НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ АЕРОКОСМІЧНИХ ТЕХНОЛОГІЙ
(повна назва інституту/факультету)

Кафедра систем керування літальними апаратами
(повна назва кафедри)

«На правах рукопису»
УДК 004.021

«До захисту допущено»

Завідувач кафедри
_____ Олександр ЗБРУЦЬКИЙ
(підпис) (власне ім'я, прізвище)

"___" _____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Системи керування літальними апаратамита комплексами»
(назва)

зі спеціальності 173 «Авіоніка»
(код та назва спеціальності)

на тему:

Система автоматичного керування групою мультикоптерів на основі оптичного розпізнавання та радіо позиціювання

Виконав: студент _II_ курсу, групи ___АС-з91мп___
(шифр групи)

Забродський Антон Віталійович
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник

завідувач кафедри СКЛА д.т.н., професор Збруцький О. В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент

професор кафедри АРБ д.т.н. професор Сухов В. В.
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут/факультет аерокосмічних технологій
(повна назва)

Кафедра систем керування літальними апаратами
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність 173 «Авіоніка»
(код і назва)

Освітньо-професійна програма Системи керування літальними апаратами та комплексами)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр ЗБРУЦЬКИЙ
(підпис) (власне ім'я, прізвище)

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Забродському Антону Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Система автоматичного керування групою мультикоптерів на основі оптичного розпізнавання та радіо позиціювання
науковий керівник дисертації Збруцький Олександр Василійович, д.т.н, проф,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «04» грудня 2020 р. № 3446-с

2. Термін подання студентом дисертації 17 грудня 2020 року

3. Об'єкт дослідження є процес польоту групи мультикоптерів, в якому використовуються оптичне розпізнавання та радіопозиціонування.

4. Предмет дослідження система автоматичного керування, що забезпечує рух групи мультикоптерів, а також методи та моделі, що є частиною такої систем.

5. Перелік завдань, які потрібно розробити необхідно розробити програмне забезпечення, за допомогою якого можна здійснювати моделювання руху групи мультикоптерів, причому з урахуванням зовнішніх впливів на групу різного.

6. Перелік графічного (ілюстративного) матеріалу
1) Плакат з актуальністю досліджень – 1л. формату А-1.
2) Плакат з метою та задачею роботи – 1л. формату А-1.

3) Плакат з характеристикою роботи 1л. формату А-1.
4) Плакат з біонічними аналогіями – 1л. формату А-1.
5) Плакат з математичними моделями– 3л. ф. А-1.
6) Плакат з прийнятими проектними рішеннями – 1л. ф. А-1.
7) Плакат з отриманням координат мультикоптерів – 1л. ф. А-1.
8) Плакат з реалізацією та тестуванням системи – 1л. ф. А–1.

7. Орієнтовний перелік публікацій

1) Стаття у фаховому виданні за результатами досліджень.
2) Доповіді на науково-технічних конференціях за темою досліджень.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання 24 вересня 2019 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1.	Аналіз особливостей проблеми організації групового польоту, та огляд літератури.	25/09/2019 30/11/2019	
2.	Особливості вирішення задачі слідування заданою траєкторією у випадку групового польоту мультикоптерів	25/09/2019 30/11/2019	
3.	Вирішення задачі перегруповування мультикоптерів при виникненні мотивуючих факторів	02/12/2019 30/01/2020	
4.	Обґрунтування проектних рішень при вирішенні задачі керування груповим польотом мулькоптерів	02/12/2019 28/02/2020	
5.	Реалізація системи автоматичного керування груповим польотом мультикоптерів на основі використання СКС.	02/03/2020 29/05/2020	
6.	Економічне обґрунтування розробки системи автоматичного керування груповим польотом мультикоптерів на основі використання оптичного розпізнавання та радіо позиціювання	01/06/2020 02/10/2020	
7.	Оформлення магістерської дисертації, та опублікування результатів досліджень у фаховому виданні.	01/09/2020 30/12/2020	
8.	Захист магістерської дисертації	22/12/2020	

Студент _____ Антон ЗАБРОДСЬКИЙ
(підпис) (власне ім'я, прізвище)

Науковий керівник дисертації _____ Олександр ЗБРУЦЬКИЙ
(підпис) (власне ім'я, прізвище)

РЕФЕРАТ

Розроблена система автоматичного керування групи мультикоптерів на основі оптичного розпізнавання та радіо позиціювання, яка забезпечує можливості безпечного та результативного руху групи безпілотних літальних апаратів (БПЛА). Створена математична модель руху групи БПЛА з врахуванням обмежень, що накладаються на групу та умови руху. Розроблене програмне забезпечення та проведене моделювання групового руху з урахуванням дії зовнішніх збуруючих факторів, яке показало достовірність та ефективність розробленої системи.

Ключові слова: мультикоптер, безпілотний, радіопозиціонування, система керування, біоніка, оптичне розпізнавання.

Робота складається з 75 сторінок, у ході дослідження було опрацьовано велику кількість джерел та літератури, завдяки чому вдалося виконати поставлену мету роботи, а саме розробку системи автоматичного керування групи мультикоптерів на основі оптичного розпізнавання та радіопозиціювання. Створена система створює можливості для безпечного та результативного руху групи мультикоптерів, чого важко добитися при використанні праці численної команди людей-операторів.

Створені підходи можуть застосовуватися при побудові систем керування рухом групи літальних апаратів. Розроблена математична модель руху групи БПЛА та програмне забезпечення, які показала ефективність запропонованих рішень.

Розробка може розвиватися шляхом залучення більш витончених алгоритмів ухилення від перехоплювачів, а також оминання просторових перешкод, що планується зробити у майбутньому.

REFERAT

The paper considers issues related to finding opportunities for safe and efficient movement of the multicopter group. The object of this study is the flight process of a group of multicopters, which uses optical recognition and radio positioning, while the subject of the study is an automatic control system that ensures the movement of a group of multicopters, as well as methods and models that are part of such a system.

Key words: multicopter, unmanned, radio positioning, control system, bionics, optical recognition.

The work consists of 75 pages, during the study a large number of sources and literature were processed, thanks to which the goal of the work was achieved, namely the development of an automatic control system for a group of multicopters based on optical recognition and radio positioning. The created system creates opportunities for safe and efficient movement of a group of multicopters, which is difficult to achieve when using the work of a large team of people-operators.

The created approaches can be applied at construction of control systems of movement of group of aircraft. A mathematical model of UAV group movement and software was developed, which showed the effectiveness of the proposed solutions.

Development can be developed by involving more sophisticated interceptor avoidance algorithms, as well as bypassing spatial obstacles that are planned to be done in the future.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ4

GUI	Graphics User Interface
PC	Personal Computer
WWW	World Wide Web
БД	База даних
БПЛА	Безпілотний літальний апарат
ЗІ	Захист інформації
ІТ	Інформаційні технології
ООП	Об’єктно-орієнтоване програмування
ПЗ	Програмне забезпечення
ПК	Персональний комп’ютер
СКС	Спільне комунікаційне середовище
СУБД	Система управління базами даних

ВСТУП

Розвиток сучасної мікроелектроніки із її постійною все вищою мініатюризацією, а також галузі електромеханіки в частині створення мікродвигунів, призвели до того, що на сьогоднішній день на часі вже створення безпілотних мультикоптерних літальних апаратів (далі – БПЛА) вагою до 100 г та габаритними розмірами до 10 см. Очевидно, такі апарати не можуть нести на борту якихось складних пристроїв, отже наділені порівняно простою функціональністю, та, зазвичай, орієнтовані на виконання лише однієї певної функціональної задачі. У наявних на сьогоднішній день умовах найменші БПЛА призначені у переважній більшості для задач спостереження та передачі оперативної інформації.

В цілому, слід відмітити, що на даний час, коли переважна більшість існуючих моделей БПЛА працюють під управлінням людини-оператора (особливо малих розмірів, які не можуть нести на борту важку та габаритну обчислювально потужну електроніку, необхідну для вирішення задач керування роботою апарату у повністю автономному режимі польоту), задача керування груповим польотом пристроїв не є досить поширеною. Однак, очевидно, що уже на одному з наступних етапів розвитку безпілотних літальних апаратів виникатиме потреба у керуванні ними в режимі масового польоту, коли кількість їх наближатиметься до десятків, і навіть сотень одиниць. Біонічні спостереження за зграями птахів та косяками рибин свідчать про надзвичайну важливість постійного відстеження положення інших учасників руху групи (причому якомога більшої їх кількості, а не лише найближчих з них). Відповідно і для групи БПЛА **актуальною є задача** створення системи автоматичного керування (далі – САК) нею, основаної на розпізнаванні інших учасників групи та їх взаємному позиціонуванні, що можна виконувати за допомогою двох підходів: оптичного розпізнавання та радіопозиціонування.

Метою роботи при цьому є створення можливостей для безпечного та результативного руху групи БПЛА. Для досягнення цієї мети необхідно проробити наступні задачі дослідження:

- здійснити аналіз проблеми організації руху групи БПЛА: встановити існуючі науково-технічні рішення, а також схожі природні системи з точки зору біоніки, виділити основні труднощі такого процесу та невирішені раніше питання;

- розробити надійні методи та підходи до оптичного розпізнавання та радіопозиціонування членів групи БПЛА;

- створити математичну модель процесів групового руху БПЛА, спроектувати відповідні алгоритми та структури даних;

- обґрунтувати вибір адекватних засобів розробки та здійснити програмну реалізацію створених математичних моделей;

- провести моделювання групового руху БПЛА, причому з урахуванням різних варіантів зовнішньої дії на групу: ухилення від точкового перехоплювача, прольот групи крізь отвір невеликого розміру у перешкоді, ухилення від просторової перешкоди, тощо;

- проаналізувати отримані результати, зробити висновки по роботі та окреслити перспективи подальших досліджень.

Об’єктом дослідження є процес польоту групи БПЛА, в якому використовуються оптичне розпізнавання та радіопозиціонування.

Предметом дослідження є система автоматичного керування, що забезпечує рух групи БПЛА, а також методи та моделі, що є частиною такої системи.

В роботі застосовано **методи** аналізу та синтезу, методи математичного аналізу та моделювання, технології програмування.

Наукова новизна:

1. Розроблено систему автоматичного керування та моделі руху групи БПЛА, побудованих на основі оптичного розпізнавання та

радіопозиціонування, що дозволяє покращити безпеку руху всієї групи за рахунок дотримання безпечних відстаней між її елементами.

2. На основі проведеного аналізу геометричних та фізичних параметрів процесу вироблено правила (у вигляді математичних формул) безпечного руху групи апаратів.

3. На основі аналізу мурмурації птахів запропоновано нову методику ухилення елементів групи БПЛА від точкового перехоплювача, що дозволяє підвищити ефективність руху групи в умовах протидії супротивника.

Практичне значення роботи полягає у розробці програмного забезпечення, за допомогою якого можна здійснювати моделювання руху групи БПЛА, причому з урахуванням зовнішніх впливів на групу різного типу.

В **перспективі** модель і програмний продукт можуть розширюватися шляхом залучення інших зовнішніх впливів, окрім тих, що уже розглянуті у даній роботі. Також можна розглянути процеси об’єднання двох (чи більшої кількості) груп, а також розбиття однієї великої групи на дві (або декілька) груп поменше.

1 АНАЛІЗ ОСОБЛИВОСТЕЙ ПРОБЛЕМИ ОРГАНІЗАЦІЇ ГРУПОВОГО ПОЛЬОТУ БПЛА

1.1. Задачі, що приводять до необхідності організації групових польотів БПЛА

Перед розв'язуванням завдань, окреслених для даної роботи вище, слід встановити коло задач, що можуть потребувати групового режиму руху безпілотних літальних апаратів, який розглядається, оскільки в подальшому особливості таких задач можуть, відповідно, визначати особливості проектованих моделей, алгоритмів, підходів, тощо.

В першу чергу, задача руху саме групи пристроїв за одним (в цілому) спільним маршрутом може бути обумовлена простим бажанням відповідальної за операцію особи підвищити надійність її успішного виконання. Відомим фактом із теорії надійності є те, що при збільшенні у N разів числа ідентичних елементів, які призначені для виконання однієї і тієї ж самої функції і дублюють одне одного, у стільки ж разів підвищується надійність системи [1]. Розумна кратність дублювання будь-якого технічного пристрою завжди має верхню межу на рівні 3-5 елементів, що з одного боку і не дуже багато (при озвучених вище цифрах у десятки членів групи БПЛА), а з іншого – і в такій групі також можливі зіткнення, що, очевидно, вкрай негативно позначатиметься на надійності успішного виконання поставленої функціональної задачі. Таким чином, слід занотувати, що доцільною є і невелика кількість апаратів у групі (як уже сказано вище, до 3-5, адже дублювання якогось технічного пристрою хоча б 10-20 ідентичними ланками видається абсолютно неймовірним із економічних міркувань: при такій необхідності слід було би підвищувати надійність одного пристрою, а не вводити 20 дублюючих елементів).

Відмітимо, що у цьому випадку усі члени групи виконують одну функціональну задачу, наприклад, передають (або потенційно можуть передавати, а реально передачу веде лише один апарат) відеосигнал, або

несуть зброю для ураження віддаленої цілі (якщо ураження цілі відбулося першим апаратом, то інші можуть економити боєзапас і повертатися на базу заповненими, або також здійснювати вражаючу дію – залежно від конкретної задачі, запрограмованої особою, відповідальною за проведення операції). Таким чином, мова іде про точне повторення (копіювання) дій першого апарату другим, якщо першому не вдалося їх успішно реалізувати, потім третім – у випадку невдачі другого, і т.д.

У наступному випадку, що потребує використання цілої групи БПЛА, елементи групи можуть працювати для досягнення спільної мети (наприклад, ураження одного віддаленого об'єкту, просторові розміри якого на два-три порядки більші за розміри БПЛА, чи організації спостереження з різних боків за таким об'єктом), але виконувати різні тактичні задачі. При цьому чисельність групи може бути досить великою, вона деякий час може рухатися до віддаленої цілі, а при її досягненні – розосередитися навколо об'єкту за певними правилами. Наприклад, найпростішим з алгоритмічної точки зору, але при цьому таким, що може бути корисним для практики, є розміщення як завгодно великої кількості апаратів навколо об'єкта рівномірно по поверхні сфери (або верхньої півсфери), що забезпечує можливість спостереження або атаки цього об'єкта рівномірно з усіх боків.

Нарешті, кожен апарат групи може працювати зі своєю власною метою, які можуть корелювати одна з іншою, а може і ні (зв'язок очевидно може бути наявним на вищих рівнях абстракції, недоступних для програмного забезпечення групи). В той же час переміщення апаратів протягом довгих дистанцій (від бази до цільової місцевості) може відбуватися в межах спільного геометрично вузького коридору, де можливі зіткнення та заважання апаратів один одному за умови наближення їх один до одного ближче певної критичної відстані δ_1, про яку детальніше будемо говорити нижче. Очевидно, у цьому варіанті також керування групою БПЛА є не тільки доцільним, а й життєво необхідним для виконання пристроями поставлених перед ними задач.

1.2. Якісний та кількісний аналіз труднощів організації групового польоту БПЛА

1.2.1. Складність організації автономного польоту одного БПЛА по заданій траєкторії.

При аналізі проблеми групового руху слід пам’ятати, що, на відміну від одиночного польоту, кожен апарат керується виключно за допомогою САК, а не за допомогою людини-оператора.

Робота БПЛА у повністю автономному режимі сама по собі являє собою надзвичайно складний для керування процес, в якому слід ураховувати велику кількість факторів, багато з яких до того ж складно піддаються формалізації, тобто опису у термінах математики. У нульовому наближенні можна сказати, що апарат має слідувати за певною траєкторією у тривимірному просторі, яку технічно простіше всього задати у параметричній формі:

$$\begin{cases} x = x(t) \\ y = y(t), \\ z = z(t) \end{cases} \tag{1.1}$$

де x, y, z – декартові координати якоїсь характерної точки БПЛА, за яку доцільно взяти центр мас (ЦМ) апарату (в подальшому ці координати відноситимуться саме до ЦМ, якщо не буде вказано інше);

t – час, який на практиці буде дискретним і, отже, являє собою послідовність окремих моментів (але для спрощення зовнішнього вигляду моделі в подальшому не будемо акцентувати на цьому увагу).

З курсу аналітичної геометрії відомо, що спосіб завдання просторової лінії (1.1) не є єдино можливим, а часто лінії задаються, наприклад, за допомогою перетину двох поверхонь та ін. Але варіант (1.1) точно є найзручнішим для практики, адже він дозволяє використовувати не лише аналітичні розв’язки (яких важко досягти при вирішенні реальних

практичних задач), а й чисельні, які найчастіше і є результатом у більшості випадків моделювання.

Очевидно, що реальний процес польоту об'єкта є набагато складнішим за «просте» слідування заданою траєкторією, адже його весь час супроводжують випадкові збурюючі фактори, причому, наприклад один з головних із них – вітрові впливи – може, по-перше, вносити суттєві зміни у рух апарату, а, по-друге, не є малоймовірною подією, якою можна знехтувати (а навпаки, такі впливи зустрічаються повсюдно майже при будь-яких польотах БПЛА). Очевидно, усі такі дії мають відстежуватися та компенсуватися відповідною САК та виконуючими пристроями апарату. Однак, слід відмітити, що метою даної роботи є дослідження процесу керування групою БПЛА, а не одного з них (чому присвячені інші науково-технічні дослідження, наприклад [2]), тому в подальшому викладі будемо вважати, що рух одного апарату заданою траєкторією виду (1.1) уже забезпечено, а розробці підлягає та частина системи керування, що відповідає саме за організацію руху групи пристроїв.

Отже, першою (загальною) складністю, яка відноситься власне до руху будь-якого (одного, чи у складі групи) повністю автономного БПЛА, є забезпечення безпеки та адекватності процесу слідування заданою траєкторією. Для оцінки ступеня небезпеки при даному процесі можна використовувати згадану раніше відстань δ_1 від самої траєкторії до найближчої точки зовнішньої перешкоди (ця ж величина буде використовуватися пізніше і для оцінки інших небезпек). Очевидно, що для різних точок траєкторії (кожній з яких, відповідно до (1.1) відповідає певний момент часу t) дана відстань δ_1 буде різною, тобто можна говорити про функцію $\delta_1(t)$. Умовою безпечного руху тоді буде відсутність виходу цієї функції за певне граничне безпечне значення Δ_1 (в меншу сторону) під час всього часу руху апарату:

$$\delta_1(t) - r_a > \Delta_1 \ \forall t, \qquad (1.2)$$

де враховано, що насправді безпечна відстань Δ_1 має відраховуватися не від точок самої траєкторії, а від крайніх точок власне апарату, для урахування чого слід ввести максимальний радіус апарату r_a, тобто відстань від його ЦМ до найбільш віддаленої від ЦМ точки БПЛА – рис. 1.1.

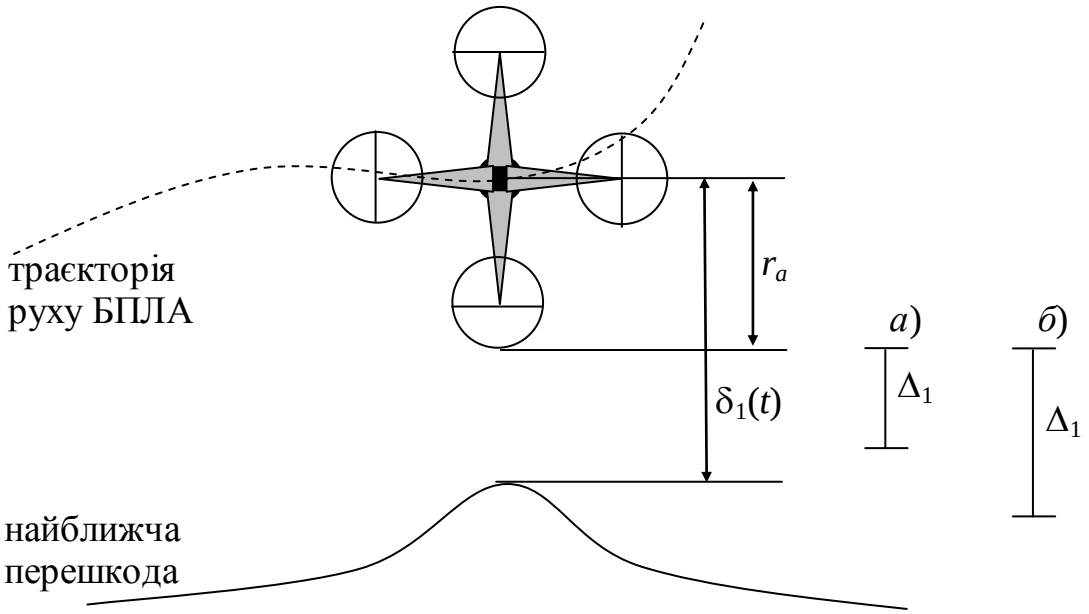


Рис. 1.1. Варіант визначення безпеки польоту БПЛА при наявному у даний момент часу значенні $\delta_1(t)$, а також максимальному радіусі апарату r_a:
а) при даному значенні Δ_1 виконується (1.2), тому такий політ є безпечним;
б) при даному Δ_1 порушується (1.2), тому політ небезпечний.

1.2.1. Аналіз труднощів організації руху групи БПЛА по заданій траєкторії.

Іншою, специфічною складністю є необхідність забезпечення руху по заданій траєкторії уже не одного, а рою апаратів, завдяки чому виникають дві загрози, які розглянуто нижче.

Першою з них є можливість зіткнення апаратів один з одним при русі в межах групи. Для уникнення цього слід задатися мінімальною безпечною відстанню Δ_3 (індекс «3» зручно відповідає випадку зіткнення апаратів між собою) між двома апаратами, яка характеризується наступними властивостями. Якщо поточна відстань $\delta_{ij}(t)$ між ЦМ *i*-того та ЦМ *j*-того

апарату більше значення Δ_3, то імовірність зіткнення цих БПЛА є близькою до нуля, а якщо менше, ця імовірність різко зростає:

$$P(A_{ij}) \approx \begin{cases} 0, \delta_{ij}(t) > \Delta_3 \\ 1, \delta_{ij}(t) < \Delta_3 \end{cases}, \quad (1.3)$$

де A_{ij} – подія, що відповідає зіткненню *i*-того та *j*-того апаратів.

Зрозуміло, що імовірність є неперервною величиною, однак перехід від неймовірного значення 0 до достовірного значення 1 є досить різким (розміщується поблизу точок, де виконується умова $\delta_{ij}(t) \approx \Delta_3$), тому ним на практиці можна знехтувати для спрощення моделі, яке є допустимим, оскільки не призводить до суттєвих втрат точності.

Щодо вибору конкретних значень величини Δ_3 слід зауважити, що вони мають суттєво перевищувати суму характерних розмірів самих апаратів $2r_a$ через наступні причини:

- по-перше, як уже зауважено вище, під час польоту весь час діють випадкові фактори (пориви вітру, наприклад), що можуть спричинювати різке відхилення апаратів від плавної траєкторії, тому для уникнення зіткнень між БПЛА має бути певний запас відстані;

- по-друге, як буде зазначено нижче (і весь час пізніше це буде використовуватися в роботі), апарати у групі в реальних умовах не можуть рухатися «у замороженому стані», тобто на незмінних відстанях один від одного. Насправді група має бути динамічною, форма якої може мінятися, як і, відповідно, взаємні положення та відстані між різними апаратами. Зважаючи на це, кожному апарату потрібний певний додатковий вільний простір навколо нього для забезпечення можливості починання безпечних маневрувань (адже сусідні апарати почнуть реагувати на зсуви даного апарату лише через деякий час, обумовлений інерційністю датчиків, САК та виконуючих пристроїв).

Для визначення оптимальних значень Δ_3 можна провести спеціальне дослідження, але у даній роботі обмежимося введенням подвійного запасу необхідної відстані і приймемо:

$$\Delta_3 = 4r_a \,. \tag{1.4}$$

Тоді умова безпеки в частині відсутності парних зіткнень між усіма членами групи буде мати вид:

$$\min_{i,\,j} \delta_{ij}(t) > 4r_a \,. \tag{1.5}$$

Другою загрозою, характерною саме для групового польоту, є зіткнення з іншими, сторонніми об’єктами, які взагалі-то безпечно оминаються самою траєкторією (1.1), тобто при русі нею одного апарату колізія була би неможливою, але, при русі цією траєкторією цілої групи, окремі апарати (що летять на її границях) можуть здійснити зіткнення з близькими перешкодами чи стінками отворів, за умови, що відстань від траєкторії до стороннього об’єкта $\delta_1(t)$ менше середнього радіуса групи r_g – рис. 1.2.

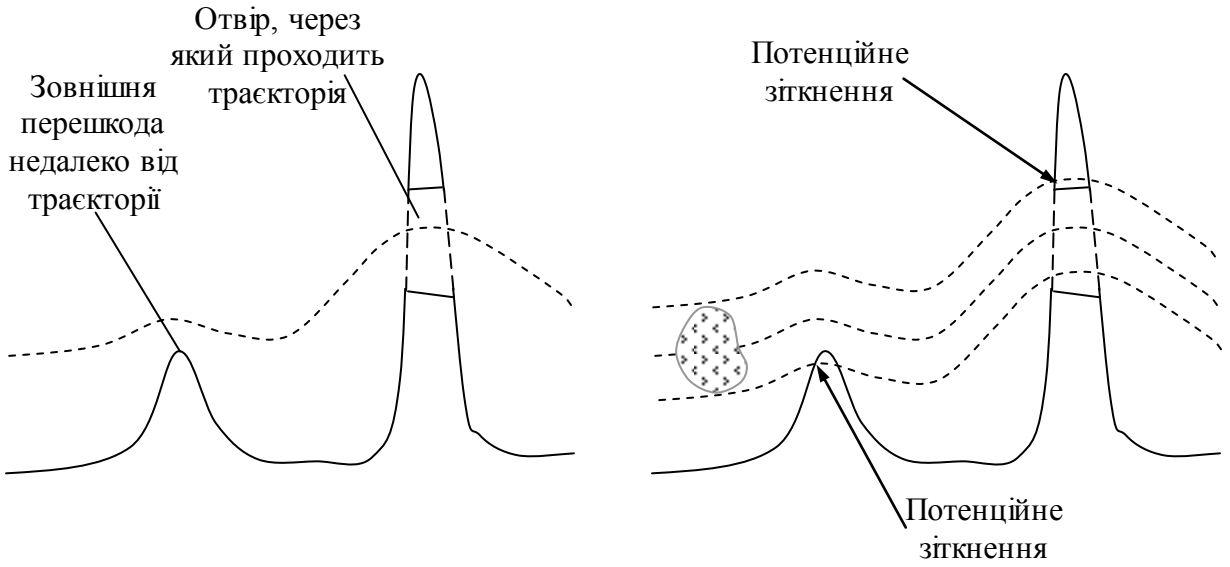


Рис. 1.2. Проходження літальних пристроїв по заданій траєкторії:
а – безпечний рух одного апарату; *б* – при проходженні групи біля перешкод чи через отвори, відстані до яких від траєкторії менше (або порівнянні) із

середнім радіусом групи, можливі зіткнення (які відбудуться, якщо не застосувати спеціальних заходів, наприклад, перешикування групи).

Таким чином, для нейтралізації другої із озвучених загроз при польоті групи, слід задатися безпечною допустимою відстанню Δ_2 між групою та найближчою до групи точкою стороннього об'єкту. Умовою безпечного руху групи тоді буде рівність, аналогічна (1.2), але в якій використовується характерний радіус групи, а не одного апарату:

$$\delta_1(t) - r_g > \Delta_2 \ \forall t, \qquad (1.6)$$

Тут, як і у (1.2), враховано, що насправді безпечна відстань Δ_2 має відраховуватися не від точок самої траєкторії, а від крайніх точок власне групи, для урахування чого і введено радіус групи r_g – рис. 1.3.

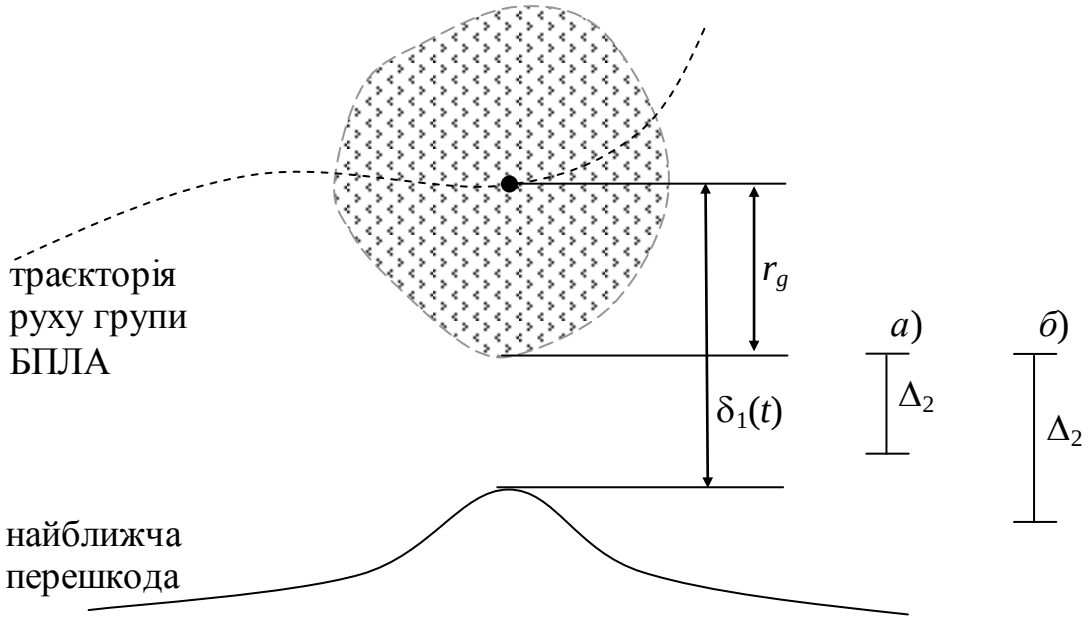


Рис. 1.3. Варіант визначення безпеки польоту групи БПЛА при наявному у даний момент часу значенні $\delta_1(t)$, а також максимальному радіусі групи r_g: а) при даному значенні Δ_2 виконується (1.6), тому такий політ є безпечним; б) при даному Δ_2 порушується (1.6), тому політ небезпечний.

Величина r_g, на відміну від радіусу апарату r_a, не є постійною величиною, а може (і повинна – відповідно до зовнішніх умов) змінюватися під час реального польоту групи по траєкторії. Можна запропонувати просту процедуру по визначенню r_g:

$$r_g = \max_n |MC| + r_a, \tag{1.7}$$

де MC – відстань між точкою $M(x; y; z)$, яка належить траєкторії і є поточним центром усієї групи, а точка $C(x_n; y_n; z_n)$ є поточним ЦМ n-ого апарату.

Таким чином, у формулі (1.7) обирається найбільш віддалений від поточного центру групи (що завжди лежить на траєкторії руху) у даний момент часу апарат, на основі положення якого і визначається радіус групи, з тією лише поправкою, що до відстані MC додається ще радіус самого апарату (адже MC визначається до центра мас крайнього апарату, тобто його частина при цьому виходить за межі сфери радіусом MC, тому до цього значення слід додати ще радіус самого БПЛА).

Оскільки r_g не є константою, то рівність (1.6) можна уточнити:

$$\delta_1(t) - r_g(t) > \Delta_2 \ \forall t. \tag{1.6а}$$

Якщо (1.6а) на якомусь черговому кроці не виконується, то це є сигналом для початку невідкладного перешикування всієї групи: зменшення її поперечних розмірів та відповідно витягування поздовжніх або збільшення густини розміщення апаратів у групі, якщо (або і витягування і збільшення густини). З іншого боку, якщо ліва частина (1.6а) значно перевищує праву, це є сигналом до оптимізації форми групи, та можливості перешикування в обернений бік (аби наблизити форму групи до кулеподібної).

Ще однією специфічною загрозою, яка виникає в груповому режимі польоту БПЛА є підвищення імовірності втрати одного апарату в умовах атаки ворожого перехоплювача при збільшенні загального числа апаратів у групі. При русі вздовж траєкторії тільки одного БПЛА на нього може бути здійснено напад ворожим перехоплювачем і при цьому існує певна

імовірність $P(B_1)$ успішного перехоплення цілі. Коли по траєкторії рухається два апарати (мінімально можлива група), то також можна розглянути таку характеристику, як імовірність ураження одного апарату $P(B_2)$, так само, як і в загальному випадку при русі N апаратів, можна розглядати імовірність ураження одного з них на рівні $P(B_N)$. Задача забезпечення якісного керування рухом групи БПЛА полягає у додержанні співвідношень:

$$P(B_1) \geq P(B_2),\ P(B_1) \geq P(B_3),\ \ldots,\ P(B_1) \geq P(B_N),$$

або одним словом, при фіксованій кількості N членів групи (у конкретному вильоті) має бути витриманим співвідношення:

$$P(B_N) \leq P(B_1), \qquad (1.8)$$

де B_1 – подія, що відповідає успішному ураженню супротивником БПЛА, який сам один рухається визначеною траєкторією;

B_N – подія, що відповідає успішному ураженню супротивником одного даного конкретного БПЛА (наприклад, апарату під номером «1»), при тому, що він є членом групи із N пристроїв всього (при цьому вважаємо, що імовірність ураження апарату №2 така ж сама за величиною, і апарату №3 також, і для всіх інших апаратів цієї групи із рівно N елементів імовірність ураження також складає $P(B_N)$).

Досягти такого ефекту (зменшення імовірності ураження одного апарату при збільшенні кількості членів групи) можна завдяки відповідній реакції групи в цілому на наближення перехоплювача. Таку поведінку демонструють птахи та риби при колективній протидії хижакові, якому стає важче вхопити якого-небудь члена групи при збільшенні їх загальної кількості, що докладніше буде досліджуватися нижче.

Таким чином, тут розглянуто основні проблеми, що слід вирішити при організації групового польоту на базі існуючої можливості проведення польоту для одиничного пристрою.

1.3. Дослідження особливостей організації групового просторового руху у біологічних системах за принципами біоніки

Як було зазначено у попередньому підрозділі, поведінка природних об’єктів часто може надати надзвичайно корисну інформацію для вирішення деяких питань практики технічними засобами. В цьому полягає мета науки біоніки, яка на основі вивчення природних систем, продукує технічні рішення, що їх імітують.

Не виключенням є і організація групового польоту літальних апаратів, для чого корисним може бути аналіз аналогічних процесів у природних системах, де можливим є груповий тривимірний рух, а саме:

- рух великих зграй птахів;
- рух великих косяків рибин;
- рух великих хмар літаючих комах.

Слід відмітити, що через дуже малі розміри і дуже велику кількість особин рух комах досліджений набагато гірше, ніж для більш зручних для спостереження людиною птахів та рибин, тому розглянемо докладніше тільки ці два випадки.

1.3.1. Дослідження польоту зграй птах

В першу чергу, коли говорять про рух зграй птахів, на думку можуть прийти зграї перелітних птахів, які рухаються переважно клином. Цей випадок руху спрямований на використання наступними птахами вихорів, які утворили передні, більш сильні птахи, і дозволяє економити сили. При русі малого ґвинтокрилового БПЛА аеродинамічна картина відрізняється від руху птаха значно меншою турбулентністю, адже у першому випадку число Рейнольда (записане, наприклад, через кінематичну в’язкість) складатиме приблизно:

$$\mathrm{Re}_{БПЛА} = \frac{ul}{\nu} = \frac{25 \cdot 0{,}1}{1{,}5 \cdot 10^{-5}} = 1{,}6 \cdot 10^{5},$$

де u = 25 м/с – досить велика, як для малого БПЛА швидкість, взята з метою виконання більш обережних порівняльних оцінок;

l = 0,1 м – характерні розміри малого БПЛА, прийняті раніше;

$\nu = 1{,}5 \cdot 10^{-5}$ м^2/с – кінематична в'язкість повітря при нормальних умовах.

Число Рейнольдса, наприклад, для журавля звичайного складатиме величину близько:

$$\mathrm{Re}_{жур} = \frac{ul}{\nu} = \frac{14 \cdot 1}{1{,}5 \cdot 10^{-5}} = 9{,}3 \cdot 10^5,$$

де u = 50 км/год ≈ 14 м/с – середня швидкість польоту журавля;

l = 1 м – характерні розміри птаха.

Таким чином, отримана для птаха величина приблизно у 6 разів більше, ніж для БПЛА, отже, для останнього роль використання вихорів є значно меншою, ніж для птахів, навіть за умови, якби аеродинаміка їхнього польоту була ідентичною (хоча з іншого боку цілком очевидно, що аеродинамічні процеси у ґвинтокрилих літальних апаратів та птахів із крилами, що відштовхуються від зон підвищеного тиску, відрізняються). Таким чином, політ «клином» для БПЛА не представляє серйозного інтересу.

Зовсім іншу картину маємо при розгляді надзвичайно красивого явища, яке виконують у небі великі зграї деяких видів птахів (шпаки, галки, ворони) – рис. 1.4.

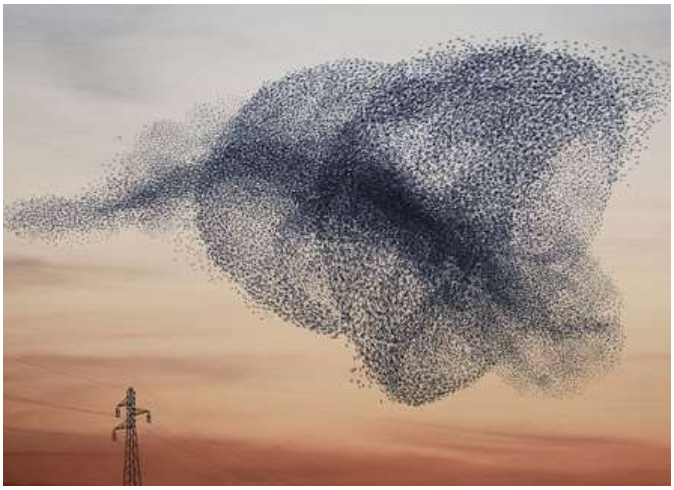


Рис. 1.4. Приклади мурмурації, яку найчастіше виконують зграї саме шпаків.

В загальному випадку мурмурацією називають скоординований рух величезних зграй птахів, що утворюють динамічні об’ємні фігури змінної щільності [3]. Слід відмітити, що дане явище є достатньо загадковим і у вчених відсутнє спільне бачення суті цього процесу, що в першу чергу стосується мети його виконання.

Деякі дослідження [4] показують, що мурмурація дозволяє зграї краще ухилятися від хижаків, увага яких розосереджується через величезну кількість швидких рухомих цілей, в результаті чого їм стає практично неможливо вихопити одну пташку із таким чином рухомої зграї. Логіка підказує, що при проведенні регулярних рухів зграєю, хижакам було би значно легше пристосуватися до них, «вичисливши» траєкторії окремих пташок і, відповідно, вхопити їх.

Такі ж міркування доцільно використовувати і для перехоплювачів автоматичної дії із самонаведенням. При цьому завжди існує певна методика попадання ракети (керованого снаряду) у рухому ціль – по тепловому сліду, максимуму електромагнітних випромінювань або акустичних хвиль, тощо. Однак, у будь-якому випадку джерелом таких дій є конкретний БПЛА, але якщо їх багато і вони безперервно рухаються, то вибір кінцевої цілі буде весь час змінюватися, що може привести до випадкового спрацьовування заряду перехоплювача, його виходу з ладу, або, що більш імовірно, влучання у випадкові сусідні перешкоди.

Встановленою особливістю мурмурації [5] є те, що у кожен конкретний момент часу шпак «бачить» (або відчуває) усіх інших шпаків зграї, кількість яких може складати тисячі особин. Як заявляють автори згаданого дослідження, кожен птах для вироблення наступних своїх дій керується положеннями абсолютно усіх членів зграї, а не лише найближчих сусідів, як вважалося до даної роботи. На нашу думку, питання про урахування положень усіх птахів, або тільки найближчих сусідів, є спірним. У випадку реалізації керування групою БПЛА оптичне розпізнавання є можливим

тільки для найближчих сусідів, оскільки жодні маркери, чи інші засоби оптичної ідентифікації не будуть надійно розпізнаватися через рій аналогічних міток, розташованих значно ближче до апарату, який здійснює розпізнавання.

Таким чином, корисним моментом, який однозначно слід запозичити у птахів при реалізації поведінки групи БПЛА, є їх взаємодія із перехоплювачем-хижаком. В інших аспектах явище мурмурації досить малодосліджене для його активного використання в біонічних цілях.

1.3.2. Дослідження особливостей плавання зграй риб.

Моделювання поведінки зграй рибин також дає деякі корисні результати, які пояснюють групову поведінку живих істот, і які також можна використовувати при розробці САК групою БПЛА. Так, встановлено, що у всіх без винятку випадках, зграя рибин раніше помічала хижака, що до неї наближається, аніж одна усамітнена риба при всіх інших однакових умовах. Очевидно, це пов’язано із кількістю очей та мозків, що обробляють отриману зорову інформацію на предмет пошуку хижаків (одному ж організму доводиться поступово «сканувати» очима оточуючий простір, тому і хижак, який наближається з одного із безлічі боків, буде помічений пізніше. Таким чином, сенсорні пристрої членів групи БПЛА (якщо, звичайно, апарати обладнані ними) повинні бути направлені на різні точки небесної сфери, аби мінімізувати час виявлення супротивника.

При плаванні риб у дослідників великі питання викликає напрям руху зграї (який, ймовірно задають декілька найбільш зголоднілих особин, що пливуть попереду зграї, а інші рибини слідують за ними), але у випадку БПЛА такої проблеми не виникає, оскільки тактична задача для групи є цілком визначеною, аж до конкретної траєкторії руху центру групи виду (1.1).

Деякою особливістю зграй рибин є дещо більша різноманітність їх розмірів у порівнянні, наприклад, із шпаками. Дослідженнями встановлено,

що у зграї збиваються рибини якомога більш схожого розміру: малі окремо, великі окремо. Це не дозволяє хижакові вибрати із великої кількості особин якусь одну, що відрізняється розмірами: всі члени групи «на око» абсолютно однакові. Для групи БПЛА це також є важливим зауваженням, оскільки у випадку неоднорідної групи значно збільшується імовірність ураження тих апаратів, що є унікальними і вирізняються з поміж основної маси інших. Із цих міркувань категорично не рекомендується оточення якогось одного БПЛА, що несе унікальне обладнання, чи має унікальну функцію (тобто дуже важливий і вирізняється з поміж інших) великою кількістю інших (які на перший погляд можуть видатися його захисниками). При цьому перша ж ракета-перехоплювач ймовірно уразить саме той апарат, що відрізняється (тобто найважливіший), оскільки інші будуть нерозрізненними і ухилятимуться від перехоплювача, кидаючи його один на одного (до тих пір, поки перехоплювач не вийде на апарат, що вирізняється з поміж інших).

В цілому, поведінка птахів більш підходить для імітації (принаймні окремих її елементів) при розробці системи керування групою БПЛА, в першу чергу, через однакове фізичне середовище, що використовується для руху, по-друге, через уніфікованість органів чуттів птахів (в деяких дослідженнях підіймаються питання використання рибами та водними організмами інших органів чуттів, окрім зорових), а також, через у деякій мірі глибшу наукову проробку питань моделювання рухів птахів у порівнянні з рибами. Окремі елементи поведінки птахів доцільно запозичити для керування рухом групи БПЛА, зокрема їх взаємодії з хижаками, що нападають на зграю.

1.4. Уточнена постановка задачі дослідження

У даній роботі розробці підлягає система автоматичного керування групи БПЛА, для чого попередньо слід розробити та розв’язати чисельним методом наступні математичні моделі:

- взаємодії членів групи одне з одним;

- взаємодії групи із близькими перешкодами та стінками отворів;

- взаємодії групи із перехоплювачем.

У якості чисельного методу використати найбільш простий метод розв'язку диференціальних рівнянь – метод Ейлера, який, зважаючи на високі обчислювальні потужності сучасних ПК, дає задовільні результати по збіжності та точності за цілком прийнятні інтервали часу.

Реалізацію моделей (у вигляді відповідних підпрограм), тестування та аналіз результатів провести в одній із математичних програм CAD-типу: MathCad, MatLab, Maple та ін. – за допомогою вбудованих у них засобів та мов програмування.

Час виконання розрахунків в рамках моделювання не повинен виходити за межі кількох хвилин на комп'ютері з наступними системними вимогами:

- частота центрального процесора (далі – ЦП) – 3 ГГц;

- кількість ядер ЦП – 4 шт.;

- об'єм оперативної пам'яті – 6 Гб;

- вільне місце на жорсткому диску – до 10 Гб;

- операційна система сімейства Windows, версії 7 та вище.

Необхідно запропонувати апаратні елементи (на базі однієї із популярних платформ), на основі яких можна звести розроблену САК.

Для розроблених програмних продуктів слід створити комплект необхідної документації для можливості використання іншими користувачами, окрім розробника.

1.5. Висновки по розділу

Таким чином, у даному розділі докладно розглянута проблема організації руху групи безпілотних літальних апаратів. Проаналізовано задачі, що можуть вирішуватися при такому русі (зокрема, спостереження,

або наступального плану), а також основні труднощі, які при цьому можуть виникнути. Виділено такі елементи всієї повної задачі, що підлягають розв’язку: забезпечення відсутності зіткнень апаратів між собою, з просторовими перешкодами, а також стінками отворів, через які групі слід пройти. Також доцільно запозичити у птахів елементи процесу їх взаємодії із хижаком, що нападає на зграю.

На основі даної інформації сформовано завдання для подальшого дослідження, керуючись яким, можна переходити до процесу розробки та реалізації вказаних моделей, а також САК в цілому.

2 ОСОБЛИВОСТІ ВИРІШЕННЯ ЗАДАЧІ СЛІДУВАННЯ ЗАДАНОЮ ТРАЄКТОРІЄЮ У ВИПАДКУ ГРУПОВОГО ПОЛЬОТУ БПЛА

2.1 Введення базових геометричних понять, що описують груповий рух точкових об’єктів

Розглянемо геометричні параметри, що описують групу літальних апаратів, причому, що рухаються вздовж заданої траєкторії.

В першу чергу, для керування групою об’єктів потрібно знати координати кожного з них, отже використовуватимуться масиви координат:

$$\begin{cases} X(t) = \{x_1(t), x_2(t), \ldots, x_N(t)\} = \{x_n(t) \mid n = 1..N\} \\ Y(t) = \{y_1(t), y_2(t), \ldots, y_N(t)\} = \{y_n(t) \mid n = 1..N\} \\ Z(t) = \{z_1(t), z_2(t), \ldots, z_N(t)\} = \{z_n(t) \mid n = 1..N\} \end{cases} \quad (2.1)$$

Враховуючи, що на практиці час є дискретним, замість (2.1) можна оперувати тривимірним масивом координат розміром N x 3 x K, який складається з набору двовимірних «перерізів» виду:

$$X_k = \begin{pmatrix} x_{1k} & y_{1k} & z_{1k} \\ x_{2k} & y_{2k} & z_{2k} \\ \ldots & \ldots & \ldots \\ x_{Nk} & y_{Nk} & z_{Nk} \end{pmatrix}. \quad (2.2)$$

Кожен такий «підмасив» X_k або по-іншому переріз виду (2.2), задає положення усіх літальних апаратів в даний конкретний k-тий момент часу. На основі цього двовимірного масиву існує можливість розраховувати відстані δ_{ij}, що входять до складу (1.5). Значення такої відстані у k-тий момент часу обчислюватиметься за формулою:

$$\delta_{ijk} = \sqrt{(x_{ik} - x_{jk})^2 + (y_{ik} - y_{jk})^2 + (z_{ik} - z_{jk})^2}\,. \quad (2.3)$$

Із усього набору значень δ_{ijk}, очевидно, найбільший інтерес представляє найменше значення $\min\limits_{i,j} \delta_{ijk}$, оскільки саме воно визначає можливість виникнення зіткнення апаратів групи між собою. Для розрахунку усіх δ_{ijk} слід

провести $C_N^2 = \frac{N(N-1)}{2}$ операцій виду (2.3), що при числі елементів групи до 100 складатиме відповідно до 5 тис. операцій виду (2.3) на один крок по часу. Відмітимо, що тут під C_n^m мається на увазі стандартне математичне позначення кількості різних сполучень по m об'єктів із їх загальної кількості, рівної n, як кажуть «із n по m»; C_N^2 рівне кількості різних пар, що можна розглядати, якщо всього об'єктів у групі N.

Для оцінки значення доцільного кроку по часу Δt, можна використати співвідношення $C_k C_{k+1} << r_a$ (тут під C_k мається на увазі точка, що є центром мас апарату у k-тий момент часу), яке означає, що за один крок по часу центр мас БПЛА повинен зсуватися на відстань $C_k C_{k+1}$ значно меншу (хоча би на порядок), ніж габаритні розміри апарату (які раніше задавалися відстанню r_a від ЦМ до найбільш віддаленої від нього точки БПЛА).

Якщо задатися відстанню $r_a \approx 10$ см, прийнятою раніше, та швидкістю апарату 25 м/с, що розглядалася раніше, то рекомендований крок по часу складатиме $(0{,}1/10)/25 = 4 \cdot 10^{-4}$ с. При такому кроці будуть відстежуватися усі зміщення апаратів, більші за 1 см. Для введення певного запасу доцільно прийняти:

$$\Delta t = 10^{-4} \text{ с}. \qquad (2.4)$$

2.2. Проектування спільного комунікаційного середовища (СКС) групи БПЛА

2.2.1. Взаємодія на основі використання технологій радіозв'язку.

Для повноцінного використання усіх розроблених формул та залежностей слід знати поточні координати (2.2) усіх членів групи. Таку задачу порівняно просто можна реалізувати за допомогою GPS-датчика (надає координати x та y) та альтиметру (координата z). Передача даних, що

роблять можливим застосування технології GPS в принципі, як відомо, ведеться за допомогою радіосигналів на частотах порядку 1-1,5 ГГц.

Супутники випромінюють відкриті для використання сигнали в діапазонах: L1 = 1575,42 МГц і L2 = 1227,60 МГц (починаючи з Блоку IIR-M), а моделі IIF випромінюють також на частоті L5 = 1176,45 МГц. Ці частоти є відповідно 154-й, 120-й і 115-й гармоніками фундаментальної частоти 10,23 МГц, що генерується бортовим атомним годинником супутника з добовою нестабільністю не гірше 10^{-13}; при цьому частота атомного годинника має зсув до значення 10,229 999 995 43 МГц, щоб компенсувати релятивістське зрушення, обумовлене рухом супутника щодо наземного спостерігача і різницею гравітаційних потенціалів супутника і спостерігача на поверхні Землі [6]. Навігаційна інформація може бути прийнята нескладною антеною (зазвичай в умовах прямої видимості супутників) і оброблена за допомогою GPS-приймача.

У якості такого приймача може бути використаний пристрій GPS NEO-6M, який зі своєю антеною показаний на рис. 2.1. Даний пристрій відрізняється найнижчою ціною серед усіх інших пристроїв даного класу, але, зважаючи на те, що він використовується дуже широко, в т.ч. і у мобільних телефонах, ефективність його роботи не викликає сумнівів.

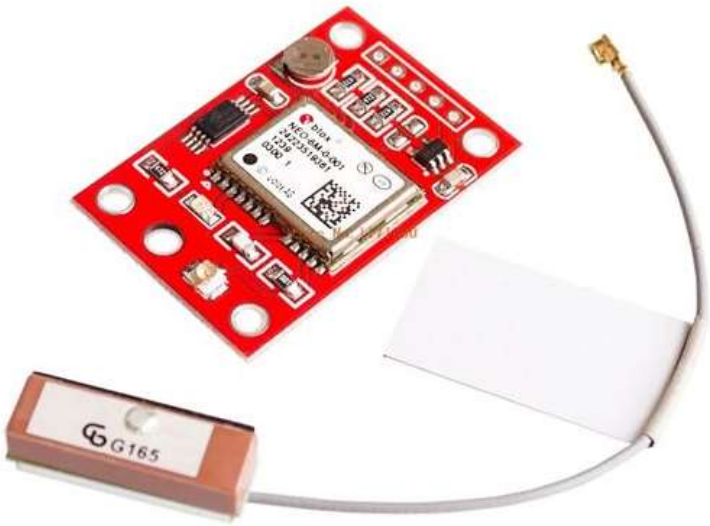


Рис. 2.1. Зовнішній вигляд доступного, але ефективного датчику GPS NEO-6M

Таким чином, з використанням датчику GPS стає можливим визначення координат кожного окремого БПЛА за допомогою відповідного датчика, що у нього вбудований.

Також за допомогою радіохвиль слід проводити «спілкування» апаратів один з одним, необхідне для передачі усієї комунікаційної інформації. Для цього можна використовувати також досить доступні, але ефективні в умовах коротких дистанцій радіочастотні приймачі та передавачі (поставляються парами) типу MX-05V, XD-RF-5V, XY-FST, XY-MK-5V – рис. 2.2. Частота передачі в них складає 433 МГц, що повністю сумісно із системою GPS, описаною вище.

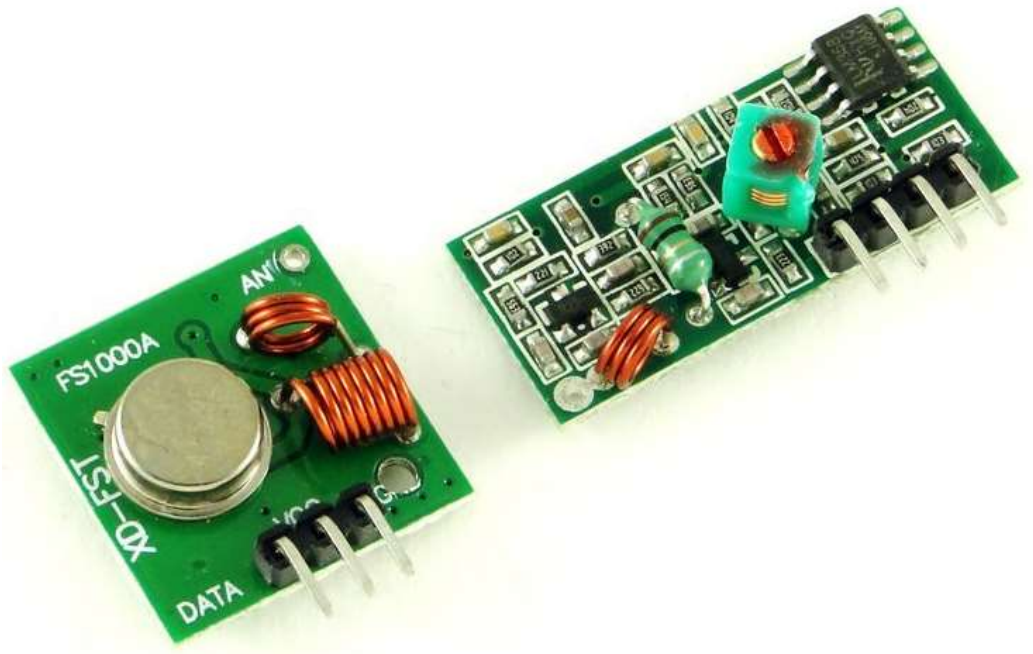


Рис. 2.2. Варіант приймача та передавача системи радіозв’язку для членів групи БПЛА.

2.2.2. Взаємодія на основі технологій оптичного розпізнавання.

Окрім радіозв’язку, можуть існувати і інші канали отримання інформації, наприклад, візуальний, звуковий, ультразвуковий, механічний і

т.п. Одним із перспективних рішень, що тісно пов’язано із швидко прогресуючою галуззю штучного інтелекту, може бути оптична взаємодія.

Так, для визначення положення найближчих сусідів апарати можуть використовувати оптичне розпізнавання, для чого, очевидно, слід передбачити певні мітки, які мають бути розміщені на корпусі БПЛА. Ці мітки повинні відповідати деяким умовам, порушення яких може звести нанівець саму суть ідеї оптичного розпізнавання. Отже, мітки мають:

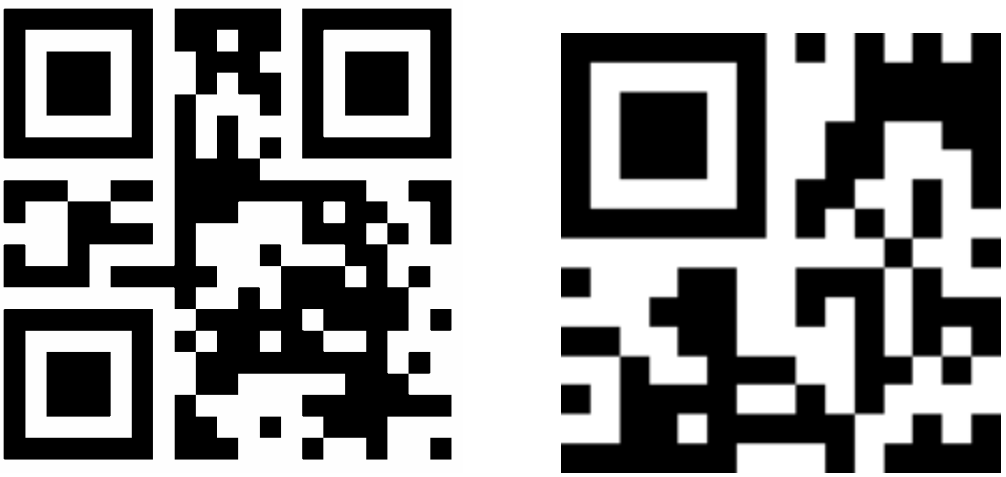
- бути оригінальними, тобто такими, що не зустрічаються (або надзвичайно рідко зустрічаються) у природі та навколишньому середовищі в цілому. Це необхідно для того, щоб виключити можливість хибної ідентифікації сторонніх об’єктів у якості членів групи;

- бути добре розпізнаваними, тобто такими, що при потраплянні у поле зору камери будь-якого апарату групи, імовірність успішного розпізнавання наближалася до 1;

- мати елементи орієнтації, за якими можна було би робити висновок про просторову орієнтацію даного апарату, на якому встановлена камера, та апарату, на якому було ідентифіковано мітку;

- у деякій мірі відрізнятися для різних апаратів групи, тобто включати в себе ідентифікаційний код апарату, на якому розміщена дана мітка.

Усім цим вимогам відповідає така існуюча технологія, як QR-коди – рис. 2.3.



а) *б*)

Рис. 2.3. Приклади QR-кодів: *а* – найменший із QR-кодів (приклад);

б – Micro QR (приклад).

Звичайні QR-коди мають дуже високу інформаційну місткість, яка є непотрібною у даному випадку. Корисною інформацією буде лише ідентифікаційний номер конкретного апарату, а усю іншу частину коду складатиме візерунок групи. Для таких цілей цілком достатньо можливостей Micro QR (161 біт інформації), навіть якщо кількість членів групи буде числитися тисячами (як відомо, навіть 16 біт інформації може вмістити 65536 різних чисел, і якщо в 161 біті виділити 16, то 145 бітів, що залишаються, описуватимуть мітку групи).

У роботі для виконання поставленої задачі було створено програмний продукт – рис. 2.4, що на модельному прикладі (за допомогою веб-камери ПК) демонструє запропонований підхід і здійснює розпізнавання QR-кодів, що мають бути нанесені на літальний апарат з різних сторін. Для цього використана взаємодія з веб-камерою персонального комп’ютера, яка за командою користувача робить знімок у момент, коли камера направлена на об’єкт. Після цього натисканням кнопки «Визначити відстань» виконується оцінка. За бажанням користувача код може бути перевірений на коректність.

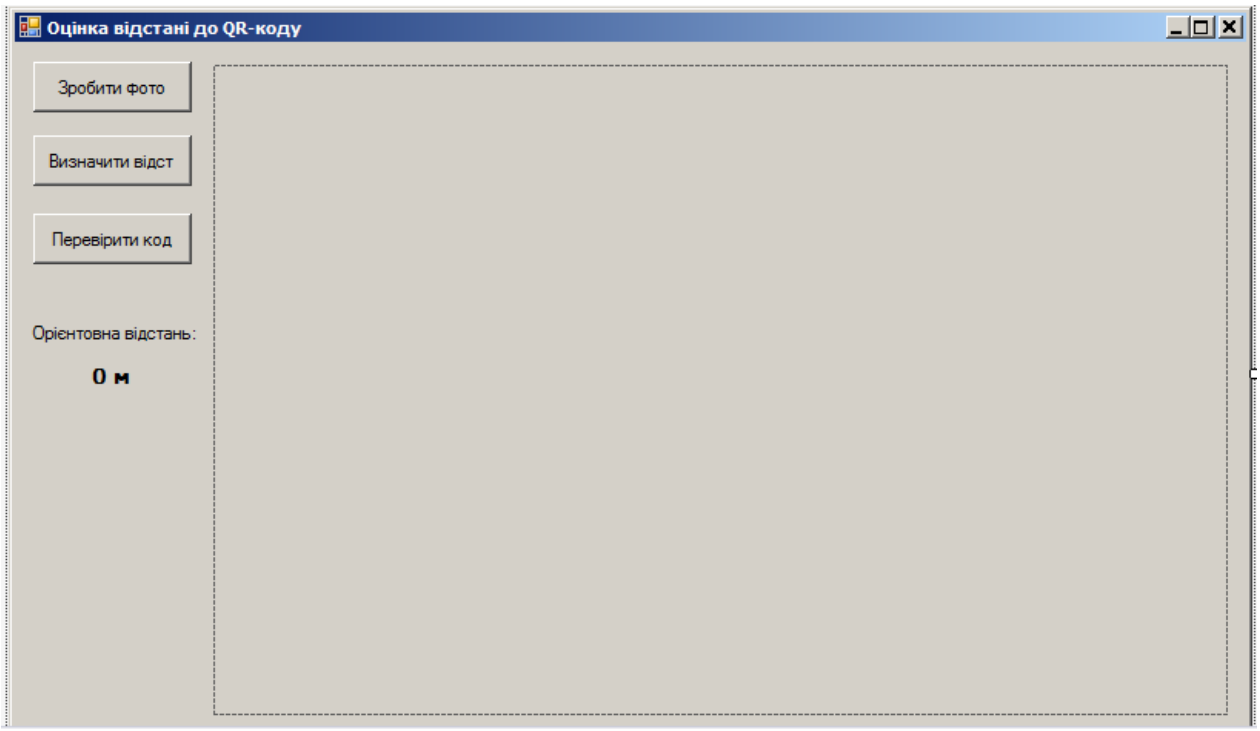


Рис. 2.4. Загальний вигляд вікна програми, що у модельному середовищі демонструє запропонований підхід до оцінки відстаней між БПЛА.

Робота даної програми показала задовільне визначення відстані до QR-коду в діапазоні від 20 см до 4 м.

2.3. Розробка умов забезпечення безпеки групового польоту БПЛА з використанням СКС

Як зазначено у попередньому підрозділі, на кожному БПЛА мають бути присутніми GPS-датчик та альтиметр, які надають йому в реальному часі поточні координати апарату. Однак, при такій організації даному конкретному апаратові будуть доступними тільки його координати, і більше ніякі (за виключенням деяких, що методом оптичного розпізнавання встановлені у полі зору камери апарату). В той же час для ефективної роботи усієї системи на кожному апараті мають бути присутніми координати усіх членів групи, а не лише даного апарату. Відповідно, виникає задача обміну координатами між членами групи, яку можна здійснити по радіоканалу, з використанням пристроїв, описаних у пункті 2.2.1 (напр. на рис. 2.2). При

цьому очевидно слід розробити певний протокол передачі, адже якщо вона буде відбуватися одночасно з усіх апаратів, то через явище інтерференції майже жоден з апаратів не отримає корисної інформації. Передача на різних частотах також не є виходом, через те, що кількість членів групи може бути великою (отже, резервуванню має підлягати широкий діапазон частот),а гірше всього – може бути змінною.

В той же час, коли «співбесідників» багато, нормальною практикою є надання «слова» кожному з них по циклічній системі один-за-одним. При початку передачі апарат має назвати себе, передавши ідентифікаційний код, а після цього координати – поступово, x, y та z. Для передачі інформації можна використовувати будь-який завадостійкий код, бажано із низькою обчислювальною складністю. При такому способі передачі слід ввести певну затримку-мовчання для унеможливлення накладання сигналів різних апаратів один на одного. Часові параметри сигналів тоді можуть бути наступними. На передачу ідентифікаційного номеру та координат (тобто всього - 4 чисел) при частотах біля 400 МГц можна виділити 5 мкс. Тоді при мовчанні рівної тривалості на передачу одного апарату ітиме 10 мкс. Відповідно, на один повний цикл передач усіма членами групи ітиме час $10N$ мкс, що при $N = 100$ елементах групи цей час складатиме 1 мс, що задовільно корелює з прийнятим раніше значенням (2.4).

2.4. Урахування небезпеки внутрішніх колізій БПЛА групи з використанням СКС

Важливим параметром роботи проектованої системи є її обчислювальна складність, особливо в задачі забезпечення захисту від внутрішніх зіткнень апаратів у межах групи. Розглянемо цю проблему докладніше.

При значенні Δt за (2.4) та озвучених раніше міркуваннях, пов’язаних із формулою (2.3), можна розрахувати, що на секунду буде необхідним

виконувати $10^4 \cdot 5000 = 50$ млн. операцій виду (2.3), що є забагато, особливо, якщо взяти до уваги їх математичну складність та слабкі обчислювальні потужності мікроконтролерів, на базі яких будуються усі сучасні БПЛА. Таким чином, на відміну від зграї птахів відстежувати взаємні положення доцільно лише для найближчих сусідів.

Слід зауважити, що задача визначення апаратів, які є такими найближчими сусідами для кожного *n*-ного апарату (будемо говорити про перелік апаратів, які знаходяться у «сфері близькості», яка для кожного конкретного апарату буде своєю, і відповідний перелік також), не є досить простою ані обчислювально, ані алгоритмічно. Дійсно, без прямого обчислення відстаней C_iC_j не можливо встановити, наскільки віддалені *i*-тий та *j*-тий БПЛА. Для зменшення кількості необхідних обчислень можна проводити постійний (мається на увазі на кожному кроці по часу) моніторинг відстаней лише для тих апаратів, що уже були у «сфері близькості» даного *n*-ного апарату на попередньому кроці по часу. Якщо виявляється, що на черговому кроці якийсь апарат виходить із «сфери близькості», віддаляючись від її центру, тобто центру *n*-ного апарату, то для нього розрахунок відстані більше не ведеться і на наступному кроці він бере участь в розрахунках уже на рівних з віддаленими апаратами. Для усіх апаратів, які не входять у «сферу близькості» на даному кроці, розрахунки відстаней (2.3) із перевіркою (1.5) виконуватимуться випадково. При цьому слід задатися відсотком усіх апаратів, що не входять до «сфери близькості», які будуть випадково перевірятися на близькість. Для забезпечення мінімальної обчислювальної складності, можна, наприклад, говорити про 1 % усіх апаратів, які не входять у «сферу близькості», і повинні перевірятися на даному кроці по часу. Враховуючи статистичний зміст поняття імовірності, можна стверджувати, що для одного конкретного апарату ця цифра означає, що імовірність бути перевіреним складе 0,01, а, відповідно, імовірність бути не перевіреним складне 0,99. Всього за секунду, враховуючи (2.4), буде

виконано 10 тис. перевірок і імовірність того, що якийсь апарат не буде перевірений складе

$$0{,}99^{10000} \approx 2 \cdot 10^{-44},$$

тобто така подія є неймовірною: за секунду усі апарати будуть точно перевірені. Можна також оцінити час (точніше кількість кроків по часу k_1), протягом якого усі апарати, які початково не входили в «сферу близькості», будуть перевірені хоча б один раз (для цього задамося імовірністю не потрапляння результатів досліду у межі 3σ нормального розподілу, що, як відомо, складає 0,28%, а така подія для практики вважається цілком неймовірною):

$$0{,}99^{k_1} = 0{,}0028$$

$$k_1 = \frac{\ln 0{,}0028}{\ln 0{,}99} \approx 585 \text{ кроків.}$$

Таким чином, при перевірці всього лише 1 % БПЛА на кожному кроці по часу, уже через біля 600 кроків із достовірною імовірністю (99,72%) будуть перевірені усі апарати групи, тому такий рівень (1 %) можна вважати цілком задовільним. Тоді можна оцінити необхідну кількість кроків і при умові перевірки 0,1 % усіх апаратів:

$$k_2 = \frac{\ln 0{,}0028}{\ln 0{,}999} \approx 5875 \text{ кроків.} \qquad (2.5)$$

Таким чином, навіть при випадковому виборі для перевірки всього лише 0,1 % всіх БПЛА на кожному кроці по часу, можна із впевненістю стверджувати, що приблизно за 0,6 секунди (із дуже високою імовірністю 99,72 %) усі апарати будуть перевірені на входження до «сфери близькості». Такий показник також є задовільним, тому остаточно приймаємо для кількості апаратів, що перевіряються на кожному кроці, число рівне 0,1 %.

Таким чином, у даному розділі розглянуто особливості вирішення задачі слідування заданою траєкторією у випадку групового польоту БПЛА: обрано геометричні та часові параметри процесів, запропоновано протоколи

передачі даних між членами групи та способи отримання вхідної інформації для забезпечення безпечного руху групи.

3 ВИРІШЕННЯ ЗАДАЧІ ПЕРЕГРУПОВУВАННЯ БПЛА ПРИ ВИНИКНЕННІ МОТИВУЮЧИХ ФАКТОРІВ

3.1. Аналіз переліку задач, що потребують перегруповування

3.1.1. Ухилення від точкового перехоплювача.

Однією із найважливіших для практики задач є ухилення групи від точкового перехоплювача, що до неї наближається. Тут мається на увазі керований снаряд, або ракету, ворожий БПЛА чи інший об’єкт, траєкторія якого спрямована у бік рухомої групи апаратів. Важливою особливістю перехоплювача, який являє собою реальну загрозу для групи, є те, що при умові збереження поточних векторів швидкостей її елементів, перехоплювач достовірно потрапить всередину групи, а отже, із дуже високою імовірністю вразить один із апаратів. Умова такого «перехоплення» може бути записана наступним чином:

$$\min_i |C_i C_0| < 2r_a, \tag{3.1}$$

де C_0 – центр мас перехоплювача.

При записі (3.1) використовувалися наступні припущення:

- максимальний радіус перехоплювача приблизно рівний максимальному радіусу БПЛА групи;

- у формулі використано поточні положення центрів мас елементів групи та перехоплювача, які є змінними в часі, тому у цій залежності насправді ліва частина є функцією часу;

- мінімум шукається по всім елементам групи, тобто фактично знаходиться той апарат, який найближче всього розміщений у даний момент часу до центра перехоплювача.

Відмітимо, що теоретично перехоплювач спочатку може бути і не направлений до групи, а зберігати певний «паралельний» чи «дотичний» курс

(відповідно, такий об’єкт не буде довгий час вважатися небезпечним, стаючи таким лише після досягнення ним певної малої відстані від всієї групи

Отже, зважаючи на особливість небезпечних перехоплювачів, група має адекватно реагувати на них (сторонні об’єкти, що наближаються до групи). Для цього слід передбачити такий варіант перегруповування, як тимчасове утворення коридору (або значної порожнини), де рухається перехоплювач, ніби пронизуючи групу.

3.1.2. Прольот групи крізь отвір невеликого розміру.

Ще однією практично цікавою задачею є забезпечення прольоту групи апаратів через отвір у великій перешкоді, облітати яку згори або з боків не доцільно. По-перше, слід визначитися з умовою, при якій взагалі прольот є можливим: для цього характерні розміри отвору r_o мають бути більшими, ніж розміри апарату r_a (інакше навіть один апарат не зможе протиснутися через такий, дуже малий отвір), причому із певним запасом, який приймаємо на подвійному рівні:

$$r_o \geq 2r_a \tag{3.2}$$

Виконання умови (3.2) є обов’язковим не лише при русі заданою траєкторією групи БПЛА, а і взагалі, для забезпечення прольоту хоча б одного апарату. З іншого боку отвір може бути дуже великим, і тоді жодні перегруповування не будуть потрібними, оскільки через нього зможе пройти уся група без зміни форми та розмірів. Така ситуація є можливою, якщо виконується умова:

$$r_o \geq r_g. \tag{3.3}$$

Отже, необхідність перегруповування при прольоті групи отвору існує при виконанні (3.3), а щоб задача взагалі мала розв’язок має виконуватися (3.2).

3.1.3. Ухилення від просторової перешкоди.

Нарешті, ще однією ситуацією, яка потребує перегруповування групи БПЛА, є проходження траєкторії польоту недалеко від перешкод, при чому стає можливою ситуація, показана на рис. 1.2, *б* у його нижній частині, а саме: один апарат цією траєкторією пройшов би без проблем, але при русі нею цілої групи, крайні апарати (такі, що рухаються на периферії групи) можуть зіткнутися з перешкодою.

Очевидно, для уникнення описаної ситуації необхідно здійснювати зміну просторової конфігурації групи, яка полягає у зсуві її передньої частини в напрямку від перешкоди, проходженні її та поверненні до початкової конфігурації.

3.2. Створення моделей поведінки груп БПЛА при вирішення виявлених задач перегруповування з використанням СКС

У попередньому підрозділі обґрунтована актуальність задач перегруповування групи БПЛА при трьох типових ситуаціях: ухиленні від перехоплювача, проходженні групи через не дуже великий отвір та проходження групи недалеко від просторової перешкоди. У даному підрозділі слід розробити конкретні моделі поведінки групи, що забезпечували би ефективне виконання озвучених операцій.

3.2.1. Ухилення від точкового перехоплювача.

По-перше, відмітимо, що у перехоплювача є певні координати, які без обмеження загальності можна розглядати у тій же системі координат, у якій задаються і координати апаратів групи. Розгляд задачі визначення координат перехоплювача виходить за рамки даного дослідження (може базуватися на вирішенні задач розпізнавання образів), тому координати перехоплювача у кожен момент часу вважаємо відомими (зокрема, він може рухатися певною заданою траєкторією) і рівними x_0, y_0:

$$x_0 = x_0(t),\ y_0 = y_0(t). \qquad (3.4)$$

На значному віддаленні від перехоплювача усі елементи групи рухаються заданою траєкторією і мають певні швидкості $\vec{V}_i$. Чим ближче наближається перехоплювач, тим більшим має бути його вплив на рух апаратів групи, що має втілюватися у виникненні додаткової компоненти швидкості $\vec{V}_{0i}$, що забезпечуватиме ухилення *i*-того апарату від перехоплювача. Як у будь-якого вектору, у цієї компоненти швидкості можна розглядати окремо модуль та напрям. Очевидно, для найбільш ефективного ухилення від перехоплювача апарат має віддалятися від нього, тобто напрямок швидкості має бути направлений вздовж вектору $\overrightarrow{C_0C_i}$, тобто строго від перехоплювача:

$$\vec{V}_{0i} \uparrow\uparrow \overrightarrow{C_0C_i} \qquad (3.5)$$

Величину же швидкості можна робити обернено пропорційною до відстані між даним апаратом та перехоплювачем:

$$\left|\vec{V}_{0i}\right| \sim \frac{1}{\left|\overrightarrow{C_0C_i}\right|}, \qquad V_{0i} = \frac{k}{C_0C_i}. \qquad (3.6)$$

Очевидно, при перебільшенні значення (3.6) максимально можливої швидкості апарату, вона буде обмежена цим природним чином. Коефіцієнт пропорційності *k*, наявний у (3.6), має підбиратися експериментальним чином при дослідженнях готової математичної моделі з метою мінімізації імовірності перехоплення.

При завданні швидкості у вигляді (3.5)-(3.6) досягатиметься задача ухилення кожного члена групи (а, отже, і всієї групи) від точкового перехоплювача, який є небезпечним, тобто без проведення додаткових дій загрожує знищенням БПЛА групи.

3.2.2. Прольот групи крізь отвір невеликого розміру.

Задача детектування отвору (так само, як і визначення координат перехоплювача) виходить за рамки даного дослідження (має вирішуватися

методами галузі розпізнавання образів, або бути заданою на основі даних розвідки). Отже, будемо вважати, що координати (тобто будь-які геометричні параметри) отвору є відомими.

В першу чергу, визначенню підлягає радіус групи, що є безпечним для прольоту отвору. Очевидно, він має задаватися із певним запасом для забезпечення можливості незначного маневрування або випадкових збурень траєкторії кожного даного БПЛА (наприклад, через вітрові впливи). Вводимо, як і раніше, подвійний запас по даному показнику.

По-друге, при підльоті до отвору передні апарати мають сформувати заданий радіус групи, для чого швидкість периферійних пристроїв зменшується для того, щоби вони пропустили апарати, які летять ближче до траєкторії.

По-третє, після прольоту отвору слід відновити геометричні розміри, що були наявні в групі до перегруповування (тобто форму близьку до кулеподібної, радіусом r_g).

3.2.3. Ухилення від просторової перешкоди.

Аналогічно до попереднього пункту можна діяти і у випадку прольоту над просторовою перешкодою (або у загальному випадку «біля» неї, оскільки перешкода взагалі може розміщуватися і строго справа/зліва від траєкторії, і для її прольоту також потрібно застосовувати викладені нижче міркування). А саме, можна діяти шляхом зменшення радіусу групи, причому до таких значень, що є достатніми для безпечного прольоту поруч із перешкодою. Цей варіант описаний у попередньому пункті.

Також можливий і варіант комбінації двох попередніх пунктів: зменшення радіусу групи може виконуватися разом із ухиленням від перешкоди (як раніше описувалося ухилення від перехоплювача). Після проходження перешкоди слід повернутися до сферичної форми групи із початковим радіусом r_g, центр якої лежить на заданій траєкторії.

3.3. Висновки по розділу.

Таким чином, у даному розділі розглянуто задачі перегруповування групи БПЛА, що рухається заданою траєкторією. Виділено три типи таких задач: ухилення від точкового перехоплювача, прольот отвору через який проходить траєкторія, а також проходження біля просторової перешкоди, яка розміщена поруч із траєкторією руху групи. Запропоновано правила перегруповування групи апаратів, що дозволятимуть безпечно проводити вказані операції.

4 ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ ПРИ ВИРІШЕННІ ЗАДАЧІ КЕРУВАННЯ ГРУПОВИМ ПОЛЬОТОМ БПЛА

4.1 Вибір технології програмування

Першочерговим питання, яке постає перед розробниками практично будь-якого програмного забезпечення, в т.ч. і для керування польотом групи БПЛА, є вибір моделі або технології його розробки. Такими, що реально використовуються на сьогоднішні день у виробничій практиці, є технології структурного (процедурного) та об’єктно-орієнтованого програмування (описані, наприклад, у [7]). Кожна з них має свої особливості, переваги і недоліки, які розглянемо докладніше.

Структурне, або як його ще називають практикуючі програмісти, процедурне програмування засноване на використанні окремих структурних блоків - в першу чергу, підпрограм (процедур і функцій).

Історично перші комп’ютерні програми були відносно простими і мали пакетний режим роботи: отримуючи на вхід якусь інформацію (можливо, навіть на перфокарті) вони виконували певний обсяг операцій з обробки цих даних і видавали результат. При цьому існувала сувора функціональна залежність виходу від входу – рис. 4.1.

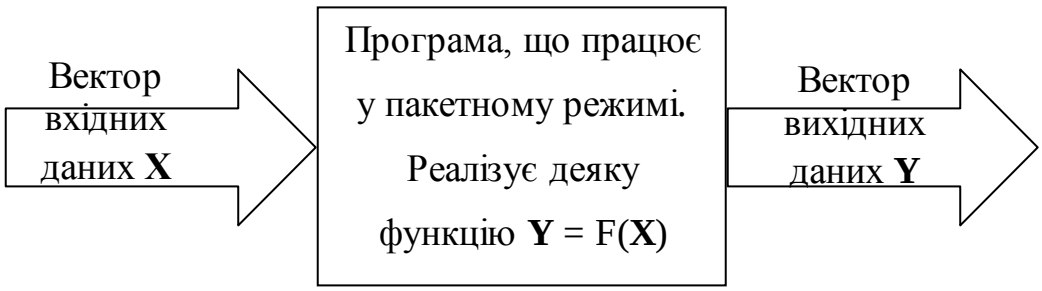


Рис. 4.1. Схема роботи пакетної програми.

Незважаючи на відсутність явного зв’язку функціональності і структури, пакетні програми в основному мали просту лінійну послідовність

виконання. Тобто в них практично відсутні будь-які підпрограми, принаймні, використання підпрограм не було наріжним каменем самої методики програмування.

В міру ускладнення функціональності програмного забезпечення, змінювалася і його внутрішня структура: поступово розвинулася інтерактивна модель взаємодії користувача і програми – рис. 4.2. Програми стали запитувати інформацію і активно реагувати на дії людини. Ускладнення функціональності привело до відповідного ускладнення програмного коду, якого, в першу чергу, стало просто багато. Багато для того, щоб людина-програміст без всяких спеціальних хитрувань швидко і легко розібралася з незнайомим кодом. Розміщувати код у простій лінійній послідовності без виділення великих блоків стало незручно, в першу чергу, для розуміння цього коду.

Рис. 4.2. Схема роботи програми, що активно взаємодіє з користувачем.

Тут слід зазначити психологічні особливості сприйняття людиною складних «великих» завдань. Неструктуроване «велике» завдання (наприклад, написання дипломної роботи) зазвичай викликає певний психологічний ступор і, як результат, повну неможливість поступово розібратися з ним. Людині зручно розбити проблему на не надто велику (зазвичай до десятка, а краще 3-4) кількість завдань (наприклад, розділів у

дипломній роботі), не замислюючись про реалізацію кожного з них. Коли є ясність і розуміння проблеми на найвищому рівні абстракції, слід приступати до деталізації підзадач, кожну з яких слід розбити на окремі «підпідзадачі» тобто підзадачі нижчого рівня, більш дрібні. Уже після такого розбиття слід аналізувати всі перераховані підзадачі. На певному етапі зупиняються і виконують не розбиття чергової підзадачі на більш дрібні, а безпосередню її реалізацію в програмних кодах.

Отже, можна сказати, що розвиток методики програмування відбувався в ногу з розвитком призначеного для користувача інтерфейсу: і грубо кажучи, розвиненому інтерфейсу командного рядка відповідає парадигма структурного програмування.

Говорячи більш строго, структурне програмування має на увазі побудова програми відповідно до трьох основних принципів: слідування, розгалуження, повторення.

Слідування має на увазі, що оператори і блоки програмного коду слідують і виконуються один за іншим. Розгалуження реалізується різними умовними операторами типу if, і дозволяє вибирати один з декількох подальших варіантів виконання програми. Повторення зазвичай відносять на рахунок циклів (що йдуть підряд багаторазових повторів одного і того ж ділянки коду), хоча цей же принцип можна віднести і до підпрограм.

Взагалі ж, структурне програмування у деякій мірі є застарілою методикою програмування, на зміну якій разом з віконним інтерфейсом прийшло об’єктно-орієнтоване програмування (деякі сучасні мови програмування загального призначення навіть не дозволяють створити структурну програму, тільки об’єктно-орієнтовану – як, наприклад, Visual C# чи Java). Проте, при створенні невеликих програм (наприклад, до 10000 рядків коду і без передбачуваного розширення) застосування цієї методики програмування більш виправдано і код краще сприймається, ніж його об’єктно-орієнтований варіант.

Суть же методології об’єктно-орієнтованого програмування полягає в тому, що система розглядається, як сукупність окремих сутностей - об’єктів, які мають набір якихось своїх внутрішніх параметрів - властивостей, а також можуть взаємодіяти між собою за допомогою деяких дій - викликів методів (або трохи більше непрямим чином - шляхом надсилання повідомлень, оброблюваних методами об’єктів; для цього необхідна присутність активної сутності, яка роздає повідомлення адресатам, як, наприклад, менеджер вікон в ОС Windows).

Якщо говорити про програмний код, то для того, щоб оперувати об’єктом, його спочатку потрібно створити. Об’єкти створюються як змінні, у яких типом виступає клас об’єкта. Клас - це просто опис, які властивості можуть мати об’єкти такого типу (тобто яку інформацію вони можуть зберігати), і які у них є методи (тобто які дії вони можуть виконувати). Об’єкт - це набір значень, чому саме рівні властивості даного об’єкта (свої методи кожен об’єкт отримує від свого класу, тобто методи однакові у всіх об’єктів, що належать даному класу).

Для чого потрібен цей специфічний підхід, адже самі по собі об’єкти не додають нічого корисного (навпаки, введення об’єктів ускладнює програму, вносить в неї нові сутності)? Виявляється, реалізовуючи всі сутності, необхідні, згідно з алгоритмом, для роботи програми, у вигляді класів і об’єктів, ми спрощуємо її розуміння для самих себе. Саме тому ОО-підхід рекомендується до застосування для великих проектів (більше десятків тисяч рядків коду), коли утримувати «в голові» всю систему цілком стає важко. Можна сказати, що розбиття програми на об’єкти і проектування їх класів наближає розуміння предметної області до звичного людського образу мислення (в разі «великих» проектів). Людина мислить класами, об’єктами і зв’язками між ними.

Порівняльна складність проектів, що мають однакову функціональність, але побудованих по-різному (згідно структурному і об’єктно-орієнтованого підходів до програмування), як функція їх обсягу

показана на рис. 4.3. З графіка рис. 4.3 слід, що, якщо потрібно реалізувати продукт з невеликою функціональністю (тобто кількість рядків коду, що її реалізують буде очевидно невеликою, порядку кількох тисяч рядків), то це краще робити без застосування класів, так як вони будуть тільки ускладнювати всю справу. Якщо ж програма має більш-менш значну функціональність, а значить, реалізується хоча б кількома тисячами рядків коду, то вже є сенс замислюватися про застосування об'єктно-орієнтованого підходу. Однозначно будуватися за принципами ООП повинні програми, які мають 10000 рядків коду і більш.

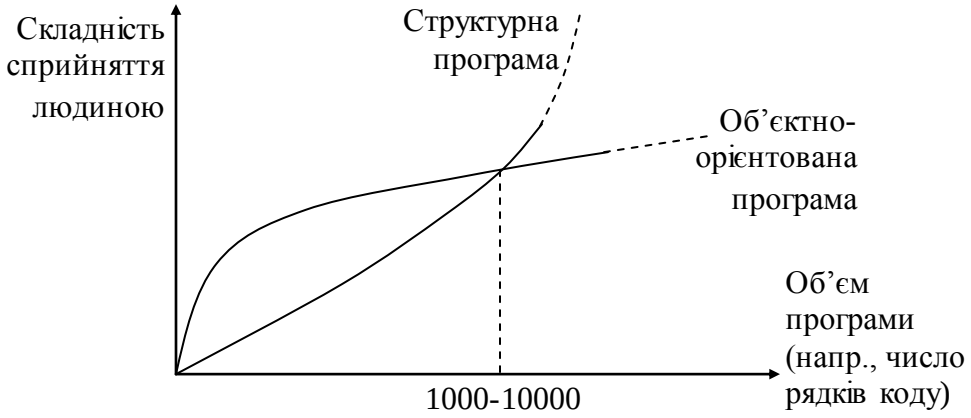


Рис. 4.3. Порівняльна складність висхідного тексту двох програм, що мають однакову функціональність, але реалізованих по-різному: згідно об'єктно-орієнтованому та структурному підходам.

Відзначимо, що часто крім розглянутих міркувань, також на вибір методики програмування впливають інші чинники, наприклад, можливість майбутнього розширення функціональності, створення якомога більш зрозумілого коду (для роботи над проектом цілої команди, а не одного програміста), або просто побажання замовника застосувати найбільш сучасний підхід до програмування.

Крім розбиття (декомпозиції) всієї предметної області на об'єкти (класи) і співвідношення між ними, також ОО-підхід має на увазі дотримання трьох основних його принципів: інкапсуляція, наслідування, поліморфізм.

Під інкапсуляцією мається на увазі об’єднання даних (значення властивостей класу у деякого конкретного об’єкта) та засобів їх обробки (методи класу). Це знову ж таки зручно психологічно, так як дозволяє реалізовувати окремі завершені сутності - класи, які самі обробляють свої дані. Звернення до об’єктів цих класів відбувається за допомогою методів, що утворюють інтерфейс класу.

Наслідування дуже корисно, тому що дозволяє сильно скоротити обсяги повторюваного коду (до чого потрібно завжди прагнути при розробці будь-якого програмного забезпечення). Згідно з цим принципом виділяється клас, який має загальний набір властивостей і методів для декількох більш розширених класів. Цей клас оголошується батьком, базовим класом для декількох похідних від нього (нащадків, спадкоємців). Всі класи-нащадки успадковують від базового всі його властивості та методи, але до цього ще мають свої власні оригінальні властивості і / або методи.

Наприклад, клас Студент є похідним від класу Людина, тому що кожна людина має властивість Ім’я, Прізвище, метод Відпочити(). Однак у Студента є свої специфічні властивості і / або методи, які як раз і відрізняють його від просто Людини: СереднійБал, НомерЗаліковки, ЗдатиЕкзамен(), і т.д.

При наслідуванні іноді методи батьківського і похідного класу мають однакове призначення, але реалізовуються по-різному. Такі методи називаються перевантаженими. Наприклад, метод Відпочити() у класу Людина реалізується як відпочинок на дивані, а у класу Студент - як похід в клуб. При цьому ще раз підкреслимо, що призначення методу в обох випадках одне і той саме.

Поліморфізм [8] є можливістю деякої функції приймати об’єкти як батьківського, так і похідних класів, і вміти викликати перевантажені методи саме того класу, об’єкт якого був переданий в функцію. Слід сказати, що це досить специфічна можливість і в загальному багато програмістів використовують ОО-підхід і без звернення до поліморфізму.

Нехай, наприклад, у програмі є функція ПровестиВихідні(), припустимо яка не належить якомусь класу (хоча це не принципово). Нехай аргументом цієї функції є об’єкт класу Людина. Тоді в неї можна передавати об’єкти всіх похідних від Людини класів: Студент, Службовець, Пенсіонер, і т.д., тому що всі вони є Людиною (спадкоємці цього класу). Ясно, що ця функція повинна включати різні дії: ПрибратиКвартиру(), ПітиНаРинок(), і в тому числі Відпочити(). Так ось поліморфізм дозволяє всередині цієї функції просто вказати назву методу Відпочити(), не вказуючи якого саме класу він повинен бути викликаний, а вже в процесі виконання програми, якщо в функцію переданий об’єкт класу Студент, то викликається саме його метод Відпочити(), а якщо переданий об’єкт класу Пенсіонер, то автоматично викликається саме його метод Відпочити(), і т.д. Кажуть, що функція ПровестиВихідні() - поліморфна, і вона є такою завдяки тому, що реалізує принцип поліморфізму.

Важливими поняттями в ООП також є: статичні члени класу, абстрактні методи і класи, дружба функцій і класів, і т.д.

Грунтуючись на перерахованих особливостях двох існуючих методів програмування, вибираємо структурний підхід як більш простий, що відповідає невеликим (не промисловим) масштабам проектованого ПЗ, а також вимогам до його складу. Також структурний підхід прекрасно реалізується засобами та мовами (практично усіма), які вбудовані у популярні математичні пакети для вирішення прикладних задач (MathCad, Maple, MatLab). Таким чином, через усі перераховані причини обираємо структурний підхід у якості технології програмування, та будемо реалізовувати функціональні блоки системи керування у вигляді підпрограм.

4.2 Вибір мови програмування

Наступним кроком після обрання технології програмування традиційно є вибір мови програмування. Для реалізації тих моделей та алгоритмів, що описуються у даній роботі можна використовувати як мови програмування загального призначення, так і спеціалізовані математичні пакети (CAD-системи). Відмітимо, що у формулах, наведених вище, є присутніми чимало специфічних математичних операцій, які, звичайно, не є наявними у мовах програмування загального призначення, а потребують написання додаткового коду для своєї реалізації. В той же час у математичних пакетах такі дії реалізуються в один малий рядок. Це, наприклад, операція підсумовування по усім індексам масиву, пошук мінімального елемента у наборі, та ще багато інших.

Складні алгоритмічні конструкції, введені раніше у попередньому викладі, також можуть бути реалізовані як звичайними мовами програмування, так і вбудованими у математичний пакет засобами. Так, наприклад, у пакет MathCAD вбудована мова, яка не має окремої назви (тому може так і називатися «мова програмування MathCAD»), і яка має дуже широкі можливості по реалізації різноманітних алгоритмів та логіки програмних продуктів.

Так само може розглядатися і мова програмування Matlab, яка, окрім широкої підтримки усіх елементів структурного програмування, навіть підтримує об’єктно-орієнтовану розробку, яка, однак, не буде цікавою у даній роботі, відповідно до змісту підрозділу 4.2.

Також і ПЗ Maple має свою вбудовану мову процедурного програмування - Maple-мову (більше всього схожу на старий BASIC). Ця мова має цілком традиційні засоби структурування програм: оператори циклів, оператори умовних і безумовних переходів, оператори порівняння, логічні оператори, команди керування зовнішніми пристроями, функції користувача, процедури і т. д. Вона також включає в себе всі команди і

функції вхідної мови, їй доступні всі спеціальні оператори та функції. Багато з них є досить серйозними програмами, наприклад, для символьного диференціювання, інтегрування, розкладання в ряд Тейлора, побудови складних тривимірних графіків і т. д.

Зважаючи на специфіку задачі, що розглядається, більш доцільно обрати мову програмування не загального призначення, а спеціалізовану – для вирішення математичних задач. Конкретний вибір цілком і повністю визначається вибраним математичним пакетом, програмою, визначення якої буде виконано у наступному підрозділі.

4.3 Вибір засобів розробки

Таким чином, слід обґрунтувати вибір одного із декількох популярних засобів розробки для вирішення математичних задач. У попередньому підрозділі вже були названі основні програмні продукти, що є найбільш популярними для вирішення такого класу задач, а саме:

- MathCAD від MathSoft;
- Maple від компанії Waterloo Maple Inc.;
- Matlab від компанії MathWorks;
- Mathematica від Wolfram Research.

Відмітимо, що останні два пакети мають досить високий поріг входження у продуктивну роботу. В першу чергу це викликано їх надзвичайно широкою функціональністю, від якої у непідготовленого користувача просто «розбігаються очі» при перших кроках при роботі з даним ПЗ. Можна сказати, що для цілей даного дослідження їх функціональність є надмірною.

Пакет Maple має досить серйозний нахил у бік символьних розрахунків, що не є необхідним у даному дослідженні. Таким чином, найбільш зручним програмним продуктом, за допомогою якого можна реалізувати розрахунки по даній роботі, є пакет від MathSoft – MathCAD.

Перевагами його є наявність досить функціональної вбудованої мови програмування, достатній рівень наповненості готовими програмами для стандартних математичних розрахунків, а також і наявність безкоштовних варіантів ліцензії для навчальних цілей (дану роботу можна віднести до навчальних робіт).

Таким чином, обираємо пакет MathCAD у якості основного програмного засобу для моделювання системи автоматичного керування групою БПЛА відповідно до розроблених у даній роботі алгоритмів та підходів.

4.4. Особливості алгоритмічних рішень для керування польотом групи БПЛА

Таким чином, на базі усіх вище запропонованих рішень слід виробити конкретні алгоритми для безпечного та ефективного польоту групи безпілотних літальних апаратів. Вхідними даними будуть:

- задана траєкторія польоту у вигляді (1.1);

- кількість апаратів N у групі;

- характерні розміри (максимальний радіус, як показано на рис. 1.1) одного апарату r_a;

- значення безпечної відстані Δ_1 між найбільш віддаленою від центра точкою апарату та перешкодами (за умовчанням може бути прийнята використовувана вище методика подвійного запасу, тобто дану відстань можна прийняти на рівні $\Delta_1 = 2r_a$);

- значення безпечної відстані Δ_2 між границею групи та перешкодами (за умовчанням аналогічно попередньому пункту, може бути прийнято $\Delta_2 = 2r_a$);

- значення безпечної відстані Δ_3 між двома апаратами (якщо воно відрізняється від прийнятого раніше співвідношення (1.4);

- оптимальна відстань d між членами групи під час її руху заданою траєкторією за умови, що будь-які перешкоди чи інші зовнішні фактори відсутні (за умовчанням тут також може бути прийнята використовувана вище методика подвійного запасу, тобто дану відстань можна прийняти на рівні $d = 2\Delta_3$).

Інформацію про просторові перешкоди, близькі до траєкторії, та отвори, через які проходить траєкторія, можуть бути отримані групою в результаті попередньої розвідки, або за допомогою використання інфразвукових датчиків відстані. У будь-якому випадку, збір, обробка та представлення інформації про просторові сторонні об'єкти не є предметом дослідження у даній роботі, тому цю інформацію будемо вважати відомою у вигляді відповідних координат та відстаней. Задачею даної роботи є вироблення адекватної реакції членів групи (враховуючи їх кількість N, що є нефіксованою) на такі сторонні об'єкти та їх безпечний проліт.

Координати перехоплювача в задачі ухилення так само вважатимемо відомими, оскільки задача динамічного розпізнавання сторонніх рухомих об'єктів, причому із виконанням їх класифікації по шкалі небезпечності (щоби не брати до уваги рухомі об'єкти, які не несуть певних загроз), сама по собі є досить складною, якій може бути присвячене окреме науково-технічне дослідження. Таким чином, як і у попередньому випадку, у даній роботі задачею є вироблення адекватних реакцій групою для оптимального прольоту перехоплювача відповідно до його поведінки (координат).

При відсутності будь-яких перешкод чи інших збурюючих факторів (як-то перехоплювача), група має рухатися із певною оптимальною (крейсерською) швидкістю $u_к$ по заданій траєкторії. При цьому усі члени групи повторюють траєкторію один одного, але із певним зсувом у просторі. Між членами групи в середньому утримуються відстані, наближені до значення d.

Навіть для такої простої задачі має бути розроблений алгоритм розміщення БПЛА по конкретним місцям у кулеподібному елементі

простору, який займає група в даний момент часу. В першу чергу, оцінці підлягає радіус групи r_g. Він може бути приблизно оцінений через кількість елементів у групі та оптимальну відстань між кожними двома елементами (приймаючи, що БПЛА розміщуються у вузлах простої кубічної решітки):

$$V_g = \frac{4}{3}\pi r_g^3 \rightarrow r_g = \sqrt[3]{\frac{3V_g}{4\pi}}$$

$$V_g = NV_{комір} = N\Delta_3^3$$

$$r_g = \sqrt[3]{\frac{3N\Delta_3^3}{4\pi}} = \Delta_3 \sqrt[3]{\frac{3N}{4\pi}}. \qquad (4.1)$$

Наприклад, якщо скористатися (1.4) та вважати, як було прийнято раніше, що r_a = 0,1 м, то радіус групи при 100 апаратах буде рівним:

$$r_g = \sqrt[3]{\frac{3N\Delta_3^3}{4\pi}} = 4 \cdot 0{,}1 \cdot \sqrt[3]{\frac{3 \cdot 100}{4\pi}} = 1{,}2 \text{ м},$$

що є цілком задовільним як з технічної, так і організаційної токи зору.

Як було зазначено раніше (підрозділ 2.2), у кожного БПЛА має бути наявним ідентифікаційний код, який може бути рівним просто номеру апарата у групі, тобто числу від 1 до N, включаючи ці межі. Відповідно до цього номеру апарати можуть розміщуватися по об'єму кулі радіуса r_g. У найбільш передній точці кулі, що лежить строго на траєкторії, завжди розміщуватимемо один апарат з найменшим номером. Наступні апарати будемо розміщувати у перерізі групи, що лежить на відстані Δ_3 від першого апарату. Радіус цього перерізу обчислюємо по формулі:

$$r_1 = \sqrt{r_g^2 - (r_g - 1 \cdot \Delta_3)^2} = \sqrt{2r_g \cdot 1 \cdot \Delta_3 - 1^2 \cdot \Delta_3^2} = \sqrt{\left(2r_g - 1 \cdot \Delta_3\right) \cdot 1 \cdot \Delta_3}$$

І взагалі радіус кожного наступного s-того перерізу обчислюємо за формулою аналогічного типу:

$$r_s = \sqrt{r_g^2 - (r_g - s \cdot \Delta_3)^2} = \sqrt{2r_g \cdot s \cdot \Delta_3 - s^2 \cdot \Delta_3^2} = \sqrt{\left(2r_g - s \cdot \Delta_3\right) \cdot s \cdot \Delta_3}, \quad (4.2)$$

де s – номер перерізу, який змінюється у межах від 1 до значення, при якому підкореневий вираз ставатиме від'ємним.

У кожному перерізі апарати розміщуються по вузлам сітки, де центральний вузол перерізу міститься на траєкторії, а інші апарати розміщуються у цілочисельних точках двовимірної декартової системи координат, масштаб якої по обом осям є однаковим і рівним відстані Δ_3. Власне заповнення виконується простим перебором усіх точок з координатами в діапазоні:

$$\text{від } -[r_s/\Delta_3] \text{ до } +[r_s/\Delta_3], \tag{4.3}$$

де квадратними дужками позначено цілу частину від дробового виразу. В процесі перебору перевіряється виконання умови, що дана точка потрапляє у коло радіуса r_s, і, якщо це не так, то вона відкидається.

Відмітимо, що при визначенні координат усіх задіяних об'єктів постійно будуть використовуватися дві системи координат:

- глобальна, координати якої пов'язані із певним фіксованим об'єктом на поверхні землі (вираховуються на основі даних GPS шляхом простого зсуву початку координат у цю фіксовану обрану точку);

- місцева, початок якої міститься на траєкторії, причому у тій точці, де знаходиться центр групи у даний момент часу, а осі орієнтовані по дотичній до траєкторії (вісь *Ox* співпадає з дотичною, а *Oy* та *Oz* перпендикулярні їй).

Перехід між системами має виконуватися відповідно до правил аналітичної геометрії, наприклад, за допомогою використання кутів Ейлера.

Після початкового вишикування апаратів в межах кулеподібного об'єму, який займатиме вся група, можна починати її рух, задаючи швидкість кожного апарату у вигляді вектору із модулем u_k та напрямом, що задається одиничним вектором:

$$\vec{n}_0 = \{x'(t), y'(t), z'(t)\}. \tag{4.4}$$

При такому варіанті дій початкова форма хмари зберігатиметься і уся група буде просуватися по заданому маршруту. Алгоритм таких дій можна навести на рис. 4.4.

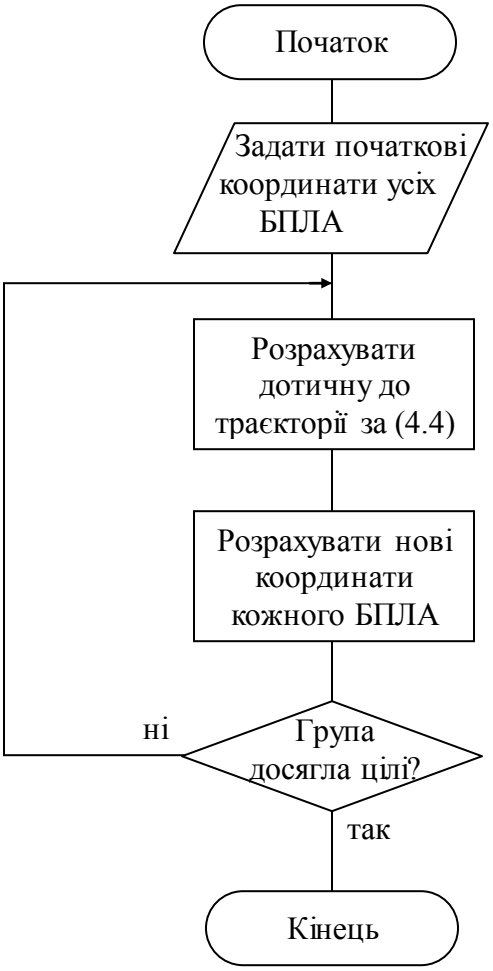


Рис. 4.4 Алгоритм керування рухом групи БПЛА заданою траєкторією у найпростішому випадку – без перешкод та інших зовнішніх збурень

Якщо в процесі руху будуть виникати ситуації, пов’язані із необхідністю прольоту близької просторової перешкоди, або прольоту крізь отвір не дуже великих розмірів, у цих випадках алгоритм має бути ускладненим і на найвищому рівні абстракції виглядатиме як на рис. 4.5.

Якщо ж приймати до уваги необхідність ухилення від ворожого перехоплювача, то алгоритм рис. 4.5 змінюється ще й таким чином, як показано на рис. 4.6.

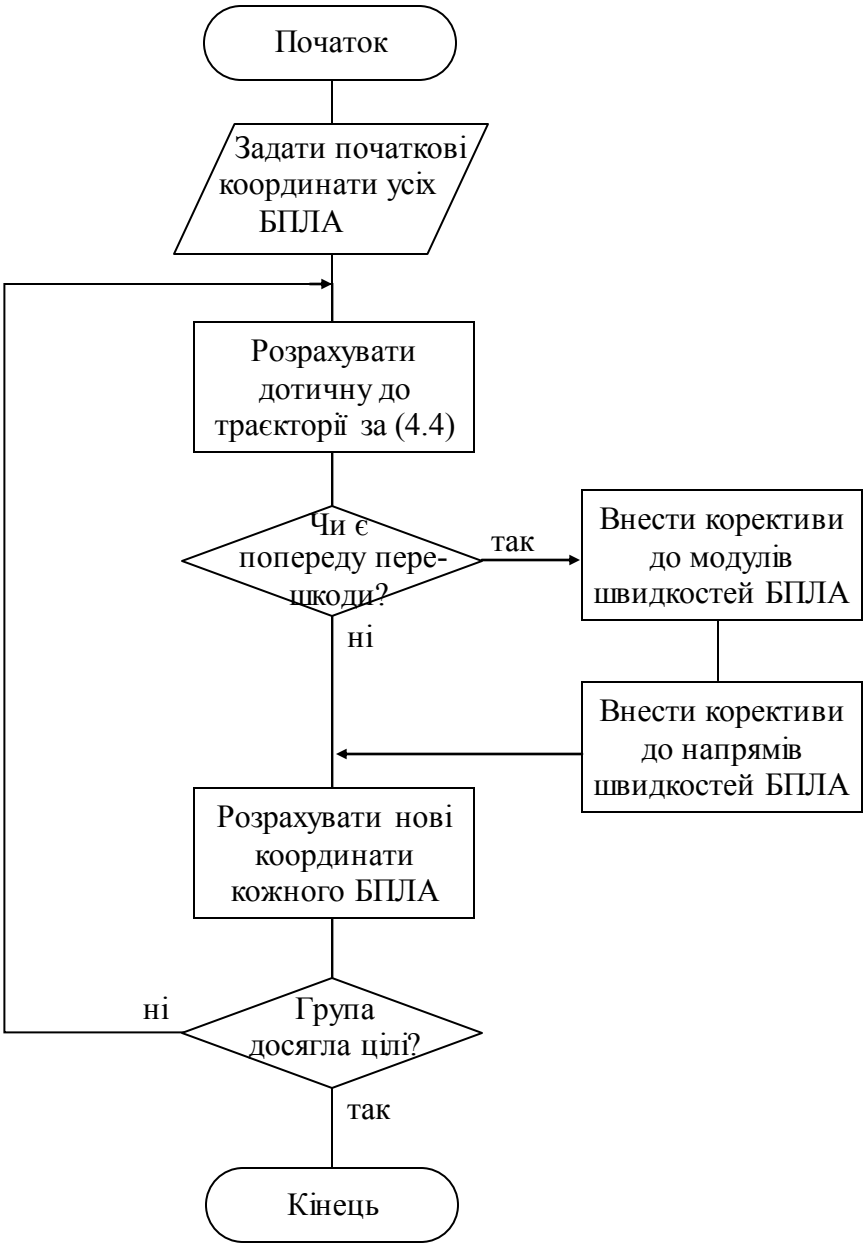


Рис. 4.5 Алгоритм керування рухом групи БПЛА заданою траєкторією з урахуванням наявних просторових перешкод поруч із траєкторією

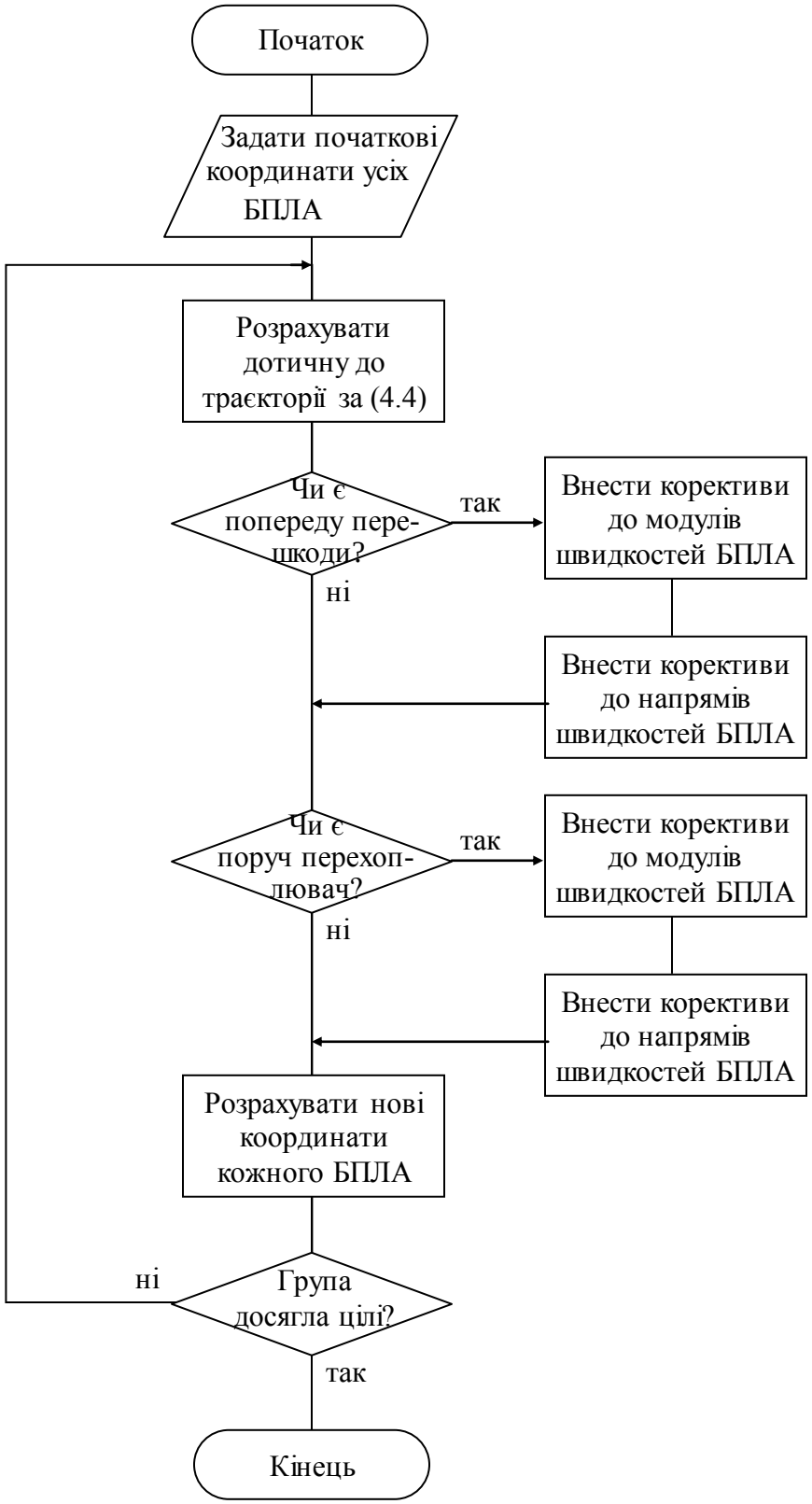


Рис. 4.6 Алгоритм керування рухом групи БПЛА заданою траєкторією з урахуванням наявних просторових перешкод поруч із траєкторією, а також наявних перехоплювачів

Відмітимо, що при деталізації вказаних алгоритмів використовуються усі попередні наробітки, описані у розділах 1-3. Це, наприклад, визначення швидкості кожного БПЛА при атаці перехоплювача на групу, недопущення зіткнень крайніх членів групи із близькими до траєкторії просторовими перешкодами, і т.п.

Використовуючи усі розроблені матеріали, можна переходити до реалізації системи керування, алгоритми якої будуть відтворені в обраній вище системі математичних розрахунків MathCAD.

5 РЕАЛІЗАЦІЯ СИСТЕМИ АВТОМАТИЧНОГО КЕРУВАННЯ ГРУПОВИМ ПОЛЬОТОМ БПЛА НА ОСНОВІ ВИКОРИСТАННЯ СКС

5.1 Проектування інтерфейсу системи

Раніше для реалізації алгоритмів роботи системи керування груповим польотом БПЛА було обрано математичний пакет MathCAD. Загальний вигляд його вікна наведено на рис. 5.1.

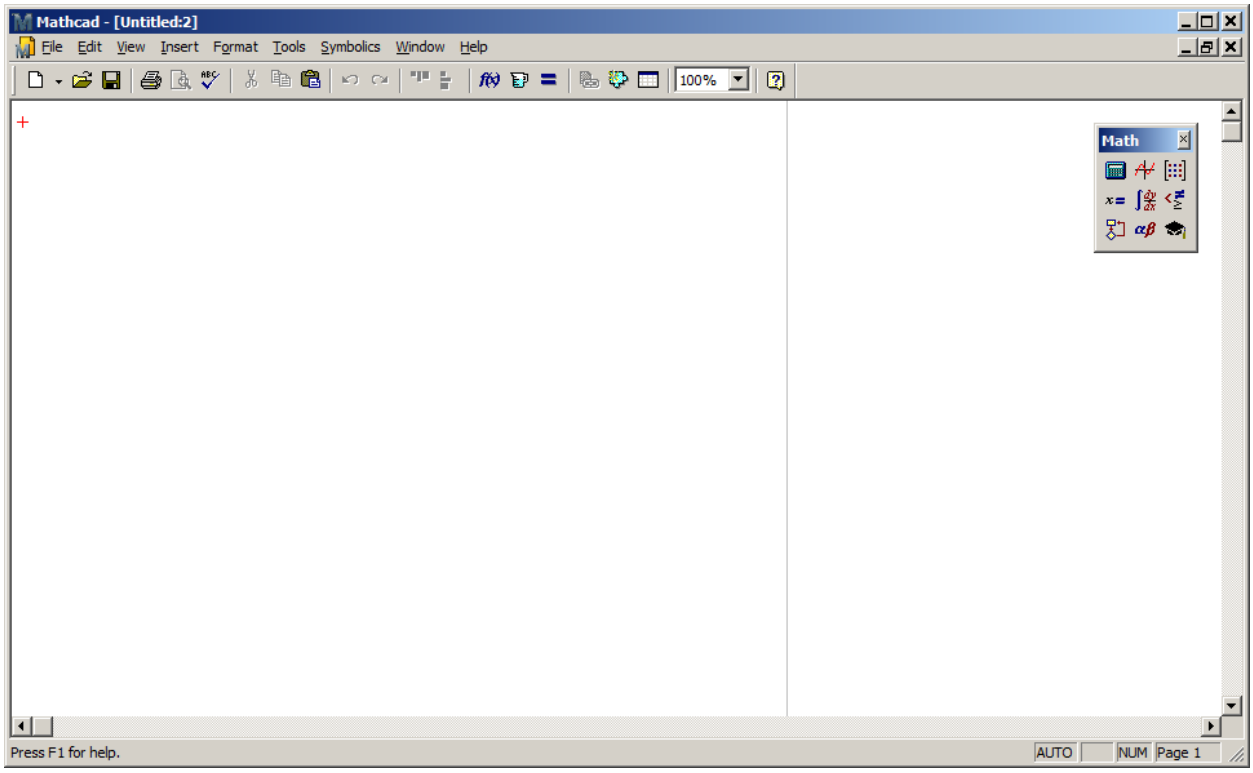


Рис. 5.1. Загальний вигляд вікна програми MathCAD, у якій реалізуються алгоритми роботи системи керування

Характерними рисами вікна є порівняно мала кількість елементів на стандартних панелях програми (взагалі зовнішній вигляд MathCAD дуже сильно може мінятися, залежно від бажання користувача, оскільки можливості програми це дозволяють). Це, в першу чергу, пов’язано із тим, що уся «математична» функціональність програми винесена у плаваючі (за умовчанням, адже їх можна прикріпити до стандартних) панелі, головною з яких є панель Math – рис. 5.1, справа вгорі. Обираючи різні елементи цієї

панелі, можна викликати інші (більш специфічні) панелі, які після виконання чергової операції найчастіше всього закривають, тримаючи весь час видимою лише панель Math. Якщо цю панель випадково буде закрито, її, а також і всі інші панелі, можна знову відкрити із головного меню (пункт View – Toolbars).

Таким чином, інтерфейс продукту, що розробляється у даній роботі, буде стандартним для того типу програмного забезпечення (готового математичного пакету сторонніх виробників), вибір якого було обґрунтовано раніше у підрозділі 4.3.

5.2 Особливості програмної реалізації окремих алгоритмів керування польотом групи БПЛА

В результаті реалізації алгоритмів керування польотом групи БПЛА було розроблено програмний продукт, що надає наочну візуалізацію положення членів групи – рис. 5.2. Система MathCAD дозволяє в динаміці спостерігати за роботою системи керування.

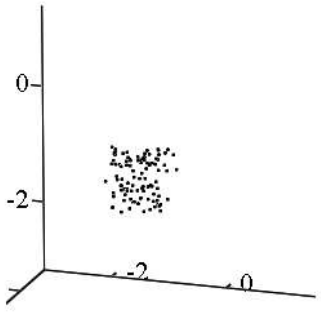


Рис. 5.2 Приклад візуалізації положення групи (при N = 100) БПЛА при русі заданою траєкторією в один певний момент часу

5.3 Розробка документаційного забезпечення системи автоматичного керування груповим польотом БПЛА

Для будь-якого програмного продукту, що розробляється, зазвичай поставляється комплект супутньої документації, серед якої можна виділити:

- керівництво користувача;
- керівництво адміністратора;
- керівництво по інсталяції (встановленню) продукту;
- інша додаткова технічна документація.

5.4 Тестування розробленої системи автоматичного керування та аналіз результатів її роботи

Тестування роботи створеного програмного продукту проводилося у двох напрямках:

- пошук та виправлення системних помилок, що викликають видимі збої (повідомлення про помилку) у роботі створених програм;
- відловлювання логічних помилок, які не викликають видимих збоїв або якихось повідомлень, але в той же час сприяють отриманню кінцевого результату, що не відповідає (або не у повній мірі відповідає) дійсності.

До першого пункту можна віднести такі види помилок, як відсутність джерел введення даних, ділення на нуль, вихід індексу за межі допустимого діапазону та звернення до неіснуючих елементів масивів, і т.п. Помилки другого типу виникають при неправильній реалізації алгоритмів програмістами (або навіть при проектуванні самого алгоритму).

Для уникнення помилок першого типу потрібно певний, не дуже малий час пропрацювати із створеним програмним забезпеченням, користуючись усіма впровадженими у нього функціональними можливостями. При цьому для тестування кожної функції ПЗ бажано використовувати різні набори вхідної інформації, аби зробити весь процес якомога більш різноманітним, і

при цьому не обов'язково аналізувати результати, що надає програма, а варто лише виявляти повідомлення про помилки, що може викликати та, чи інша комбінація вхідних даних.

У розробленій програмі було здійснено тестування із різними значеннями констант та параметрів процесу руху групи БПЛА, яке показало, що продукт є стабільним та не викликає системних, чи якихось інших помилок у роботі комп'ютера.

Для виявлення другого типу помилок необхідно мати набір вхідних даних та відповідних ним вихідних, на якому перевіряється адекватність результатів роботи ПЗ. Або (як у нашому випадку) результати роботи можуть оцінюватися експертним шляхом. Цей варіант є більш прийнятним, коли результат важко оцінити чисельним шляхом (наприклад, результатом є торт, створений автоматизованою системою-поваром, або малюнок-візерунок і т.п.). У нашому випадку результатом є комплексний результат:

- формально можна оцінювати такі параметри, як зіткнення двох апаратів, зіткнення апарату із близькою просторовою перешкодою (або границями отвору, через який пролітає група), а також контакт апарату групи із перехоплювачем;

- важко піддається формалізації (тобто її слід оцінювати «на око», експертним чином) поведінка групи апаратів в цілому, а саме, скупченість при русі по траєкторії, узгодженість дій окремих апаратів за весь час польоту, утримування форми та розмірів групи, і т.д.

Тестування, проведені над програмним забезпеченням, показали, що система керування забезпечує наступні результати:

- злагоджений рух апаратів групи вздовж траєкторії руху на великих відстанях (що на порядки перевищують лінійні розміри одного апарату);

- відсутність зіткнень апаратів групи один з одним протягом тривалого руху вздовж траєкторії (на відстані, що на порядки перевищують лінійні розміри одного апарату);

- відсутність зіткнень із просторовими перешкодами, що зустрічаються поруч із траєкторією польоту на відстанях порядку до 0,5-1 розмірів групи;

- імітацію поведінки ухилення від перехоплювача, що ще потребує удосконалення ефективності даної тактики.

Таким чином, можна констатувати, що розроблене програмне забезпечення працює відповідно до мети роботи та окресленої на початку розробки функціональності.

5.5. Висновки по розділу

У даному розділі здійснено реалізацію програмного продукту, що побудований за алгоритмами роботи системи автоматичного керування груповим польотом БПЛА на основі використання спільного комунікаційного середовища. Встановлено, що робота програми є ефективною, не викликає системних та не має логічних помилок, і адекватно виконує поставлену перед нею задачу.

6 ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ СИСТЕМИ АВТОМАТИЧНОГО КЕРУВАННЯ ГРУПОВИМ ПОЛЬОТОМ БПЛА НА ОСНОВІ ВИКОРИСТАННЯ ОПТИЧНОГО РОЗПІЗНАВАННЯ ТА РАДІО ПОЗИЦІЮВАННЯ

При початку будь-якої розробки традиційно проводиться її техніко-економічне обґрунтування (ТЕО), тобто встановлюються та порівнюються фінансові показники, які в першу чергу, відповідають на два питання:

- скільки коштів слід затратити на створення розробки (проектування, реалізацію, дослідну експлуатацію, введення у виробництво);

- скільки коштів можна отримати від реалізації запланованих обсягів готової продукції, що буде створена відповідно до даної розробки.

Таким чином, перше питання полягає у визначенні затрат на виконання дослідження. Слід відмітити, що окрім економічного ефекту існує і ряд інших позитивних наслідків, що несе виконання тієї, чи іншої розробки, і які часто не враховують, обмежуючись виконанням ТЕО. Справа в тому, що деякі розробки можуть мати не економічний, а інші види ефектів, а саме:

- соціальний ефект (наприклад, підвищення рівня освіченості населення, покращення морально-психологічних якостей окремих груп людей, і т.п.);

- політичний ефект (наприклад, розробка нової ефективної зброї, навіть без її жодного застосування в реальних обставинах, сприяє підвищенню політичної ваги країни, стримуванню агресії потенційного супротивника, тощо);

- екологічний ефект (наприклад, очищення повітря у певних регіонах чи окремих зонах);

- і т.п.

Усі перелічені види ефектів настільки важко оцінити у фінансових показниках, що при будь-якій спробі проведення відповідного точного

економічного розрахунку, він одразу може бути поставлений під сумнів, що повністю нівелює цінність та вагомість самих таких розрахунків.

В нашому випадку розробка ведеться в рамках магістерського дослідження, тому головним результатом тут є не фінансова мотивація, а отримання диплому магістра. Отже, затрати на оплату праці розробника можна вважати нульовими. Дослідження не потребує спеціального дорогого технічного обладнання, окрім наявного у розробника ПК із усім встановленим на ньому ПЗ. Таким чином, затрати на розробку системи можуть вважатися малими, так що ними можна знехтувати.

Друга оцінка має бути присвячена вигоді, яку отримає компанія, що буде використовувати систему керування, яка створена у даній роботі. Слід відмітити, що її використання, по-перше, взагалі робить можливим автоматичний рух групи апаратів без необхідності залучення людей-операторів для кожного апарату окремо. По-друге, навіть при використанні праці операторів існує ризик зіткнення апаратів, що не можливо (дуже малоймовірно) при автоматичному керуванні та використанні відповідних датчиків та підсистем. Таким чином, за один політ групою із $N = 100$ літальних апаратів тривалістю 20 хвилин зекономлена вартість оплати праці людей-операторів (у 2020 році мінімальна погодинна оплата праці складає 28,31 грн за годину) складе:

28,31 грн/год * 1/3 год * 100 людей = 940 грн.

При втраті апарату разові збитки можуть скласти до 100 умовних одиниць, що є досить високим порогом, узятим із певним запасом, для малих БПЛА розмірами порядку 0,1 м.

Приймаючи до уваги наведені міркування (в першу чергу про мізерно малі витрати на розробку системи керування групою БПЛА, яка виконується в рамках написання магістерської дисертації, а також про вивільнення великої кількості робочого часу людей-операторів, працю яких можна замінити автоматичним керуванням; і це не говорячи вже про те, що організаційно надзвичайно складно організувати узгоджену роботу N людей-

операторів навіть при N рівному кількому десяткам, тим більше – сотням) можна із впевненістю констатувати, що проектування, реалізація та впровадження даної розробки, що створена в рамках виконання магістерського дослідження, є цілком актуальною та економічно обґрунтованою.

ВИСНОВКИ

Таким чином, у даній роботі проведено розробку системи автоматичного керування групи БПЛА на основі оптичного розпізнавання та радіопозиціювання. Створена система створює можливості для безпечного та результативного руху групи БПЛА, чого важко (а при значній кількості елементів у групі і взагалі неможливо) добитися при використанні праці численної команди людей-операторів.

По-перше, у роботі проведено докладний аналіз геометричних та фізичних параметрів процесу руху групи літальних апаратів заданою траєкторією. Встановлено, що на відміну від переміщення усамітненого апарату, при русі групи існує чимало особливостей, які, в першу чергу полягають у можливості зіткнень апаратів один з одним, а також із об'єктами навколишнього середовища, що були б безпечно пройдені при польоті даною траєкторією одного апарату. Надано конкретні рекомендації, як повинні діяти члени групи у подібних ситуаціях. Також розглянуто алгоритм поведінки БПЛА групи при наявності поруч одного точкового перехоплювача. Основою цих алгоритмів є можливість точного визначення координат об'єктів, для чого запропоновано використовувати методи радіопозиціювання (по GPS та за допомогою локального радіообміну координатами, який ведеться усіма БПЛА по черзі – разом із передачею власного ідентифікатора) та оптичного розпізнавання (по Micro QR кодам, для підтвердження ефективності використання яких створена спеціальна програма мовою C#).

Створені підходи можуть застосовуватися при побудові систем керування рухом групи літальних апаратів (причому не тільки у тривимірному просторі, а й, частково, на двовимірній поверхні). У даній роботі розроблена математична модель руху групи БПЛА, що показала ефективність запропонованих рішень.

Слід відмітити, що дана розробка може в подальшому розвиватися шляхом залучення більш витончених алгоритмів ухилення від перехоплювачів, а також оминання просторових перешкод, що планується зробити у майбутньому.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Акишин, Б. А. Прикладные математические пакеты. Часть 1. MathCAD / Б.А. Акишин, Н.Х. Эркенов. - М.: РадиоСофт, 2009. - 132 с. 10. Васильев, А.Н. Mathcad 13 на примерах (+ CD-ROM) / А.Н. Васильев. - М.: БХВ-Петербург, 2006. - 763 с.

2. Видеопроцессоры: Новейшие разработки ведущих производителей: Matsushita, Micronas, Mitsubishi, Philips, Sanyo, STMicrotlectronics, Toshiba: Справочник (сост. Авраменко Ю.Ф.). - Москва: Высшая школа, 2004. - 256 с.

3. Доев, В. С. Сборник заданий по теоретической механике на базе Mathcad / В.С. Доев, Ф.А. Доронин. - М.: Лань, 2010. - 592 с.

4. Дьяконов, В. Mathcad 2001: учебный курс / В. Дьяконов. - М.: СПб: Питер, 2001. - 624 с.

5. Дьяконов, В.П. Справочник по MathCAD PLUS 6.0 PRO (+ дискета) / В.П. Дьяконов. - М.: СК Пресс, 1997. - 336 с.

6. Дьяконов, Владимир VisSim+Mathcad+MATLAB. Визуальное математическое моделирование / Владимир Дьяконов. - М.: Солон-Пресс, 2004. - 384 с.

7. Каганов, В. И. Радиотехника + компьютер + Mathcad. Для высших учебных заведений / В.И. Каганов. - М.: Горячая линия - Телеком, 2001. - 416 с.

8. Кирьянов, Дмитрий Самоучитель Mathcad 13 / Дмитрий Кирьянов. - М.: БХВ-Петербург, 2006. - 528 с.

9. Любчик Л.М., Костенко Ю.Т. Системы управления с динамическими моделями. – Х.: Основа, 1996. – 312 с.

9. Макаров, Е.Г. Сопротивление материалов на базе Mathcad (+ CD-ROM) / Е.Г. Макаров. - Москва: Гостехиздат, 2004. - 346 с.

10. Макаров, Евгений Инженерные расчеты в Mathcad 15. Учебный курс / Евгений Макаров. - М.: Питер, 2011. - 400 с.

11. Максфилд, Б. MathCAD в инженерных расчетах (+ CD-ROM) / Б. Максфилд. - М.: Корона-Век, 2010. - 310 с.

12. Охорзин, В. А. Компьютерное моделирование в системе Mathcad / В.А. Охорзин. - М.: Финансы и статистика, 2006. - 144 с.

13. Охорзин, В. А. Оптимизация экономических систем. Примеры и алгоритмы в среде Mathcad / В.А. Охорзин. - М.: Финансы и статистика, 2005. - 144 с.

14 Охорзин, В. А. Прикладная математика в системе Mathcad / В.А. Охорзин. - М.: Лань, 2009. - 352 с.

15. Поршнев, С. В. Компьютерное моделирование физических процессов с использованием пакета MathCad / С.В. Поршнев. - М.: Горячая линия - Телеком, 2002. - 252 с.

16. Ракитин, В. И. Руководство по методам вычислений и приложения MATHCAD / В.И. Ракитин. - М.: ФИЗМАТЛИТ, 2005. - 264 с.

17. Фриск, В.В. Mathcad. Расчеты и моделирование цепей на ПК / В.В. Фриск. - М.: Солон-Пресс, 2006. - 588 с.

18. Черняк, А. А. Высшая математика на базе Mathcad. Общий курс / А.А. Черняк, Ж.А. Черняк, Ю.А. Доманова. - М.: БХВ-Петербург, 2004. – 608 с.

19. Черняк, А.А. Математика для экономистов на базе Mathcad / А.А. Черняк. - М.: БХВ-Петербург, 2003. - 602 с.

20. Шушкевич, Г.Ч. Введение в MathCAD 2000 / Г.Ч. Шушкевич, С.В. Шушкевич. - М.: [не указано], 2001. - 913 с.

21. Щепетов, А. Г. Автоматизация инженерных расчетов в среде Mathcad / А.Г. Щепетов. - М.: Стандартинформ, 2006. - 264 с.

22. Yu Andrjejew, D Breslavsky, S Pashchenko, O Tatarinova. Development the Algorithms of Anthropomorphic Robot's Motion Control by Use of AI Algorithms // 2020 IEEE KhPI Week on Advanced Technology (KhPIWeek). – 2020.

Додаток. Код розробленого програмного забезпечення.

1.

```
package com.example.myapplication;
import android.content.Context;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import org.json.JSONException;
import org.json.json_objectect;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.text.date_format;
import java.text.ParseException;
import java.text.Simpledate_format;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.Locale;


   public class App {
        private static App instance;
        private Database_helper my_database_helper;
        private SQLiteDatabase my_database;
        private User my_logged_user;
        private String my_user_password_hash;
        private Message my_current_message;
        private static SharedPreferences shared_preferences;
        private ArrayList<Message> messages_list;
        private ArrayList<Message> deleted_messages_list;
        public static final date_format date_format = new
Simpledate_format("HH:mm:ss dd.MM.yyyy", Locale.ENGLISH);
        public static final String LOG_TAG = "App";
        public static final int NOTE_EDIT_OK = 5;
        public static final int LOGIN_OK = 7;

        private App() { }

        public void init(Context context, SharedPreferences
prefs) {
            Log.d(LOG_TAG, "init started");
            shared_preferences = prefs;
            my_database_helper = new Database_helper(context);
            my_database =
my_database_helper.getWritableDatabase();
            messages_list = new ArrayList<Message>();
            deleted_messages_list = new ArrayList<Message>();
            String logged_user_string =
App.getInstance().getPref().getString("loggedUser", "");
```

```
            json_objectect json_object = null;
            if (logged_user_string.equals("")) {
                App.getInstance().set_logged_user(null);
                Log.d(LOG_TAG, "No logged user");
            } else {
                try {
                    json_object = new
json_objectect(logged_user_string);
                    int id = json_object.getInt("id");
                    String login =
json_object.getString("login");
                    String password_hash =
json_object.getString("password_hash");
                    String email =
json_object.getString("email");
                    String about =
json_object.getString("about");
                    String additional =
json_object.getString("additional");
                    App.getInstance().set_logged_user(new
User(id, login, email, about, additional));

App.getInstance().set_user_password_hash(password_hash);
                    Log.d(LOG_TAG, "set_logged_user: " +
App.getInstance().get_logged_user().get_login());
                    Log.d(LOG_TAG, "password_hash: " +
password_hash);
                } catch (Exception e) {
                    e.printStackTrace();
                }
                if (App.getInstance().get_logged_user() != null)
{
                    Log.d(LOG_TAG, "Logged user: " +
App.getInstance().get_logged_user().get_login());

App.getInstance().setmessages_list(App.getInstance().get_user_me
ssages_from_database(App.getInstance().get_logged_user()));
                    Log.d(LOG_TAG, "Loaded " +
App.getInstance().get_messages_list().size() +
                            " messages for user " +
App.getInstance().get_logged_user().get_login());
                }
            }
            Log.d(LOG_TAG, "init finished");
        }

        public static App getInstance() {
            if (instance == null) {
                instance = new App();
            }
            return instance;
        }
```

```
        public SharedPreferences getPref() {
            return shared_preferences;
        }

        public void set_logged_user(User user) {
            this.my_logged_user = user;
        }

        public User get_logged_user() { return my_logged_user;
        }

        public void set_user_password_hash(String password_hash)
{
            this.my_user_password_hash = password_hash;
        }

        public String get_user_password_hash() {
            return my_user_password_hash;
        }

        public void set_current_message(Message message) {
            this.my_current_message = message;
        }

        public Message get_current_message() {
            return my_current_message;
        }

        public ArrayList<Message> get_messages_list() {
            return messages_list;
        }

        public void setmessages_list(ArrayList<Message>
messages_list) {
            this.messages_list = messages_list;
        }

        public boolean save_message_to_database(Message message)
{
            Date date_now = Calendar.getInstance().getTime();
            String date_now_string =
date_format.format(date_now);
            if (message == null) {
                Log.e(LOG_TAG, "save_message_to_database ERROR:
message is null");
                return false;
            }
            if (message.get_id() != -1) {
                Log.e(LOG_TAG, "save_message_to_database ERROR:
message already was saved");
                return false;
            }
            my_database.execSQL("INSERT INTO messages (" +
```

```
                                "remote_id," +
                                "author_login," +
                                "create_date," +
                                "change_date," +
                                "title," +
                                "text," +
                                "additional) VALUES (?, ?, ?, ?, ?,
?, ?);",
                        new String[] {

Integer.toString(App.getInstance().get_current_message().get_rem
ote_id()),

App.getInstance().get_logged_user().get_login(),
                                date_now_string,
                                date_now_string,

App.getInstance().get_current_message().get_title(),

App.getInstance().get_current_message().get_text(),

App.getInstance().get_current_message().get_additional() });
return true;
        }

        public boolean update_message_in_database(Message
message) {
            Date date_now = Calendar.getInstance().getTime();
            String date_now_string =
date_format.format(date_now);
            if (message == null) {
                Log.e(LOG_TAG, "update_message_in_database
ERROR: message is null");
                return false;
            }
            if (message.get_id() == -1) {
                Log.e(LOG_TAG, "update_message_in_database
ERROR: message was not saved yet");
                return false;
            }
            my_database.execSQL("UPDATE messages SET " +
                                "change_date = ?," +
                                "title = ?," +
                                "text = ?," +
                                "additional = ? WHERE id = ?;",
                        new String[] {
                                date_now_string,

App.getInstance().get_current_message().get_title(),

App.getInstance().get_current_message().get_text(),

App.getInstance().get_current_message().get_additional(),
```

```
Integer.toString(App.getInstance().get_current_message().get_id(
)) }); return true;
        }

        public void refresh_messages_list_from_database() {
            App.getInstance().clear_messages_list();

App.getInstance().setmessages_list(App.getInstance().get_user_me
ssages_from_database(App.getInstance().get_logged_user()));
        }

          public boolean
save_messages_list_to_database(ArrayList<Message>
messages_to_save) {
                for (Message i : messages_to_save) {
                  if (i.get_id() != -1) {
                      return false;
                  }
        }
          for (Message i : messages_to_save) {
          my_database.execSQL("INSERT INTO messages (" +
        "remote_id," +
        "author_login," +
        "create_date," +
        "change_date," +
        "title," +
        "text," +
        "additional) VALUES (?, ?, ?, ?, ?, ?, ?);",
        new String[] {
            Integer.toString(i.get_remote_id()),
                    i.get_author_login(),
                    date_format.format(i.get_create_date()),
                    date_format.format(i.get_change_date()),
                    i.get_title(),
                    i.get_text(),
                    i.get_additional()});
    }    return true;
        }

public void addDeletedMessage(Message message) {
        Log.d(LOG_TAG, "addDeletedMessage adding message: " +
message.get_title());
        deleted_messages_list.add(message);
        }

public void clear_deleted_messages_list() {
        deleted_messages_list.clear();
        }

public ArrayList<Message> get_deleted_messages_list() {
        return this.deleted_messages_list;
        }
```

```
public boolean delete_message_by_id(int id) {
        Log.d(LOG_TAG, "delete_message_by_id started");
        for (int i = 0; i < messages_list.size(); i++) {
        if (messages_list.get(i).get_id() == id) {
        Log.d(LOG_TAG, "delete_message_by_id deleting message: "
+ messages_list.get(i).get_title());

delete_message_from_database_by_id(messages_list.get(i).get_id()
);
        messages_list.remove(i);
        return true;
        }
        }
        Log.d(LOG_TAG, "delete_message_by_id finished");
        return false;
        }

private void delete_message_from_database_by_id(int id) {
        my_database.execSQL("DELETE FROM messages WHERE id =
?;", new String[] { Integer.toString(id) });
        }

public ArrayList<Message> get_user_messages_from_database(User
user) {
        int messages_count = 0;
        ArrayList<Message> list_from_database = new
ArrayList<Message>();
        int id;
        int remote_id;
        Date create_date = null;
        Date change_date = null;
        String title;
        String text;
        String additional;
        Cursor cursor = my_database.rawQuery("SELECT " +
        "id," +
        "remote_id," +
        "create_date," +
        "change_date," +
        "title," +
        "text," +
        "additional FROM messages WHERE author_login = ?;", new
String[] { user.get_login() });
        messages_count = cursor.getCount();
        if (messages_count == 0) {
        Log.d(LOG_TAG, "Did not found messages for user " +
user.get_login());
        cursor.close();
        return list_from_database;
        }
        Log.d(LOG_TAG, "Found " + messages_count + " messages
for user " + user.get_login());
```

```
        cursor.moveToFirst();
        for (int i = 0; i < messages_count; i++) {
        id = cursor.getInt(cursor.getColumnIndex("id"));
        remote_id =
cursor.getInt(cursor.getColumnIndex("remote_id"));
        try {
        create_date =
date_format.parse(cursor.getString(cursor.getColumnIndex("create
_date")));
        change_date =
date_format.parse(cursor.getString(cursor.getColumnIndex("change
_date")));
        } catch (ParseException ex) {
        Log.d(LOG_TAG, "get_user_messages_from_database
ParseException");
        }
        title =
cursor.getString(cursor.getColumnIndex("title"));
        text = cursor.getString(cursor.getColumnIndex("text"));
        additional =
cursor.getString(cursor.getColumnIndex("additional"));
        list_from_database.add(new Message(id, remote_id, user,
create_date, change_date, title, text, additional));
        Log.d(LOG_TAG, "Message " + i + "\nid = " + id +
"\nremote_id = " + remote_id + "\ncreate_date = " +
        date_format.format(create_date) + "\nchange_date = " +
date_format.format(change_date) +
        "\ntitle = " + title + "\ntext = " + text +
"\nadditional = " + additional);
        cursor.moveToNext();
        }
        cursor.close();
        return list_from_database;
        }

public void clear_messages_list() {
        messages_list.clear();
        }

public void delete_all_messages_of_user_in_database(User user) {
        my_database.execSQL("DELETE FROM messages WHERE
author_login = ?;",
        new String[] { user.get_login() });
        }

public String get_additional_author(Message message) {
        json_objectect additional_json;
        try {
        additional_json = new
json_objectect(message.get_additional());
        String author = additional_json.getString("sender");
        if (author != "") {
        return author;
```

```
        }
        } catch (JSONException e) {
        return null;
        }
        return null;
        }

public static String get_password_hash(String password_to_hash)
{
        String generated_password = null;
        try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(password_to_hash.getBytes());
        byte[] bytes = md.digest();
        StringBuilder sb = new StringBuilder();
        for(int i = 0; i < bytes.length; i++)
        {
        sb.append(Integer.toString((bytes[i] & 0xff) + 0x100,
16).substring(1));
        }
        generated_password = sb.toString();
        }
        catch (NoSuchAlgorithmException e)
        {
        e.printStackTrace();
        }
        return generated_password;
        }
        }
```

2.

```
package com.example.myapplication;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class Database_helper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "Messages";
    public static final String LOG_TAG = "Database_helper";
    public Database_helper(Context context) {
        super(context, DATABASE_NAME, null, 1);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        Log.d(LOG_TAG, "onCreate database");
        db.execSQL("CREATE TABLE users ("
                + "id INTEGER PRIMARY KEY,"
                + "login TEXT,"
                + "email TEXT,"
                + "about TEXT,"
                + "additional TEXT" + ");");
```

```
        db.execSQL("CREATE TABLE passwords (id INTEGER PRIMARY
KEY," +
                "user_id INTEGER," +
                "password_hash TEXT" + ");");
        db.execSQL("CREATE TABLE messages ("
                + "id INTEGER PRIMARY KEY,"
                + "remote_id INTEGER,"
                + "author_login TEXT,"
                + "create_date TEXT,"
                + "change_date TEXT,"
                + "title TEXT,"
                + "text TEXT,"
                + "additional TEXT" + ");");
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
    }
}
```

3.

```
package com.example.myapplication;
import android.content.Intent;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;
import android.widget.Toast;
import org.json.JSONException;
import org.json.json_objectect;

public class Edit_message_activity extends AppCompatActivity {
    private EditText my_title;
    private EditText my_text;
    public static final String LOG_TAG = "App";
    Intent intent;
    @Override
    protected void onCreate(Bundle saved_instance_state) {
        super.onCreate(saved_instance_state);
        setContentView(R.layout.activity_edit_message);
        intent = getIntent();
        this.set_title(intent.getStringadditional("title"));
        my_title = (EditText) findViewById(R.id.message_title);
        my_text = (EditText) findViewById(R.id.message_text);

my_title.set_text(App.getInstance().get_current_message().get_ti
tle());

my_text.set_text(App.getInstance().get_current_message().get_tex
t());
```

```
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.edit_message, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
// Handle action bar item clicks here. The action bar will
// automatically handle clicks on the Home/Up button, so long
// as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemy_id();
        switch (id) {
            case R.id.action_save: // сохранение заголовка и
текста заметки из полей Activity
App.getInstance().get_current_message().set_title(my_title.get_t
ext().toString());

App.getInstance().get_current_message().set_text(my_text.get_tex
t().toString()); // если заметка уже была сохранена в удаленной
базе данных
                json_objectect additional_json;
                try {
                    additional_json = new
json_objectect(App.getInstance().get_current_message().get_addit
ional());
                    Log.d(LOG_TAG, "got json object from
additional field");
                    additional_json.put("update", true);
                    Log.d(LOG_TAG, "writing json data to
additional field");

App.getInstance().get_current_message().set_additional(additiona
l_json.toString());
                } catch (JSONException e) {
                    try {
                        Log.d(LOG_TAG, "creating new json
object");
                        additional_json = new json_objectect();
                        additional_json.put("update", true);
                        Log.d(LOG_TAG, "writing json data to
additional field");

App.getInstance().get_current_message().set_additional(additiona
l_json.toString());
                    } catch (JSONException ee) {
                        ee.printStackTrace();
                        Log.e(LOG_TAG, "ERROR: cannot set data
to additional field"); }
                }
        }
```

```
App.getInstance().save_message_to_database(App.getInstance().get
_current_message());
    } else {

App.getInstance().update_message_in_database(App.getInstance().g
et_current_message()); }
Main_activity.getAdapter().notifyDataSetChanged();
this.finish();
return true;
case R.id.action_cancel:
            App.getInstance().set_current_message(null);
this.finish();
return true;
    default:
            break;
}
return super.onOptionsItemSelected(item);
        }
        }
```

4.

```
package com.example.myapplication;
import android.os.AsyncTask;
import android.util.Log;
import org.json.json_objectect;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.SocketTimeoutException;
import java.net.URL;
import java.net.URLEncoder;
import java.util.HashMap;
import java.util.Map;

public class Http_post_request extends AsyncTask<json_objectect,
Void, String> {
    public static final String LOG_TAG = "Http_post_request";
    private String host = "http://177.201.73.217:3128";
    @Override
    protected String do_in_background(json_objectect... params)
{
        Log.d(LOG_TAG, "do_in_background started");
        URL url;
        String response = "";
        try {
            url = new URL(host);
            Log.d(LOG_TAG, "url.openConnection()...");
```

```
            HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
            conn.setReadTimeout(1000); // TODO: set timeouts to
15000
            conn.setConnectTimeout(1000);
            conn.setRequestMethod("POST");
            conn.setRequestProperty("Content-Type",
"application/json; charset=UTF-8");
            conn.setDoInput(true);
            conn.setDoOutput(true);
            Log.d(LOG_TAG, "conn.getOutputStream()...");
            OutputStream os = conn.getOutputStream();
            BufferedWriter writer = new BufferedWriter(new
OutputStreamWriter(os, "UTF-8"));
            Log.d(LOG_TAG, "writer.write()...");
            writer.write(params[0].toString());
            Log.d(LOG_TAG, "writer.flush()");
            writer.flush();
            writer.close();
            os.close();
            Log.d(LOG_TAG, "conn.getResponseCode()");
            int response_code = conn.getResponseCode();
            if (response_code == HttpURLConnection.HTTP_OK) {
                Log.d(LOG_TAG, "reading input stream");
                String line;
                BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
                while ((line = reader.readLine()) != null) {
                    response += line;
                }
            }
        } catch (SocketTimeoutException e) {
            return "SocketTimeoutException";
        } catch (Exception e) {
            e.printStackTrace();
        }
        Log.d(LOG_TAG, "do_in_background finished");
        return response;
    }
}
```

5.

```
package com.example.myapplication;
import android.app.Application;
import android.util.Log;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.json_objectect;
import java.net.SocketTimeoutException;
import java.util.ArrayList;
import java.util.List;

public class Internet_database {
```

```
    public static final String LOG_TAG = "Internet_database";
    public static User find_user_in_database_by_login(String
login) throws SocketTimeoutException {
        Log.d(LOG_TAG, "find_user_in_database_by_login()
started");
        json_objectect request = new json_objectect();
        json_objectect data = new json_objectect();
        try {
            request.put("message",
"find_user_in_database_by_login");
            data.put("login", login);
            request.put("data", data);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
        Log.d(LOG_TAG, "find_user_in_database_by_login() JSON
object made");
        Http_post_request post = new Http_post_request();
        Log.d(LOG_TAG, "find_user_in_database_by_login()
Http_post_request object executing");
        post.execute(request);
        String http_response;
        Log.d(LOG_TAG, "find_user_in_database_by_login() trying
to get response");
        try {
            http_response = post.get();
        } catch (Exception e) {
            http_response = "exception in
find_user_in_database_by_login";
            e.printStackTrace();
        }
        if (http_response.equals("SocketTimeoutException")) {
            throw new SocketTimeoutException();
        }
        Log.d(LOG_TAG, "find_user_in_database_by_login()
response got");
        Log.d(LOG_TAG, "HTTP POST RESPONSE:\n" + http_response);
        json_objectect json_response = null;
        Log.d(LOG_TAG, "find_user_in_database_by_login() trying
to get JSON response object");
        try {
            json_response = new json_objectect(http_response);
            Log.d(LOG_TAG, "find_user_in_database_by_login()
JSON response object made");
            if
(json_response.getString("message").equals("userFound")) {
                json_objectect json_data =
json_response.getjson_objectect("data");
                return new User(json_data.getInt("id"),
json_data.getString("login"),
```

```
                        json_data.getString("email"),
json_data.getString("about"),
json_data.getString("additional"));
            }
            return null;
        } catch (Exception e) {
            e.printStackTrace(); return null;
        }
    }

    public static boolean create_user_in_database(User user,
String password) {
        Log.d(LOG_TAG, "create_user_in_database started");
        json_objectect request = new json_objectect();
        json_objectect data = new json_objectect();
        try {
            request.put("message", "create_user_in_database");
            data.put("id", user.get_id());
            data.put("login", user.get_login());
            data.put("password_hash",
App.get_password_hash(password));
            data.put("email", user.get_email());
            data.put("about", user.get_about());
            data.put("additional", user.get_additional());
            request.put("data", data);
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
        Log.d(LOG_TAG, "create_user_in_database JSON object
created");
        Http_post_request post = new Http_post_request();
        Log.d(LOG_TAG, "create_user_in_database
Http_post_request executing...");
        post.execute(request);
        String http_response;
        Log.d(LOG_TAG, "create_user_in_database trying to get
result of Http_post_request");
        try {
            http_response = post.get();
        } catch (Exception e) {
            http_response = "exception in
create_user_in_database";
            e.printStackTrace();
        }
        Log.d(LOG_TAG, "HTTP POST RESPONSE:\n" + http_response);
        json_objectect json_response = null;
        try {
            json_response = new json_objectect(http_response);
            if
(json_response.getString("message").equals("userCreatedSuccessfu
lly")) {
                return true;
```

```
            }
            return false;
        } catch (Exception e) {
            e.printStackTrace(); return false;
        }
    }

    public static boolean check_user_password(User user, String
password) {
        json_objectect request = new json_objectect();
        json_objectect data = new json_objectect();
        try {
            request.put("message", "checkUserpassword_hash");
            data.put("login", user.get_login());
            data.put("password_hash",
App.get_password_hash(password));
            request.put("data", data);
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
        Http_post_request post = new Http_post_request();
        post.execute(request);
        String http_response;
        try {
            http_response = post.get();
        } catch (Exception e) {
            http_response = "exception in check_user_password";
            e.printStackTrace();
        }
        Log.d(LOG_TAG, "HTTP POST RESPONSE:\n" + http_response);
        json_objectect json_response = null;
        try {
            json_response = new json_objectect(http_response);
            if
(json_response.getString("message").equals("passwordMatch")) {
                return true;
            }
            return false;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }

    public static boolean sync_messages_with_database(User user,
String password_hash) {
        Log.d(LOG_TAG, "sync_messages_with_database started");
        if (App.getInstance().get_messages_list() != null &&
App.getInstance().get_deleted_messages_list() != null) {
if (App.getInstance().get_deleted_messages_list().size() != 0) {
            Log.d(LOG_TAG, "sync_messages_with_database
deleted_messages_list is not empty");
```

```
            ArrayList<Message> messages_to_delete = new
ArrayList<Message>();
            for (Message i :
App.getInstance().get_deleted_messages_list()) {
                if (i.get_remote_id() != -1) {
messages_to_delete.add(i);
                    Log.d(LOG_TAG, "sync_messages_with_database
delete from remote db: " + i.get_title());
                }
            }
            Log.d(LOG_TAG, "sync_messages_with_database
messages_to_delete size is " + messages_to_delete.size());
            if (messages_to_delete.size() != 0) {
                return false;
            }
        }
        Log.d(LOG_TAG, "sync_messages_with_database clearing
deleted_messages_list");
        App.getInstance().clear_deleted_messages_list();
    }
    ArrayList<Message> messages_to_update = new
ArrayList<Message>();
for (Message i : App.getInstance().get_messages_list()) {
        json_objectect additional_json;
        try {
            additional_json = new
json_objectect(i.get_additional());
            if (additional_json.getBoolean("update")) {
                messages_to_update.add(i);
                additional_json.put("update", false);
                i.set_additional(additional_json.toString());
                Log.d(LOG_TAG, "sync_messages_with_database
change message " + i.get_title());
            }
        } catch (JSONException e) {
            continue;
        }
    }
if (messages_to_update.size() != 0) {
        if
(!Internet_database.update_user_messages_in_database(user,
password_hash, messages_to_update)) {
            return false;
        }
    }
for (Message i : App.getInstance().get_messages_list()) {
        if (i.get_remote_id() == -1) {
messages_to_send.add(i);
        }
    }
if (messages_to_send.size() != 0) {
        Log.d(LOG_TAG, "sync_messages_with_database saving " +
messages_to_send.size() + " messages to remote db");
```

```
        if
(!Internet_database.save_user_messages_to_database(user,
password_hash, messages_to_send)) {
            return false;
        }
    }
if (messages_from_remote_database != null) {
        if (messages_from_remote_database.size() != 0) { // если
в удаленной базе данных были заметки Log.d(LOG_TAG, "There are "
+ messages_from_remote_database.size() + " messages in remote db
for current user");
App.getInstance().save_messages_list_to_database(messages_from_r
emote_database);
        }
    } else {
        Log.e(LOG_TAG, "ERROR: failed to load messages list from
remote db");
        return false;
    }
} else {
        Log.e(LOG_TAG, "ERROR. messages_list is NULL and/or
deleted_messages_list is NULL");
        }
        Log.d(LOG_TAG, "sync_messages_with_database finished");
        return true;
        }

public static boolean delete_user_messages_from_database(User
user, String password_hash, ArrayList<Message>
messages_to_delete) {
        Log.d(LOG_TAG, "delete_user_messages_from_database
start");
        json_objectect request = new json_objectect();
        json_objectect data = new json_objectect();
        JSONArray messages_array_json = new JSONArray();
        Log.d(LOG_TAG, "delete_user_messages_from_database
login: " + user.get_login() + "; password_hash: " +
password_hash);
        try {
        request.put("message",
"delete_user_messages_from_database");
        data.put("id", user.get_id());
        data.put("login", user.get_login());
        data.put("password_hash", password_hash);
        data.put("email", user.get_email());
        data.put("about", user.get_about());
        data.put("additional", user.get_additional());
        for (Message i : messages_to_delete) {
        json_objectect item = new json_objectect();
        item.put("id", i.get_remote_id());
        item.put("author_login", i.get_author_login());
        item.put("create_date",
App.date_format.format(i.get_create_date()));
```

```
        item.put("change_date",
App.date_format.format(i.get_change_date()));
        item.put("title", i.get_title());
        item.put("text", i.get_text());
        item.put("additional", i.get_additional());
        messages_array_json.put(item);
        }
        data.put("messages", messages_array_json);
        request.put("data", data);
        } catch (Exception e) {
        e.printStackTrace();
        return false;
        }
        Http_post_request post = new Http_post_request();
        post.execute(request);
        String http_response;
        try {
        http_response = post.get();
        } catch (Exception e) {
        http_response = "exception in
delete_user_messages_from_database";
        e.printStackTrace();
        }
        Log.d(LOG_TAG, "HTTP POST RESPONSE:\n" + http_response);
        json_objectect json_response = null;
        try {
        json_response = new json_objectect(http_response);
        if
(json_response.getString("message").equals("messagesDeletedSucce
ssfully")) {
        return true;
        }
        return false;
        } catch (Exception e) {
        e.printStackTrace(); return false;
        }
        }

public static boolean save_user_messages_to_database(User user,
String password_hash, ArrayList<Message> messages_to_save) {
        Log.d(LOG_TAG, "save_user_messages_to_database start");
        json_objectect request = new json_objectect();
        json_objectect data = new json_objectect();
        JSONArray messages_array_json = new JSONArray();
        Log.d(LOG_TAG, "save_user_messages_to_database login: "
+ user.get_login() + "; password_hash: " + password_hash);
        try {
        request.put("message",
"save_user_messages_to_database");
        data.put("id", user.get_id());
        data.put("login", user.get_login());
        data.put("password_hash", password_hash);
        data.put("email", user.get_email());
```

```
        data.put("about", user.get_about());
        data.put("additional", user.get_additional());
        for (Message i : messages_to_save) {
        json_objectect item = new json_objectect();
        item.put("author_login", i.get_author_login());
        item.put("create_date",
App.date_format.format(i.get_create_date()));
        item.put("change_date",
App.date_format.format(i.get_change_date()));
        item.put("title", i.get_title());
        item.put("text", i.get_text());
        item.put("additional", i.get_additional());
        messages_array_json.put(item);
        }
        data.put("messages", messages_array_json);
        request.put("data", data);
        } catch (Exception e) {
        e.printStackTrace();
        return false;
        }
        Http_post_request post = new Http_post_request();
        post.execute(request);
        String http_response;
        try {
        http_response = post.get();
        } catch (Exception e) {
        http_response = "exception in
save_user_messages_to_database";
        e.printStackTrace();
        }
        Log.d(LOG_TAG, "HTTP POST RESPONSE:\n" + http_response);
        json_objectect json_response = null;
        try {
        json_response = new json_objectect(http_response);
        if
(json_response.getString("message").equals("messagesSavedSuccess
fully")) {
        return true;
        }
        return false;
        } catch (Exception e) {
        e.printStackTrace(); return false;
        }
        }

public static boolean update_user_messages_in_database(User
user, String password_hash, ArrayList<Message>
messages_to_update) {
        Log.d(LOG_TAG, "update_user_messages_in_database
start");
        json_objectect request = new json_objectect();
        json_objectect data = new json_objectect();
        JSONArray messages_array_json = new JSONArray();
```

```
        Log.d(LOG_TAG, "update_user_messages_in_database login:
" + user.get_login() + "; password_hash: " + password_hash);
        try {
        request.put("message",
"update_user_messages_in_database");
        data.put("id", user.get_id());
        data.put("login", user.get_login());
        data.put("password_hash", password_hash);
        data.put("email", user.get_email());
        data.put("about", user.get_about());
        data.put("additional", user.get_additional());
        for (Message i : messages_to_update) {
        json_objectect item = new json_objectect();
        item.put("id", i.get_remote_id());
        item.put("change_date",
App.date_format.format(i.get_change_date()));
        item.put("title", i.get_title());
        item.put("text", i.get_text());
        item.put("additional", i.get_additional());
        messages_array_json.put(item);
        }
        data.put("messages", messages_array_json);
        request.put("data", data);
        } catch (Exception e) {
        e.printStackTrace();
        return false;
        }
        Http_post_request post = new Http_post_request();
        post.execute(request);
        String http_response;
        try {
        http_response = post.get();
        } catch (Exception e) {
        http_response = "exception in
update_user_messages_in_database";
        e.printStackTrace();
        }
        Log.d(LOG_TAG, "HTTP POST RESPONSE:\n" + http_response);
        json_objectect json_response = null;
        try {
        json_response = new json_objectect(http_response);
        if
(json_response.getString("message").equals("messages_updated_suc
cessfully")) {
        return true;
        }
        return false;
        } catch (Exception e) {
        e.printStackTrace(); return false;
        }
        }
```

```
public static ArrayList<Message>
get_user_messages_from_database(User user, String password_hash)
{
        json_objectect request = new json_objectect();
        json_objectect data = new json_objectect();
        try {
        request.put("message",
"get_user_messages_from_database");
        data.put("id", user.get_id());
        data.put("login", user.get_login());
        data.put("password_hash", password_hash);
        data.put("email", user.get_email());
        data.put("about", user.get_about());
        data.put("additional", user.get_additional());
        request.put("data", data);
        } catch (Exception e) {
        e.printStackTrace();
        return null;
        }
        Http_post_request post = new Http_post_request();
        post.execute(request);
        String http_response;
        try {
        http_response = post.get();
        } catch (Exception e) {
        http_response = "exception in
get_user_messages_from_database";
        e.printStackTrace();
        }
        Log.d(LOG_TAG, "HTTP POST RESPONSE:\n" + http_response);
        if (http_response.equals("SocketTimeoutException")) {
        return null;
        }
        json_objectect json_response = null;
        try {
        json_response = new json_objectect(http_response);
        String response_string =
json_response.getString("message");
        if (response_string.equals("usermessages_list")) {
        JSONArray arr = json_response.getJSONArray("data");
        ArrayList<Message> messages_list = new
ArrayList<Message>();
        for (int i = 0; i < arr.length(); i++) {
        json_objectect item = arr.getjson_objectect(i);
        messages_list.add(new Message(-1, item.getInt("id"),
user,
        App.date_format.parse(item.getString("create_date")),
        App.date_format.parse(item.getString("change_date")),
        item.getString("title"), item.getString("text"),
item.getString("additional")));
        }
        return messages_list;
```

```
        } else if
(response_string.equals("usermessages_listIsEmpty")) {
        return new ArrayList<Message>();
        }
        return null;
        } catch (Exception e) {
        e.printStackTrace();
        return null;
        }
        }

public static String send_message_to_user(String receiver_login,
String sender_password_hash, Message message) {
        Log.d(LOG_TAG, "send_message_to_user start");
        if (receiver_login == null || sender_password_hash ==
null || message == null) {
        return "One or more parameters are null"; }
        try {
        additional_json = new
json_objectect(message.get_additional());
        additional_json.put("sender",
App.getInstance().get_logged_user().get_login());
        message.set_additional(additional_json.toString());
        } catch (JSONException e) {
        try {
        additional_json = new json_objectect();
        additional_json.put("sender",
App.getInstance().get_logged_user().get_login());
        message.set_additional(additional_json.toString());
        } catch (JSONException ee) {
        Log.e(LOG_TAG, "ERROR: failed to add info to additional
field");
        ee.printStackTrace();
        return
Messages_list_application.getAppContext().getString(R.string.mes
sage_send_fail);
        }
        }
        json_objectect request = new json_objectect();
        json_objectect data = new json_objectect();
        json_objectect message_json = new json_objectect();
        try {
        request.put("message", "send_message_to_user");
        data.put("receiver_login", receiver_login);
        data.put("password_hash", sender_password_hash);
        message_json.put("create_date",
App.date_format.format(message.get_create_date()));
        message_json.put("change_date",
App.date_format.format(message.get_change_date()));
        message_json.put("title", message.get_title());
        message_json.put("text", message.get_text());
        message_json.put("additional",
message.get_additional());
```

```
        data.put("message", message_json);
        request.put("data", data);
        } catch (JSONException e) {
        e.printStackTrace();
        return
Messages_list_application.getAppContext().getString(R.string.mes
sage_send_fail);
        }
        Log.d(LOG_TAG, "send_message_to_user password_hash: " +
sender_password_hash);
        Http_post_request post = new Http_post_request();
        post.execute(request);
        String http_response;
        try {
        http_response = post.get();
        } catch (Exception e) {
        http_response = "exception in
save_user_messages_to_database";
        e.printStackTrace();
        }
        Log.d(LOG_TAG, "HTTP POST RESPONSE:\n" + http_response);
        json_objectect json_response = null;
        try {
        json_response = new json_objectect(http_response);
        String response_string =
json_response.getString("message");
        if (response_string.equals("messageSentSuccessfully")) {
        return
Messages_list_application.getAppContext().getString(R.string.mes
sage_send_success);
        } else if (response_string.equals("receiverNotFound")) {
        return
Messages_list_application.getAppContext().getString(R.string.mes
sage_send_receiver_not_found);
        }
        return
Messages_list_application.getAppContext().getString(R.string.mes
sage_send_fail);
        } catch (Exception e) {
        e.printStackTrace();
        return
Messages_list_application.getAppContext().getString(R.string.mes
sage_send_fail);
        }
        }
        }
```

6.

```
package com.example.myapplication;
import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.annotation.TargetApi;
import android.app.Activity;
```

```
import android.content.Context;
import android.content.DialogInterface;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.support.annotation.NonNull;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.app.LoaderManager.LoaderCallbacks;
import android.content.CursorLoader;
import android.content.Loader;
import android.database.Cursor;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.text.TextUtils;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.inputmethod.EditorInfo;
import android.view.inputmethod.InputMethodManager;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
import org.json.json_objectect;
import java.net.SocketTimeoutException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;
//import java.util.logging.Handler;
import android.os.Handler;
import static android.Manifest.permission.READ_CONTACTS;

public class Login_activity extends AppCompatActivity implements
LoaderCallbacks<Cursor> {
    private static final Integer LOGIN_OK = 0;
    private static final Integer LOGIN_PASSWORD_INCORRECT = 1;
    private static final Integer LOGIN_ACCOUNT_NOT_FOUND = 2;
    private Activity instance;
    private LinearLayout linear_layout;
    public static final String LOG_TAG = "Login_activity";
    private static final int REQUEST_READ_CONTACTS = 0;
    private AutoCompleteTextView my_user_name_view;
    private EditText my_password_view;
    private View my_progress_view;
```

```
    private View my_login_form_view;
    @Override
    protected void onCreate(Bundle saved_instance_state) {
        super.onCreate(saved_instance_state);
        setContentView(R.layout.activity_login);
        setupActionBar();
        instance = this;
        linear_layout = (LinearLayout)
findViewById(R.id.layout_linear_login_activity);
        my_user_name_view = (AutoCompleteTextView)
findViewById(R.id.username);
        populate_auto_complete();
        my_password_view = (EditText)
findViewById(R.id.password);
        my_password_view.setOnEditorActionListener(new
TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView textView, int
id, KeyEvent keyEvent) {
                if (id == R.id.login || id ==
EditorInfo.IME_NULL) {
                    attemptLogin();
                    return true;
                }
                return false;
            }
        });
        Button my_sign_in_button = (Button)
findViewById(R.id.sign_in_button);
        my_sign_in_button.setOnClickListener(new
OnClickListener() {
            @Override
            public void onClick(View view) {
                attemptLogin();
            }
        });
        my_login_form_view = findViewById(R.id.login_form);
        my_progress_view = findViewById(R.id.login_progress);
    }
    private void populate_auto_complete() {
        if (!may_request_contacts()) {
            return;
        }
        getLoaderManager().initLoader(0, null, this);
    }
    private boolean may_request_contacts() {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
            return true;
        }
        if (checkSelfPermission(READ_CONTACTS) ==
PackageManager.PERMISSION_GRANTED) {
            return true;
        }
```

```
        if (shouldShowRequestPermissionRationale(READ_CONTACTS))
{
            Snackbar.make(my_user_name_view,
R.string.permission_rationale, Snackbar.LENGTH_INDEFINITE)
                    .setAction(android.R.string.ok, new
View.OnClickListener() {
                        @Override
                        @TargetApi(Build.VERSION_CODES.M)
                        public void onClick(View v) {
                            requestPermissions(new
String[]{READ_CONTACTS}, REQUEST_READ_CONTACTS);
                        }
                    });
        } else {
            requestPermissions(new String[]{READ_CONTACTS},
REQUEST_READ_CONTACTS);
        }
        return false;
    }
    @Override
    public void onRequestPermissionsResult(int requestCode,
@NonNull String[] permissions,
                                           @NonNull int[]
grantResults) {
        if (requestCode == REQUEST_READ_CONTACTS) {
            if (grantResults.length == 1 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                populate_auto_complete();
            }
        }
    }

    @TargetApi(Build.VERSION_CODES.HONEYCOMB)
    private void setupActionBar() {
        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.HONEYCOMB) {

getSupportActionBar().setDisplayHomeAsUpEnabled(false); // BACK
BUTTON
        }
    }

    private void attemptLogin() { // hide keyboard
        View v = this.getCurrentFocus();
        if (v != null) {
            InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);

imm.hideSoftInputFromWindow(my_user_name_view.getWindowToken(),
0);
        }
        my_user_name_view.setError(null);
        my_password_view.setError(null);
```

```
        String username =
my_user_name_view.get_text().toString();
        String password =
my_password_view.get_text().toString();
        boolean cancel = false;
        View focus_view = null;
        if (password.equals("") || !is_password_valid(password))
{

my_password_view.setError(getString(R.string.error_invalid_passw
ord));
            focus_view = my_password_view;
            cancel = true;
        }
        if (TextUtils.isEmpty(username)) {

my_user_name_view.setError(getString(R.string.error_field_requir
ed));
            focus_view = my_user_name_view;
            cancel = true;
        } else if (!is_user_name_valid(username)) {

my_user_name_view.setError(getString(R.string.error_invalid_user
name));
            focus_view = my_user_name_view;
            cancel = true;
        }
        if (cancel) {
            focus_view.requestFocus();
        } else {
            show_progress(true);
            final String my_login = username;
            final String my_password = password;
            User return_user = null;
            Integer login_result = -1;
            User user = null;
            try {
                user =
Internet_database.find_user_in_database_by_login(my_login);
            } catch (SocketTimeoutException e) {
                Snackbar snackbar = Snackbar.make(linear_layout,
"Connection error", Snackbar.LENGTH_LONG);
                snackbar.show();
                show_progress(false);
                return;
            }
            if (user != null) { // user found
                if (Internet_database.check_user_password(user,
my_password)) { // password is correct
                    return_user = user;
                    login_result = LOGIN_OK;
                } else { // password is incorrect
                    login_result = LOGIN_PASSWORD_INCORRECT;
```

```
                }
            } else { // user not found
                login_result = LOGIN_ACCOUNT_NOT_FOUND;
            }
            show_progress(false);
            if (login_result.equals(LOGIN_OK)) {
                Snackbar snackbar = Snackbar.make(linear_layout,
"Login success", Snackbar.LENGTH_LONG);
                snackbar.show();
                SharedPreferences.Editor ed =
App.getInstance().getPref().edit();
                ed.putString("loggedUser",
make_user_json(return_user,
App.get_password_hash(my_password)).toString());
                ed.commit();
                App.getInstance().set_logged_user(return_user);
                Log.d(LOG_TAG, "LOGIN_OK set_logged_user, login:
" + return_user.get_login());

App.getInstance().set_user_password_hash(App.get_password_hash(m
y_password));

App.getInstance().setmessages_list(App.getInstance().get_user_me
ssages_from_database(return_user));

Internet_database.sync_messages_with_database(return_user,
App.get_password_hash(my_password));

Main_activity.getNavUsername().set_text(return_user.get_login())
;
                finish();
            } else if
(login_result.equals(LOGIN_PASSWORD_INCORRECT)) {

my_password_view.setError(getString(R.string.error_incorrect_pas
sword));
                my_password_view.requestFocus();
            } else if
(login_result.equals(LOGIN_ACCOUNT_NOT_FOUND)) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(instance);

builder.set_title(getResources().getString(R.string.account_crea
te_prompt_title));

builder.setMessage(getResources().getString(R.string.account_cre
ate_prompt_text))

builder.setPositiveButton(getResources().getString(R.string.acco
unt_create_prompt_positive), new
DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface
dialog, int which) {
```

```
                                    User user = new User(my_login);
                                    if
(!Internet_database.create_user_in_database(user, my_password))
{
                                        Toast.makeText(instance,
getResources().getString(R.string.account_create_error),
Toast.LENGTH_LONG).show();
                                    } else {
                                        attemptLogin();
                                        dialog.cancel();
                                        Toast.makeText(instance,
getResources().getString(R.string.account_create_success),
Toast.LENGTH_LONG).show();
                                        finish();
                                    }
                                }
                            })

.setNegativeButton(getResources().getString(R.string.account_cre
ate_prompt_negative), null) .show();
            }
        }
    }

    private json_objectect make_user_json(User user, String
password_hash) {
        json_objectect json_user = new json_objectect();
        try {
            json_user.put("id",
(Integer.toString(user.get_id())));
            json_user.put("login", user.get_login());
            json_user.put("password_hash", password_hash);
            json_user.put("email", user.get_email());
            json_user.put("about", user.get_about());
            json_user.put("additional", user.get_additional());
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        } return json_user;
    }

    private boolean is_user_name_valid(String user_name) {
        return user_name.length() > 4;
    }

    private boolean is_password_valid(String password) {
        return password.length() >= 6 && password.length() <=
24;
    }
    @TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
    private void show_progress(final boolean show) {
        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.HONEYCOMB_MR2) {
```

```
            int short_animation_time =
getResources().getInteger(android.R.integer.config_short_animati
on_time);
            my_login_form_view.setVisibility(show ? View.GONE :
View.VISIBLE);

my_login_form_view.animate().setDuration(short_animation_time).a
lpha(
            show ? 0 : 1).setListener(new
AnimatorListenerAdapter() {
                @Override
                public void onAnimationEnd(Animator animation) {
                    my_login_form_view.setVisibility(show ?
View.GONE : View.VISIBLE);
                }
            });
            my_progress_view.setVisibility(show ? View.VISIBLE :
View.GONE);

my_progress_view.animate().setDuration(short_animation_time).alp
ha(
                    show ? 1 : 0).setListener(new
AnimatorListenerAdapter() {
                @Override
                public void onAnimationEnd(Animator animation) {
                    my_progress_view.setVisibility(show ?
View.VISIBLE : View.GONE);
                }
            });
        } else {
            my_progress_view.setVisibility(show ? View.VISIBLE :
View.GONE);
            my_login_form_view.setVisibility(show ? View.GONE :
View.VISIBLE);
        }
    }
    @Override
    public Loader<Cursor> onCreateLoader(int i, Bundle bundle) {
        return new CursorLoader(this,

Uri.withAppendedPath(ContactsContract.Profile.CONTENT_URI,

ContactsContract.Contacts.Data.CONTENT_DIRECTORY),
ProfileQuery.PROJECTION,
                ContactsContract.Contacts.Data.MIMETYPE +
                        " = ?", new
String[]{ContactsContract.CommonDataKinds.Email
                .CONTENT_ITEM_TYPE},
                ContactsContract.Contacts.Data.IS_PRIMARY + "
DESC");
    }
    @Override
```

```
    public void onLoadFinished(Loader<Cursor> cursorLoader,
Cursor cursor) {
        List<String> emails = new ArrayList<>();
        cursor.moveToFirst();
        while (!cursor.isAfterLast()) {
            emails.add(cursor.getString(ProfileQuery.ADDRESS));
            cursor.moveToNext();
        }
        add_emails_to_auto_complete(emails);
    }
    @Override
    public void onLoaderReset(Loader<Cursor> cursorLoader) {
    }
    private void add_emails_to_auto_complete(List<String>
email_address_collection) {
        ArrayAdapter<String> adapter =
                new ArrayAdapter<>(Login_activity.this,

android.R.layout.simple_dropdown_item_1line,
email_address_collection);
        my_user_name_view.setAdapter(adapter);
    }
    private interface ProfileQuery {
        String[] PROJECTION = {
                ContactsContract.CommonDataKinds.Email.ADDRESS,

ContactsContract.CommonDataKinds.Email.IS_PRIMARY,
        };
        int ADDRESS = 0;
        int IS_PRIMARY = 1;
    }
}
```

Додаток. Код програми, що виконує оцінку відстані до QR-коду відомого фактичного розміру.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using QRCodeEncoderLibrary;

namespace Recogn
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            // create QR Code encoder object
            QRCodeEncoder Encoder = new QREncoder();
            // Error correction
            // error correction low (7%)
            Encoder.ErrorCorrection = ErrorCorrection.L;

            // or
            // error correction medium (15%) The Default
            Encoder.ErrorCorrection = ErrorCorrection.M;

            // or
            // error correction quarter (25%)
            Encoder.ErrorCorrection = ErrorCorrection.Q;

            // or
            // error correction high (30%)
            Encoder.ErrorCorrection = ErrorCorrection.H;

            // module size in pixels (default is 2)
            Encoder.ModuleSize = 2;

            // Quiet zone around the symbol (default is 8)
            // Quiet zone must be at least 4 times the module
size
            Encoder.QuietZone = 8;

            // ECI Assignment Value (default is -1 not used)
            // The ECI value is a number in the range of 0 to
999999.
```

```
            // or -1 if it is not used
            Encoder.ECIAssignValue = -1;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Log.Debug("Начинается снимок");
            string adress = Common.directoryEXE + @"\Temp";
            Directory.CreateDirectory(adress);
            string adressV = adress + @"\" +
DateTime.Now.ToString("HH-mm-ss-fffffff");

            IntPtr hWndC = IntPtr.Zero;
            IntPtr hBmp = IntPtr.Zero;
            try
            {
                string deviceIndex = Convert.ToString(index);

                hWndC = capCreateCaptureWindowA(deviceIndex,
WS_POPUP | WS_CHILD, 0, 0, windowWidth, windowHeight, 0, 0); //
узнать дескриптор камери
                SendMessage(hWndC, WM_CAP_DRIVER_CONNECT, 0,
IntPtr.Zero); //подключиться к камере

                hBmp = Marshal.StringToHGlobalAnsi(adressV);
                SendMessage(hWndC, WM_CAP_SAVEDIB, 0, hBmp); //
сохранить скриншот

                SendMessage(hWndC, WM_CAP_DRIVER_DISCONNECT, 0,
IntPtr.Zero); //отключить камеру

                string path = Common.directoryEXE +
@"\PictureCam\" + DateTime.Now.ToString("yyyy-MM-dd") + @"\" +
index;
                Directory.CreateDirectory(path);

                string ad = path + @"\" +
DateTime.Now.ToString("HH-mm-ss-fffffff") + ".jpeg";

                using (var tr = Image.FromFile(adressV))
                {
                    Bitmap bmp = new Bitmap(tr, windowWidth,
windowHeight);
                    bmp.Save(ad, ImageFormat.Jpeg);
                    bmp.Dispose();
                }

                Log.Debug("Снимок");
            }
            catch (Exception ex)
            {
                Log.Message("Трудности при снимке с камеры " +
ex);
```

```
            }
            finally
            {
                File.Delete(adressV);
                Marshal.FreeHGlobal(hBmp);
                DestroyWindow(hWndC);
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            QRDecoder Decoder = new QRDecoder();

            // call image decoder methos with
<code>Bitmap</code> image of QRCode barcode
            byte[][] DataByteArray =
Decoder.ImageDecoder(InputImage)

// get the ECI Assignment value
            ECIValue = Decoder.ECIAssignValue;
            if (ECIValue == -1)
            {
                // Assignment value not defined
            }
            else
            {
                // Assignment value between 0 to 999999
            }
            // convert binary result to text string
            string Result =
QRCode.ByteArrayToStr(DataByteArray[Index]);
            // The QRDecoder converts byte array to text string
the class using this conversion
            public static string ByteArrayToStr(byte[]
DataArray)
            {
                Decoder = Encoding.UTF8.GetDecoder();
                int CharCount = Decoder.GetCharCount(DataArray,
0, DataArray.Length);
                char[] CharArray = new char[CharCount];
                Decoder.GetChars(DataArray, 0, DataArray.Length,
CharArray, 0);
                return new string(CharArray);
            }

        }

        private void button3_Click(object sender, EventArgs e)
        {
            // create QREncoder object
            QREncoder Encoder = new QREncoder()

// set optional parameters
```

```
            Encoder.ErrorCorrection = ErrorCorrection.Q;
            Encoder.ModuleSize = 4;
            Encoder.QuietZone = 16;

            // encode input text string
            // Note: there are 4 Encode methods in total
            Encode("QR Code barcode example text string");

            // save the barcode to PNG file
            // This method DOES NOT use Bitmap class and is
suitable for net-core and net-standard
            // It produces files significantly smaller than
SaveQRCodeToFile.
            Encoder.SaveQRCodeToPngFile("File-Name");

            // or
            // save barcode image to an open stream
            Encoder.SaveQRCodeToPngFile(OutputStream);

            // or
            // save barcode image to file using Bitmap class
            // this is for net-framework and not for net-code or
net-standard
            // you can produce PNG file but consider the
QRCOdeToPngFile method above.
            Encoder.SaveQRCodeToFile("File-Name", ImageFormat);

            // or
            // save barcode image to an open stream
            Encoder.SaveQRCodeToFile(OutputStream, ImageFormat);

            // or
            // create Bitmap class (net-framework only)
            Bitmap bitmap = CreateQRCodeBitmap();

            // or
            // create Bitmap class with different colors (net-
framework only)
            Bitmap bitmap = CreateQRCodeBitmap(WhiteBrush,
BlackBrush);

            // or
            // create bool array of image pixels
            Bool[,] ImagePixels = ConvertQRCodeMatrixToPixels();
        }
    }
}
```