

**POLITECHNIKA WARSZAWSKA**

**Wydział Mechatroniki**

**Praca dyplomowa inżynierska**

Michał Kapiczyński

Analiza możliwości wykorzystania darmowych map do utworzenia prostego programu do śledzenia położenia obiektów wraz z wizualizacją obszaru przeszukanego

Opiekun pracy:  
dr inż. Paweł Wnuk

Jednostka dyplomująca

Warszawa, 2016

# Spis treści

<b>1. WSTĘP .....</b>	<b>5</b>
<b>1.1. Wprowadzenie .....</b>	<b>5</b>
<b>2. DZIEDZINA PROBLEMU .....</b>	<b>7</b>
<b>2.1. Najważniejsze pojęcia .....</b>	<b>7</b>
<b>2.2. Geodezyjne powierzchnie odniesienia .....</b>	<b>8</b>
<b>2.2.1. Geoida .....</b>	<b>9</b>
<b>2.2.2. Elipsoida.....</b>	<b>10</b>
<b>2.2.3. Porównanie modeli WGS84 oraz EGM96.....</b>	<b>11</b>
<b>2.3. Numeryczny Model Terenu NMT .....</b>	<b>12</b>
<b>2.3.1. Ogólnie o NMT.....</b>	<b>12</b>
<b>2.3.2. Misja SRTM .....</b>	<b>13</b>
<b>2.4. System GPS .....</b>	<b>14</b>
<b>2.4.1. Ogólny opis.....</b>	<b>15</b>
<b>2.4.2. Zasada działania .....</b>	<b>15</b>
<b>2.4.3. Dokładność.....</b>	<b>16</b>
<b>2.4.4. Określanie wysokości .....</b>	<b>16</b>
<b>3. TECHNOLOGIE DOSTĘPNE NA RYNKU.....</b>	<b>18</b>
<b>3.1. Wybór rodzaju aplikacji klienckiej .....</b>	<b>18</b>
<b>3.1.1. Aplikacja desktopowa.....</b>	<b>19</b>
<b>3.1.2. Aplikacja webowa.....</b>	<b>19</b>
<b>3.1.3. Aplikacja mobilna.....</b>	<b>19</b>
<b>3.1.4. Podsumowanie wyboru.....</b>	<b>20</b>
<b>3.2. Wybór platformy .....</b>	<b>20</b>
<b>3.2.1. Android.....</b>	<b>20</b>
<b>3.2.2. iOS.....</b>	<b>21</b>
<b>3.2.3. Windows Phone .....</b>	<b>21</b>
<b>3.2.4. Podsumowanie wyboru.....</b>	<b>21</b>
<b>3.3. Wybór narzędzia .....</b>	<b>22</b>
<b>3.3.1. SDK .....</b>	<b>22</b>
<b>3.3.2. NDK.....</b>	<b>22</b>
<b>3.3.3. Xamarin .....</b>	<b>23</b>
<b>3.3.4. HTML5 .....</b>	<b>23</b>
<b>3.3.5. Podsumowanie wyboru.....</b>	<b>23</b>
<b>3.4. Wybór sposobu komunikacji między serwerem, a aplikacją kliencką .....</b>	<b>24</b>
<b>3.4.1. Model TCP/IP.....</b>	<b>24</b>
<b>3.4.2. HTTP .....</b>	<b>25</b>
<b>3.4.3. WebSocket.....</b>	<b>26</b>
<b>3.4.4. Podsumowanie wyboru.....</b>	<b>28</b>
<b>3.5. Wybór formatu przesyłanych danych .....</b>	<b>28</b>
<b>3.5.1. XML.....</b>	<b>29</b>
<b>3.5.2. JSON.....</b>	<b>29</b>
<b>3.5.3. Podsumowanie wyboru.....</b>	<b>30</b>

<b>3.6. Opis API OpenStreetMap .....</b>	<b>30</b>
<b>4. ANALIZA SYSTEMU.....</b>	<b>32</b>
<b>4.1. Opis systemu .....</b>	<b>32</b>
<b>4.2. Identyfikacja aktorów.....</b>	<b>33</b>
4.2.1. <i>Użytkownik systemu.....</i>	34
4.2.2. <i>Administrator systemu .....</i>	34
4.2.3. <i>Użytkownik aplikacji wizualizującej.....</i>	34
4.2.4. <i>Dron.....</i>	34
<b>4.3. Przypadki użycia .....</b>	<b>34</b>
4.3.1. <i>Przypadki użycia - administrator systemu .....</i>	35
4.3.2. <i>Przypadki użycia - dron .....</i>	36
4.3.2.1. Scenariusz przypadku użycia - Zmień położenie.....	36
4.3.3. <i>Przypadki użycia - użytkownik aplikacji wizualizującej .....</i>	37
4.3.3.1. Scenariusz przypadku użycia - Załóż konto .....	38
4.3.3.2. Scenariusz przypadku użycia - Zaloguj się.....	39
4.3.3.3. Scenariusz przypadku użycia - Oglądaj wizualizację .....	40
4.3.3.4. Scenariusz przypadku użycia - Edytuj preferencje wizualizacji .....	41
4.3.3.5. Scenariusz przypadku użycia - Oglądaj symulację.....	42
4.3.3.6. Scenariusz przypadku użycia - Przeglądaj historię wizualizacji.....	43
<b>5. ALGORYTM OBLICZANIA OBSZARU PRZESZUKANEGO.....</b>	<b>44</b>
<b>5.1. Założenia .....</b>	<b>44</b>
<b>5.2. Implementacja .....</b>	<b>45</b>
5.2.1. <i>Dane wejściowe .....</i>	45
5.2.2. <i>Algorytm wyznaczania otoczki obszaru przeszukanego.....</i>	46
5.2.3. <i>Algorytm wyznaczania obszarów wewnętrz otoczki, niezarejestrowanych przez kamerę .....</i>	49
5.2.4. <i>Algorytm łączenia obszarów.....</i>	52
5.2.4.1. Definicja problemu.....	52
5.2.4.2. Wyznaczanie otoczki wewnętrznej - kształt α .....	53
5.2.4.3. Łączenie obszarów, a części zasłonięte wewnątrz obszarów.....	59
5.2.5. <i>Podsumowanie.....</i>	60
<b>6. PROJEKT SYSTEMU .....</b>	<b>61</b>
<b>6.1. Diagram komponentów.....</b>	<b>62</b>
<b>6.2. Diagramy klas .....</b>	<b>63</b>
6.2.1. <i>Diagram klas części serwerowej .....</i>	63
6.2.2. <i>Diagram klas aplikacji klienckiej .....</i>	63
<b>6.3. Diagram związków encji.....</b>	<b>64</b>
<b>6.4. Diagramy aktywności.....</b>	<b>65</b>
6.4.1. <i>Diagram aktywności - Dron - Zmień położenie.....</i>	65
6.4.2. <i>Diagram aktywności - Użytkownik aplikacji wizualizującej - Oglądaj wizualizację .....</i>	66
6.4.3. <i>Diagram aktywności - Użytkownik aplikacji wizualizującej - Edytuj preferencje .....</i>	67
6.4.4. <i>Diagram aktywności - Użytkownik aplikacji wizualizującej - Przeglądaj historię wizualizacji .....</i>	68
6.4.5. <i>Diagram aktywności - Użytkownik aplikacji wizualizującej - Oglądaj Symulację .....</i>	69
<b>6.5. Diagramy sekwencji .....</b>	<b>70</b>
6.5.1. <i>Diagram sekwencji - DronTracker - Zmiana położenia .....</i>	70

6.5.2. <i>Diagram sekwencji - DronServer - Nowa wiadomość od DronTracker'a</i> .....	71
6.5.3. <i>Diagram sekwencji - DronVision - Nowa wiadomość wizualizacyjna</i> .....	72
<b>6.6. Projekt interfejsu graficznego (GUI).....</b>	<b>73</b>
<b>7. PREZENTACJA ROZWIĄZANIA .....</b>	<b>74</b>
<b>7.1. Opis aplikacji DronTracker .....</b>	<b>74</b>
<b>7.2. Opis aplikacji DronServer .....</b>	<b>75</b>
<b>7.3. Opis aplikacji DronVision .....</b>	<b>75</b>
7.3.1. <i>Widoki logowania</i> .....	76
7.3.2. <i>Widoki rejestracji</i> .....	77
7.3.3. <i>Panel nawigacyjny - menu</i> .....	78
7.3.4. <i>Widoki wizualizacji</i> .....	79
7.3.5. <i>Widok ustawień preferencji wizualizacji</i> .....	80
7.3.6. <i>Widoki historii wizualizacji</i> .....	81
7.3.7. <i>Widoki symulacji</i> .....	82
7.3.8. <i>Widoki dodatkowe</i> .....	83
<b>7.4. Testy.....</b>	<b>83</b>
<b>8. PODSUMOWANIE .....</b>	<b>84</b>
<b>9. BIBLIOGRAFIA.....</b>	<b>86</b>
<b>10. LISTA ZAŁĄCZNIKÓW .....</b>	<b>90</b>

# **1. Wstęp**

## **1.1. Wprowadzenie**

Branża bezzałogowych statków latających, czy jak są one powszechnie nazywane dronów jest w tej chwili jedną z najbardziej przyszłościowych i najszybciej rozwijających się gałęzi rynku nowoczesnych technologii na świecie.

Pojazdy te przez długi czas były wykorzystywane głównie w przemyśle militarnym, jednak na przestrzeni ostatnich lat coraz częściej znajdują zastosowanie na rynku usług i w przemyśle sektora cywilnego. Są wykorzystywane chociażby przez służby ratunkowe do poszukiwania osób zaginionych lub ofiar katastrof, do patrolowania granic, monitorowania upraw rolnicznych, obserwacji zwierząt na pastwiskach, nagrywania wydarzeń z perspektywy niedostępnej dla człowieka, kontroli bezpieczeństwa podczas imprez masowych czy też na dużych obszarach rekreacyjnych lub terenach zagrożonych pożarami. Są one doskonałym rozwiązaniem zawsze tam, gdzie trzeba przeprowadzić badania, bądź obserwacje w trudno dostępnych lub toksycznych dla człowieka warunkach np. w miejscach skażonych, bądź w górach, czy w gęstych lasach.

Jednoczesne wykorzystanie dronów wraz z zaawansowanymi systemami wizyjnymi, postępujący rozwój systemów informacji geograficznych GIS oraz łatwość integracji z urządzeniami mobilnymi stwarzają szerokie możliwości na opracowywanie nowych rozwiązań wspomagających wykorzystanie bezzałogowych statków latających.

Niniejsza praca skupia się właśnie na analizie i realizacji takiego rozwiązania, czyli systemu informatycznego służącego do kontroli położenia i wizualizacji obszaru przeszukanego, rozumianego jako reprezentacja terenu zarejestrowanego przez kamery zamontowane na dronach.

Celem niniejszej pracy jest opracowanie algorytmu umożliwiającego wyznaczanie wspomnianego obszaru przeszukanego przez bezzałogowe statki powietrzne, na podstawie zebranych danych geolokalizacyjnych obiektów i znajomości parametrów kamer oraz zaprojektowanie i implementacja systemu informatycznego, umożliwiającego określanie położień geograficznych pojazdów w czasie rzeczywistym oraz ich wizualizację na urządzeniach mobilnych wraz z analizą możliwości wykorzystania do tego celu systemu OpenStreetMap.

Na system składają się aplikacja mobilna do zbierania danych satelitarnych, pełniąca rolę geolokalizatora do potencjalnego zamontowania na dronie, część serwerowa komunikują-

jąca się z innymi elementami systemu i zajmująca się analizą zebranych danych, wraz z wyznaczaniem obszaru przeszukanego na podstawie opracowanego w trakcie tworzenia tej pracy algorytmu, aplikacja administracyjna do zarządzania dronami, użytkownikami i ich uprawnieniami oraz mobilna aplikacja kliencka służąca do wizualizacji zebranych danych.

Potencjalnych możliwości zastosowania projektowanego systemu jest tak samo wiele, ile jest sposobów wykorzystania dronów. Co więcej istnieje szeroka gama rozwiązań, umożliwiających w przyszłości rozszerzanie podstawowych funkcji wizualizujących zapewnianych przez system, poprzez dodatkową integrację z czujnikami, czy bardziej zaawansowanymi systemami wizyjnymi, takimi jak chociażby systemy termowizyjne. Na potrzeby tej pracy system został opracowany z myślą o zadaniach związanych z nadzorem i kontrolą dużych obszarów terytorialnych, takich jak obszary przemysłowe lub rolnicze przez bezzałogowe statki powietrzne.

## **2. Dziedzina problemu**

Głównym aspektem teoretycznym, związanym z projektowanym systemem jest sposób określania położenia i obliczania wysokości nad ziemią statków. Należy zrozumieć takie kwestie jak to: jak działa system nawigacji satelitarnej GPS, jakie są rodzaje wysokości i co się przyjmuje za model odniesienia w przypadku geolokalizacji. Dodatkowo, aby być w stanie odnieść dane geolokalizacyjne drona uzyskane z pomiarów do rzeczywistej powierzchni ziemskiej potrzebna jest znajomość zestawu danych modelujących kształt kuli ziemskiej. Ten rozdział został poświęcony właśnie tym kwestiom.

### **2.1. Najważniejsze pojęcia**

Na początku zostaną przedstawione najważniejsze pojęcia związane z teoretycznym aspektem pracy. W dalszych podrozdziałach zostaną one omówione w bardziej szczegółowy sposób.

**Bezzałogowy statek powietrzny BSP** - (UAV – Unmanned Aircraft Vehicle) - potocznie dron. Jest to pojazd, który nie wymaga do lotu załogi obecnej na pokładzie oraz nie ma możliwości zabierania pasażerów, pilotowany zdalnie lub wykonujący lot autonomicznie na podstawie zaprogramowanej trasy [5].

**Dane geolokalizacyjne** - dane określające położenie geograficzne obiektów w przestrzeni w stosunku do przyjętego układu odniesienia.

**System wizyjny** - układ współpracujących ze sobą elektronicznych urządzeń, którego funkcją jest automatyczna analiza wizyjna otoczenia na podobieństwo zmysłu wzroku u ludzi [39].

**Obszar przeszukany** - reprezentacja obszaru zarejestrowanego przez kamerę zamontowaną w dronie.

**Numeryczny model terenu NMT** - numeryczna reprezentacja powierzchni ziemskiej [4].

**SRTM** - Shuttle Radar Topography Mission - międzynarodowa misja kosmiczna, mającą na celu zebranie najbardziej kompleksowego i dokładnego numerycznego modelu terenu (NMT) Ziemi [1].

**Geoida** - teoretyczna powierzchnia ekwipotencjalna, pokrywająca się w przybliżeniu z powierzchnią oceanów przy pełnej równowadze znajdujących się w nich mas wody [2].

**Elipsoida ziemska** - spłaszczona elipsoida obrotowa, której powierzchnia jest najbardziej zbliżona do powierzchni geoidy na całej Ziemi [2].

**Wysokość ortometryczna** - odległość mierzona w stosunku do geoidy, wzduż linii pionu w rzeczywistym polu siły ciężkości, utożsamiana z wysokością nad poziomem morza [15].

**Wysokość elipsoidalna** - odległość mierzona w stosunku do elipsoidy [15].

**Undulacja** - różnica wysokości między geoidą, a elipsoidą [6].

**WGS84** - model powierzchni ziemskiej opracowany w stosunku do elipsoidy WGS84.

**EGM96** - model powierzchni ziemskiej opracowany w stosunku do geoidy EGM96.

## 2.2. Geodezyjne powierzchnie odniesienia

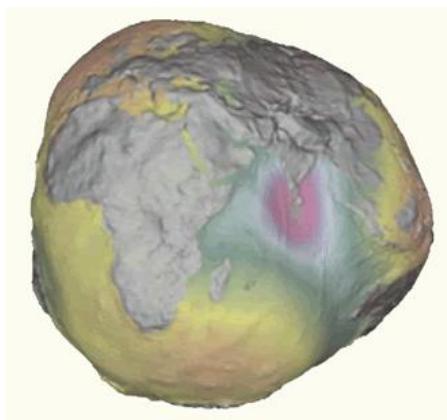
Geolokalizacja to proces określania położenia geograficznego przedmiotów lub osób w stosunku do przyjętego układu odniesienia, związanego z Ziemią. Jednak powszechnie wiadomo, że rzeczywisty kształt powierzchni ziemskiej jest bardzo nieregularny i praktycznie niemożliwy do opisania matematycznego. Wpływa na niego wiele czynników takich jak: ruch obrotowy i obiegowy planety, cieplny i grawitacyjny wpływ ciał niebieskich, własności fizyczne litosfery i hydrosfery i wiele innych [10]. Systemy nawigacyjne, aby być w stanie określić położenie obiektów w stosunku do powierzchni ziemskiej muszą znać jej rzeczywisty kształt i rozmiar, a przynajmniej takie ich przybliżenie, które zapewni wymaganą dokładność wykonywania pomiarów i obliczeń nawigacyjnych. W związku z tym w rzeczywistości wy-

korzystuje się modele Ziemi, które w zależności od skali i zakładanej dokładności za powierzchnię odniesienia przyjmują geoidę lub elipsoidę.

### 2.2.1. Geoida

Jeśli wyobrażymy sobie, że powierzchnia ziemska jest w pełni pokryta wodą oraz zignorujemy wpływy falowań oraz prądów morskich, na cząsteczki wody powstałego w ten sposób "globalnego oceanu" będzie oddziaływać jedynie siła grawitacji. Otrzymana w ten sposób powierzchnia nie będzie jednak regularna ponieważ jej kształt będzie zależny od nieregularnego rozmieszczenia masy wewnętrz planety, wpływającego na kierunki działania siły grawitacji dla każdego punktu powierzchni. W rzeczywistości na nieregularność poziomu wód ma wpływ więcej czynników takich jak temperatura wody, jej zasolenie, czy wpływ grawitacyjny innych obiektów takich jak góry. Geoida jest zatem teoretyczną powierzchnią ekwipotencjalną, pokrywającą się w przybliżeniu z powierzchnią oceanów przy pełnej równowadze znajdujących się w nich mas wody i jest w każdym swym punkcie prostopadła do kierunku siły ciężkości [6].

Ze względów praktycznych geoida często jest utożsamiana ze średnim poziomem morza, chociaż w rzeczywistości w niektórych miejscach może odbiegać od niego nawet o kilka metrów.

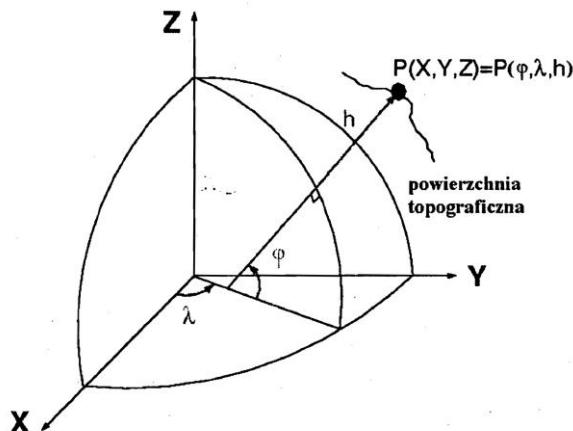


Rys. 2.1 Model powierzchni geoidy, pokazujący stopień jej skomplikowania [6]

## 2.2.2. Elipsoida

Elipsoida jest powierzchnią powstałą na skutek obrotu elipsy wokół jej osi symetrii. W przypadku Ziemi osią tą jest mała oś elipsy, odpowiadająca osi ziemskiej. Elipsoidą ziemską nazywamy spłaszczoną elipsoidę obrotową, której powierzchnia jest najbardziej zbliżona do hydrostatycznej powierzchni Ziemi. Współcześnie parametry elipsoidy wyznaczane są na podstawie pomiarów satelitarnych.

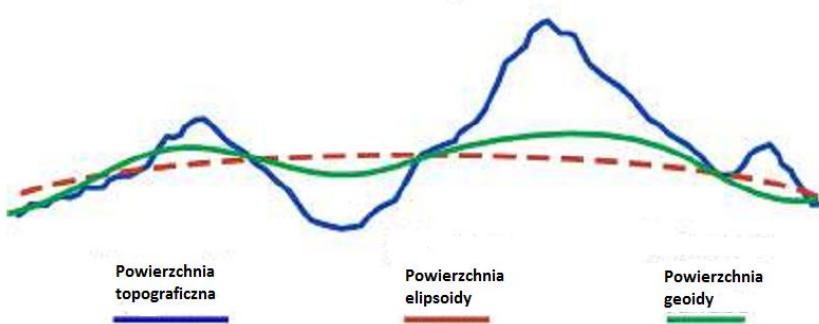
Z elipsoidą ziemską poza układem współrzędnych kartezjańskich, wiąże się układ współrzędnych geograficznych (elipsoidalnych). W układzie tym występują trzy współrzędne: szerokość geograficzna, długość geograficzna oraz wysokość elipsoidalna. Szerokością geograficzną nazywamy kąt  $\varphi$  zawarty między normalną do elipsoidy w punkcie obserwatora, a płaszczyzną równika. Długością geograficzną nazywamy kąt dwuścienny  $\lambda$  pomiędzy płaszczyzną południka zerowego i płaszczyzną południka obserwatora. Wysokość elipsoidalna  $h$  to odległość euklidesowa pomiędzy punktem obserwatora, a jego rzutem wzduż normalnej na elipsoidzie [2].



Rys. 2.2 Współrzędne geograficzne (elipsoidalne) [2]

Powiązanie układu współrzędnych geograficznych z punktem związanym z Ziemią daje nam ziemski układ odniesienia. Natomiast powiązanie ze środkiem mas Ziemi i nadanie wartości liczbowych poszczególnym parametrom elipsoidy (oraz określenie pewnym stałych geofizycznych) daje nam geocentryczny układ odniesienia [2].

Powierzchnie geoidy oraz elipsoidy znacznie się różnią [Rys. 2.3]. Odstępy między geoidą, a elipsoidą nazywamy undulacjami geoidy, a ich wartości wahają się od -110m do +84m [15].



Rys. 2.3 Porównanie powierzchni [9]

### 2.2.3. Porównanie modeli WGS84 oraz EGM96

Biorąc pod uwagę wysoce nieregularny kształt Ziemi praktycznie niemożliwym jest stworzenie modelu jednolicie dokładnego dla całej planety. W związku z tym przy dokładnych pomiarach i badaniach często wykorzystuje się modele opracowane lokalnie. Jednak popularyzacja systemu nawigacji satelitarnej oraz konieczność współpracy międzynarodowej przy badaniach wymagały stworzenia modeli, które mogłyby być wykorzystywane na całej planecie.

Grawitacyjny model EGM96 w bardzo dokładny sposób definiuje globalną geoidę ziemską, w ogólności tożsamą ze średnim poziomem morza, podczas gdy WGS84 przybliża powierzchnię ziemską jako geocentryczną elipsoidę, generalizującą kształt tej geoidy.

EGM96 jest modelem znacznie bardziej złożonym i dokładnym, opartym na szczegółowej analizie ziemskiej siły grawitacyjnej, jednak jego użycie wymaga znacznie większych nakładów obliczeniowych. WGS84 określa elipsoidę najbardziej pasującą do geoidy EGM96. Model ten został utworzony w 1984 roku, jednak na przestrzeni lat był aktualizowany wraz z coraz to dokładniejszymi pomiarami geoidy. WGS84 nie jest modelem przestarzałym, jest po prostu znacznie uproszczonym modelem matematycznym, który mimo mniejszej dokładności wciąż jest często wykorzystywany ze względu na łatwość wykorzystania w analizie matematycznej. Model ten dla większości zastosowań jest wystarczająco dokładny. Jego niedokładność osiąga maksymalnie wartość 110m ze średnim błędem na poziomie 20-30m. Istnieje możliwość zredukowania tego błędu poprzez wprowadzenie korekty na podstawie znajomości undulacji geoidy na interesującym nas regionie.

## **2.3. Numeryczny Model Terenu NMT**

### **2.3.1. Ogólnie o NMT**

Teoretyczne modele planety byłyby bezużyteczne, gdyby nie dało się ich wykorzystać w obliczeniach matematycznych. W tym celu wykorzystuje się właśnie numeryczny model rzeźby terenu (NMT), będący numeryczną reprezentacją powierzchni ziemskiej, utworzoną zazwyczaj przez zbiór punktów tej powierzchni oraz algorytmy, służące do aproksymacji jej położenia i kształtu na podstawie współrzędnych  $x$ ,  $y$ ,  $z$  tych punktów. Zawiera informacje na temat wysokości poszczególnych punktów powierzchni terenu ponad ustalonym poziomem odniesienia np. nad poziomem morza [3].

Numeryczny model terenu można opracować na podstawie dowolnego zestawu danych mających charakter pola skalarnego, tzn. zawierających informacje o położeniu (współrzędne  $x$  i  $y$  w układzie współrzędnych) oraz wielkości skalara (współrzędna  $z$ ) [3].

Skonstruowanie prawidłowego modelu rzeźby terenu jest jednym z kluczowych wyzwań współczesnej geomatyki. Znajomość Numerycznego Modelu Terenu o wysokiej dokładności jest bardzo ważna z punktu widzenia wielu zastosowań praktycznych. Modele wysokości, w formie map topograficznych umożliwiają projektowanie przebiegu tras komunikacyjnych, wyznaczanie optymalnych lokalizacji masztów telefonii komórkowej, czy też wyznaczanie stref zalewowych. Dane reprezentujące NMT są niezbędne do prawidłowego funkcjonowania powietrznych systemów nawigacyjnych oraz powszechnie wykorzystywane w licznych Systemach Informacji Geograficznej [3].

Dane do opracowania NMT mogą być pozyskiwane na wiele sposobów takich jak: bezpośrednie pomiary terenowe, zdjęcia lotnicze i ich opracowania fotogrametryczne, mapy topograficzne, interferometria satelitarna, czy lotniczy skaning laserowy. Najczęściej stosowaną techniką pozyskiwania danych jest technologia fotogrametryczna. Na podstawie odpowiednich zdjęć lotniczych (tzw. fotogramów) budowany jest model stereoskopowy, na którym wykonuje się następnie stereodigitalizację powierzchni terenu, co w efekcie umożliwia odtwarzanie kształtów, rozmiarów i wzajemnego położenia obiektów w terenie. Tworzenie NMT na obszarach silnie zalesionych, bądź zurbanizowanych jest możliwe za pomocą technologii kartograficznej, polegającej na wektoryzacji warstw na podstawie zeskanowanych diapozytywów map topograficznych. Do szybkiej budowy NMT o dużej dokładności jest stosowany skaning laserowy, który jest niezależny od warunków oświetlenia i pozwala na uzy-

skanie precyzyjnego modelu rzeźby terenu, także dla obszarów o zwartej pokrywie roślinnej, czy w terenach zabudowy miejskiej [3].

Wymienione wyżej sposoby gromadzenia danych do modelowania rzeźby terenu łączy jedna wspólna cecha. Mają naturę lokalną tzn. wymagają w mniejszym lub większym stopniu bezpośredniej eksploracji badanego terenu, co jest bardzo kosztowne i czasochłonne oraz często niemożliwe z powodów politycznej niedostępności niektórych regionów świata. W efekcie przez wiele lat nie istniał jednolity dla całej ziemi numeryczny model terenu. Większość rozwiniętych państw tworzyło swoje własne dane kartograficzne, które znacznie różniły się między sobą pod względem skali, rozdzielczości oraz punktów odniesienia i co za tym idzie były wysoce niespójne. Co więcej globalny obszar pokrycia był bardzo niejednolity. Wiele części świata takich jak Ameryka Południowa, czy Afryka cechowało się brakiem danych topograficznych o wysokiej dokładności [1].

Uzyskanie międzynarodowych, cyfrowych danych wysokościowych, o spójnej skali i rozdzielczości, okazało się praktycznie niemożliwe z wykorzystaniem opisanych wcześniej metod, a jedynym sposobem na ich ujednolicenie było zastosowanie globalnie jednolitej techniki pozyskiwania danych. Taką techniką okazała się interferometria satelitarna wykorzystana podczas misji SRTM.

### **2.3.2. Misja SRTM**

SRTM - The Shuttle Radar Topography Mission - jest międzynarodową misją kosmiczną mającą na celu zebranie najbardziej kompleksowego i dokładnego numerycznego modelu terenu (NMT) Ziemi [1].

Projekt został przeprowadzony w lutym 2000 roku podczas lotu promu kosmicznego Endeavour wspólnie przez trzy agencje kosmiczne: Narodową Agencję Aeronautyki i Przestrzeni Kosmicznej Stanów Zjednoczonych NASA, Niemiecką Agencję Kosmiczną DRL oraz Włoską Agencję Kosmiczną ASI [1].

Główną ideą misji było zapewnienie jednolitego pod względem dokładności i skali modelu topograficznego o rozdzielczości terenowej 1" dla wszystkich obszarów lądowych znajdujących się pomiędzy równoleżnikami  $60^{\circ}$  szerokości geograficznej północnej, a  $56^{\circ}$  szerokości geograficznej południowej, stanowiących około 80% powierzchni lądowej Ziemi. Jednym z założeń misji był maksymalny błąd wysokości, jakim miał się charakteryzować

NMT, który został ustalony na +/- 16m dla wysokości bezwzględnej oraz +/- 10m dla wysokości względnej na poziomie ufności równemu 90% [1].

W trakcie trwania całej misji zebrano ponad 12 terabajtów danych, które następnie zostały poddane dokładnej analizie. W sierpniu 2001 roku ogłoszono zakończenie misji SRTM, a dane wynikowe zostały umieszczone w Internecie [1].

Zgodnie z oficjalną specyfikacją misji dokładność pionowa danych wysokościowych SRTM-1 na obszarze Europy wynosi 6,2m dla wysokości bezwzględnej oraz 8,7m dla wysokości względnej. Błąd geolokalizacji jest na poziomie 8,8m. Wartości te są różne w zależności od kontynentu, ale wszędzie mieszczą się w założonej dokładności +/- 16m [1].

Wadą interferometrii satelitarnej do pozyskiwania modelu NMT jest uwzględnianie pokrywy roślinnej i zabudowy przy określaniu wysokości, co znacznie zmniejsza dokładność danych na tego typu obszarach [1].

Numeryczne modele SRTM są bezpłatnie dostępne w Internecie w formie plików reprezentujących rastrowy numeryczny model terenu o zadanej rozdzielcości na poziomie 1", 3" lub 30" w zależności od wersji danych. Modele te są szeroko wykorzystywane do badań naukowych takich jak badania potencjału energii wiatrowej, analiza geomorfometryczna, czy analiz hydrologicznych. Są również podstawowym źródłem danych wysokościowych dla wielu aplikacji takich jak Google Earth, Nasa World Wind, Google Maps, Yahoo Maps, czy OpenStreetMap [7].

Wysokości w modelu SRTM-C odniesione są do geoidy EGM-96, natomiast położenia pikseli są przedstawione względem elipsoidy WGS 84 [7].

## 2.4. System GPS

Istnieje wiele sposobów określania położen geograficznych obiektów. Do najbardziej znanych należą: ustalanie lokalizacji na podstawie "widoczności" innych obiektów o znanej pozycji, takich jak np. stacje bazowe telefonów komórkowych, geolokalizacja z wykorzystaniem radaru lub sonaru, polegająca na wyznaczaniu położenia geograficznego na podstawie znajomości własnego położenia i wektora wodzącego względem innego obiektu oraz geolokalizacja IP, czyli określanie położenia na podstawie adresu IP urządzenia i bazy adresów. Jednak najczęściej wykorzystywanym systemem określania położenia jest system GPS [40].

#### **2.4.1. Ogólny opis**

Skrót GPS pochodzi od angielskich słów Global Positioning System, czyli Światowy System Określania Położenia. Jest to amerykański system nawigacji satelitarnej przeznaczony do szybkiego i dokładnego wyznaczania współrzędnych geograficznych, określających pozycję anteny odbiornika w przestrzeni, obejmujący zasięgiem całą kulę ziemską.

#### **2.4.2. Zasada działania**

Zasada działania systemu opiera się na pomiarze odległości pomiędzy odbiornikiem sygnału GPS, a sztucznym satelitą, nadającym ten sygnał. Satelita jest ciałem niebieskim o względnie małej masie, obiegającym inne ciało o większej masie po torze nazywanym orbitom, a sztuczny satelita jest obiektem wprowadzonym przez człowieka na orbitę wokół planety [14].

Na wysokości ponad 20 000 km nad powierzchnią Ziemi, we fragmencie przestrzeni okołoziemskiej nazywanym strefą orbit średnich znajduje się 31 sztucznych satelitów, krążących po orbitach nachylonych do powierzchni równika pod kątem 55°. Każdy z tych satelitów transmituje dwa rodzaje sygnałów: L1(1575,42 MHz) i L2 (1227,60MHz). Sygnał L1 jest przetwarzany dwoma pseudo-przypadkowymi sygnałami zagłuszającymi: chronionym kodem P i kodem C/A (Course Aquisition - pseudolosowa sekwencja bitów, powtarzająca się cyklicznie co 1ms). Sygnał L2 zawiera jedynie kod P. Każdy satelita wysyła inny sygnał, co ułatwia odbiornikom rozpoznanie, z którego satelity pochodzi dany sygnał [12].

Zasada działania systemu opiera się na pomiarze odległości pomiędzy satelitą, który porusza się po ściśle wyznaczonej orbicie, a odbiornikiem. Znana odległość od satelity lokuje odbiornik na sferze o promieniu równym zmierzonej odległości. Znana odległość od dwóch satelitów lokuje odbiornik na okręgu będącym przecięciem dwóch sfer. Kiedy odbiornik zmierzy odległość od trzech satelitów, istnieją tylko dwa punkty, w których może się znajdować. Jeden z tych punktów można wykluczyć jako znajdujący się zbyt wysoko lub poruszający się zbyt szybko i w ten sposób wyznaczyć swoją pozycję. Aby poznać odległość od satelitów emitujących bardzo słabe sygnały (o mocy zbliżonej do szumu tła) dokonuje się pomiaru opóźnienia sygnału odebranego z poszczególnych satelitów. Odbiornik GPS dysponuje jednak tylko własnym zegarem kwarcowym. Wyznaczanie godziny z dokładnością do nanose-

kund odbywa się poprzez odbieranie sygnału nie od trzech, a od czterech satelitów. Można wówczas wyliczyć zarówno rzeczywisty czas, jak i położenie [17].

### **2.4.3. Dokładność**

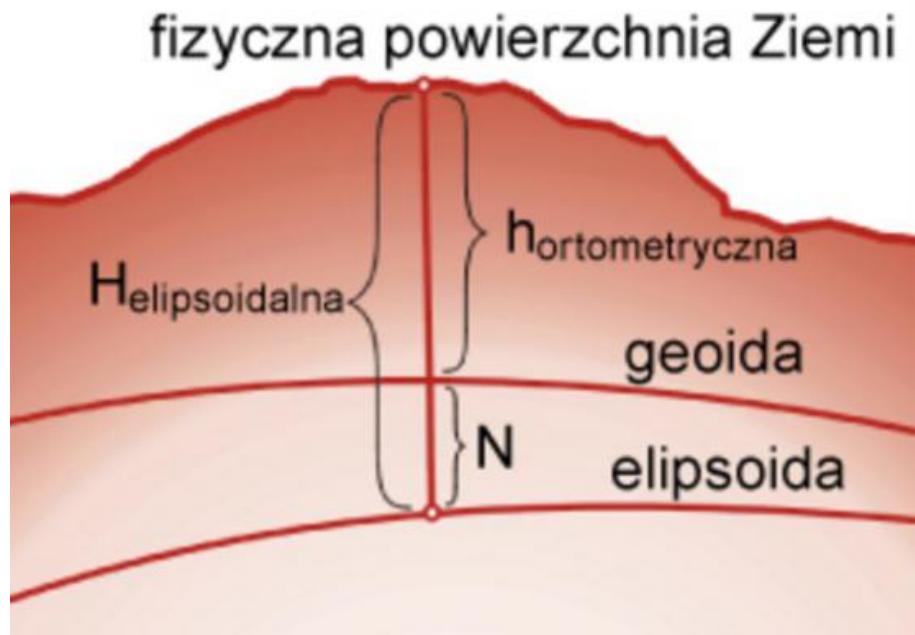
Dla typowych zastosowań dokładność pozycjonowania z wykorzystaniem systemu GPS wynosi do 10m. W przypadku dodatkowych wymogów dokładnościowych stosuje się metody zmniejszające błąd pozycjonowania takie jak metody uśredniające lub pomiar wzajemny DGPS. Można wówczas uzyskać dokładność na poziomie kilku metrów [17].

Błąd wyznaczania wysokości jest zazwyczaj 1,5 razy większy od błędu położenia poziomego. Wynika to z geometrii rozmieszczenia satelitów, z których korzysta odbiornik, na orbitach. W celu uzyskania jak najbardziej dokładnej wysokości, należy używać satelitów zlokalizowanych jak najdalej od siebie, ale niezbyt nisko nad horyzontem oraz jednego dokładnie nad głową. Z reguły jednak odbiornik częściej wybiera satelity bliższe linii horyzontu w celu uzyskania bardziej dokładnej pozycji poziomej, na której zależy większość użytkowników odbiorników GPS. Powoduje to duże różnice wartości wysokości w danym punkcie [17].

### **2.4.4. Określanie wysokości**

W wyniku pomiarów GPS otrzymujemy przestrzenne współrzędne prostokątne X, Y, Z pozycji anten odbiorników, które mogą następnie zostać przeliczone na współrzędne geodezyjne B, L, H odniesione do układu WGS84, czyli do elipsoidy WGS84, gdzie B i L to długość i szerokość geograficzna, a H to wysokość elipsoidalna [15].

Jednak rzadko kiedy wysokość elipsoidalna jest wysokością docelową, którą chcemytrzymać. Zazwyczaj interesuje nas wysokość w odniesieniu do poziomu morza, który jak zostało opisane w podrozdziale "Geodezyjne powierzchnie odniesienia" w przybliżeniu odpowiada geoidzie.



Rys. 2.4 Pomiar wysokości [15]

Wysokość liczona od poziomu morza jest określana wysokością ortometryczną i otrzymujemy ją odejmując od wysokości elipsoidalnej, otrzymanej z pomiarów GPS, wysokość geoidy N, czyli wartość odstępu między geoidą, a elipsoidą w danym punkcie. Wartości odstępów geoidy od elipsoidy wahają się od -110m do +84m. W Europie elipsoida WGS84 jest średnio o około 30m wyżej niż rzeczywisty poziom morza [15]. Część nowoczesnych odbiorników GPS posiada wbudowaną możliwość konwersji wysokości, jednak większość dostępnych urządzeń na rynku nie posiada takiej opcji. W celu określenia wysokości nad poziomem morza należy wówczas wprowadzić własne poprawki na podstawie znajomości lokalnych undulacji geoidy.

Alternatywą dla pomiaru wysokości z wykorzystaniem systemu GPS są pomiary różnicowe. Przykładem takiego pomiaru jest pomiar, wykorzystujący wysokościomierz barometryczny, określający wysokość na podstawie zależności między zmianą ciśnienia atmosferycznego, a zmianą wysokości [13]. Jednak ze względu na brak czujników barometrycznych w większości urządzeń mobilnych dostępnych na rynku, w pracy zostanie wykorzystany pomiar wysokości z wykorzystaniem systemu nawigacji satelitarnej.

### **3. Technologie dostępne na rynku**

Posiadając wiedzę związaną z aspektem teoretycznym projektowanego systemu można przejść do kolejnego etapu, którym jest wybór technologii i narzędzi informatycznych umożliwiających jego stworzenie. W przypadku projektów informatycznych nie istnieje nigdy jedna jedyna technologia, konkretnie przypisana do danego zastosowania. W zależności od wielkości systemu, jego przeznaczenia oraz grupy docelowej klientów istnieje szereg rozwiązań pozwalających na realizację założeń. W tym rozdziale zostanie dokonany opis i porównanie dostępnych na rynku technologii związanych z głównymi aspektami projektowanego systemu oraz zostaną wybrane te, które zostaną uznane za najlepsze dla danego rozwiązania.

Przed rozpoczęciem rozważań na temat dostępnych technologii należy zdefiniować poszczególne komponenty systemu. Projektowany system został podzielony na cztery części, o różnej funkcjonalności:

- część serwerowa - zapewniająca komunikację między pozostałymi elementami systemu oraz odpowiadająca za przeprowadzanie skomplikowanym obliczeń,
- aplikacja geolokalizacyjna - geolokalizator - prosta aplikacja do zbierania danych geolokalizacyjnych,
- część administracyjna - panel administracyjny do zarządzania dronami, użytkownikami i ich uprawnieniami,
- aplikacja kliencka - aplikacja służąca do wizualizacji położen i obszaru przeszukanego przez bezzałogowe statki powietrzne,

#### **3.1. Wybór rodzaju aplikacji klienckiej**

Podchodząc do procesu projektowania systemu informatycznego jedną z podstawowych kwestii jaką należy rozważyć jest urządzenie docelowe, na którym ma funkcjonować oprogramowanie. Wybór ten zależy w głównej mierze od przeznaczenia systemu i grupy docelowej jego użytkowników. Dokonano porównania trzech rodzajów aplikacji: aplikacji desktopowej, aplikacji webowej oraz aplikacji mobilnej.

### **3.1.1.      Aplikacja desktopowa**

Aplikacja desktopowa tworzona jest dla użytkowników laptopów i komputerów stacjonarnych. Wymaga zainstalowania na urządzeniu i jest zależna od systemu operacyjnego i parametrów sprzętowych urządzenia. Z założenia do działania nie wymaga połączenia z Internetem, jednak w zależności od konkretnego zastosowania brak łączności może znacznie ograniczyć jej funkcjonalność. W przypadku konieczności aktualizacji takiej aplikacji, użytkownik jest zmuszony do ręcznego instalowania nowych komponentów na każdym urządzeniu, na którym chce korzystać z systemu.

### **3.1.2.      Aplikacja webowa**

Aplikacja webowa jest rodzajem aplikacji, nie wymagającej bezpośredniego instalowania na urządzeniu ponieważ komunikacja z użytkownikiem odbywa się poprzez okno przeglądarki internetowej. Aplikacja webowa umożliwia pracę z różnych urządzeń, niezależnie od lokalizacji, a jedynym jej ograniczeniem jest dostęp do Internetu. Aktualizacja aplikacji odbywa się w większości przypadków bez udziału użytkownika. Stworzenie aplikacji webowej, która sprawnie działa na małych wyświetlaczach urządzeń mobilnych i jest niezależna od rodzaju przeglądarki jest często bardzo kłopotliwe.

### **3.1.3.      Aplikacja mobilna**

Jest to rodzaj oprogramowania dedykowanego dla urządzeń przenośnych takich jak smartfon lub tablet. Aplikacje te są zależne od systemów operacyjnych (np. Android, iOS, Windows Phone) i pozwalają użytkownikowi aplikacji na pełną mobilność. Podobnie do aplikacji desktopowych do działania nie potrzebują połączenia z Internetem jednak jest ono często wymagane, aby zagwarantować pełną funkcjonalność aplikacji.

### **3.1.4. Podsumowanie wyboru**

Aplikacja desktopowa została odrzucone ze względu na ograniczoną mobilność użytkownika takiej aplikacji, a z aplikacji webowej zrezygnowano ze względu na zależność od rodzaju przeglądarki internetowej oraz konieczność posiadania połączenia internetowego w celu działania aplikacji. Stwierdzono, iż aplikacja mobilna najlepiej spełnia wymagania związane z mobilnością użytkownika aplikacji przy pracy z dronami oraz wymogiem dostępu do Internetu i dlatego ostatecznie wybrano ją jako docelowy rodzaj aplikacji klienckiej.

## **3.2. Wybór platformy**

Jak zostało to już wcześniej napisane aplikacje mobilne są zależne od systemu operacyjnego urządzenia, na którym mają funkcjonować. Stworzenie aplikacji na wszystkie dostępne platformy jest możliwe, jednak zazwyczaj bardzo kosztowne i pracochłonne. W związku z tym decydując się na stworzenie aplikacji mobilnej należy wybrać platformę docelową, na jakiej ma działać aplikacja. Do najbardziej znanych systemów mobilnych należą: Android, iOS oraz Windows Phone.

### **3.2.1. Android**

Android jest systemem operacyjnym, opartym na jądrze Linuks OS, tworzonym przez organizację Open Handset Alliance składającą się z 84 firm, na czele których stoi Google. Android jest najpopularniejszym mobilnym systemem, kontrolującym ponad 80% urządzeń na rynku. Zapewnia największą otwartość dla developerów oraz wiele narzędzi wspomagających rozwój aplikacji, co w efekcie wpływa na szeroki wybór funkcji dostępnych dla użytkowników. Za główną wadę systemu można uznać najgorszą stabilność w porównaniu do konkurentów. W celu publikacji aplikacji w sklepie Google Play należy zarejestrować konto developerskie oraz uiścić jednorazową opłatę rejestracyjną w wysokości 25\$ USD [20].

### **3.2.2. iOS**

System operacyjny Apple Inc. dla urządzeń mobilnych iPhone, iPod Touch oraz iPad, bazujący na systemie operacyjnym Max OS X 10,5 i na tym samym jądrze Darwin. iOS kontroluje blisko 16% urządzeń mobilnych. Z punktu widzenia developerskiegoową kwestią jest łatwa dostępność narzędzi SDK oraz konieczność uiszczenia rocznej opłaty członkowskiej w wysokości 99\$ USD w celu przystąpienia do Apple Developer Program, umożliwiającego publikowanie aplikacji [21]. Aplikacje przed trafieniem do App Store podlegają znacznie bardziej rygorystycznej kontroli niż ma to miejsce w przypadku systemu Google'a. iOS cechuje się dobrą stabilnością i wygodą pracy, jednak ze względu na wysokie ceny urządzeń cieszy się znacznie mniejszą popularnością od systemu Android.

### **3.2.3. Windows Phone**

System operacyjny dla platform mobilnych opracowany przez firmę Microsoft. To co go wyróżnia to płynność i niezawodność porównywalna do systemu iOS przy jednoczesnym stosunkowo niskim koszcie urządzeń z nim współpracujących. System ten stanowi jedynie około 3% rynku urządzeń mobilnych. Koszt konta developerskiego, umożliwiającego publikowanie aplikacji wynosi 19\$ USD [22].

### **3.2.4. Podsumowanie wyboru**

System iOS został odrzucony z dalszych rozważań ze względu na wysokie koszty urządzeń oraz znacznie wyższą opłatę rejestracyjną dla developerów w porównaniu z pozostałymi platformami. Windows Phone został odrzucony ze względu na bardzo mały udział w rynku urządzeń mobilnych, a co za tym idzie mniejszą grupę potencjalnych klientów. Jako platformę, na którą zostanie stworzona aplikacja wizualizująca ze względu na największą popularność na rynku urządzeń mobilnych oraz na najszerszą ofertę narzędzi programistycznych i najprężej działającą społeczność developerską wybrano system Android.

### **3.3. Wybór narzędzia**

Pisanie aplikacji na platformę Android jest możliwe z wykorzystaniem kilku narzędzi. W tym podrozdziale dokonano porównania najpopularniejszych z nich: SDK, NDK, Xamarin, oraz HTML5.

#### **3.3.1. SDK**

Android SDK (Source Development Kit) jest to zestaw narzędzi dla programistów przeznaczony do tworzenia aplikacji na platformę Android w języku Java. W jego skład wchodzą wymagane biblioteki, debugger, emulator, dokumentacja, przykładowe programy oraz samouczki. Składa się z dwóch części: SDK Tools, wymaganej do tworzenia aplikacji niezależnie od wersji Androida oraz Platform Tools, czyli narzędzi zmodyfikowanych pod kątem konkretnych wersji systemu. SDK jest modularne, dzięki czemu cechuje się bardzo łatwą instalacją i deinstalacją potrzebnych komponentów. Jest najbardziej popularnym narzędziem programistycznym, wykorzystywany do pisania aplikacji na platformę Android [19].

#### **3.3.2. NDK**

Android NDK (Native Development Kit) jest zestawem narzędzi, pozwalającym na tworzenie aplikacji na platformę Android z wykorzystaniem języka C lub C++. Aplikacje takie można uruchomić jako proces systemu Linux, bez konieczności uruchamiania ich wewnątrz maszyny wirtualnej. Wykorzystanie NDK daje programistom większy dostęp do składników systemu, jednak wymaga od niego większej wiedzy i stwarza niebezpieczeństwo uszkodzenia systemu lub urządzenia w przypadku złego wykorzystania. Najczęściej wykorzystywane jest przy programach wymagających wysokiej wydajności, takich jak aplikacje o wysokiej mocy obliczeniowej typu gry lub symulatory [19] [20].

### **3.3.3.     Xamarin**

Xamarin to narzędzie programistyczne, wspierane przez Microsoft, umożliwiające pisanie wieloplatformowych, natywnych aplikacji mobilnych na platformy Android, iOS oraz Windows Phone z wykorzystaniem języka C# i platformy .NET. Pozwala na dzielenie znacznej części kodu między aplikacjami napisanymi pod różne platformy, co znacznie zmniejsza koszt utworzenia i utrzymania wieloplatformowych systemów mobilnych. Korzystanie z pełnej funkcjonalności Xamarina jest płatne, a koszt licencji biznesowej to 999\$ rocznie. Wersja bezpłatna jest bardzo ograniczona i pozwala jedynie na tworzenie małych aplikacji, zawierających nie więcej niż 128Kb skompilowanego kodu, bez możliwości korzystania z natywnych bibliotek języków takich jak C, C++, czy Java [23].

### **3.3.4.     HTML5**

Istnieje również możliwość tworzenia aplikacji mobilnych z wykorzystaniem technologii webowych takich jak HTML5, CSS oraz Javascript. Aplikacja stworzona w ten sposób jest osadzana w natywnym kontenerze, pozwalającym na dystrybucję w AppStore, czy GooglePlay. Rozwiązanie takie cechuje się stosunkowo prostą implementacją, umożliwia tworzenie aplikacji wieloplatformowych, a raz napisany kod może zostać również wykorzystany do stworzenia zwykłej aplikacji webowej. Aplikacje napisane przy użyciu HTML5 nie posiadają bezpośredniej integracji ze sprzętem i takimi składnikami jak system plików, aparat, akcelerometr, czy system nawigacji satelitarnej. Cechuje je brak wielowątkowości, a ich wydajność jest zwykle gorsza od wydajności aplikacji natywnych. Dodatkowo tworzenie aplikacji w technologiach webowych wymaga często dodatkowego nakładu pracy na implementacje niektórych elementów powszechnie dostępnych w technologiach natywnych [24].

### **3.3.5.     Podsumowanie wyboru**

Stwierdzono, iż wykorzystanie NDK w przypadku projektowanej aplikacji, ze względu na brak konieczności dostępu do niskopoziomowych elementów systemu oraz ze względu na brak zaawansowanych silników graficznych lub obliczeniowych w aplikacji, byłoby nieefektywne. Technologia Xamarin została odrzucona ze względu na wysoki koszt licencji, a z

technologii webowych zrezygnowano ze względu na gorszą wydajność od aplikacji natywnych. Biorąc pod uwagę powyższe argumenty jako narzędzie wykorzystane do stworzenia aplikacji klienckiej wybrano Android SDK.

### **3.4. Wybór sposobu komunikacji między serwerem, a aplikacją kliencką**

Aplikacja geolokalizacyjna ma za zadanie zbieranie danych określających położenie obiektów i przesyłanie ich do serwera. Część serwerowa odpowiada za przetworzenie otrzymanych danych, ich dalszą dystrybucję do aplikacji wizualizujących oraz współpracę z bazą danych. Panel administracyjny ma za zadanie umożliwić użytkownikom z uprawnieniami administratora zarządzanie danymi trwałymi. Należy zatem określić sposób w jaki będzie się odbywała komunikacja między tymi elementami systemu. W tym podrozdziale zostaną opisane dwa najpowszechniejsze wykorzystywane sposoby komunikacji sieciowej: z wykorzystaniem protokołu HTTP oraz WebSocket. Jednak w pierwszej kolejności zostanie omówiony ogólny model komunikacji sieciowej TCP/IP.

#### **3.4.1. Model TCP/IP**

Podstawą komunikacji internetowej są obecnie dwa protokoły: IP oraz TCP. Model komunikacji sieciowej oparty na tych protokołach nazywany jest modelem TCP/IP i jest teoretycznym modelem warstwowej struktury protokołów komunikacyjnych. Możemy w nim wyróżnić cztery warstwy: warstwę aplikacji, warstwę transportową, warstwę Internetu oraz warstwę dostępu do sieci.

W celu przesłania danych przez Sieć, klient inicjuje połączenie, najczęściej z wykorzystaniem protokołu TCP z serwerem. Protokół IP, będący protokołem warstwy internetowej, spręga ich adresy IP i porty punktów końcowych oraz zapewnia nadzór nad przesyaniem danych. Jednak przesłane informacje wymagają nie tylko przetransportowania, za co odpowiada warstwa transportowa TCP, ale również zrozumienia. Dlatego ponad warstwą TCP działa dodatkowo protokół aplikacji [30].

### **3.4.2. HTTP**

Głównym protokołem warstwy aplikacji używanym współcześnie w przeglądarkach internetowych jest protokół HTTP. HTTP to skrót od Hypertext Transfer Protocol. Jest to protokół bezstanowy tzn. ani serwer, ani klient nie przechowują informacji o wcześniejszych zapytaniach między sobą i nie posiadają stanu wewnętrznego. Klient otwiera połaczenie i wysyła komunikat żądania do serwera HTTP. Serwer następnie zwraca komunikat odpowiedzi, zawierający zasób, o który został poproszony, lub kod błędu. Po wysłaniu odpowiedzi serwer zamyka połaczenie i nie przechowuje o nim żadnych informacji, co znacznie zmniejsza jego obciążenie, jednak jest kłopotliwe w sytuacjach, w których wymagane jest zapamiętanie jakiegoś stanu związanego z konkretnym klientem. Protokół ten nie pozwala na iniciowanie połaczenia z klientem ze strony serwera i co za tym idzie uniemożliwia wysyłanie wiadomości typu PUSH od serwera do klienta. Adresowanie zapytań odbywa się poprzez ujednolicony format adresowania zasobów URL (ang. Uniform Resource Location), a operacja jaką klient chce wykonać jest określana poprzez ustandaryzowane słowa kluczowe, z których najbardziej podstawowymi są :

- GET - pobieranie zasobów,
- POST - tworzenie nowych zasobów,
- PUT - aktualizowanie istniejących zasobów,
- DELETE - kasowanie istniejących zasobów

W odpowiedzi na zapytanie klienta serwer wraz z wiadomością zwrotną przesyła kod statusu zapytania. Kod statusu zapytania mówi klientowi jak interpretować odpowiedź serwera. Kody odpowiedzi są również ustandaryzowane poprzez specyfikację protokołu HTTP:

- 1xx - Wiadomość informacyjna
- 2xx - Sukces - Zapytanie zostało prawidłowo przetworzone przez serwer,
- 3xx - Przekierowanie,
- 4xx - Błąd po stronie klienta
- 5xx - Błąd po stronie serwera



**Rys. 3.1 Schemat działania komunikacji klient-serwer, z wykorzystaniem protokołu HTTP [26]**

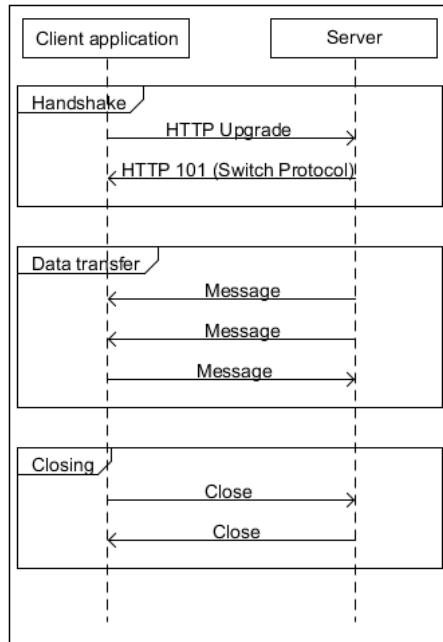
Brak możliwości wysyłania wiadomości typu PUSH ze strony serwera jest możliwy do ominienia poprzez zastosowanie techniki nazywanej "pollingiem", która polega na stałym odpytywaniu serwera przez klienta o nowe dane. Jeśli oczekiwany zasób uległ zmianie serwer wysyła go klientowi, w innym wypadku wysyła pustą odpowiedź. Wariancją tego rozwiązania jest tzw. "Long polling", który różni się od tradycyjnego pollingu tym, że po otrzymaniu zapytania serwer w przypadku braku nowych danych nie odsyła klientowi pustej wiadomości, a utrzymuje otwarte połączenie, aż do momentu dostępności żądanego zasobu. Techniki te są jednak wysoce nieefektywne, w związku z tym, że obciążają zarówno aplikację kliencką jak i część serwerową systemu niezależnie od tego, czy dany zasób jest dostępny, czy nie [29].

### 3.4.3. WebSocket

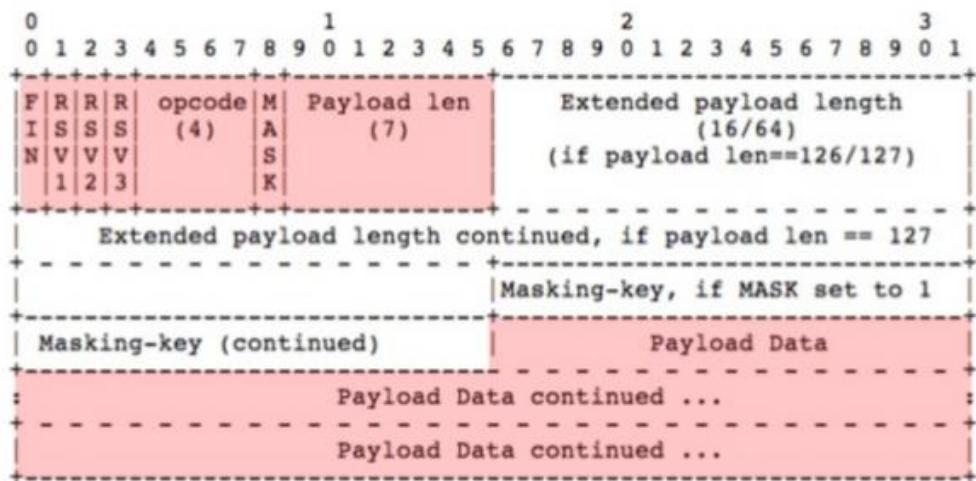
Technologia WebSocket umożliwia utworzenie trwałego, dwustronnego połączenia między klientem, a serwerem. WebSocket wykorzystuje do budowy połączenia zmianę protokołu. Klient wysyła do serwera normalne zapytanie HTTP, o specyficznym nagłówku, zawierającym klucz zabezpieczający kodowany, za pomocą algorytmu base64, który informuje serwer o chęci nawiązania połączenia poprzez protokół WebSocket. Jeśli serwer obsługuje protokół WebSocket wysyła do klienta odpowiedź, uzupełnioną o dodatkowy ciąg znaków. W ten sposób oba punkty końcowe informują się, że używają protokołu WebSocket. Jako dodatkowy mechanizm zabezpieczający, protokół wykorzystuje szyfrowanie pakietów danych [28].

Po nawiązaniu połączenia za pomocą protokołu WebSocket, w przeciwieństwie do HTTP, zarówno klient jak i serwer mogą wysyłać między sobą dane, równocześnie, przez jedno połączenie TCP. Skutkuje to zmniejszeniem opóźnienia i obciążenia Sieci. Połączenia WebSocket, podobnie jak HTTP, wykorzystują standardowe porty: 80, gdy nie są szyfrowane i 443, w przeciwnym wypadku. Websocket posiada analogiczny format adresowania do

HTTP: ws (Web Services) dla połączeń nieszyfrowanych oraz wss (Web Secure Services) dla połączeń szyfrowanych [28].



Rys. 3.2 Schemat działania połączenia, między klientem, a serwerem przy wykorzystaniu protokołu Web-Socket [29]



Rys. 3.3 Struktura ramki danych dla protokolu WebSocket [27]

Jedna wiadomość może składać się z wielu ramek, z których każda ma strukturę taką, jak przedstawiono na Rys.3.3.

- FIN - bit identyfikujący ostatnią ramkę wiadomości,
- SRV[1-3] - 0 lub informacja o rozszerzeniu wiadomości,

- OPCODE - identyfikator typu ramki: tekstowa (1), binarna (2), kontrolna np. zamykająca połączenie (8), ping (9), pong(10),
- MASK - bit określający, czy ramka jest maskowa (dla wiadomości ze strony klienta),
- Payload len - wielkość danych:
  - Jeśli 0-125 - taka jest długość ramki,
  - Jeśli 126 - kolejne 2 bajty reprezentują 16-bitową liczbę typu unsigned integer, określającą długość ramki,
  - Jeśli 127 - kolejne 8 bajtów reprezentują 64-bitową liczbę typu unsigned integer, określającą długość ramki,
- Masking-key - zawiera 32-bitową wartość, używaną do maskowania danych (jeśli MASK 1),
- Payload Data - Zawiera przesyłane dane i dane dotyczące niestandardowych rozszerzeń, takich jak: sprecyzowany format wysyłania danych, czy wymogi semantyczne narzucone przez konkretną implementację protokołu.

[25]

#### **3.4.4. Podsumowanie wyboru**

W tworzonym systemie założono wykorzystanie obu standardów. Komunikacja z wykorzystaniem protokołu HTTP, ze względu na łatwość implementacji, zostanie wykorzystana do pobierania danych trwałych z serwera, podczas gdy protokół WebSocket zostanie użyty do przesyłania danych związanych z aktualizacją położenia oraz obszaru przeszukanego. Rozróżnienie takie wynika z konieczności wysyłania wiadomości typu PUSH od serwera do klienta, zawsze wtedy, gdy zmianie ulegnie położenie drona.

### **3.5. Wybór formatu przesyłanych danych**

Zostały już omówione sposoby transportowania danych. Jednak efektywna komunikacja sieciowa wymaga jeszcze ujednolicenia formatowania przesyłanych danych, tak aby odbiorca był w stanie wykorzystać to, co otrzyma od nadawcy wiadomości. Najprostszym formatem, zrozumiałym dla człowieka jest format tekstowy, a dwoma najpopularniejszymi formatami tekstowymi, wykorzystywany w komunikacji sieciowej są XML oraz JSON.

### **3.5.1. XML**

XML (ang. Extensible Markup Language), czyli uniwersalny język znaczników, jest otwartym standardem opracowanym przez World Wide Web Consortium (W3C) umożliwiającym reprezentację danych w ustrukturyzowany sposób. XML jest językiem opisującym dane, czyli meta językiem. Został stworzony z myślą o przechowywaniu danych, jednak służy również do ich transportowania oraz do tworzenia innych języków (aplikacji XML) służących do przechowywania informacji. Jest czytelny dla człowieka, co znacznie ułatwia jego wykorzystanie [33].

```
<?xml version="1.0" encoding="utf-8"?>
<dane>
    <user>
        <imie>jan</imie>
        <nazwisko>Kowalski</nazwisko>
    </user>
    <user>
        <imie>Piotr</imie>
        <nazwisko>Nowak</nazwisko>
    </user>
</dane>
```

**Rys. 3.4 Przykład danych w XML [32]**

### **3.5.2. JSON**

JSON (ang. Javascript Object Notation) jest tekstowym formatem wymiany danych opartym o literał obiektowy. Został stworzony jako lżejsza alternatywa dla XML. Podczas gdy dane zapisane w formacie XML muszą zostać dodatkowo zinterpretowane i przekształcone na obiekty, z użyciem JSON'a dane są od razu zakodowane jako obiekt JavaScriptowy, łatwo rozumiany również przez inne języki programowania. JSON jest mniej czytelny dla człowieka, jednak ze względu na brak znaczników jest bardziej wydajnym standardem jeśli chodzi o transportowanie oraz serializację i deserializację danych [32] [33].

```

{
  "dane" : {
    "user" : [
      {
        "imie" : "Jan",
        "nazwisko" : "Kowalski"
      },
      {
        "imie" : "Piotr",
        "nazwisko" : "Nowak"
      }
    ]
  }
}

```

**Rys. 3.5 Przykład danych w JSON [32]**

### 3.5.3. Podsumowanie wyboru

Ze względu na wyższą wydajność oraz łatwość serializacji i deserializacji jako format przesyłanych danych wybrano format JSON.

## 3.6. Opis API OpenStreetMap

OpenStreetMap (OSM) to globalny projekt, mający na celu stworzenie darmowej oraz swobodnie dostępnej mapy świata. Wszelkie dane zawarte w projekcie udostępniane są na otwartej licencji Open Database License (ODbL). Umożliwia to ich wykorzystanie praktycznie przez każdego, przez co OpenStreetMap jest określone mapowym odpowiednikiem Wikipedii i jest darmową alternatywą dla map komercyjnych [34].

Oficjalne API OpenStreetMap różni się przykładowo od API GoogleMaps tym, że udostępnia jedynie usługi do odczytu i zapisu danych w surowej formie wektorowej, podczas gdy GoogleMaps udostępnia użytkownikom konkretne usługi do rysowania i edycji map. Jednak w związku z otwartością projektu OSM istnieje wiele zewnętrznych bibliotek, które sumarycznie udostępniają praktycznie każdą funkcję, którą można znaleźć w mapach komercyjnych. Ponadto dostępność do surowych danych w postaci pojedynczego pliku, w formacie binarnym PBF lub skompresowanym XML, zawierającego wszystkie punkty, drogi i relacje, tworzące mapę pozwala na dowolną implementację własnych usług. Dane te są aktualizowane co tydzień i możliwe do pobrania również w mniejszych paczkach dla konkretnych kontynentów, czy państw [35].

Mapy OpenStreetMap, podobnie jak mapy GoogleMaps generowane są w procesie nazywanym renderingiem, który konwertuje dane wektorowe na dane typu rastrowego. Na dane te zostają następnie nałożone dodatkowe style wizualne, a mapy zostają podzielone na "kafelki" i w tej formie udostępnione aplikacjom webowym i mobilnym przez zewnętrznych usługodawców. Użytkownicy OSM mają również możliwość zaimplementowania własnego sposobu renderowania map, co znacznie rozszerza możliwości programistyczne w porównaniu do GoogleMaps [35].

W przypadku tworzenia oprogramowania na platformę Android, wykorzystującego dane OpenStreetMap, istnieje możliwość wykorzystania zewnętrznych bibliotek, zapewniających podstawowe funkcje związane z rysowaniem map, ich edycją, malowaniem śladów, znaczników itd. Do najbardziej znanych bibliotek, o takiej funkcjonalności, należą Mapsforge oraz Osmdroid. Uznano, iż obie biblioteki spełniają wymagania dla tworzonej aplikacji i ostatecznie postanowiono wykorzystać bibliotekę Osmdroid.

Na tym zostaną na ten moment zakończone rozważania związane z technologiami wykorzystanymi przy tworzeniu systemu. Oczywiście nie są to wszystkie aspekty technologiczne związane z projektowanym systemem. W rzeczywistości należałoby dokonać analizy rozwiązań dla wszystkich komponentów systemu np. sposobu realizacji części serwerowej, sposobu przechowywania danych trwałych, czy wyboru silnika bazodanowego. Jednak nie jest to praca o tematyce stricte informatycznej, a prezentowany system jest jedynie jej częścią dlatego w powyższym rozdziale ograniczono się do kwestii uznanych za najważniejsze z punktu widzenia integracji z API OpenStreetMap oraz wizualizacji obszaru przeszukanego.Więcej na temat realizacji poszczególnych części systemu zostanie napisane w rozdziale "Projekt systemu".

## **4. Analiza systemu**

Tworzenie systemu informatycznego jest procesem złożonym, dlatego w celu jego uproszczenia dokonuje się podziału całego procesu na mniejsze części. Inżynieria oprogramowania stara się zidentyfikować i opisać podstawowe fazy tworzenia i funkcjonowania oprogramowania. Istnieje wiele różnych sposobów podziału na te fazy, lecz wszystkie prowadzą się mniej więcej do jednego przybliżenia, zgodnie z którym można wyróżnić podstawowe czynności związane z tworzeniem systemu oprogramowania [36] :

- Analiza - określanie wymagań i specyfikacji,
- Projektowanie,
- Implementacja,
- Testowanie,
- Konserwacja,

W tym rozdziale zostanie dokonana analiza systemu, na którą składają się te fazy rozwoju projektu, które głównie koncentrują się na problemie biznesowym, niezależnie od jakiejkolwiek technologii, która mogłaby zostać użyta do zaimplementowania rozwiązania tego problemu. Podczas analizy ustala się co system ma robić, aby zaspokoić wymagania użytkownika, a jej wynikiem jest logiczny model systemu, opisujący sposób realizacji przez system postawionych wymagań, lecz abstrahujący od szczegółów implementacyjnych [37]. W pierwszym podrozdziale przedstawiono słowny opis systemu, skupiający się na głównych funkcjach systemu.

### **4.1. Opis systemu**

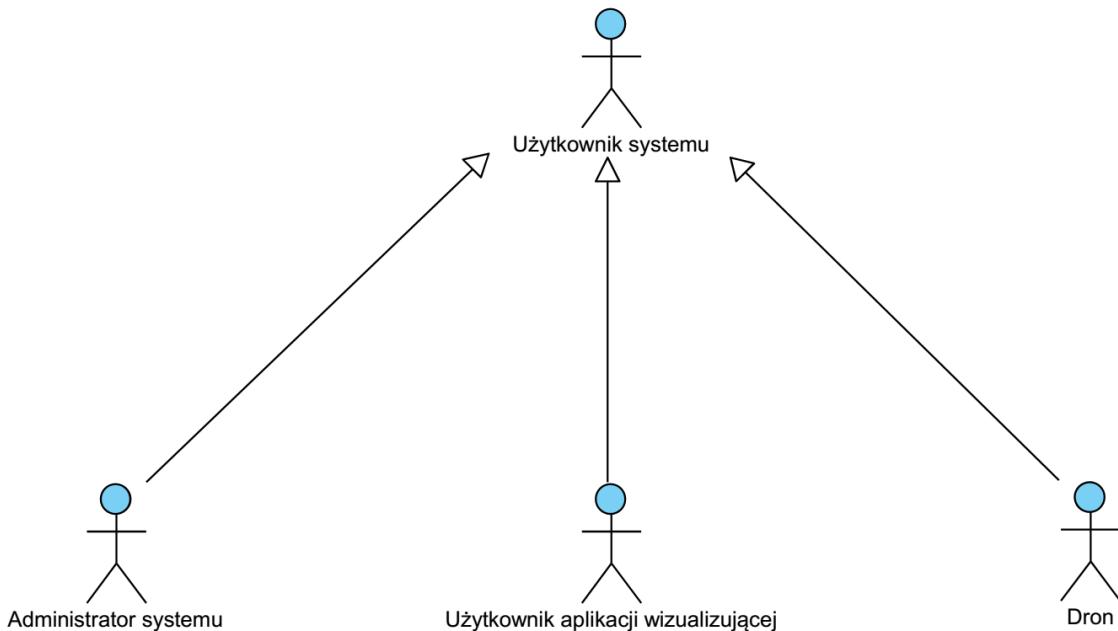
System został nazwany "DronVision". Ma on za zadanie służyć do nadzoru i kontroli dużych obszarów terytorialnych, takich jak obszary przemysłowe lub rolnicze, poprzez wizualizację obszaru przeszukanego przez bezzałogowe statki latające.

Można wyróżnić kilka głównych funkcji jakie ma realizować projektowany system. Ma umożliwiać zbieranie danych geolokalizacyjnych dronów w czasie rzeczywistym, analizowanie tych danych zgodnie z opracowanym algorytmem oraz ich wizualizację na urządzeniach mobilnych.

Wymagane jest, aby wizualizacja była możliwa do przeprowadzenia dla kilku dronów jednocześnie, z możliwością wybrania przez użytkownika drona za którym widok mapy ma podążać, dronów pokazanych na mapie, jedynie jako znaczniki określające położenie pojazdów oraz tych, dla których ma być realizowana pełna wizualizacja. Dodatkowo system ma zapewniać możliwość przeprowadzenia symulacji, demonstrującej działanie aplikacji oraz przeglądanie historii zarchiwizowanych wizualizacji. Ponadto należy udostępnić panel administracyjny, umożliwiający zarządzanie użytkownikami systemu, pojazdami oraz uprawnieniami użytkowników do wizualizacji konkretnych dronów.

## 4.2. Identyfikacja aktorów

Znając główne zadania, jakie ma realizować system można przejść do identyfikacji aktorów, czyli wszystkich zewnętrznych bytów (użytkowników lub zewnętrznych systemów), z którymi projektowany system wchodzi w interakcję. Rozpoznani dla analizowanego systemu aktorzy zostali przedstawieni na poniższym diagramie (Rys. 4.1).



Rys. 4.1 Identyfikacja aktorów systemu

Aby umożliwić lepsze zrozumienie powyższego podziału dokonano opisu każdego z bytów.

#### **4.2.1. Użytkownik systemu**

Użytkownik systemu jest pojęciem nieostrym i dotyczy każdego bytu wchodzącego w interakcję z systemem.

#### **4.2.2. Administrator systemu**

Administratorem jest osoba posiadająca specjalne uprawnienia względem funkcji systemu. Administrator ma możliwość zarządzania użytkownikami systemu, zarządzania pojazdami w systemie, nadawania użytkownikom uprawnień do korzystania z aplikacji względem konkretnych dronów oraz zarządzania symulacjami.

#### **4.2.3. Użytkownik aplikacji wizualizującej**

Użytkownik aplikacji wizualizującej to użytkownik korzystający z aplikacji klienckiej. Jest to osoba zarejestrowana w systemie oraz posiadająca konto aktywowane przez administratora.

#### **4.2.4. Dron**

Bezzałogowy statek latający, który posiada zamontowany geolokalizator z oprogramowaniem, umożliwiającym komunikację z systemem.

### **4.3. Przypadki użycia**

Gdy wiadomo jakie są główne funkcje systemu oraz kto będzie z niego korzystał można przystąpić do określania wymagań funkcjonalnych. Można tego dokonać poprzez opracowanie diagramów przypadków użycia i scenariuszy przypadków użycia, przedstawiających interakcje między systemem i jego użytkownikami.

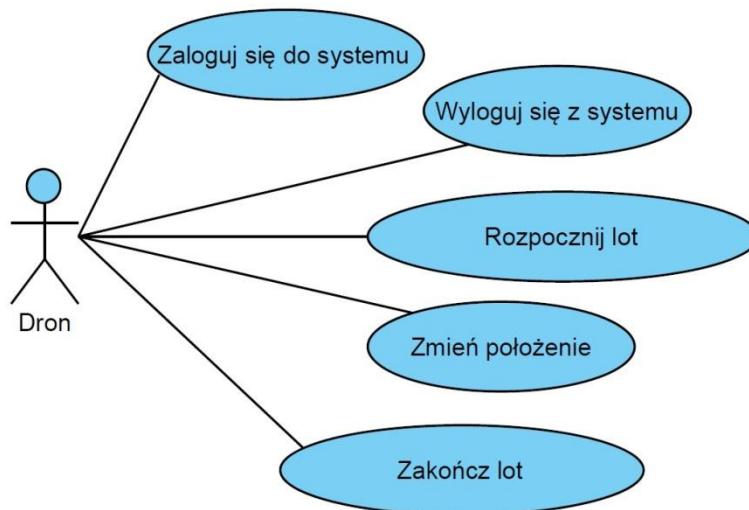
Dla każdego z zidentyfikowanych aktorów stworzono diagram przypadków użycia, przedstawiający wszystkie możliwe interakcje między nim, a systemem, a przypadki uznane za najistotniejsze dodatkowo zostały opisane w postaci scenariuszy przypadków użycia.

#### 4.3.1. Przypadki użycia - administrator systemu



Rys. 4.2 Przypadki użycia - administrator systemu

#### 4.3.2. Przypadki użycia - dron

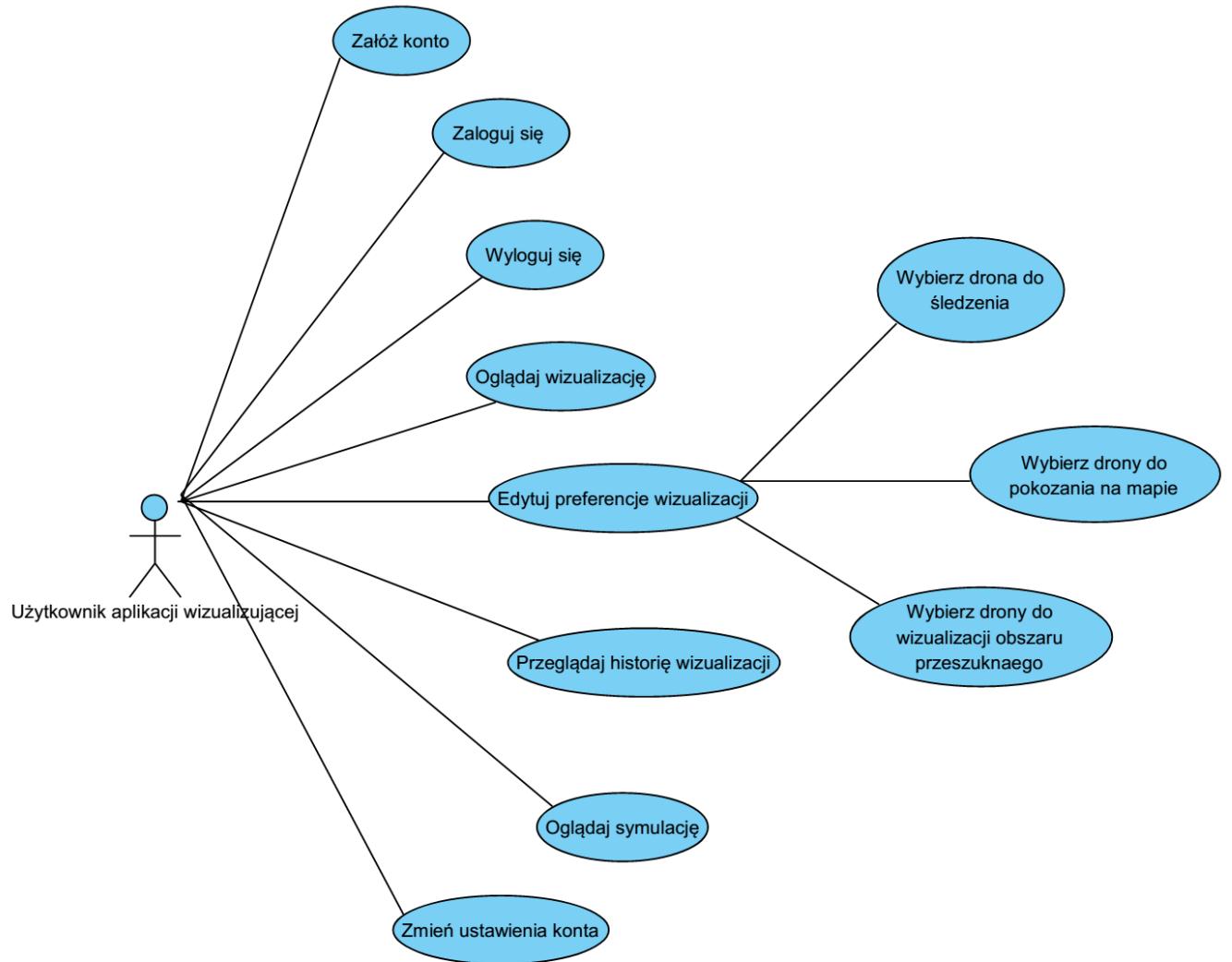


Rys. 4.3 Przypadki użycia - dron

##### 4.3.2.1. Scenariusz przypadku użycia - Zmień położenie

<b>Nazwa przypadku użycia</b>	Zmień położenie
<b>Aktorzy uczestniczący</b>	<ul style="list-style-type: none"> <li>Dron</li> </ul>
<b>Przepływ zdarzeń</b>	<ol style="list-style-type: none"> <li>System rejestruje zmianę położenia drona</li> <li>Dron wysyła swoje nowe położenie</li> <li>System analizuje nowe dane geolokalizacyjne drona</li> </ol>
<b>Warunki wstępne</b>	<ul style="list-style-type: none"> <li>Aplikacja geolokalizująca drona została aktywowana przez administratora</li> </ul>
<b>Warunki końcowe</b>	<ul style="list-style-type: none"> <li>Dane dotyczącej nowego położenia drona zostały zarejestrowane w systemie</li> </ul>
<b>Wymagania jakościowe</b>	<ul style="list-style-type: none"> <li>System powinien rejestrować zmiany położenia drona z dokładnością do 2m</li> </ul>

### 4.3.3. Przypadki użycia - użytkownik aplikacji wizualizującej



Rys. 4.4 Przypadki użycia - użytkownik aplikacji wizualizującej

#### 4.3.3.1. Scenariusz przypadku użycia - Załóż konto

<b>Nazwa przypadku użycia</b>	Zarejestruj się w systemie
<b>Aktorzy uczestniczący</b>	<ul style="list-style-type: none"> <li>• Użytkownik aplikacji wizualizującej</li> <li>• Administrator systemu</li> </ul>
<b>Przepływ zdarzeń</b>	<p>1. Użytkownik aplikacji wizualizującej aktywuje funkcję "Zarejestruj się w systemie" w swoim panelu aplikacji.</p> <p>2. System wyświetla w odpowiedzi stosowny formularz</p> <p>3. Użytkownik wypełnia formularz uzupełniając wszystkie wymagane dane i wysyła wypełniony formularz do systemu.</p> <p>4. System informuje użytkownika o przyjęciu zgłoszenia rejestracji i informuje, że konto oczekuje na zatwierdzenie przez administratora.</p> <p>5. System informuje administratora o nowym wniosku rejestracyjnym.</p> <p>6. Administrator systemu analizuje informacje zawarte w formularzu po czym zatwierdza lub odrzuca wniosek.</p> <p>7. System informuje użytkownika aplikacji wizualizującej mailowo o powodzeniu lub niepowodzeniu rejestracji.</p>
<b>Warunki wstępne</b>	-
<b>Warunki końcowe</b>	<ul style="list-style-type: none"> <li>• Użytkownik aplikacji wizualizującej otrzymał potwierdzenie rejestracji lub</li> <li>• Użytkownik aplikacji wizualizującej otrzymał informację o niepowodzeniu rejestracji</li> </ul>
<b>Wymagania jakościowe</b>	-

#### 4.3.3.2. Scenariusz przypadku użycia - Zaloguj się

<b>Nazwa przypadku użycia</b>	Zaloguj się w systemie
<b>Aktorzy uczestniczący</b>	<ul style="list-style-type: none"> <li>• Użytkownik aplikacji wizualizującej</li> </ul>
<b>Przepływ zdarzeń</b>	<p>1. Użytkownik aplikacji wizualizującej wprowadza w wyznaczone miejsca dane uwierzytelniające w postaci loginu i hasła.</p> <p>2. System sprawdza poprawność wprowadzonych danych oraz czy dane konto jest aktywne. W przypadku powodzenia operacji przenosi użytkownika do głównego widoku. W przypadku niepowodzenia informuje o tym użytkownika poprzez odpowiedni komunikat.</p>
<b>Warunki wstępne</b>	<ul style="list-style-type: none"> <li>• Użytkownik aplikacji wizualizującej jest zarejestrowany w systemie</li> <li>• Konto użytkownika zostało aktywowane przez administratora systemu</li> </ul>
<b>Warunki końcowe</b>	<ul style="list-style-type: none"> <li>• Użytkownik aplikacji wizualizującej został uwierzytelniony i przeniesiony do widoku głównego aplikacji lub</li> <li>• Użytkownik aplikacji wizualizującej otrzymał informację o niepowodzeniu operacji logowania.</li> </ul>
<b>Wymagania jakościowe</b>	<ul style="list-style-type: none"> <li>• Komunikat o niepowodzeniu operacji logowania zawiera dane dotyczące powodu niepowodzenia: złe dane uwierzytelniające lub konto nieaktywne.</li> </ul>

#### **4.3.3.3. Scenariusz przypadku użycia - Oglądam wizualizację**

<b>Nazwa przypadku użycia</b>	Oglądam wizualizację
<b>Aktorzy uczestniczący</b>	<ul style="list-style-type: none"><li>• Użytkownik aplikacji wizualizującej</li></ul>
<b>Przepływ zdarzeń</b>	<ol style="list-style-type: none"><li>1. Użytkownik aplikacji wizualizującej wybiera z menu opcję wizualizacja.</li><li>2. System wyświetla na ekranie widok wizualizacji</li></ol>
<b>Warunki wstępne</b>	<ul style="list-style-type: none"><li>• Użytkownik aplikacji wizualizującej jest zalogowany do systemu</li></ul>
<b>Warunki końcowe</b>	<ul style="list-style-type: none"><li>• Użytkownik aplikacji wizualizującej został przeniesiony do widoku wizualizacji</li></ul>
<b>Wymagania jakościowe</b>	

#### 4.3.3.4. Scenariusz przypadku użycia - Edytuj preferencje wizualizacji

<b>Nazwa przypadku użycia</b>	Edytuj preferencje wizualizacji
<b>Aktorzy uczestniczący</b>	<ul style="list-style-type: none"> <li>• Użytkownik aplikacji wizualizującej</li> </ul>
<b>Przepływ zdarzeń</b>	<p>1. Użytkownik aplikacji wizualizującej wybiera z menu opcję "Preferencje wizualizacji".</p> <p>2. System wyświetla widok "Preferencje wizualizacji" z zapamiętanymi uprzednio ustawieniami</p> <p>3. Użytkownik zmienia preferencje i zapisuje zmiany zamknięcie widoku "Preferencje wizualizacji"</p> <p>4. System sprawdza, czy dotychczasowe preferencje uległy zmianie i jeśli tak zapisuje zmiany.</p> <p>5. System zamyka widok "Preferencje wizualizacji"</p>
<b>Warunki wstępne</b>	<ul style="list-style-type: none"> <li>• Użytkownik aplikacji wizualizującej jest zalogowany do systemu</li> </ul>
<b>Warunki końcowe</b>	<ul style="list-style-type: none"> <li>• Wprowadzone przez użytkownika zmiany dot. preferencji wizualizacji zostały zapisane w systemie</li> </ul>
<b>Wymagania jakościowe</b>	<ul style="list-style-type: none"> <li>• System powinien umożliwić zmianę preferencji dotyczących: <ul style="list-style-type: none"> <li>- śledzonego drona,</li> <li>- dronów pokazanych na mapie,</li> <li>- dronów do wizualizacji obszaru przeszukanego.</li> </ul> </li> </ul>

#### 4.3.3.5. Scenariusz przypadku użycia - Oglądaj symulację

<b>Nazwa przypadku użycia</b>	Oglądaj symulację
<b>Aktorzy uczestniczący</b>	<ul style="list-style-type: none"> <li>• Użytkownik aplikacji wizualizującej</li> </ul>
<b>Przepływ zdarzeń</b>	<ol style="list-style-type: none"> <li>1. Użytkownik aplikacji wizualizującej wybiera z menu opcję "Symulacja".</li> <li>2. System wyświetla widok "Symulacja"</li> <li>3. Użytkownik uruchamia symulację</li> <li>4. System rozpoczyna symulację</li> <li>5. Po zakończeniu symulacji użytkownik wyłącza tryb symulacji</li> <li>5. System zamyka widok "Symulacja"</li> </ol>
<b>Warunki wstępne</b>	<ul style="list-style-type: none"> <li>• Użytkownik aplikacji wizualizującej jest zalogowany do systemu</li> </ul>
<b>Warunki końcowe</b>	
<b>Wymagania jakościowe</b>	<ul style="list-style-type: none"> <li>• Użytkownik powinien dodatkowo mieć możliwość przerwania i wznowienia symulacji w dowolnym momencie oraz jej zrestartowania po zakończeniu</li> </ul>

#### 4.3.3.6. Scenariusz przypadku użycia - Przeglądaj historię wizualizacji

<b>Nazwa przypadku użycia</b>	Przeglądaj historię wizualizacji
<b>Aktorzy uczestniczący</b>	<ul style="list-style-type: none"> <li>• Użytkownik aplikacji wizualizującej</li> </ul>
<b>Przepływ zdarzeń</b>	<p>1. Użytkownik aplikacji wizualizującej wybiera z menu opcję "Historia wizualizacji".</p> <p>2. System wyświetla widok "Historia wizualizacji" z listą dostępnych dronów</p> <p>3. Użytkownik wybiera drona, dla którego chce przejrzeć исторię</p> <p>4. System wyświetla listę dostępnych sesji</p> <p>5. System zamyka widok "Historia wizualizacji"</p> <p>6. System pokazuje mapę wraz zaznaczonym obszarem przeszukanym dla danej sesji</p> <p>5. Po zakończeniu przeglądania historii wizualizacji użytkownik wyłącza widok historii.</p>
<b>Warunki wstępne</b>	- Użytkownik aplikacji wizualizującej jest zalogowany do systemu
<b>Warunki końcowe</b>	
<b>Wymagania jakościowe</b>	<ul style="list-style-type: none"> <li>• Lista dostępnych sesji powinna zawierać informacje o tym kiedy sesja została rozpoczęta i kiedy się zakończyła</li> </ul>

## **5. Algorytm obliczania obszaru przeszukanego**

Przed przejściem do projektowania systemu zostanie omówiona kolejna kluczowa kwestia, czyli algorytm, służący do obliczaniu obszaru przeszukanego przez drony.

Obszar przeszukany jest rozumiany jako reprezentacja terenu zarejestrowanego przez kamerę zamontowaną na dronie. Jego poprawne obliczenie jest kluczowe z punktu widzenia funkcjonowania systemu, biorąc pod uwagę, iż jego podstawowym zadaniem jest właśnie wizualizacja tego obszaru.

### **5.1. Założenia**

W pracy przyjęto pewne założenia dotyczące wymagań dla opracowywanego algorytmu:

- Algorytm za dane wejściowe ma przyjmować: położenie geograficzne drona oraz kąt widzenia kamery zamontowanej na dronie,
- W pracy przyjęto, iż kamera zamontowana na dronie jest skierowana pionowo w dół,
- Algorytm ma obliczać zarejestrowany obszar, przy wykorzystaniu danych modelujących powierzchnię ziemską,
- Przyjęto dodatkowe uproszczenie zakładające, iż kamera rejestruje obraz o powierzchni kołowej, podczas, gdy w rzeczywistości matryce kamer mają kształt prostokątny.

## 5.2. Implementacja

### 5.2.1. Dane wejściowe

Do pobierania danych geolokalizacyjnych wykorzystano system nawigacji satelitarnej GPS. Zatem dane otrzymane z odbiornika, jak zostało to opisane w rozdziale "Dziedzina problemu" to szerokość i długość geograficzna odniesione do elipsoidy WGS84 oraz wysokość elipsoidalna.

Jako dane modelowe wykorzystano dane z misji SRTM, które są w postaci szerokości i długości geograficznej odniesionych do elipsoidy WGS84 oraz wysokości ortometrycznej odniesionej do geoidy EGM96.

Współrzędne określające szerokość i długość geograficzną z danych SRTM oraz z systemu GPS są zatem odniesione do tego samego układu odniesienia, jednak układy odniesienia dla wysokości są różne i w celu ich porównania należy wprowadzić odpowiednie poprawki.

W tym celu przeprowadzono eksperyment, którego głównym założeniem było wyznaczenie wartości średniej undulacji dla lokalnego terenu, na którym był testowany system. Dokonano porównania danych wysokościowych pobranych z GPS i tych z modelu SRTM (dla około 500 próbek), a następnie obliczono średnią, maksymalną i minimalną wartość undulacji. Wyniki obliczeń przedstawiono w tabeli.

<b>Średnia undulacja</b>	6,6m
<b>Maksymalna undulacja dodatnia</b>	18,2m
<b>Maksymalna undulacja ujemna</b>	-17,2m

**Tabela 1 Wartości undulacji przed wprowadzeniem korekty**

Na podstawie tych danych wywnioskowano, iż średnio wysokości z misji SRTM są większe od wysokości GPS o 6,6m, przy czym maksymalny błąd dodatni to 18,2m, a maksymalny błąd ujemny to -17,2m.

W celu poprawy tych parametrów do danych GPS'owych wprowadzono korektę poprzez dodanie do wartości wysokości GPS wartość średniej undulacji. Wyniki po korekcji przedstawiono w tabeli.

<b>Średnia undulacja</b>	-0,01465m
<b>Maksymalna undulacja dodatnia</b>	11,6m
<b>Maksymalna undulacja ujemne</b>	-23,8m

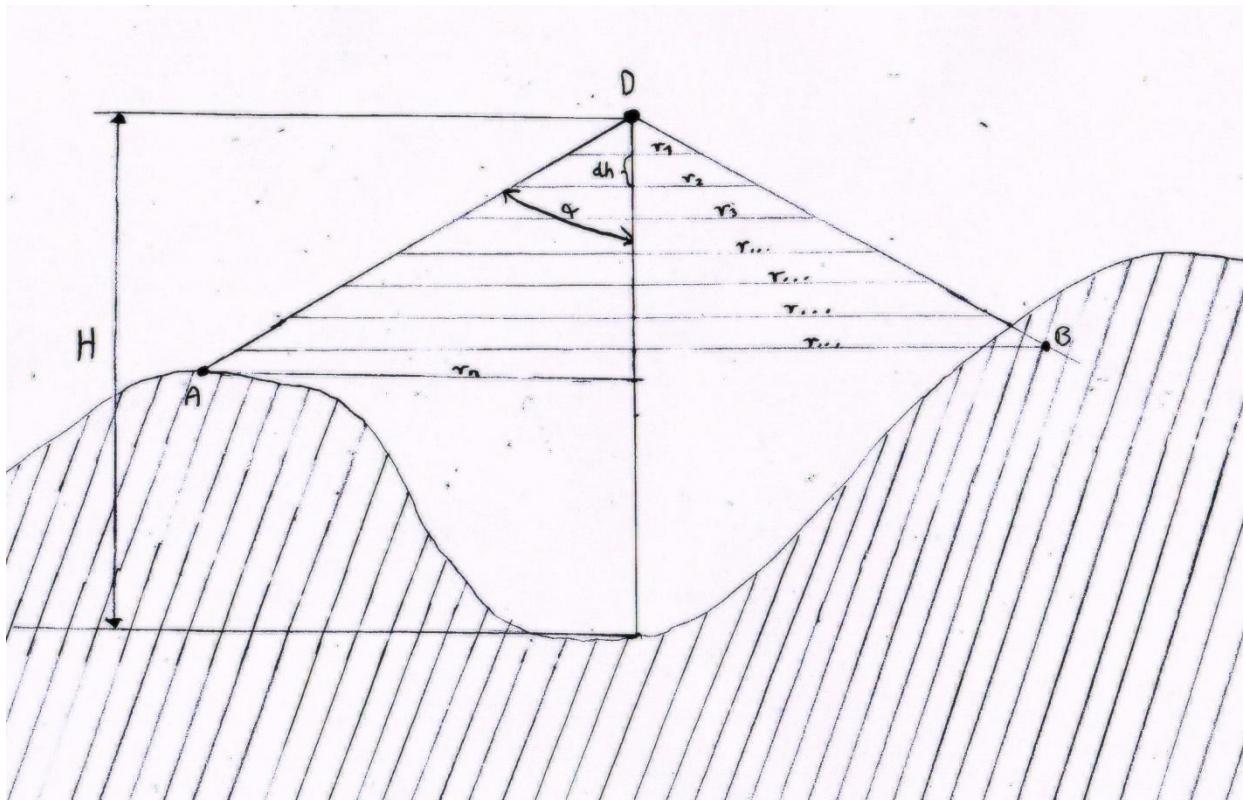
**Tabela 2 Wartości undulacji po wprowadzeniu korekty**

Jak widać średnia wartość undulacji została zredukowana do bardzo małej wartości. Wartość maksymalna undulacji dodatniej zmalała do 11,6m, podczas gdy wartość minimalnej undulacji ujemnej wzrosła do -23,8m.

Zaprezentowany powyżej proces jest jedynie przykładem rozwiązania problemu porównywania wysokości odniesionych do geoidy i do elipsoidy. Rzeczywisty wpływ korekcji na pracę systemu nie został zauważony ze względu na testowanie systemu na terenach nizinnych, gdzie uwzględnienie undulacji miało jedynie niewielki wpływ na promień okręgu obszaru przeszukanego. W przypadku implementacji systemu na rzeczywistym obszarze, np. konkretnym obszarze przemysłowym, należałoby przeprowadzić dokładne badania terenu i stworzyć tabelę undulacji dla konkretnych współrzędnych danego terenu.

### **5.2.2. Algorytm wyznaczania otoczki obszaru przeszukanego**

Pełen algorytm został podzielony na trzy części. Algorytm podstawowy służący do wyznaczenia otoczki obszaru zarejestrowanego, algorytm dodatkowy służący do wyznaczenia obszarów wewnętrz otoczki, których kamera nie rejestruje ze względu na zasłaniające obiekty oraz algorytm służący do łączenia kolejno wyznaczonych obszarów w całość. Na rys. 6.1 przedstawiono zasadę działania pierwszej części algorytmu.



Rys. 5.1 Rysunek przedstawiający zasadę działania algorytmu wyznaczania otoczki obszaru przeszukiwanego

Punkt **D** odpowiada położeniu drona. Wysokość **H** jest wysokością statku nad ziemią, wyliczoną poprzez odjęcie od wysokości drona wysokości modelu powierzchni ziemskiej w danym punkcie. Znając położenie i wysokość drona nad ziemią oraz kąt widzenia kamery **α** rozpoczęto wyznaczanie kolejnych okręgów reprezentujących potencjalną otoczkę zarejestrowanego obszaru. Wyznaczanie okręgów rozpoczęto od wysokości drona pomniejszonej o skok **dh**, którego wartość na drodze eksperymentalnej dobrano na 2 m. Znając wartość **dh** oraz kąt **α** widzenia kamery wyznaczono wartość promienia **r**.

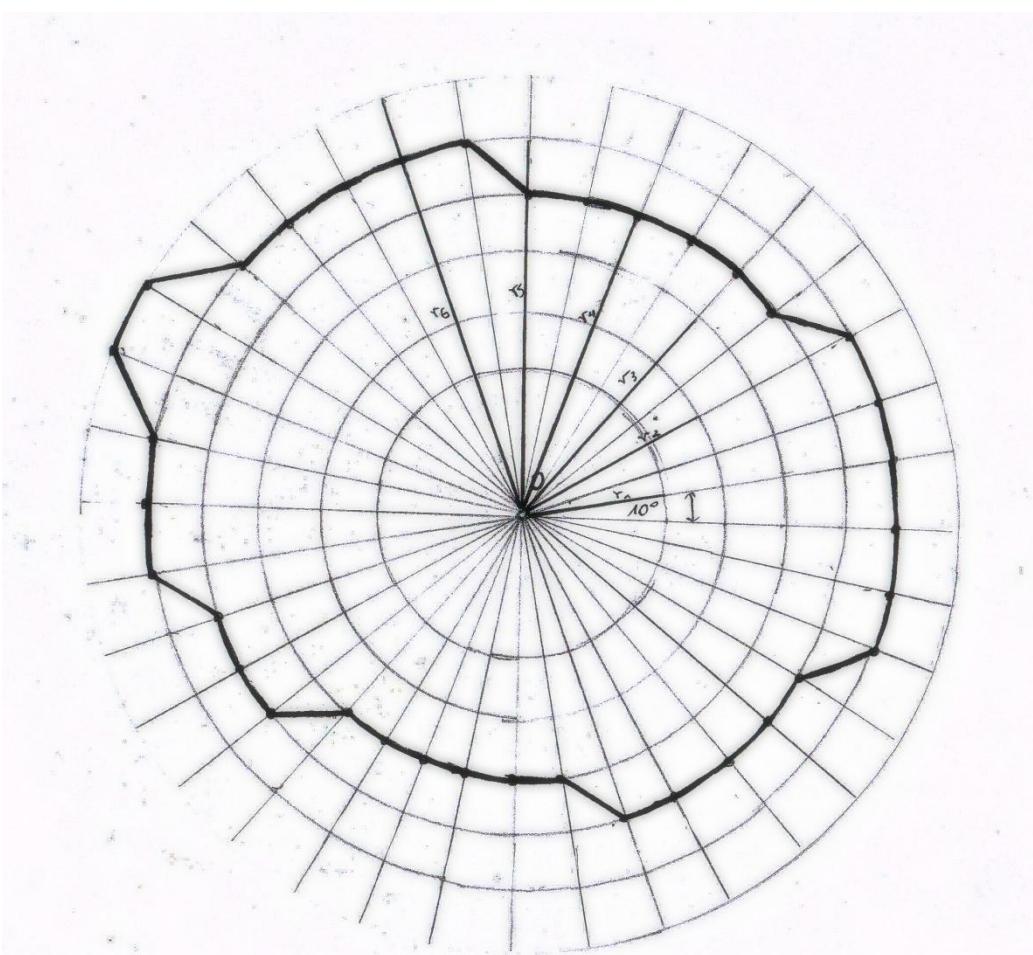
$$r = dh \cdot \operatorname{tg}(\alpha) \quad (1)$$

W dalszym kroku znając współrzędne drona i wartość promienia **r** wyznaczono punkty na okręgu o promieniu **r** i środku w punkcie odpowiadającym położeniu drona. Liczba wyznaczonych punktów, dobrana na drodze eksperymentalnej, to 360 punktów rozmieszczonych na okręgu co  $1^\circ$ .

Dla każdego z dobranych punktów dokonano następnie porównania jego wysokości (równej wysokości drona pomniejszonej o **dh**) z wysokością modelu Ziemi w danym punkcie.

Jeśli wysokość punktu na okręgu była mniejsza lub równa wysokości modelu punkt zostawał uznany za punkt otoczki obszaru przeszukanego, a wartość stopnia na okręgu, odpowiadająca temu punktowi zostawała usunięta z dalszych rozważań. Następnie wartość **dh** zostawała zwiększoną, a algorytm ten był powtarzany, aż do momentu, gdy dla każdej z 360 wartości stopni na okręgu znaleziono odpowiadający punkt modelu.

Na Rys. 6.2 przedstawiono przykładowy wynik działania algorytmu. Fragmenty okręgów o mniejszym promieniu reprezentują tereny o większej wysokości, a te o większym promieniu o mniejszej. W ten sposób można łatwo odczytać gdzie wystąpił spadek, a gdzie wzniesienie terenu i którą część terenu kamera zarejestrowała, a która została zasłonięta.

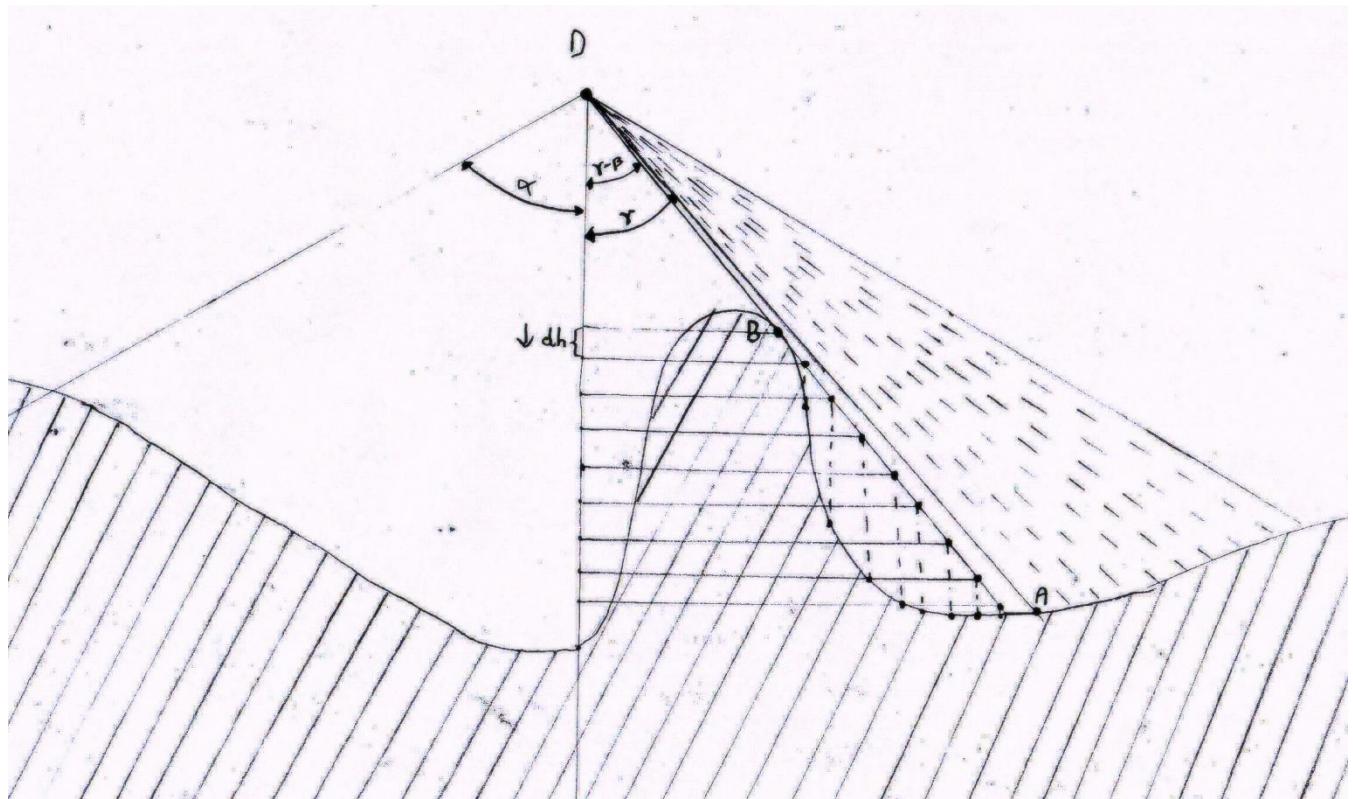


Rys. 5.2 Przykładowy wynik działania algorytmu wyznaczania otoczki obszaru przeszukanego

### **5.2.3. Algotytm wyznaczania obszarów wewnątrz otoczki, niezarejestrowanych przez kamerę**

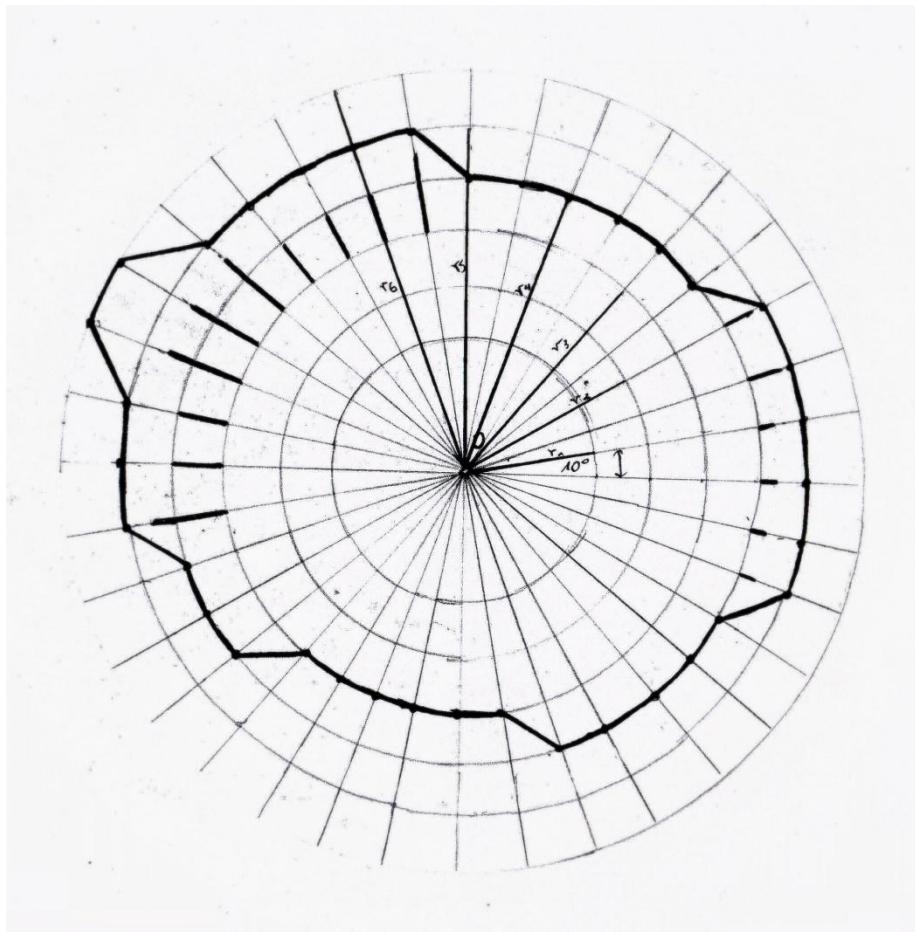
Druga część algorytmu jest rozwinięciem algorytmu podstawowego. Po wyznaczeniu otoczki obszaru przeszukanego dla kąta widzenia kamery  $\alpha$ , powtórzono tę samą operację dla kąta kamery zmniejszonego o określzoną stałą wartość  $\beta$ , jednocześnie zapamiętując wyznaczoną otoczkę dla poprzedniej wartości kąta kamery. Iterację tę powtarzano, aż do osiągnięcia kąta zerowego. W efekcie po każdej iteracji otrzymywano dwie otoczki: jedną dla kąta widzenia kamery  $\gamma$  (kąt kamery z poprzedniej iteracji) oraz drugą dla kąta  $\gamma - \beta$ .

Dla obu otoczek porównano wysokości odpowiednich punktów, leżących na odpowiadających promieniach, czyli leżących na tej samej współrzędnej kątowej okręgu. Jeśli wysokość punktu **B** (leżącego na otoczce " $\gamma - \beta$ ") jest większa od wysokości punktu **A** (leżącego na otoczce " $\gamma$ ") oznacza to, że między tymi dwoma punktami występuje wznieśenie. Wówczas zaczynając od wysokości punktu **B** powtórzono operację wykorzystaną przy wyznaczaniu otoczki, czyli stopniowe zmniejszanie wysokości o stały skok **dh**, aż do wysokości punktu **A**. Dla każdego "schodka" dokonano porównania wysokości punktu na promieniu wodzącym kamery o kącie  $\gamma - \beta$  z wysokością modelu punktu o tej samej długości i szerokości geograficznej. Jeśli wysokość punktu na promieniu wodzącym była większa od wysokości punktu modelowego zostawał on uznany za niewidoczny.



Rys. 5.3 Rysunek przedstawiający zasadę działania algorytmu wyznaczania obszarów wewnętrz otoczki, niezarejestrowanych przez kamerę

Po osiągnięciu wysokości punktu A z listy niewidocznych punktów wybrano punkt pierwszy i ostatni, czyli dwa punkty definiujące linię, będącą reprezentacją niewidocznego obszaru. Operacja ta była powtarzana dla każdego z 360 kątów, dla których wyznaczano punkty otoczki. W ten sposób otrzymano listę dziur w obszarze przeszukanym, w postaci linii, które sumarycznie reprezentują powierzchnię obszarów niezarejestrowanych przez kamerę. Koncepcyjny wynik działania algorytmu przedstawiono na rys. 6.4.



**Rys. 5.4 Przykładowy wynik algorytmu wyznaczania obszarów wewnętrz otoczki niezarejestrowanych przez kamerę**

Grubsze linie, poprowadzone wzdłuż promieni reprezentują obszary niezarejestrowane przez kamerę. Odpowiednie dobranie skoku kątowego, przy wyznaczaniu punktów okręgu oraz pogrupowanie punktów reprezentujących sąsiednie linie pozwoli na wyznaczenie zbiorów punktów reprezentujących poszczególne dziury. Następnie dla każdego zbioru zostanie wyznaczony wielokąt, zawierający wszystkie punkty zbioru zgodnie z algorytmem wyznaczania otoczki  $\alpha$ -wklęslej opisany w kolejnym podrozdziale.

## **5.2.4. Algotrithm łączenia obszarów**

W zaprojektowanym systemie istnieje podział obszaru przeszukanego na obszar ostatnio przeszukany i obszar dotychczas przeszukany. Obszar ostatnio przeszukany to obszar zarejestrowany przez kamerę dla ostatniego znanego położenia drona, podczas gdy obszar dotychczas przeszukany jest sumą wszystkich obszarów zarejestrowanych do tej pory, nie wliczając w to obszaru ostatnio przeszukanego. Rozdział ten został poświęcony opisowi algorytmu wykorzystanego do wyznaczania obszaru dotychczas przeszukanego. W pierwszej kolejności przedstawiono definicję problemu, jaki algorytm ma rozwiązać, a następnie opisano wykorzystany algorytm.

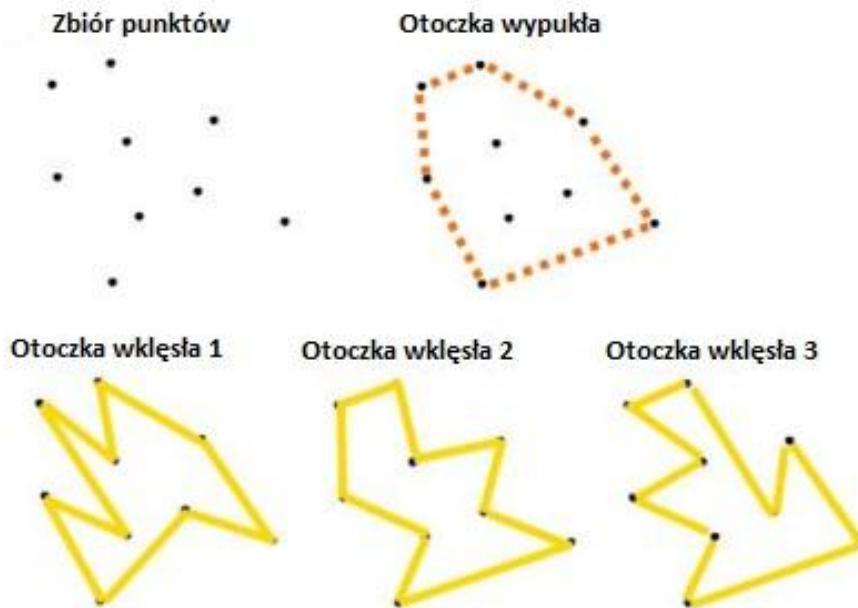
### **5.2.4.1. Definicja problemu**

Każda zmiana położenia drona skutkuje wyznaczeniem nowego obszaru przeszukanego. Zgodnie z opisanym w poprzednim podrozdziale algorytmem każdy z takich obszarów składa się z listy punktów, uszeregowanych według współrzędnych biegunowych, tworzących otoczkę wyznaczonego obszaru oraz listy punktów definiujących dziury w tymże obszarze. Aby być w stanie prawidłowo wizualizować obszar przeszukany nie tylko dla pojedynczych położen drona, ale dla całych śladów potrzebna jest metoda na łączenie tych obszarów w jeden.

Zadanie na pierwszy rzut oka mogłoby się wydawać trywialne, jednak po dokładniejjszej analizie okazuje się, że wcale takie nie jest. Co oznacza łączenie obszarów w całość? Najprościej rzecz ujmując jest to wyznaczenie wielokąta takiego, że każdy z punktów należących do obu obszarów leży albo na brzegu wielokąta albo w jego wnętrzu. Taka figura płaska, w zależności od jej charakteru, określana jest otoczką wypukłą lub wklęską zbioru punktów. W przypadku otoczki wypukłej istnieje wiele stosunkowo prostych w implementacji algorytmów na wyznaczenie takiego wielokąta, jednak problem pojawia się w momencie, gdy któryś z tych obszarów jest wielokątem wklęsłym, co w analizowanym przykładzie jest częstą sytuacją.

**Def. 5.1** Otoczka wklęsła zbioru punktów P to wielokąt wklęsły taki, że każdy z punktów należących do tego zbioru leży albo na brzegu wielokąta, albo w jego wnętrzu.

Z matematycznego punktu widzenia dla dowolnego zestawu punktów P nie istnieje jednoznaczny wielokąt wklęsły zawierający wszystkie te punkty. Zazwyczaj istnieje wiele takich wielokątów. Problem ten został zobrazowany na rys. 6.5.



Rys. 5.5 Zobrazowanie problemu jednoznacznego wyznaczania otoczki wklęszej [43]

Jak widać otoczka wypukła byłaby jedynie bardzo niedokładnym przybliżeniem obszaru zarejestrowanego przez kamery, a kolejne otoczki wklęsłe różnią się znacznie między sobą. Zatem potrzebny jest algorytm, który nie tylko będzie wyznaczał otoczke wklęsłą dla danego zbioru punktów, ale będzie to robił w sposób powtarzalny i dokładny. W kolejnym podrozdziale opisano rozwiązywanie tego problemu.

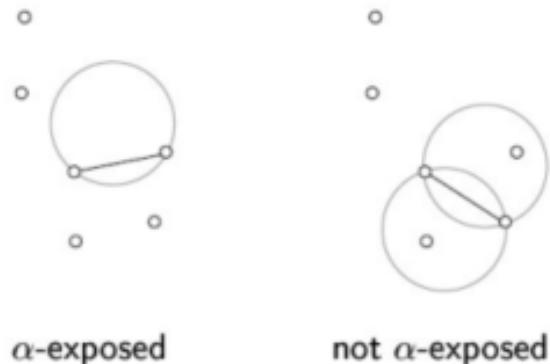
#### 5.2.4.2. Wyznaczanie otoczki wklęszej - ksztalt $\alpha$

Jak można wywnioskować na podstawie rys. 6.5 otoczka wypukła nie zawsze dokładnie przedstawia obszar reprezentowany przez dany zbiór punktów i czasami jej wyznaczenie okazuje się niewystarczające pod względem dokładności.

Alternatywą jest wyznaczenie otoczki wklęszej. Jednak dokonanie tego wymaga zastosowania bardziej skomplikowanych algorytmów, takich jak algorytm "Swing Arm" Antony'ego Galton'a oraz Matt'a Duckham'a oparty na algorytmie Jarvisa, nazywanym algorytmem

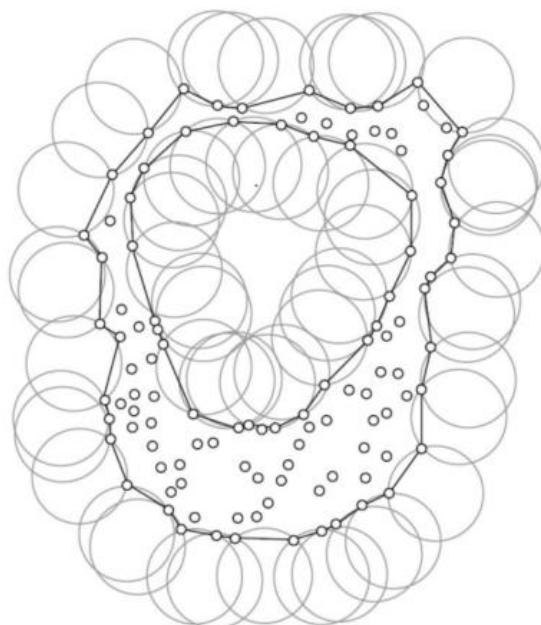
owijania prezentów [44], czy metoda oparta na algorytmie k-najbliższych sąsiadów, opracowana przez Adriano Moreira i Maribel Yasmina Santos [45].

Kolejną możliwością jest wykorzystanie koncepcji kształtu  $\alpha$ , zdefiniowanej w 1983 roku przez trzech naukowców Edelsbrunner Herber'a, Kirkpatrick David'a i Seidel Raimnud'a [45].



Rys. 5.6 Zasada wyznaczanie pojedynczych krawędzi grafu [50]

Kształt  $\alpha$  dowolnego zbioru punktów jest subgrafem triangulacji Delaunay tych punktów, takim że dwa punkty stanowią jego krawędź jeśli istnieje pusta kula o promieniu  $1/\alpha$  stykająca się z tymi dwoma punktami. Jeśli  $\alpha=0$  kula zostaje zastąpiona półpłaszczyzną i wówczas kształt alfa jest równy otoczce wypukłej danego zbioru punktów [47].

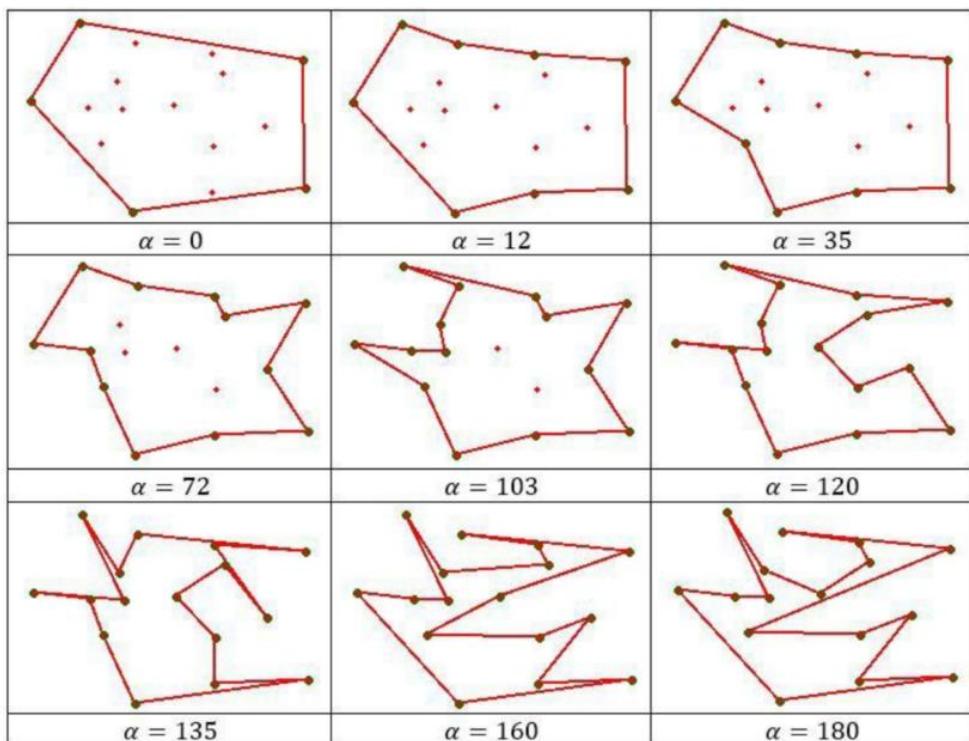


Rys. 5.7 Intuicyjny przykład wyznaczania kształtu alfa dla danego zbioru punktów [50]

Kształt alfa jest generalizacją otoczki wypukłej, to znaczy każda otoczka wypukła jest kształtem alfa, ale nie każdy kształt alfa jest otoczką wypukłą.

W dalszej części tego rozdziału zostanie zdefiniowana otoczka  $\alpha$ -wklęsła, czyli otoczka wklęsła zdefiniowana poprzez kształt  $\alpha$  dla danego zbioru punktów. W celu poprawnej definicji otoczki  $\alpha$ -wklęszej w pierwszej kolejności zostanie przedstawiona definicja wielokąta alfa.

**Def. 5.2** Prosty wielokąt A jest wielokątem alfa, wtedy i tylko wtedy, gdy wszystkie jego wewnętrzne kąty są mniejsze lub równe wartości  $180^\circ + \alpha$  [47].



Rys. 5.8 Przykładowe wyniki wyznaczania wielokąta alfa dla różnych wartości parametrów  $\alpha$  [47]

**Def. 5.3** Otoczka  $\alpha$ -wklęsła zbioru punktów P jest wielokątem alfa, o najmniejszej powierzchni, który zawiera wszystkie punkty zbioru P [47].

W przypadku łączenia obszaru ostatnio przeszukanego z dotychczas przeszukanym danymi wejściowymi algorytmu są dwa zbiory punktów, tworzące wielokąty zamknięte. W

celu wyznaczenia otoczki wklęsłej dla zbioru  $P$ , będącego sumą tych zbiorów przeprowadzono kolejno takie operacje :

1. Dla każdego ze zbiorów wejściowych, w celu zwiększenia dokładności działania algorytmu wyznaczania otoczki, dokonano densyfikacji punktów.

**Def. 5.4** Densyfikacja punktów to operacja polegająca na takim zagęszczeniu punktów, że dowolne dwa, kolejne punkty zbioru są od siebie oddalone o wartość nie większą niż określona wartość.

Operacja densyfikacji polegała na sprawdzeniu odległości między kolejnymi punktami zbioru i porównaniu jej z parametrem  $\lambda$ , o wartości dobranej na drodze eksperymentu, zbliżonej do przyjętej wartości parametru  $\alpha$ . Jeśli odległość między dwoma kolejnymi punktami była większa od wartości  $\lambda$ , między te dwa punkty zostawał wstawiony dodatkowy punkt. Operacja ta była powtarzana, do momentu, gdy dowolne dwa, kolejne punkty zbioru były od siebie oddalone o wartość nie większą niż wartość parametru  $\lambda$ .

2. Następnie dokonano połączenia zbiorów wejściowych punktów, reprezentujących obszar ostatnio przeszukany i dotychczas przeszukany w sumaryczny zbiór punktów  $P$ .
3. Dla zbioru punktów  $P$  przeprowadzono triangulację Delaunay.

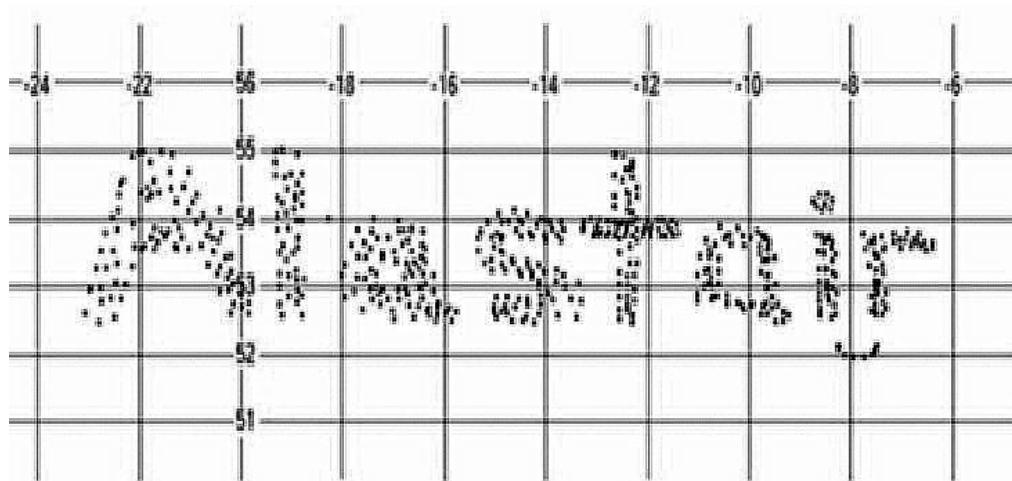
**Def. 5.5** Triangulacja Delaunay zbioru punktów  $P$  jest takim podziałem obszaru wyznaczonego przez ten zbiór punktów na trójkąty, że żaden z punktów tego zbioru nie znajduje się we wnętrzu któregokolwiek z okręgów opisanych na trójkątach powstałych podczas triangulacji.

W wyniku triangulacji zamiast zbioru punktów otrzymano zbiór nienakładających się na siebie trójkątów.

4. Poprzez połączenie wszystkich trójkątów, dla których promień okręgu na nich opisanego jest mniejszy od dobranej wartości parametru  $\alpha$ , wyznaczono graf reprezentujący poszukiwaną otoczkę.

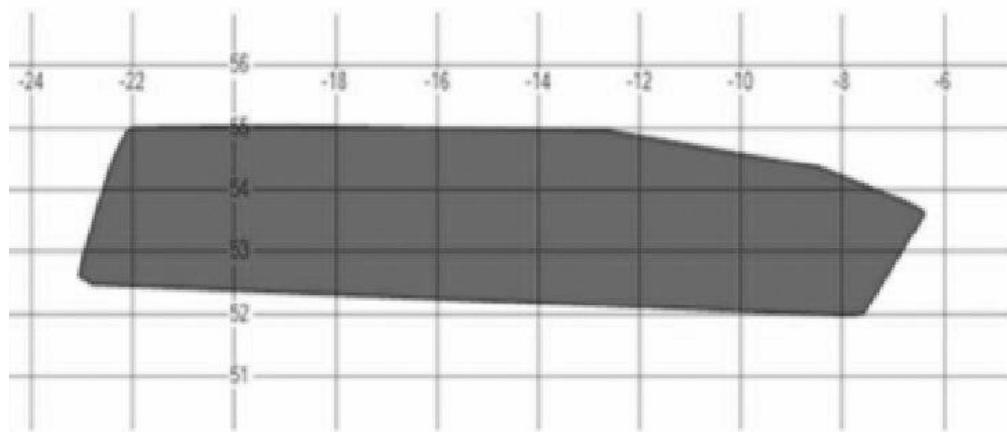
Zastosowanie powyższego algorytmu, przy dobraniu odpowiedniej wartości parametru  $\alpha$  pozwoliło na uzyskanie zadowalających rezultatów. Należy jednak mieć na uwadze, jak zostało to już wcześniej napisane, że dla dowolnego zbioru punktów nie istnieje jednoznaczna reprezentacja otoczki wklęszej, a koncepcja otoczki  $\alpha$ -wklęszej jest przybliżeniem poszukiwanego wielokąta.

Na zakończenie rozważań dotyczących kształtów alfa zostanie przedstawiony przykład zastosowania opisanego w tym rozdziale algorytmu dla zbioru punktów  $P$  przedstawionego na rys. 6.9.



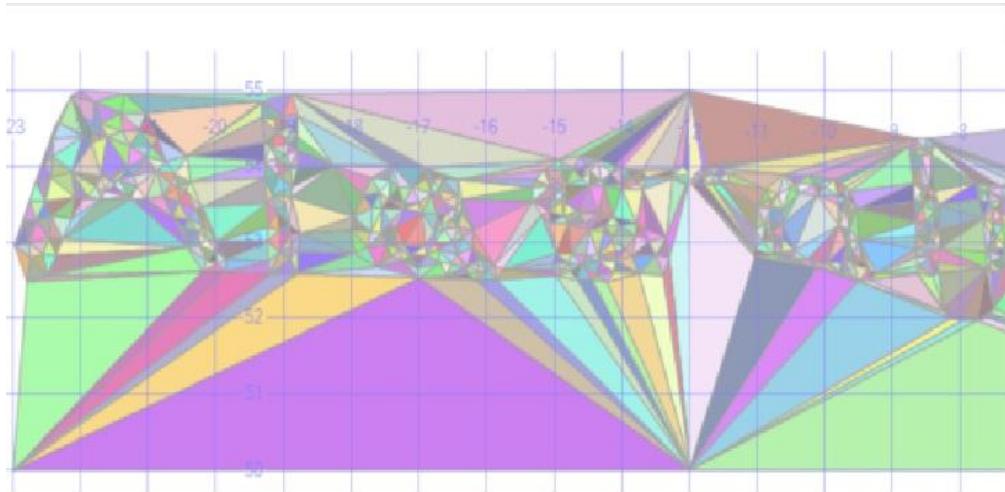
Rys. 5.9 Przykładowy zbiór punktów  $P$  [43]

Najprostszą aproksymacją tego zbioru jest jego otoczka wypukła przedstawiona na rys. 6.10.



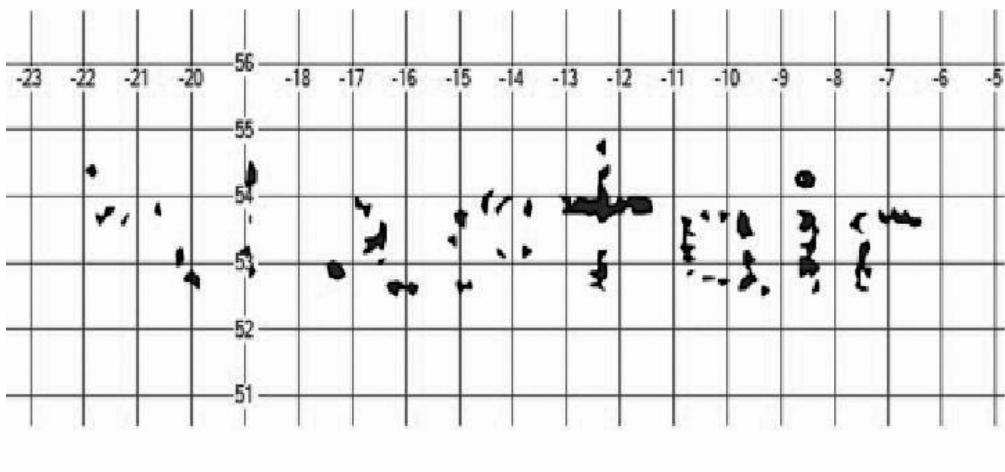
Rys. 5.10 Otoczka wypukła zbioru punktów  $P$  [43]

Na rys. 6.11 został przedstawiony zbiór trójkątów, będący efektem triangulacji Delaunay'ego zbioru punktów P.

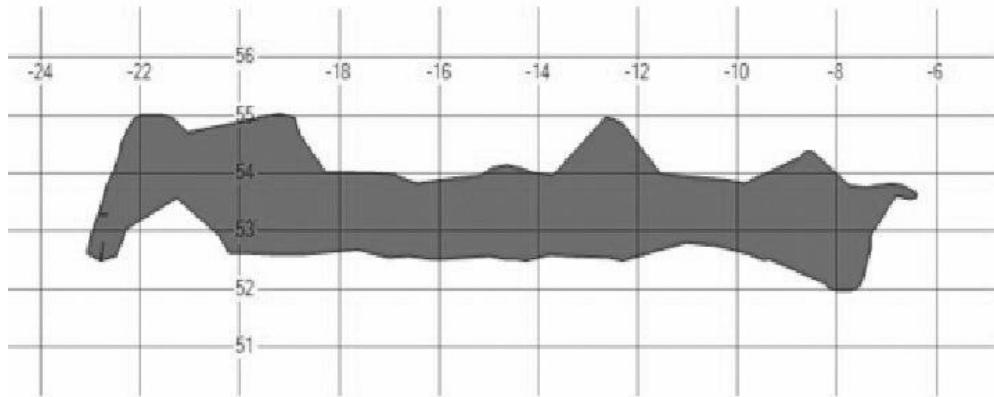


Rys. 5.11 Efekt przeprowadzenia triangulacji Delaunay na zbiorze punktów P [43]

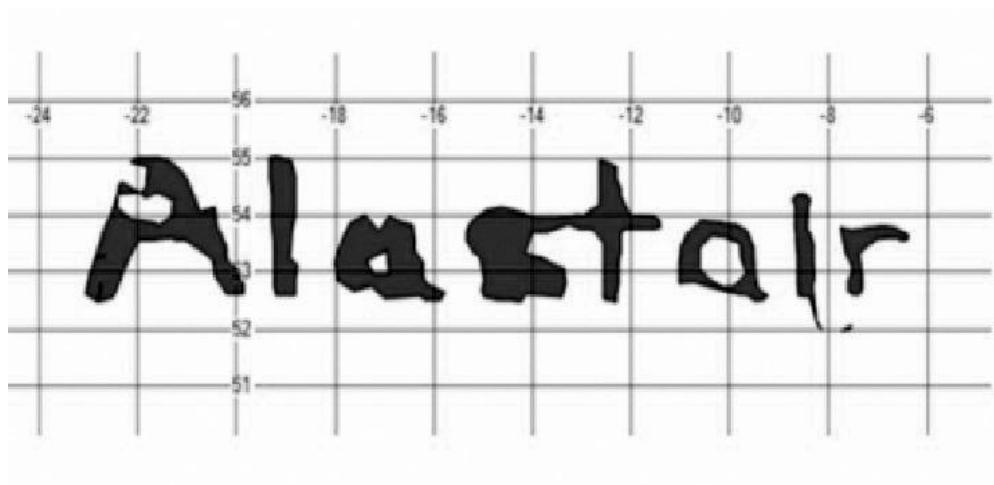
Algorytm wyznaczenia kształtu alfa zależy odbranej wartości parametru  $\alpha$ . Na rysunkach poniżej zostały przedstawione trzy różne wyniki algorytmu dla wartości parametru kolejno  $\alpha=0.1$ ,  $\alpha=1$  oraz  $\alpha=0.24$ . Jak widać w przypadku za małej wartości parametru  $\alpha$  wygenerowany zostaje zestaw wielu niepołączonych obszarów (rys. 6.12), a dla zbyt dużej wartości parametru wyznaczony kształt zaczyna być coraz bardziej zbliżony do otoczki wypukłej (rys. 6.13). Poprzez dobranie odpowiedniej wartości  $\alpha$ , w tym wypadku  $\alpha=0.24$ , otrzymano wynik bardzo zbliżony do oczekiwanej reprezentacji punktów (rys. 6.14).



Rys. 5.12 Wynik algorytmu dla parametru  $\alpha = 0.1$  [43]



Rys. 5.13 Wynik algorytmu dla parametru  $\alpha = 1$  [43]



Rys. 5.14 Wynik algorytmu dla parametru  $\alpha = 0.24$  [43]

#### 5.2.4.3. Łączenie obszarów, a części zasłonięte wewnątrz obszarów

Podczas łączenia obszarów przeszukanych w jeden należy również uwzględnić części wewnętrz nich, uznane za zasłonięte.

W tym celu przy każdym wyznaczeniu nowej wartości obszaru dotychczas przeszukanego dokonywano sprawdzenia, czy któryś z punktów, należących do zbioru, reprezentującego pojedynczą dziurę w tym obszarze znajduje się we wnętrzu obszaru ostatnio przeszukanego. Jeśli tak punkt ten zostawał usunięty, a wielokąt reprezentujący daną dziurę zostawał raz jeszcze wyznaczony zgodnie z algorytmem obliczania otoczki  $\alpha$ -wklesłej. W ten sposób w obszarze wynikowym były uwzględnione jedynie te części dziur obszaru dotychczas przeszuku-

kanego, które nie znajdowały się wewnątrz obszaru ostatnio przeszukanego oraz wszystkie dziury wyznaczone dla obszaru ostatnio przeszukanego.

### 5.2.5. Podsumowanie

Opracowany algorytm pozwala na prawidłowe wyznaczenie obszarów zarejestrowanych przez kamerę, przy uwzględnieniu założeń i uproszczeń przedstawionych na początku tego rozdziału. Rzeczywiste efekty działania algorytmu, w zaprojektowanym systemie, zostaną przedstawione w rozdziale pt. "Prezentacja rozwiązania". Należy jednak mieć na uwadze dokładność algorytmu, która jest zależna od wielu zmiennych.

W pierwszej kolejności zarówno dane z odbiornika GPS, jak i dane modelowe z misji SRTM są obarczone pewnym błędem, a ich dokładność została opisana w rozdziale "Dziedzina problemu". Dodatkowo w przypadku dostosowywania systemu do funkcjonowania na konkretnym terenie należałoby w dokładniejszy sposób określić średnią wartość lokalnej undulacji lub stworzyć całą tablicę undulacji, dla konkretnych położen. Dokładność działania algorytmu zależy również od prawidłowego doboru takich parametrów jak: skok wysokości  $\Delta h$  przy wyznaczaniu otoczki pojedynczego obszaru zarejestrowanego oraz dziur w tym obszarze, skok kątowy  $\beta$  przy znajdowaniu zasłoniętych obszarów, odległość kątowa między kolejnymi punktami na okręgu, służącymi do wyznaczenia otoczki obszaru, odpowiednie dobranie parametru  $\alpha$  dla algorytmu wyznaczania otoczki wklęszej oraz wartości parametru  $\lambda$  wykorzystywanego przy densyfikacji punktów poszczególnych zbiorów.

Powyższe parametry wpływają nie tylko na dokładność wyznaczania obszaru przeszukanego, ale również na wydajność działania systemu, rozumianą jako czas między zarejestrowaniem zmiany położenia drona, a wizualizacją nowo zarejestrowanego obszaru na urządzeniu klienckim. Jednak optymalizacja algorytmu wymagałaby przeprowadzenia wielu testów wydajnościowych, z wykorzystaniem rzeczywistego drona i nie wchodzi w zakres tej pracy.

## **6. Projekt systemu**

Kolejną fazą po analizie i opracowaniu algorytmu jest faza projektowania systemu, która polega na przekształceniu modelu analitycznego, czyli opisu systemu z punktu widzenia aktorów, zrozumiałego dla klienta, w model zawierający informację o wewnętrznej strukturze systemu, jego konfiguracji sprzętowej oraz sposobach jego realizacji. W efekcie w trakcie tego etapu powstają: modele struktury, opisujące statyczną budowę systemu oraz modele zachowania, opisujące aspekty dynamiczne systemu.

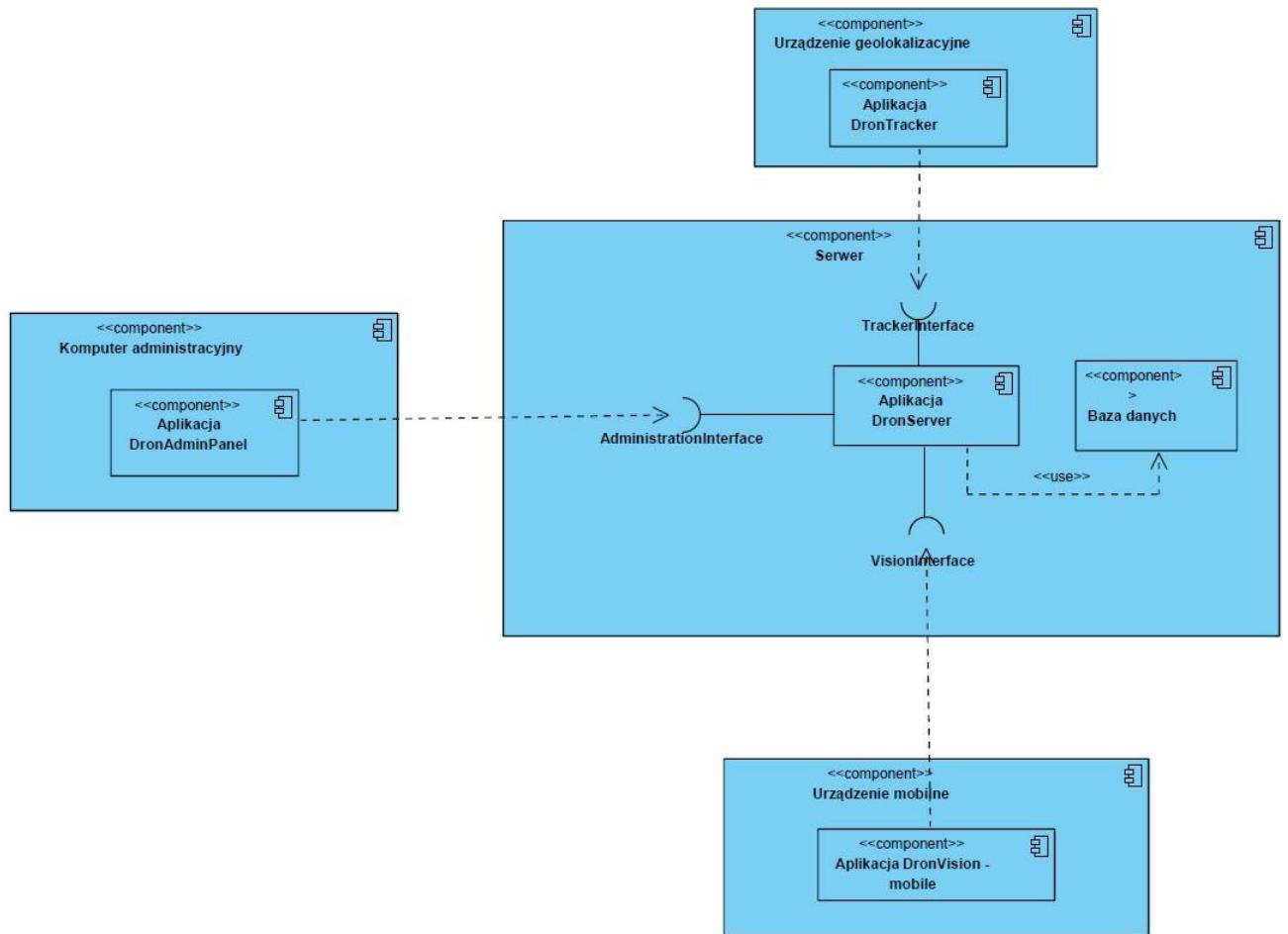
W tym rozdziale zostaną przedstawione kolejno:

- diagram komponentów,
- diagram klas części serwerowej,
- diagram aplikacji wizualizującej,
- model danych,
- diagramy aktywności,
- diagramy sekwencji,
- projekt interfejsu graficznego aplikacji wizualizującej

Wszystkie diagramy zawarte w tym rozdziale zostały dodatkowo dołączone jako załączniki o lepszej czytelności.

## 6.1. Diagram komponentów

Diagram komponentów pokazuje podział systemu na mniejsze podsystemy. Za komponent uznaje się wymienialny, wykonywalny fragment systemu, z ukrytymi szczegółami implementacyjnymi [38].



Rys. 6.1 Diagram komponentów

Jak widać system został podzielony na cztery podsystemy:

- **DronServer** - część serwerowa systemu,
- **DronAdminPanel** - Panel administracyjny
- **DronTracker** - Aplikacja zbierająca dane geolokalizacyjne i wysyłające je do serwera
- **DronVision** - Główna aplikacja kliencka, służąca do wizualizacji obszaru przeszukanego

## **6.2. Diagramy klas**

Diagram klas jest statycznym diagramem strukturalnym. Przedstawia strukturę systemu w modelu obiektowym przez ilustrację struktury klas i zależności między nimi.

Ze względu na skomplikowaną budowę systemu diagramy klas, dołączone do pracy, zostały ograniczone do przedstawienia zależności między klasami, bez opisu ich wewnętrznej struktury takiej jak lista atrybutów, czy metod. Dodatkowo, w celu poprawienia czytelności diagramów, klasy zostały podzielone na mniejsze logiczne grupy, takie jak klasy odpowiadające za logikę biznesową, czy klasy reprezentujące obiekty bazodanowe, a powiązanie między klasami z różnych grup zostało przedstawione jako powiązanie między całymi grupami tzn. "Klasy logiki biznesowej wykorzystują klasy z modelu domenowego oraz z modelu bazodanowego".

### **6.2.1. Diagram klas części serwerowej**

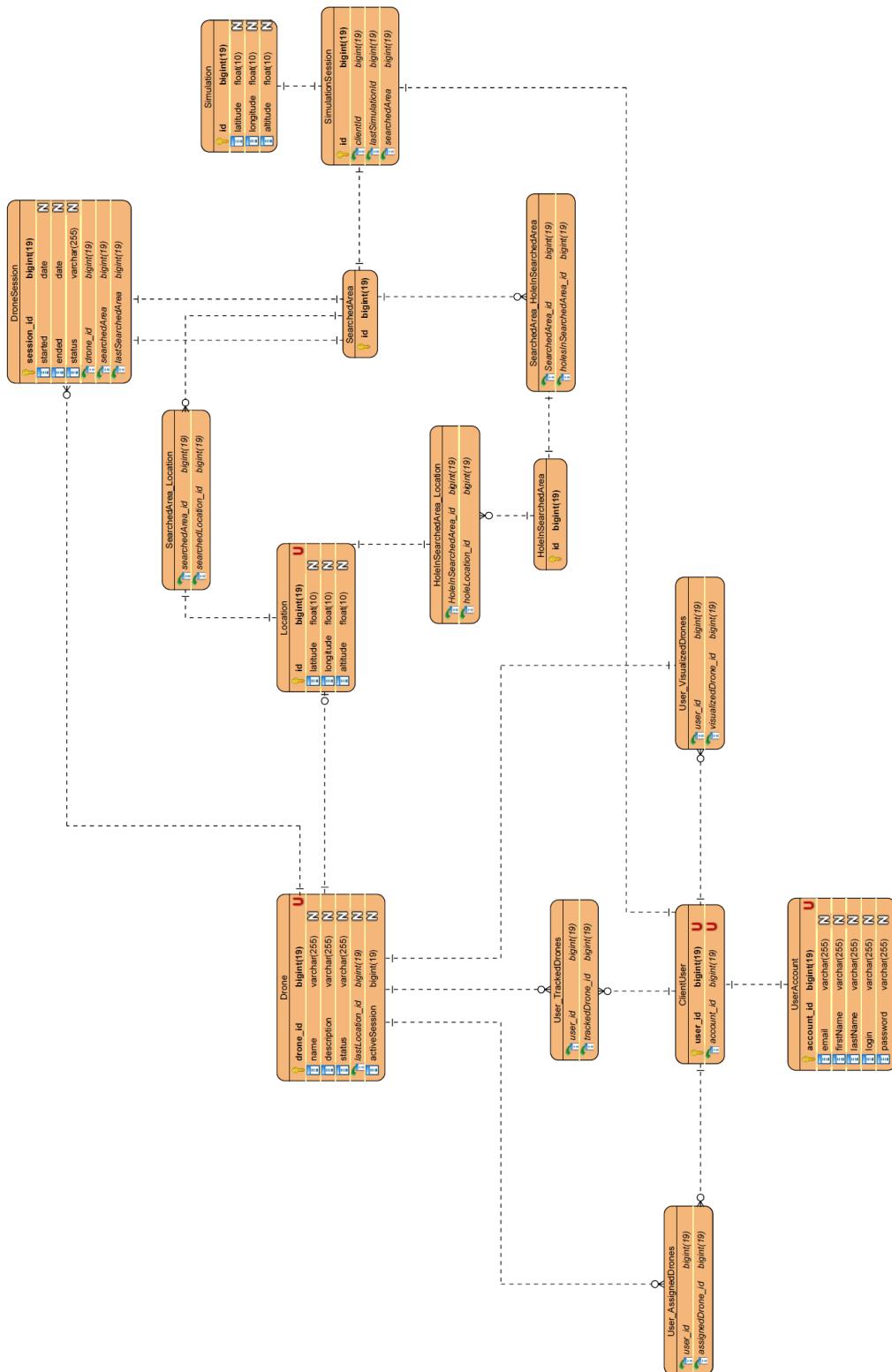
Diagram klas części serwerowej ze względu na duży rozmiar został dołączony do pracy jako załącznik nr 1.

### **6.2.2. Diagram klas aplikacji klienckiej**

Diagram klas aplikacji klienckiej ze względu na duży rozmiar został dołączony do pracy jako załącznik nr 2.

### 6.3. Diagram związków encji

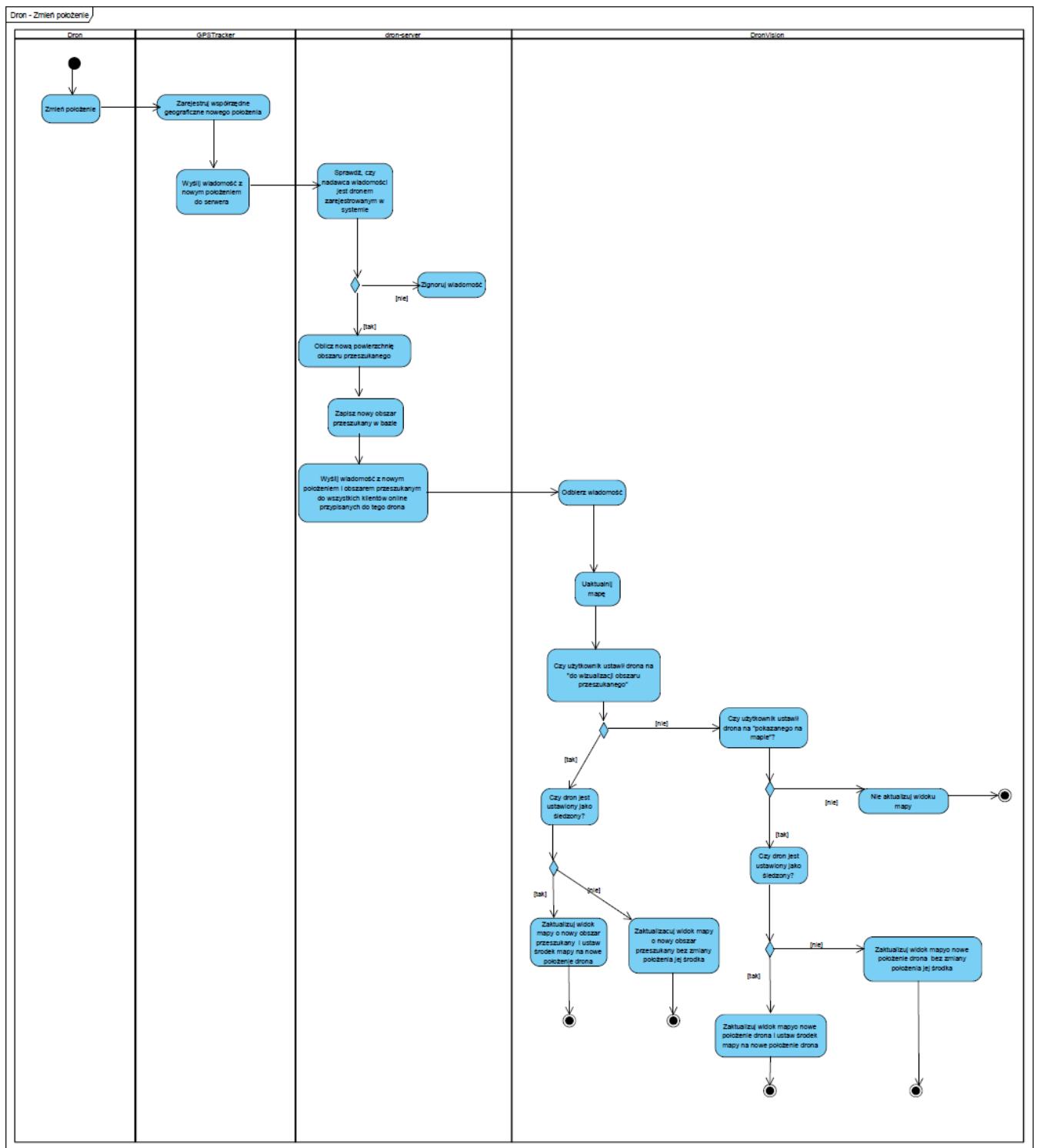
Jest to rodzaj diagramu, przedstawiającego związki pomiędzy encjami, opisujący konceptualny model danych używanych w systemie informatycznym.



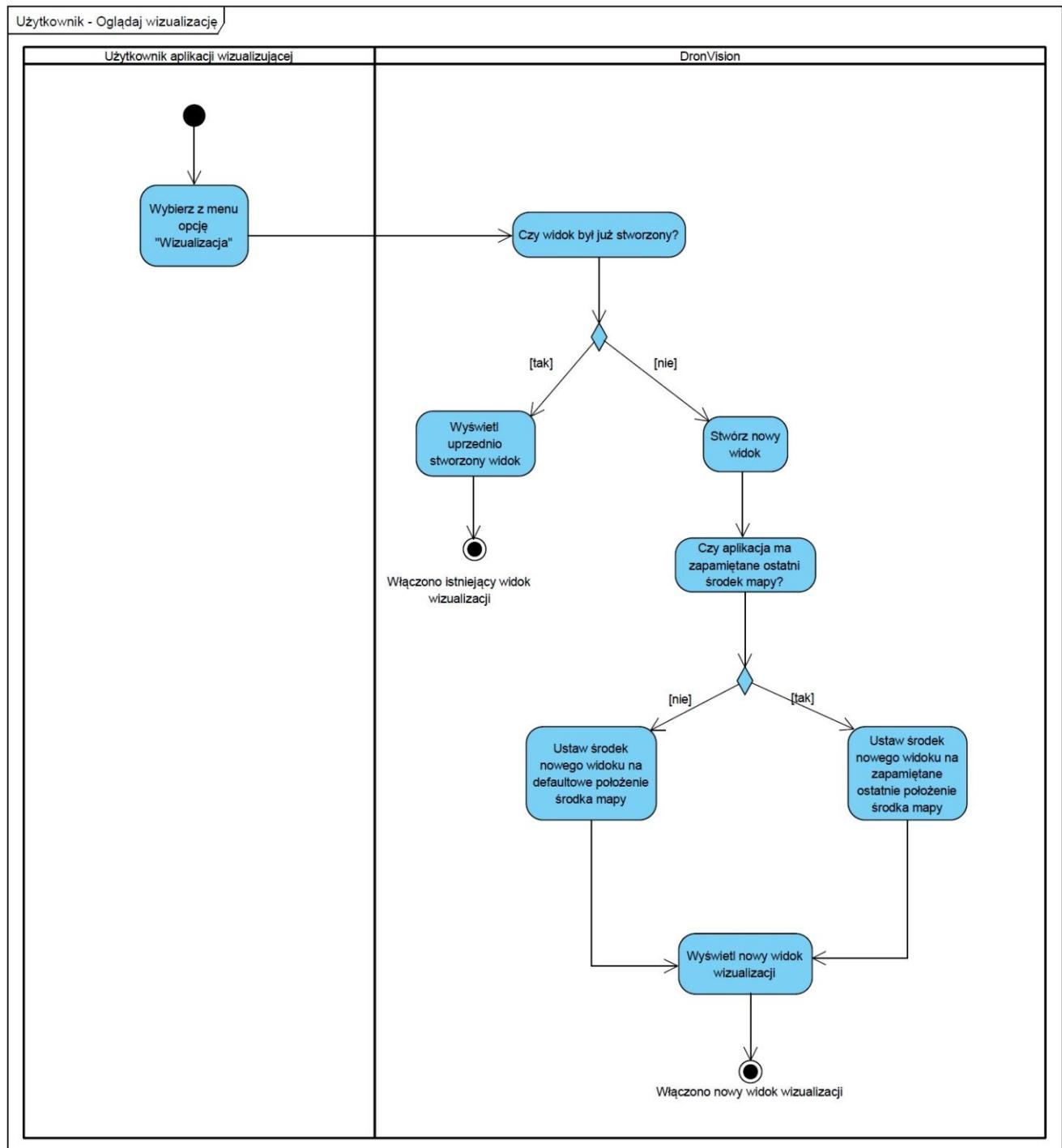
Rys. 6.2 Diagram związków encji

## 6.4. Diagramy aktywności

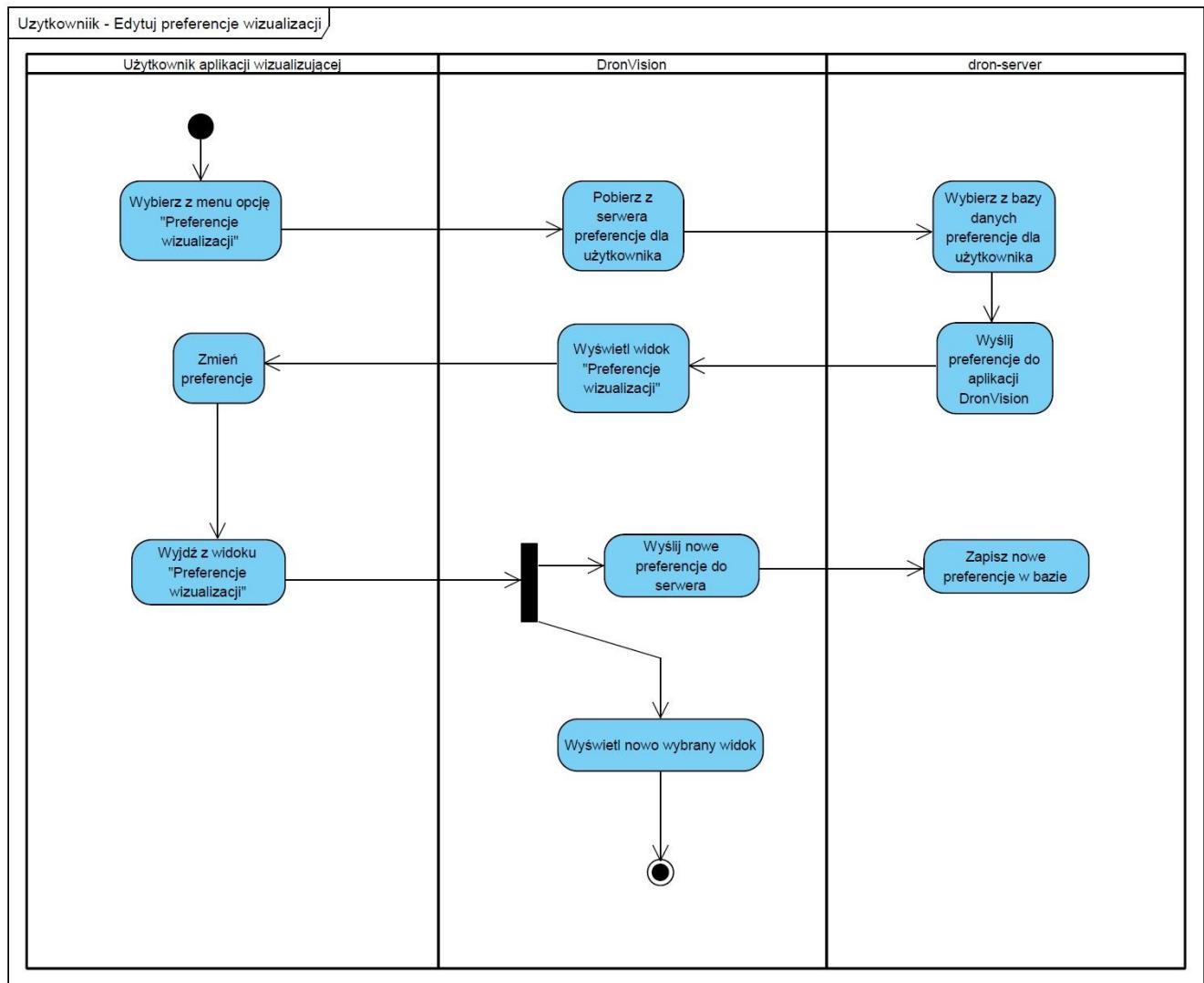
### 6.4.1. Diagram aktywności - Dron - Zmień położenie



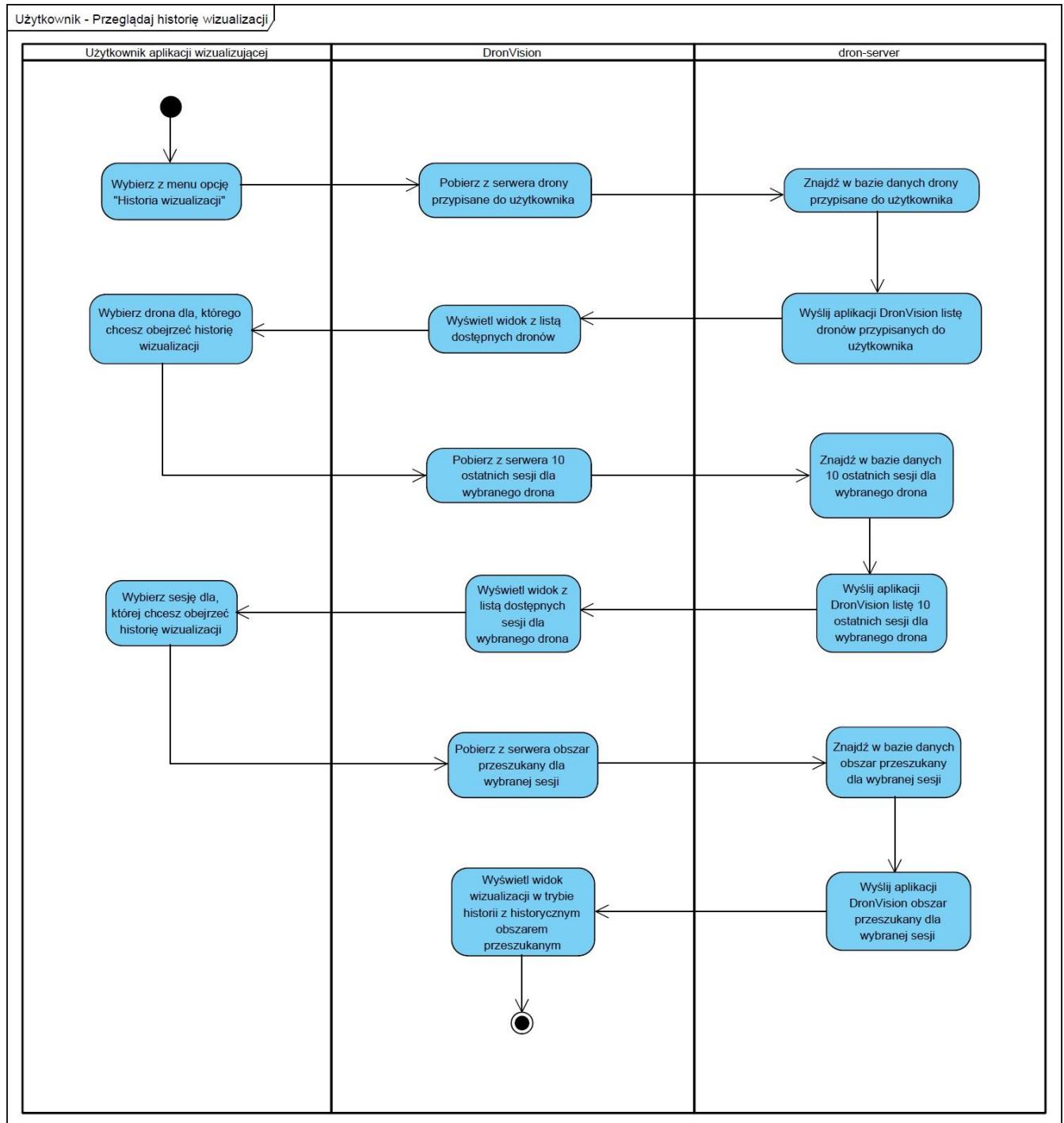
## 6.4.2. Diagram aktywności - Użytkownik aplikacji wizualizującej - Oglądaj wizualizację



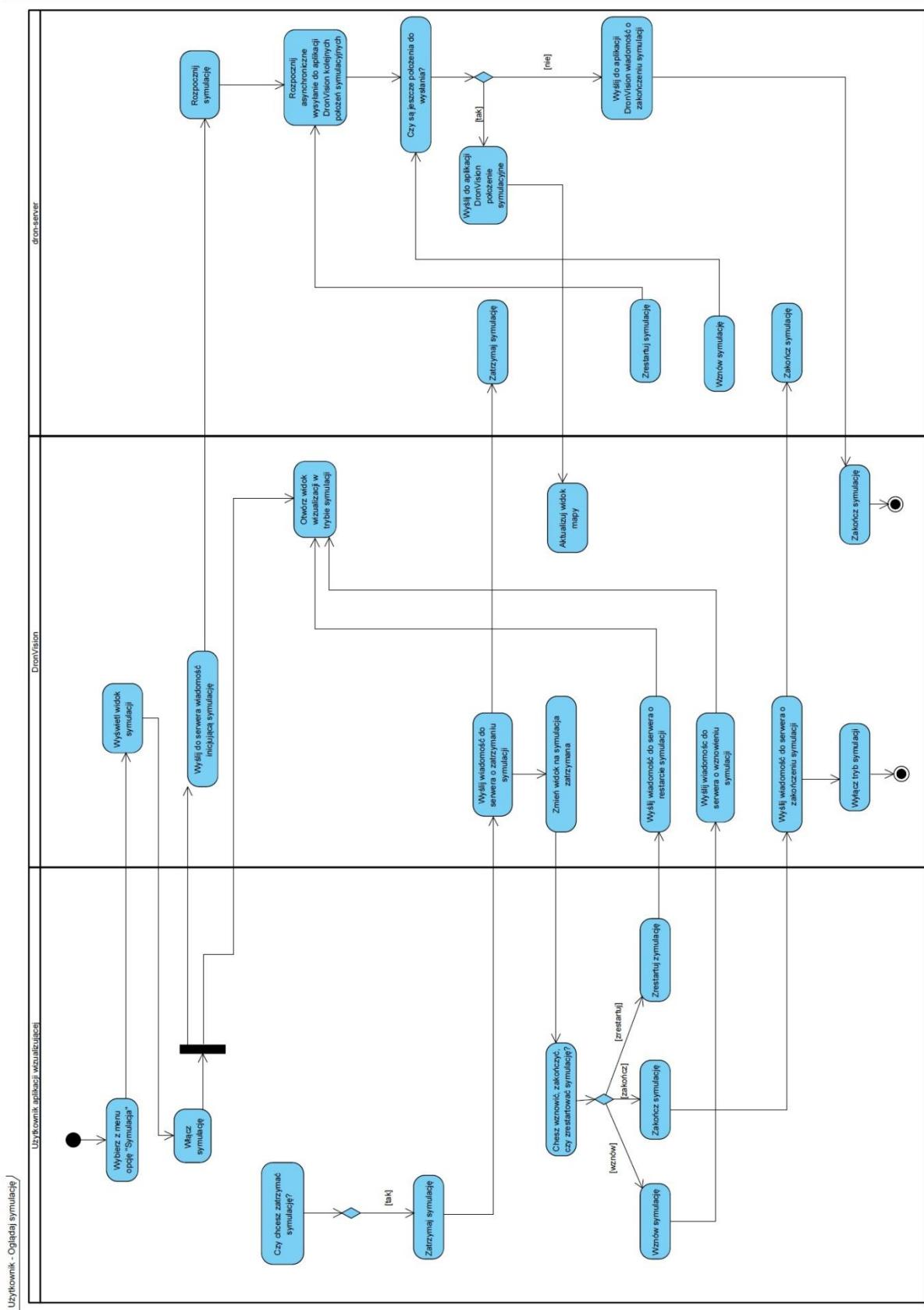
### 6.4.3. Diagram aktywności - Użytkownik aplikacji wizualizującej - Edytuj preferencje



#### 6.4.4. Diagram aktywności - Użytkownik aplikacji wizualizującej - Przeglądaj historię wizualizacji

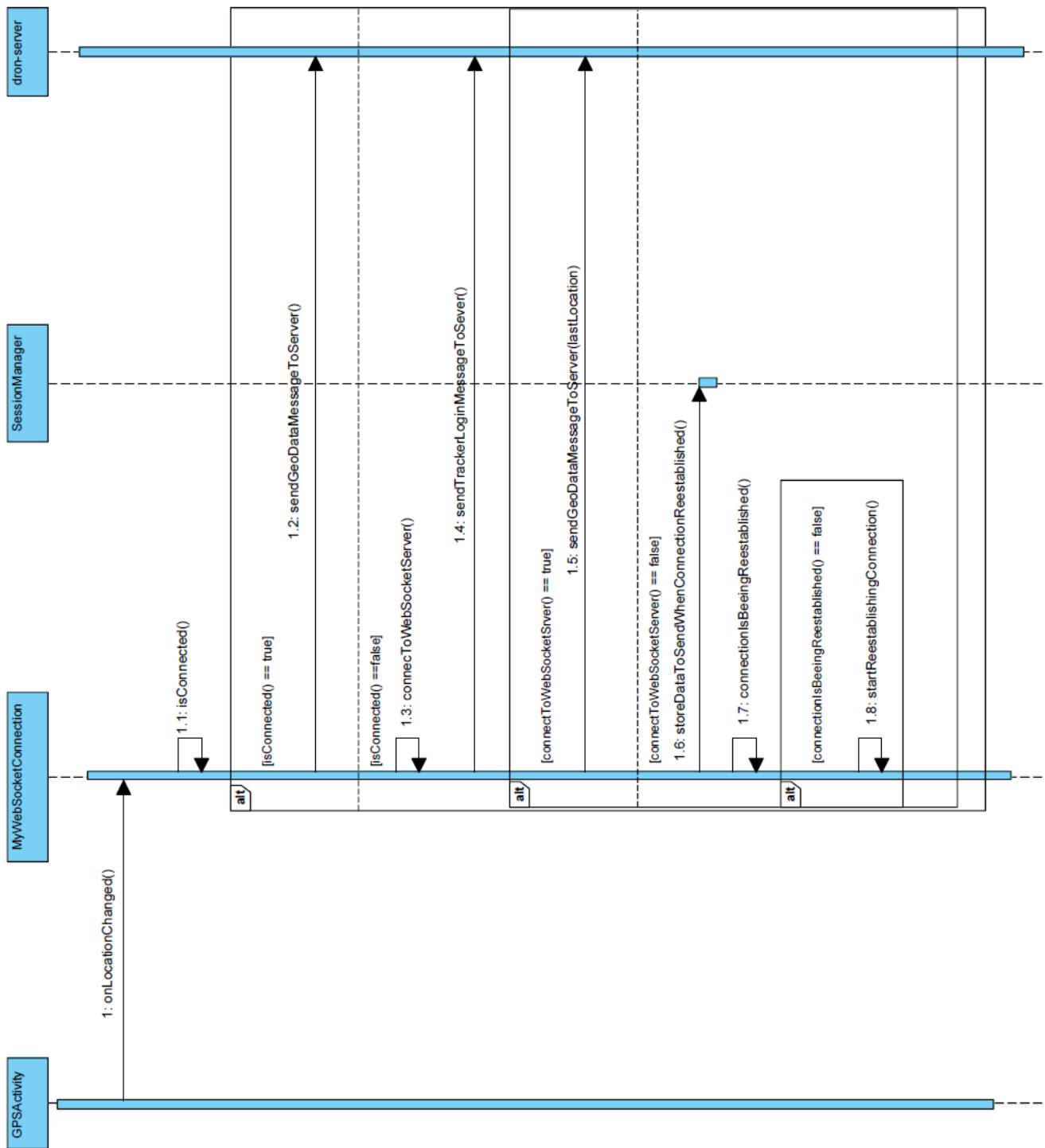


#### **6.4.5. Diagram aktywności - Użytkownik aplikacji wizualizującej - Oglądaj Symulację**

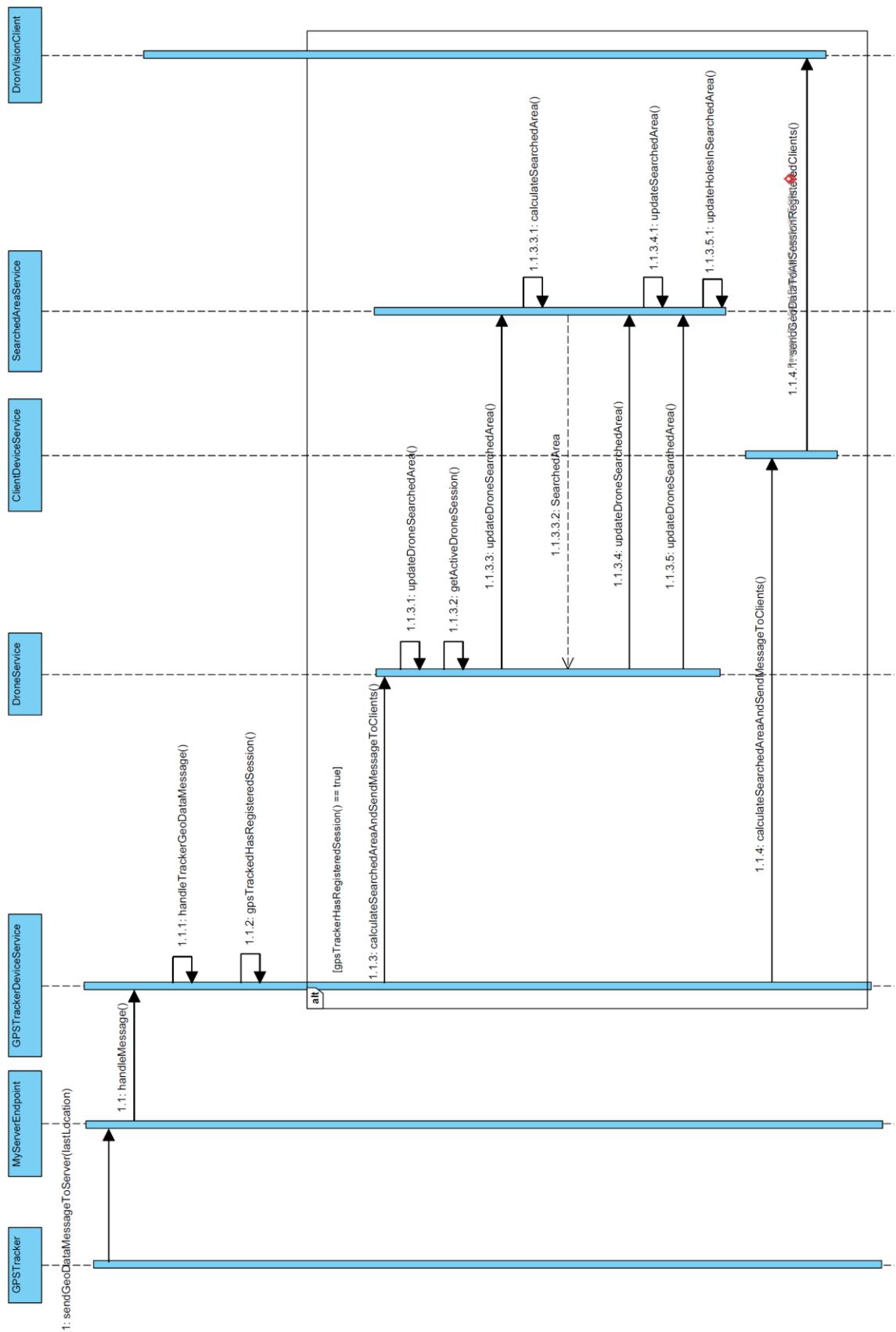


## 6.5. Diagramy sekwencji

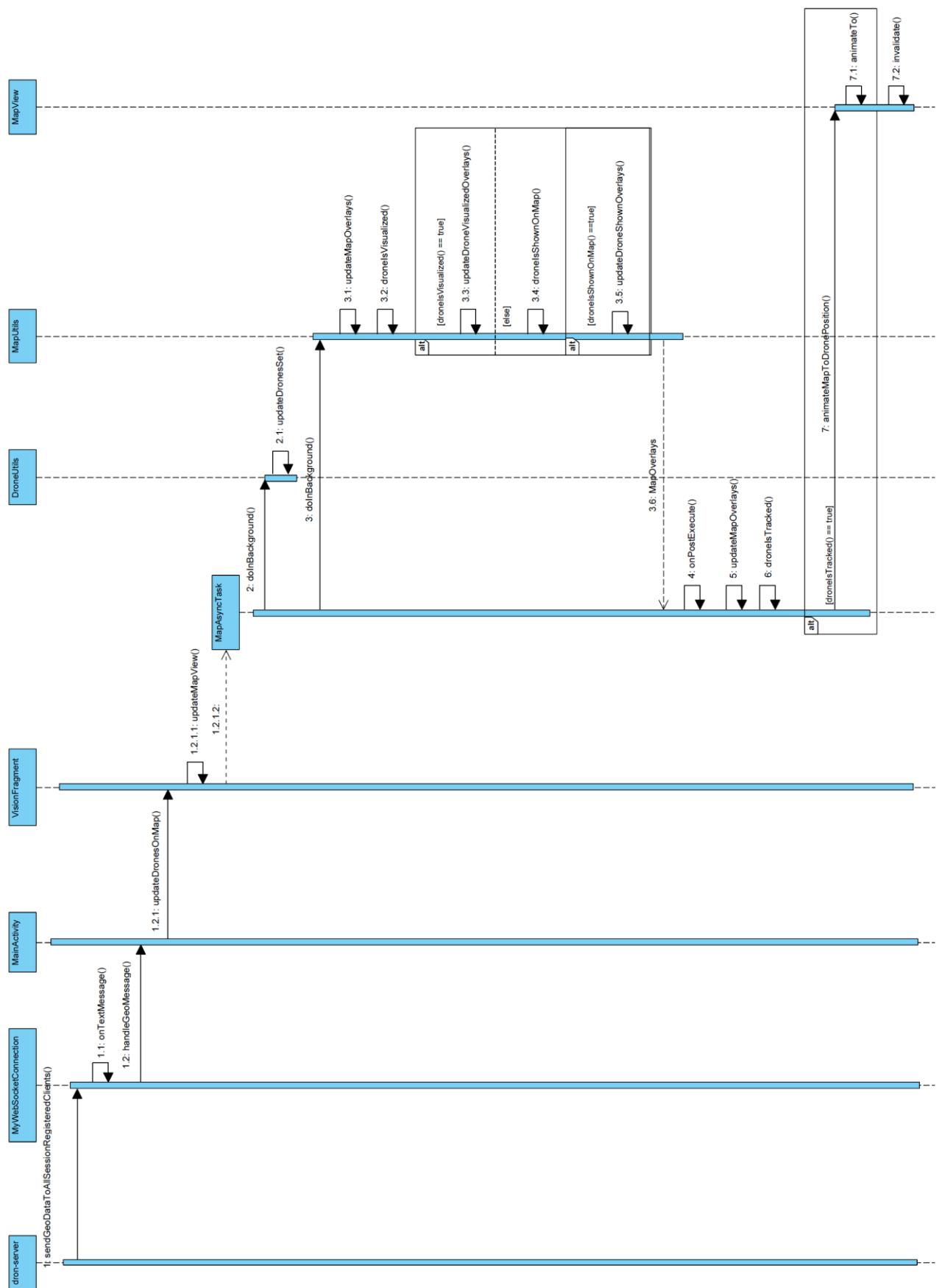
### 6.5.1. Diagram sekwencji - DronTracker - Zmiana położenia



## 6.5.2. Diagram sekwencji - DronServer - Nowa wiadomość od Dron-Tracker'a



### 6.5.3. Diagram sekwencji - DronVision - Nowa wiadomość wizualizacyjna



## **6.6. Projekt interfejsu graficznego (GUI)**

Projektu interfejsu graficznego, ze względu na duże rozmiary został dołączony do pracy jako załącznik nr 3.

## **7. Prezentacja rozwiązania**

Ten rozdział zostanie poświęcony prezentacji stworzonego systemu, który jak zostało to już wcześniej opisane został podzielony, pod względem funkcjonalności, na cztery podsystemy:

- **DronServer** - część serwerowa systemu,
- **DronAdminPanel** - Panel administracyjny
- **DronTracker** - Aplikacja zbierająca dane geolokalizacyjne i wysyłające je do serwera
- **DronVision** - Główna aplikacja kliencka, służąca do wizualizacji obszaru przeszukanego

Ze względu na szeroki zakres pracy, w trakcie jej tworzenia skupiono się na implementacji jedynie trzech z czterech części systemu, tych najbardziej powiązanych z wizualizacją obszaru przeszukanego, czyli: DronTracker, DronServer oraz DronVision. W pierwszej kolejności zostaną pokrótkę opisane aplikacja geolokalizacyjna oraz część serwerowa, a w dalszej części rozdziału zostanie szczegółowo przedstawiona najbardziej ciekawa, z punktu widzenia użytkownika część systemu, czyli aplikacja DronVision.

### **7.1. Opis aplikacji DronTracker**

Aplikacja DronTracker to aplikacja, której zadaniem jest rejestrowanie zmian położień urządzenia, na którym została zainstalowana oraz wysyłanie tych danych geolokalizacyjnych do serwera. Pozwala na imitowanie przez urządzenie mobilne z systemem operacyjnym Android modułu zamontowanego na dronie. Każda z instancji tej aplikacji ma przypisany unikalny numer identyfikacyjny, umożliwiający jej rozróżnienie przez system od innych aplikacji-dronów i w celu współpracy z innymi elementami systemu musi być zarejestrowana w systemowej bazie danych jako dron. Do jej implementacji wykorzystano API programistyczne android.location.

## **7.2. Opis aplikacji DronServer**

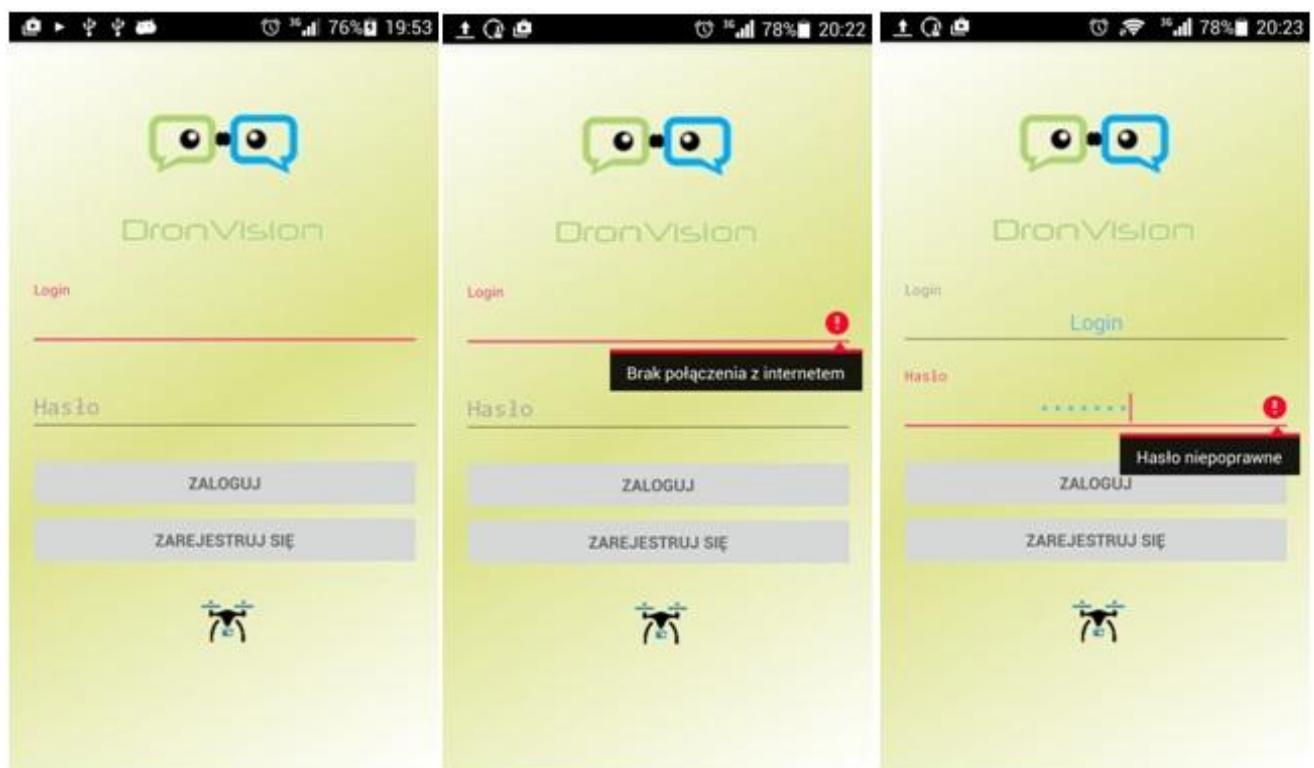
DronServer to aplikacja klasy Enterprise odpowiadająca za główną logikę biznesową systemu. Pełni rolę serwera i odpowiada za komunikację między podsystemami, zbieranie danych geolokalizacyjnych od dronów, ich analizę, wyznaczanie obszaru przeszukanego, zgodnie z opisany w tej pracy algorytmem, komunikację z bazą danych, logikę administracyjną i rozsyłanie danych do aplikacji klienckich DronVision. Moduł ten został stworzony z wykorzystaniem Javy Enterprise Edition i uruchomiony na serwerze aplikacyjnym Wildfly.

## **7.3. Opis aplikacji DronVision**

Aplikacja DronVision jest główną aplikacją kliencką i jako jedyna z prezentowanych modułów posiadana interfejs użytkownika. Pozwala na interakcję z systemem, zgodnie ze specyfikacją ustaloną w fazie analizy wymagań, a jej podstawowym zadaniem jest wizualizacja obszaru przeszukanego. Aplikacja ta zapewnia użytkownikowi szereg funkcji takich jak: zmiana preferencji wizualizacji, przeglądanie historii wizualizacji, obejrzenie symulacji, czy zmiana ustawień konta. W dalszych podrozdziałach zostaną zaprezentowane poszczególne widoki aplikacji wraz z opisem zapewnianych przez nie funkcji.

### 7.3.1. Widoki logowania

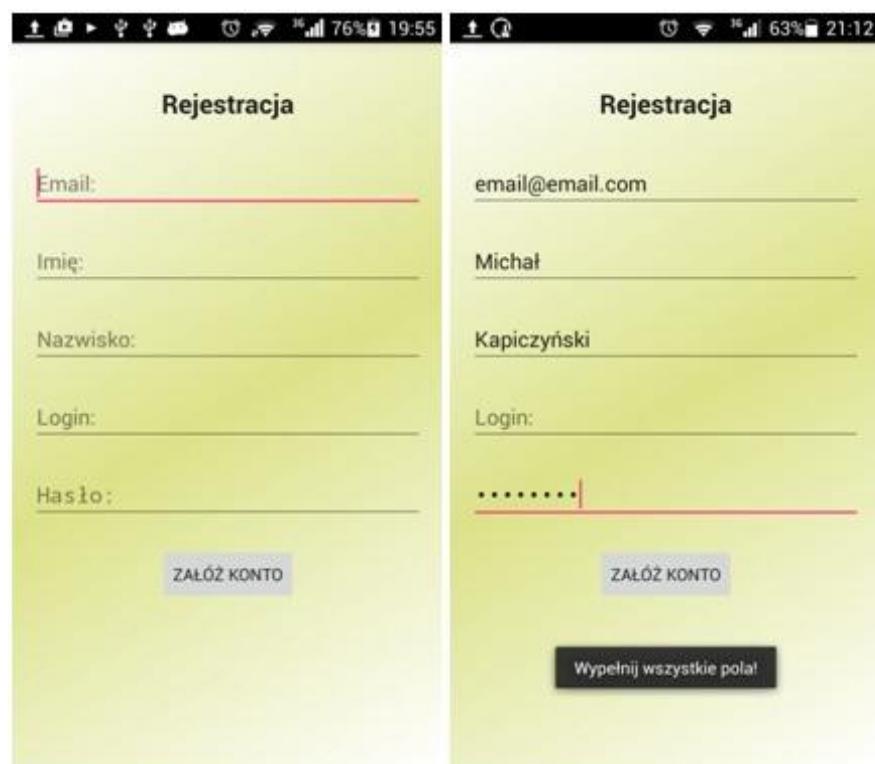
Jest to pierwszy widok jaki ukazuje się użytkownikowi po włączeniu aplikacji. Logowanie do systemu odbywa się poprzez podanie loginu i hasła. W momencie kliknięcia przycisku "Zaloguj" następuje wstępna walidacja sprawdzająca, czy aplikacja posiada połączenie z Internetem oraz czy oba wymagane pola są wypełnione. W przypadku negatywnego wyniku walidacji zostaje wyświetlony odpowiedni komunikat, a jeśli dane zostały wprowadzone poprawnie aplikacja wysyła zapytanie HTTP do serwera i w zależności od odpowiedzi uwierzytelnia użytkownika lub wyświetla komunikat o błędzie.



Rys. 7.1 Aplikacja DronVision - widoki logowania

### 7.3.2. Widoki rejestracji

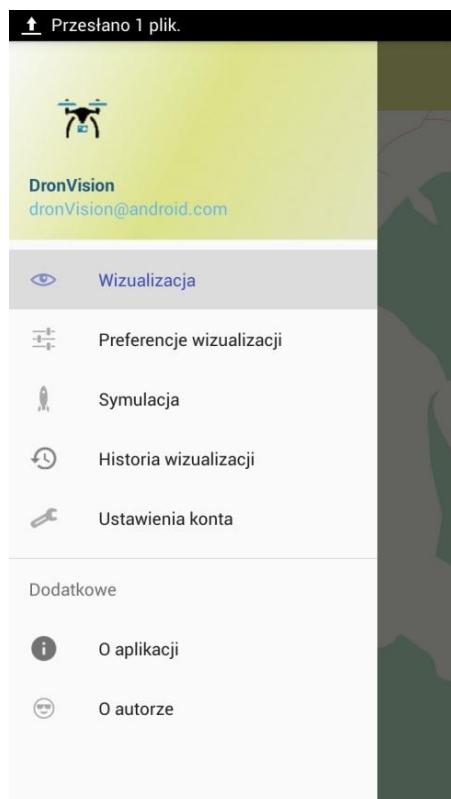
Widok rejestracji to prosty formularz z polami dotyczącymi: adresu email, imienia, nazwiska, loginu i hasła. W momencie naciśnięcia przycisku załóż konto następuje wstępna walidacja formularza pod względem wypełnienia wszystkich pól. W przypadku poprawnego wypełnienia, aplikacja wysyła formularz do serwera, a użytkownik zostaje przeniesiony do widoku logowania wraz z wyświetleniem komunikatu: "Konto zostało utworzone i czeka na zweryfikowanie przez administratora.".



Rys. 7.2 Aplikacja DronVision - widoki rejestracji

### 7.3.3. Panel nawigacyjny - menu

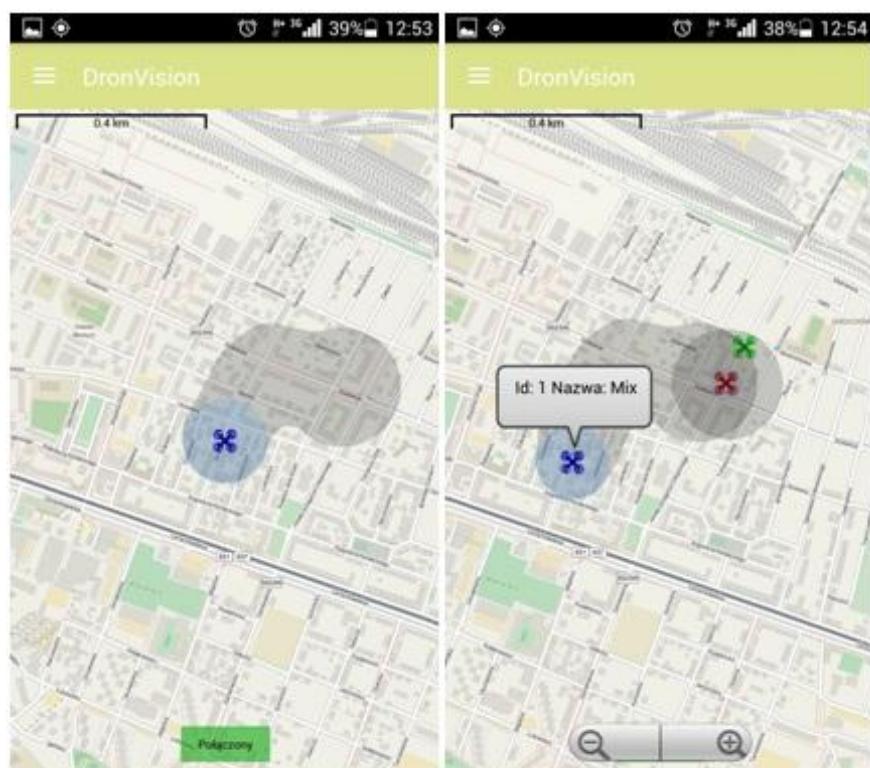
Zalogowany użytkownik aplikacji ma możliwość nawigowania między widokami po przez wysuwany z lewej strony ekranu panel nawigacyjny. Panel ten jest dostępny z każdego z widoków aplikacji, dostępnych po uwierzytelnieniu.



Rys. 7.3 Aplikacja DronVision - widok menu

#### 7.3.4. Widoki wizualizacji

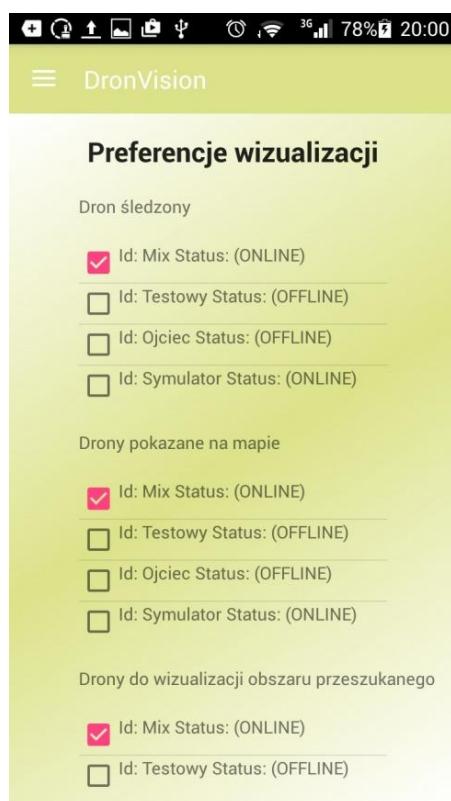
Widok wizualizacji jest głównym widokiem aplikacji, dostępnym dla zalogowanego użytkownika. Przedstawia mapę, na której wizualizowane są drony i ich obszary przeszukane. Widok ten sam w sobie nie zapewnia dodatkowych opcji i służy jedynie do śledzenia efektów wizualizacji.



Rys. 7.4 Aplikacja DronVision - widoki wizualizacji

### 7.3.5. Widok ustawień preferencji wizualizacji

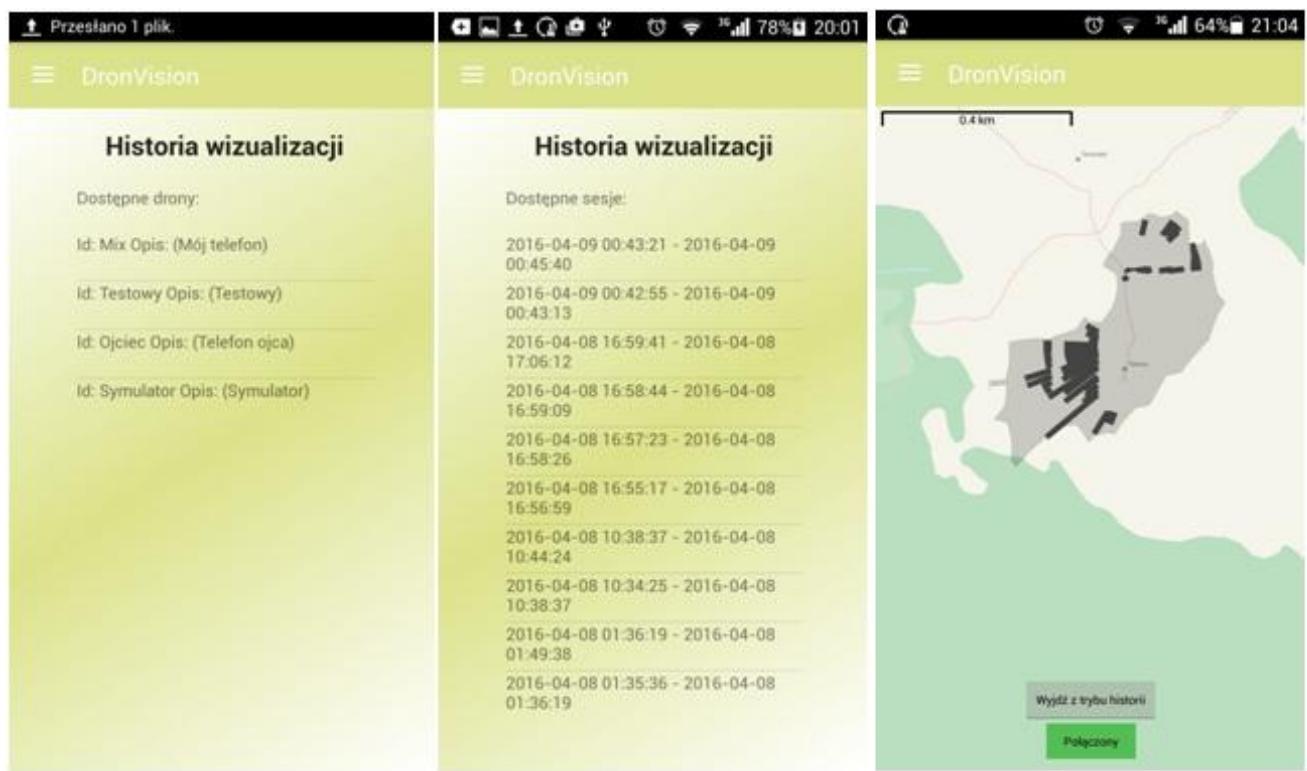
Ten widok pozwala użytkownikowi aplikacji na określenie preferencji dotyczących wizualizacji. Jest on widokiem przewijanym i składa się z trzech list, z których każda wyświetla te same drony, przypisane do użytkownika przez administratora systemu. Użytkownik ma możliwość określenia jednego drona, za którym będzie podążać kamera mapy w trakcie wizualizacji oraz od zera do maksymalnej liczby dostępnych dronów pokazanych na mapie oraz tych, dla których będzie rysowany obszar przeszukany.



Rys. 7.5 Aplikacja DronVision - widok preferencje wizualizacji

### 7.3.6. Widoki historii wizualizacji

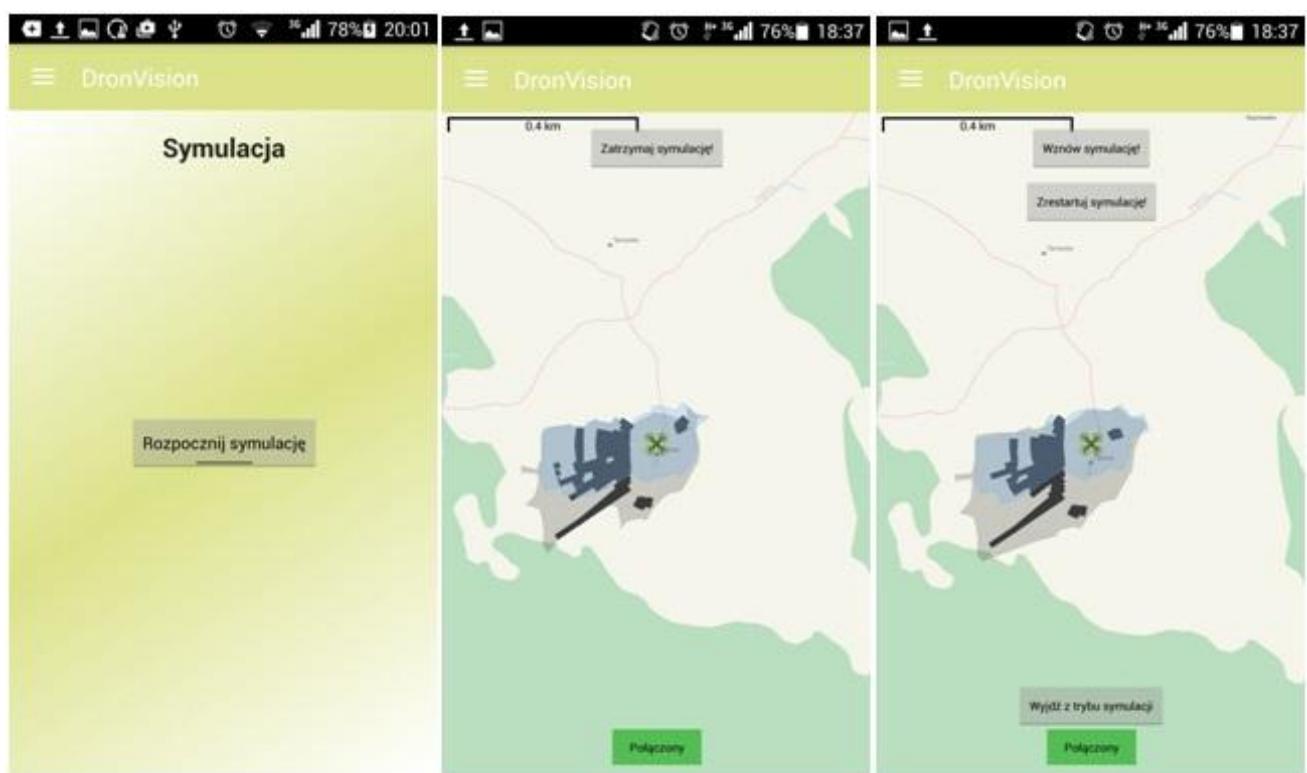
Aplikacja udostępnia możliwość przeglądania historii wizualizacji dla konkretnych dronów i ich sesji. Po wybraniu z menu tej funkcji zostanie wyświetlony widok z listą przypisanych do użytkownika dronów. Po wybraniu jednego z pojazdów, w kolejnym widoku użytkownik ma możliwość wybrania jednej z dziesięciu ostatnich sesji wizualizujących, poszeregowanych malejąco zgodnie z ich datą przeprowadzenia. Po wybraniu sesji wyświetlany jest widok mapy w trybie historii, wraz z narysowanym obszarem przeszukanym dla danej sesji. Powrót do wizualizacji bieżącej jest możliwy poprzez naciśnięcie przycisku wstecz telefonu lub programowego przycisku "Wyjdź z trybu historii".



7.6 Aplikacja DronVision - widoki historii wizualizacji

### 7.3.7. Widoki symulacji

Widok ten demonstruje działanie aplikacji na przykładzie drona znajdującego się nad najwyższym szczytem polskich Bieszczadów, czyli Tarnicą. Po uruchomieniu symulacji użytkownik w dowolnym momencie ma możliwość jej zatrzymania, a później ponownego wznowienia, zrestartowania lub po prostu zakończenia. Jeśli symulacja zostanie uruchomiona podczas trwania normalnej wizualizacji widok mapy zostanie przeniesiony nad Tarnicę, a rzeczywista wizualizacja zatrzymana. W momencie wyjścia z trybu symulacji aplikacja powróci do ostatniej znanej symulacji, z uwzględnieniem zmian położenia dronów powstałych w czasie trwania symulacji.



Rys. 7.7 Aplikacja DronVision - widoki symulacji

### **7.3.8. Widoki dodatkowe**

W ramach widoków dodatkowych, niezwiązańych bezpośrednio z funkcją wizualizacji obszaru przeszukanego, aplikacja zapewnia możliwość edycji ustawień konta takich jak adres email, imię, nazwisko, login, czy hasło oraz dostęp do widoków informacyjnych "O aplikacji" i "O autorze". Ze względu na ich niewielkie znaczenie dla merytorycznej części pracy, widoki te nie zostały zaprezentowane.

## **7.4. Testy**

Ze względu na szeroki zakres pracy oraz brak dostępu do rzeczywistego drona testy miały charakter wysoce ograniczony. Były to głównie testy funkcjonalne oraz integracyjne przeprowadzane z wykorzystaniem techniki czarnej skrzynki. System został przetestowany na kilku obszarach Warszawy w trybie pieszym oraz mobilnym. Wyniki testów były zadowalające, jednak pokazały, że niektóre elementy aplikacji wymagają optymalizacji wydajnościowej. Stwierdzono, iż może mieć to związek z wykorzystaniem darmowej wersji serwera aplikacyjnego Wildfly zamiast wersji komercyjnej oraz jego uruchomienia na prywatnym komputerze zamiast na dedykowanej ku temu maszynie wirtualnej.

Dodatkowo, przy wykorzystaniu techniki białej skrzynki, przeprowadzono testy wydajnościowe polegające na pomiarze czasu przetwarzania poszczególnych metod wywoływanych podczas wyznaczania obszaru przeszukanego. Testy te wykazały niską wydajność metody odpowiadającej za aktualizację dziur w obszarze dotychczas przeszukanym, co pozwoliło na wprowadzeniem poprawek, które skróciły czas przetwarzania niewydajnej metody.

Testy algorytmu przeprowadzono na danych symulacyjny na obszarze polskiej części Bieszczad. Stwierdzono, iż algorytm prawidłowo rozpoznaje wzniesienia i spadki terenu oraz uwzględnia obiekty zasłaniające kamerze część terenu. Testy wykazały również prawidłowe funkcjonowanie algorytmu odpowiadającego załączenie obszarów.

## **8. Podsumowanie**

Celem niniejszej pracy było stworzenie algorytmu umożliwiającego wizualizację obszaru przeszukanego przez bezzałogowe statki powietrzne, potocznie nazywane dronami, na podstawie zebranych danych geolokalizacyjnych obiektów i znajomości parametrów kamer oraz zaprojektowanie i implementacja systemu informatycznego realizującego tę funkcję.

W pierwszej części pracy zostały omówione aspekty teoretyczne związane z poruszana problematyką. Zakres tematyczny pracy okazał się bardzo szeroki i wiązał się ze zgłębiением takich tematów jak geodezyjne powierzchnie odniesienia, numeryczne modele terenu i sposoby ich pozyskiwania, zasada działania systemu nawigacji satelitarnej GPS oraz różne sposoby określania wysokości nad Ziemią. Dogłębne zapoznanie się z tymi pojęciami pozwoliło na zdobycie niezbędnej wiedzy do realizacji dalszych etapów pracy.

Po części opisującej dziedzinę problemu dokonano analizy i porównania technologii dostępnych na rynku związanych z projektowanym rozwiązaniem. Przeanalizowano rodzaje aplikacji, dostępne platformy i narzędzia pozwalające na tworzenie oprogramowania. Zdecydowano się na konkretne sposoby komunikacji między elementami systemu oraz wybrano format przesyłanych danych. W dalszej części analizy skoncentrowano się na zadaniach jakie projektowany system miał spełniać. Wynikiem tej części był logiczny model systemu opisujący sposób realizacji przez system postawionych wymagań, lecz abstrahujący od szczegółów implementacyjnych.

Kolejną fazą po analizie była faza projektowania systemu, która polegała na przekształceniu modelu analitycznego, czyli opisu systemu z punktu widzenia aktorów, zrozumiałego dla klienta, w model zawierający informację o wewnętrznej strukturze systemu, jego konfiguracji sprzętowej oraz sposobach jego realizacji. W efekcie w trakcie tego etapu powstały: modele struktury, opisujące statyczną budowę systemu, takie jak diagram komponentów, diagramy klas, czy diagram związków encji oraz modele zachowania, opisujące aspekty dynamiczne systemu takie jak: diagramy aktywności i sekwencji. W ramach etapu projektowania stworzono również projekt interfejsu graficznego dla aplikacji wizualizującej.

Aby umożliwić poprawne funkcjonowanie systemu koniecznym było opracowanie algorytmu umożliwiającego wyznaczanie obszaru przeszukanego. W tym celu w pierwszej kolejności dokonano analizy wymagań i założeń związanych z algorytmem, a następnie podzielono algorytm na trzy części: algorytm wyznaczania otoczki obszaru przeszukanego, algorytm wyznaczania obszarów wewnętrz otoczki, niezarejestrowanych przez kamerę oraz algorytm łączenia pojedynczych obszarów w całość. Pierwsze dwa algorytmy zostały opraco-

wane z wykorzystaniem zasad heurystyki, podczas gdy do implementacji algorytmu łączącego obszary wykorzystano naukową teorię kształtów alfa oraz algorytm wyznaczania otoczki alfa-wklęsłej.

Ostatnią częścią pracy była implementacja systemu oraz prezentacja rozwiązania stanowiąca opis stworzonego systemu, z uwzględnieniem podziału na podsystemy oraz przegląd widoków aplikacji klienckiej, służącej do wizualizacji obszaru przeszukanego. Tę część pracy można potraktować jako swoistą instrukcję korzystania z aplikacji DronVision.

W efekcie powyższych działań zrealizowano postawione na początku tej pracy założenia i stworzono narzędzie współpracujące z bezzałogowymi statkami powietrznymi. Praca pozwoliła autorowi na znaczne poszerzenie swojej wiedzy w zakresie informatyki, matematyki, inżynierii oprogramowania, algorytmiki, geomatyki, geofizyki i w wielu innych aspektach, które ciężko przyporządkować do jednej kategorii.

## **9. Bibliografia**

1. **Farr Tom [i inni]** The Shuttle Radar Topography Mission [Dziennik] // Reviews of Geophysics - AN AGU JOURNAL. - 2007. - 2 : Tom 45.
2. **Banachowicz Andrzej** Elementy Geometryczne Elipsoidy Ziemskiej W Prace Wydziału Nawigacyjnego Akademii Morskiej w Gdyni [Online] // am.gdynia.pl. - 2006. - [http://wn.am.gdynia.pl/pw/static/pdf.php?id\\_referat=277](http://wn.am.gdynia.pl/pw/static/pdf.php?id_referat=277). - s. 17.
3. **Gotlib Dariusz, Iwaniak Adam i Olszewski Robert** GIS - Obszary zastosowań [Książka]. - Warszawa : Wydawnictwo Naukowe PWN, 2007.
4. **Izdebski Waldemar** Numeryczny Model Terenu [Online] // <http://gisplay.pl/>. - <http://gisplay.pl/gis/numeryczny-model-terenu.html>.
5. **Kąciecki Dominik** Bezzalogowe Statki Powierzne Drony [Online] // Witryna Wyższej Szkoły Społeczno-Ekonomicznej. - <http://www.wsse-uczelnia.edu.pl/bezzalogowe-statki-powietrzne-drony>.
6. **Knippers Richard** Reference surfaces for mapping [Online] // kartoweb. - 08 2009. - <http://kartoweb.itc.nl/geometrics/Reference%20surfaces/refsurf.html>.
7. Shuttle Radar Topography mission [Online] // Wikipedia. - [https://pl.wikipedia.org/wiki/Shuttle\\_Radar\\_Topography\\_Mission](https://pl.wikipedia.org/wiki/Shuttle_Radar_Topography_Mission).
8. Numeryczny model terenu [Online] // Wikipedia. - [https://pl.wikipedia.org/wiki/Numeryczny\\_model\\_terenu](https://pl.wikipedia.org/wiki/Numeryczny_model_terenu).
9. **UNAVCO** The Geoid and Receiver Measurements [Online] // UNAVCO. - 10 12 2014. - <https://www.unavco.org/education/resources/educational-resources/tutorial/geoid-gps-receivers.html>.
10. **Uriasz Janusz** Powierzchnie odniesienia [Online] // uriasz.am. - [http://uriasz.am.szczecin.pl/naw\\_bezp/Powierzchnie.html](http://uriasz.am.szczecin.pl/naw_bezp/Powierzchnie.html).
11. **Witold Fraczek** Mean Sea Level, GPS and the Geoid [Online] // esri. - 2013. - <http://www.esri.com/news/arcuser/0703/geoid1of3.html>.
12. **Wójcik Krystian** Działanie GPS [Online] // TechnologiaGPS. - <http://www.technologiagps.org.pl/dzialanie-gps.htm>
13. Wysokościomierz barometryczny [Online] // Aircraft. - 18 02 2013. - <http://aircraft.cba.pl/?p=366>.
14. **iwierdza.net** Sztuczne satelity i prędkość kosmiczna [Online] // iwierdza. - 2006. - [http://www.iwiedza.net/materiały/astr\\_m029.html](http://www.iwiedza.net/materiały/astr_m029.html).

15. **Janusz Śledziński** Niwelacja GPS [Online] // Geoforum.pl. - <http://geoforum.pl/?menu=46813,46833,46921&link=gnss-krotki-wyklad-alfabet-gps-niwelacja-gps>.
16. **MF Avionics** Nawigacja w lotnictwie - metody i przykłady [Online] // MF Avionics. - <http://mfavionics.blogspot.com/2012/07/nawigacja-w-lotnictwie-metody-i-przykady.html>.
17. **Górka, Michał.** Technika satelitarna w geofizyce. [Online] 2003. [http://zasoby1.open.agh.edu.pl/dydaktyka/inzynieria\\_srodowiska/c\\_technika\\_satelitarna/gps.html](http://zasoby1.open.agh.edu.pl/dydaktyka/inzynieria_srodowiska/c_technika_satelitarna/gps.html).
18. **Krzaczyńska Maja.** Aplikacja mobilna, webowa, czy desktopowa? Co Wybrać? *EXACO Blog.* [Online] <https://blog.exaco.pl/aplikacja-mobilna-webowa-czy-desktopowa-co-wybrac/>.
19. **Nabożny Maciej.** Wstęp do Androida (SDK i NDK). *Maciej[CC].* [Online] <http://www.mnabozny.pl/blog/wstep-do-androida-sdk-i-ndk/>.
20. **Android NDK.** *Android Developers.* [Online] <http://developer.android.com/intl/ru/tools/sdk/ndk/index.html>.
21. **Apple Developer.** [Online] <https://developer.apple.com/support/comparermemberships/>
22. **Windows Mobile Developer.** [Online] <https://dev.windows.com>.
23. **Xamarin.** [Online] <https://xamarin.com/>.
24. **Marcin, Krzych.** Tworzenie Aplikacji Mobilnych - 3 możliwości: HTML5, natywna, czy hybryda? *KRZYMAR.NET.* [Online] <http://krzymar.net/index.php/2014/06/24/tworzenie-aplikacji-mobilnych-3-mozliwosci-html5-natywna-hybryda/>.
25. **Ilya, Gregorik.** *High Performance Browser Networking.* Sebastopol : O'Reilly Media, 2013.
26. **Podila, Pavan.** HTTP: The Protocol Every Web Developer Must Know - Part 1. *Envato Tuts+.* [Online] 08 04 2013. <http://code.tutsplus.com/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177>.
27. **Falkner, James.** Asynchronous Web Programming with HTML5 WebSockets and Java. *SlideShare.* [Online] 08 05 2013. <http://www.slideshare.net/schtool/asynchronous-web-programming-with-html5-websockets-and-java>.

28. **Krzywy, Edward.** Protokół WebSocket - Internet w czasie rzeczywistym. *CHIP.pl*. [Online] 03 01 2013. <http://www.chip.pl/artykuly/technika/2013/01/protokol-websocket-internet-w-czasie-rzeczywistym>.
29. **Ćmil, Michał, Matłoka, Michał i Marchioni, Francesco.** *Java EE 7 Development with WildFly*. Birmingham : PACKT, 2014.
30. **SOISK.** Model ISO/OSI i TCP/IP. *Systemy operacyjne i sieci komputerowe*. [Online] <http://www.soisk-me.pl/klasa-iv-sieci/model-iso-osi-i-tcp-ip?showall=&limitstart=>.
31. **Lindo, Sean.** XML vs. JSON - A Primer. *ProgrammableWeb*. [Online] 07 11 2013. <http://www.programmableweb.com/news/xml-vs.-json-primer/how-to/2013/11/07>.
32. **Yarpo, Patryk.** JSON - format wymiany danych. *Blog Webdeveloperski Patryk Yarpo*. [Online] 06 03 2011. <http://www.yarpo.pl/2011/03/06/json-jako-format-wymiany-danych/>.
33. **Jędrzejewski, Adam.** JSON - lekka alternatywa dla XML. *DAJP Programmers Group*. [Online] 10 10 2016. <http://www.dajp.org/blog/3-dajp/9-json.html>.
34. **Stowarzyszenie OpenStreetMap.** Portal polskiej społeczności OpenStreetMap. *OpenStreetMap Polska*. [Online] <http://openstreetmap.org.pl/>
35. WikiProject. [Online] [http://wiki.openstreetmap.org/wiki/Comparision\\_Google\\_services\\_-\\_OSM](http://wiki.openstreetmap.org/wiki/Comparision_Google_services_-_OSM).
36. **Wnuk, Paweł.** *Inżynieria Oprogramowania Preskrypt*. Warszawa : Politechnika Warszawska, 2011.
37. **Muszyńska Karolina.** Wprowadzenie do analizy systemów informacyjnych. [Online] [iiwz.wneiz.pl/karolina/Pliki/wstepA.doc](http://iiwz.wneiz.pl/karolina/Pliki/wstepA.doc).
38. Diagram komponentów. *Projektowanie Systemów Komputerowych notatki w internecie AGH*. [Online] <http://brasil.cel.agh.edu.pl/~09sbfraczek/diagram-komponentow,1,17.html>.
39. System wizyjny. *Wikipedia*. [Online] [https://pl.wikipedia.org/wiki/System\\_wizyjny](https://pl.wikipedia.org/wiki/System_wizyjny).
40. Geolokalizacja. *Wikipedia*. [Online] <https://pl.wikipedia.org/wiki/Geolokalizacja>.
41. Triangulacja. *Wikipedia*. [Online] [https://pl.wikipedia.org/wiki/Triangulacja\\_\(matematyka\)](https://pl.wikipedia.org/wiki/Triangulacja_(matematyka))
42. Triangulacja Delaunay. *Wikipedia*. [Online] [https://pl.wikipedia.org/wiki/Triangulacja\\_Delone](https://pl.wikipedia.org/wiki/Triangulacja_Delone)
43. **Aitchison, Alastair.** Alpha Shapes and Concave Hulls. [Online] 2011. <https://alastaira.wordpress.com/2011/03/22/alpha-shapes-and-concave-hulls/>.

44. **Grosso, Eric.** Concave hull based on JTS. [Online]  
[http://www.rotefabrik.free.fr/concave\\_hull/](http://www.rotefabrik.free.fr/concave_hull/).
45. Alpha shape. *Wikipedia*. [Online] [https://en.wikipedia.org/wiki/Alpha\\_shape](https://en.wikipedia.org/wiki/Alpha_shape)
46. **Szawdyński, Piotr.** Klasy problemów NP. *Serwis programistyczny C/C++*. [Online]  
<http://cpp0x.pl/kursy/Teoria-w-Informatyce/Zlozonosc-obliczeniowa/Klasy-problemow-NP/430>.
47. **Asaeedi, Saeed, Didehvar, Farzad i Mohades, Ali.** Alpha-Concave Hull, a Generalization of Convex Hull. *Cornell University Library*. [Online]  
<https://arxiv.org/ftp/arxiv/papers/1309/1309.7829.pdf>.
48. **Galton, Antony i Duckham, Matt.** What is the region occupied by a set of points? *Geographic Information Science*. 2006, Tom Springer, strony 81-98.
49. **Moreira, Adriano i Yasmina Santos, Maribel.** *Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points*. 2007.
50. **Marjan, Celikik.** Aplha shapes. *Max Planck Institute for Informatics*. [Online]  
<http://people.mpi-inf.mpg.de/~jgiesen/tch/sem06/Celikik.pdf>.
51. **Czekaj, Wiesław, Idus, Tomasz i Bogacz, Marcin.** Geometria obliczeniowa. *riad.pl.edu.pl*. [Online]  
[http://riad.pk.edu.pl/~zk/GO\\_LAB\\_00/triangulacja/ram\\_v.htm](http://riad.pk.edu.pl/~zk/GO_LAB_00/triangulacja/ram_v.htm).

## **10. Lista załączników**

1. **Załącznik nr 1** - Diagram klas części serwerowej
2. **Załącznik nr 2** - Diagram klas aplikacji klienckiej DronVision
3. **Załącznik nr 3** - Projekt interfejsu graficznego aplikacji DronVision
4. **Załącznik nr 4** - Diagram komponentów
5. **Załącznik nr 5** - Diagram związków encji
6. **Załącznik nr 6** - Diagram aktywności - Dron - Zmień położenie
7. **Załącznik nr 7** - Diagram aktywności - Użytkownik aplikacji wizualizującej - Oglądaj wizualizację
8. **Załącznik nr 8** - Diagram aktywności - Użytkownik aplikacji wizualizującej - Edytuj preferencje
9. **Załącznik nr 9** - Diagram aktywności - Użytkownik aplikacji wizualizującej - Przeglądaj historię wizualizacji
10. **Załącznik nr 10** - Diagram aktywności - Użytkownik aplikacji wizualizującej - Oglądaj symulację
11. **Załącznik nr 11** - Diagram sekwencji - DronTracker - Zmiana położenia
12. **Załącznik nr 12** - Diagram sekwencji - DronServer - Nowa wiadomość od DronTracker'a
13. **Załącznik nr 13** - Diagram sekwencji - DronVision - Nowa wiadomość wizualizująca