НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий комплекс «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
“___”_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 Комп’ютерні науки
(код і назва)

на тему: Автоматизація процесу відеозйомки з використанням компактного безпілотного літального апарату

Виконав (-ла): студент (-ка) 4 курсу, групи ДА-31
(шифр групи)

Косенко Олег Михайлович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н., Булах Б.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Економічний доцент, к.е.н., Рощина Н.В. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент доц. каф. ММСА, к.т.н., доцент, Тимощук О.Л. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль старший викладач, Британ О.А. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Київ – 2017 року

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

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Косенку Олегу Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи _Автоматизація процесу відеозйомки з використанням компактного безпілотного літального апарату_____ _____ ,
керівник роботи Булах Богдан Вікторович, к.т.н., ст. викладач,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10» травня 2017 р. № 1477-с

2. Термін подання студентом роботи 09.06.2017

3. Вихідні дані до роботи Моделювання польоту компактних БПЛА типу "квадрокоптер", можливість роботи з тривимірними перешкодами, візуалізація з використанням 3D-бібліотеки Unity.

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)
1. Аналіз алгоритмів пошуку шляху у просторі.
2. Візуалізація системи пошуку шляху у просторі з перешкодами.
3. Підключення контролера.
4. Проведення функціонально-вартісного аналізу програмного продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) _____

1. Ілюстрація результатів роботи програми – плакат.
2. Алгоритмічна частина проекту – плакат.
3. Технічна частина проекту – плакат.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічно-організаційна частина	Рощина Н.В., к.е.н.		

7. Дата видачі завдання_____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	01.02.2017	
2	Планування дипломного проекту	15.02.2017	
3	Дослідження існуючих рішень	28.02.2017	
4	Вибір математичної моделі	15.03.2017	
5	Реалізація програмного продукту	15.04.2017	
6	Тестування програмного продукту	30.04.2017	
7	Аналіз результатів роботи	15.05.2017	
8	Оформлення роботи	30.05.2017	
9	Проходження нормо-контролю	10.06.2017	

Студент _____ О.М. Косенко
(підпис) (ініціали, прізвище)

Керівник роботи _____ Б.В. Булах
(підпис) (ініціали, прізвище)

*Консультантом не може бути зазначено керівника дипломної роботи.

АНОТАЦІЯ

бакалаврської дипломної роботи Косенка Олега Михайловича на тему «Автоматизація процесу відеозйомки з використанням компактного безпілотного літального апарату»

Метою даної роботи є створення системи автоматизації керування БПЛА, що знаходить оптимальний шлях до цілі замість оператора. Було виконано огляд предметної області і доведено актуальність даної роботи.

В роботі було розроблено модель автоматизації процесу відеозйомки з використанням компактного безпілотного літального апарату. Дипломна робота включає в себе розробку двох частин цієї моделі: знаходження оптимального шляху та передачу даних до польотного контролера. Для вирішення задачі знаходження оптимального шляху були розглянуті різні алгоритми. Алгоритм A*, що виявився найкращим, було модифіковано і реалізовано у 3D просторі Unity. Задачу передачі даних до польотного контролера було вирішено шляхом емуляції контролера за допомогою мікроконтролера Arduino Uno. Планується вдосконалення даної системи, що дозволить її практичне використання.

Загальний обсяг роботи: 96 сторінок, 50 рисунків, 7 таблиць та 25 посилань.

Ключові слова: дрон, квадрокоптер, БПЛА, UAV, RPAS, відеозйомка, PID регулятор, крен, тангаж, рискання, евристика, мікроконтролер, A*.

АННОТАЦИЯ

бакалаврской дипломной работы Косенка Олега Михайловича на тему «Автоматизация процесса видеосъемки с использованием компактного безпилотного летательного аппарата»

Целью данной работы является создание системы автоматизации управления БПЛА, которая находит оптимальный путь к цели вместо оператора. Был сделан осмотр предметной области и доказана актуальность данной работы.

В работе была разработана модель автоматизации процесса видеосъемки с использованием компактного беспилотного летательного аппарата. Дипломная работа включает в себя разработку двух частей этой модели: нахождение оптимального пути и передача данных на полетный контроллер. Для решения задачи нахождения оптимального пути были рассмотрены различные алгоритмы. Алгоритм A*, который оказался наилучшим, был модифицирован и реализован в 3D пространстве Unity. Задача передачи данных на полетный контроллер была решена путем эмуляции контроллера с помощью микроконтроллера Arduino Uno. Планируется усовершенствование данной системы, которое позволит ее практическое применение.

Общий объем работы: 96 страниц, 50 рисунков, 7 таблиц и 25 источников.

Ключевые слова: дрон, квадрокоптер, БПЛА, UAV, RPAS, видеосъемка, PID регулятор, крен, тангаж, рысканье, эвристика, микроконтроллер, A*.

ABSTRACT

to the bachelor thesis by Kosenko Oleg Mykhailovich on “Automation of the video recording process using compact Unmanned Aerial Vehicle”

The purpose of this work is to create an automation system for controlling UAV, which finds the optimal path to the goal instead of the operator. An examination of the subject area was made and the relevance of this work was proved.

A model for automating of the video recording process using a compact unmanned aerial vehicle was developed. The diploma work includes the development of two parts of this model: finding the optimal path and transferring data to the flight controller. To solve the problem of finding the optimal path various algorithms were considered. The algorithm A*, which turned out to be the best, was modified and implemented in the 3D Unity space. The task of data transfer to the flight controller was solved by emulating the controller with the Arduino Uno microcontroller. It is planned to improve this system, which will allow its practical application.

Includes 96 pages, 50 figures, 7 tables and 25 sources.

Keywords: drone, quadrocopter, UAV, RPAS, video recording, PID controller, roll, pitch, yaw, heuristics, microcontroller, A*.

ABSTRACT

Zur Bachelorarbeit von Kosenko Oleg Mykhailovich zum Thema "Automatisierung des Videoaufzeichnungsprozesses mit kompakten unbemannten Luftfahrzeugen"

Der Zweck dieser Arbeit ist es, ein Automatisierungssystem zur Steuerung von UAV zu schaffen, das den optimalen Weg zum Ziel statt des Betreibers findet. Eine Untersuchung des Fachgebiets wurde durchgeführt und die Relevanz dieser Arbeit wurde bewiesen.

Es wurde ein Modell zur Automatisierung des Videoaufzeichnungsprozesses mit einem kompakten, unbemannten Luftfahrzeug entwickelt. Die Diplomarbeit beinhaltet die Entwicklung von zwei Teilen dieses Modells: die Suche nach dem optimalen Weg und die Übertragung von Daten an den Flugcontroller. Um das Problem der Suche nach dem optimalen Weg zu lösen, wurden verschiedene Algorithmen betrachtet. Der Algorithmus A*, der sich als der Beste erwies, wurde im 3D Unity Space modifiziert und implementiert. Die Aufgabe der Datenübertragung an den Flugregler wurde durch die Emulation des Reglers mit dem Arduino Uno Mikrocontroller gelöst. Es ist geplant, dieses System zu verbessern, das seine praktische Anwendung ermöglicht.

Enthält 96 Seiten, 50 Figuren, 7 Tabellen und 25 Quellen.

Schlüsselwörter: Drohne, Quadrocopter, UAV, RPAS, Videoaufzeichnung, PID-Regler, Rolle, Tonhöhe, Gier, Heuristik, Mikrocontroller, A*.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БПЛА – Безпілотний літальний апарат

UAV – Unmanned Aerial Vehicle

RPAS – Remotely Piloted Aircraft System

PID controller – Proportional Integral Differential controller

GPS – Global Positioning System

OSD – On-Screen Display

IDE – Integrated Development Environment

ВСТУП

Безпілотні літальні апарати є сектором авіації, який розвивається дуже швидко і має великий потенціал для зростання і створення нових робочих місць. Термін «безпілотний літальний апарат» включає як великі літаки, аналогічні за розміром і складністю пілотованому літаку так і невеликі електронні пристрої для персонального використання.

Сьогодні технології з використанням дронів застосовуються в агрогосподарствах для точного, своєчасного і ефективного внесення добрив та пестицидів, для інспектування безпеки інфраструктурних об’єктів, таких як залізничні колії, дамби, канали, лінії електропередач, трубопроводи, автодороги. Впроваджуються технології попередження катастроф та пом’якшення їх наслідків, наприклад, обльоти затоплених територій та підтримка пожежогасіння.

Актуальність теми розвитку дронів полягає в застосуванні їх для моніторингу природних ресурсів, охорони оточуючого середовища, атмосферних досліджень, сфери медіа та розваг, спортивної фотографії, створення фільмів про живу природу, досліджень, боротьби з локальними витоками газу та хімічних речовин, а також могли б бути запрограмовані щоб діяти як бджоли для штучного запилення рослин.

Метою даної роботи є створення системи автоматизації керування БПЛА, що знаходить оптимальний шлях до цілі оминаючи можливі перешкоди.

Основні завдання:

Дослідження та вивчення особливостей алгоритмів пошуку оптимальної траєкторії руху у просторі.

Розробка прототипу системи моделювання переміщення БПЛА у просторі з перешкодами з урахуванням задач відеозйомки.

Передача знайденого оптимального шляху на емулятор польотного контролера.

1 Огляд предметної області

1.1 Роль БПЛА в сучасних технологіях

Безпілотні літальні апарати (більшість людей називають їх «безпілотники» або «дрони») є сектором авіації, який розвивається дуже швидко і має великий потенціал для зростання і створення нових робочих місць. Термін «безпілотний літальний апарат» включає як великі літаки, аналогічні за розміром і складністю пілотованому літаку так і невеликі електронні пристрої для персонального використання. Особливо швидко розвивається сектор невеликих дронів.

У квітні 2014 р. Європейська Комісія прийняла звернення до Європейського Парламенту та Ради «A new era for aviation. Opening the aviation market to the civil use of remotely piloted aircraft systems in a safe and sustainable manner» [1]. Як сказано в цьому зверненні, дистанційно керовані авіаційні системи (RPAS) зможуть запропонувати «міріади нових послуг», значно змінюючи наше щоденне життя. Як технології інтернету на початку дев'яностих дали початок багатьом різноманітним застосуванням, RPAS технології мають привести в найближчі роки до розвитку широкого різноманіття послуг, особливо в поєднанні з іншими технологіями, такими як точне позиціювання з допомогою супутникової системи Galileo, у поєднанні з телекомунікаційними системами – для попередження та пом'якшення наслідків природних катастроф, для динамічного збільшення пропускної здатності комунікаційних мереж. Незважаючи на те, що точну природу і обсяг потенціального застосування RPAS наразі важко передбачити, очікується що промисловість зможе генерувати достатні прибутки для швидкого розвитку цього нового напрямку.

Сьогодні технології з використанням дронів застосовуються в агрогосподарствах для точного, своєчасного і ефективного внесення добрив та

пестицидів. В Європі дрони застосовуються для інспектування безпеки інфраструктурних об’єктів, таких як залізничні колії, дамби, канали, лінії електропередач, трубопроводи, автодороги. Впроваджуються технології попередження катастроф та пом’якшення їх наслідків, наприклад, обльоти затоплених територій та підтримка пожежогасіння. Також застосування дронів актуальне для моніторингу природних ресурсів, охорони оточуючого середовища, атмосферних досліджень, сфери медіа та розваг, спортивної фотографії, створення фільмів про живу природу, досліджень, полювання та моніторингу дотримання правил мисливцями, тощо.

В майбутньому дрони могли б піднімати в атмосферу гігантські вітряні турбіни для продукування «зеленої» електроенергії. З іншого боку інженери працюють над мікро-дронами, що могли б боротися з локальними витоками газу та хімічних речовин а також які могли б бути запрограмовані щоб діяти як бджоли для штучного запилення рослин.

Дрони включають в себе багато різних типів літальних апаратів, що мають підйомну силу від кількох грамів до понад 10 тон, характеризуються швидкісними показниками від зависання на місці до швидкості понад 1000 км/год, можуть проводити в повітрі від кількох хвилин до місяців, з точки зору технології підіймання вони можуть бути роторними, з фіксованим крилом або легші за повітря.

Окрім виробників та системних інтеграторів, індустрія дронів також включає широку мережу поставок різноманітних пристроїв та технологій (польотні контролери, засоби комунікації, двигуни, джерела живлення, сенсори, прилади телеметрії, тощо).

Дрони – це зростаючий ринок, що створює робочі місця і позитивно впливає на зростання економіки. Розвиток технологій RPAS буде ключовим для майбутньої конкурентоздатності Європейської авіаційної промисловості. На

сьогодні США та Ізраїль глобально домінують у цьому секторі промисловості, базуючись на досвіді будівництва великих військових RPAS . Інші країни, такі як Бразилія, Китай, Індія та Росія також демонструють потенціал стати сильними конкурентами.

Точні масштаби потенційного ринку БПЛА важко передбачити. Очікується, що глобальний бюджет на розробку та придбання БПЛА, враховуючи військові та державні закупівлі виросте з теперішніх більш ніж 5 млрд. долл. до понад 11,6 млрд. долл. у 2023 році. У 2014 році глобальні статистичні дані показали існування 1708 конструкцій дронів, виготовлених 471 виробником. Ринок розвивається надзвичайно стрімко. Так, кількість операторів БПЛА в Японії зросла з 18 осіб у 1993 році до близько 14000 у 2005 році. Головним рушієм цього росту було використання БПЛА у сільськогосподарській галузі.

Ріст застосування дронів спричиняє значний ріст робочих місць. У Сполучених Штатах прогнозується створення 100 000 робочих місць до 2025 року. У Європі прогнозується створення 150 000 робочих місць до 2050 року, не рахуючи безпосередніх операторів дронів.

Європейська стратегія спрямована на створення єдиного ринку БПЛА, щоб пожинати соціальні вигоди від цієї іноваційної технології, та подолати занепокоєння громадськості через публічні обговорення та у разі необхідності, захисні заходи. Також ця стратегія має створити умови для міцної і конкурентної індустрії виробництва та використання дронів для конкуренції на глобальному ринку.

1.2 Законодавча база

1.2.1 Законодавча база за кордоном

Законодавча база для використання і застосування дронів фрагментарна та недосконала. Але в багатьох країнах, зокрема в ЕС та США робляться цілеспрямовані кроки для врегулювання відповідного законодавства. В Європі цим питанням займається Європейська агенція з авіаційної безпеки EASA [2].

Загальний підхід у врегулюванні використання дронів базується на оцінці ризиків що від них походять. Відповідно дрони поділяються на 3 категорії:

Категорія низького ризику – авторизація для використання дрона не потрібна;

Категорія середнього ризику – необхідний дозвіл для початку використання дрона, накладаються певні обмеження щодо його використання;

Категорія високого риску – літальний апарат повинен бути сертифікований, а пілот (оператор) мати відповідну ліцензію.

Інформація щодо законодавства та правил використання дронів у ЄС, Норвегії та Швейцарії публікується на сайті [3].

Один з базових принципів використання дронів – це безпека, зокрема безпечна інтеграція дронів з загальною авіаційною системою безпеки.

Правила безпеки мають бути пропорційними до ризику, беручи до уваги вагу, швидкість, складність, авіаційний клас та місце або особливості застосування дрона, тощо. Традиційна концепція авіаційної сертифікації, ліцензування пілотів та операторів має бути доповнена формами спрощеного регулювання. Виробники та користувачі малих дронів лобіюють гармонізацію правил управління для полегшення їх комерційного поширення та застосування.

Європейською Комісією був розроблений та опублікований проект Правил щодо використання дронів: Prototype Commission Regulation on Unmanned Aircraft Operations [4].

Але на даний момент законодавчо сфера застосування дронів у Європі регулюється на національному рівні. Законодавчі акти країн членів ЄС публікуються на сайті міжурядової організації EUROCONTROL [5].

Згідно Європейського законодавства використання дронів не повинне приводити до порушення фундаментальних прав, включаючи повагу до приватного та сімейного життя а також захист особистих даних.

Враховуючи те, що використання дрона може привести до нещасного випадку та нанесення шкоди, Європейська комісія працює над розробкою вимог щодо визначення винуватця нещасного випадку та щодо страхування цивільної відповідальності власника/оператора дрона.

В США використання безпілотних літальних апаратів регулюється Федеральною Адміністрацією з авіації, у відповідності до Федерального Кодексу «Code of Federal Regulations 14 C.F.R». У 2015 році Федеральною Адміністрацією з Авіації був прийнятий законодавчий акт, відомий як Part 107 [6], що вимагає обов'язкової реєстрації всіх БПЛА (рис. 1.1).

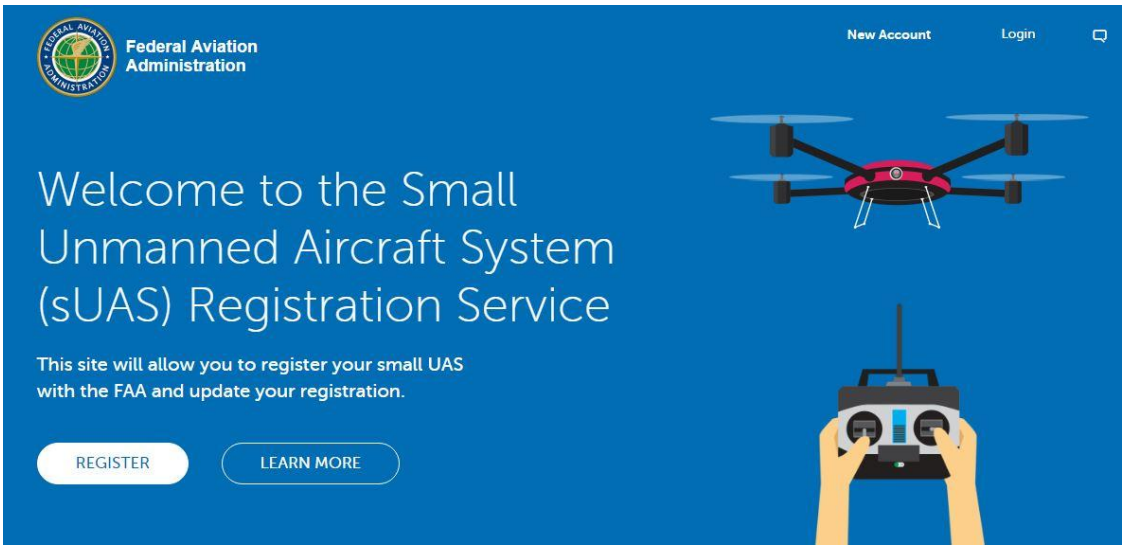


Рисунок 1.1 – Сайт реєстрації БПЛА в FAA

Однак 19.05.2017 Апеляційний суд округу Колумбія у справі Джон А. Тейлор проти Федеральної Адміністрації з Авіації виніс рішення, що вимога обовязкової реєстрації не може застосовуватися до авіамоделей, вивівши таким чином значну частину дронів з під дії вимоги щодо реєстрації [7].

Відповідно до Кодексу авіамоделлю вважається дрон, що відповідає таким критеріям:

Використовується як хобі або для розваги;

Не порушує місцевих правил безпеки;

Важить не більше 55 фунтів (25 кг), або у разі більшої ваги, пройшов відповідну сертифікацію;

Не перешкоджає жодному пілотованому літальному апарату;

У разі виконання польоту в радіусі 5 миль (8 км) від аеропорту – необхідно завчасно попередити оператора аеропорту та контрольну вежу.

1.2.2 Законодавча база в Україні

В Україні вже є законодавча база щодо дронів — вони підпадають під визначення безпілотного повітряного судна, що міститься в Повітряному кодексі України. Більше того, такі пристрої наразі не підлягають обов’язковій реєстрації повітряних суден за умови, що їхня злітна вага не перевищує 20 кг та мета використання обмежується розвагами і спортом (пункт 2.1.5 Правил реєстрації цивільних повітряних суден в Україні). Тобто звичайний цивільний безпілотник прирівнюється до радіокерованої авіамоделі за способом використання. Очевидно, що будь-яка інша мета використання дрона, аніж розваги, є підставою для реєстрації його як повітряного судна, що передбачає одержання від Державіаслужби (ДАС) відповідного сертифіката льотної придатності та

реєстраційного номера (можна навіть спробувати нанести номер на борт свого безпілотника, якщо фізичні розміри дозволяють).

Окремо ще стоїть питання, чи треба проходити курси з керування таким повітряним судном. Згідно з озвученою ДАС концепцією регулювання безпілотних польотів, планується впровадження поняття «зовнішній пілот», якому видаватиметься свідоцтво певного типу. Буде воно видаватися за заявою чи все ж доведеться складати іспити — поки незрозуміло [8].

В Європейському Союзі галузь БПЛА отримує серйозну підтримку. Європейська Комісія підтримує зростання ринку RPAS та конкурентні умови у пов’язаних промислових секторах, що включають середні і малі підприємства та стартапи. Використовуючи інструменти Євросоюзу, такі як Програми Horizon 2020 та COSME Єврокомісія буде просувати розвиток і застосування БПЛА в широкому діапазоні секторів, стимулюючи інновації і сприяючи створенню крос-секторних промислових ланцюжків створення доданої вартості. Єврокомісія також буде створювати програми для надання можливостей просування та застосування цієї інноваційної технології. Наприклад, RPAS можуть зіграти свою роль у програмах Copernicus, EU’s Earth Observation Programme.

1.3 Різновиди дронів

Дрони стають реальністю і є комерційно доступними. Ринок дронів дає реальну можливість створення робочих місць та є джерелом інновацій та економічного зростання в наступні роки. Також вони несуть нові виклики пов’язані з безпекою та повагою прав громадян. Необхідне вдосконалення законодавства а також зусилля з розробки та впровадження технологій, для того щоб інтегрувати дрони у загальний простір цивільної авіації, та підвищити рівень впевненості у безпеці та дотриманні прав приватності.

Враховуючи різноманіття БПЛА важливою є їх класифікація. На основі узагальнення відомих класифікацій та тактико-технічних характеристик існуючих безпілотних літальних апаратів запропоновано їхню класифікацію, за основними ознаками: використання; тип системи керування; правила польоту; клас; тип; тип крила; спосіб зльоту/посадки; тип двигуна; паливна система; тип паливного бака; кількістьвикористань; категорія (з урахуванням маси і максимальної дальності дії); радіус дії; висота; функціональне призначення.

У цій роботі буде розглядатися БПЛА типу «коптер». Коптери класифікуються за кількістю приводних двигунів. Існують три-, квадро-, гексо-, окто- та мультикоптери (рис. 1.2).

Рисунок 1.2 – Квадрокоптер, гексокоптер та октокоптер відповідно

Найбільш поширеними коптерами для розваг та аматорської відеозйомки є квадрокоптери. Також вони є найбільш поширеними для моделювання.

Найвідомішим на сьогодні виробником квадрокоптерів обладнаних камерою для високоякісної відеозйомки є компанія DJI [9].

На ринку представлені такі її моделі квадрокоптерів як Phantom, Mavic, Inspire, Matrix, та інші (рис. 1.3).

Рисунок 1.3 – Квадрокоптери DJI Phantom, Mavic, Inspire і Matrix

1.4 Складові дронів

1.4.1 Польотний контролер

Польотом дрона безпосередньо керує польотний контролер. Існують десятки моделей популярних польотних контролерів різних виробників. Їх характеристики у кожній ціновій категорії загалом подібні, і вибір польотного контролера залежить від особистих уподобань конструктора дрона. Цінова шкала доступних на ринку польотних контролерів сягає від кількох десятків доларів до кількох тисяч доларів США [10].

До функцій польотного контролера відносяться:

Стабілізація апарата в повітрі;

Утримання висоти (за допомогою барометра) та позиції (за допомогою GPS);

Автоматичний політ за заданими точками (опція);

Передача на землю поточних параметрів польоту за допомогою модема або Bluetooth (опція)

Забезпечення безпеки польоту (повернення в точку злету при втриті сигналу, автопосадка)

Підключення додаткової периферії: OSD (накладання параметрів польоту на відео), світлодіодна індикація і т.д.

Кількість функцій залежить від наявності на борту коптера відповідної периферії, в дешевих контролерах ряд функцій може бути відсутнім.

У таблиці 1.1 наведено порівняння функціональності різних польотних контролерів різних виробників.

Таблиця 1.1 – Порівняння польотних контролерів

Контролер	**Стабілізація польоту**	**Утримання висоти**	**Утримання позиції**	**Політ по точкам**	**Модем/телеметрія**	**OSD**
MultiWii	+	+	+	+	+	+
ArduCopter	+	+	+	+	+	+
Rabbit	+	+	+	-	-	-
DJI Naza Lite	+	+	+	-	-	-
DJI Naza V1/V2	+	+	+	+	+	+
DJI Wookong	+	+	+	+	+	+
Zero UAV X4/X6	+	+	+	+	+	-
XAircraft	+	+	+	-	-	-
XAircraft SuperX	+	+	+	-	-	+
FY-DOS	+	+	+	-	-	-
FY-41AP	+	+	+	-	+	+
KK	+	-	-	-	-	-
MicroKopter	+	+	+	+	+	+
GU-344	+	-	-	-	-	-
Autoquad	+	+	+	+	+	+
CopterControl	+	-	-	-	-	-

1.4.1.1 MultiWii

Польотний контролер MultiWii (рис. 1.4): популярний через низьку ціну і відкритість вихідних кодів.

Компіляція програми виконується з допомогою безкоштовного середовища розробки Arduino IDE. Останні версії Multiwii мають більшість функцій, необхідних для польоту, в тому числі політ по точкам. Проект некомірційний і підтримується ентузіастами.

Рисунок 1.4 – Контролер MultiWii

1.4.1.2 ArduCopter

Польотний контролер ArduCopter (рис. 1.5): найбільш функціональний серед польотних контролерів з відкритим кодом. Має всі необхідні для польоту функції, у т.ч. автоматичний польот по точкам, накладання параметрів польоту на відео (OSD) утримання позиції та ін. Контролери випускаються компанією 3D Robotics (плати APM 2.5, 2.6), їх ціна складає близько 150$. Існують також китайські клони (HKPilot 2.5), ідентичні по «залізу» і сумісні по прошивкам, їх ціна близько 50$.

Плата APM має лише базову функціональність, решта модулів (GPS, OSD, модем и т.д.) купуються окремо.

Рисунок 1.5 – Контролер ArduCopter

1.4.1.3 Польотні контролери DJI

Польотні контролери DJI випускаються компанією DJI Innovations, мають закриті прошивки і схему. Натепер випускається 3 види контролерів:

DJI Naza-M Lite (рис. 1.6): базова версія, має режими стабілізації польоту та основні GPS-функції (утримання та повернення додому). Не підтримує можливість підключення зовнішніх модулів, в іншому функціональність аналогічна старщій моделі DJI Naza-M V2. Ціна близько 250$.

DJI Naza-M V2: більш нова версія, відрізняється можливістю підключення OSD, модуля Bluetooth або наземної станції, ціна близько 400$.

Рисунок 1.6 – Контролер DJI Naza-M Lite

DJI Wookong (рис. 1.7): професійна версія, ціна близько 1000$.

Рисунок 1.7 – Контролер DJI Wookong

За відгуками користувачів, контролери DJI мають високу стабільність польоту, кращу ніж у більш дешевих моделей. Хоча функціональність самих

контролерів доволі обмежена, її можна значно розширити з допомогою зовнішніх модулів (крім DJI Naza-M Lite).

Наприклад для отримання можливості безпровідного налаштування потрібно придбати додатковий модуль Bluetooth (50$), для ведення розширених польотних логів необхідний DJI iOSD MARK ll (255$), для польоту по точкам, передачі телеметрії або керування з планшета iPad необхідний 2.4G Bluetooth Datalink & iPAD Ground Station (300$). Таким чином головним недоліком цих контролерів є висока ціна як самого контролера так і додаткових модулів.

1.4.1.4 Zero UAV

Польотні контролери Zero UAV (рис. 1.8) виробництва компанії Zero UAV Intelligence Technology, мають закриті прошивки і схему. Випускається 2 види контролерів:

Zero UAV YS-X4: базова версія, має практично всі польотні режими, включаючи польоти по точкам, ведення логів полботу, передачу телеметрії та ін. Версія має обмеження по кількості точок автоматичного польоту і відстані між ними. Контролер більш функціональний ніж DJI Naza за приблизно ту ж ціну (~400$). Одним з недоліків є велика вага контролера, з GPS модулем близько 300г, що робить неможливим використання на легких моделях. Також є нарікання на низьку надійність.

Zero UAV YS-X6: професійна версія, ціна контролера близько 1000$. Існують різні версії контролерів, в залежності від кількості доступних точок автономного польоту ціна може варіювати в діапазоні 1000—2500$.

Рисунок 1.8 – Контролер ZeroUAV

1.4.1.5 KapteinKuk

Контролери сімейства KapteinKuk (рис. 1.9) історично одні з перших та недорогих контролерів для квадрокоптерів. Мають відкритий вихідний код, мінімальний набір датчиків і периферії, за рахунок чого мають мінімальну ціну, що складає 20-30$. Кількість налаштувань і можливостей також мінімальна, однак дякуючи простоті і дешевизні ці моделі мають своїх прихильників та певну ринкову нішу. За великим рахунком, зараз придбання таких контролерів недоцільне.

Рисунок 1.9 – Контролер KapteinKuk

1.4.1.6 MicroCopter

Німецькі контролери MicroKopter (рис. 1.10), історично були найпершими, саме на такому контролері був запущений перший квадрокоптер. Мають високу ціну (від 500 євро) і використовуються в професійній фото і відеозйомці. Налаштування досить заплутане, тому в любительських цілях застосування MicroKopter недоцільне.

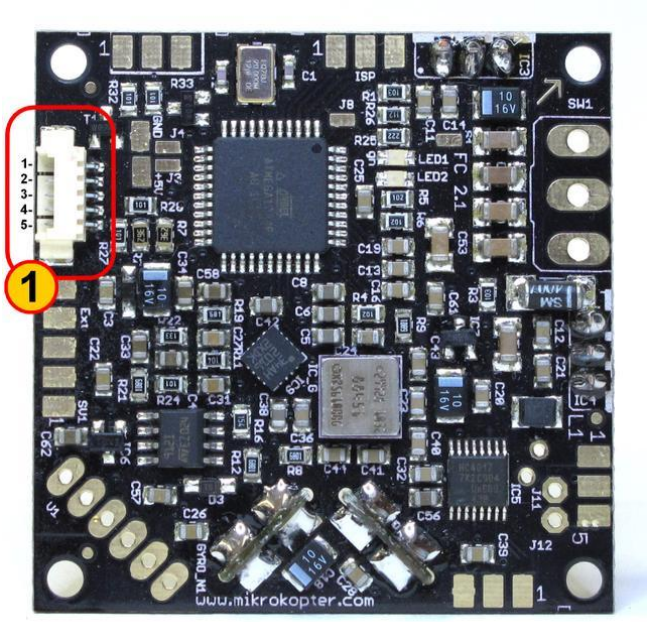


Рисунок 1.10 – Контролер MicroCopter

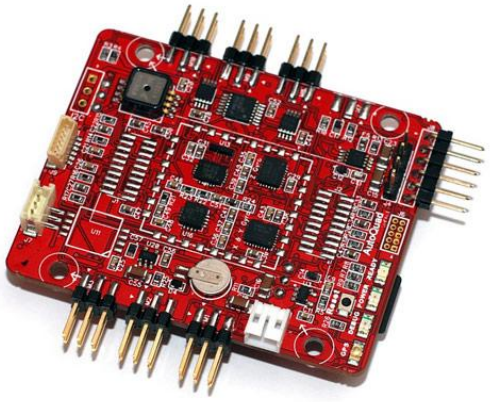
1.4.1.7 GU-344

Квадрокоптер GAUI-330 і контролер GU-344 (рис. 1.11) були першою системою, доступною для любителів на ринку. На сьогодні придбання цього контролера недоцільне, він має лише історичну цінність.

Рисунок 1.11 – Контролер GU-344

1.4.1.8 Auto Quad

Auto Quad (рис. 1.12) – контролер з відкритим вихідним кодом, що позиціонується як кращий для професійного застосування. Має хороші польотні характеристики, однак вкрай складний в налаштуванні, наприклад плата потребує температурного калібрування в морозильній камері з охолодженням до −18 ^{0}C і поступовим прогріванням. Проект підтримується невеликою командою розробників, і його перспективи поки неясні.



1.12 – Контролер AutoQuad

1.4.1.9 Open Pilot

Open Pilot (рис. 1.13) – відкритий проект. Випущені версії контролерів CopterControl, CopterControl3D та OpenPilot Revolution. Настройка контролерів виконується через ПЗ OpenPilot GCS. Можливе використання в мультикоптерах, вертольотах, літаках.

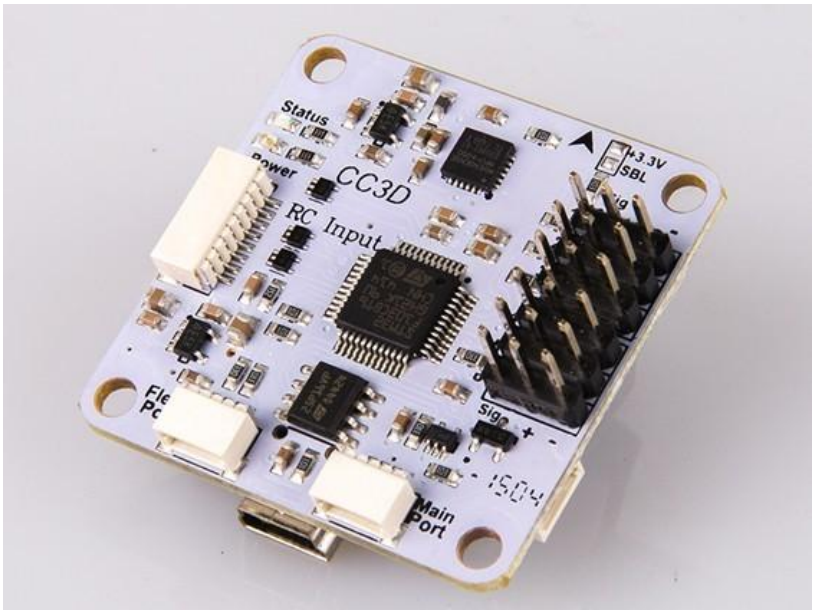


Рисунок 1.13 – Контролер Open Pilot

Існують також інші розробники польотних контролерів, а також аматори та аматорські групи.

Різні польотні контролери можуть використовувати додаткову периферію, необхідну для роботи.

1.4.2 GPS

GPS (рис. 1.14) – використовується для утримання позиції, повернення в точку зльоту або автоматичного польоту по точкам.

Рисунок 1.14 – Модуль GPS з компасом

1.4.3 Телеметрія

Телеметрія (рис. 1.15) - модем (Bluetooth, WiFi) - для дистанційного налаштування і перегляду параметрів польоту можуть використовуватися безпровідні канали зв’язку: модем на 433 або 910 МГц, Bluetooth або WiFi-модуль. Дальність їх роботи відповідно, може складати від 50 м до 10 км.

Рисунок 1.15 – Телеметрія PixHawk

1.4.4 Відеолінк

Відеолінк (рис. 1.16) - для можливості перегляду зображення з мультикоптера, на нього встановлюють міні-камеру та відеопередавач. Частота

передачі зазвичай складає 900МГц, 1.2, 2.4 або 5.8ГГц. Більш високі частоти зручні більш компактними антенами, однак більш низькочастотні сигнали краще огинають перешкоди. Приблизна дальність прийому відеосигналу за допомогою передавача 200-400мВт складає близько кілометра, дальність може бути збільшена або направленими антенами, або більш потужним передавачем.

Для передачі використовують аналогові камери стандартів PAL або NTSC, цифрові канали поки не використовуються через їх дороговизну і більшу масу.

Рисунок 1.16 – Відеолінк Boscam

1.4.5 OSD

OSD (On Screen Display) (рис. 1.17) – підключається між камерою і відеопередавачем, використовується для накладання параметрів польоту (швидкість, висота, координати, і т.д.) на відео (рис. 1.18).

Рисунок 1.17 – OSD Minim

Рисунок 1.18 – Зображення OSD

1.4.6 Підвіс

Підвіс для фото/відеокамери (рис. 1.19) - при встановленні на мультикоптер фото або відеокамери, контролер может керувати моторами підвісу, забезпечуючи стабілізацію камери при нахилянні коптера. Це забезпечує більш плавну відеозйомку і фотозйомку без нахилів та спотворень. Також користувач може сам керувати положенням камери.

Рисунок 1.19 – Підвіс

1.4.7 Стабілізатор

Квадрокоптери призначені для високоякісної відеозйомки обладнуються стабілізаторами (рис. 1.20).

Рисунок 1.20 – Професійний стабілізатор DJI Ronin MX

1.4.8 Камера

Дрони обладнуються любительськими, напівпрофесійними та професійними відеокамерами. Популярними є рішення на базі GoPro (рис. 1.21).

Рисунок 1.21 – БПЛА з камерою GoPro

1.4.9 Прилади руху

Квадрокоптер обладнаний чотирма електромоторами (рис. 1.22) для забезпечення польотної тяги.

Рисунок 1.22 – Електромотор TMotor UAV

Швидкість обертання кожного окремого електромотора регулюється за допомогою регулятора обертів (рис. 1.23).

Рисунок 1.23 – Регулятор обертів Maytech.

Живляться квадрокоптери від акумуляторних батарей (рис. 1.24).

Рисунок 1.24 – Акумуляторна батарея Phantom

1.4.10 Прилади керування

Управляється дрон за допомогою пульта дистанційного керування (рис. 1.25). До пульта може бути підключений планшет чи смартфон на платформі iOS або Android.

Рисунок 1.25 – Пульт дистанційного керування

1.5 Польотна математика

Орієнтація квадрокоптера в просторі визначається та задається кутами тангажу, крену та рискання (pitch, roll, yaw) [11].

Політ квадрокоптера в необхідному напрямку досягається зміною цих трьох кутів. Наприклад, щоб летіти вперед квадрокоптер має нахилитися за рахунок того, що задні мотори крутяться трохи сильніше ніж передні.

Газ квадрокоптера – середнє арифметичне між швидкостями обертання всіх моторів. Чим більший газ – тим більша сумарна тяга моторів, тим сяльніше вони тягнуть квадрокоптер вгору. Газ висіння – мінімальний рівень газу, що необхідний для того, аби квадрокоптер не втрачав висоту.

Щоб керувати квадрокоптером необхідно керувати газом, тангажем та рисканням. Їх називають каналами управління. Для повноцінного управління квадрокоптером, зі змінами режиму польоту, необхідний п’ятиканальний пульт управління. Для управління наклоном та повторотом камери на борту необхідно ще два канали (або окремий пульт для професійних систем відеозйомки).

Існує багато режимів польоту. Для їх реалізації застосовуються GPS, барометр та дальномір. В базовому режимі стабілізації (stab, stabilize) кводрокоптер тримає ті кути, які йому задаються з пульта, незалежно від зовнішніх факторів. В цьому режимі за відсутності вітру квадрокоптер може висіти майже на місці. Вітер доводиться компенсувати оператору.

Напрям обертання гвинтів вибирається не випадково. Якби всі мотори обертались в одну сторону, квадрокоптер би обертався в іншу через створювані обертальні моменти. Тому одна пара протилежних моторів завжди обертається в одну сторону, а друга пара в іншу. Ефект виникнення обертальних моментів використовується, щоб міняти кут рискання: одна пара моторів починає обертатися швидше другої, і квадрокоптер обертається навколо вертикальної осі.

Швидкістю обертання моторів керує польотний контролер.

Узагальненою задачею польотного контролера є кілька десятків разів на секунду виконати цикл управління в який входить: зчитування показів сенсорів, зчитування каналів управління, обробка інформації і видача управляючих сигналів моторам, щоб виконати команди оператора.

Різних видів сенсорів, які можна задіяти, дуже багато. Майже обовязково у всіх квадрокоптерах застосовуються трьохосьовий гіроскоп та трьохосьовий акселерометр. Акселерометр вимірює прискорення, гіроскоп вимірює кутову швидкість. Завдяки їм польотний контролер визначає поточні кути тангажу, крену та рискання. Ці сенсори бувають вмонтованими в польотний контролер, а бувають зовнішніми. Обрахування трьох кутів за показами сенсорів виконує окрема плата (MPU-6050). Вона передає на контролер значення кутів за протоколом i2c. Контролер їх зчитує, обробляє разом з рештою даних і видає управляючі сигнали моторам.

Мотори на мультикоптерах споживають великі рівні струму, тому польотний контролер керує ними не напряму, а через спеціальні апаратні драйвери, що

називаються регуляторами швидкості (ESC). Ці регулятори живляться від бортового акумулятора, управляючий сигнал отримують від контроллера, а на виході мають по три проводи (A, B, C), які безпосередньо йдуть до моторів (для кожного мотора – окремий регулятор).

Контролер програмно керує регуляторами. Найбільш поширені регулятори управляються за допомогою широтно-імпульсної модуляції (ШИМ, PWM) сигналом прямокутної форми з мінімумом 0 Вольт і максимумом 3-5 Вольт.

Щоб дати команду мотору обертатися з максимальною швидкістю контролер має відправляти імпульси тривалістю 2 мс, розмежовані логічним нулем тривалістю 10-20 мс. Тривалість імпульса в 1 мс відповідає зупинці мотора, 1,1 мс – 10% від максимальної швидкості, 1,2 мс – 20% і т.п. Тривалість нуля практично не грає ролі, важлива тільки тривалість самого імпульсу.

Значення діапазону регулювання від 1 до 2 мс не є універсальними, і в залежності від багатьох впливаючих факторів може виявитися, що на практиці діапазон регулювання знаходиться, наприклад, в межах 1,1 — 1,9 мс. Для того, щоб регулятор і контролер «розуміли» один одного існує процедура калібрування. В ході цієї процедури діапазони регуляторів змінюються і стають рівними діапазону контролера. Процедура зашита в програму кожного регулятора і включає в себе кілька простих кроків (процедура може відрізнятися в залежності від виробника):

Відключити живлення регулятора;

Зняти з мотора пропеллер;

Подати на вхід регулятора сигнал, що відповідає максимальній швидкості обертання;

Подати на регулятор живлення, мотор при цьому має бути нерухомим;

Дочекатися звукового сигналу через 1-2 с.;

Подати на вхід регулятора сигнал, що відповідає мінімальній швидкості обертання;

Дочекатися звукового сигналу через 1-2 с.;

Відключити живлення регулятора.

Після цього в регулятор будуть занесені відповідні межі інтервалу. При намаганні злетіти з невідкаліброваними регуляторами наслідки можуть бути несподіваними, від неочікуваного ривка в довільному напрямку до повної нерухомості моторів.

ШІМ (PWM) з точно таким же принципом використовує і бортовий приймач. Це невеликий пристрій, що отримує сигнали радіоуправління з землі і передає їх у польотний контролер. Найчастіше у польотному контролері для кожного каналу управління (газ, тангаж, крен і т.п.) є свій вхід на який надходить ШІМ (PWM). Логіка взаємодії проста, команда, наприклад «70% газ» безперервно йде з землі на приймач, де перетворюється в ШІМ і по окремому проводу надходить у польотний контролер. Аналогічно з тангажем, креном, рисканням.

Приймач і контролер також доведеться калібрувати, оскільки вони спілкуються за допомогою ШІМ. Контролер має підстроїтися під приймач. Процедура калібровки радіо, на відміну від калібровки регуляторів, пишеться як частина польотної програми. Загальний план калібрування такий:

Зняти пропеллери з моторів;

Перевести контролер в режим калібровки радіо;

Контролер запускає калібровку радіо на декілька десятків секунд;

За відведений час рухаємо всіма стіками пульта в усі сторони до упорів;

Контролер запам’ятовує максимуми і мінімуми для всіх каналів у внутрішню пам'ять.

Таким чином, під час калібрування радіо польотний контролер запам’ятовує діапазони приймача по всіх каналах управління; під час калібрування регуляторів діапазон польотного контролера заноситься у всі регулятори.

Крім інтерфейса для польотного контролера необхідна ще одна програма – інтерфейс налаштування польотного контролера. Найчастіше це програма для PC, котра зєднується з польотним контролером по USB і дозволяє користувачу налаштовувати і перевіряти польотну програму, наприклад – запускати калібрування радіо, настроювати параметри стабілізації, перевіряти роботу датчиків, задавати маршрут польоту на карті, визначати поведінку мультикоптера при втраті сигналу і багато іншого.

У випадку незапланованого запуску моторів квадрокоптер може нанести оператору серйозні травми гвинтами, тому необхідно передбачити режими armed/disarmed. Стан квадрокоптера «disarmed» означає, що мотори відключені і навіть команда повного газу з пульта не має ніякого ефекту, хоча живлення подано. Стан «armed» квадрокоптера означає, що команди з пульта виконуються польотним контролером. В цьому стані квадрокоптери злітають, літають і сідають. Квадрокоптер вмикається і повинен відразу потрапити в стан disarmed на той випадок, якщо неуважний пілот вмикає його, тримаючи стік газу на пульті не на нулі. Щоб перевести коптер в стан armed пілоту необхідно зробити передбачений програмно «жест» стіками пульта. Часто цим жестом є утримання лівого стіка в правому нижньому куті (газ = 0%, рискання = 100%) протягом двох секунд. Після цього польотний контролер робить мінімальну самоперевірку і при її успішному проходженні переходить в режим armed (готовий до польоту). Іншим жестом (газ = 0%, рискання = 0%) квадрокоптер «дизармиться». Ще один хороший захід безпеки – автодизарм, якщо газ був на нулі протягом 2-3 секунд.

Налаштування PID-регулятора

(пропорційний-інтегрально-диференціальний регулятор)

Математичний апарат PID-регулятора застосовується майже в усіх задачах стабілізації: стабілізація кутів квадрокоптера в повітрі, політ і утримання позиції по GPS, утримання висоти по барометру, безколекторні механізми стабілізації відеокамери в польоті (підвіс камери).

Якщо, наприклад, на дроні встановлений двохосьовий підвіс з камерою GoPro, а відео зображення з камери вібрує і смикається, хоча всі сенсори відкалібровані і механічні проблеми усунені, то причина у некоректних параметрах PID-регуляторів.

Якщо зібраний квадрокоптер після запуску ледве летить, або навпаки, занадто швидкий і нестабільний, причина та ж - PID-регулятори.

A proportional–integral–derivative controller (PID controller) [12].

Для наочного розуміння роботи PID-регулятора у застосуванні до курування квадрокоптером, розглянемо спрощену модель, квадрокоптер у двомірному просторі, де у нього є тільки один кут – кут крену, і два мотори: лівий та правий (рис. 1.26).

Рисунок 1.26 – 2D квадрокоптер

В польотний контролер безперервно надходять команди з землі: «крен 30 градусів», «крен -10 градусів», «крен 0 градусів (тримати горизонт)»; його задача – якомога швидше і точніше їх виконувати з допомогою моторів з урахуванням: вітру, нерівномірного розподілу ваги квадрокоптера, нерівномірного зносу моторів, інерції квадрокоптера і т.п. Таким чином, польотний контролер має

безперервно вирішувати задачу, яку швидкість обертання передавати на кожний мотор, з урахуванням наявного та заданого значень кута крену. Тут має значення вплив конкретного «заліза». Все зележить від обчислювальних можливостей, наприклад на Arduino можна одну ітерацію циклу обробки і управління вмістити в 10 мілісекунд. Це означає, що раз на 10 мілісекунд будуть зчитуватися значення кутів квадрокоптера, і на їх основі будуть відправлятися керуючі сигнали до моторів. Ці 10 мілісекунд називаються періодом регулювання. Зрозуміло, що чим він менший, тим частіше і точніше відбувається регулювання.

Рівень газу надходить з приймача в контролер. Позначимо його *throttle*.

Як було сказано вище, це середнє арифметичне між швидкостями обертання всіх моторів, виражене в процентах від максимальної швидкості обертання. Якщо *left* та *right* - швидкості обертання лівого та правого моторів, то:

$$left = throttle + force$$

$$right = throttle - force$$

де *force* –реакція квадрокоптера (зусилля), котре створює момент обертання за рахунок того, що лівий мотор обертається на *force* швидше, ніж газ, а правий – на стільки ж повільніше.

force може приймати і від'ємні значення, тоді правий мотор закрутиться швидше. Якщо ми навчимося вираховувати цю величину на кожній ітерації циклу обробки, значить ми зможемо управляти квадрокоптером. Зрозуміло що *force* як мінімум має залежати від поточного кута крену (*roll*) і бажаного кута крену (*target_roll*), який надходить з пульта управління.

Уявимо ситуацію: надходить команда «тримати горизонт» (*target_roll* = 0), а квадрокоптер має крен вліво (рис. 1.27).

Рисунок 1.27 – Квадрокоптер з креном вліво

error – різниця (похибка) між *target_roll* і *roll*, котру контролер намагається мінімізувати.

Чим більша різниця між бажаним кутом крену і поточним, тим сильнішою має бути реакція, тим швидше лівий мотор маж закрутитися по відношенню до правого. Якщо це записати з використанням наших позначень:

*force = P * error*,

де *P* – коефіцієнт пропорційності. Чим він більший, тим сильнішою буде реакція, тим різкіше квадрокоптер буде реагувати на відхилення від необхідного кута крену. Ця інтуїтивно зрозуміла і проста формула описує роботу пропорційного регулятора. Його суть така: чим сильніше квадрокоптер відхилився від необхідного положення, тим сильніше треба намагатися його повернути. Нажаль, цю формулу доведеться ускладнити. Головна причина – перерегулювання.

За кілька десятків мілісекунд (декілька ітерацій циклу обробки) під дією пропорційного регулятора квадрокоптер повернеться в необхідне (в даному випадку горизонтальне) положення. Ввесь цей час похибка *error* та зусилля *force* будуть мати один і той же знак, хоч і ставатимуть усе меншими по модулю.

Набравши якусь швидкість повороту (кутову швидкість) квадрокоптер просто перевалиться на другий бік, бо ніхто його не зупинить у необхідному положенні. Так як пружина, котра завжди намагається повернутися у початкове положення, але якщо її відтягнути і відпустити – буде коливатися, поки тертя не візьме верх. Звичайно, на квадрокоптер також буде діяти тертя, але практика показує, що його не достатньо.

З цієї причини у пропорційний регулятор треба додати ще одну складову, котра буде тормозити обертання квадрокоптера і перешкоджати перерегулюванню (перевалюванню в протилежну сторону) – свого роду імітація тертя у в'язкому середовищі: чим швидше повертається квадрокоптер, тим сильніше треба намагатися його зупинити, звичайно, в розумних межах. Швидкість обертання (швидкість зміни похибки) позначимо як *spin*, тоді:

$$force = P * error + D * spin,$$

де D – коефіцієнт настройки, чим він більший, тим сильніше зупиняюче зусилля. Оскільки швидкість зміни будь-якої величини – це похідна цієї величини по часу:

$$spin = \mathrm{d}\, error / \mathrm{d}t,$$

і ось пропорційний регулятор перетворюється в пропорційно-диференційний (з пропорційною і диференційною складовими).

$$force = P * error + D * \mathrm{d}\, error / \mathrm{d}t.$$

Похибку error вирахувати легко, адже на кожній ітерації ми знаємо *roll* та *target_roll*; P та D – параметри які настроюються перед запуском. Для обрахування похідної (швидкості зміни *error*) необхідно зберігати попереднє значення *error*,

знати поточне значення *error* і знати час, що пройшов між вимірюваннями (період регулювання). Оскільки швидкість = відстань / час:

$$spin = \mathrm{d}\,error / \mathrm{d}t \quad (error - error_{previous}) / \Delta t,$$

де Δt – період регулювання; $error_{previous}$ – значення похибки з попередньої ітерації циклу регулювання. Ця формула – найпростіший спосіб чисельного диференціювання, і вона нам повністю тут підходить.

Тепер у нас є пропорційно-диференційний регулятор в пласкому «бікоптері», але залишилась ще одна проблема. Нехай лівий край буде важити трохи більше правого, або, що те ж саме, лівий мотор працює трохи гірше правого.

Квадрокоптер трохи нахилений вліво і не повертається назад: диференційна складова дорівнює нулю, а пропорційна складова хоч і приймає додатнє значення, але його недостатньо, щоб повернути квадрокоптер в горизонтальне положення, адже лівий край важить трохи більше правого. Як наслідок – квадрокоптер буде весь час тягнути вліво.

Необхідний механізм, який би відслідковував такі відхилення і виправляв їх. Характерною особливістю таких похибок є те, що вони проявляють себе з часом. На допомогу приходить інтегральна складова, збільшуючи реакцію *force* і квадрокоптер приймає необхідний кут крену.

Тут є нюанс. Припустимо *error* дорівнює 1 градусу, цикл регулювання – 0,1 с. Тоді за одну секунду сума похибок прийме значення 10 градусів. А якщо цикл обробки – 0,01 с, то сума набере аж 100 градусів. Щоб за один і той же час інтегральна складова набирала одне й те ж значення при різних періодах регулювання, отриману суму будемо множити на сам період регулювання. Легко порахувати, що в обох випадках з наведеного прикладу виходить сума в 1 градус. Ось вона – інтегральна складова (поки що без настройочного коефіцієнта):

$\Delta t \, \Sigma \, error$.

Ця формула – не що інше, як чисельний інтеграл по часу функції *error* в інтервалі від нуля до поточного моменту. Саме тому ця складова називається інтегральною:

$$\int_0^T error \, \mathrm{d}t \approx \Delta t \sum error,$$

де T – поточний момент часу.

Остаточна формула пропорційно-інтегрально-диференційного регулятора:

$$force = P * error + I * \Delta t \sum error + D * \frac{error - error_{previous}}{\Delta t},$$

де *I* - один з настроюваних параметрів, котрих тепер три: *P, I, D*. Ця формула зручна в застосуванні програмного коду, а ось формула, яка наводиться в підручниках:

$$force = P * error + I * \int_0^T error \, \mathrm{d}t + D * \frac{\mathrm{d}\, error}{\mathrm{d}x}.$$

Існує декілька її варіацій, наприклад, можна обмежити модуль інтегральної складової, щоб він не перевищував певний допустимий поріг.

Якщо під час настроювання регулятора квадрокоптер надто повільно реагує на управління – можна збільшити P, але надто велике значення P може призвести до перерегулювання. З перерегулюванням може справитися параметр D, але його надто великі значення призведуть до частих коливань, або знову до

перерегулювання. Параметр I, зазвичай, в 10 – 100 разів менший, ніж параметр P, оскільки його сила в накопиченні у часі, а не в швидкому реагуванні.

Ручна настройка PID-параметрів потребує практики. Існують аналітичні методи їх обчислення, але вони потребують хорошої підготовки та точного знання багатьох параметрів конкретної системи, що настроюється. Як середнє між ручним підбором і аналітичним обчисленням є широкий ряд емпіричних методів, запропонованих різними дослідниками.
В нашому 2D квадрокоптері змінюється тільки один кут – кут крену. В справжньому 3D квадрокоптері необхідні три незалежних PID-регулятори для кожного з кутів, а управління конкретним мотором буде представляти суму зусиль по всім регуляторам.

1.6 Зразки схем квадрокоптера на мікроконтролері Arduino

У даному підрозділі наведено схеми квадрокоптерів на основі мікроконтролера Arduino YUN (рис. 1.28) [13] та Arduino Uno (рис. 1.29) [14].

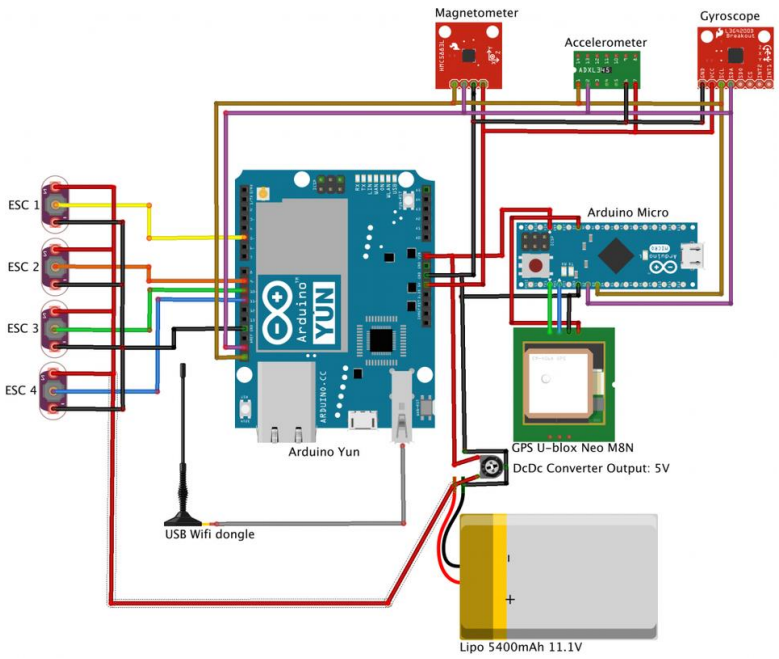


Рисунок 1.28 – Схема квадрокоптера на основі мікроконтролера Arduino YUN

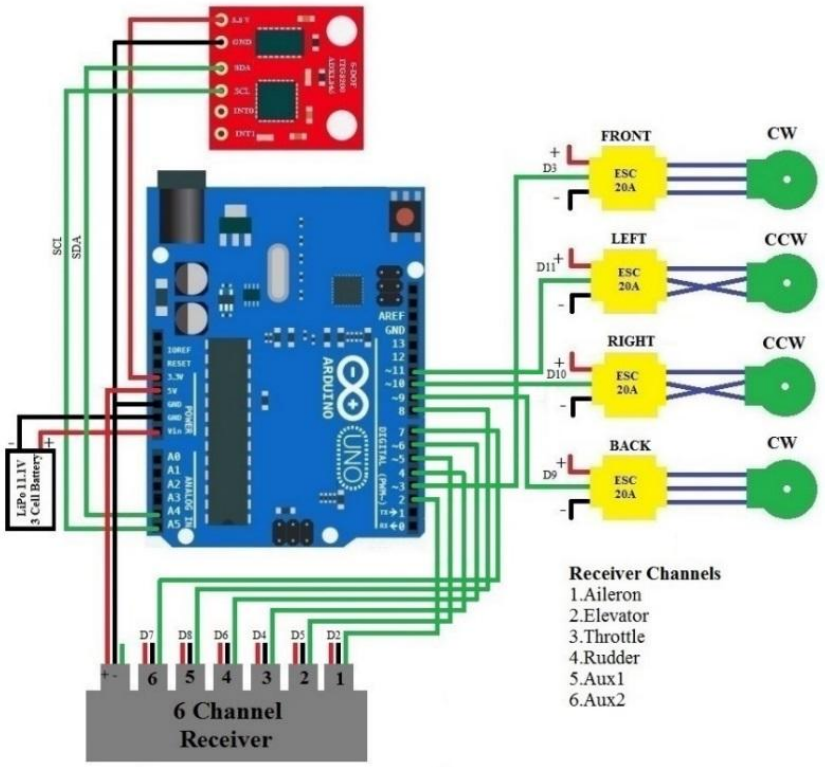


Рисунок 1.29 – Схема квадрокоптера на основі мікроконтролера Arduino Uno

Також наведено функціональну схему квадрокоптера (рис. 1.30).

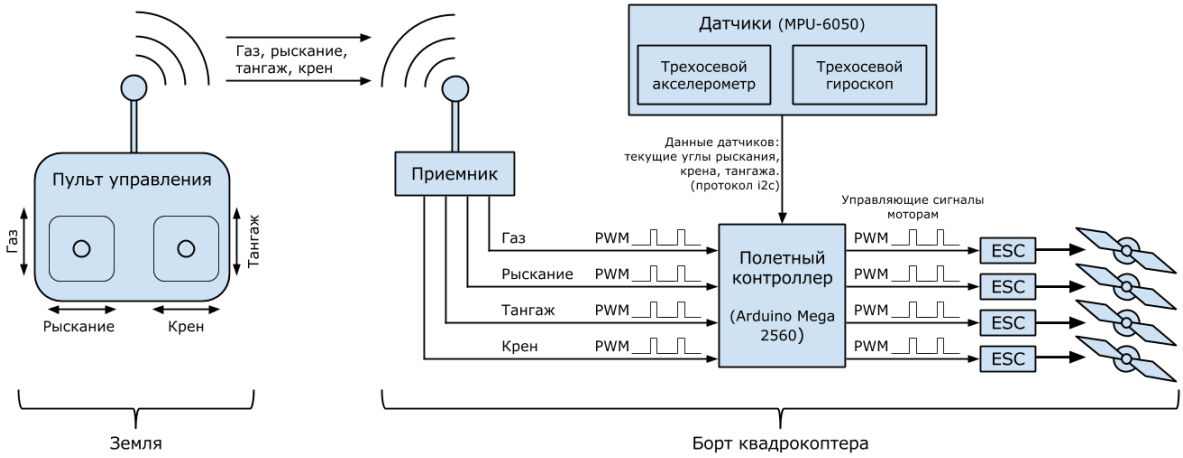


Рисунок 1.30 – Функціональна схема квадрокоптера

1.7 Висновок

В даному розділі було проаналізовано предметну область дослідження, розглянуто призначення, область застосування та види безпілотних літальних апаратів, наведено схеми та складові частини дронів, було досліджено законодавчу базу, що регулює БПЛА. Було приведено порівняння різних польотних контролерів, виявлено їх переваги і недоліки. На основі розглянутого матеріалу було зроблено висновок, що робота з автоматизації відеозйомки за допомогою компактного бузпілотного літального апарату є доцільною.

2 Вибір засобів реалізації

2.1 Розрахунок маршруту та його відображення

Unity [15] — кросплатформенний інструмент для розробки дво- та тривимірних застосунків та ігор, що працює на операційних системах Windows і OS X. Створені за допомогою Unity застосунки працюють під системами Windows, OS X, Android, Apple iOS, Linux, а також на гральних консолях Wii, PlayStation 3 і XBox 360.

Є можливість створювати інтернет-застосунки за допомогою спеціального під'єднуваного модуля для браузера Unity, а також за допомогою експериментальної реалізації в межах модуля Adobe Flash Player. Застосунки, створені за допомогою Unity, підтримують DirectX та OpenGL.

Редактор Unity має простий Drag & Drop інтерфейс, який легко налаштовувати, що складається з різних вікон, завдяки чому можна проводити налагодження гри прямо в редакторі. Рушій підтримує дві скриптові мови: C # і JavaScript. Проект в Unity ділиться на сцени (рівні) - окремі файли, що містять свої ігрові світи зі своїм набором об'єктів, скриптів, і налаштувань. Сцени можуть містити в собі як, об'єкти (моделі), так і порожні ігрові об'єкти – тобто ті які не мають моделі. Об'єкти, в свою чергу містять набори компонентів, з якими і взаємодіють скрипти. Також у них є назва (в Unity допускається наявність двох і більше об'єктів з однаковими назвами), може бути тег (мітка) і шар, на якому він повинен відображатися. Так, у будь-якого предмета на сцені обов'язково присутній компонент Transform - він зберігає в собі координати місця розташування, повороту і розмірів по всіх трьох осях. У об'єктів з видимою геометрією також за замовчуванням присутній компонент Mesh Renderer, що робить модель видимою.

Також Unity підтримує фізику твердих тіл і тканини, фізику типу Ragdoll (ганчіркова лялька). У редакторі є система успадкування об'єктів; дочірні об'єкти будуть повторювати всі зміни позиції, повороту і масштабу батьківського об'єкта. Скрипти в редакторі прикріплюються до об'єктів у вигляді окремих компонентів.

При імпорті текстури в рушій можна згенерувати alpha-канал, mip-рівні, normal-map, light-map, карту відображень, проте безпосередньо на модель текстуру прикріпити не можна - буде створено матеріал, з яким буде призначений шейдер, і потім матеріал прикріпиться до моделі. Редактор Unity підтримує написання і редагування шейдерів. Крім того він містить компонент для створення анімації, яку також можна створити попередньо в 3D-редакторі та імпортувати разом з моделлю, а потім розбити на файли.

Unity ідеально підходить для двох цілей одночасно: розрахунок маршруту та візуальне відображення. Можливі й інші варіанти, такі як CryEngine, або UnrealEngine, проте Unity показав себе як найпрстіший, найшвидший і найзрозуміліший інструмент для розробки системи, тому було вибрано саме його.

З двох варіантів скриптової мови (C#; JavaScript) було вибрано C# через її більшу зрозумілість і кращу документованість.

2.2 Емуляція польотного контролеру БПЛА

Arduino Uno [16] – дешевий і зручний мікроконтроллер, що програмується знайомою мені мовою C++. До того ж я маю досвід праці з цим мікроконтроллером з виробничої практики. Є можливість працювати з платою не через Arduino IDE, але немає потреби. Це середовище розробки є простим, зручним і призначеним для роботи з платами Arduino.

2.3 Висновок

В даному розділі проведено вибір засобів реалізації, що найкраще підходять для використання при розробці системи автоматизації відеозйомки. Для розрахунку маршруту та водночас для його відображення було вибрано ігровий рушій Unity. Для написання скриптів було вибрано мову C#. Емуляцію польотного контролера буде зроблено за допомогою мікроконтролера Arduino Uno. Код для якого буде написано на мові C++ у середовищі Arduino IDE.

3 Реалізація та аналіз результатів

3.1 Постановка задачі на реалізацію

Під час написання дипломної роботи мною була створена модель автоматизації процесу відеозйомки за допомогою компактного безпілотного літальногго апарату (рис. 3.1).

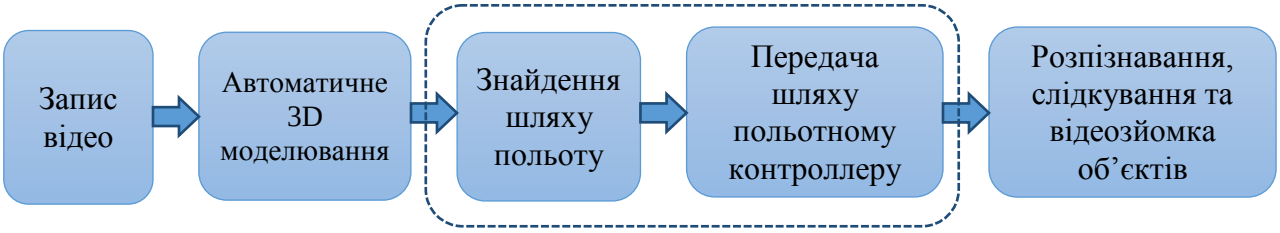


Рисунок 3.1 – Модель автоматизації процесу відеозйомки.

До реалізації у дипломній роботі були прийняті два пункти цієї моделі:

Знайдення шляху польоту;

Передача шляху польотному контроллеру.

Після написання дипломної роботи, розроблена система має працювати згідно з наступними пунктами:

Оператор вводить координати певної віддаленої від БПЛА точки у просторі;

БПЛА має знайти й використати найкоротший безпечний шлях;

У випадку виникнення на шляху нових перешкод під час руху, БПЛА має знайти новий шлях.

В процессі роботи система має передавати поточні координати БПЛА разом з його значенням рискання на емулятор польотного контроллера, де дані будуть прийняті.

3.2 Аналіз алгоритмів пошуку шляху

В теорії графів, задача про найкоротший шлях полягає в знаходженні такого шляху між двома вершинами (або вузлами) графу, що сума ваг ребер з яких він складається мінімальна. Під час переддипломної практики були розглянуті наступні 6 алгоритмів.

3.2.1 Алгоритм Дейкстри

Алгоритм Дейкстри [17] — алгоритм на графах, відкритий Дейкстрою. Знаходить найкоротший шлях від однієї вершини графа до всіх інших вершин. Класичний алгоритм Дейкстри працює тільки для графів без циклів від'ємної довжини.

Найпростіша реалізація алгоритма Дейкстри потребує $O(V^2)$ дій. У ній використовується масив відстаней та масив позначок. На початку алгоритму відстані заповнюються великим позитивним числом (більшим максимального можливого шляху в графі), а масив позначок заповнюється нулями. Потім відстань для початкової вершини вважається рівною нулю і запускається основний цикл.

Нехай u — вершина, від якої шукаються відстані, V — множина вершин графа, d_i — відстань від вершини u до вершини i, $w_{(i, j)}$ — вага «ребра» (i, j).

Алгоритм:

1. Множина вершин U, до яких відстань відома, встановлюється рівною {u}.
2. Якщо U=V, алгоритм завершено.
3. Потенційні відстані Di до вершин з V\U встановлюються нескінченними.
4. Для всіх ребер (i, j), де i∈U та j∈V\U, якщо $D_j > d_i + w_{(i, j)}$, то D_j присвоюється $d_i + w_{(i, j)}$.
5. Шукається i∈V\U, при якому D_i мінімальне.

6. Якщо Di дорівнює нескінченності, алгоритм завершено. В іншому випадку di присвоюється значення D_i, U присвоюється U∪{i} і виконується перехід до кроку 2.

3.2.2 Алгоритм Беллмана-Форда

Алгоритм Беллмана—Форда [18] — алгоритм пошуку найкоротшого шляху в зваженому графі. Знаходить найкоротші шляхи від однієї вершини графа до всіх інших. На відміну від алгоритму Дейкстри, алгоритм Беллмана—Форда допускає ребра з негативною вагою. Запропоновано незалежно Річардом Беллманом і Лестером Фордом.

Алгоритм носить ім'я двох американських вчених: Річарда Беллмана (Richard Bellman) і Лестера Форда (Lester Ford). Форд фактично винайшов цей алгоритм в 1956 році при вивченні іншої математичної задачі, підзадача якої звелася до пошуку найкоротшого шляху в графі, і Форд зробив начерк остаточного вирішення завдання цього алгоритма. Беллман в 1958 р. опублікував статтю, присвячену конкретно завданню знаходження найкоротшого шляху, і в цій статті він чітко сформулював алгоритм у тому вигляді, в якому він відомий нам зараз.

Алгоритм маршрутизації RIP (алгоритм Беллмана — Форда) був вперше розроблений в 1969 році, як основний для мережі ARPANET.

Вхідними даними для алгоритму є граф G, вагова функція w та стартова вершина s. Потрібно знайти найкоротші шляхи від вершини s до всіх вершин графа. Алгоритм Беллмана-Форда повертає логічне значення, яке вказує на те, чи міститься в графі цикл з негативною вагою, досяжний з витоку. Якщо такий цикл існує у графі G, алгоритм повідомляє, що найкоротших шляхів не існує. Якщо ж таких циклів немає, алгоритм видає найкоротші шляхи і їх вагу.

Сам алгоритм Форда-Беллмана представляє з себе кілька фаз. На кожній фазі проглядаються всі ребра графа, і алгоритм намагається справити релаксацію (relax,

ослаблення) уздовж кожного ребра (u,v) ваги w(u,v). Релаксація вздовж ребра — це спроба поліпшити значення v.d значенням v.u+w(u,v). Фактично це означає, що ми намагаємося поліпшити значення для вершини v, користуючись ребром (u,v) і поточним значенням для вершини u. Стверджується, що достатньо |G.V|-1 фази алгоритму, щоб коректно порахувати довжини всіх найкоротших шляхів у графі (цикли негативної ваги відсутні). Для недосяжних вершин відстань v.d залишиться нескінченністю.

Якщо граф заданий списком ребер: ініціалізація потребує O(V) часу, кожен з |V|-1 проходів потребує O(E) часу, прохід по усім ребрам для перевірки наявності негативного циклу займає O(E) часу. Отже алгоритм працює за O(VE) часу.

Якщо граф заданий матрицею суміжності, то алгоритм буде виконуватись за $O(E^3)$часу.

3.2.3 Алгоритм пошуку A*

Алгоритм пошуку A* [19] («A зірочка» або англ. «A star») — належить до евристичних алгоритмів пошуку. Використовується для пошуку найкоротшого шляху між двома вершинами графу з додатніми вагами ребер. Описаний 1968 р. Пітером Хартом, Нільсом Нільсоном та Бертрамом Рафаелєм.

Алгоритм використовує допоміжну функцію (евристику), аби скеровувати напрям пошуку та скорочувати його тривалість. Алгоритм повний в тому сенсі, що завжди знаходить оптимальний розв'язок, якщо він існує.

Алгоритм A* спершу відвідує ті вершини, які ймовірно ведуть до найкоротшого шляху до мети. Аби розпізнати такі вершини, кожній відомій вершині x співставляється значення f(x), яке дорівнює довжині найкоротшого шляху від початкової вершини до кінцевої, який пролягає через обрану вершину. Вершини з найменшим значенням f обираються в першу чергу.

Функція f(x) для вершини x визначається так:

$f(x)=g(x)+h(x)$

де:

$g(x)$ – функція, значення якої дорівнюють вартості шляху від початкової вершини до x,

$h(x)$ – евристична функція, оцінює вартість шляху від вершини x до кінцевої.

Використана евристика не повинна давати завищену оцінку вартості шляху. Прикладом оцінки може служити пряма лінія: загальний шлях не може бути коротшим за пряму лінію.

3.2.4 Алгоритм Флойда-Воршелла

В інформатиці, алгоритм Флойда-Воршелла [20] використовується для розв'язання задачі про найкоротший шлях у зваженому графі з додатними або від'ємними вагами ребер (але без від'ємнозначних циклів). При звичайній реалізації алгоритм видасть довжини (сумарні ваги) найкоротших шляхів між всіма парами вершин, хоча він не видасть інформацію про самі шляхи. Різні версії алгоритму також використовуються для знаходження транзитивного замикання в відношенні R, або (враховуючи Метод Шульце), для знаходження найбільшого шляху (англ. widest path problem) між всіма парами вершин у зваженому графі.

Алгоритм Флойда-Воршелла — це приклад динамічного програмування. Його було опубліковано в звичній сьогодні формі Робертом Флойдом 1962 року. Проте, це практично той самий алгоритм, що було опубліковано Бернардом Роєм 1959 року і Стівеном Воршеллом 1962 року для знаходження транзитивного замикання в графі, і є досить тісно пов'язаним з алгоритмом Кліні (опублікованим 1956 року) для перетворення детермінованих скінченних автоматів у регулярні вирази. Сучасне формулювання алгоритму, як трьох вкладених циклів було вперше подано Пітером Інгерманом 1962 року.

Алгоритм також називають Алгоритм Флойда, Алгоритм Роя-Воршелла, Алгоритм Роя-Флойда, або Алгоритм WFI.

Алгоритм Воршелла порівнює всі можливі шляхи в графі між кожною парою вершин. Він виконується за $\Theta(|V|^3)$ порівнянь. Це доволі примітивно, враховуючи, що в графі може бути до $\Omega(|V|^2)$ ребер, і кожну комбінацію буде перевірено. Він виконує це шляхом поступового поліпшення оцінки по найкоротшому шляху між двома вершинами, поки оцінка не стає оптимальною.

Розгляньмо граф G з ребрами V, пронумерованими від 1 до N. Крім того розгляньмо функцію shortestPath(i, j, k), яка повертає найкоротший шлях від i до j, використовуючи вершини з множини {1,2,…,k} як внутрішні у шляху. Тепер, маючи таку функцію, нам потрібно знайти найкоротший шлях від кожного i до кожного j, використовуючи тільки вершини від 1 до k + 1.

Для кожної з цих пар вершин, найкоротший шлях може бути або (1)- шлях, у якому є тільки вершини з множини {1, …, k}, або (2)- шлях, який проходить від i до k + 1 а потім відk + 1 до j.Найкоротший шлях від i to j that only uses vertices 1 через k визначається функцією shortestPath(i, j, k), і якщо є коротший шлях віді до (k + 1 до j), то довжина цього шляху буде сумою(конкатенацією) найкоротшого шляху віді до k + 1 (використовуючи вершини{1, …, k}) і найкоротший шлях від k + 1 до j (також використовуючи вершини з {1, …, k}).

w(i,j) — це вага ребра між i таj. Можна визначити shortestPath(i, j, k + 1) наступною рекурсивною формулою:

shortestPath(i,j,0)=w(i,j)

рекурсивна частина:

shortestPath(i,j,k+1)=min(shortestPath(i,j,k),
shortestPath(i,k+1,k)+shortestPath(k+1,j,k))

Ця формула є основною частиною алгоритму Флойда-Воршелла. Алгоритм спочатку обчислює shortestPath(i, j, k) для всіх пар(i, j) де k = 1, потім k = 2, і т. д.

Цей процес продовжується, поки k = N, і поки не знайдено всі найкоротші шляхи для пар (i, j).

3.2.5 Алгоритм Джонсона

Алгоритм Джонсона [21] дозволяє знайти найкоротші шляхи між усіма парами вершин зваженого орієнтованого графа. Цей алгоритм працює, якщо у графі містяться ребра з позитивною чи негативною вагою, але відсутні цикли з негативною вагою.

Якщо в алгоритмі Дейкстри неспадними чергу з пріоритетами реалізована у вигляді фібоначчійової купи, то час роботи алгоритму Джонсона дорівнює $O(V^2 \log V + VE)$. При простіший реалізації неубутною черги з пріоритетами час роботи стає рівним $O(VE \log V)$, але для розріджених графів ця величина в асимптотичному межі веде себе краще, ніж час роботи алгоритму Флойда-Воршелла.

3.3 Алгоритм A*

Мною був обраний алгоритм A* через його швидкодію та зручність. Алгоритм був застосований до графу, що був побудований на тривимірній системі координат з цілочисленними координатами вузлів. Нижче наведена блок-схема алгоритму (рис. 3.2).

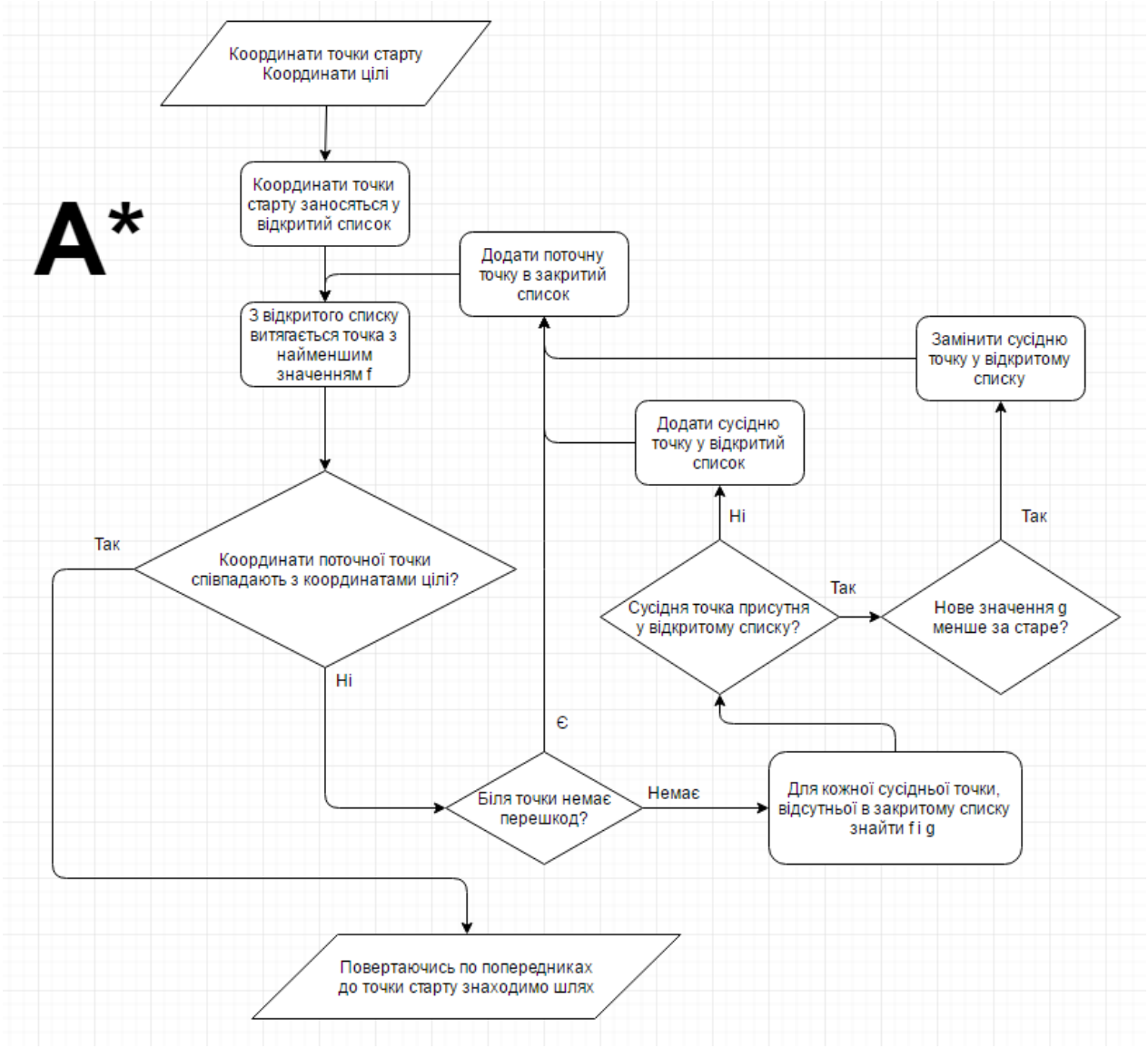


Рисунок 3.2 – Блок-схема алгоритму A*

3.4 Аналіз евристик

3.4.1 Евристика

Евристичний [22] — термін стосується способу вибору цілі або напрямку в розв'язуванні задачі, правильність якого на кожному кроці невідома або не може бути підтверджена. Такі методи як генетичний алгоритм або нейронна сітка використовують для приймання рішень евристичні способи, які можуть

ґрунтуватися на суто емпіричній інформації, що не піддається суворій раціоналізації. Використовується в комбінаторній хімії та при плануванні експериментів.

Евристичні методи (друга назва Евристики) дозволяють пришвидшити процес розв'язання задачі. Значний інтерес до їх дослідження виник у зв'язку з можливістю вирішення ряду задач (розпізнавання об'єктів, доведення теорем і т. д.), в яких людина не може дати точний алгоритм вирішення з допомогою технічних засобів. Метою Евристики є побудова моделей процесу розв'язання якої-небудь нової задачі.

3.4.2 Евристика в A*

Евристична функція h(n) дає алгоритму A* приблизну мінімальну відстань від будь якої вершини n до цілі. Важливо використовувати доцільну евристичну функцію [23].

Евристика може бути використана для контролю за поведінкою алгоритму.

Якщо h(n) менше або дорівнює ціні руху від n до цілі, то A* гарантовано знайде найкоротший шлях. Наприклад, якщо h(n) дорівнює 0 і лише g(n) грає роль, A* перетворюється на алгоритм Дейкстри, що гарантовано знаходить найкоротший шлях. Чим меншим є значення h(n), тим більше вузлів алгоритм досліджує вшир, що робить алгоритм повільнішим.

Якщо h(n) точно дорівнює ціні руху від n до цілі, тоді A* буде мати найбільшу швидкодію і досліджувати найменшу кількість вузлів, знаходячи найкоротший шлях.

Якщо h(n) більше ціни руху до цілі, не можна стверджувати, що алгоритм гарантовано знайде найкоротший шлях, проте швидкодія може збільшитись.

3.4.3 Манхеттенська відстань

Стандартною евристикою для квадратної сітки по якій можна пересуватися у чотирьох напрямках є Манхеттенська відстань (рис. 3.3). Ця евристика чисельно дорівнює відстані між сусідніми клітинами D помноженій на суму модулів різниць відповідних координат.

```
function heuristic(node) =
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    return D * (dx + dy)
```

За відсутності перешкод і незмінному значенні D, переміщення на один крок ближче до цілі повинне підвищувати g на D і зменшити h на D. В сумі вони дають незмінне значення f.

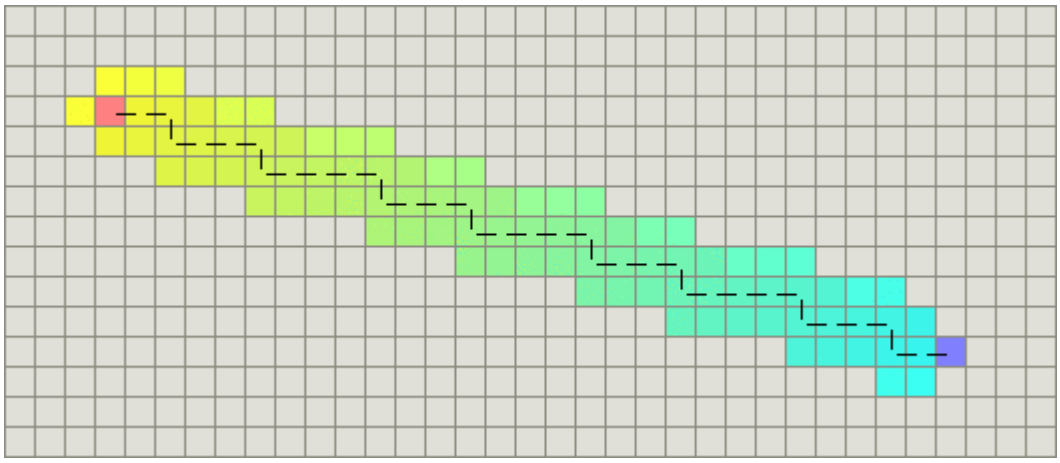


Рисунок 3.3 – Манхеттенська відстань

3.4.4 Діагональна відстань

Якщо дозволений перехід по діагоналі, доцільніше використовувати діагональну евристику (рис. 3.4).

```
function heuristic(node) =
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    return D * (dx + dy) + (D2 - 2 * D) * min(dx, dy)
```

де D – ціна руху за однією координатою, а D2 – ціна руху по діагоналі.

Тут ми вираховуємо максимальну кількість переміщень по діагоналі та залишок за однією координатою, перемножуємо їх на D2 і D відповідно та сумуємо добутки.

Випадок, коли D = 1 і D2 = 1 називається відстанню Чебишева, D = 1 і D2 = sqrt(2) – октантною відстанню.

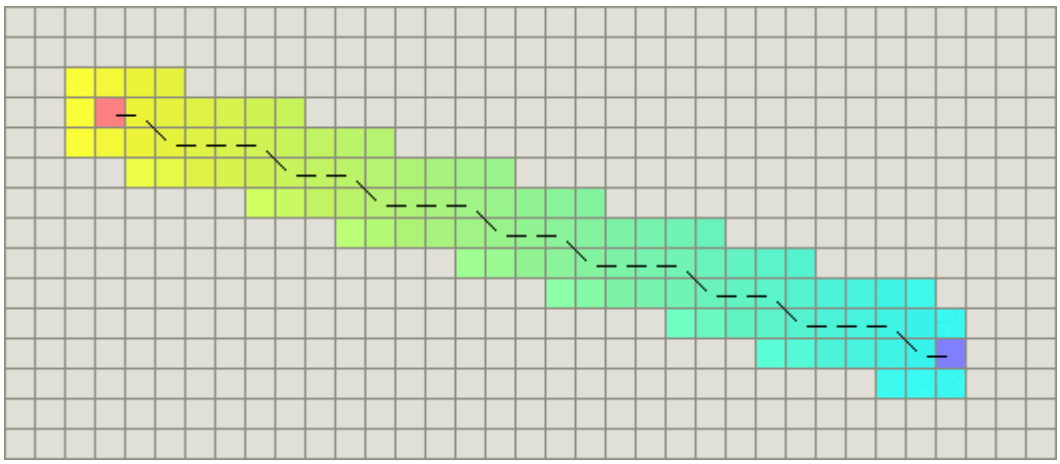


Рисунок 3.4 – Діагональна відстань

3.4.5 Евклідова відстань

Евклідова відстань (рис. 3.5) – точна відстань між двома точками. Вона дорівнює квадратному кореню із суми квадратів відстаней по кожній координаті.

```
function heuristic(node) =
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    return D * sqrt(dx * dx + dy * dy)
```

Оскільки Евклідова відстань менша за можливий найкоротший шлях, алгоритм буде розглядати більше точок, тому буде працювати повільніше.

Рисунок 3.5 – Евклідова відстань

3.4.6 Квадрат Евклідової відстані

Можна використовувати квадрат Евклідової відстані.

```
function heuristic(node) =
    dx = abs(node.x - goal.x)
```

```
dy = abs(node.y - goal.y)
return D * (dx * dx + dy * dy)
```

Ця евристика працює значно швидше попередньої, оскільки при підрахунку значення евристики просто не бере корінь, а також, оскільки через переоціненість h над g, швидше прямує до цілі, проте при перешкодах у вигляді комірок A* може потрапляти у пастки (рис. 3.6). Метод є доцільним, якщо є можливість виправити помилки алгоритму без особливих затрат.

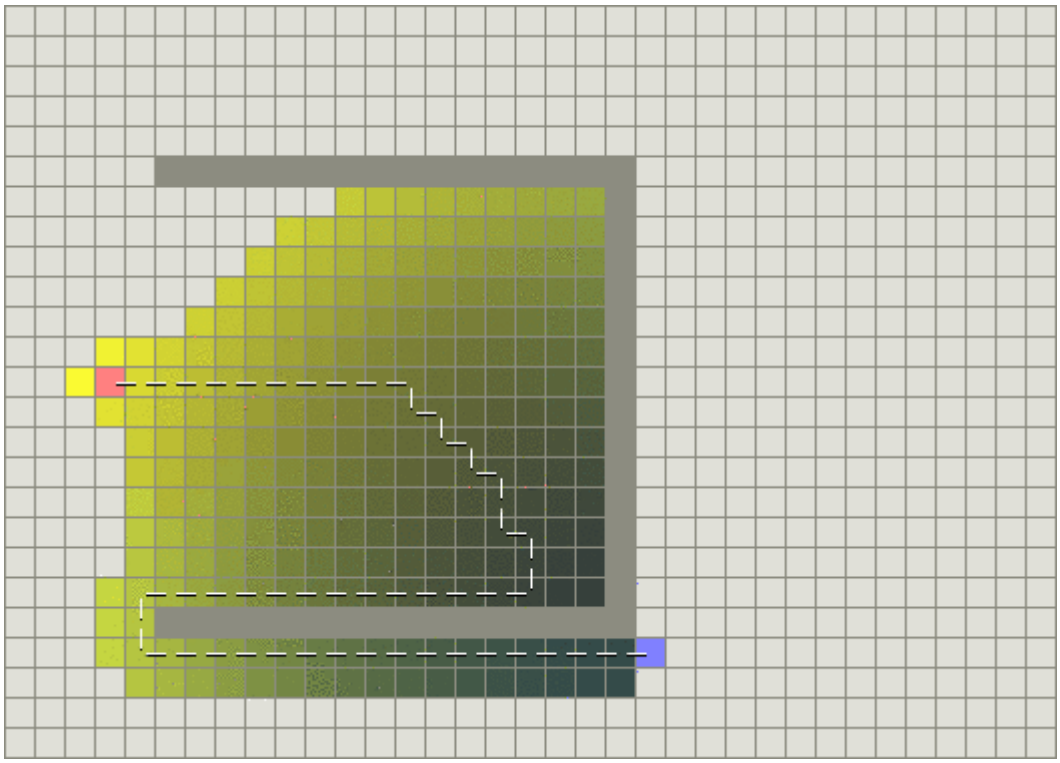


Рисунок 3.6 – Квадрат Евклідової відстані

3.5 Візуалізація

Для візуалізації роботи алгоритму (рис. 3.7) в середовищі Unity були використані об’єкти класу GameObject для позначення дрона, цілі та перешкод. Перешкоди мають колайдери, на які реагує алгоритм, оминаючи їх.

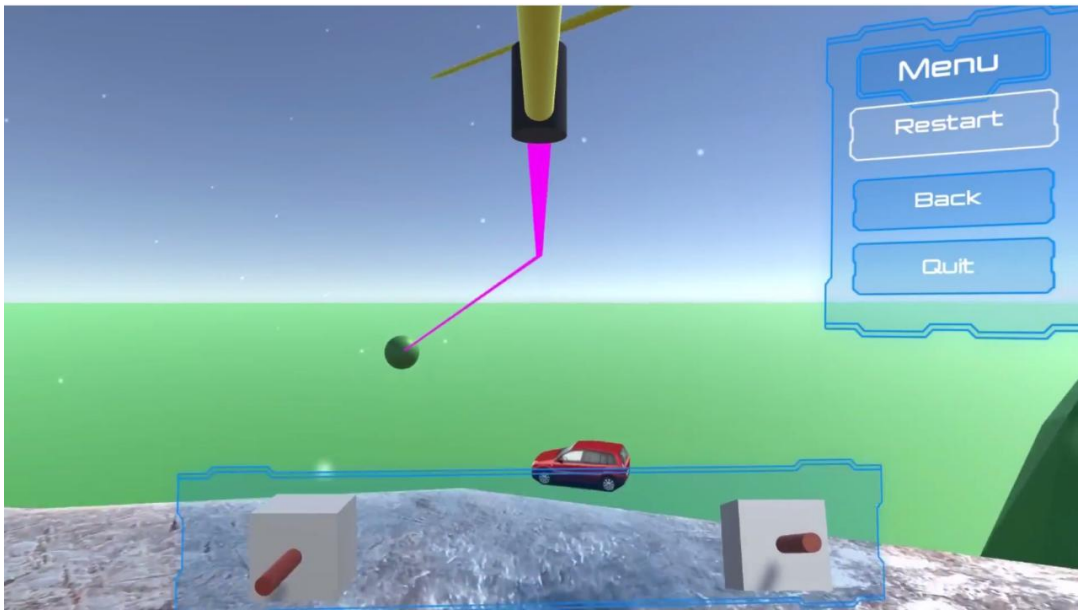


Рисунок 3.7 – Візуалізація системи

3.6 Елементи зовнішнього середовища

БПЛА у просторі Unity має вигляд чорно-жовтого квадрокоптера (рис. 3.8), до моторів якого було застосовано скрипт rotator.cs. Скрипт приймає у вигляді змінної напрям обертання і обертає об’єкт навколо власної осі.

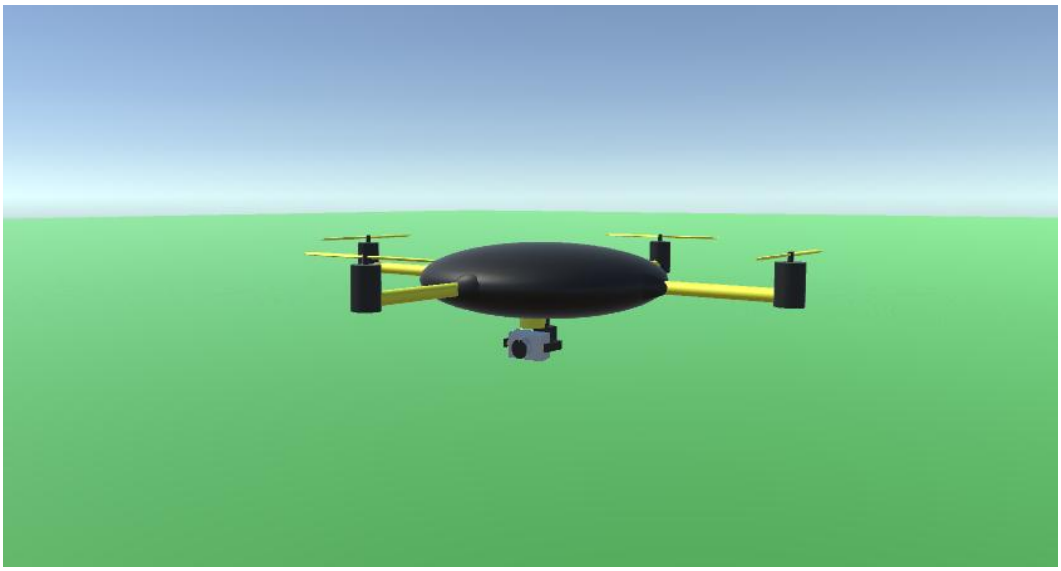


Рисунок 3.8 – БПЛА

Окрім нерухомих перешкод у вигляді дерев (рис. 3.9), на сцені присутня рухома перешкода у вигляді ковадла (рис. 3.10), що падає з неба на шляху в БПЛА. До ковадла застосовано скрипт anvil.cs, що через певний час, що був вказаний аргументом скрипта, швидко переміщує ковадло до землі.

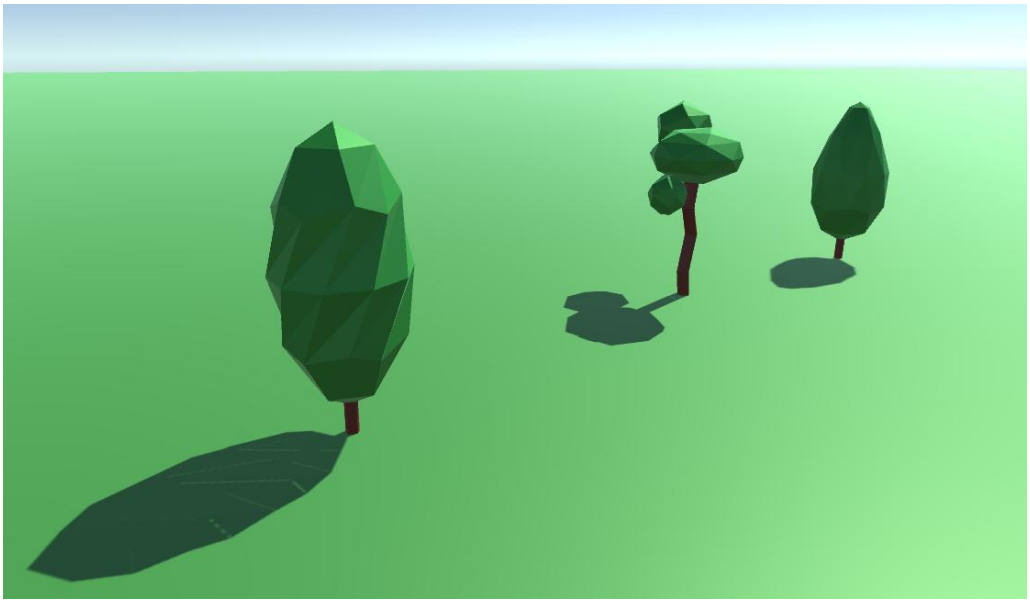


Рисунок 3.9 – Дерева

Рисунок 3.10 – Ковадло

Як об’єкт, за яким можна спостерігати, на сцені присутня машина (рис. 3.11), що рухається по колу за допомогою скрипта Car.cs. Скрипт приймає аргументами точку навколо якої буде їздити машина й відстань від цієї точки. В даній сцені машина їздить навколо одного з дерев.

Для виконання задач відеозйомки під дроном закріплено камеру (рис. 3.12), що працює за допомогою скрипта CameraController.cs. Камера може працювати у двох режимах:

ручному (дивиться в напрямку, вказаному оператором);

трекінговому (переслідує ціль, що була передана скрипту у вигляді аргумента).

Рисунок 3.11 – Машина

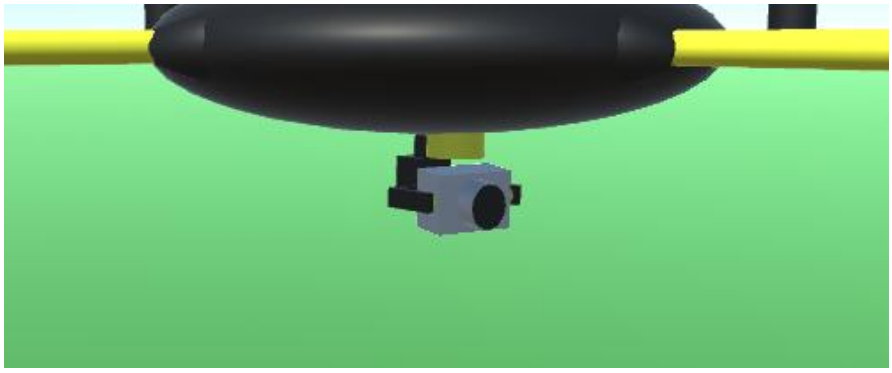


Рисунок 3.12 – Камера

Для зручності користування було створено меню (рис. 3.13), за допомогою якого можна запустити, перезапустити та зупинити роботу системи [24]. До того ж, внизу екрана була створена панель із двома стіками керування дроном (рис. 3.14), що відображають параметри керування дроном у кожен момент часу. Вісь Y лівого стіка відповідає за газ, вісь X – за рискання, вісь Y правого стіка – за тангаж, X – за крен. За положення стіків відповідає скрипт stickRot.cs.

Рисунок 3.13 – Меню

Рисунок 3.14 – Панель зі стіками керування

3.7 Основний скрипт

e3.cs – скрипт, що прикріплений до дрона, він містить три C# класи. Перший клас Node позначає точку у просторі, містить у своїх полях три просторові координати, значення g і c, де g – відстань від точки старту, c – відстань від попередньої точки, а також адресу попереднього вузла. В A* шлях шукається переходами між вузлами з класом Node. В класі є метод GetNeighbor, котрий створює новий вузол, сусідній з даним для додання його у відкритий список. Другий клас Queue є чергою з пріоритетими, для діставання з відкритого списку вузла з найменшим значенням f. Третій клас, власне e3, має 5 основних методів:

LineBuilder:

Переглядає відкритий список, бере вузол з найменшим значенням f, перевіряє, чи не дісталися ми цілі, запускає ExpandNode для кожного вузла.

ExpandNode:

Перевіряє відсутність перешкод біля вузла. Ігноруючи сусідів із закритого списку, перевіряє відкритий список на наявність в ньому сусідів. У разі відсутності сусідів у відкритому списку, додає їх туди, у разі присутності, перевіряє, чи новий шлях не коротший за попередній. Якщо новий шлях коротший, замінює адресу попередника сусідньої точки на адресу даної.

LineSmootherer:

Перевіряє готовий шлях на наявність зайвих точок. Якщо без певної точки шлях не зустрічає перешкод, без цієї точки можна обійтись. Цей метод виконує дві важливі функції: зменшує кількість поворотів на шляху, а також виправляє помилки занадто жадібної евристики квадратної відстані від цілі.

LineDrawer:

Після снаходження шляху рисує рожеву лінію між точками шляху в просторі Unity для наглядності.

LineChecker:

На кожному кадрі перевіряє, чи не з’явилися на шляху нової перешкоди. В разі виявлення нової перешкоди, зупиняє БПЛА, чекає секунду і заново перераховує шлях. Алгоритм чекає секунду, поки система перейде в стабільний стан. Наприклад, при падінні на шлях з неба ковадла, не доцільно знаходити шлях на одну координату нижче, адже на наступному кадрі ковадло впаде і на новий шлях.

Весь код пректу доступний за посиланням: https://github.com/Caesar-8C/UAV.

3.8 Виявлення перешкод

Для виявлення перешкод було використано функцію Unity Physics.OverlapBox(). Ця функція перевіряє точку на наявність колайдерів на певній відстані від неї. Певна відстань була вибрана виходячи з розмірів справжнього дрона та його ймовірних польотних характеристик такою, щоб справжній дрон при поворотах, відхиляючись від планованого шляху, не міг зачепити собою жодної перешкоди.

3.9 Підключення контролера

Використовуючи метод SerialPort.Write() із скрипта e3.cs виконується передача даних із середовища Unity до мікроконтролера Arduino Uno. Для самого мікроконтролера був написаний скетч sketch.ino, котрий приймає дані з Unity. Цим з'єднанням передаються чотири дробових числа: три просторові координати та значення кута рискання. Дані передаються 10 разів на секунду. Використовуваний мікроконтролер емулює польотний мікроконтролер справжнього дрона, що базуючись на переданих даних буде оперувати моторами БПЛА.

3.10 Результати роботи і їх аналіз

В результаті роботи було розроблено модель автоматизації відеозйомки за допомогою компактного безпілотного літального апарату і реалізовано її частину. До реалізованої частини моделі відносяться побудова найкоротшого шляху для БПЛА і емуляція його передачі на польотний контролер справжнього дрона. В середовищі Unity була побудована 3D модель простору, в якому БПЛА має знайти найкоротший шлях до цілі. Модель будується достатньо простим способом, що дозволяє їй легко описувати навіть складні умови. Для знаходження найкоротшого шляху було написано скрипт, котрий працює на основі алгоритму пошуку шляху A*. Було розглянуто декілька різних евристик, а саме: Манхеттенська відстань, діагональна відстань, Евклідова відстань, а також квадрат Евклідової відстані. Аналіз результатів показав, що всі евристики дають задовільний результат за точністю, але евристика квадрату Евклідової відстані показала найкращий результат за часом роботи. В результаті підключення мікроконтролера було досягнуто емуляції передачі даних про положення дрона на польотний контролер справжнього дрона. Частота передачі є достатньо високою, дані не втрачаються, цілі було досягнуто.

3.11 Подальший розвиток

В подальших дослідженнях планується реалізувати модель повністю. Плани включають автоматичну побудову 3D моделі простору, базуючись на відеозйомці стереокамерою або на даних сонару, що знаходяться на дроні. Два методи будуть порівняні, з них буде вибрано найкращий. Замість емулятора польотного контролера буде використано справжній польотний контролер, який буде встановлено на справжньому дроні. Щодо відеозйомки, буде розроблено програму, що буде аналізувати відео з камери дрона і застосовувати до нього штучний інтелект. Наприклад, розпізнавати образи, використовуючи методи бібліотеки OpenCV [25].

3.12 Висновок

В даному розділі було проведено реалізацію моделі автоматизації відеозйомки за допомогою компактного безпілотного літального апарату. Було написано скрипт для знаходження найкоротшого шляху до цілі. Результат роботи даного скрипта було візуалазовано в середовищі Unity. Було емульовано передачу даних від реалізованої системи до польотного контролера дрона за допомогою мікроконтролера Arduino Uno.

4 Функціонально-вартісний аналіз програмного продукту

4.1 Вступ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для автоматизації процесу відеозйомки за допомогою компактного безпілотного літального апарату. Програмний продукт був розроблений за допомогою мови програмування C# у середовищі розробки Unity.

Програмний продукт призначено для використання на персональних комп’ютерах під управлінням операційної системи Windows.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

Визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

для кожної функції визначаються повні річні витрати й кількість робочих часів.

для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

4.2 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки системи аналізу нелінійних нестаціонарних процесів. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу гетероскедастичних процесів в економіці та фінансах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;

забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

передбачати мінімальні витрати на впровадження програмного продукту.

4.2.1 Обґрунтування функцій програмного продукту

Головна функція F0– розробка програмного продукту, розраховує найкоротший безпечний маршрут для БПЛА. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F1 – вибір мови програмування;

F2 – створення 3D моделі;

F3 – вибір евристики.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F1:

а) мова програмування C#;

б) мова програмування JavaScript.

Функція F2:

а) створення 3D моделі вручну;

б) автоматичне створення 3D моделі.

Функція F3:

а) Евклідова відстань;

б) квадрат Евклідової відстані.

4.2.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (табл. 4.1).

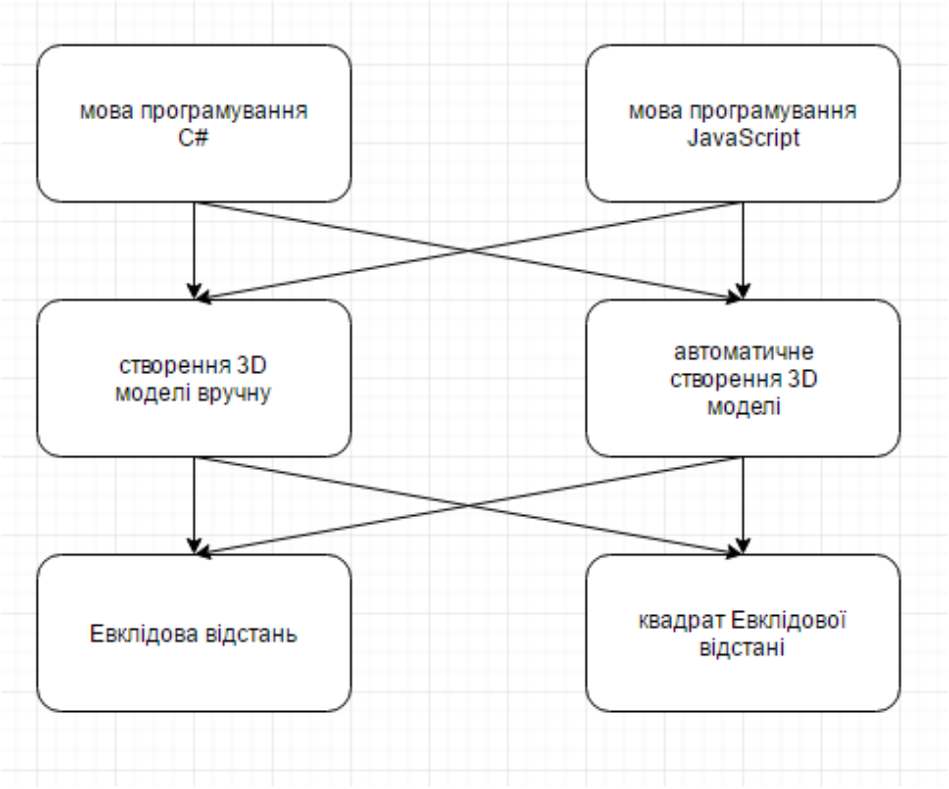


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

На основі аналізу позитивно-негативної матриці (табл. 4.1) робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Оскільки проект є великим доцільніше використовувати варіант а), до того ж недоцільно спеціально вивчати іншу мову програмування, тому варіант б) має бути відкинутий.

Таблиця 4.1 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Гарно документована, легка у застосуванні, відома, краще підходить для великих проектів	–
	Б	Краще підходить для малих проектів	Необхідність вивчання
F2	А	Доцільно для одноразового використання	Необхідність перебудови 3Д моделі при багаторазовому використанні
	Б	Доцільно для багаторазового використання	Нераціональні витрати ресурсів для одноразового використання
F3	*А*	Завжди знаходить найкоротший шлях	Великі витрати часу через розгляд за великої кількості вузлів
	Б	Невеликі витрати часу через більш жадібне прямування до цілі	Не завжди знаходить найкоротший шлях

Функція *F2:*

Оскільки метою даної роботи є розробка алгоритму знаходження найкоротшого шляху, буде достатньо ручного моделювання. Систему автоматичного моделювання буде доцільніше реалізовувати на наступному кроці розробки проекту автоматизації відеозйомки з використанням компактного безпілотного літального апарату, тому зараз варіант б) має бути відкинутий

Функція *F3:*

Обидві евристики є доцільними для застосування, тому вважаємо варіанти а) та б) гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3а
2. F1а – F2а – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів ПП

4.3.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

X1 – швидкодія програми;

X2 – об'єм пам'яті для збереження даних;

X3 – точність результату;

X4 – потенційний об'єм програмного коду;

X 5–час розрахунку значення функції.

X1: Відображає швидкодію операцій залежно від обраної евристики.

X2: Відображає об'єм пам'яті в оперативній пам'яті персонального комп'ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає наскільки велике відхилення знайденого шляху від ідеального.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

X 5:Відображає час, який витрачається на розрахунок значення функцій.

4.3.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

Таблиця 4.2 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Швидкодія програми	X1	мс	6000	4200	800
Об’єм пам’яті для збереження даних	X2	Мб	32	16	8
Точність результату	X3	%	10	5	0
Потенційний об’єм програмного коду	X4	кількість строк коду	2000	1000	500
Час розрахунку значення функції	X5	мкс	200	70	20

За даними таблиці 4.2 будуються графічні характеристики параметрів (рис. 4.2 – рис. 4.6)

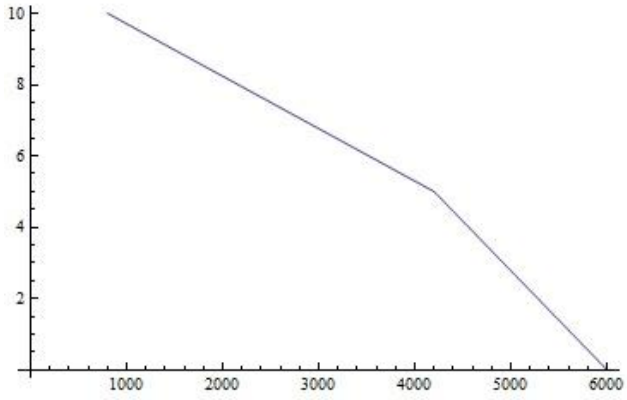


Рисунок 4.2 – X1, швидкодія програми

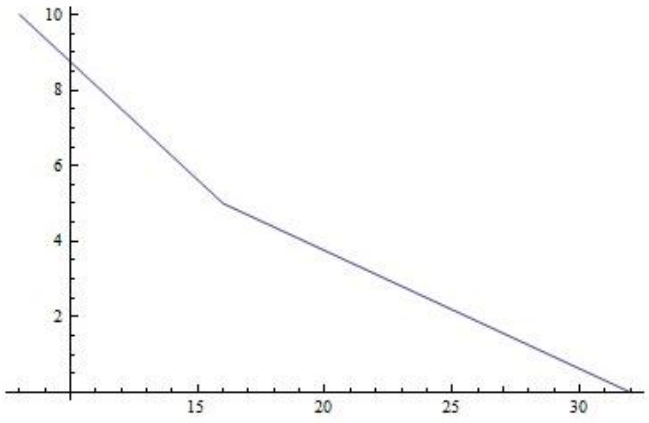


Рисунок 4.3 – X2, об’єм пам’яті для збереження даних

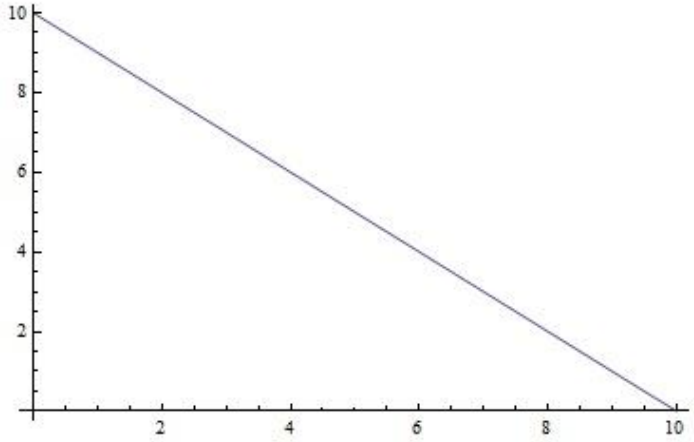


Рисунок 4.4 – X3, точність результату

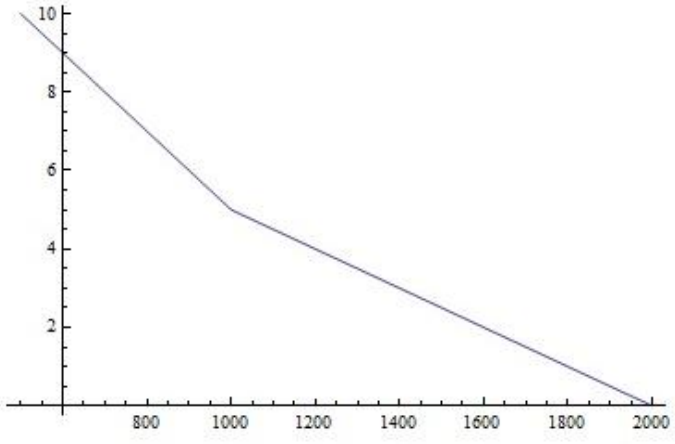


Рисунок 4.5 – X4, потенційний об’єм програмного коду

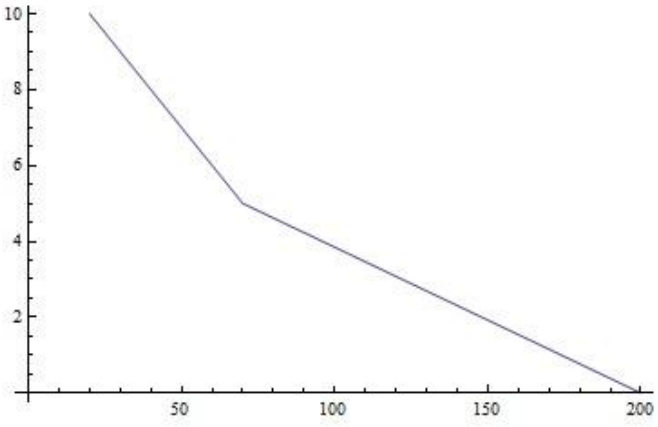


Рисунок 4.6 – X5, час розрахунку значення функції

4.3.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

Визначення рівня значимості параметра шляхом присвоєння різних рангів;

Перевірку придатності експертних оцінок для подальшого використання;

Визначення оцінки попарного пріоритету параметрів;

Обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ	Δ^2
			1	2	3	4	5	6	7			
X1	Швидкодія програми	мс	5	6	5	6	5	5	6	38	17	289
X2	Об’єм пам’яті для збереження даних	Мб	3	2	3	3	3	4	2	20	-1	1
X3	Точність результату	%	2	2	3	2	2	2	3	16	-5	25
X4	Потенційний об’єм програмного коду	кількість строк коду	3	2	2	2	3	2	1	15	-6	36
X 5	Час розрахунку значення функції	мкс	2	3	2	2	2	2	3	16	-5	25
	Разом		15	15	15	15	15	15	15	105	0	376

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105,$$

де N=7 – число експертів, n=5 – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 21,$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T,$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 376.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 376}{7^2(5^3-5)} = 0{,}767 > W_k = 0{,}67,$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1,5
X1 і X3	>	>	>	>	>	>	>	>	1,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X1 і X5	>	>	>	>	>	>	>	>	1,5
X2 і X3	>	=	=	>	>	>	<	>	1,5
X2 і X4	=	=	>	=	=	>	>	=	1
X2 і X5	>	<	>	>	>	>	<	>	1,5
X3 і X4	<	=	>	=	<	=	>	=	1
X3 і X5	=	<	>	=	=	=	=	=	1
X4 і X5	>	<	=	=	>	=	<	=	1

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається за формулою:

$$a_{ij} \quad \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{в\ i}$ за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{де } b_i = \sum_{i=1}^{N} a_{ij}.$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j.$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j					Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	X5	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	1,5	1,5	1,5	1,5	7,0	0,28	34	0,287	160,75	0,287
X2	0,5	1,0	1,5	1,0	1,5	5,5	0,22	25,5	0,215	120,25	0,215
X3	0,5	0,5	1,0	1,0	1,0	4,0	0,16	18,75	0,158	88,75	0,158
X4	0,5	1,0	1,0	1,0	1,0	4,5	0,18	21,5	0,181	101,5	0,181
X5	0,5	0,5	1,0	1,0	1,0	4,0	0,16	18,75	0,158	88,75	0,158
Всього:						25	1	118,5	1	560	1

4.4 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо. Абсолютні значення параметрів *X 2* (об’єм пам’яті для збереження даних) та *X1* (швидкодія програми) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X 3* (точність результату) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 800 мс або варіанту б) 80мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (табл. 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{i,j} B_{i,j},$$

де n – кількість параметрів; K_i – коефіцієнт вагомості i–го параметра; B_i – оцінка i–го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X4)	А	500	6,8	0,181	1,231
F2(X3)	А	3	7,2	0,158	1,138
F3(X2,X1)	А	16	5	0,215	1,075
	Б	800	10	0,287	2,870

За даними з таблиці 4.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

$K_{K\,1}$ = 1,231+1,138+1,075=3,444;

K_{K2} = 1,231+1,138+2,870=5,239.

Як видно з розрахунків, кращим є другій варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.5 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка логічної частини програмного продукту;

2. Розробка візуальної частини програмного продукту.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}, \quad (5.1)$$

де T_P – трудомісткість розробки ПП; $K_П$ – поправочний коефіцієнт; $K_{СК}$ – коефіцієнт на складність вхідної інформації; $K_М$ – коефіцієнт рівня мови програмування; $K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто T_P =27 людино-днів, $K_П$ =0.9, $K_{СК}$ = 1, $K_{СТ}$ =0.8:

$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44$ людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$T_I = (122.4 + 19.44 + 4.8 + 19.44) \cdot 8 = 1328{,}64$ людино-годин;

T_{II} = (122.4 + 19.44 + 6.91 + 19.44) · 8 = 1345.52 людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 20000 грн., один фінансовий аналітик з окладом 25000 грн. Визначимо зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} грн.,$$

де М – місячний оклад працівників; T_m – кількість робочих днів тиждень; t – кількість робочих годин в день.

$$С_ч = \frac{20000 + 20000 + 25000}{3 \cdot 21 \cdot 8} = 128{,}97 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д$,

де $С_ч$– величина погодинної оплати праці програміста; $Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП}$ = 128,97· 1328,64 · 1.2 = 205622,86 грн.

II. $С_{ЗП}$ = 128,97· 1345.52 · 1.2 = 208238,06 грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 22%:

I. $С_{ВІД} = С_{ЗП}$ · 0.22 = 205622,86 · 0.22 = 45237,03 грн.

II. $С_{ВІД} = С_{ЗП}$ · 0.22 = 208238,06 · 0,22 = 45812,37 грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 20000 грн., з коефіцієнтом зайнятості 0,2, то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 20000 \cdot 0,2 = 48000$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 48000 \cdot (1 + 0.2) = 57600$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0,22 = 57600 \cdot 0,22 = 12672$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 30000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 30000 = 8625$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 30000 \cdot 0.05 = 1725$ грн.,

де $К_Р$ –відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 1,94 = 3310,42$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнт зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 30000 \cdot 0,67 = 20100$ грн.

Тоді, річні експлуатаційні витрати будуть:

$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$

$С_{ЕКС} = 57600 + 12672 + 8625 + 1725 + 3310,42 + 20100 = 104032,42$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М-Г} = С_{ЕКС} / Т_{ЕФ} = 104032,42/1706,4 = 60,97$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$С_М = С_{М-Г} \cdot Т$

I. $С_М = 64,28 \cdot 1328,64 = 85404,98$ грн.;

II. $С_М = 64,28 \cdot 1345,52 = 86490,03$ грн.

Накладні витрати складають 67% від заробітної плати:

$С_Н = С_{ЗП} \cdot 0,67$

I. $С_Н = 205622,86 \cdot 0,67 = 137767,32$ грн.;

II. $С_Н = 208238,06 \cdot 0,67 = 139519,50$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$

I. $С_{ПП} = 205622,86 + 45237,03 + 85404,98 + 137767,32 = 474032,19$ грн.

II. $С_{ПП} = 208238,06 + 45812,37 + 86490,03 + 139519,50 = 480059,96$ грн.

4.6 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$К_{ТЕРj} = К_{Кj} / С_{Фj}$,

$К_{ТЕР1} = 5,214 / 474032,19 = 1,10 \cdot 10^{-5}$;

$К_{ТЕР2} = 3,569 / 480059,96 = 0,74 \cdot 10^{-5}$.

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}= 1,1 \cdot 10^{-5}$.

4.7 Висновок

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконанняфункціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 1,1 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

Мова програмування – C#;

Створення 3D моделі вручну;

Квадрат Евклідової відстані.

Даний варіант виконання програмного комплексу дає користувачу необхідну швидкодію без втрат точності та зміни кількості коду.

ВИСНОВКИ

В цій роботі було досліджено предметну область і розроблено модель автоматизації процесу відеозйомки з використанням компактного безпілотного літального апарату. Дипломна робота включає в себе розробку двох частин цієї моделі: знаходження оптимального шляху та передачу даних до польотного контролера.

Для рішення першої задачі було використано ігровий рушій Unity для побудови 3D моделі простору. Цю модель було використано для знаходження найкоротшого шляху та запису відео. Були вивчені різні алгоритми для знаходження шляху і було прийнято рішення, що алгоритм A* найкраще підходить для рішення цієї задачі. Було написано C# скрипт, що знаходить найкоротший шлях між двома об’єктами в Unity використовуючи модифікований алгоритм A*. Щодо запису відео, дрон в Unity обладнано відеокамерою, що може керуватися вручну або слідкувати за об’єктом Unity.

Другу задачу було вирішено за допомогою мікроконтролера Arduino Uno. Координати, а також значення рискання дрона з Unity передаються на мікроконтролер, що емулює справжній БПЛА.

Базуючись на цих досягненнях, я планую розвивати проект сконструювавши справжній БПЛА, що буде обладнано відеокамерою для автоматичної побудови 3D моделі простору, та додати штучного інтелекту, що дозволить розпізнавати та слідкувати за об’єктами методами бібліотеки OpenCV.

ПЕРЕЛІК ПОСИЛАНЬ

1. COMMUNICATION FROM THE COMMISSION TO THE EUROPEAN PARLIAMENT AND THE COUNCIL – EASA – [Електронний ресурс] – Режим доступу: https://www.easa.europa.eu/system/files/dfu/Communication_Commission_Drones.pdf – Дата доступу: 09/06/2017.
2. Civil drones – EASA – [Електронний ресурс] – Режим доступу: https://www.easa.europa.eu/easa-and-you/civil-drones-rpas – Дата доступу: 09/06/2017.
3. DroneRules – [Електронний ресурс] – Режим доступу: http://dronerules.eu/en/ – Дата доступу: 09/06/2017.
4. ‘Prototype’ Commission Regulation on Unmanned Aircraft Operations – EASA – [Електронний ресурс] – Режим доступу: https://www.easa.europa.eu/system/files/dfu/UAS%20Prototype%20Regulation%20final.pdf – Дата доступу: 09/06/2017.
5. National RPAS Regulations – Eurocontrol – [Електронний ресурс] – Режим доступу: http://www.eurocontrol.int/articles/national-rpas-regulations – Дата доступу: 09/06/2017.
6. Advisory Circular – Federal Aviation Administration – [Електронний ресурс] – Режим доступу: https://www.faa.gov/uas/media/AC_107-2_AFS-1_Signed.pdf – Дата доступу: 09/06/2017.
7. United States Court of Appeals – [Електронний ресурс] – Режим доступу: https://www.cadc.uscourts.gov/internet/opinions.nsf/FA6F27FFAA83E20585258125004FBC13/$file/15-1495-1675918.pdf – Дата доступу: 09/06/2017.

8. Правила гарного дрону – Axon – [Електронний ресурс] – Режим доступу: http://axon.partners/uk/uncategorized/the-rules-of-good-drone/ – Дата доступу: 09/06/2017.
9. Drone.UA – Офіційний дилер продукції компании DJI в Україні. – [Електронний ресурс] – Режим доступу: http://drone.ua/dji/dji/?gclid=CLr72ZTUodQCFQWUsgodw50FCQ – Дата доступу: 09/06/2017.
10. Полётный контроллер – Вікіпедія – [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Полётный_контроллер – Дата доступу: 09/06/2017.
11. Программируем квадрокоптер на Arduino – Habrahabr – [Електронний ресурс] – Режим доступу: https://habrahabr.ru/post/227425/ – Дата доступу: 09/06/2017.
12. PID Controller – Вікіпедія – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/PID_controller – Дата доступу: 09/06/2017.
13. Arduino-based drone – Atmel – [Електронний ресурс] – Режим доступу: https://atmelcorporation.wordpress.com/2015/06/08/this-diy-quadcopter-is-built-around-an-arduino-yun/ – Дата доступу: 09/06/2017.
14. AeroQuad Forum – [Електронний ресурс] – Режим доступу: http://aeroquad.com/showthread.php?951-AeroQuad-v1-8-v1-9-Shield/page7 – Дата доступу: 09/06/2017.
15. Unity – [Електронний ресурс] – Режим доступу: https://unity3d.com/ – Дата доступу: 09/06/2017.
16. Arduino UNO – [Електронний ресурс] – Режим доступу: https://www.arduino.cc/en/main/arduinoBoardUno – Дата доступу: 09/06/2017.
17. Алгоритм Дейкстри – Вікіпедія – [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Алгоритм_Дейкстри – Дата доступу: 09/06/2017.

18. Алгоритм Беллмана—Форда – Вікіпедія – [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Алгоритм_Беллмана—Форда – Дата доступу: 09/06/2017.
19. Алгоритм пошуку A* – Вікіпедія – [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/ Алгоритм_пошуку_A* – Дата доступу: 09/06/2017.
20. Алгоритм Флойда — Воршелла – Вікіпедія – [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Алгоритм_Флойда_—_Воршелла – Дата доступу: 09/06/2017.
21. Алгоритм Джонсона – Вікіпедія – [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Алгоритм_Джонсона – Дата доступу: 09/06/2017.
22. Евристика – Вікіпедія – [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Евристика – Дата доступу: 09/06/2017;
23. Heuristics – Red Blob Games – [Електронний ресурс] – Режим доступу: http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html – Дата доступу: 09/06/2017.
24. Creating a Main Menu – Unity You Tube – [Електронний ресурс] – Режим доступу: https://www.youtube.com/watch?v=OWtQnZsSdEU – Дата доступу: 09/06/2017.
25. OpenCV – [Електронний ресурс] – Режим доступу: opencv.org – Дата доступу: 09/06/2017.