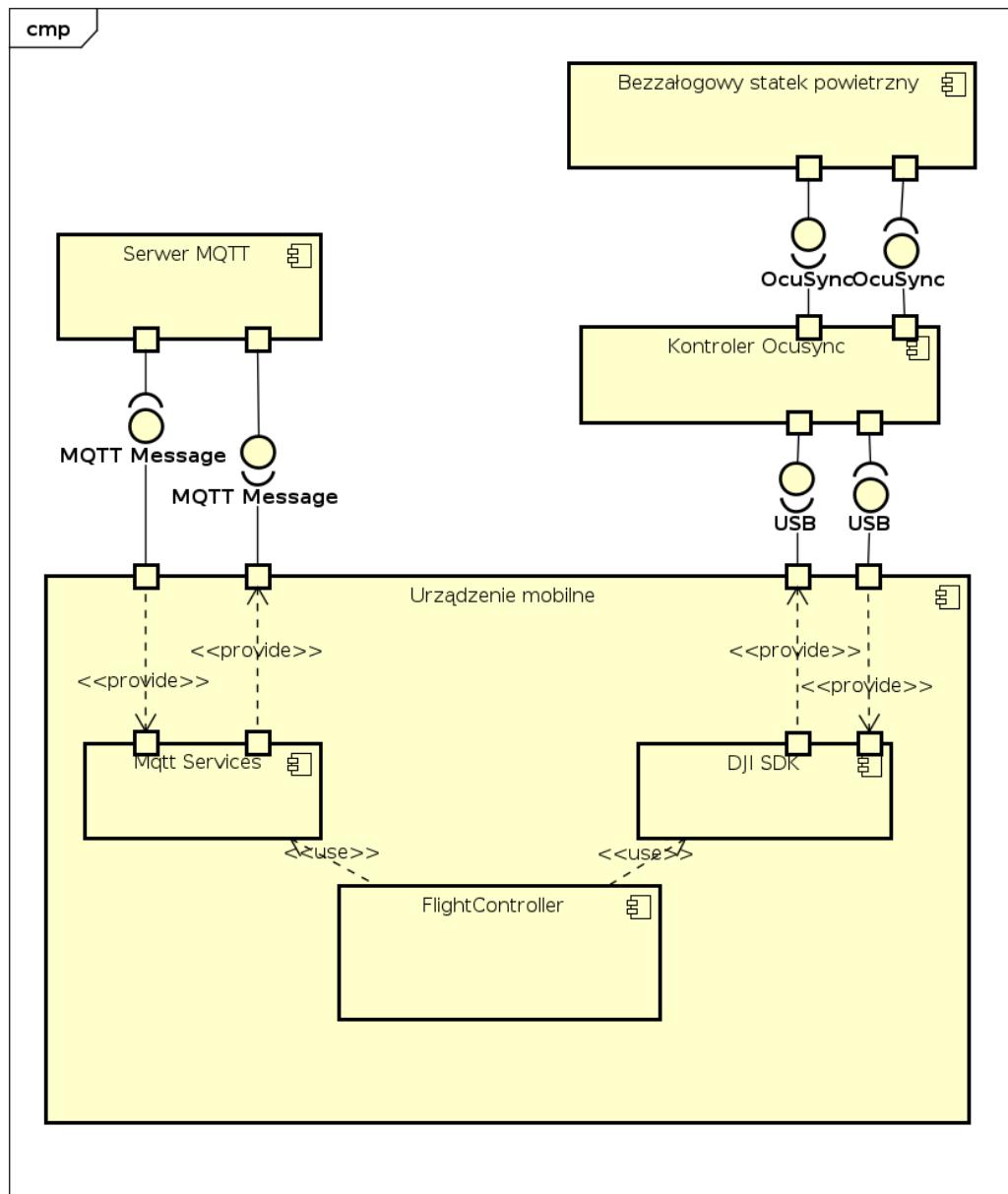
III.3.6. MQTT

TODO

III.4. Wysokopoziomowy diagram systemu

TODO tutaj diagram całości jako rój

III.5. Diagram komponentów



Rys. 18. Diagram komponentów

Źródło: Własne

Komunikacja dronów w ramach systemu będzie odbywała się za pomocą serwera MQTT. Będzie on odbierał dane ze statku i nim sterował za pomocą określonych komend. W ramach urządzenia mobilnego zostaną wyróżnione trzy komponenty:

- **Mqtt Services** to serwisy, które będą realizowały komunikacje z brokerem MQTT. Ich projekt i implementacja dozyć będzie do tego, aby API wystawione do komponentu *FlightController* wymagało jak najmniej działania.

- **DJI SDK** to dwie biblioteki dostarczane od producenta: *DJI SDK* i *DJI UXSDK*. Pierwsza z nich udostępnia metody umożliwiające sterowanie dronem i odczytywanie jego parametrów, a druga dostarcza komponenty w Androidzie umożliwiające obsługę drona za pomocą elementów graficznych.
- **FlightController** będzie korzystał z dwóch wcześniej wymienionych komponentów, w nim będzie zawarta logika działania kontrolera. Praktycznie będzie stanowił jako punkt centralny w systemie z punktu widzenia pojedynczego drona.

Kontroler Ocusync, a dalej dron, będą sterowane za pomocą biblioteki dostarczonej przez producenta.

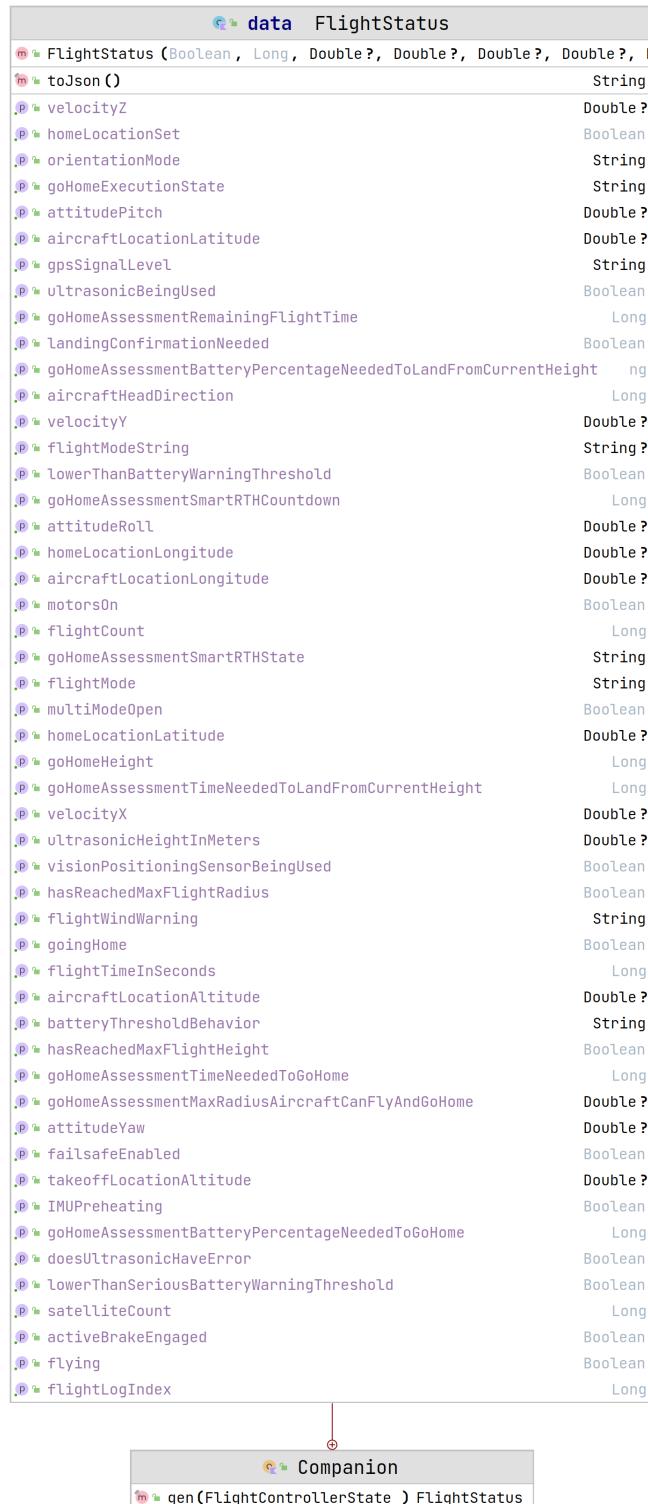
III.6. Diagram klas i sekwencji



Rys. 19. Diagram klas ograniczony do dostępnych komend w ramach systemu

Źródło: Własne

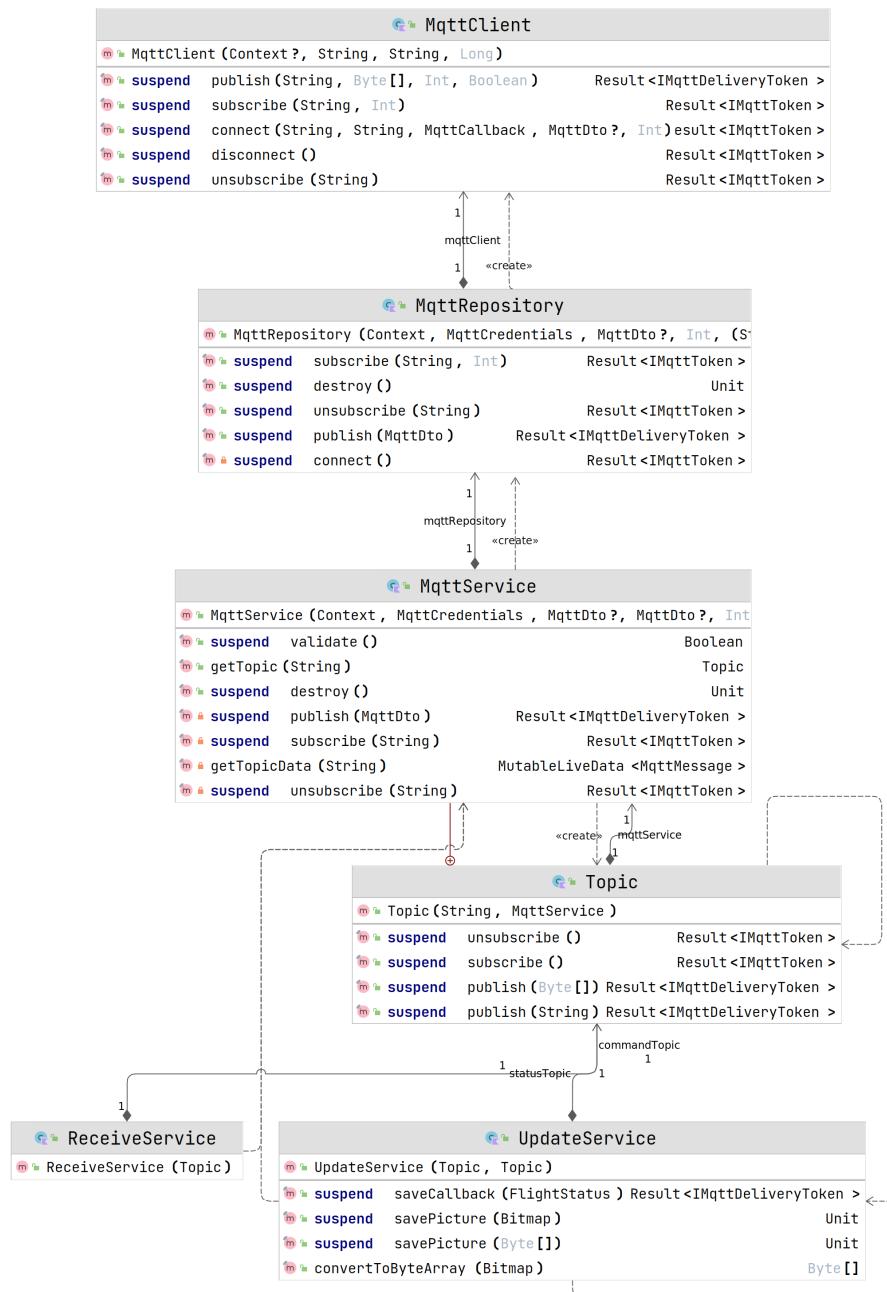
W przypadku, w którym definicja komend miała by się ograniczać do definiowania w obiekcie typu enum, w *CommandHandler* należało by obsłużyć każdy przypadek osobno, za pomocą struktury *when*. Dodatkowo każda jej realizacja musiałaby być zdefiniowana w tej klasie, co zmniejszałoby czytelność całości. Z każdą kolejną komendą klasa ta rozrastałaby się do coraz większych rozmiarów. Takie podejście stanowiłoby naruszenie zasad SOLID *Open/closed principle* i *Single responsibility principle*. Zasady te zostały przedstawione przez autorytet programistów, Roberta C. Martina i są uznawane za dobre praktyki programowania obiektowego.[38]



Rys. 20. Diagram klas klasy służącej do przechowywania stanu BSP
 Źródło: Własne

Diagram klas przedstawia `FlightStatus`, która będzie przechowywała dane dotyczące statusu BSP w danej chwili czasu. Posiada ona tylko metodę `toJson`. Zwraca ona stan obiektu w formacie JSON, które będzie przesyłany za pomocą serwisu MQTT.

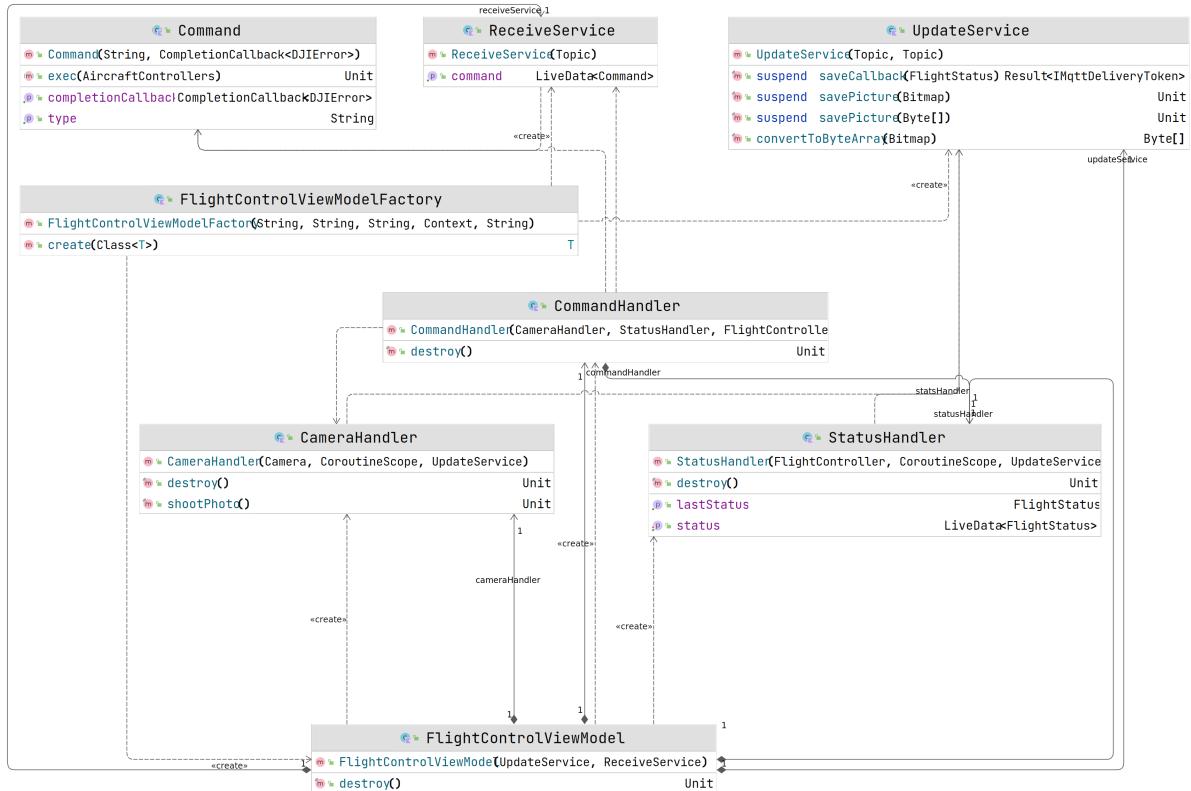
Klasy, które nie realizują żadnej logiki, a służą do przechowywania danych, określa się jako DTO (Data Transfer Object).



Rys. 21. Diagram klas klasy ograniczony do serwisów dzielących w ramach systemu

Źródło: Własne

TODO



Rys. 22. Diagram klas klasy ograniczony do klas działających na DJI SDK

Źródło: Własne

TODO

III.7. Wykorzystane urządzenia



Rys. 23. DJI Mini 2

Źródło: <https://laptrinhx.com/dji-mini-2-review-same-compact-size-more-confidence-flying-3163139258/>

W trakcie implementacji i testów zostaną wykorzystane następujące urządzenia:

- **Laptop**, który posłuży jako serwer MQTT, a także do wykonywania skryptów wysyłających do systemy określone komendy.
- **Dron DJI Mini 2 wraz z kontrolerem**, czyli produkt docelowy, którego sterowaniem ma umożliwiać zaimplementowany system.
- **Smartfon z systemem Android**, który będzie podłączony do kontrolera DJI. Aplikacja mobilna na tym urządzeniu będzie odbierała komendy z serwera MQTT, które będą definiowały jak sterować dronem. Ponadto będzie na bieżąco przesyłała status drona do serwera. Wszystko to będzie odbywało się na unikalnym topicu MQTT, który będzie zawierał numer seryjny urządzenia.



Rys. 24. Kontroler do DJI Mini 2

Źródło: <https://www.gohero.pl/userdata/public/gfx/6191/Kontroler-do-DJI-Mavic-Air-2.jpg>

Rozdział IV. Implementacja systemu

IV.1. Klasa komend i jej wykonywanie

Kod. 1. Klasa: Command

```

1 abstract class Command(val type: String, val completionCallback: CommonCallbacks.CompletionCallback<DJIError>) {
2     abstract fun exec(aircraftControllers: AircraftControllers)
3
4     companion object {
5         const val TAG = "Command"
6     }
7 }
```

Źródło: Badania własne.

Kod. 2. Klasa: StartMotorsCommand

```

1 class StartMotorsCommand(completionCallback : CommonCallbacks.CompletionCallback<DJIError>) : Command(type, completionCallback) {
2     companion object {
3         const val type = "start_motors"
4     }
5
6     override fun exec(aircraftControllers: AircraftControllers) {
7         val status = aircraftControllers.statsHandler.getLastStatus()
8         if (!status.isFlying && !status.motorsOn) {
9             aircraftControllers.flightController.turnOnMotors(
10                 completionCallback
11             )
12         } else {
13             FeedbackUtils.setResult("Forbidden_state_can't_start_motors", TAG)
14         }
15     }
16 }
```

Źródło: Badania własne.

Kod. 3. Fragment kodu z CommandHandler

```
1 private val commandObserver = Observer<Command> {
2     it.exec(aircraftControllers)
3 }
```

Źródło: Badania własne.

Kod. 4. Klasa: CommandFactory

```
1 class CommandFactory {
2     companion object {
3         const val TAG = "CommandFactory"
4     }
5
6     private val gson: Gson = Gson()
7
8     fun from(value: String): Command {
9         return try {
10             val jsonObject = gson.fromJson(value, JsonObject::class.java)
11             val missionType = jsonObject.get("type").asString
12             createCommand(missionType, jsonObject)
13         } catch (e: Exception) {
14             FeedbackUtils.setResult(e.toString(), level = LogLevel.ERROR, tag = TAG)
15             UnrecognizedCommand(value, getCompletionCallback(UnrecognizedCommand.type))
16         }
17     }
18
19     private fun getCompletionCallback(type: String): CompletionCallbackImpl<DJIError> {
20         //returns commpletion callback object
21     }
22
23     @Throws(Throwable::class)
24     private fun createCommand(missionType: String, jsonObject: JsonObject): Command {
25         return when (missionType) {
26             StartGoHomeCommand.type -> {
27                 StartGoHomeCommand(getCompletionCallback(StartGoHomeCommand.type))
28             }
29             LandCommand.type -> {
30                 LandCommand(getCompletionCallback(LandCommand.type))
31             }
32             ShootPhotoCommand.type -> {
33                 ShootPhotoCommand(getCompletionCallback(ShootPhotoCommand.type))
34             }
35             // ...
36             //and more cases
37             // ..
38             PauseWaypointCommand.type -> {
39                 PauseWaypointCommand(getCompletionCallback(PauseWaypointCommand.type))
40             }
41             else -> {
42                 UnrecognizedCommand(
43                     jsonObject.toString(),
44                     getCompletionCallback(UnrecognizedCommand.type)
45                 )
46             }
47         }
48     }
49 }
50 }
```

Źródło: Badania własne.

W ramach systemu utworzono klasę abstrakcyjną *Command*. Reprezentuje ona komendy, które mogą zostać wysłane do wykonania przez BSP. Jej abstrakcyjna metoda

exec, odpowiada za zrealizowanie tej komendy na dronie. Operacje te są wykonywane za pomocą obiektu *AircraftControllers*, który jest praktycznie wrapelem na obiekty z SDK DJI, za pomocą których można wykonywać operacje np. startu. *StartMotorsCommand* jest przykładem jej rozszerzenia, realizuje ona całą operację włącznie silników w BSP.

Odpowiedzialnością *CommandFactory* jest generowanie instancji klas dziedziczących po *Command*, na podstawie dostarczonego obiektu tekstowego w formacie JSON. Realizuje to metoda *from*, która obsługuje przypadki, w których wartość tekstowa nie może zostać poprawnie rozpoznana, wtedy zwracany jest obiekt *UnrecognizedCommand*.

Takie podejście architektoniczne sprawia, że realizacja komendy w *CommandHandler* ogranicza się do 3 linii kodu. Dodanie kolejnej komendy do systemu ogranicza się do zdefiniowania klasy dziedziczącej po *Command* i dodanie jednego przypadku w strukturze *when* w metodzie *createCommand* z *CommandFactory*.

W przypadku, w którym definicja komend miała by się ograniczać do definiowania w obiekcie typu enum, w *CommandHandler* należało by obsługiwać każdy przypadek osobno, za pomocą struktury *when*. Dodatkowo każda jej realizacja musiałaby być zdefiniowana w tej klasie, co zmniejszałoby czytelność całości. Z każdą kolejną komendą klasa ta rozrastałaby się do coraz większych rozmiarów. Takie podejście stanowiłoby naruszenie zasad SOLID *Open/closed principle* i *Single responsibility principle*. Zasady te zostały przedstawione przez autorytet programistów, Roberta C. Martina i są uznawane za dobre praktyki programowania obiektowego.[38]

IV.2. Obsługa MQTT

Kod. 5. Klasa *MqttService*

```

1 class MqttService(
2     context: Context,
3     mqttCredentials: MqttCredentials,
4     lastWill: MqttDto? = null,
5     birth: MqttDto? = null,
6     keepAliveInterval: Int = GlobalConfig.KEEP_ALIVE_INTERVAL,
7 ) {
8
9     private val subscribedTopics: MutableSet<String> = HashSet()
10    private val messagesArrived: HashMap<String, MutableLiveData<MqttMessage>> =
11        HashMap()
12    private val connectionLostFun: (throwable: Throwable) -> Unit = { cause ->
13        Log.d(this.javaClass.name, "Connection_lost_$cause")
14
15        Log.d(this.javaClass.name, "Trying_to_subscribe_last_topic")
16        subscribedTopics.stream()
17            .forEach { DjiaApplication.mainScope.launch(Dispatchers.IO) { mqttRepository
18                .subscribe(it) } }
19    }
20
21    private val deliveryCompleteFun: (token: IMqttDeliveryToken) -> Unit = {
22        Log.d(this.javaClass.name, "Delivery_completed")
23    }
24
25    private val messageArrivedFun: (topic: String, message: MqttMessage) -> Unit =
26        { topic, message ->
27            if (subscribedTopics.contains(topic)) {
28                Log.d(this.javaClass.name, "Receive_message:_$message_from_topic:_"
29                    $topic")
30                getTopicData(topic).postValue(message)
31
32            } else {
33                Log.e(

```

```

29             this.javaClass.name,
30             "Receive_message:_$message_from_unsubscribed_topic:_$topic"
31         )
32     }
33 }
34
35 private val mqttRepository = MqttRepository(
36     context,
37     mqttCredentials,
38     lastWill,
39     keepAliveInterval,
40     messageArrivedFun,
41     connectionLostFun,
42     deliveryCompleteFun
43 )
44
45 init {
46     this.let { service ->
47         DjiaApplication.mainScope.launch(Dispatchers.IO) {
48             birth?.let { service.publish(it) }
49         }
50     }
51 }
52
53
54 private fun getTopicData(topic: String): MutableLiveData<MqttMessage> {
55     return messagesArrived
56         .getOrPut(topic) { MutableLiveData() }
57 }
58
59 private suspend fun unsubscribe(value: String): Result<IMqttToken> {
60     val res = mqttRepository.unsubscribe(value)
61     if (res.isSuccess) {
62         subscribedTopics.remove(value)
63     }
64     return res
65 }
66
67 private suspend fun subscribe(value: String): Result<IMqttToken> {
68     val res = mqttRepository.subscribe(value)
69     if (res.isSuccess) {
70         subscribedTopics.add(value)
71     }
72     return res
73 }
74
75 private fun getSubscribed(): Set<String> {
76     return subscribedTopics
77 }
78
79 private suspend fun publish(data: MqttDto): Result<IMqttDeliveryToken> {
80     return mqttRepository.publish(data)
81 }
82
83 fun getTopic(value: String): Topic {
84     return Topic(value = value, this)
85 }
86
87 suspend fun destroy() {
88     mqttRepository.destroy()
89 }
90
91 suspend fun validate(): Boolean {
92     return mqttRepository.publish(MqttDto(ETopic.VALIDATE, "validate" + UUID.
93         randomUUID()))
94         .isSuccess
95 }
```

```

95     class Topic(private val value: String, private val mqttService: MqttService) {
96         //... inner class
97     }
98
99 }
```

Źródło: Badania własne.

Kod. 6. Klasa wewnętrzna *Topic*

```

1  class Topic(private val value: String, private val mqttService: MqttService) {
2      suspend fun publish(payload: ByteArray): Result<IMqttDeliveryToken> {
3          return mqttService.publish(MqttDto(value, payload))
4      }
5
6      suspend fun publish(payload: String): Result<IMqttDeliveryToken> {
7          return mqttService.publish(MqttDto(value, payload))
8      }
9
10     fun isSubscribed(): Boolean {
11         return mqttService.getSubscribed().contains(value)
12     }
13
14     suspend fun subscribe(): Result<IMqttToken> {
15         return mqttService.subscribe(value)
16     }
17
18     suspend fun unsubscribe(): Result<IMqttToken> {
19         return mqttService.unsubscribe(value)
20     }
21
22     fun getData(): LiveData<MqttMessage> {
23         if (!isSubscribed()) {
24             Log.w(this.javaClass.name, "getData_without_subscribing_topic_$value")
25         }
26         return mqttService.getTopicData(value)
27     }
28
29     fun getValue(): String {
30         return value
31     }
32 }
```

Źródło: Badania własne.

Rozdział V. Testy systemu oraz prezentacja użycia na wybranym case study

TODO

V.1. Testy jednostkowe TODO

W kontekście testowania oprogramowania na platformie Android można wyróżnić dwa rodzaje testów jednostkowych.

- **androidTest** - Testy, które są uruchamiane na rzeczywistych lub wirtualnych urządzeniach z systemem Android. Obejmują one m.in. test integracyjne i end-to-end. Szczególnie takie, w których samo JVM nie może sprawdzić działania kodu. W ramach tego rodzaju testów przetestowano serwisy odpowiedzialne za komunikacje za pomocą MQTT. Konieczność ta jest spowodowana tym, że są to operacje asynchroniczne, które do swojego wykonania korzystają z narzędzi do zarządzania cyklem życia obiektów dostarczanych przez system Android, dlatego muszą być wykonane w ramach tego kontekstu.
- **test** - Klasyczne testy jednostkowe, które mogą być uruchamiane na lokalnej maszynie JVM. W ramach tych testów przetestowano m.in. mapowanie obiektów JSON na klasy wykorzystywane do wykonywania poszczególnych komend (klasa *pl.edu.wat.droman.data.model.command.Command*) na urządzeniu.

V.1.1. Testowanie MqttService

Klasa *pl.edu.wat.droman.data.service.MqttService* zarządza Topic-ami MQTT. Dostarcza ona łatwy w obsłudze abstrakt (klasa *pl.edu.wat.droman.data.service.MqttService.Topic*), a jego wygenerowanie odbywa się na podstawie podanej ścieżki dla transmisji Mqtt. Można na nim wykonywać opcje publikowania i subskrybowania wiadomości. Ponieważ zwraca ona obiekty za pomocą klasy *androidx.lifecycle.LiveData* operacje te są wykonywane z kontekstem aplikacji. Dodatkowo w całym procesie wykorzystywany jest do tego serwer MQTT, którego parametry logowania i adresu są pobierane z pliku konfiguracyjnego gradle, który jest przechowywany na lokalnej maszynie. Taki zastosowanie pozwala na nie przechowywania kluczy wraz z repozytorium, co jest istotne, jeżeli udostępnia się kod na zewnętrzne repozytorium lub uruchamia testy na środowiskach z różnym dostępem do serwera MQTT.

Kod. 7. Inicjalizowanie wartości w klasie przed uruchomieniem poszczególnych testów

```

1  @Before
2  fun init() {
3      appContext = InstrumentationRegistry.getInstrumentation().targetContext
4
5      val metadata: Bundle = appContext.packageManager.getApplicationInfo(
6          appContext.packageName,
7          PackageManager.GET_META_DATA
8      ).metaData
9
10     password = metadata.getString("mosquitto.password")!!
11     user = metadata.getString("mosquitto.user")!!

```

```

12         uri = "tcp://" + metadata.getString("mosquitto.ip")
13         mqttService = MqttService(appContext, MqttCredentials(uri, clientID, user,
14             password))
    }
```

Źródło: Badania własne.

Przed wszystkimi testami wykonywane jest inicjowanie obiektów, które będą wykorzystywane w trakcie wszystkich testów. W tym przypadku jest to wyciągnięcie z metadanych danych logowania do serwera MQTT, a następnie utworzenie testowanego serwisu.

Kod. 8. Test publikowania danych za pomocą Mqtt

```

1  @Test
2  fun publish() = runBlocking {
3      //given
4
5      val message = "message:" + UUID.randomUUID()
6      val topicVal = "/test"
7
8      //then
9      val topic = mqttService.getTopic(topicVal)
10     val res = topic.publish(message)
11
12     //except
13     Assert.assertTrue(res.isSuccess)
14 }
```

Źródło: Badania własne.

Jest to prosty przypadek testowy który sprawdza czy publikowanie wiadomości do serwera MQTT odbywa się bez zwrócenia błędu.

Kod. 9. Test pobierania danych za pomocą Mqtt

```

1  @Test
2  fun getData() {
3      //given
4      val message = "message:" + UUID.randomUUID()
5      val topicVal = "/test"
6
7      //then
8      val topic = mqttService.getTopic(topicVal)
9      runBlocking {
10          topic.subscribe()
11          topic.publish(message)
12      }
13      val result = topic.getData().getOrAwaitValue(time = 5).toString()
14
15      //except
16      Assert.assertEquals(message, result)
17 }
```

Źródło: Badania własne.

Drugi z prezentowanych testów sprawdza cały proces przesyłania wiadomości za pomocą MQTT, od włączenia subskrypcji wskazanego Topic-a, aż do pozyskania danych na tym Topic-u ze wskazanego serwera.

V.1.2. Testowanie CommandFactory

Klasa *CommandFactory* odpowiada za generowanie obiektów klas komend, na podstawie obiektu, które są w późniejszym etapie wykonywane za pomocą klasy *Comman-*

Run: MqttServiceTest ×		4 passed	4 tests, 53s 518ms
Filter tests:		Duration	asus ASUS...
Tests			
Test Results		673 ms	4/4
MqttServiceTest		673 ms	4/4
publish		514 ms	✓
getData		59 ms	✓
subscribe		1 ms	✓
unsubscribe		99 ms	✓

Rys. 25. Wynik testów MqttRepository

Źródło: Własne

dHandler. Klasy te są generowane na podstawie obiektu String w formacie JSON.

Kod. 10. Test tworzenia komendy *UploadWaypointCommand* za pomocą *CommandFactory*

```

1  @Test
2  fun testMappingToUploadMissionCommand() {
3      //given
4      val commandFactory = CommandFactory()
5      val missionValue = "{\n" +
6          "    \"type\": \"upload_waypoint_mission\", \n" +
7          "    \"finished_action\": \"NO_ACTION\", \n" +
8          "    \"auto_flight_speed\": 0.01, \n" +
9          "    \"max_flight_speed\": 0.5, \n" +
10         "    \"heading_mode\": \"AUTO\", \n" +
11         "    \"waypoints\": [\n" +
12             {\n" +
13                 "        \"latitude\": 1.0, \n" +
14                 "        \"longitude\": 1.0, \n" +
15                 "        \"latitude\": 1.0\n" +
16             }, \n" +
17             {\n" +
18                 "        \"latitude\": 1.0, \n" +
19                 "        \"longitude\": 1.0, \n" +
20                 "        \"latitude\": 1.0\n" +
21             }, \n" +
22             {\n" +
23                 "        \"latitude\": 1.0, \n" +
24                 "        \"longitude\": 1.0, \n" +
25                 "        \"latitude\": 1.0\n" +
26             }\n" +
27         ]\n" +
28     }"
29
30     //then
31     val command = commandFactory.from(missionValue)
32
33     //expect
34     assertNotNull(command)
35     assertEquals(UploadWaypointCommand.type, command.type)
36     assertEquals(UploadWaypointCommand::class.java, command.javaClass)
37 }
```

Źródło: Badania własne.

Powyżej przykładowy test, w którym niestety nie można przetestować bezpośrednio wartości m.in. współrzędnych, ponieważ dostęp do nich jest chroniony. W aplikacji został wykorzystany framework Junit4, który nie umożliwia dostępu do obiektów chro-

nionych, ale warto dodać, że inne, jak np. Spock, już takie rzeczy umożliwiają. Sam test służy do upewnienia się, że wartość tekstowa została zmapowana na poprawny obiekt.

Kod. 11. Test tworzenia komendy *TakeOffCommand* za pomocą *CommandFactory*

```

1  @Test
2  fun testMappingToTakeOffCommand() {
3      //given
4      val commandFactory = CommandFactory()
5      val missionValue = "{\"type\":\"take_off\"}"
6      //then
7      val command = commandFactory.from(missionValue)
8
9      //expect
10     assertNotNull(command)
11     assertEquals(TakeOffCommand::class.java, command.javaClass)
12     assertEquals(TakeOffCommand.type, command.type)
13 }
```

Źródło: Badania własne.

Klasa *TakeOffCommand* nie posiada w sobie żadnych parametrów, które wpływają na jej wykonanie, więc problem, o którym była mowa powyżej nie występuje.

Kod. 12. Test tworzenia komendy *UnrecognizedCommand* w przypadku nieroznaznania wartości za pomocą *CommandFactory*

```

1  @Test
2  fun testMappingToFailUploadMissionCommand() {
3      //given
4      val commandFactory = CommandFactory()
5      val missionValue = "{\"type\":\"dsadas\"}"
6      //then
7      val command = commandFactory.from(missionValue)
8
9      //expect
10     assertNotNull(command)
11     assertEquals(UnrecognizedCommand::class.java, command.javaClass)
12     assertEquals(UnrecognizedCommand.type, command.type)
13 }
14 }
```

Źródło: Badania własne.

Ten test sprawdza przypadek, w którym z jakiegoś powodu nie będzie można utworzyć instancji klasy *Command*, wtedy zostaje zwrócona specjalna komenda *UnrecognizedCommand*. Taki sposób działa jest lepszy od zwracania wartości *null*, co jest uznaniane za złą praktykę.

V.2. Testy autonomiczne

W ramach systemy przygotowano dwa skrypty w Pythonie, który umożliwia przeprowadzanie testów automatycznych.

V.2.1. Publikowanie na topicu przeznaczonego do przesyłania komend

Kod. 13. Skrypt przeznaczony do publikowania na nasłuchiwanym przez urządzenie latające topicu komend.

```
1 if __name__ == '__main__':
```

```

2  client = mqt.Client()
3  client.username_pw_set(username=USERNAME, password=PASSWORD)
4  client.on_connect = on_connect
5  client.connect(MQTT_SERVER, 1883, 60)
6  allowed_args = ["test", "set_home_location", "land", "upload_waypoint_mission", "
    take_off", "start_motors",
7      "stop_motors", "shoot_photo", "load_waypoint_mission", "
    take_off_and_land", "stop_waypoint_mission",
8          "start_waypoint_mission", "go_home"]
9  print(allowed_args)
10 if sys.argv[1] == "test":
11     client.publish(topic=MQTT_PATH + CLIENT_ID, payload='{"type":"shoot_photo"}')
12     time.sleep(1)
13     client.publish(topic=MQTT_PATH + CLIENT_ID, payload='{"type":"start_motors"}')
14     time.sleep(3)
15     client.publish(topic=MQTT_PATH + CLIENT_ID, payload='{"type":"stop_motors"}')
16 elif sys.argv[1] == "set_home_location":
17     payload = {
18         "type": "set_home_location",
19         "longitude": 20.0,
20         "latitude": 113.0,
21     }
22     client.publish(topic=MQTT_PATH + CLIENT_ID, payload=json.dumps(payload))
23 elif sys.argv[1] == "load_waypoint_mission":
24     payload = {
25         "type": "load_waypoint_mission",
26         "finished_action": "NO_ACTION",
27         "auto_flight_speed": 1.0,
28         "max_flight_speed": 5.0,
29         "heading_mode": "AUTO",
30         "waypoints": [
31             {
32                 "altitude": 10,
33                 "longitude": START_POINT_LONGITUDE,
34                 "latitude": START_POINT_LATITUDE + ONE_METER_OFFSET * 0.5
35             },
36             {
37                 "altitude": 11,
38                 "longitude": START_POINT_LONGITUDE + ONE_METER_OFFSET * 1,
39                 "latitude": START_POINT_LATITUDE - ONE_METER_OFFSET * 1
40             },
41             {
42                 "altitude": 12,
43                 "longitude": START_POINT_LONGITUDE + ONE_METER_OFFSET * 2,
44                 "latitude": START_POINT_LATITUDE + ONE_METER_OFFSET * 1
45             }
46         ]
47     }
48     print(json.dumps(payload))
49     client.publish(topic=MQTT_PATH + CLIENT_ID, payload=json.dumps(payload))
50 elif sys.argv[1] == "take_off_and_land":
51     client.publish(topic=MQTT_PATH + CLIENT_ID, payload='{"type":"take_off"}')
52     time.sleep(3)
53     client.publish(topic=MQTT_PATH + CLIENT_ID, payload='{"type":"shoot_photo"}')
54     time.sleep(3)
55     client.publish(topic=MQTT_PATH + CLIENT_ID, payload='{"type":"land"}')
56 elif sys.argv[1] in allowed_args:
57     client.publish(topic=MQTT_PATH + CLIENT_ID, payload='{"type":"' + sys.argv[1] +
58         '"}')
58 else:
59     print("forbidden_command_" + sys.argv[1])

```

Źródło: Badania własne.

Przedstawiony powyżej skrypt umożliwia w prosty sposób przesyłanie komend do urządzenia końcowego. Dodatkowo w ramach skryptu zostały udostępnione dwie se-

kwencje *test* i *test_with_take_off*. Pierwsza z nich wykonuje trzy następujące po sobie komendy, które odpowiadają kolejno za:

- uruchomienie silników;
- zrobienie zdjęcia;
- wyłącznie silników.

Druga z nich wykonuje dodatkowo wystartowanie, tj. uniesienie się na wysokość 1,4 metra i wylądowanie. Metoda *test* w trakcie implementacji stanowiła szybko weryfikacje działania, nawet w zamkniętym pomieszczeniu.

V.2.2. Subskrybowanie topicu przeznaczonego do przesyłania zdjęć

Kod. 14. Skrypt nasuchojący na podanym topicu MQTT i zapisujący dane z wiadomości MQTT w formacie /textjpeg

```

1 def on_connect(_client, userdata, flags, rc):
2     print("Connected with result code " + str(rc))
3     _client.subscribe(MQTT_PATH)
4
5
6 def on_message(_client, userdata, msg):
7     f = open('output.jpeg', "wb")
8     f.write(msg.payload)
9     print("Image Received")
10    f.close()
11
12
13 if __name__ == '__main__':
14     client = mqtt.Client()
15     client.username_pw_set(username=USERNAME, password=PASSWORD)
16     client.on_connect = on_connect
17     client.on_message = on_message
18     client.connect(MQTT_SERVER, 1883, 60)
19     client.loop_forever()
```

Źródło: Badania własne.

Ten test służy do sprawdzenia, czy wysłany obraz z urządzenia DJI został przekazane w formacie, który umożliwia jego rozkodowanie. W trakcie jego działania należy wykonać komendę wcześniej opisanym skryptem, która wysyła do urządzenia żądanie zrobienie zdjęcia. Po jego wykonaniu jest ono automatycznie wysyłane na wskazany topic MQTT.

V.3. Coś o sprawdzeniu statusu statku powitronego

TODO

Podsumowanie

Bibliografia

- [1] Sarah E. Kreps “Drony. Wprowadzenie Technologie Zastosowania” Wydawnictwo Naukowe PWN, Warszawa, 2019;
- [2] A. Kelsey “Flying Robots 101: Everthing You Need to Know about Drones”, Popular Science, TODO TOM, TODO STRONY, 08-03-2013
- [3] “NB-IoT vs Lora” <https://ubidots.com/blog/lorawan-vs-nb-iot/#lorawan-vs-nb-iot-a-quick-overview> [dostęp: 20-04-2022];
- [4] “machine-to-machine (M2M)” <https://www.techtarget.com/iotagenda/definition/machine-to-machine-M2M> [dostęp: 20-04-2022];
- [5] “LPWA wikipedia” <https://pl.wikipedia.org/wiki/LPWAN> [dostęp: 20-04-2022];
- [6] “LoRa Technology - An Overview, IEEE, 2018” <https://ieeexplore.ieee.org/document/8474715> [dostęp: 20-04-2022];
- [7] “Wikipedia: Yuneec International” https://en.wikipedia.org/wiki/Yuneec_International [dostęp: 25-04-2022];
- [8] “On the Performance of Narrow-band Internet of Things (NB-IoT) for Delay-tolerant Services, IEEE, 2019” <https://ieeexplore.ieee.org/document/8768871> [dostęp: 20-04-2022];
- [9] “Wikipedia: SZ DJI Technology Co., Ltd.”, <https://en.wikipedia.org/wiki/DJI> [dostęp: 20-04-2022];
- [10] “De Havilland DH-82 "Tiger Moth"("Queen Bee"), 1931”, <http://www.samolotypolskie.pl/samoloty/782/126/De-Havilland-DH-82-Tiger-Moth-Queen-Bee> [dostęp: 22-04-2022];
- [11] “DJI market share: here’s exactly how rapidly it has grown in just a few years”, <https://www.thedronegirl.com/2018/09/18/dji-market-share/> [dostęp: 22-04-2022];
- [12] “Wojna w Ukrainie: Pasjonaci dronów namierają rosyjskie wojska”, <https://fotoblogia.pl/17711,wojna-w-ukrainie-pasjonaci-dronow-namierzaja-rosyjskie-wojska> [dostęp: 22-04-2022];
- [13] “Tureckie drony na Ukrainie pokazały wojnę przyszłości. Bayraktar TB2 wyrządzają ogromne szkody”, <https://www.chip.pl/2022/03/tureckie-drony-w-ukrainie-pokazaly-wojne-przyszlosci-bayraktar-tb2-wyrzadzaja-ogromne-szkod> [dostęp: 22-04-2022];

- [14] "Zaskakująca skuteczność Bayraktarów. Ekspert o rosnącej roli dronów w wojnie", <https://www.pap.pl/aktualnosci/news%2C1129159%2Czaskakujaca-skutecznosc-bayraktarow-ekspert-o-rosnacej-roli-dronow-w> [dostęp: 22-04-2022];
- [15] "DJI Mavic Mini SE", <https://www.dji.com/pl/mini-se?site=brandsite&from=nav> [dostęp: 20-04-2022];
- [16] "DJI store", <https://store.dji.com> [dostęp: 20-04-2022];
- [17] "DJI Lightbridge", <https://www.dji.com/pl/dji-lightbridge/info> [dostęp: 20-04-2022];
- [18] "DJI Lightbridge2", <https://www.dji.com/pl/lightbridge-2/info#specs> [dostęp: 20-04-2022];
- [19] "Wikipedia: OFDM", <https://pl.wikipedia.org/wiki/OFDM> [dostęp: 20-04-2022];
- [20] "Wikipedia: Ryan Model 147 Lightning Bug", https://en.wikipedia.org/wiki/Ryan_Model_147 [dostęp: 22-04-2022];
- [21] "Wikipedia: MQ-1 Predator", https://en.wikipedia.org/wiki/General_Atomics_MQ-1_Predator [dostęp: 22-04-2022];
- [22] "Wikipedia: Baykar", <https://en.wikipedia.org/wiki/Baykar> [dostęp: 22-04-2022];
- [23] "Wikipedia: FHSS", <https://pl.wikipedia.org/wiki/FHSS> [dostęp: 20-04-2022];
- [24] "DJI Mavic 2 - Ocusync 2.0 What is it & What's Compatible ? + How is it different from Lightbridge", <https://www.youtube.com/watch?v=gfqcSv9sR0A> [dostęp: 20-04-2022];
- [25] "DJI Google", https://u.cyfrowe.pl/600x0/2/7/2_732250420.png [dostęp: 20-04-2022];
- [26] "Konkurs MON na bezzałogowe sytemy powietrzne, lądowe, morskie", <https://www.wojsko-polskie.pl/wat/articles/aktualnosci-w-konkurs-mon-na-bezzalogowe-systemy-powietrzne-ladowe-i-morskie/> [dostęp: 21-04-2022];
- [27] "Wikipedia: General Atomics Aeronautical Systems", https://en.wikipedia.org/wiki/General_Atomics_Aeronautical_Systems [dostęp: 26-04-2022];
- [28] "General Dynamics: Our history", <https://www.gd.com/about-gd/our-history> [dostęp: 26-04-2022];
- [29] "PANSA: Ruszają regularne loty transportowe BSP", <https://www.pansa.pl/ruszaja-regularne-loty-transportowe-bsp/> [dostęp: 28-04-2022];

- [30] "Akcja TOPR dron dostarczył koce i ogrzewacze", <http://www.swiatdronow.pl/akcja-topr-dron-dostarczyl-koce-i-ogrzewacze/> [dostęp: 28-04-2022];
- [31] "Drones to the Rescue? How Drones are Changing the Landscape of High Mountain Rescue Efforts", <https://snowbrains.com/drones-to-the-rescue-how-drones-are-changing-the-landscape-of-high-mountain-rescue-efforts/> [dostęp: 29-04-2022];
- [32] "Drones guide rescuers to hunter lost in North Carolina woods, officials say", <https://www.newsobserver.com/news/state/north-carolina/article260714992.html> [dostęp: 29-04-2022];
- [33] "Wleciał dronem do wnętrza wulkanu na Islandii. Ostatnie sekundy nagrania jeżą włosy na głowie [WIdeo]", <https://www.twojapogoda.pl/wiadomosc/2021-06-02/wlecial-dronem-do-wnetrza-wulkanu-na-islandii-ostatnie-sekundy-nagrania-jeza-wlosy-na-glow> [dostęp: 29-04-2022];
- [34] "It's a bird! It's a plane! It's a drone that makes movies!", <https://www.washingtonpost.com/news/the-switch/wp/2013/08/15/its-a-bird-its-a-plane-its-a-drone-that-makes-movies/> [dostęp: 29-04-2022];
- [35] "How to Build a Drone: Construct Your Drone from Scratch", <https://www.mydroneLab.com/blog/how-to-build-a-drone.html> [dostęp: 29-04-2022];
- [36] "Drone Delivery Was Supposed to be the Future. What Went Wrong?", <https://www.youtube.com/watch?v=J-M98KLgaUU> [dostęp: 29-04-2022];
- [37] "Drony przetransportują sprzęt medyczny nad Sosnowcem. Przed nami spotkanie z mieszkańcami", <https://sosnowiec.naszmiasto.pl/drony-przetransportuja-sprzet-medyczny-nad-sosnowcem-przed/ar/c1-8493233> [dostęp: 29-04-2022];
- [38] "SOLID czyli dobre praktyki w programowaniu obiektowym", <https://www.samouczekprogramisty.pl/solid-czyli-dobre-praktyki-w-programowaniu-obiektowym/> [dostęp: 20-05-2022];

Spis rysunków

Rys. 1. Latający gołąb Archytasa z Tarentu	7
Rys. 2. <i>De Havilland Queen Bee</i> i premier Wielkiej Brytanii Winston Churchill ...	8
Rys. 3. <i>Ryan Model 147 Lightning Bug</i> umieszczony pod skrzydłem samolotu transportowego	8
Rys. 4. <i>MQ-1 Predator</i> , wyposażony w rakiety <i>AGM-114 Hellfire</i>	9
Rys. 5. Bayraktar TB2	10
Rys. 6. Dron osobowy Cezarei w trakcie testów w Stambule	12
Rys. 7. Dron <i>Farada G1</i> , za pomocą którego będzie obywał się transport medyczny w okolicach Warszawy	14
Rys. 8. Strefy wyłączone dla dronów w okolicach Warszawy.....	16
Rys. 9. Kontroler lotu Pixhawk	17
Rys. 10.DJI Tello	23
Rys. 11.Pierwsza wersja gogli do FPV od DJI	24
Rys. 12.Dji OcuSync Air Unit	24
Rys. 13.DJI Phantom v2.0	25
Rys. 14.DJI RC plus	25
Rys. 15.Widmo OFDM i FHSS	26
Rys. 16.Widmo OcuSync z zaznaczoną modulacją FHSS i OFDM	26
Rys. 17.Porównanie widma OcuSync i Lightbridge	27
Rys. 18.Diagram komponentów	30
Rys. 19.Diagram klas ograniczony do dostępnych komend w ramach systemu.....	31
Rys. 20.Diagram klas klasy służącej do przechowywania stanu BSP	33
Rys. 21.Diagram klas klasy ograniczony do serwisów działających w ramach systemu	35
Rys. 22.Diagram klas klasy ograniczony do klas działających na DJI SDK	36
Rys. 23.DJI Mini 2	37
Rys. 24.Kontroler do DJI Mini 2	38
Rys. 25.Wynik testów MqttRepository	45

Spis tabel

Tab. 1. Porównanie rodzajów technologii M2M.[6]	20
Tab. 2. Porównanie technologii LoRa i NB-IoT [3]	22

Załączniki

1. Płyta CD/DVD zawierająca:
 - a) Prezentację wyników pracy dyplomowej
 - b) Kody źródłowe oprogramowania
 - c) Biblioteki programowe niezbędne do zbudowania i uruchomienia oprogramowania
 - d) ...
2. ...