

## АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці програмного забезпечення для безпілотних літальних апаратів літакового типу з метою полегшення налаштування параметрів безпілотного літального апарата та пульта керування. У рамках роботи розроблено прикладну програму та прошивку під Arduino, що дозволяє користувачеві інтуїтивно налаштовувати параметри безпілотного літального апарата та пульта, після чого ці параметри передаються через com-port до платформи Arduino. Arduino, спільно з модулем nrf24L01, використовується для керування безпілотного літального апарата, виступаючи у ролі "польотного контролера".

Робота вирішує актуальну проблему складності налаштування параметрів безпілотного літального апарата для некваліфікованих користувачів шляхом розробки інтуїтивно зрозумілої програми та використання доступної та економічно вигідної технології на базі Arduino. Результати роботи включають розроблену програму, що дозволяє легко налаштовувати параметри безпілотного літального апарата та пульта керування, а також прошивку для Arduino, що забезпечує ефективне управління безпілотного літального апарата з використанням переданих параметрів. Використання Arduino для розробки програмного забезпечення для безпілотних літальних апаратів є доцільним з точки зору економічної вигоди та легкості доступу до технології для авіамоделістів.

## ANOTATION

The qualification work is dedicated to the development of software for fixed-wing unmanned aerial vehicles (unmanned aircraft) to facilitate the configuration of aircraft parameters and the control unit. As part of the work, an application program and firmware for Arduino were developed, allowing the user to intuitively configure the parameters of the unmanned aircraft and the control unit, after which these parameters are transmitted via com-port to the Arduino platform. Arduino, together with the nrf24L01 module, is used to control the unmanned aircraft, acting as a "flight controller."

The work addresses the current problem of the complexity of setting unmanned aircraft parameters for unqualified users by developing an intuitive program and using accessible and cost-effective technology based on Arduino. The results of the work include a developed program that allows easy configuration of unmanned aircraft and control unit parameters, as well as firmware for Arduino that ensures efficient control of the unmanned aircraft using the transmitted parameters. The use of Arduino for the development of software for unmanned aircraft is feasible from the point of view of economic benefits and ease of access to technology for aircraft model enthusiasts.

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

EEPROM - Electrically Erasable Programmable Read-Only Memory (Електрично стираємо програмована пам'ять)

SRAM - static random access memory (Статична оперативна пам'ять з довільним доступом)

SPI - Serial Peripheral Interface (Послідовний периферійний інтерфейс)

ООП – об'єктно-орієнтоване програмування

Дебагер – debugger (Налагоджувач)

БПЛА – Безпілотний Літальний Апарат

Сервопривід – пристрій, що забезпечує точне керування положенням, швидкістю або прискоренням механізмів за допомогою двигуна, редуктора та датчика зворотного зв'язку.

Пін – Pin(Контакт)

Бод – одиниця швидкості символів або швидкості модуляції у символах в секунду або імпульсах в секунду. Для com-порта дорівнює 1 біту.

AVR – лінійка мікроконтролерів фірми Atmel.

CLR – Common language runtime(Середовище виконання програм сумісних з .Net Framework).

USB – Universal Serial Bus(Універсальна послідовна шина).

COM - Communication Port(Послідовний порт).

Гц – Герц. Одиниця вимірювання частоти.

Wifi – Технологія передачі даних по радіоканалу, що використовує стандарт передачі IEEE 802.11.

## ЗМІСТ

ВСТУП .....	9
Розділ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ .....	11
1.1 Поняття Безпілотного Літального Апарату.....	11
1.1.1 Основні принципи керування БПЛА .....	12
1.1.2 Принцип польоту БПЛА літакового типу з аспекту аеродинаміки..	12
1.2 Огляд технологій у сфері безпілотних літальних апаратів літакового типу.....	16
1.3 Arduino та його застосування у робототехніці.....	16
1.4 Вибір мови програмування для розробки програмного забезпечення для БПЛА .....	17
1.4.1 Мова програмування C++ .....	17
1.4.2 Фреймворк ArduinoC .....	18
1.4.3 Бібліотека Servo.h .....	19
1.4.4 Бібліотека RF24.h.....	19
1.4.5 Фреймворк .Net Framework.....	20
1.4.6 Бібліотека EEPROM.h .....	22
1.5 Інтегроване середовище розробки Microsoft Visual Studio 2022 .....	23
1.6 Інтегроване середовище розробки Arduino IDE .....	26
1.7 Використання nRF24L01 для радіозв'язку .....	28
Розділ 2. ОПИС ОБ'ЄКТУ РОЗРОБКИ.....	33
2.1 Постановка завдання.....	33
2.2 Практична цінність розробки програмного забезпечення для БПЛА ....	33
2.3 З'єднання електронних компонентів з Arduino .....	33
2.4 Передача даних з прикладної програми в Arduino .....	36
2.5 Збереження налаштувань в файловій системі комп'ютера .....	39
2.6 Прикладна програма .....	40
2.6.1 Головна Сторінка .....	40
2.6.2 Сторінка «Двигун».....	42
2.6.3 Сторінка «Налаштування сервоприводів» .....	44
2.6.4 Сторінка «Радіозв'язок».....	45

2.6.5 Сторінка «Додаткові функції» .....	47
2.6.6 Сторінка «Налаштування пульта» .....	47
2.7 Перевірка правильності введення даних .....	48
2.8 Структури, глобальні змінні .....	50
2.9 Передача даних між пультом керування та БПЛА .....	52
2.9.1 Відправлення даних .....	53
2.9.2 Отримання даних .....	54
2.10 Апаратно-програмні вимоги .....	56
2.11 Інсталятор прикладної програми .....	57
ВИСНОВКИ .....	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ... <b>Помилка! Закладку не визначено.</b>	
ДОДАТКИ .....	64

## ВСТУП

У сучасному світі безпілотні літальні апарати (БПЛА) займають важливе місце в різних сферах, від військових дій до цивільних застосувань, таких як зйомка з повітря, агрономія, пошук та рятувальні операції. Розвиток програмного забезпечення для управління БПЛА стає все більш актуальним завданням для інженерів та розробників.

Тема даної кваліфікаційної роботи полягає в розробці програмного забезпечення для БПЛА літакового типу, яке дозволяє користувачеві легко налаштувати параметри апарата та пульта керування. Особливістю цього програмного забезпечення є можливість передачі встановлених параметрів через COM-порт у мікроконтролер Arduino. Arduino спільно з модулем бездротового зв'язку nRF24L01 використовується для керування БПЛА.

Дана кваліфікаційна робота ставить за мету дослідити та реалізувати програмне забезпечення, яке дозволить ефективно керувати та налаштовувати БПЛА, розширити можливості їх використання та забезпечити безпеку та надійність управління.

Враховуючи, що дана розробка орієнтована на простих авіамоделістів, важливо забезпечити простий та зрозумілий інтерфейс користувача для налаштування параметрів БПЛА та його пульта керування. Arduino забезпечує не лише ефективне управління, але й легку інтеграцію з різноманітними компонентами та сенсорами, що робить його ідеальним вибором для даної задачі.

Для досягнення поставленої мети було проведено аналіз сучасних технологій у сфері безпілотних літальних апаратів, розроблено та реалізовано програмне забезпечення для налаштування параметрів БПЛА та його пульта керування, а також вивчено процес взаємодії між Arduino та модулем nRF24L01 для бездротового передавання даних.

Ця робота не лише спрямована на розробку програмного забезпечення для БПЛА, але й ставить за мету зробити цю технологію більш доступною та зрозумілою для широкого кола користувачів, зокрема, для любителів

авіамоделізму. Інтеграція Arduino у процес розробки дозволить знизити витрати на розробку та створення БПЛА, що сприятиме поширенню та використанню цієї технології в різних сферах.

## Розділ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

### 1.1 Поняття Безпілотного Літального Апарату

Безпілотні літальні апарати, більш відомі як дрони, представляють собою важливу технологічну інновацію, що впливає на широкий спектр сфер. БПЛА виникли як відповідь на потреби військових у виконанні завдань, які були небезпечними чи нудними для пілотів. Протягом двадцятого століття вони стали невід'ємними активами для військових формувань, а з розвитком технологій управління та зниження вартості їх використання розширилося на цивільні галузі.

Сьогодні БПЛА знаходять застосування у багатьох сферах, включаючи аерофотозйомку, контроль території, сільське господарство, моніторинг лісових пожеж, річковий моніторинг, екологічний моніторинг, поліцейське спостереження, інспекцію інфраструктури, а також у розвагах.

Існує два основних типи безпілотних літальних апаратів: літаковий тип і вертолітний тип. Кожен з цих типів має свої особливості, переваги та обмеження.

Літакові БПЛА мають схожу конструкцію з традиційними літаками, з фіксованими крилами, які забезпечують підйом і стабільність польоту. Основою польоту є ті самі принципи, що й в літака, тобто підйомна сила створюється перепадами тиску над і під крилом. Вони часто мають великі розмахи крил і ефективні аеродинамічні форми для підвищення польотної продуктивності. Найбільш популярне виконання БПЛА літакового типу — це планер з нормальною(класичною) аеродинамічною схемою. Також практикується використання не типових аеродинамічних схем, таких як: «качка», «безхвістка», «літаюче крило» .

Вертолітні БПЛА використовують ротори для підйому і стабілізації у повітрі. Основні принципи польоту збігаються з принципами польоту гелікоптера, тобто підйомна сила створюється від обертання роторів з гвинтами. Їхні конструкції дозволяють їм виконувати вертикальний зліт та



посадку, а також маневрувати у повітрі без необхідності довгих злітно-посадкових смуг. Найбільш популярне виконання БПЛА вертолітного типу — це квадрокоптери та октокоптери.

#### 1.1.1 Основні принципи керування БПЛА

Існує кілька основних принципів керування польотом БПЛА, це:

- Безпосереднє керування оператором з пульта;
- Використання системи автопілоту за заданим курсом;
- Повністю автономний політ за допомогою штучного інтелекту.

Також існує кілька основних принципів передачі сигналу і отримання інформації та зображення з бортової камери:

- Використання частоти 2.4 ГГц;
- Використання технології wi-fi 2.4 ГГц;
- Використання технології wi-fi 5 ГГц;
- Використання частоти 5.5 ГГц з аналоговою, або цифровою модуляцією.

Кожен з цих принципів мають свої переваги та недоліки. Найкраще для керування БПЛА використовувати частоту 2.4 ГГц, а для отримання зображення використовувати 5.5 ГГц з аналоговою, або цифровою модуляцією.

Кожен БПЛА обладнаний так званим «польотним контролером», або іншою системою, яка дозволяє оператору керувати безпілотником. На ринку існує безліч «польотних контролерів», в кожного існують свої переваги та недоліки. Основним недоліком більшості «польотних контролерів» це відсутність зручного і гнучкого налаштування різних параметрів, наприклад: кутів повороту сервоприводів.

#### 1.1.2 Принцип польоту БПЛА літакового типу з аспекту аеродинаміки

БПЛА літакового типу використовує ті ж самі принципи польоту, що й літак. Підймальна сила створюється від зустрічного потоку повітря, яке

взаємодіє з крилом. Крило має свій профіль, який може бути симетричним та асиметричним. При використанні симетричного профілю крила потрібно встановлювати крило з певним кутом атаки, інакше політ буде неможливий. При використанні асиметричного профілю кут атаки робити не обов'язково, але при зльоті, посадці та виконанні фігур вищого пілотажу БПЛА може отримувати певний кут атаки під час польоту.

Найпопулярніша аеродинамічна схема для БПЛА літакового типу – нормальна(класична) аеродинамічна схема, тобто стабілізатор встановлений позаду крила і є чітко відділений від крила фюзеляж. Один з найвідоміших БПЛА з нормальною(класичною) аеродинамічною схемою — український БПЛА «UJ-22 Airborne» (див. Рис. 1.1). Якщо використовується аеродинамічна схема «качка», то горизонтальний стабілізатор встановлюється попереду крила, а вертикальний стабілізатор може залишатись позаду крила. Найбільш відомий БПЛА з аеродинамічною схемою «качка» — український БПЛА «UJ-26 Бобер» (див. Рис. 1.2). Перевага класичної аеродинамічної схеми, це стабільність на високих швидкостях, перевагою аеродинамічної схеми «качка» є стабільність на низьких швидкостях.



Рис. 1.1 БПЛА «UJ-22 Airborne» [1].



Рис. 1.2 БПЛА «UJ-26 Бобер» [2].

В рух БПЛА приводиться за допомогою двигуна. Найпопулярнішими є електричні безколекторні двигуни. Менше розповсюджені мають нітродметанолові двигуни внутрішнього згорання, а також реактивні двигуни. При використанні електричного, або двигуна внутрішнього згорання встановлюється гвинт, який крутиться за допомогою крутного моменту двигуна і створює тягу. При використанні реактивного двигуна тяга створюється реактивним струменем газу.

Також існують БПЛА які не мають двигуна і розганяються іншими способами. Такі БПЛА не мають популярності, так як політ є коротким і є складність розігнати БПЛА до потрібної швидкості.

Разом з електричним двигуном використовується спеціальний контролер двигуна, який в залежності від частоти ШІМ може змінювати оберти двигуна. При використанні двигуна внутрішнього згорання потрібно використовувати сервопривід, який мінятиме позицію дросельної заслінки.

Зміна траєкторії польоту можлива за допомогою керуючих поверхонь. При класичній аеродинамічній схемі це — елерони, які відповідають за крен, рулі напрямку, які відповідають за рискання і не є обов'язковими, рулі висоти, які відповідають за вісь тангажу. Також інколи на БПЛА встановлюють закрилки, які використовуються для збільшення підйімальної сили під час зльоту та посадки і при достатній підйімальній силі не є обов'язковими.

Кожна рульова поверхня повинна бути обладнана сервоприводом(Див. Рис. 1.3). В компактних БПЛА часто використовують 1 сервопривод для елеронів, які працюють в різні сторони, а також 1 сервопривід для руля висоти(Див. Рис 1.4), де обидві частини руля висоти рухаються разом. Використання 1 сервопривода для двох елеронів разом вимагає точного встановлення тяг та точного встановлення елеронів, щоб при нормальному положення сервопривода не було крену БПЛА.



Рис 1.3 Елерон з встановленим сервоприводом.



Рис 1.4 Рулі висоти з встановленим сервоприводом.

При розробці БПЛА потрібно враховувати масу всіх компонентів, при занадто великій масі БПЛА не зможе взлетіти.

## 1.2 Огляд технологій у сфері безпілотних літальних апаратів літакового типу

У сучасному світі безпілотні літальні апарати широко використовуються в різних галузях, від медіа та розваг до наукових досліджень та комерційних застосувань. У цьому контексті важливим стає розгляд технологій, які використовуються для розробки програмного забезпечення для керування БПЛА.

На сьогоднішній день на платформі Arduino часто створюються аматорські прошивки для управління БПЛА. Ці прошивки, хоч і мають певну популярність серед аматорів і авіамоделістів, зазвичай не досягають рівня універсальності та стабільності, як професійні рішення. Проте вони можуть бути корисними для створення прототипів та проведення експериментів у вузькоспеціалізованих областях.

У той же час, Ardupilot вважається однією з найпопулярніших відкритих платформ для автономного управління БПЛА. Вона надає широкі можливості для автоматизації польоту, включаючи стабілізацію, навігацію, виконання місій та взаємодію з датчиками. Ardupilot базується на відкритому програмному забезпеченні і має активну спільноту розробників, яка постійно вдосконалює та розширює його функціональні можливості.

Крім того, існують польотні контролери з власними прошивками та можливостями, які пропонують альтернативу Ardupilot та аматорським реалізаціям на базі Arduino. Ці контролери можуть мати свої особливі функції та переваги в залежності від конкретних потреб користувача та вимог проекту.

## 1.3 Arduino та його застосування у робототехніці

Arduino - це відкрита платформа для розробки електронних пристроїв, яка включає в себе мікроконтролер та середовище розробки програмного забезпечення. Мікроконтролери, які використовуються у платформі Arduino, належать до сімейства AVR виробництва Microchip Technology. До сімейства мікроконтролерів AVR належать серії tinyAVR, megaAVR та XMEGA.

Найпопулярнішою є серія megaAVR, яка є складовою популярних моделей Arduino UNO та Arduino NANO.

Серія megaAVR має достатні технічні характеристики для більшості простих обчислень та для побудови простих комп'ютерно-інтегрованих пристроїв. В Arduino UNO та Arduino NANO використовується мікроконтролер ATMEGA 328P, котрий має такі технічні характеристики:

- Тактова частота – 16МГц;
- Вбудована пам'ять: 32 КБ флеш-пам'яті, 2 КБ оперативної пам'яті (SRAM), 1 КБ EEPROM;
- Введення/виведення: 14 цифрових входів/виходів, 6 аналогових входів;
- Інтерфейси: USB, UART, I2C, SPI.

Arduino широко використовуються в робототехніці для створення різноманітних автоматизованих систем та роботів. Мікроконтролери можуть бути застосовані у проектах з контролю руху, зчитування сенсорних даних, автоматизації завдань та багато іншого. Їх простота використання та доступність роблять Arduino популярним вибором для аматорів.

## 1.4 Вибір мови програмування для розробки програмного забезпечення для БПЛА

### 1.4.1 Мова програмування C++

C++ — універсальна мова програмування, що підтримує різноманітні парадигми, такі як об'єктно-орієнтоване програмування (ООП), процедурне програмування, узагальнене програмування та інші. Вона володіє статичною типізацією, що в деяких випадках є перевагою, а в інших може стати недоліком. C++ базується на мові програмування C і успадковує багато її особливостей.

Мова програмування C++ відома своєю потужністю і широким спектром застосувань. На її основі написано безліч бібліотек (наприклад, iostream,

math.h, windows.h) та фреймворків (.Net Framework, ArduinoC), що робить її популярним вибором для розробників у різних галузях.

C++ використовується для створення різноманітного програмного забезпечення, включаючи прикладне програмне забезпечення, прошивки для мікроконтролерів, драйвери, прошивки для промислового обладнання, відеоігри тощо. Перевагою C++ порівняно з іншими популярними мовами програмування, такими як Java, Python, C#, є більша швидкість виконання програм, кращий доступ до оперативної пам'яті та менші вимоги до ресурсів комп'ютера.

#### 1.4.2 Фреймворк ArduinoC

Фреймворк ArduinoC є інструментом для розробки програмного забезпечення для мікроконтролерів Arduino. Він забезпечує зручний та простий інтерфейс для програмування функцій керування, введення/виведення, обробки даних та взаємодії зі зовнішніми пристроями.

ArduinoC створений на основі C++ і успадкував багато переваг, але також має недоліки та обмеження. До прикладу: в C++ програма пишеться в функції `main()`, яка також є початковою точкою виконання програми, натомість в ArduinoC програма пишеться в функціях `setup()` і `loop()`, де `setup` є стартовою точкою і виконується один раз, натомість функція `loop` виконується після функції `setup` і є циклічною. Також в ArduinoC відсутні вказівники, так як мікроконтролери Arduino не містять операційної системи і оператори `new` і `delete` були виключені за відсутністю необхідності. Також добавлені методи керування часом (читання з лічильників часу), методи керування пінами (вхід/вихід), методи бездіяльності (`delay()`, `delaymicros()`), методи роботи з СОМ-портом, методи ШІМ (Широтно-імпульсна модуляція), методи обробки переривань та багато іншого.

ArduinoC як і C++ має статичну типізацію даних, що накладає певні обмеження для програміста. Також комірки EEPROM пам'яті мають розмір 1 байт і це потрібно враховувати.

ArduinoC містить в собі кілька стандартних бібліотек для роботи з датчиками, пристроями, інтерфейсами, наприклад: Servo.h, SPI.h, EEPROM.h, LiquidCrystal.h, Stepper.h та інші. Бібліотеки оголошуються подібним чином як і на C++ використовуючи `#include` і так само компілюються препроцесором. Робота з бібліотеками зазвичай пов'язана з роботою з об'єктами.

При написанні коду для платформи Arduino потрібно враховувати малу кількість оперативної пам'яті і за необхідності потрібно оптимізовувати використання оперативної пам'яті. Також при написанні коду потрібно враховувати малу кількість та ресурс EEPROM пам'яті, яка використовується для постійного зберігання даних і інформації.

#### 1.4.3 Бібліотека Servo.h

Бібліотека Servo.h – це бібліотека створена для можливості зручного керування сервоприводами, а також іншими пристроями, такими як контролери двигунів. Бібліотека містить кілька методів, які дозволяють зручно керувати сервоприводами. Вона містить в собі такі методи: `attach`, `write`, `writeMicroseconds`, `read`, `attached`, `detach`.

Ця бібліотека є достатньо оптимізована, щоб забезпечити швидкий відгук на команду і забезпечує зручне керування сервоприводами і контролерами двигунів.

#### 1.4.4 Бібліотека RF24.h

Бібліотека RF24.h - це бібліотека, яка дозволяє взаємодіяти з бездротовими модулями на базі чіпів nRF24L01 в середовищі Arduino. RF24.h надає зручні засоби для передачі даних через бездротове з'єднання між різними пристроями, що використовують модулі nRF24L01. Це дозволяє створювати мережі з кількох пристроїв, які можуть обмінюватися даними один з одним. Також бібліотека надає інтуїтивно зрозумілі класи та функції для ініціалізації модулів nRF24L01, налаштування параметрів з'єднання і передачі даних. Бібліотека дозволяє налаштовувати різні параметри



бездротового з'єднання, такі як канал передачі, потужність сигналу, швидкість передачі даних та багато іншого. Це дозволяє гнучко налаштувати з'єднання для різних ситуацій.

Однією з ключових особливостей бібліотеки RF24.h є її висока продуктивність та надійність. Модулі nRF24L01, які використовуються з цією бібліотекою, відомі своєю стабільністю та широким спектром застосувань в різних проектах з бездротовим зв'язком. Бібліотека RF24.h відповідає цій надійності, надаючи розробникам потужні інструменти для створення стабільних та ефективних бездротових мереж.

Бібліотека і контролер nRF24L01 використовують інтерфейс SPI для взаємодії з Arduino. SPI є послідовною синхронною шиною для передачі даних.

Бібліотека побудована на ООП парадигмі, тому при роботі з ній використовується клас RF24. Клас RF24 містить багато методів для керування nRF24L01. Є багато методів для конфігурації, налаштування, для керування, а також кілька методів для перевірки справної роботи модуля[3].

#### 1.4.5 Фреймворк .Net Framework

.NET Framework - це розширена платформа розробки програмного забезпечення, розроблена компанією Microsoft. .NET Framework є потужним фреймворком, який дозволяє розробляти різні прикладні програми під операційну систему Windows. .NET Framework підтримує розширення мови програмування C++, яка називається C++/CLI(Common Language Infrastructure).

.NET Framework підтримує кілька мов програмування: C#, F#, J#, VB.NET, Jscript.NET та C++/CLI.

Під час компіляції .NET Framework перетворює код в байт-код для середовища CLR(Common Language Runtime), після чого під час виконання

компілює код в машинні коди. Це забезпечує кращу сумісність програми з різними пристроями.

При роботі з C++/CLI .NET Framework потрібно використовувати керований код. Керований код є ключовою складовою розвитку програмного забезпечення в сучасному програмуванні, а концепція керованого коду розкривається на прикладі технологій, які надають середовище виконання програм, таке як .NET Framework або Mono.

У керованому коді програми виконуються на віртуальній машині Common Language Runtime (CLR), що забезпечує управління пам'яттю, виконанням, а також інші ресурси, необхідні для програмного забезпечення. Однією з ключових переваг керованого коду є його можливість перенесення між різними платформами, оскільки CLR забезпечує абстракцію від апаратного забезпечення.

Слово "керований" у цьому контексті означає, що в будь-який момент виконання програми виконавче середовище може контролювати процес виконання програми та отримувати необхідні дані, специфічні для поточного стану. Це дозволяє забезпечити відлагодження, моніторинг та керування виконанням програм.

Ця концепція дозволяє покращити надійність, безпеку та продуктивність програмного забезпечення, а також спрощує розробку та підтримку програм.

.NET Framework також підтримує створення візуального інтерфейсу за допомогою Windows Forms. Windows Forms є потужним інструментом для розробки графічного інтерфейсу користувача в рамках Microsoft .NET Framework. Використання Windows Forms спрощує процес створення і взаємодії з елементами інтерфейсу користувача в програмах для операційних систем Windows.

Інтерфейс Windows Forms надає зручний доступ до широкого спектру елементів інтерфейсу, таких як кнопки, тексти, таблиці, вікна і багато інших. Всі ці елементи можна легко розміщувати на формі і налаштовувати їх вигляд та поведінку за допомогою властивостей.

Однією з головних переваг Windows Forms є те, що вона надає можливість розробки кросплатформеного інтерфейсу. Це означає, що програми, створені з використанням Windows Forms, можуть працювати на різних версіях операційної системи Windows без необхідності змінювати код.

Проте важливо зауважити, що Windows Forms є лише обгорткою для Windows API-компонентів. Це означає, що вона базується на технологіях Win32 API і використовує їх для взаємодії з операційною системою. Це робить Windows Forms потужним і гнучким інструментом для розробки Windows-додатків, але вона обмежена платформою Windows.

#### 1.4.6 Бібліотека EEPROM.h

Бібліотека EEPROM.h є однією з ключових компонентів для роботи з пам'яттю EEPROM (Electrically Erasable Programmable Read-Only Memory) на платформі Arduino. EEPROM - це тип пам'яті, який забезпечує можливість зберігання даних навіть при відключенні живлення пристрою. Це робить його ідеальним для зберігання налаштувань, даних користувача та інших інформаційних значень, які потрібно зберегти на довготривалий термін.

Бібліотека EEPROM.h для Arduino пропонує широкий спектр функціональних можливостей, спрощуючи роботу з пам'яттю EEPROM мікроконтролера:

- Читання та запис даних: EEPROM.h забезпечує простий інтерфейс для зчитування та запису даних в пам'ять EEPROM. Завдяки цьому, користувач може легко зберігати та отримувати різноманітні дані з EEPROM за допомогою лише кількох команд;
- Використання адресації: Для доступу до конкретних регістрів пам'яті EEPROM, бібліотека використовує адресацію. Кожен регістр має унікальну адресу, яка дозволяє здійснювати операції зчитування та запису даних;
- Підтримка різних типів даних: EEPROM.h дозволяє зберігати та отримувати дані різних типів, таких як цілі числа, знакові числа,

дійсні числа та символічні рядки. Це робить її універсальним інструментом для зберігання різноманітних інформаційних значень;

- Інтеграція з іншими бібліотеками: EEPROM.h легко інтегрується з іншими бібліотеками та скетчами Arduino. Це дозволяє розширити можливості програмування мікроконтролера та забезпечує більш гнучкий підхід до розробки проектів.

### 1.5 Інтегроване середовище розробки Microsoft Visual Studio 2022

Visual Studio — інтегроване середовище розробки (IDE), створене корпорацією Microsoft. Це надійний інструмент, що використовується для розробки різноманітних програм, включаючи веб-сайти, мобільні додатки та десктопні програми. Visual Studio базується на платформах розробки програмного забезпечення Microsoft, таких як Windows API, Windows Forms, Windows Presentation Foundation (WPF), Windows Store і Microsoft Silverlight. Visual Studio може генерувати як рідний, так і керований код, що робить його універсальним інструментом для різних типів проектів.

Інтегроване середовище включає в себе редактор коду, який підтримує IntelliSense, що полегшує написання коду шляхом автозаповнення та підказок. Крім того, воно має вбудований налагоджувач, який працює як на рівні джерела, так і на рівні машини, допомагаючи розробникам виправляти помилки та оптимізувати швидкодію програм. Серед інших важливих інструментів є профайлер коду, конструктори для створення графічного інтерфейсу користувача, веб-дизайнер та інструменти для роботи з базами даних.

Однією з ключових переваг Visual Studio є його розширюваність. Завдяки плагінам розробники можуть розширити функціональність середовища, додаючи підтримку різних систем керування версіями, використання нових мов програмування та інші корисні інструменти. Visual Studio підтримує

широкий спектр мов програмування, включаючи популярні мови, такі як C#, JavaScript, Python, Ruby, PHP, C++.

Це робить Visual Studio універсальним інструментом для розробки, який може задовольнити потреби розробників з різноманітними вимогами до мов програмування та технологій розробки.

Visual Studio обладнано редактором коду, що забезпечує підсвічування синтаксису та автодоповнення коду за допомогою IntelliSense для змінних, функцій, методів, циклів та запитів LINQ. Функціонал IntelliSense доступний для включених мов, а також для XML, CSS та JavaScript під час створення веб-сайтів та веб-додатків. Пропозиції автодоповнення відображаються у немодальному полі списку над вікном редактора коду, поруч із позицією курсора. У версіях Visual Studio з 2008 року й пізніше можна зробити редактор коду тимчасово напівпрозорим, щоб легше спостерігати за кодом, який він приховує.

Редактор коду в Visual Studio (Див. Рис. 1.5) також дозволяє встановлювати закладки для швидкої навігації в коді. Інші інструменти навігації включають згортання блоків коду та інкрементний пошук, доповнюючи стандартні текстові та регулярні вирази. У ньому також є багатоеlementний буфер обміну та список завдань. Редактор підтримує фрагменти коду, які можуть бути збережені як шаблони для повторного використання та налаштовані для конкретного проекту. Додатково, він включає інструменти управління фрагментами коду, які можна налаштувати на автоматичне приховування або закріплення збоку від екрана. Редактор також підтримує рефакторинг коду, зокрема зміну порядку параметрів, перейменування змінних та методів, а також операції вилучення інтерфейсу та інкапсуляції членів класу всередині властивостей.

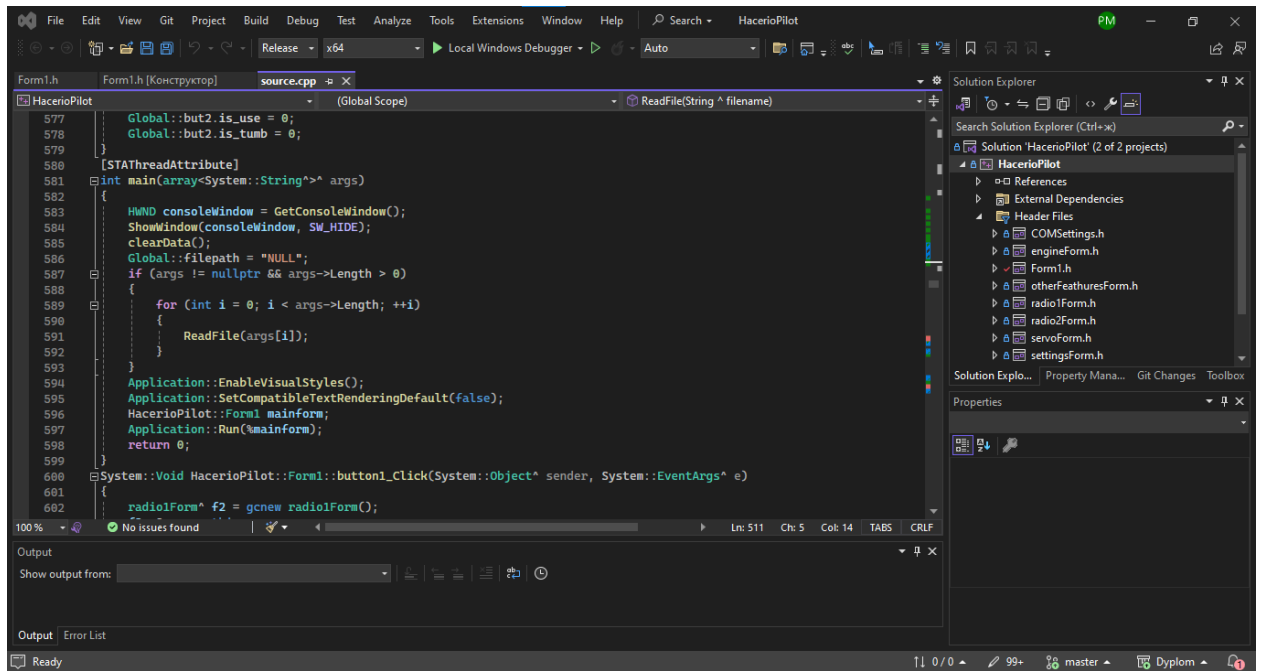


Рис.1.5. Редактор коду Visual Studio 2022

Visual Studio володіє дебагером, який працює на рівні вихідного та машинного коду. Він здатний обробляти як керований, так і некерований код, використовуватися для налагодження програм, написаних майже будь-якою мовою, яку підтримує Visual Studio. Також він може приєднуватися до запущених процесів, моніторити та відлагоджувати їх. Якщо вихідний код доступний, він відображає код під час виконання, інакше - розбирає програму. Дебагер Visual Studio має можливість створювати та завантажувати дампи пам'яті для подальшого аналізу. Підтримується також багатопотоковість програм. Його можна налаштувати на автоматичний запуск у разі аварійного завершення роботи програми поза середовищем Visual Studio. Дебагер Visual Studio дозволяє встановлювати точки зупину, які тимчасово призупиняють виконання в певних місцях коду, і спостерігати за значеннями змінних під час виконання. Точки зупину можуть бути умовними, спрацьовуючи тільки при заданій умові. Код можна виконувати по кроці, один рядок за один раз. Можливо увійти в функції для налагодження всередині них, або пропустити їх виконання, щоб не втручатися в процес вручну. Дебагер також підтримує



для мов програмування Processing та Wiring, і володіє широким спектром функціональних можливостей. Основою Arduino IDE є потужний редактор коду з рядом корисних опцій, таких як вирізання, вставлення, пошук та заміна тексту, автоматичний відступ та підсвічування синтаксису. Завдяки цьому розробники можуть швидко та зручно працювати зі своїм кодом. Програма також надає прості механізми одним кліком для компіляції та завантаження програм на плату Arduino, що значно спрощує процес розробки та тестування[4].

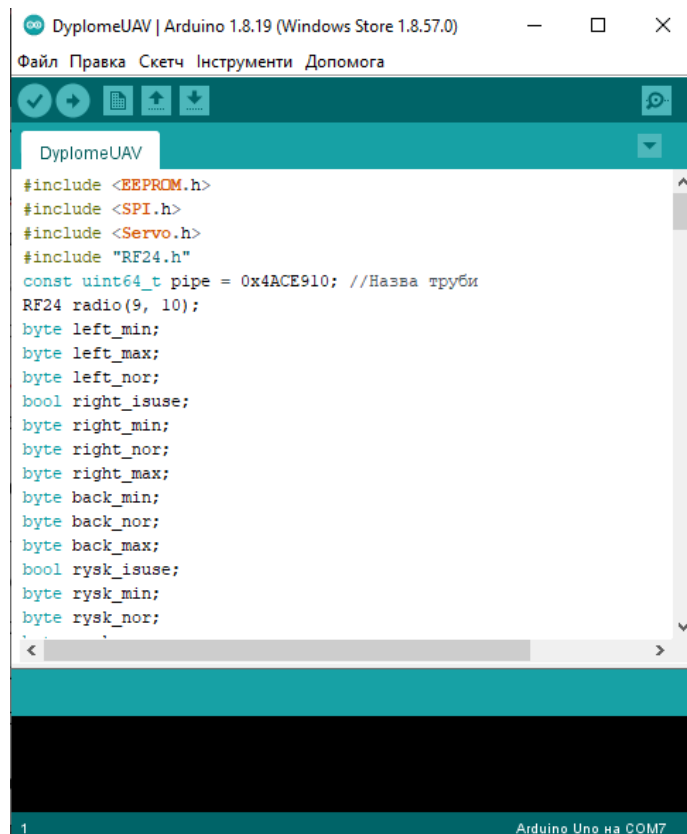


Рис 1.7. Редактор коду Arduino IDE

Окрім цього, Arduino IDE містить зручні інструменти для взаємодії з користувачем, такі як область повідомлень, текстова консоль, панель із кнопками для звичайних функцій та ієрархія операційних меню, що дозволяє зосередитися на процесі розробки без зайвих перешкод.



За допомогою Arduino IDE розробники можуть легко створювати програми на мовах C та C++, використовуючи спеціальні правила структуризації коду. Вона постачається з програмною бібліотекою проекту Wiring, що надає багато загальних процедур введення та виведення, спрощуючи процес розробки скетчів для плат Arduino.

Власне виконання коду здійснюється завдяки використанню програми avrdude, яка перетворює виконавчий код в текстовий файл у шістнадцятковому кодуванні. Цей файл потім завантажується на плату Arduino завантажувачем у вбудоване програмне забезпечення плати, що забезпечує ефективне виконання програми на мікроконтролері.

Однією з ключових можливостей Arduino IDE є легкий доступ до великого асортименту бібліотек. Завдяки цьому, розробники можуть легко використовувати готові рішення для різноманітних задач, від керування різними типами датчиків до роботи з різними пристроями і модулями.

### 1.7 Використання nRF24L01 для радіозв'язку

Модуль nRF24L01 є популярним бездротовим модулем для радіозв'язку, який здатний забезпечити стабільний зв'язок на відстані до кількох сотень метрів при відсутності перешкод. В основі його роботи лежить технологія бездротового зв'язку на базі радіочастотного протоколу.

Модуль nRF24L01 складається зі спеціалізованого радіочіпу, який забезпечує надійний та швидкий бездротовий зв'язок, та додаткових компонентів для підтримки роботи. Він працює у діапазоні частот 2.4 ГГц і використовує частотно-модульовану схему доступу до каналу (FHSS) для уникнення перешкод та інтерференції з іншими пристроями[5].

Модуль nRF24L01 володіє високою ефективністю та надійністю зв'язку на великих відстанях, що робить його ідеальним вибором для використання в системах безпілотного керування. Його використання у контексті БПЛА дозволяє забезпечити стабільний та швидкий зв'язок між пультом керування

та апаратом, що забезпечує ефективне управління пристроєм навіть на великих відстанях.

Модуль nRF24L01 підключається до Arduino за допомогою інтерфейсу SPI. Для керування ним використовується бібліотека RF24.h, яка надає зручні функції для керування ним.

Існує також версія nRF24L01+ PA + LN, яка має підсилювач (PA) та малошумний підсилювач (LNA)(Див. рис. 1.8). Ця версія модуля відрізняється вищою потужністю передачі та підвищеною чутливістю, що робить її ідеальним вибором для використання в умовах збільшених відстаней та перешкод.

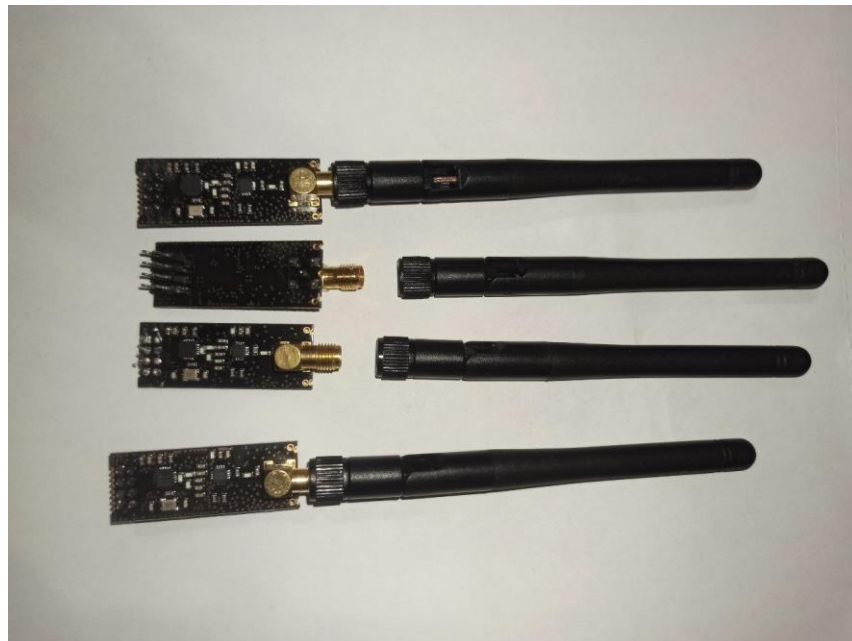


Рис 1.8. Модулі nRF24L01 + PA + LNA з зовнішніми антенами.

Основні функції модуля nRF24L01+ PA + LNA аналогічні стандартній версії nRF24L01, але завдяки підсилювачу та малошумному підсилювачу вони досягаються на більшій відстані та в умовах з великою кількістю перешкод. Модуль забезпечує стабільний та надійний зв'язок, що є критично важливим у безпілотних системах.

nRF24L01 вимагає стабільної напруги 3.3В для стабільної роботи. Стандартний стабілізатор Arduino не завжди здатний забезпечити стабільну напругу 3.3В. Наслідком перепаду напруги може бути періодична втрата сигналу, а також можливий навіть вихід з ладу nRF24L01. Для того, щоб забезпечити стабільну напругу для nRF24L01 потрібно використовувати спеціальні модулі стабілізації для nRF24L01(Див. Рис. 1.9).

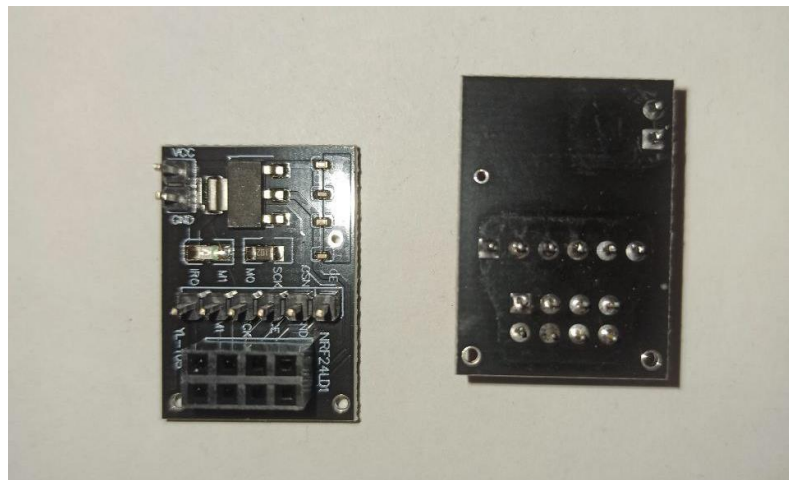


Рис. 1.9. Модулі стабілізації для nRF24L01.

Модулі стабілізації розроблені спеціально для nRF24L01 і мають спеціальний роз'єм для nRF24L01. Існують також і інші способи для стабілізації напруги для nRF24L01, але спеціальні модулі стабілізації є дешевшим і зручнішим методом стабілізації напруги(Див. Рис. 1.10).



Рис. 1.10. nRF24L01 з підключеним модулем стабілізації.

### Висновок до першого розділу

Для виконання поставленого завдання було проведено дослідження існуючих продуктів та технологій. Були розглянуті такі технології, як: мова програмування C++, фреймворки .NetFramework, ArduinoC, платформа Arduino, модуль NRF24L01+, бібліотеки Servo.h, RF24.h, EEPROM.h. В результаті проведення досліджень були обрані такі технології:

- Для розробки прикладної програми: C++, .Net Framework, Windows Forms та середовище розробки Visual Studio 2022;
- Апаратна частина: Arduino UNO або Arduino NANO, радіомодуль NRF24L01+;
- Для розробки прошивки: ArduinoC з бібліотеками: Servo.h, RF24.h, EEPROM.h.

Прикладна програма повинна забезпечувати можливість зручного налаштування БПЛА та пульта, а прошивки повинні забезпечити можливість керування та зв'язку між БПЛА та пультом.

## Розділ 2. ОПИС ОБ'ЄКТУ РОЗРОБКИ

### 2.1 Постановка завдання

Завданням кваліфікаційної роботи є створення програмного забезпечення для безпілотних літальних апаратів літакового типу. Програмне забезпечення складається з прикладної програми, яка забезпечує можливість зручного налаштування параметрів БПЛА, та з двох прошивок, які надають можливість керувати БПЛА. Для передавання даних налаштування БПЛА з комп'ютера в мікроконтролер потрібно використати СОМ-порт комп'ютера.

### 2.2 Практична цінність розробки програмного забезпечення для БПЛА

У сучасному світі з розвитком технологій та збільшенням популярності авіамоделізму зростає потреба в зручному програмному забезпеченні для безпілотних літальних апаратів. На відміну від дорогих польотних контролерів, існує доступний варіант у вигляді платформи Arduino, проте якісного та зручного програмного забезпечення для Arduino для керування БПЛА наразі немає.

З огляду на те, що ArduinoC є простим фреймворком у розробці, це надає великі можливості для розширення функціоналу БПЛА. Гнучкість платформи Arduino відкриває широкі можливості для модифікацій, включаючи встановлення різноманітних датчиків, різноманітних засобів керування БПЛА тощо.

Розроблене програмне забезпечення для БПЛА має на меті забезпечити:

- Зручний інтерфейс;
- Універсальність.

### 2.3 З'єднання електронних компонентів з Arduino

Попри універсальність, розроблене програмне забезпечення все-ж має певні вимоги стосовно підключення. Програмне забезпечення використовує пini D9, D10, D11, D12, D13 для підключення NRF24L01+ (Див. Рис. 2.1). На

ці піни користувач не зможе під'єднати сервоприводи, також прикладна програма повідомить користувача в випадку, якщо користувач спробує призначити ці піни для сервопривода(Див. Рис 2.2).

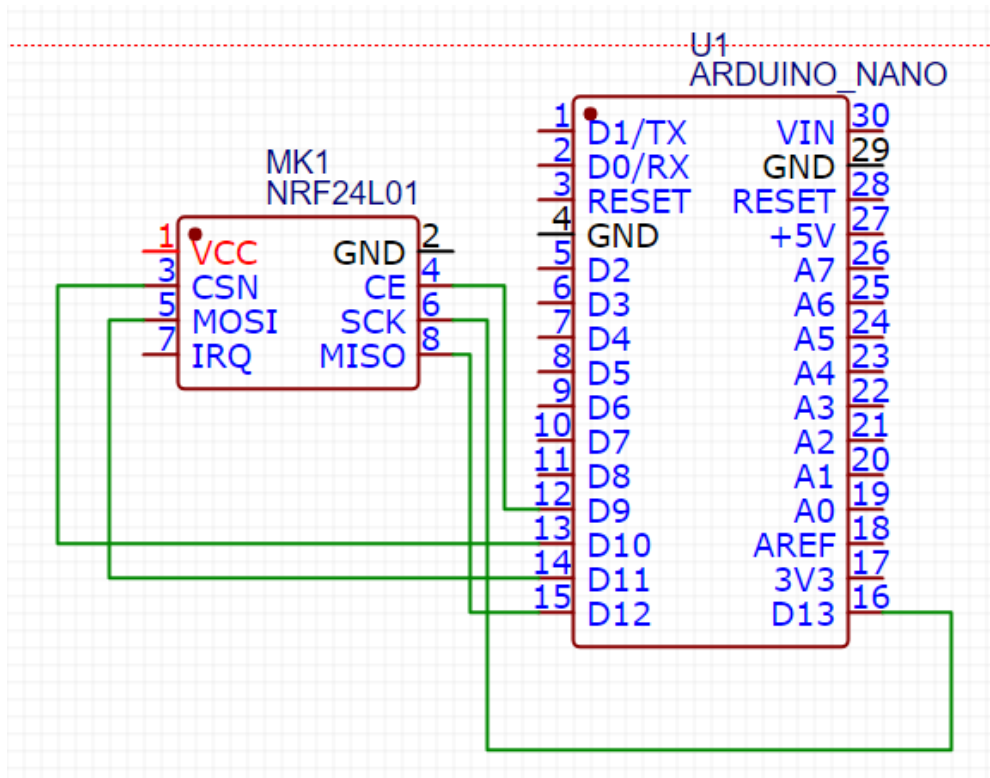


Рис 2.1 Схема підключення NRF24L01+ до ARDUINO NANO

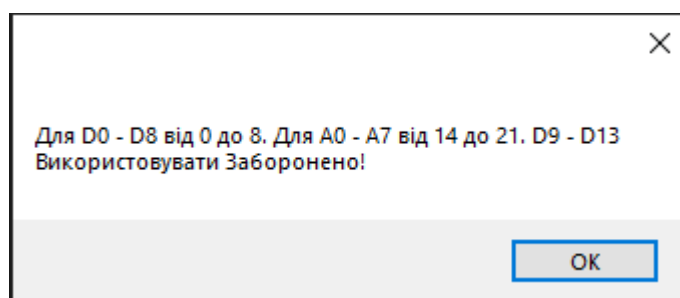


Рис 2.2 Повідомлення про спробу призначення зарезервованих пінів.

При цьому піни починаючи від D0, закінчуючи D8 та піни A0-A7 залишаються вільними і можуть бути використані користувачем для підключення до 4-х сервоприводів. При цьому користувач може в залежності від потреб використати 2, 3 або 4 сервоприводи(Див. Рис. 2.3).

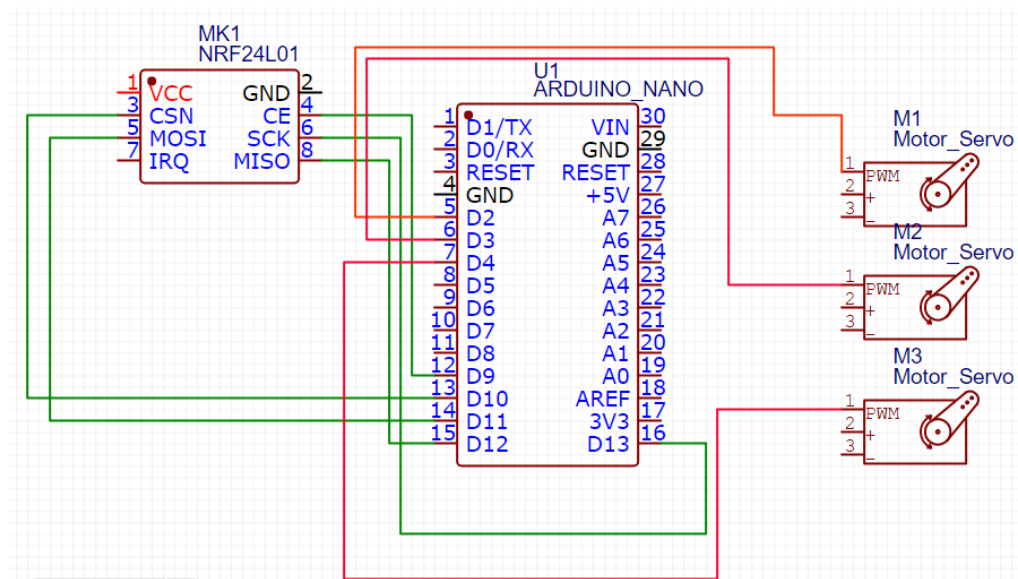


Рис 2.3 Приклад схеми підключення трьох сервоприводів

Підключення контролера двигуна, або сервопривода дросельної заслінки відбуваються аналогічним чином. Контролер двигуна(Див. Рис 2.4) так само як і сервоприводи підключаються одним керуючим дротом, на який мікроконтролер надсилає ШІМ сигнал. При цьому, більшість контролерів двигунів містять в собі DC-DC перетворювачі, що понижують напругу до 5В і цю напругу можна використати для живлення мікроконтролера. Але виробники контролерів не рекомендують підключати до нього більше ніж 2 сервоприводи, так як сервоприводи можуть викликати перепади напруги, внаслідок чого можливі втрати сигналу. Сервоприводи рекомендується живити від іншого джерела живлення, наприклад: окремий акумулятор, або окремий DC-DC перетворювач(Див. Рис. 2.5).





Рис. 2.4 Контролер керування двигуном



Рис 2.5 DC-DC перетворювач

#### 2.4 Передача даних з прикладної програми в Arduino

Платформа Arduino надає можливість передавати дані з комп'ютера до мікроконтролера, а також отримувати дані з мікроконтролера. Мікроконтролери серії AVR використовують для цього інтерфейс COM-порту. В Arduino, окрім мікроконтролера, встановлений також додатковий контролер, який виконує функції «шлюза» між інтерфейсом USB та COM-портом мікроконтролера. Це забезпечує зручне підключення Arduino до комп'ютера за допомогою USB-інтерфейсу.

При підключенні до комп'ютера, Arduino визначається в операційній системі як віртуальний COM-порт (Див. рис. 2.6). Це означає, що комп'ютер сприймає підключений мікроконтролер як стандартний COM-порт, що значно полегшує передачу даних між комп'ютером і мікроконтролером. Фреймворк

.Net Framework надає можливості для виконання операцій з COM-портами. Завдяки цьому, можливості .Net Framework можна використати для зв'язку між прикладною програмою та мікроконтролером Arduino. Це дозволяє створювати додатки, які можуть легко взаємодіяти з мікроконтролером, передавати й отримувати дані, а також виконувати необхідні операції для управління пристроєм.

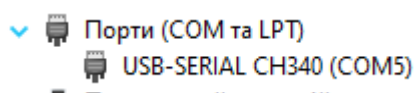


Рис. 2.6 Arduino визначається в операційній системі як COM-порт.

Буфер Arduino може одночасно зберігати 48 байт вхідних даних. Дані передаються через COM-порт як текстові дані типу char, що займають 1 байт пам'яті. Таким чином, Arduino може приймати за раз пакет даних до 48 символів. Це обмеження визначає максимальний розмір повідомлення, яке можна передати за одну операцію.

Перед відправкою даних через COM-порт необхідно синхронізувати швидкість передачі. Стандартною швидкістю передачі даних для Arduino вважається 9600 бод, що еквівалентно 9600 бітам за секунду. Ця швидкість є типовою і забезпечує надійну передачу даних для більшості проектів. Розроблене програмне забезпечення орієнтоване на універсальність і передбачає можливість модифікації прошивки для задоволення специфічних потреб користувачів, які вміють програмувати Arduino. Для того, щоб зберегти сумісність прикладної програми з можливими модифікаціями прошивки, передбачено функцію вибору швидкості передачі даних в прикладній програмі.

Ця можливість дозволяє користувачам налаштовувати швидкість передачі відповідно до вимог їхніх проектів, забезпечуючи гнучкість і зручність у використанні програми. Наприклад, користувач може змінити швидкість передачі даних на 115200 бод для більш швидкої комунікації або

залишити стандартні 9600 бод для більш надійного з'єднання. Для того, щоб зберегти сумісність прикладної програми з модифікаціями прошивки передбачена можливість вибору швидкості передачі даних в прикладній програмі(Див. рис. 2.7).

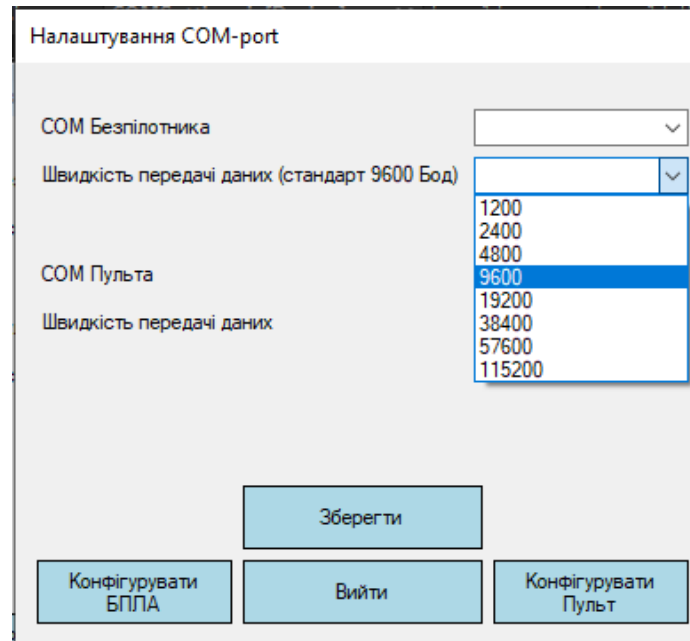


Рис 2.7 Інтерфейс вибору швидкості передачі даних

Для передачі даних за допомогою C++/CLI та .NET Framework використовуються стандартні можливості для роботи з COM-портами. Цей процес включає створення об'єктів для керування COM-портами та забезпечення надійної передачі даних між програмою та апаратним забезпеченням.

Для передачі даних у безпілотник (БПЛА) та на пульт керування були створені два об'єкти: `serialPort1` та `serialPort2`. Об'єкт `serialPort1` використовується для передачі даних до БПЛА, тоді як `serialPort2` обслуговує передачу даних до пульта керування.

Перед тим як передати дані через COM-порт, їх необхідно упакувати у масив типу `BYTE`. Це забезпечує коректне представлення даних для передачі через послідовний інтерфейс. Для відправки даних через COM-порт і

перевірки успішності цієї операції були створені дві функції: SendCOMUAV та SendCOMPULT. Ці функції не тільки здійснюють передачу даних, але й перевіряють, чи були дані успішно відправлені, і чи не виникало помилок під час передачі.

## 2.5 Збереження налаштувань в файловій системі комп'ютера

Для забезпечення зручності користування додатком потрібно забезпечити можливість збереження та читання інформації. Розроблено функціонал для збереження інформації про налаштування в файловій системі комп'ютера. З використанням Windows Forms та стандартного провідника Windows розроблено інтерфейс для зручного вибору файлу та місця для зберігання (Див. Рис. 2.8).

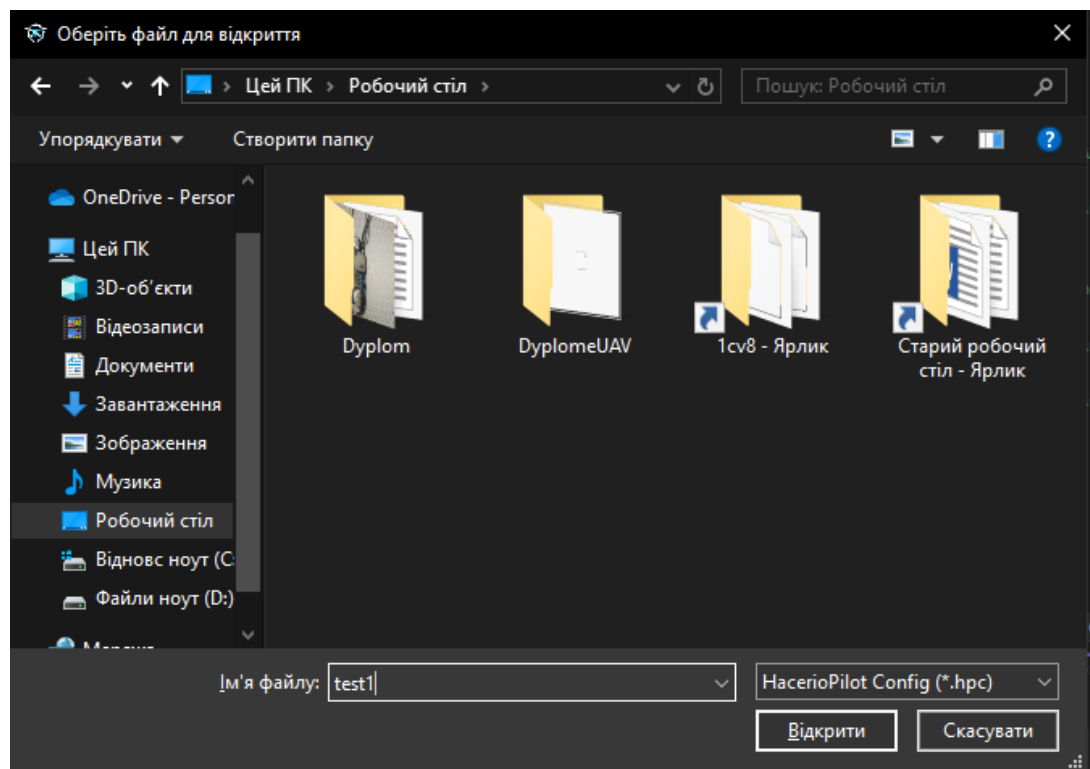


Рис 2.8 Інтерфейс стандартного провідника Windows для збереження файлів

Щоб в користувача не виникало плутанини важливо забезпечити відділення файлів створених розробленою прикладною програмою використовуючи певне розширення файлу. Для цього було обране розширення .hrc, яке зустрічається не часто серед інших додатків. Важливо при виклику стандартного провідника Windows забезпечити фільтри, щоб користувач міг швидко знайти потрібний файл з налаштуваннями серед інших файлів.

## 2.6 Прикладна програма

З використанням C++/CLI, .NET Framework та Windows Forms була розроблена прикладна програма, яка отримала назву «HacerioPilot». Ця програма забезпечує інтуїтивно зрозумілий візуальний інтерфейс для користувача, а також надає можливість збереження даних у файл і відправки даних через COM-порт на Arduino.

### 2.6.1 Головна Сторінка

Для зручності користування програмою розроблено візуальний інтерфейс за допомогою Windows Forms. Інтерфейс прикладної програми складається з кількох сторінок, що дозволяє користувачам легко орієнтуватися та використовувати всі можливості програми. При запуску програми користувача зустрічає головна сторінка (див. рис. 2.9). На цій сторінці розміщені кнопки для переходу на інші сторінки програми. Головна сторінка виконує роль навігаційного центру, звідки користувач може швидко переходити до потрібних функцій та налаштувань.

Верхня панель керування головної сторінки містить головне меню, які забезпечують доступ до різних функцій програми. Серед цих пунктів головного меню знаходяться: «Файл», «СОМ», «Довідка».

У меню «Файл» (див. рис. 2.10) користувач може знайти опції для роботи з файлами, такі як збереження, завантаження та створення нових файлів. Це дозволяє зручно зберігати конфігурації та дані.

Меню COM (див. рис. 2.11) надає можливість налаштування COM-порту для зв'язку з Arduino. Користувач може вибрати необхідний порт та налаштувати швидкість передачі даних, що забезпечує стабільну і надійну комунікацію між програмою та апаратним забезпеченням.

Меню Довідка (див. рис. 2.12) надає можливість користувачу дізнатись інформацію про програму, інструкції з використання. Це меню допомагає користувачам швидко знайти необхідну інформацію та вирішити можливі проблеми під час роботи з програмою.

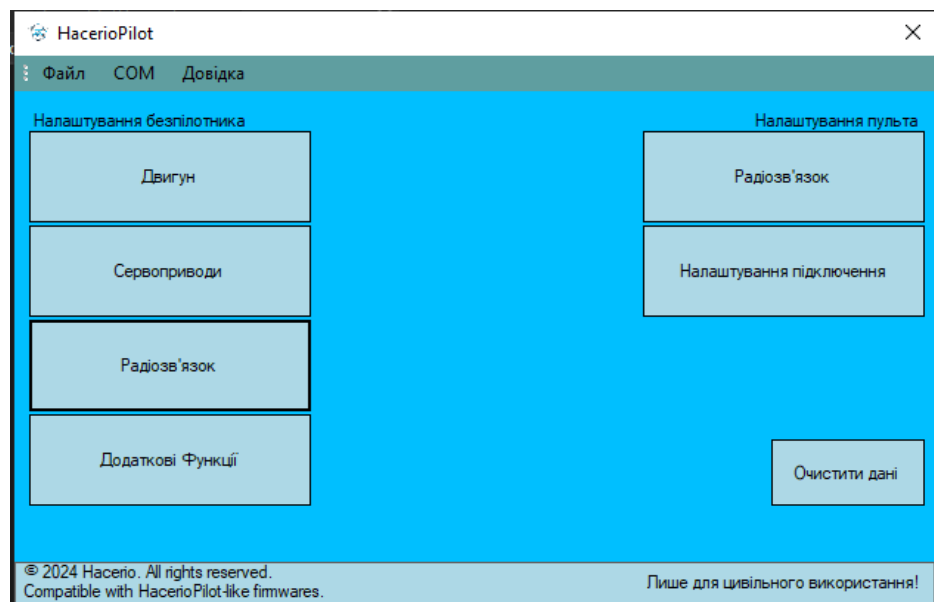


Рис 2.9 Головна сторінка додатку

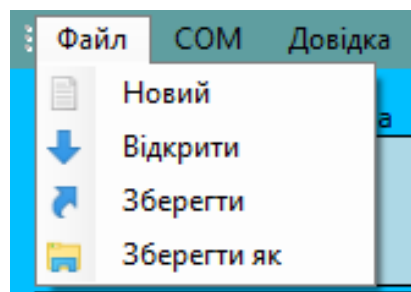


Рис 2.10 Меню Файл

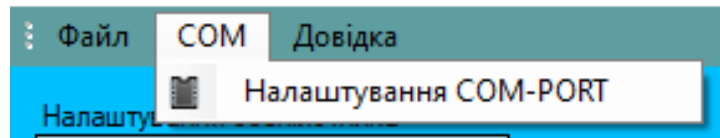


Рис 2.11 Меню COM

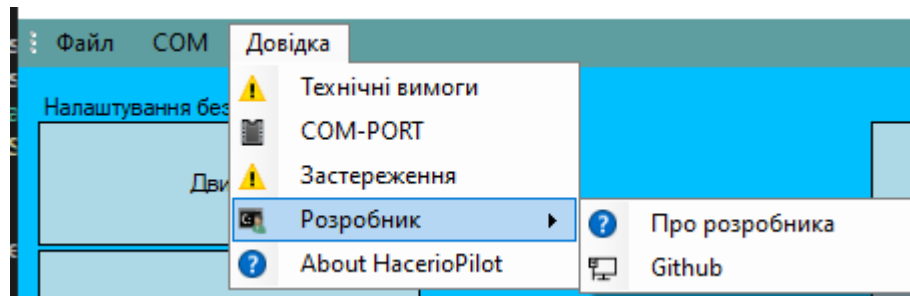


Рис. 2.12 Меню Довідка

### 2.6.2 Сторінка «Двигун»

На сторінці «Двигун»(Див. Рис. 2.13) користувач може вибрати пін, до якого підключений двигун, вибрати обмеження оборотів а також вибрати, чи двигун керується сервоприводом, чи контролером двигуна.

Рис 2.13 Сторінка «Двигун»

Після натискання кнопки «Зберегти» прикладна програма перевіряє, чи введені дані є коректні та чи не зайнятий пін і якщо дані є коректними зберігає їх. Якщо користувач вводить дані не правильно, то показується вікно з попередженням про не правильно введені дані(Див. Рис. 2.14, Рис. 2.15, Рис. 2.16, Рис. 2.17, Рис. 2.18).

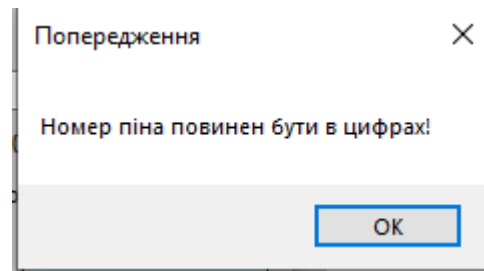


Рис 2.14 Попередження про неправильно введені вхідні дані

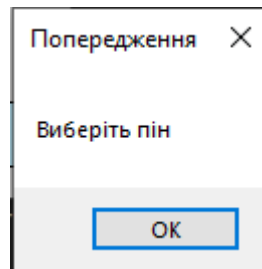


Рис 2.15 Попередження про пусте поле вхідних даних вибору піна

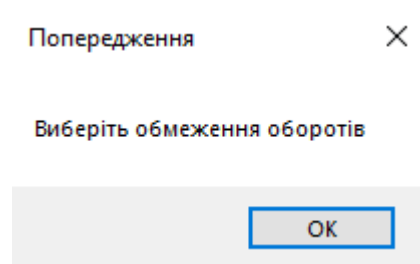


Рис 2.16 Попередження про пусте поле вхідних даних обмеження оборотів

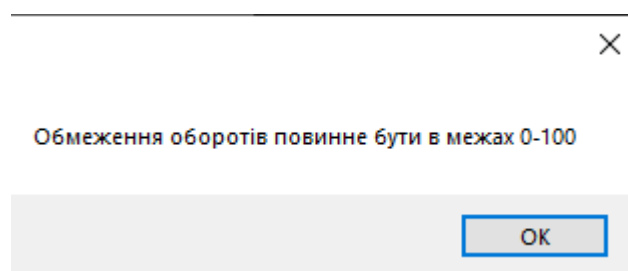


Рис 2.17 Попередження про вихід вхідних даних за обмеження



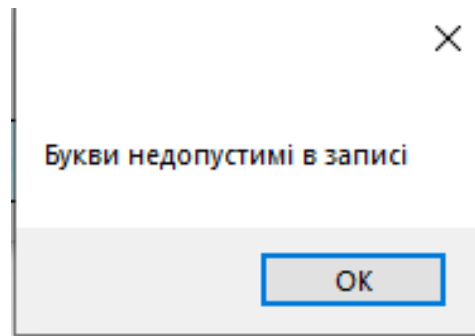


Рис 2.18 Попередження про не правильний тип вхідних даних

### 2.6.3 Сторінка «Налаштування сервоприводів»

На сторінці «Налаштування сервоприводів» (Див. Рис 2.19) користувач має можливість налаштувати кількість сервоприводів, налаштувати кінцеві та нормальні позиції сервоприводів.

 The screenshot shows a window titled "Налаштування сервоприводів" (Servo Motor Settings). It contains four sections for different servos:
 

- Лівий Елерон (Left Aileron):** Pin 2. Range: Нижня (Lower) 0, Нормальна (Normal) 90, Верхня (Upper) 180. Includes a "Тест" (Test) button.
- Правий Елерон (Right Aileron):** Pin 7. ☒ Використання (Usage). Range: Нижня 180, Нормальна 90, Верхня 0. Includes a "Тест" button.
- Руль висоти (Elevator):** Pin 3. Range: Нижня 50, Нормальна 80, Верхня 110. Includes a "Тест" button.
- Руль Напрямку (Rudder):** Pin 6. ☐ Використання. Range: Ліва (Left) 40, Нормальна 60, Права (Right) 80. Includes a "Тест" button.

 At the bottom of the window are two large buttons: "Зберегти" (Save) and "Вийти" (Exit).

Рис 2.19 Сторінка «Налаштування сервоприводів»

Після натискання кнопки «Зберегти» програма перевіряє правильність введення вхідних даних, при цьому ігнорує вхідні дані, якщо даний сервопривід є відключеним в полі перевірки «Використання», опісля програма

зберігає вхідні дані. В випадку введення не правильних вхідних даних показується вікно з попередженням(Див. Рис. 2.14, Рис. 2.15, Рис 2.18, Рис. 2.20).

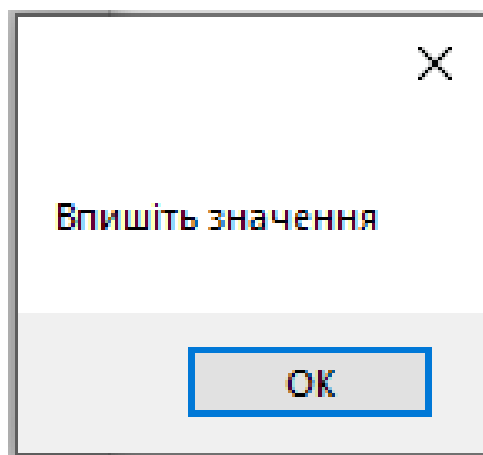


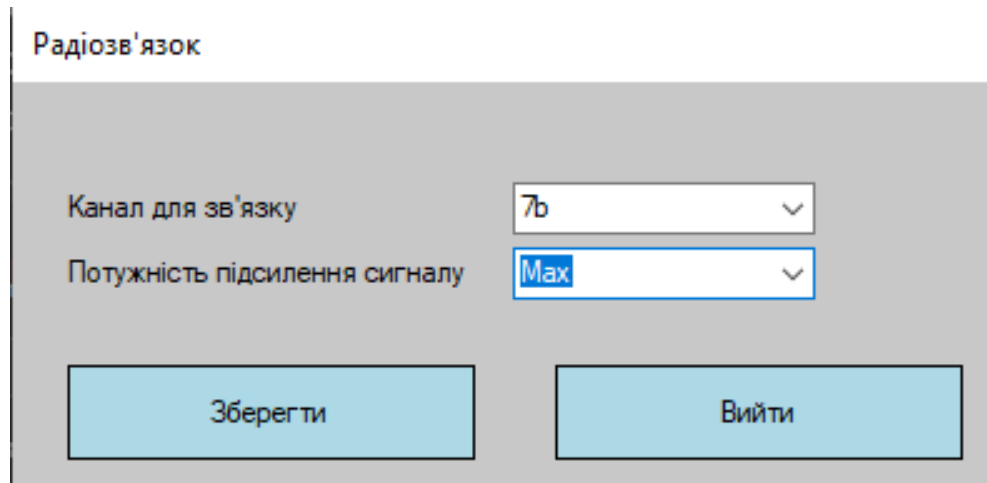
Рис 2.20 Повідомлення про пусте поле вхідних даних.

Також на сторінці розміщені кнопки «Тест», після натискання яких прикладна програма відправляє в Arduino налаштування сервоприводу. Після цього Arduino відправляє дані сервоприводу в такій послідовності: Нижня позиція(Ліва позиція) -> Верхня позиція(Права позиція) -> Нормальна позиція. Це надає змогу користувачу побачити як ці налаштування впливають на позицію сервоприводу та за потреби користувач може швидко змінити налаштування. Для того, щоб відправити дані користувач повинен спочатку на сторінці «Налаштування СОМ-порта» вибрати СОМ-порт, після чого матиме змогу використовувати функцію тестування.

#### 2.6.4 Сторінка «Радіозв'язок»

На сторінці «Радіозв'язок»(Див. Рис. 2.21) користувач має змогу налаштувати канал зв'язку та потужність передавача/приймача. nRF24L01+ має змогу оперувати 128 каналами, від 0 до 7f. Також nRF24L01+ має 4 режими керування потужність PA та LNA підсилювачів: Min, Low, High, Max.

Після натискання кнопки «Зберегти» програма перевіряє правильність введених даних, після чого зберігає їх. В випадку не правильно введених вхідних даних показується повідомлення (Див. Рис 2.22, Рис 2.23, Рис 2.24).



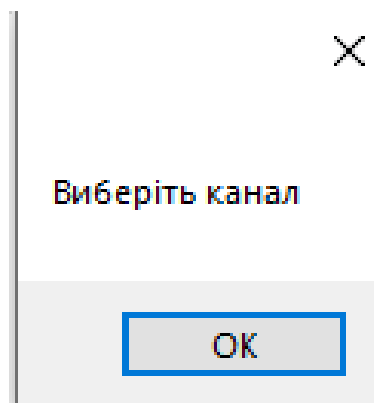
Радіозв'язок

Канал для зв'язку 7b

Потужність підсилення сигналу Max

Зберегти Вийти

Рис 2.21 Сторінка «Радіозв'язок»

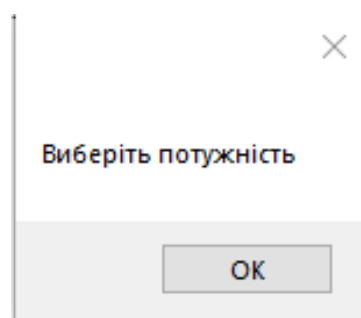


×

Виберіть канал

OK

Рис 2.22 Повідомлення про пусте поле введення



×

Виберіть потужність

OK

Рис 2.23 Повідомлення про пусте поле введення

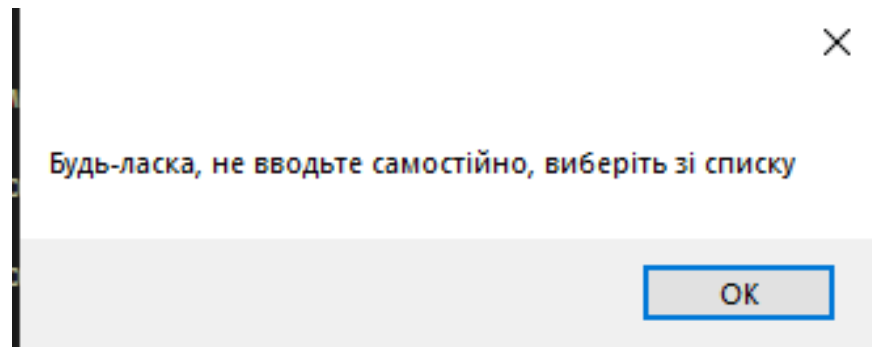


Рис 2.24 Повідомлення про не правильне введення вхідних даних

#### 2.6.5 Сторінка «Додаткові функції»

На сторінці «Додаткові функції»(Див. Рис.2.25) користувач може вибирати чи використовується додаткова функція та вибрати пін. До цього піна користувач може підключити світлові прилади, реле, або інші прилади.

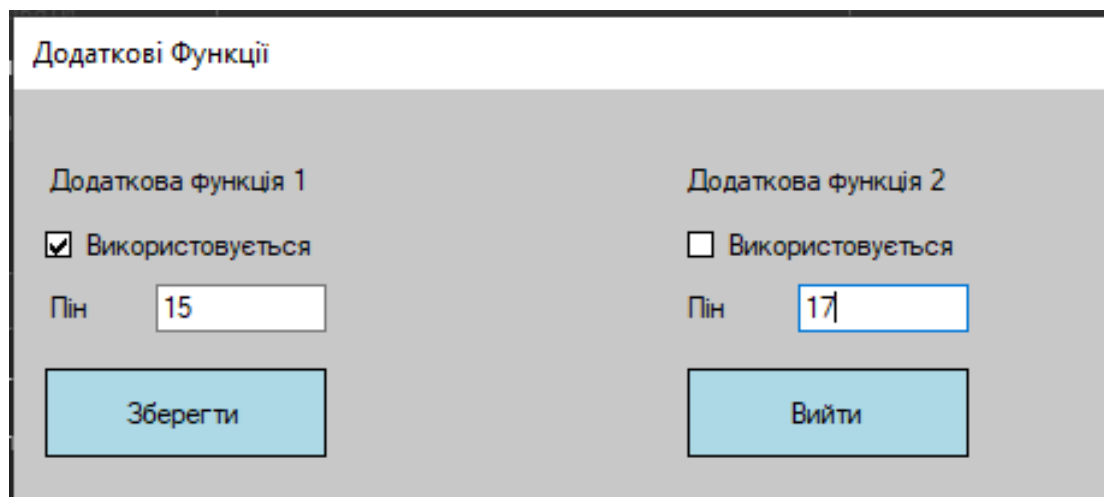


Рис. 2.25 Сторінка «Додаткові функції»

Після натискання кнопки «Зберегти» програма перевіряє правильність введення даних. При цьому, програма ігнорує дані, якщо функція не активована кнопкою «Використовується».

#### 2.6.6 Сторінка «Налаштування пульта»

Сторінка «Налаштування пульта»(Див. Рис. 2.26) надає змогу користувачу налаштувати параметри пульта керування. Користувач має змогу

налаштувати мертві зони «стіків», налаштувати тип додаткової функції. Після натискання кнопки «Зберегти» програма перевіряє правильність введення вхідних даних

Рис 2.26 Сторінка «Налаштування пульта»

## 2.7 Перевірка правильності введення даних

Для забезпечення правильного функціонування потрібно забезпечити перевірку вхідних даних. Було розроблено функції, що забезпечують перевірку вхідних даних. Перевіряються такі параметри, як:

- Перевірка наявності даних в полі введення;
- Перевірка типу даних;
- Перевірка чи входять дані в область допустимих значень;
- Перевірка чи пін не зайнятий(лише для полів вибору піна)
- Перевірка чи пін не є зарезервований(лише для полів вибору піна).

Функція `checkpin` створена для перевірки правильності вибраного піна. Функція отримує 1 аргумент типу `Byte`. Функція має тип `bool`, тому в

залежності від результату функція повертає TRUE або FALSE, де TRUE – пін вільний, FALSE – зайнятий, зарезервований або виходить за область значень.

```
bool checkpin(Byte pin) {
    if (pin == Global::left.pin || pin == Global::right.pin || pin == Global::back.pin
    || pin == Global::rysk.pin || pin == Global::other1.pin || pin ==
    Global::other2.pin || pin == Global::engine.pin)
    {
        MessageBox::Show("Пін " + pin + " зайнятий");
        return 0;
    }
    else if ((pin > 8) && (pin < 14)) {
        MessageBox::Show("Для D0 - D8 від 0 до 8. Для A0 - A7 від 14 до
        21. D9 - D13 Використовувати Заборонено!");
        return 0;
    }
    else if (pin < 0) {
        MessageBox::Show("Пін не може бути від'ємним");
        return 0;
    }
    else if (pin > 21) {
        MessageBox::Show("Пін не може бути більшим 21");
    }
    else {
        return 1;
    }
}
```

Функція перевіряє, чи аргумент не дорівнює будь якій з глобальних змінних, які відповідають за піни. Якщо пін зайнятий функція показує сповіщення «Пін ‘номер піна’ зайнятий»(Див. Рис. 2.27). Якщо вибраний пін займає область зарезервованих пінів показується повідомлення «Для D0 - D8 від 0 до 8. Для A0 - A7 від 14 до 21. D9 - D13 Використовувати Заборонено!». Якщо вибраний номер піна перевищує 21 показується повідомлення «Пін не може бути більшим 21». Якщо вибраний номер піна від'ємний показується повідомлення «Пін не може бути від'ємним».

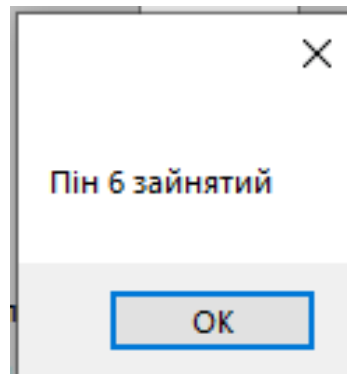


Рис 2.27 Повідомлення про зайнятий пін

Для перевірки типу введених даних була розроблена функція, яка перевіряє чи введені дані є числовими, та в залежності від результату повертає True або False. Функція приймає аргумент типу String та перевіряє кожен символ на те, чи являється він цифрою. Функція є типу bool, тому якщо вхідні дані є числом – повертається результат True, якщо вхідні дані не є числом – повертається FALSE.

```
bool IsNumeric(String^ input) {
    for each (char c in input) {
        if (!Char::IsDigit(c)) {
            return false;
        }
    }
    return true;
}
```

## 2.8 Структури, глобальні змінні

Для зручності в розробці були створені та використані структури. Ці структури мають назви: Engine(Див. Таб. 2.1), Servo(Див. Таб 2.2), Radio(Див. Таб. 2.3), Other(Див. Таб. 2.4), button(Див. Таб. 2.5). У всіх перелічених структурах змінні використовують тип даних Byte. Це забезпечує ефективне використання пам'яті та дозволяє зберігати значення в межах від 0 до 255, що є достатнім для всіх параметрів та налаштувань.

Таблиця 2.1

## Структура даних структури Engine

Назва	Тип даних
pin	Byte
limit	Byte
is_servo	Byte

Таблиця 2.2

## Структура даних структури Servo

Назва	Тип даних
is_use	Byte
pin	Byte
min	Byte
nor	Byte
max	Byte

Таблиця 2.3

## Структура даних структури Radio

Назва	Тип даних
chanel	Byte
power	Byte

Таблиця 2.4

## Структура даних структури Other

Назва	Тип даних
pin	Byte
is_use	Byte

Таблиця 2.5

## Структура даних структури button

Назва	Тип даних
is_use	Byte
is_tumb	Byte

Для передачі даних піж різними методами, необхідно використати глобальні змінні. Так як .Net Framework вимагає «керований» код потрібно створити ref-клас. Клас містить дані введені та збережені користувачем та «службові» дані(Див. Таб.2.6).



Таблиця 2.6

## Структура даних класу Global

Назва	Тип даних	Призначення
engine	Engine	Вхідні дані
left	Servo	Вхідні дані
right	Servo	Вхідні дані
back	Servo	Вхідні дані
rysk	Servo	Вхідні дані
radio1	Radio	Вхідні дані
radio2	Radio	Вхідні дані
other1	Other	Вхідні дані
other2	Other	Вхідні дані
baud1	int	Вхідні, Службові дані
baud2	int	Вхідні, Службові дані
COM1	String	Вхідні, Службові дані
COM2	String	Вхідні, Службові дані
but1	button	Вхідні дані
but2	button	Вхідні дані
is_test	byte	Службові дані
is_write	byte	Службові дані
engine_joy_is_joy	bool	Вхідні дані
engine_joy_deadzone	int	Вхідні дані
kren_joy_deadzone	int	Вхідні дані
rysk_joy_isuse	bool	Вхідні дані
rysk_joy_deadzone	int	Вхідні дані
tang_joy_deadzone	int	Вхідні дані
filepath	String	Вхідні, службові дані

## 2.9 Передача даних між пультом керування та БПЛА

Радіомодуль nRF24L01+ надає можливість передавати дані різних типів, проте в кожному пакеті даних усі елементи повинні бути одного типу. Це обмеження було враховане під час розробки системи передачі даних для забезпечення стабільної та надійної комунікації між пультом керування та БПЛА. Для забезпечення стабільного з'єднання та передачі даних оптимальним рішенням є використання масиву даних типу `byte`. Це пояснюється тим, що передача масиву даних типу `byte` забезпечує кращу узгодженість даних, мінімізує ризик помилок під час передачі та спрощує

обробку даних на обох кінцях зв'язку – як на пульті керування, так і на БПЛА. Використання масиву даних типу `byte` для передачі керуючих команд має кілька важливих переваг:

- Передача однорідних даних забезпечує стабільне з'єднання та зменшує ймовірність помилок;
- Дані типу `byte` легко обробляються як на стороні пульта керування, так і на стороні БПЛА, що спрощує розробку програмного забезпечення.
- Масиви `byte` займають менше місця, що дозволяє ефективніше використовувати пропускну здатність радіомодуля.

### 2.9.1 Відправлення даних

Для того, щоб радіомодулі безпілотного літального апарату (БПЛА) та пульта керування змогли обмінюватись інформацією, необхідно виконати певні налаштування:

- Канали передачі даних: канал передачі даних можна вибрати в прикладній програмі користувача. Важливо, щоб канали передачі даних обох радіомодулів були однаковими для забезпечення коректного обміну інформацією;
- Швидкість передачі даних: для забезпечення надійної передачі даних було обрано мінімальну швидкість передачі даних `RF24_250KBPS`. Ця швидкість дозволяє знизити ймовірність втрати даних та підвищити стабільність з'єднання;
- Ім'я радіомодулів: Ім'я для обох радіомодулів було встановлено як `0x4ACE910`, що являє собою число у шістнадцятковому форматі. Це ім'я повинно бути однаковим на обох пристроях для забезпечення коректної комунікації.

Для точного керування рульовими поверхнями та двигуном безпілота необхідно використовувати аналогові вхідні дані. Аналогово-цифровий перетворювач (АЦП), встановлений в `Arduino`, забезпечує вхідні дані в

діапазоні 0-1023. Значення 0 відповідає відсутності напруги на вході, а значення 1023 відповідає напрузі +5В. Для отримання аналогових вхідних даних потрібно використати змінні резистори, або джойстики. Змінні резистори використовуються для отримання змінного напруги, яка конвертується в аналоговий сигнал. Джойстики являють собою 2-х площинні змінні резистори з автоматичною фіксацією в середній точці при відпусканні «стіка». Вони дозволяють точно контролювати положення керуючих поверхонь та оберти двигуна.

Після отримання аналогових вхідних даних їх необхідно перетворити з діапазону 0-1023 в діапазон, що буде поміщатись в тип даних byte. Для цього обрано діапазон 0-100, оскільки він забезпечує достатню точність у керуванні та є зручнішим для програмування, ніж повний діапазон 0-255. Для перетворення діапазонів використовується стандартна функція map[6], що входить до фреймворка ArduinoC. Ця функція дозволяє масштабувати значення з одного діапазону в інший. Після перетворення даних, що включають кут положення крену, тангажу, руль напрямку, оберти двигуна та стан двох кнопок, ці дані збираються в масив типу byte (Див. Таб 2.7). Потім цей масив відправляється за допомогою модуля nRF24L01+ та бібліотеки RF24.

Таблиця 2.7

Структура даних, що передаються від пульта до бпла

Назва	Тип даних	Призначення	Діапазон значень
power	BYTE	Оберти двигуна	0-100
kren	BYTE	Керування креном	0-100
tang	BYTE	Керування тангажем	0-100
rysk	BYTE	Керування рисканням	0-100
dod1	BYTE	Значення кнопки 1	0-1
dod2	BYTE	Значення кнопки 2	0-1

### 2.9.2 Отримання даних

Бібліотека RF24.h має особливість, яка дозволяє отримувати дані з буфера радіомодуля без необхідності постійного опитування. Дані автоматично

накопичуються в буфері, тому синхронізація відправки та отримання з високою точністю не є обов'язковою.

Для отримання даних з радіомодуля потрібно використати кілька методів з бібліотеки RF24.h. Для цього було розроблено алгоритм, що включає наступні кроки:

- Перевірка наявності даних: Для перевірки наявності даних у буфері використовується метод `available()`;
- Зчитування даних: Після перевірки наявності даних, вони зчитуються методом `read()` і записуються в масив типу `BYTE`.

Після отримання даних необхідно їх обробити. Для контролера двигуна та сервоприводів дані обробляються з діапазону 0-100 в різні діапазони, котрі задає користувач під час налаштування крайніх і нормальних позицій сервоприводів. Для двигуна з контролером дані перетворюються з діапазону 0-100 в діапазон 800-2300. Якщо встановлено обмеження обертів, верхній ліміт значень обмежується під час перетворення. Для ініціалізації контролера двигуна використовується повний діапазон значень. Якщо для керування двигуном використовується сервопривід, то дані перетворюються з діапазону 0-100 в діапазон 0-180.

Для керування рульовими поверхнями дані перетворюються в два діапазони для кожного сервоприводу. Для цього була розроблена функція `SetServoPos`, яка отримує наступні вхідні дані: об'єкт сервоприводу, значення мінімального, нормального та максимального положення сервоприводу, дані про позицію в діапазоні 0-100. Ця функція забезпечує правильне встановлення положення сервоприводу відповідно до отриманих даних.

```
void SetServoPos(Servo servo, byte minimum, byte maximum, byte normal,
byte pos){
    if(pos <= 48){
        servo.write(map(pos, 0, 100, minimum, maximum));
    }
    else if(pos >= 52){
        servo.write(map(pos, 0, 100, minimum, maximum));
    }
}
```

```

else{
    servo.write(normal);
}
}

```

Функція за допомогою функції `map` перетворює дані з діапазону 0-100 у два діапазони. Якщо дані про позицію менші за 49, то функція перетворює їх у діапазон від мінімального до нормального значення. Якщо ж вхідні дані більші за 51, то вони перетворюються в діапазон від нормального до максимального значення. Це створює невелику «мертву зону» при значеннях вхідних даних від 49 до 51, що забезпечує прямолінійний політ при невеликих коливаннях вхідних даних. Створення «мертвої зони» дозволяє мінімізувати вплив незначних коливань вхідних даних на керування рульовими поверхнями, що забезпечує більш стабільний та прямолінійний політ. Ця функція забезпечує коректне керування сервоприводом на основі вхідних даних, гарантуючи стабільність і точність керування.

## 2.10 Апаратно-програмні вимоги

Розроблена прикладна програма має наступні апаратно-програмні вимоги:

- Операційна система: Windows 10 або Windows 11;
- Процесор: згідно апаратних вимог Windows 10;
- Оперативна пам'ять: згідно апаратних вимог Windows 10;
- Встановлений .Net Framework 4.7 та пакет Visual Studio 2022 SDK;
- Встановлені драйвера для Arduino, а в випадку неофіційної копії Arduino – драйвера CH340;
- Вільне місце в пам'яті: 100МБ.

В випадку використання інсталятора для встановлення прикладної програми наявність .Net Framework 4.7 та пакету Visual Studio 2022 SDK не вимагається.

Розроблена прошивка для БПЛА має наступні апаратно-програмні вимоги:

- Встановлена Arduino IDE на комп'ютері;
- Мікроконтролер: Arduino UNO або Arduino NANO або неофіційні копії;
- Радіомодуль: nRF24L01+;
- Використання від 2-х до 4-х сервоприводів в випадку підключення двигуна через контролер, від 3-х до 5-ти сервоприводів в випадку використання двигуна внутрішнього згорання.

Розроблена прошивка для пульта керування має наступні апаратно-програмні вимоги:

- Встановлена Arduino IDE на комп'ютері;
- Мікроконтролер: Arduino UNO або Arduino NANO або неофіційні копії;
- Радіомодуль nRF24L01+;
- Використання від 3-х до 4-х джерел аналогового вхідного сигналу(2 «джойстика», або 3-4 змінні резистори).

### 2.11 Інсталятор прикладної програми

Для зручності в встановленні прикладної програми за допомогою інструментів Visual Studio Installer[7] розроблений інсталятор. Інсталятор під час процесу інсталяції встановлює файли програми(Див. Рис. 2.28), додаткові файли, які необхідні для роботи програми та створює ярлики на робочому столі(Див. Рис. 2.29) та панелі «Пуск»(Див. Рис. 2.20).

Ім'я	Дата змінення	Тип	Розмір
170242232726768793.ico	10.06.2024 22:48	Icon File	68 КБ
api-ms-win-crt-heap-l1-1-0.dll	16.01.2024 22:53	Розширення заст...	19 КБ
api-ms-win-crt-locale-l1-1-0.dll	10.06.2024 22:48	Розширення заст...	19 КБ
api-ms-win-crt-math-l1-1-0.dll	10.06.2024 22:48	Розширення заст...	29 КБ
api-ms-win-crt-runtime-l1-1-0.dll	10.06.2024 22:48	Розширення заст...	23 КБ
api-ms-win-crt-stdio-l1-1-0.dll	10.06.2024 22:48	Розширення заст...	24 КБ
HacerioPilot	10.06.2024 22:48	Застосунок	806 КБ
HacerioPilot	10.06.2024 22:49	Ярлик	4 КБ
MSVCP140.dll	10.06.2024 22:48	Розширення заст...	560 КБ
VCRUNTIME140.dll	10.06.2024 22:48	Розширення заст...	117 КБ

Рис. 2.28. Файли, що встановлюють в вибрану користувачем папку



Рис. 2.29. Ярлик, що встановлюється на робочий стіл

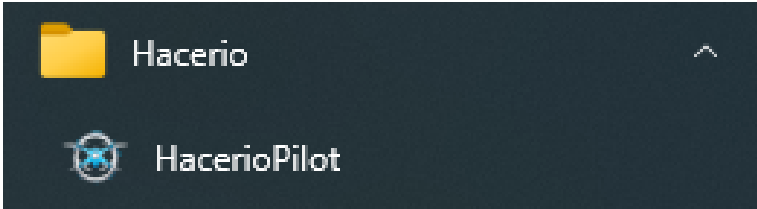


Рис 2.30. Ярлик, що встановлюється в меню «Пуск»

Після встановлення прикладної програми операційна система Windows записує її в реєстр встановлених програм. Це надає змогу деінсталювати програму через налаштування Windows(Див. Рис. 2.31). Також це надає змогу відкривати файли конфігурації за допомогою функції «відкрити за допомогою»(Див. Рис. 2.32).

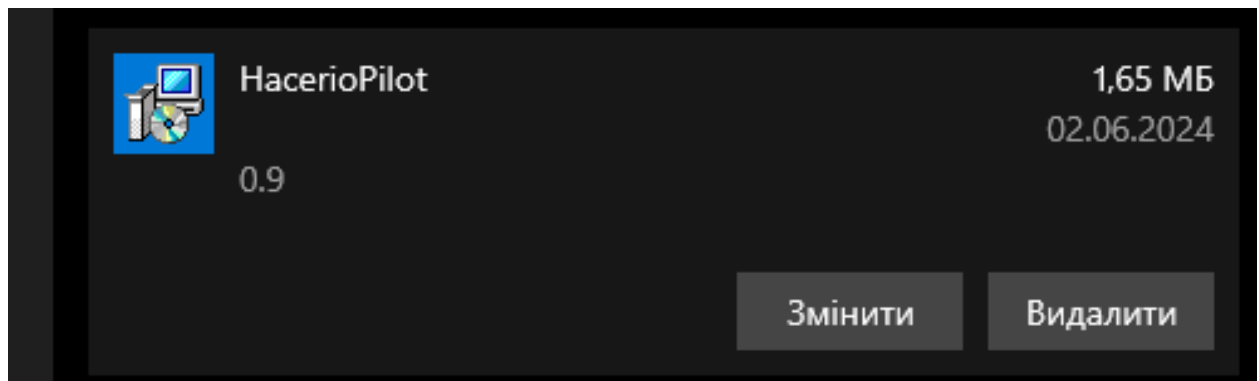


Рис. 2.31. Прикладна програма визначається операційною системою в налаштуваннях

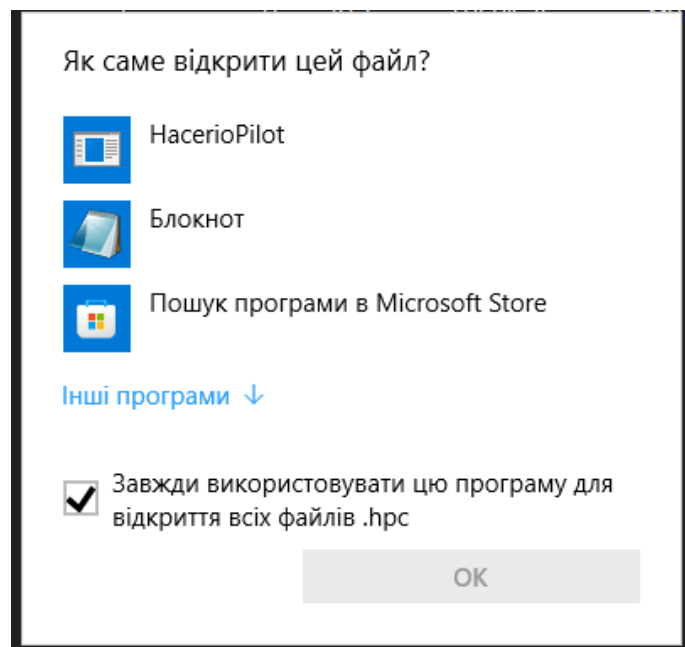


Рис. 2.32 Прикладна програма в функції «Відкрити за допомогою»



### Висновок до другого розділу

Результатом виконаної роботи є розроблена прикладна програма та розроблені прошивки для бпла та пульта керування. Для прикладної програми використано C++ та .Net Framework з Windows Forms. Прошивки розроблені за допомогою Arduino C.

## ВИСНОВКИ

Виконана кваліфікаційна робота присвячена розробці програмного забезпечення для безпілотних літальних апаратів літакового типу. З розвитком безпілотних літальних апаратів та збільшенням популярності авіамоделізму з'явилося питання щодо створення зручного та універсального програмного забезпечення для безпілотних літальних апаратів літакового типу з використанням платформи Arduino. Використання Arduino та радіомодуля nRF24L01+ для простих безпілотників або рухомих авіамоделей є доцільним з економічної точки зору.

Основною метою даного проєкту було створення універсальної системи, що дозволяє користувачеві точно налаштувати параметри БПЛА та пульта управління для досягнення максимальної ефективності безпілотного літального апарату. У результаті роботи була досягнута поставлена мета: розроблено прикладну програму та дві прошивки для мікроконтролерів Arduino — одна для безпілотного літального апарату, інша для пульта управління.

Користувач має змогу точно налаштувати кінцеві та нормальні позиції сервоприводів, налаштувати кількість сервоприводів, вибрати піни для підключення сервоприводів, налаштувати радіозв'язок, налаштувати параметри двигуна, налаштувати додаткові дискретні виходи. Також користувач має змогу налаштувати пульт керування: — Налаштувати «мертві зони» стіків; — Вибрати кількість стіків; — Налаштувати тип вхідного сигналу для керування двигуном: стік або змінний резистор; — Налаштувати тип вхідного сигналу для керування дискретними виходами: кнопка або тумблер.

Розроблена прикладна програма надає користувачу інтуїтивно зрозумілий візуальний інтерфейс. Використовуючи СОМ-порт, прикладна програма надсилає вибрані налаштування до Arduino безпілотника або пульта керування.

Розроблені прошивки орієнтовані на найпопулярніші моделі Arduino — Arduino UNO та Arduino NANO, які є дешевими та зручними у використанні.

## СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. [https://upload.wikimedia.org/wikipedia/commons/thumb/e/e2/UKRJET\\_UJ-22\\_Airborne%2C\\_Kyiv\\_2021%2C\\_07.jpg/1200px-UKRJET\\_UJ-22\\_Airborne%2C\\_Kyiv\\_2021%2C\\_07.jpg](https://upload.wikimedia.org/wikipedia/commons/thumb/e/e2/UKRJET_UJ-22_Airborne%2C_Kyiv_2021%2C_07.jpg/1200px-UKRJET_UJ-22_Airborne%2C_Kyiv_2021%2C_07.jpg)
2. [https://turdef.com/storage/images/articles/4050/v1703948044/UJ\\_26\\_turdef\\_8b4e79f78f.jpg](https://turdef.com/storage/images/articles/4050/v1703948044/UJ_26_turdef_8b4e79f78f.jpg)
3. <https://nrf24.github.io/RF24/>
4. [https://en.wikipedia.org/wiki/Arduino#Legacy\\_IDE](https://en.wikipedia.org/wiki/Arduino#Legacy_IDE)
5. <https://pdf1.alldatasheet.com/datasheet-pdf/view/1243924/ETC1/NRF24L01.html>
6. <https://www.arduino.cc/reference/en/language/functions/math/map/>
7. [https://learn.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2010/2kt85ked\(v%3dvs.100\)](https://learn.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2010/2kt85ked(v%3dvs.100))

## ДОДАТКИ

### Додаток А

#### Сирцевий код прикладної програми

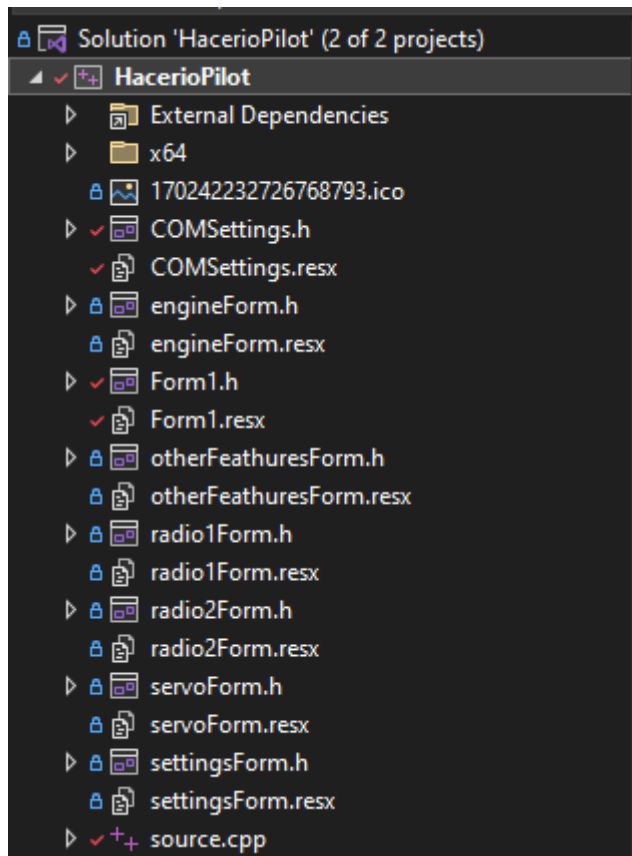


Рис 1. Структура проекту прикладної програми

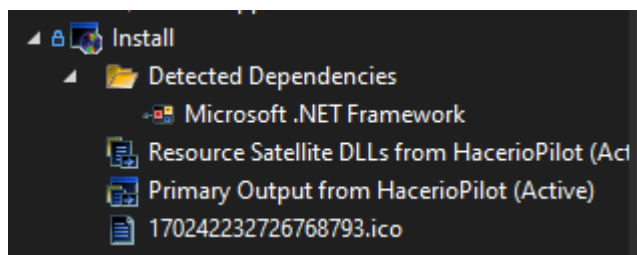


Рис 2. Структура проекту інсталятора

#### source.cpp

```
#include "Form1.h"
#include "radio1Form.h"
#include "otherFeathuresForm.h"
#include "engineForm.h"
#include "servoForm.h"
#include "radio2Form.h"
#include "settingsForm.h"
```

```

#include "COMSettings.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <vcclr.h>
#include <windows.h>
#pragma comment(lib, "User32.lib")

using namespace Microsoft::Win32;
using namespace System;
using namespace System::Windows::Forms;
using namespace System::IO;
using namespace HacerioPilot;
using namespace System::Text;
bool IsNumeric(String^ input) {
    for each (char c in input) {
        if (!Char::IsDigit(c)) {
            return false;
        }
    }
    return true;
}

Byte StringToByte(String^ str) {
    try {
        return Byte::Parse(str);
    }
    catch (FormatException^ e) {
        MessageBox::Show("Error: " + e->Message, "Error");
        return 255;
    }
}

public value struct Engine
{
    System::Byte pin;
    System::Byte limit;
    System::Byte is_servo;
};

public value struct Radio {
    System::Byte chanel;
    System::Byte power;
};

public value struct Other {
    System::Byte pin;
    System::Byte is_use;
};

public value struct Servo {
    System::Byte is_use;
    System::Byte pin;
    System::Byte min;
    System::Byte nor;
    System::Byte max;
};

public value struct button {
    System::Byte is_use;
    System::Byte is_tumb;
};

static void pinBytes(int& offset, array<Byte>^ byteArray, System::Byte value) {
    byteArray[offset++] = value;
}

public ref class Global {
public:
    static Engine engine;
    static Servo left;
    static Servo right;
    static Servo back;
    static Servo rysk;
    static Radio radio1;

```

```

static Radio radio2;
static Other other1;
static Other other2;
static String^ COM1;
static String^ COM2;
static int baud1;
static int baud2;
static button but1;
static button but2;
static System::Byte is_test;
static System::Byte is_write;
static bool engine_joy_is_joy;
static int engine_joy_deadzone;
static int kren_joy_deadzone;
static bool rysk_joy_isuse;
static int rysk_joy_deadzone;
static int tang_joy_deadzone;
static String^ filepath;
static array<Byte>^ GetAllBytesUAV() {
    array<Byte>^ byteArray = gcnew array<Byte>(sizeof(Engine) + sizeof(Servo) * 4 + sizeof(Radio) +
sizeof(Other) * 2);
    int offset = 0;
    is_test = 0;
    is_write = 1;
    pinBytes(offset, byteArray, is_write);
    pinBytes(offset, byteArray, is_test);

    pinBytes(offset, byteArray, engine.pin);
    pinBytes(offset, byteArray, engine.limit);
    pinBytes(offset, byteArray, engine.is_servo);

    pinBytes(offset, byteArray, left.pin);
    pinBytes(offset, byteArray, left.min);
    pinBytes(offset, byteArray, left.nor);
    pinBytes(offset, byteArray, left.max);

    pinBytes(offset, byteArray, right.is_use);
    pinBytes(offset, byteArray, right.pin);
    pinBytes(offset, byteArray, right.min);
    pinBytes(offset, byteArray, right.nor);
    pinBytes(offset, byteArray, right.max);

    pinBytes(offset, byteArray, back.pin);
    pinBytes(offset, byteArray, back.min);
    pinBytes(offset, byteArray, back.nor);
    pinBytes(offset, byteArray, back.max);

    pinBytes(offset, byteArray, rysk.is_use);
    pinBytes(offset, byteArray, rysk.pin);
    pinBytes(offset, byteArray, rysk.min);
    pinBytes(offset, byteArray, rysk.nor);
    pinBytes(offset, byteArray, rysk.max);

    pinBytes(offset, byteArray, radio1.chanel);
    pinBytes(offset, byteArray, radio1.power);

    pinBytes(offset, byteArray, other1.pin);
    pinBytes(offset, byteArray, other1.is_use);

    pinBytes(offset, byteArray, other2.pin);
    pinBytes(offset, byteArray, other2.is_use);

    return byteArray;
}
static array<Byte>^ GetAllBytesPult() {
    array<Byte>^ byteArray = gcnew array<Byte>(sizeof(engine_joy_is_joy) + sizeof(engine_joy_deadzone) +
sizeof(kren_joy_deadzone) + sizeof(rysk_joy_isuse) + sizeof(rysk_joy_deadzone) + sizeof(tang_joy_deadzone));
    int offset = 0;

```

```

        pinBytes(offset, byteArray, engine_joy_is_joy);
        pinBytes(offset, byteArray, engine_joy_deadzone);
        pinBytes(offset, byteArray, kren_joy_deadzone);

        pinBytes(offset, byteArray, rysk_joy_isuse);
        pinBytes(offset, byteArray, rysk_joy_deadzone);
        pinBytes(offset, byteArray, tang_joy_deadzone);

        pinBytes(offset, byteArray, radio2.chanel);
        pinBytes(offset, byteArray, radio2.power);
        return byteArray;
    }
};

bool checkpin(Byte pin) {
    if (pin == Global::left.pin || pin == Global::right.pin || pin == Global::back.pin || pin == Global::rysk.pin || pin ==
Global::other1.pin || pin == Global::other2.pin || pin == Global::engine.pin)
    {
        MessageBox::Show("Пін " + pin + " зайнятий");
        return 0;
    }
    else if ((pin > 8) && (pin < 14)) {
        MessageBox::Show("Для D0 - D8 від 0 до 8. Для A0 - A7 від 14 до 21. D9 - D13 Використовувати
Заборонено!");
        return 0;
    }
    else if (pin < 0) {
        MessageBox::Show("Пін не може бути від'ємним");
        return 0;
    }
    else if (pin > 21) {
        MessageBox::Show("Пін не може бути більшим 21");
    }
    else {
        return 1;
    }
}

void WriteToFile(String^ filename) {

    StreamWriter^ file = gcnew StreamWriter(filename, false);

    file->WriteLine("Version: 0.9");
    file->WriteLine("Engine:");
    file->WriteLine("pin: " + Global::engine.pin);
    file->WriteLine("limit: " + Global::engine.limit);
    file->WriteLine("is_servo: " + Global::engine.is_servo);

/*
    file->WriteLine("Radio1:");
    file->WriteLine("chanel: " + Global::radio1.chanel);
    file->WriteLine("power: " + Global::radio1.power);

    file->WriteLine("Radio2:");
    file->WriteLine("chanel: " + Global::radio2.chanel);
    file->WriteLine("power: " + Global::radio2.power);

*/
    file->WriteLine("Left Servo:");
    file->WriteLine("pin: " + Global::left.pin);
    file->WriteLine("min: " + Global::left.min);
    file->WriteLine("nor: " + Global::left.nor);
    file->WriteLine("max: " + Global::left.max);

    file->WriteLine("Right Servo:");
    file->WriteLine("is_use: " + Global::right.is_use);
    file->WriteLine("pin: " + Global::right.pin);
    file->WriteLine("min: " + Global::right.min);
    file->WriteLine("nor: " + Global::right.nor);
    file->WriteLine("max: " + Global::right.max);

    file->WriteLine("Back Servo:");
    file->WriteLine("pin: " + Global::back.pin);

```



```

file->WriteLine("min: " + Global::back.min);
file->WriteLine("nor: " + Global::back.nor);
file->WriteLine("max: " + Global::back.max);

file->WriteLine("Rysk Servo:");
file->WriteLine("is_use: " + Global::rysk.is_use);
file->WriteLine("pin: " + Global::rysk.pin);
file->WriteLine("min: " + Global::rysk.min);
file->WriteLine("nor: " + Global::rysk.nor);
file->WriteLine("max: " + Global::rysk.max);

file->WriteLine("Other1:");
file->WriteLine("is_use: " + Global::other1.is_use);
file->WriteLine("pin: " + Global::other1.pin);

file->WriteLine("Other2:");
file->WriteLine("is_use: " + Global::other2.is_use);
file->WriteLine("pin: " + Global::other2.pin);

file->WriteLine("Joy Engine:");
file->WriteLine("deadzone: " + Global::engine_joy_deadzone);
file->WriteLine("is_joy: " + Global::engine_joy_is_joy);

file->WriteLine("Joy Kren:");
file->WriteLine("deadzone: " + Global::kren_joy_deadzone);

file->WriteLine("Joy Rysk:");
file->WriteLine("deadzone: " + Global::rysk_joy_deadzone);
file->WriteLine("is_use: " + Global::rysk_joy_isuse);

file->WriteLine("Joy Tang:");
file->WriteLine("deadzone: " + Global::tang_joy_deadzone);

file->WriteLine("Button1:");
file->WriteLine("is_use:" + Global::but1.is_use);
file->WriteLine("is_tumb" + Global::but1.is_tumb);

file->WriteLine("Button2:");
file->WriteLine("is_use:" + Global::but2.is_use);
file->WriteLine("is_tumb" + Global::but2.is_tumb);

file->Close();
}

void ReadFile(String^ filename) {
    StreamReader^ file = gcnew StreamReader(filename);
    String^ line;
    while (file->Peek() != -1) {
        line = file->ReadLine();
        if (line == nullptr)
            break;
        array<String^>^ parts = line->Split(':');
        if (parts->Length == 2) {
            String^ key = parts[0]->Trim();
            String^ value = parts[1]->Trim();
            if (key == "Version") {
                if (value != "0.9") {
                    MessageBox::Show("Невідповідність версій");
                    return;
                }
            }
            else if (key == "Engine") {
                for (size_t i = 0; i < 5; i++)
                {
                    MessageBox::Show("1");
                    line = file->ReadLine();
                    parts = line->Split(':');
                    key = parts[0]->Trim();
                    value = parts[1]->Trim();
                }
            }
        }
    }
}

```

```

        if (key == "pin") {
            Global::engine.pin = StringToByte(value);
        }
        else if (key == "limit") {
            Global::engine.limit = StringToByte(value);
        }
        else if (key == "is_servo") {
            Global::engine.is_servo = StringToByte(value);
        }
    }
}
/*else if (key == "Radio1") {
    for (size_t i = 0; i < 2; i++)
    {
        line = file->ReadLine();
        parts = line->Split(':');
        key = parts[0]->Trim();
        value = parts[1]->Trim();
        if (key == "chanel") {
            Global::radio1.chanel = StringToByte(value);
        }
        else if (key == "power") {
            Global::radio1.power = StringToByte(value);
        }
    }
}
*/
/*else if (key == "Radio2") {
    for (size_t i = 0; i < 2; i++)
    {
        line = file->ReadLine();
        parts = line->Split(':');
        key = parts[0]->Trim();
        value = parts[1]->Trim();
        if (key == "chanel") {
            Global::radio2.chanel = StringToByte(value);
        }
        else if (key == "power") {
            Global::radio2.power = StringToByte(value);
        }
    }
}
*/
else if (key == "Left Servo") {
    for (size_t i = 0; i < 4; i++)
    {
        MessageBox::Show("2");
        line = file->ReadLine();
        parts = line->Split(':');
        key = parts[0]->Trim();
        value = parts[1]->Trim();
        if (key == "pin") {
            Global::left.pin = StringToByte(value);
        }
        else if (key == "min") {
            Global::left.min = StringToByte(value);
        }
        else if (key == "nor") {
            Global::left.nor = StringToByte(value);
        }
        else if (key == "max") {
            Global::left.max = StringToByte(value);
        }
        Global::left.is_use = 1;
    }
}
else if (key == "Right Servo") {
    for (size_t i = 0; i < 5; i++)
    {
        MessageBox::Show("3");
    }
}

```

```

        line = file->ReadLine();
        parts = line->Split(':');
        key = parts[0]->Trim();
        value = parts[1]->Trim();
        if (key == "pin") {
            Global::right.pin = StringToByte(value);
        }
        else if (key == "min") {
            Global::right.min = StringToByte(value);
        }
        else if (key == "nor") {
            Global::right.nor = StringToByte(value);
        }
        else if (key == "max") {
            Global::right.max = StringToByte(value);
        }
        else if (key == "is_use") {
            Global::right.is_use = StringToByte(value);
        }
    }
}
else if (key == "Back Servo") {
    for (size_t i = 0; i < 4; i++)
    {
        MessageBox::Show("4");
        line = file->ReadLine();
        parts = line->Split(':');
        key = parts[0]->Trim();
        value = parts[1]->Trim();
        if (key == "pin") {
            Global::back.pin = StringToByte(value);
        }
        else if (key == "min") {
            Global::back.min = StringToByte(value);
        }
        else if (key == "nor") {
            Global::back.nor = StringToByte(value);
        }
        else if (key == "max") {
            Global::back.max = StringToByte(value);
        }
        Global::back.is_use = 1;
    }
}
else if (key == "Rysk Servo") {
    for (size_t i = 0; i < 5; i++)
    {
        MessageBox::Show("5");
        line = file->ReadLine();
        parts = line->Split(':');
        key = parts[0]->Trim();
        value = parts[1]->Trim();
        if (key == "pin") {
            Global::rysk.pin = StringToByte(value);
        }
        else if (key == "min") {
            Global::rysk.min = StringToByte(value);
        }
        else if (key == "nor") {
            Global::rysk.nor = StringToByte(value);
        }
        else if (key == "max") {
            Global::rysk.max = StringToByte(value);
        }
        else if (key == "is_use") {
            Global::rysk.is_use = StringToByte(value);
        }
    }
}
}

```

```

else if (key == "Other1") {
    for (size_t i = 0; i < 2; i++)
    {
        MessageBox::Show("6");
        line = file->ReadLine();
        parts = line->Split(':');
        key = parts[0]->Trim();
        value = parts[1]->Trim();
        if (key == "is_use") {
            Global::other1.is_use = StringToByte(value);
        }
        else if (key == "pin") {
            Global::other1.pin = StringToByte(value);
        }
    }
}
else if (key == "Other2") {
    MessageBox::Show("7");
    for (size_t i = 0; i < 2; i++)
    {

        line = file->ReadLine();
        parts = line->Split(':');
        key = parts[0]->Trim();
        value = parts[1]->Trim();
        if (key == "is_use") {
            Global::other2.is_use = StringToByte(value);
        }
        else if (key == "pin") {
            Global::other2.pin = StringToByte(value);
        }
    }
}
else if (key == "Joy_Engine") {
    MessageBox::Show("8");
    for (size_t i = 0; i < 2; i++)
    {

        line = file->ReadLine();
        parts = line->Split(':');
        key = parts[0]->Trim();
        value = parts[1]->Trim();
        if (key == "deadzone") {
            Global::engine_joy_deadzone = StringToByte(value);
        }
        else if (key == "is_joy") {
            Global::engine_joy_is_joy = StringToByte(value);
        }
    }
}
else if (key == "Joy_Kren") {
    MessageBox::Show("9");
    line = file->ReadLine();
    parts = line->Split(':');
    key = parts[0]->Trim();
    value = parts[1]->Trim();
    if (key == "deadzone") {
        Global::kren_joy_deadzone = StringToByte(value);
    }
}
else if (key == "Joy_Rysk") {
    MessageBox::Show("10");
    for (size_t i = 0; i < 2; i++)
    {
        line = file->ReadLine();
        parts = line->Split(':');
        key = parts[0]->Trim();
        value = parts[1]->Trim();
        if (key == "deadzone") {

```

```

        Global::rysk_joy_deadzone = StringToByte(value);
    }
    else if (key == "is_use") {
        Global::rysk_joy_isuse = StringToByte(value);
    }
}
}
else if (key == "Joy_Tang") {
    MessageBox::Show("11");
    line = file->ReadLine();
    parts = line->Split(':');
    key = parts[0]->Trim();
    value = parts[1]->Trim();
    if (key == "deadzone") {
        Global::tang_joy_deadzone = StringToByte(value);
    }
}
else if (key == "Button1") {
    MessageBox::Show("12");
    for (size_t i = 0; i < 2; i++)
    {
        line = file->ReadLine();
        parts = line->Split(':');
        key = parts[0]->Trim();
        value = parts[1]->Trim();
        if (key == "is_use") {
            Global::but1.is_use = StringToByte(value);
        }
        else if (key == "is_tumb") {
            Global::but1.is_tumb = StringToByte(value);
        }
    }
}
else if (key == "Button2") {
    MessageBox::Show("13");
    for (size_t i = 0; i < 2; i++)
    {
        line = file->ReadLine();
        parts = line->Split(':');
        key = parts[0]->Trim();
        value = parts[1]->Trim();
        if (key == "is_use") {
            Global::but2.is_use = StringToByte(value);
        }
        else if (key == "is_tumb") {
            Global::but2.is_tumb = StringToByte(value);
        }
    }
}
}
}
file->Close();
Global::filepath = filename;
}
void clearData() {
    Global::engine.pin = 255;
    Global::engine.is_servo = 0;
    Global::engine.limit = 255;
    Global::left.pin = 255;
    Global::right.pin = 255;
    Global::back.pin = 255;
    Global::rysk.pin = 255;
    Global::radio1.chanel = 255;
    Global::radio1.power = 255;
    Global::radio2.chanel = 255;
    Global::radio2.power = 255;
    Global::other1.is_use = 0;
    Global::other2.is_use = 0;
    Global::other1.pin = 255;

```

```

Global::other2.pin = 255;
Global::COM1 = "255";
Global::COM2 = "255";
Global::baud1 = 255;
Global::baud2 = 255;
Global::left.min = 255;
Global::left.nor = 255;
Global::left.max = 255;
Global::left.is_use = 1;
Global::back.min = 255;
Global::back.nor = 255;
Global::back.max = 255;
Global::back.is_use = 1;
Global::right.is_use = 0;
Global::right.min = 255;
Global::right.nor = 255;
Global::right.max = 255;
Global::rysk.is_use = 0;
Global::rysk.min = 255;
Global::rysk.nor = 255;
Global::rysk.max = 255;
Global::engine_joy_is_joy = 0;
Global::engine_joy_deadzone = 255;
Global::tang_joy_deadzone = 255;
Global::rysk_joy_isuse = 0;
Global::rysk_joy_deadzone = 255;
Global::kren_joy_deadzone = 255;
Global::but1.is_use = 0;
Global::but1.is_tumb = 0;
Global::but2.is_use = 0;
Global::but2.is_tumb = 0;
}
[STAThreadAttribute]
int main(array<System::String^>^ args)
{
    HWND consoleWindow = GetConsoleWindow();
    ShowWindow(consoleWindow, SW_HIDE);
    clearData();
    Global::filepath = "NULL";
    if (args != nullptr && args->Length > 0)
    {
        for (int i = 0; i < args->Length; ++i)
        {
            ReadFile(args[i]);
        }
    }
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    HacerioPilot::Form1 mainform;
    Application::Run(%mainform);
    return 0;
}
System::Void HacerioPilot::Form1::button1_Click(System::Object^ sender, System::EventArgs^ e)
{
    radio1Form^ f2 = gcnew radio1Form();
    f2->Owner = this;
    f2->Show();
    Form1::Hide();
}
System::Void HacerioPilot::Form1::button4_Click(System::Object^ sender, System::EventArgs^ e)
{
    otherFeathuresForm^ f2 = gcnew otherFeathuresForm();
    f2->Owner = this;
    f2->Show();
    Form1::Hide();
}
System::Void HacerioPilot::Form1::button2_Click(System::Object^ sender, System::EventArgs^ e)
{
    engineForm^ f2 = gcnew engineForm();

```

```

        f2->Owner = this;
        f2->Show();
        Form1::Hide();
    }
    System::Void HacerioPilot::Form1::button3_Click(System::Object^ sender, System::EventArgs^ e)
    {
        servoForm^ f2 = gcnew servoForm();
        f2->Owner = this;
        f2->Show();
        Form1::Hide();
    }
    System::Void HacerioPilot::Form1::відкритиToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e)
    {
        OpenFileDialog^ openFileDialog1 = gcnew OpenFileDialog();

        openFileDialog1->Title = "Оберіть файл для відкриття";
        openFileDialog1->InitialDirectory = "C:\\";
        openFileDialog1->Filter = "HacerioPilot Config (*.hpc)|*.hpc|Всі файли (*.*)|*.*";

        if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK) {
            String^ filePath = openFileDialog1->FileName;
            MessageBox::Show("Обраний файл: " + filePath, "Інформація");
            ReadFile(filePath);
            label5->Text = Global::filepath;
            label5->Show();
            label4->Show();
        }
    }
    System::Void HacerioPilot::Form1::зберегтиЯкToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e)
    {
        OpenFileDialog^ openFileDialog1 = gcnew OpenFileDialog();

        openFileDialog1->Title = "Оберіть файл для відкриття";
        openFileDialog1->InitialDirectory = "C:\\";
        openFileDialog1->Filter = "HacerioPilot Config (*.hpc)|*.hpc|Всі файли (*.*)|*.*";

        if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK) {
            String^ filePath = openFileDialog1->FileName;
            WriteToFile(filePath);
            Global::filepath = filePath;
            label5->Text = Global::filepath;
            label5->Show();
            label4->Show();
        }
    }
    System::Void HacerioPilot::engineForm::button1_Click(System::Object^ sender, System::EventArgs^ e) {
        if ((String::IsNullOrEmpty(textBox4->Text)) && Global::engine.pin == 255) {
            MessageBox::Show("Виберіть пін", "Попередження");
        }
        else if (String::IsNullOrEmpty(textBox4->Text));
        else {
            if (!IsNumeric(textBox4->Text)) {
                MessageBox::Show("Номер піна повинен бути в цифрах!", "Попередження");
            }
            else if (Convert::ToInt32(textBox4->Text) < 0 || Convert::ToInt32(textBox4->Text) > 21) {
                MessageBox::Show("Значення повинні бути в межах 0-21 за виключенням 9-13");
            }
            else if ((Convert::ToInt32(textBox4->Text) >= 9) && (Convert::ToInt32(textBox4->Text) <= 13)) {
                MessageBox::Show("Піни D9-D13 використовувати заборонено!");
            }
            else {
                Byte pin = StringToByte(textBox4->Text);
                if (checkpin(pin)) {
                    Global::engine.pin = pin;
                }
            }
        }
        if ((String::IsNullOrEmpty(textBox3->Text)) && (Global::engine.limit == 255)) {

```

```

        MessageBox::Show("Виберіть обмеження оборотів", "Попередження");
    }
    else if (String::IsNullOrEmpty(textBox3->Text));
    else {
        if (!IsNumeric(textBox3->Text)) {
            MessageBox::Show("Букви недопустимі в записі");
        }
        else {
            Byte limit = StringToByte(textBox3->Text);
            if (limit < 0 || limit > 100) {
                MessageBox::Show("Обмеження оборотів повинне бути в межах 0-100");
            }
            else {
                Global::engine.limit = limit;
            }
        }
    }
}

}

System::Void HacerioPilot::engineForm::checkBox1_CheckedChanged(System::Object^ sender, System::EventArgs^ e)
{
    Global::engine.is_servo = checkBox1->Checked;
}

System::Void HacerioPilot::Form1::button6_Click(System::Object^ sender, System::EventArgs^ e)
{
    radio2Form^ f2 = gcnew radio2Form();
    f2->Owner = this;
    f2->Show();
    Form1::Hide();
}

System::Void HacerioPilot::Form1::button7_Click(System::Object^ sender, System::EventArgs^ e)
{
    settingsForm^ f2 = gcnew settingsForm();
    f2->Owner = this;
    f2->Show();
    Form1::Hide();
}

System::Void HacerioPilot::Form1::новийToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e)
{
    OpenFileDialog^ openFileDialog = gcnew OpenFileDialog();
    openFileDialog->Title = "Оберіть папку та назву файлу";
    openFileDialog->InitialDirectory = "C:\\";
    openFileDialog->Filter = "HacerioPilot Config (*.hpc)|*.hpc|Всі файли (*.*)|*. *";
    openFileDialog->FilterIndex = 1;
    openFileDialog->RestoreDirectory = true;
    openFileDialog->CheckFileExists = false;
    openFileDialog->CheckPathExists = true;
    openFileDialog->FileName = "NewFile";

    if (openFileDialog->ShowDialog() == System::Windows::Forms::DialogResult::OK) {
        String^ filePath = Path::GetDirectoryName(openFileDialog->FileName);
        String^ fileName = Path::GetFileNameWithoutExtension(openFileDialog->FileName);
        String^ fileFullPath = Path::Combine(filePath, fileName + ".hpc");
        if (File::Exists(fileFullPath)) {
            MessageBox::Show("Файл з такою назвою вже існує.");
            return;
        }
        FileStream^ fs = File::Create(fileFullPath);
        fs->Close();
        clearData();
        Global::filepath = fileFullPath;
        label5->Text = Global::filepath;
        label5->Show();
        label4->Show();
    }
}

System::Void HacerioPilot::Form1::налаштуванняCOMPORTToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e)
{

```



```

        COMSettings^ f2 = gcnew COMSettings();
        f2->Owner = this;
        this->Hide();
        f2->Show();
    }
    System::Void HacerioPilot::engineForm::engineForm_Load(System::Object^ sender, System::EventArgs^ e)
    {
        if (Global::engine.pin != 255) {
            engineForm::label4->Show();
            engineForm::label4->Text = Convert::ToString(Global::engine.pin);
        }
        else {
            engineForm::label4->Hide();
        }
        if (Global::engine.limit != 255) {
            engineForm::label6->Show();
            engineForm::label6->Text = Convert::ToString(Global::engine.limit);
        }
        else {
            engineForm::label6->Hide();
        }
        checkBox1->Checked = Global::engine.is_servo;
    }
    System::Void HacerioPilot::radio1Form::radio1Form_Load(System::Object^ sender, System::EventArgs^ e)
    {
        if (Global::radio1.chanel == 255) {
            radio1Form::label3->Hide();
        }
        else {
            radio1Form::label3->Show();
            radio1Form::label3->Text = Convert::ToString(Global::radio1.chanel, 16);
        }
        comboBox1->Items->Clear();
        for (int i = 0; i < 128; i++)
        {
            comboBox1->Items->Add(Convert::ToString(i, 16));
        }
        comboBox2->Items->Clear();
        comboBox2->Items->Add("Min");
        comboBox2->Items->Add("Low");
        comboBox2->Items->Add("High");
        comboBox2->Items->Add("Max");
        if (Global::radio1.power == 255) {
            radio1Form::label4->Hide();
        }
        else {
            radio1Form::label4->Show();
            if (Global::radio1.power == 0) {
                radio1Form::label4->Text = "Min";
            }
            else if (Global::radio1.power == 1) {
                radio1Form::label4->Text = "Low";
            }
            else if (Global::radio1.power == 2){
                radio1Form::label4->Text = "High";
            }
            else {
                radio1Form::label4->Text = "Max";
            }
        }
    }
}
    System::Void HacerioPilot::radio1Form::button1_Click(System::Object^ sender, System::EventArgs^ e)
    {
        String^ chanel;
        int promiz;
        if ((String::IsNullOrEmpty(comboBox1->Text)) && (Global::radio1.chanel == 255)) {
            MessageBox::Show("Виберіть канал");
        }
        else if (String::IsNullOrEmpty(comboBox1->Text)) {}
    }

```

```

else {
    promiz = Convert::ToInt32(comboBox1->Text, 16);
    chanel = Convert::ToString(promiz);
    if (promiz < 0 || promiz > 127) {
        MessageBox::Show("Будь-ласка, не вводьте самостійно, виберіть зі списку");
    }
    else {
        Global::radio1.chanel = Convert::ToByte(chanel);
    }
}
String^ power;
power = comboBox2->Text;
if ((String::IsNullOrWhiteSpace(power)) && (Global::radio1.power == 255)) {
    MessageBox::Show("Виберіть потужність");
}
else if (String::IsNullOrWhiteSpace(power)) {}
else {
    if (power == "Min") {
        Global::radio1.power = 0;
    }
    else if (power == "Low") {
        Global::radio1.power = 1;
    }
    else if (power == "High") {
        Global::radio1.power = 2;
    }
    else if (power == "Max") {
        Global::radio1.power = 3;
    }
    else {
        MessageBox::Show("Будь-ласка, не вводьте самостійно, виберіть зі списку");
    }
}
}

System::Void HacerioPilot::otherFeathuresForm::button1_Click(System::Object^ sender, System::EventArgs^ e)
{
    if (checkBox1->Checked == 0) {
        Global::other1.is_use = checkBox1->Checked;
    }
    else if ((checkBox1->Checked == 1) && (String::IsNullOrEmpty(textBox1->Text)) && (Global::other1.pin == 255)) {
        MessageBox::Show("Виберіть пін");
    }
    else if (String::IsNullOrEmpty(textBox1->Text)) {}
    else if (!IsNumeric(textBox1->Text)) {
        MessageBox::Show("Букви недопустимі в записі");
    }
    else {
        if (checkpin(Convert::ToByte(textBox1->Text))) {
            Global::other1.pin = Convert::ToByte(textBox1->Text);
        }
    }
}

if (checkBox2->Checked == 0) {
    Global::other2.is_use = checkBox2->Checked;
}
else if ((checkBox2->Checked == 1) && (String::IsNullOrEmpty(textBox2->Text)) && (Global::other2.pin == 255)) {
    MessageBox::Show("Виберіть пін");
}
else if (String::IsNullOrEmpty(textBox2->Text)) {}
else if (!IsNumeric(textBox2->Text)) {
    MessageBox::Show("Букви недопустимі в записі");
}
else {
    if (checkpin(Convert::ToByte(textBox2->Text))) {

```

```

        Global::other2.pin = Convert::ToByte(textBox2->Text);
    }
}
Global::other2.is_use = checkBox2->Checked;
Global::other1.is_use = checkBox1->Checked;
}
System::Void HacerioPilot::otherFeathuresForm::otherFeathuresForm_Load(System::Object^ sender, System::EventArgs^ e)
{
    label5->Hide();
    if (Global::other1.pin != 255) {
        label5->Text = Convert::ToString(Global::other1.pin);
        label5->Show();
    }
    checkBox1->Checked = Global::other1.is_use;
    label6->Hide();
    if (Global::other2.pin != 255) {
        label6->Text = Convert::ToString(Global::other2.pin);
        label6->Show();
    }
    checkBox2->Checked = Global::other2.is_use;
}

System::Void HacerioPilot::radio2Form::button1_Click(System::Object^ sender, System::EventArgs^ e)
{
    String^ chanel;
    int promiz;
    if ((String::IsNullOrWhiteSpace(comboBox1->Text)) && (Global::radio2.chanel == 255)) {
        MessageBox::Show("Виберіть канал");
    }
    else if (String::IsNullOrWhiteSpace(comboBox1->Text)) {}
    else {
        promiz = Convert::ToInt32(comboBox1->Text, 16);
        chanel = Convert::ToString(promiz);
        if (promiz < 0 || promiz > 127) {
            MessageBox::Show("Будь-ласка, не вводьте самостійно, виберіть зі списку");
        }
        else {
            Global::radio2.chanel = Convert::ToByte(chanel);
        }
    }

    String^ power;
    power = comboBox2->Text;
    if ((String::IsNullOrWhiteSpace(power)) && (Global::radio2.power == 255)) {
        MessageBox::Show("Виберіть потужність");
    }
    else if (String::IsNullOrWhiteSpace(power)) {}
    else {
        if (power == "Min") {
            Global::radio2.power = 0;
        }
        else if (power == "Low") {
            Global::radio2.power = 1;
        }
        else if (power == "High") {
            Global::radio2.power = 2;
        }
        else if (power == "Max") {
            Global::radio2.power = 3;
        }
        else {
            MessageBox::Show("Будь-ласка, не вводьте самостійно, виберіть зі списку");
        }
    }
}

System::Void HacerioPilot::radio2Form::radio2Form_Load(System::Object^ sender, System::EventArgs^ e)
{
    if (Global::radio2.chanel == 255) {
        radio2Form::label3->Hide();
    }
}

```

```

    }
    else {
        radio2Form::label3->Show();
        radio2Form::label3->Text = Convert::ToString(Global::radio2.chanel, 16);
    }
    comboBox1->Items->Clear();
    for (int i = 0; i < 128; i++)
    {
        comboBox1->Items->Add(Convert::ToString(i, 16));
    }
    comboBox2->Items->Clear();
    comboBox2->Items->Add("Min");
    comboBox2->Items->Add("Low");
    comboBox2->Items->Add("High");
    comboBox2->Items->Add("Max");
    if (Global::radio2.power == 255) {
        radio2Form::label4->Hide();
    }
    else {
        radio2Form::label4->Show();
        if (Global::radio2.power == 0) {
            radio2Form::label4->Text = "Min";
        }
        else if (Global::radio2.power == 1) {
            radio2Form::label4->Text = "Low";
        }
        else if (Global::radio2.power == 2) {
            radio2Form::label4->Text = "High";
        }
        else {
            radio2Form::label4->Text = "Max";
        }
    }
}
System::Void HacerioPilot::COMSettings::button3_Click(System::Object^ sender, System::EventArgs^ e)
{
    if (!String::IsNullOrEmpty(comboBox1->Text)) {
        Global::COM1 = Convert::ToString(comboBox1->Text);
    }
    if (!String::IsNullOrEmpty(comboBox3->Text)) {
        Global::COM2 = Convert::ToString(comboBox3->Text);
    }
    if (!String::IsNullOrEmpty(Convert::ToString(comboBox2->Text))) {
        if (IsNumeric(comboBox2->Text)) {
            Global::baud1 = Convert::ToInt32(comboBox2->Text);
        }
        else {
            MessageBox::Show("Букви недопустимі в записі");
        }
    }
    if (!String::IsNullOrEmpty(Convert::ToString(comboBox4->Text))) {
        if (IsNumeric(comboBox4->Text)) {
            Global::baud2 = Convert::ToInt32(comboBox4->Text);
        }
        else {
            MessageBox::Show("Букви недопустимі в записі");
        }
    }
}

System::Void HacerioPilot::COMSettings::COMSettings_Load(System::Object^ sender, System::EventArgs^ e)
{
    UpdateList();
    if (Global::COM1 != "255") {
        comboBox1->Text = Global::COM1;
    }
    if (Global::COM2 != "255") {
        comboBox3->Text = Global::COM2;
    }
}

```

```

        if (Global::baud1 != 255) {
            comboBox2->Text = Convert::ToString(Global::baud1);
        }
        if (Global::baud2 != 255) {
            comboBox4->Text = Convert::ToString(Global::baud2);
        }
    }

System::Void HacerioPilot::servoForm::button2_Click(System::Object^ sender, System::EventArgs^ e) {
    //Лівий
    if ((String::IsNullOrEmpty(textBox5->Text)) && Global::left.pin == 255) MessageBox::Show("Виберіть пін");
    else if (String::IsNullOrEmpty(textBox5->Text)) {}
    else if (!IsNumeric(textBox5->Text)) MessageBox::Show("Букви недопустимі в записі");
    else if (checkpin(Convert::ToByte(textBox5->Text))) Global::left.pin = Convert::ToByte(textBox5->Text);

    if ((String::IsNullOrEmpty(textBox4->Text)) && Global::left.min == 255) MessageBox::Show("Впишіть значення");
    else if (String::IsNullOrEmpty(textBox4->Text)) {}
    else if (!IsNumeric(textBox4->Text)) MessageBox::Show("Букви недопустимі в записі");
    else Global::left.min = Convert::ToByte(textBox4->Text);

    if ((String::IsNullOrEmpty(textBox6->Text)) && Global::left.nor == 255) MessageBox::Show("Впишіть значення");
    else if (String::IsNullOrEmpty(textBox6->Text)) {}
    else if (!IsNumeric(textBox6->Text)) MessageBox::Show("Букви недопустимі в записі");
    else Global::left.nor = Convert::ToByte(textBox6->Text);

    if ((String::IsNullOrEmpty(textBox7->Text)) && Global::left.max == 255) MessageBox::Show("Впишіть значення");
    else if (String::IsNullOrEmpty(textBox7->Text)) {}
    else if (!IsNumeric(textBox7->Text)) MessageBox::Show("Букви недопустимі в записі");
    else Global::left.max = Convert::ToByte(textBox7->Text);
    //Правий
    if (checkBox2->Checked == true) {
        Global::right.is_use = checkBox2->Checked;
        if ((String::IsNullOrEmpty(textBox10->Text)) && Global::right.pin == 255) MessageBox::Show("Виберіть
пін");
        else if (String::IsNullOrEmpty(textBox10->Text)) {}
        else if (!IsNumeric(textBox10->Text)) MessageBox::Show("Букви недопустимі в записі");
        else if (checkpin(Convert::ToByte(textBox10->Text))) Global::right.pin = Convert::ToByte(textBox10-
>Text);

        if ((String::IsNullOrEmpty(textBox11->Text)) && Global::right.min == 255) MessageBox::Show("Впишіть
значення");
        else if (String::IsNullOrEmpty(textBox11->Text)) {}
        else if (!IsNumeric(textBox11->Text)) MessageBox::Show("Букви недопустимі в записі");
        else Global::right.min = Convert::ToByte(textBox11->Text);

        if ((String::IsNullOrEmpty(textBox9->Text)) && Global::right.nor == 255) MessageBox::Show("Впишіть
значення");
        else if (String::IsNullOrEmpty(textBox9->Text)) {}
        else if (!IsNumeric(textBox9->Text)) MessageBox::Show("Букви недопустимі в записі");
        else Global::right.nor = Convert::ToByte(textBox9->Text);

        if ((String::IsNullOrEmpty(textBox12->Text)) && Global::right.max == 255) MessageBox::Show("Впишіть
значення");
        else if (String::IsNullOrEmpty(textBox12->Text)) {}
        else if (!IsNumeric(textBox12->Text)) MessageBox::Show("Букви недопустимі в записі");
        else Global::right.max = Convert::ToByte(textBox12->Text);
    }
    else Global::right.is_use = checkBox2->Checked;
    //Задній
    if ((String::IsNullOrEmpty(textBox17->Text)) && Global::back.pin == 255) MessageBox::Show("Виберіть пін");
    else if (String::IsNullOrEmpty(textBox17->Text)) {}
    else if (!IsNumeric(textBox17->Text)) MessageBox::Show("Букви недопустимі в записі");
    else if (checkpin(Convert::ToByte(textBox17->Text))) Global::back.pin = Convert::ToByte(textBox17->Text);

    if ((String::IsNullOrEmpty(textBox15->Text)) && Global::back.min == 255) MessageBox::Show("Впишіть
значення");
    else if (String::IsNullOrEmpty(textBox15->Text)) {}
    else if (!IsNumeric(textBox15->Text)) MessageBox::Show("Букви недопустимі в записі");
    else Global::back.min = Convert::ToByte(textBox15->Text);

```

```

        if ((String::IsNullOrEmpty(textBox19->Text)) && Global::back.nor == 255) MessageBox::Show("Впишіть
значення");
        else if (String::IsNullOrEmpty(textBox19->Text)) { }
        else if (!IsNumeric(textBox19->Text)) MessageBox::Show("Букви недопустимі в записі");
        else Global::back.nor = Convert::ToByte(textBox19->Text);

        if ((String::IsNullOrEmpty(textBox18->Text)) && Global::back.max == 255) MessageBox::Show("Впишіть
значення");
        else if (String::IsNullOrEmpty(textBox18->Text)) { }
        else if (!IsNumeric(textBox18->Text)) MessageBox::Show("Букви недопустимі в записі");
        else Global::back.max = Convert::ToByte(textBox18->Text);

        //Рискання
        if (checkBox1->Checked == true) {
            Global::rysk.is_use = checkBox1->Checked;
            if ((String::IsNullOrEmpty(textBox2->Text)) && Global::rysk.pin == 255) MessageBox::Show("Виберіть
пін");
            else if (String::IsNullOrEmpty(textBox2->Text)) { }
            else if (!IsNumeric(textBox2->Text)) MessageBox::Show("Букви недопустимі в записі");
            else if (checkBox1->Checked == true) Global::rysk.pin = Convert::ToByte(textBox2->Text);

            if ((String::IsNullOrEmpty(textBox1->Text)) && Global::rysk.min == 255) MessageBox::Show("Впишіть
значення");
            else if (String::IsNullOrEmpty(textBox1->Text)) { }
            else if (!IsNumeric(textBox1->Text)) MessageBox::Show("Букви недопустимі в записі");
            else Global::rysk.min = Convert::ToByte(textBox1->Text);

            if ((String::IsNullOrEmpty(textBox3->Text)) && Global::rysk.nor == 255) MessageBox::Show("Впишіть
значення");
            else if (String::IsNullOrEmpty(textBox3->Text)) { }
            else if (!IsNumeric(textBox3->Text)) MessageBox::Show("Букви недопустимі в записі");
            else Global::rysk.nor = Convert::ToByte(textBox3->Text);

            if ((String::IsNullOrEmpty(textBox8->Text)) && Global::rysk.max == 255) MessageBox::Show("Впишіть
значення");
            else if (String::IsNullOrEmpty(textBox8->Text)) { }
            else if (!IsNumeric(textBox8->Text)) MessageBox::Show("Букви недопустимі в записі");
            else Global::rysk.max = Convert::ToByte(textBox8->Text);
        }
        else Global::rysk.is_use = checkBox1->Checked;
    }
    System::Void HacerioPilot::servoForm::servoForm_Load(System::Object^ sender, System::EventArgs^ e)
    {
        if (Global::left.pin != 255) { label38->Text = Convert::ToString(Global::left.pin); label38->Show(); }else label38-
>Hide();
        if (Global::left.min != 255) { label39->Text = Convert::ToString(Global::left.min); label39->Show(); }else label39-
>Hide();
        if (Global::left.nor != 255) { label40->Text = Convert::ToString(Global::left.nor); label40->Show(); }else label40-
>Hide();
        if (Global::left.max != 255) { label41->Text = Convert::ToString(Global::left.max); label41->Show(); }else label41-
>Hide();

        if (Global::back.pin != 255) { label33->Text = Convert::ToString(Global::back.pin); label33->Show(); }else label33-
>Hide();
        if (Global::back.min != 255) { label35->Text = Convert::ToString(Global::back.min); label35->Show(); }else label35-
>Hide();
        if (Global::back.nor != 255) { label36->Text = Convert::ToString(Global::back.nor); label36->Show(); }else label36-
>Hide();
        if (Global::back.max != 255) { label37->Text = Convert::ToString(Global::back.max); label37->Show(); }else label37-
>Hide();

        checkBox2->Checked = Global::right.is_use;
        if (Global::right.pin != 255) { label17->Text = Convert::ToString(Global::right.pin); label17->Show(); }else label17-
>Hide();
        if (Global::right.min != 255) { label44->Text = Convert::ToString(Global::right.min); label44->Show(); }else label44-
>Hide();
        if (Global::right.nor != 255) { label45->Text = Convert::ToString(Global::right.nor); label45->Show(); }else label45-
>Hide();

```

```

        if (Global::right.max != 255) { label47->Text = Convert::ToString(Global::right.max); label47->Show(); }else label47-
>Hide();

        checkBox1->Checked = Global::rysk.is_use;
        if (Global::rysk.pin != 255) { label46->Text = Convert::ToString(Global::rysk.pin); label46->Show(); }else label46-
>Hide();
        if (Global::rysk.min != 255) { label48->Text = Convert::ToString(Global::rysk.min); label48->Show(); }else label48-
>Hide();
        if (Global::rysk.nor != 255) { label49->Text = Convert::ToString(Global::rysk.nor); label49->Show(); }else label49-
>Hide();
        if (Global::rysk.max != 255) { label50->Text = Convert::ToString(Global::rysk.max); label50->Show(); }else label50-
>Hide();
    }
    System::Void HacerioPilot::Form1::button5_Click(System::Object^ sender, System::EventArgs^ e){ clearData();}

    System::Void HacerioPilot::COMSettings::button2_Click(System::Object^ sender, System::EventArgs^ e)
    {
        serialPort1->BaudRate = Global::baud1;
        serialPort1->PortName = Global::COM1;
        array<Byte>^ byteArray = Global::GetAllBytesUAV();
        SendCOMUAV(byteArray);
    }

    System::Void HacerioPilot::COMSettings::button4_Click(System::Object^ sender, System::EventArgs^ e)
    {
        serialPort2->BaudRate = Global::baud2;
        serialPort2->PortName = Global::COM2;
        array<Byte>^ byteArray = Global::GetAllBytesPult();
        SendCOMPULT(byteArray);
    }

    System::Void HacerioPilot::settingsForm::settingsForm_Load(System::Object^ sender, System::EventArgs^ e)
    {
        if (Global::engine_joy_deadzone != 255) { label2->Show(); label2->Text =
Convert::ToString(Global::engine_joy_deadzone); }
        else label2->Hide();
        checkBox1->Checked = Global::engine_joy_is_joy;
        if (Global::kren_joy_deadzone != 255) { label8->Show(); label8->Text =
Convert::ToString(Global::kren_joy_deadzone); }
        else label8->Hide();
        if (Global::tang_joy_deadzone != 255) { label13->Show(); label13->Text =
Convert::ToString(Global::tang_joy_deadzone); }
        else label13->Hide();
        if (Global::rysk_joy_deadzone != 255) { label11->Show(); label11->Text =
Convert::ToString(Global::rysk_joy_deadzone); }
        else label11->Hide();
        checkBox2->Checked = Global::rysk_joy_isuse;
        checkBox7->Checked = Global::but1.is_use;
        checkBox9->Checked = Global::but2.is_use;
        checkBox5->Checked = Global::but1.is_tumb;
        checkBox8->Checked = Global::but2.is_tumb;
    }
    System::Void HacerioPilot::settingsForm::button2_Click(System::Object^ sender, System::EventArgs^ e)
    {
        if (String::IsNullOrEmpty(textBox5->Text) && Global::engine_joy_deadzone == 255)MessageBox::Show("Впишіть
значення");
        else if (String::IsNullOrEmpty(textBox5->Text)) { }
        else if (!IsNumeric(textBox5->Text))MessageBox::Show("Букви недопустимі в записі");
        else Global::engine_joy_deadzone = Convert::ToByte(textBox5->Text);
        Global::engine_joy_is_joy = checkBox1->Checked;

        if (String::IsNullOrEmpty(textBox8->Text) && Global::kren_joy_deadzone == 255)MessageBox::Show("Впишіть
значення");
        else if (String::IsNullOrEmpty(textBox8->Text)) { }
        else if (!IsNumeric(textBox8->Text))MessageBox::Show("Букви недопустимі в записі");
        else Global::kren_joy_deadzone = Convert::ToByte(textBox8->Text);

        if (checkBox2->Checked) {

```

```

        if (String::IsEmpty(textBox16->Text) && Global::rysk_joy_deadzone ==
255)MessageBox::Show("Впишіть значення");
        else if (String::IsEmpty(textBox16->Text)) { }
        else if (!IsNumeric(textBox16->Text))MessageBox::Show("Букви недопустимі в записі");
        else Global::rysk_joy_deadzone = Convert::ToByte(textBox16->Text);
    }
    Global::rysk_joy_isuse = checkBox2->Checked;

    if (String::IsEmpty(textBox12->Text) && Global::tang_joy_deadzone == 255)MessageBox::Show("Впишіть
значення");
    else if (String::IsEmpty(textBox12->Text)) { }
    else if (!IsNumeric(textBox12->Text))MessageBox::Show("Букви недопустимі в записі");
    else Global::tang_joy_deadzone = Convert::ToByte(textBox12->Text);
    Global::but1.is_tumb = checkBox5->Checked;
    Global::but2.is_tumb = checkBox8->Checked;
}

System::Void HacerioPilot::servoForm::button5_Click(System::Object^ sender, System::EventArgs^ e)
{
    if (!String::IsEmpty(Global::COM1) && (Global::baud1 != 255)) {
        serialPort1->PortName = Global::COM1;
        serialPort1->BaudRate = Global::baud1;
        int offset = 0;
        int is_ok = 0;
        Global::is_write = 0;
        Global::is_test = 1;
        array<Byte>^ byteArray = gcnew array<Byte>(sizeof(Servo) + sizeof(Global::is_write) +
sizeof(Global::is_test));
        pinBytes(offset, byteArray, Global::is_write);
        pinBytes(offset, byteArray, Global::is_test);
        if (String::IsEmpty(textBox5->Text)) { MessageBox::Show("Виберіть пін"); }
        else if (!IsNumeric(textBox5->Text)) MessageBox::Show("Букви недопустимі в записі");
        else { pinBytes(offset, byteArray, Convert::ToByte(textBox5->Text)); is_ok++; }

        if (String::IsEmpty(textBox4->Text)) { MessageBox::Show("Впишіть значення"); }
        else if (!IsNumeric(textBox4->Text)) MessageBox::Show("Букви недопустимі в записі");
        else { pinBytes(offset, byteArray, Convert::ToByte(textBox4->Text)); is_ok++; }

        if (String::IsEmpty(textBox6->Text)) { MessageBox::Show("Впишіть значення"); }
        else if (!IsNumeric(textBox6->Text)) MessageBox::Show("Букви недопустимі в записі");
        else { pinBytes(offset, byteArray, Convert::ToByte(textBox6->Text)); is_ok++; }

        if (String::IsEmpty(textBox7->Text)) { MessageBox::Show("Впишіть значення"); }
        else if (!IsNumeric(textBox7->Text)) MessageBox::Show("Букви недопустимі в записі");
        else { pinBytes(offset, byteArray, Convert::ToByte(textBox7->Text)); is_ok++; }
        if (is_ok == 4) {
            SendCOMTest(byteArray);
        }
        Global::is_write = 0;
        Global::is_test = 0;
    }
    else MessageBox::Show("Виберіть порт та швидкість передачі");
}

System::Void HacerioPilot::servoForm::button6_Click(System::Object^ sender, System::EventArgs^ e)
{
    if (!String::IsEmpty(Global::COM1) && (Global::baud1 != 255)) {
        serialPort1->PortName = Global::COM1;
        serialPort1->BaudRate = Global::baud1;
        int offset = 0;
        int is_ok = 0;
        Global::is_write = 0;
        Global::is_test = 1;
        array<Byte>^ byteArray = gcnew array<Byte>(sizeof(Servo) + sizeof(Global::is_write) +
sizeof(Global::is_test));
        pinBytes(offset, byteArray, Global::is_write);
        pinBytes(offset, byteArray, Global::is_test);
        if (String::IsEmpty(textBox10->Text)) { MessageBox::Show("Виберіть пін"); }
        else if (!IsNumeric(textBox10->Text)) MessageBox::Show("Букви недопустимі в записі");
        else { pinBytes(offset, byteArray, Convert::ToByte(textBox10->Text)); is_ok++; }
    }
}

```



```

        if (String::IsNullOrEmpty(textBox11->Text)) { MessageBox::Show("Впишіть значення"); }
        else if (!IsNumeric(textBox11->Text)) MessageBox::Show("Букви недопустимі в записі");
        else { pinBytes(offset, byteArray, Convert::ToByte(textBox11->Text)); is_ok++; }

        if (String::IsNullOrEmpty(textBox9->Text)) { MessageBox::Show("Впишіть значення"); }
        else if (!IsNumeric(textBox9->Text)) MessageBox::Show("Букви недопустимі в записі");
        else { pinBytes(offset, byteArray, Convert::ToByte(textBox9->Text)); is_ok++; }

        if (String::IsNullOrEmpty(textBox12->Text)) { MessageBox::Show("Впишіть значення"); }
        else if (!IsNumeric(textBox12->Text)) MessageBox::Show("Букви недопустимі в записі");
        else { pinBytes(offset, byteArray, Convert::ToByte(textBox12->Text)); is_ok++; }
        if (is_ok == 4) {
            SendCOMTest(byteArray);
        }
        Global::is_write = 0;
        Global::is_test = 0;
    }
    else MessageBox::Show("Виберіть порт та швидкість передачі");
}
System::Void HacerioPilot::servoForm::button4_Click(System::Object^ sender, System::EventArgs^ e)
{
    if ((!String::IsNullOrEmpty(Global::COM1)) && (Global::baud1 != 255)) {
        serialPort1->PortName = Global::COM1;
        serialPort1->BaudRate = Global::baud1;
        int offset = 0;
        int is_ok = 0;
        Global::is_write = 0;
        Global::is_test = 1;
        array<Byte>^ byteArray = gcnew array<Byte>(sizeof(Servo) + sizeof(Global::is_write) +
sizeof(Global::is_test));
        pinBytes(offset, byteArray, Global::is_write);
        pinBytes(offset, byteArray, Global::is_test);
        if (String::IsNullOrEmpty(textBox17->Text)) { MessageBox::Show("Виберіть пін"); }
        else if (!IsNumeric(textBox17->Text)) MessageBox::Show("Букви недопустимі в записі");
        else { pinBytes(offset, byteArray, Convert::ToByte(textBox17->Text)); is_ok++; }

        if (String::IsNullOrEmpty(textBox15->Text)) { MessageBox::Show("Впишіть значення"); }
        else if (!IsNumeric(textBox15->Text)) MessageBox::Show("Букви недопустимі в записі");
        else { pinBytes(offset, byteArray, Convert::ToByte(textBox15->Text)); is_ok++; }

        if (String::IsNullOrEmpty(textBox19->Text)) { MessageBox::Show("Впишіть значення"); }
        else if (!IsNumeric(textBox19->Text)) MessageBox::Show("Букви недопустимі в записі");
        else { pinBytes(offset, byteArray, Convert::ToByte(textBox19->Text)); is_ok++; }

        if (String::IsNullOrEmpty(textBox18->Text)) { MessageBox::Show("Впишіть значення"); }
        else if (!IsNumeric(textBox18->Text)) MessageBox::Show("Букви недопустимі в записі");
        else { pinBytes(offset, byteArray, Convert::ToByte(textBox18->Text)); is_ok++; }
        if (is_ok == 4) {
            SendCOMTest(byteArray);
        }
        Global::is_write = 0;
        Global::is_test = 0;
    }
    else MessageBox::Show("Виберіть порт та швидкість передачі");
}
System::Void HacerioPilot::servoForm::button8_Click(System::Object^ sender, System::EventArgs^ e)
{
    if ((!String::IsNullOrEmpty(Global::COM1)) && (Global::baud1 != 255)) {
        serialPort1->PortName = Global::COM1;
        serialPort1->BaudRate = Global::baud1;
        int offset = 0;
        int is_ok = 0;
        Global::is_write = 0;
        Global::is_test = 1;
        array<Byte>^ byteArray = gcnew array<Byte>(sizeof(Servo) + sizeof(Global::is_write) +
sizeof(Global::is_test));
        pinBytes(offset, byteArray, Global::is_write);
    }
}

```

```

pinBytes(offset, byteArray, Global::is_test);
if (String::IsEmpty(textBox2->Text)) { MessageBox::Show("Виберіть пін"); }
else if (!IsNumeric(textBox2->Text)) MessageBox::Show("Букви недопустимі в записі");
else { pinBytes(offset, byteArray, Convert::ToByte(textBox2->Text)); is_ok++; }

if (String::IsEmpty(textBox1->Text)) { MessageBox::Show("Впишіть значення"); }
else if (!IsNumeric(textBox1->Text)) MessageBox::Show("Букви недопустимі в записі");
else { pinBytes(offset, byteArray, Convert::ToByte(textBox1->Text)); is_ok++; }

if (String::IsEmpty(textBox3->Text)) { MessageBox::Show("Впишіть значення"); }
else if (!IsNumeric(textBox3->Text)) MessageBox::Show("Букви недопустимі в записі");
else { pinBytes(offset, byteArray, Convert::ToByte(textBox3->Text)); is_ok++; }

if (String::IsEmpty(textBox8->Text)) { MessageBox::Show("Впишіть значення"); }
else if (!IsNumeric(textBox8->Text)) MessageBox::Show("Букви недопустимі в записі");
else { pinBytes(offset, byteArray, Convert::ToByte(textBox8->Text)); is_ok++; }
if (is_ok == 4) {
    SendCOMTest(byteArray);
}
Global::is_write = 0;
Global::is_test = 0;
}
else MessageBox::Show("Виберіть порт та швидкість передачі");
}
System::Void HacerioPilot::Form1::зберегтиToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e)
{
    if (Global::filepath == "NULL") {
        OpenFileDialog^ openFileDialog1 = gcnew OpenFileDialog();
        openFileDialog1->Title = "Оберіть файл для відкриття";
        openFileDialog1->InitialDirectory = "C:\\";
        openFileDialog1->Filter = "HacerioPilot Config (*.hpc)|*.hpc|Всі файли (*.*)|*.*";
        label5->Text = Global::filepath;
        label5->Show();
        label4->Show();
        if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK) {
            String^ filePath = openFileDialog1->FileName;
            WriteToFile(filePath);
        }
    }
    else {
        WriteToFile(Global::filepath);
    }
}
System::Void HacerioPilot::Form1::githubToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e)
{
    LPCWSTR url = L"https://github.com/hacer1o/";
    ShellExecute(NULL, L"open", url, NULL, NULL, SW_SHOWNORMAL);
}
System::Void HacerioPilot::Form1::Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
    if (Global::filepath == "NULL") {
        label5->Hide();
        label4->Hide();
    }
    else {
        label5->Text = Global::filepath;
        label5->Show();
        label4->Show();
    }
}

```

## COMSettings.h

```
#pragma once
```

```

namespace HacerioPilot {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;

```

```

using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::Threading;
/// <summary>
/// Summary for COMSettings
/// </summary>
public ref class COMSettings : public System::Windows::Forms::Form
{
public:
    COMSettings(void)
    {
        InitializeComponent();
        //
        //TODO: Add the constructor code here
        //
    }

protected:
    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    ~COMSettings()
    {
        if (components)
        {
            delete components;
        }
    }

private: System::Windows::Forms::ComboBox^ comboBox1;
private: System::Windows::Forms::ComboBox^ comboBox2;
private: System::Windows::Forms::ComboBox^ comboBox3;
private: System::Windows::Forms::ComboBox^ comboBox4;
private: System::Windows::Forms::Button^ button1;
private: System::Windows::Forms::Button^ button2;
private: System::Windows::Forms::Button^ button3;
private: System::Windows::Forms::Button^ button4;
private: System::Windows::Forms::Label^ label1;
private: System::Windows::Forms::Label^ label2;
private: System::Windows::Forms::Label^ label3;
private: System::Windows::Forms::Label^ label4;
private: System::Windows::Forms::Button^ button5;
private: System::IO::Ports::SerialPort^ serialPort1;
private: System::IO::Ports::SerialPort^ serialPort2;

private: System::ComponentModel::IContainer^ components;
protected:

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    void InitializeComponent(void)
    {
        this->components = (gcnew System::ComponentModel::Container());
        System::ComponentModel::ComponentResourceManager^ resources = (gcnew
System::ComponentModel::ComponentResourceManager(COMSettings::typeid));
        this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());
        this->comboBox2 = (gcnew System::Windows::Forms::ComboBox());
        this->comboBox3 = (gcnew System::Windows::Forms::ComboBox());
        this->comboBox4 = (gcnew System::Windows::Forms::ComboBox());
        this->button1 = (gcnew System::Windows::Forms::Button());
        this->button2 = (gcnew System::Windows::Forms::Button());
        this->button3 = (gcnew System::Windows::Forms::Button());
        this->button4 = (gcnew System::Windows::Forms::Button());
        this->label1 = (gcnew System::Windows::Forms::Label());
        this->label2 = (gcnew System::Windows::Forms::Label());
    }
}

```

```

this->label3 = (gcnew System::Windows::Forms::Label());
this->label4 = (gcnew System::Windows::Forms::Label());
this->button5 = (gcnew System::Windows::Forms::Button());
this->serialPort1 = (gcnew System::IO::Ports::SerialPort(this->components));
this->serialPort2 = (gcnew System::IO::Ports::SerialPort(this->components));
this->SuspendLayout();
//
// comboBox1
//
this->comboBox1->FormattingEnabled = true;
this->comboBox1->Location = System::Drawing::Point(258, 26);
this->comboBox1->Name = L"comboBox1";
this->comboBox1->Size = System::Drawing::Size(121, 21);
this->comboBox1->TabIndex = 0;
//
// comboBox2
//
this->comboBox2->FormattingEnabled = true;
this->comboBox2->Location = System::Drawing::Point(258, 53);
this->comboBox2->Name = L"comboBox2";
this->comboBox2->Size = System::Drawing::Size(121, 21);
this->comboBox2->TabIndex = 1;
//
// comboBox3
//
this->comboBox3->FormattingEnabled = true;
this->comboBox3->Location = System::Drawing::Point(258, 109);
this->comboBox3->Name = L"comboBox3";
this->comboBox3->Size = System::Drawing::Size(121, 21);
this->comboBox3->TabIndex = 2;
this->comboBox3->SelectedIndexChanged += gcnew System::EventHandler(this,
&COMSettings::comboBox3_SelectedIndexChanged);
//
// comboBox4
//
this->comboBox4->FormattingEnabled = true;
this->comboBox4->Location = System::Drawing::Point(258, 136);
this->comboBox4->Name = L"comboBox4";
this->comboBox4->Size = System::Drawing::Size(121, 21);
this->comboBox4->TabIndex = 3;
//
// button1
//
this->button1->BackColor = System::Drawing::Color::LightBlue;
this->button1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button1->Location = System::Drawing::Point(258, 80);
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(121, 23);
this->button1->TabIndex = 4;
this->button1->Text = L"Refresh";
this->button1->UseVisualStyleBackColor = false;
this->button1->Click += gcnew System::EventHandler(this, &COMSettings::button1_Click);
//
// button2
//
this->button2->BackColor = System::Drawing::Color::LightBlue;
this->button2->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button2->Location = System::Drawing::Point(12, 280);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(110, 36);
this->button2->TabIndex = 5;
this->button2->Text = L"Конфігурувати БПЛА";
this->button2->UseVisualStyleBackColor = false;
this->button2->Click += gcnew System::EventHandler(this, &COMSettings::button2_Click);
//
// button3
//
this->button3->BackColor = System::Drawing::Color::LightBlue;
this->button3->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button3->Location = System::Drawing::Point(128, 238);
this->button3->Name = L"button3";
this->button3->Size = System::Drawing::Size(135, 36);
this->button3->TabIndex = 6;
this->button3->Text = L"Зберегти";
this->button3->UseVisualStyleBackColor = false;
this->button3->Click += gcnew System::EventHandler(this, &COMSettings::button3_Click);
//
// button4

```

```

//
this->button4->BackColor = System::Drawing::Color::LightBlue;
this->button4->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button4->Location = System::Drawing::Point(269, 280);
this->button4->Name = L"button4";
this->button4->Size = System::Drawing::Size(110, 36);
this->button4->TabIndex = 7;
this->button4->Text = L"Конфігурувати Пульти";
this->button4->UseVisualStyleBackColor = false;
this->button4->Click += gcnew System::EventHandler(this, &COMSettings::button4_Click);
//
// label1
//
this->label1->AutoSize = true;
this->label1->BackColor = System::Drawing::Color::Transparent;
this->label1->Location = System::Drawing::Point(12, 29);
this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(102, 13);
this->label1->TabIndex = 8;
this->label1->Text = L"COM Безпілотника";
//
// label2
//
this->label2->AutoSize = true;
this->label2->BackColor = System::Drawing::Color::Transparent;
this->label2->Location = System::Drawing::Point(12, 56);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(241, 13);
this->label2->TabIndex = 9;
this->label2->Text = L"Швидкість передачі даних (стандарт 9600 Бод)";
//
// label3
//
this->label3->AutoSize = true;
this->label3->BackColor = System::Drawing::Color::Transparent;
this->label3->Location = System::Drawing::Point(12, 112);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(70, 13);
this->label3->TabIndex = 10;
this->label3->Text = L"COM Пульта";
//
// label4
//
this->label4->AutoSize = true;
this->label4->BackColor = System::Drawing::Color::Transparent;
this->label4->Location = System::Drawing::Point(12, 139);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(137, 13);
this->label4->TabIndex = 11;
this->label4->Text = L"Швидкість передачі даних";
//
// button5
//
this->button5->BackColor = System::Drawing::Color::LightBlue;
this->button5->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button5->Location = System::Drawing::Point(128, 280);
this->button5->Name = L"button5";
this->button5->Size = System::Drawing::Size(135, 36);
this->button5->TabIndex = 12;
this->button5->Text = L"Вийти";
this->button5->UseVisualStyleBackColor = false;
this->button5->Click += gcnew System::EventHandler(this, &COMSettings::button5_Click);
//
// COMSettings
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(386, 327);
this->ControlBox = false;
this->Controls->Add(this->button5);
this->Controls->Add(this->label4);
this->Controls->Add(this->label3);
this->Controls->Add(this->label2);
this->Controls->Add(this->label1);
this->Controls->Add(this->button4);
this->Controls->Add(this->button3);
this->Controls->Add(this->button2);
this->Controls->Add(this->button1);

```

```

        this->Controls->Add(this->comboBox4);
        this->Controls->Add(this->comboBox3);
        this->Controls->Add(this->comboBox2);
        this->Controls->Add(this->comboBox1);
        this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
        this->Icon = (cli::safe_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));
        this->Name = L"COMSettings";
        this->Text = L"Налаштування COM-порт";
        this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this,
&COMSettings::COMSettings_FormClosing);
        this->Load += gcnew System::EventHandler(this, &COMSettings::COMSettings_Load);
        this->ResumeLayout(false);
        this->PerformLayout();

    }

#pragma endregion

#pragma region ComPort
    public: void UpdateList() {
        array<System::String^>^ ListPorts = SerialPort::GetPortNames();
        int bauds[] = { 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 };
        comboBox1->Items->Clear();
        comboBox3->Items->Clear();
        comboBox2->Items->Clear();
        comboBox4->Items->Clear();
        for (size_t i = 0; i < ListPorts->Length; i++)
        {
            comboBox1->Items->Add(ListPorts[i]);
            comboBox3->Items->Add(ListPorts[i]);
        }
        for (size_t i = 0; i < sizeof(bauds) / sizeof(int); i++)
        {
            comboBox2->Items->Add(bauds[i]);
            comboBox4->Items->Add(bauds[i]);
        }
    }

#pragma endregion

    public: void SendCOMUAV(array<Byte>^ byteArray) {
        try {
            if (serialPort1 != nullptr) {
                if (!serialPort1->IsOpen)
                {
                    serialPort1->Open();
                    serialPort1->Write(byteArray, 0, byteArray->Length);
                    serialPort1->Close();
                    Console::WriteLine(byteArray->Length);
                    for (size_t i = 0; i < byteArray->Length; i++)
                    {
                        Console::WriteLine(byteArray[i]);
                    }
                    MessageBox::Show("Дані відправлено успішно");
                }
            }
            else {
                MessageBox::Show("Порт зайнятий. Можливо на цей порт вже
відправляються дані, або інша програма використовує його");
            }
        }
        catch (Exception^ ex) {
            MessageBox::Show("Виникла помилка під час відправки даних в порт: " + ex-
>Message);
        }
    }

    public: void SendCOMPULT(array<Byte>^ byteArray) {
        try {
            if (serialPort2 != nullptr) {
                if (!serialPort2->IsOpen)
                {
                    serialPort2->Open();
                    serialPort2->Write(byteArray, 0, byteArray->Length);
                    serialPort2->Close();
                    MessageBox::Show("Дані відправлено успішно");
                }
            }
            else {
                MessageBox::Show("Порт зайнятий. Можливо на цей порт вже
відправляються дані, або інша програма використовує його");
            }
        }
        catch (Exception^ ex) {

```

```

        MessageBox::Show("Виникла помилка під час відправки даних в порт: " + ex-
>Message);
    }
}
private: System::Void comboBox3_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void COMSettings_Load(System::Object^ sender, System::EventArgs^ e);
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
    UpdateList();
}
private: System::Void COMSettings_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e) {
    Owner->Show();
}
private: System::Void button5_Click(System::Object^ sender, System::EventArgs^ e) {
    Owner->Show();
    this->Hide();
}
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void button4_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void button6_Click(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void button8_Click(System::Object^ sender, System::EventArgs^ e) {
}
};
}

```

## engineForm.h

#pragma once

```

namespace HacerioPilot {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Summary for engineForm
    /// </summary>
    public ref class engineForm : public System::Windows::Forms::Form
    {
    public:
        engineForm(void)
        {
            InitializeComponent();
            //
            //TODO: Add the constructor code here
            //
        }

    protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ~engineForm()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::Label^ label1;
    protected:

    private: System::Windows::Forms::Label^ label3;
    private: System::Windows::Forms::TextBox^ textBox3;
    private: System::Windows::Forms::Button^ button1;

    private: System::Windows::Forms::CheckBox^ checkBox1;
    private: System::Windows::Forms::TextBox^ textBox4;

```

```

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::Button^ button2;

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    void InitializeComponent(void)
    {
        System::ComponentModel::ComponentResourceManager^ resources = (gcnew
System::ComponentModel::ComponentResourceManager(engineForm::typeid));
        this->label1 = (gcnew System::Windows::Forms::Label());
        this->label3 = (gcnew System::Windows::Forms::Label());
        this->textBox3 = (gcnew System::Windows::Forms::TextBox());
        this->button1 = (gcnew System::Windows::Forms::Button());
        this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());
        this->textBox4 = (gcnew System::Windows::Forms::TextBox());
        this->label4 = (gcnew System::Windows::Forms::Label());
        this->label6 = (gcnew System::Windows::Forms::Label());
        this->button2 = (gcnew System::Windows::Forms::Button());
        this->SuspendLayout();
        //
        // label1
        //
        this->label1->AutoSize = true;
        this->label1->BackColor = System::Drawing::Color::Transparent;
        this->label1->Location = System::Drawing::Point(19, 33);
        this->label1->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
        this->label1->Name = L"label1";
        this->label1->Size = System::Drawing::Size(66, 13);
        this->label1->TabIndex = 0;
        this->label1->Text = L"Пін двигуна";
        //
        // label3
        //
        this->label3->AutoSize = true;
        this->label3->BackColor = System::Drawing::Color::Transparent;
        this->label3->Location = System::Drawing::Point(19, 63);
        this->label3->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
        this->label3->Name = L"label3";
        this->label3->Size = System::Drawing::Size(160, 13);
        this->label3->TabIndex = 5;
        this->label3->Text = L"Обмеження оборотів (0 -100%)";
        //
        // textBox3
        //
        this->textBox3->BackColor = System::Drawing::SystemColors::Control;
        this->textBox3->Location = System::Drawing::Point(202, 60);
        this->textBox3->Margin = System::Windows::Forms::Padding(1);
        this->textBox3->MaxLength = 3;
        this->textBox3->Name = L"textBox3";
        this->textBox3->Size = System::Drawing::Size(34, 20);
        this->textBox3->TabIndex = 6;
        //
        // button1
        //
        this->button1->BackColor = System::Drawing::Color::LightBlue;
        this->button1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
        this->button1->Location = System::Drawing::Point(10, 129);
        this->button1->Margin = System::Windows::Forms::Padding(1);
        this->button1->Name = L"button1";
        this->button1->Size = System::Drawing::Size(132, 33);
        this->button1->TabIndex = 7;
        this->button1->Text = L"Зберегти";
    }

```



```

this->button1->UseVisualStyleBackColor = false;
this->button1->Click += gcnew System::EventHandler(this, &engineForm::button1_Click);
//
// checkBox1
//
this->checkBox1->AutoSize = true;
this->checkBox1->BackColor = System::Drawing::Color::Transparent;
this->checkBox1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->checkBox1->Location = System::Drawing::Point(22, 86);
this->checkBox1->Name = L"checkBox1";
this->checkBox1->Size = System::Drawing::Size(160, 17);
this->checkBox1->TabIndex = 9;
this->checkBox1->Text = L"Керування сервоприводом";
this->checkBox1->UseVisualStyleBackColor = false;
this->checkBox1->CheckedChanged += gcnew System::EventHandler(this,
&engineForm::checkBox1_CheckedChanged);
//
// textBox4
//
this->textBox4->BackColor = System::Drawing::SystemColors::Control;
this->textBox4->Location = System::Drawing::Point(117, 33);
this->textBox4->MaxLength = 2;
this->textBox4->Name = L"textBox4";
this->textBox4->Size = System::Drawing::Size(119, 20);
this->textBox4->TabIndex = 10;
//
// label4
//
this->label4->AutoSize = true;
this->label4->BackColor = System::Drawing::Color::Transparent;
this->label4->Location = System::Drawing::Point(245, 36);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(35, 13);
this->label4->TabIndex = 11;
this->label4->Text = L"label4";
//
// label6
//
this->label6->AutoSize = true;
this->label6->BackColor = System::Drawing::Color::Transparent;
this->label6->Location = System::Drawing::Point(245, 63);
this->label6->Name = L"label6";
this->label6->Size = System::Drawing::Size(35, 13);
this->label6->TabIndex = 13;
this->label6->Text = L"label6";
//
// button2
//
this->button2->BackColor = System::Drawing::Color::LightBlue;
this->button2->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button2->Location = System::Drawing::Point(157, 129);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(123, 33);
this->button2->TabIndex = 16;
this->button2->Text = L"Вийти";
this->button2->UseVisualStyleBackColor = false;
this->button2->Click += gcnew System::EventHandler(this, &engineForm::button2_Click);
//
// engineForm
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->BackColor = System::Drawing::SystemColors::ScrollBar;
this->ClientSize = System::Drawing::Size(292, 177);
this->ControlBox = false;
this->Controls->Add(this->button2);
this->Controls->Add(this->label6);
this->Controls->Add(this->label4);
this->Controls->Add(this->textBox4);
this->Controls->Add(this->checkBox1);
this->Controls->Add(this->button1);
this->Controls->Add(this->textBox3);
this->Controls->Add(this->label3);
this->Controls->Add(this->label1);
this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
this->Icon = (cli::safe_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));
this->Margin = System::Windows::Forms::Padding(1);
this->Name = L"engineForm";

```

```

        this->Text = L"Налаштування двигуна";
        this->FormClosed += gcnew System::Windows::Forms::FormClosedEventHandler(this,
&engineForm::engineForm_FormClosed);
        this->Load += gcnew System::EventHandler(this, &engineForm::engineForm_Load);
        this->ResumeLayout(false);
        this->PerformLayout();
    }
#pragma endregion
    private: System::Void listView1_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {
    }
    private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void engineForm_FormClosed(System::Object^ sender, System::Windows::Forms::FormClosedEventArgs^ e) {
    Owner->Show();
}
private: System::Void label4_Click(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void checkBox1_CheckedChanged(System::Object^ sender, System::EventArgs^ e);
private: System::Void engineForm_Load(System::Object^ sender, System::EventArgs^ e);
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
    this->Hide();
    Owner->Show();
}
};
}

```

## Form1.h

```
#pragma once
```

```

namespace HacerioPilot {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    using namespace System::IO::Ports;
    /// <summary>
    /// Summary for Form1
    /// </summary>
    public ref class Form1 : public System::Windows::Forms::Form
    {
    public:
        Form1(void)
        {
            InitializeComponent();
            //
            //TODO: Add the constructor code here
            //
        }

    protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::Label^ label1;
    private: System::Windows::Forms::Button^ button1;
    private: System::Windows::Forms::Button^ button2;
    private: System::Windows::Forms::Button^ button3;
    private: System::Windows::Forms::Button^ button4;

    private: System::Windows::Forms::Label^ label3;
    private: System::Windows::Forms::Button^ button6;
    private: System::Windows::Forms::Button^ button7;
}

```

```

private: System::Windows::Forms::OpenFileDialog^ openFileDialog1;

private: System::Windows::Forms::MenuStrip^ menuStrip1;
private: System::Windows::Forms::ToolStripMenuItem^ файлToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ відкритиToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ зберегтиЯкToolStripMenuItem;


private: System::Windows::Forms::ToolStripMenuItem^ новийToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ прошитиToolStripMenuItem;


private: System::Windows::Forms::ToolStripMenuItem^ налаштуванняCOMPORTToolStripMenuItem;
private: System::Windows::Forms::Button^ button5;
private: System::Windows::Forms::Panel^ panel1;
private: System::Windows::Forms::Label^ label2;
private: System::Windows::Forms::Label^ label4;
private: System::Windows::Forms::Label^ label5;
private: System::Windows::Forms::ToolStripMenuItem^ зберегтиToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ довідкаToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ технічніВимогиToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ cOMPORTToolStripMenuItem;


private: System::Windows::Forms::ToolStripMenuItem^ розробникToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ githubToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ проРозробникаToolStripMenuItem;


private: System::Windows::Forms::Label^ label6;
private: System::Windows::Forms::ToolStripMenuItem^ застереженняToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^ aboutHacerioPilotToolStripMenuItem;

```

```

private: System::ComponentModel::IContainer^ components;

```

```

protected:

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    void InitializeComponent(void)
    {
        System::ComponentModel::ComponentResourceManager^ resources = (gcnew
System::ComponentModel::ComponentResourceManager(Form1::typeid));
        this->label1 = (gcnew System::Windows::Forms::Label());
        this->button1 = (gcnew System::Windows::Forms::Button());
        this->button2 = (gcnew System::Windows::Forms::Button());
        this->button3 = (gcnew System::Windows::Forms::Button());
        this->button4 = (gcnew System::Windows::Forms::Button());
        this->label3 = (gcnew System::Windows::Forms::Label());
        this->button6 = (gcnew System::Windows::Forms::Button());
        this->button7 = (gcnew System::Windows::Forms::Button());
        this->openFileDialog1 = (gcnew System::Windows::Forms::OpenFileDialog());
        this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());
        this->файлToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->новийToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->відкритиToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->зберегтиToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->зберегтиЯкToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->прошитиToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->налаштуванняCOMPORTToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
        this->довідкаToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->технічніВимогиToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->cOMPORTToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->застереженняToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->розробникToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->проРозробникаToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->githubToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->aboutHacerioPilotToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
        this->button5 = (gcnew System::Windows::Forms::Button());
        this->panel1 = (gcnew System::Windows::Forms::Panel());
        this->label6 = (gcnew System::Windows::Forms::Label());
        this->label2 = (gcnew System::Windows::Forms::Label());
        this->label4 = (gcnew System::Windows::Forms::Label());
        this->label5 = (gcnew System::Windows::Forms::Label());
        this->menuStrip1->SuspendLayout();
        this->panel1->SuspendLayout();
        this->SuspendLayout();
        //
        // label1
        //
        this->label1->AutoSize = true;
        this->label1->BackColor = System::Drawing::Color::Transparent;
        this->label1->Location = System::Drawing::Point(10, 37);
        this->label1->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
        this->label1->Name = L"label1";
        this->label1->Size = System::Drawing::Size(151, 13);
        this->label1->TabIndex = 0;
        this->label1->Text = L"Налаштування безпілотної";
        //
        // button1
        //
        this->button1->BackColor = System::Drawing::Color::LightBlue;
        this->button1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
        this->button1->ForeColor = System::Drawing::SystemColors::ControlText;
        this->button1->Location = System::Drawing::Point(10, 179);
        this->button1->Margin = System::Windows::Forms::Padding(1);
        this->button1->Name = L"button1";
        this->button1->Size = System::Drawing::Size(191, 62);
        this->button1->TabIndex = 1;
        this->button1->Text = L"Радіозв'язок";
        this->button1->UseVisualStyleBackColor = false;
        this->button1->Click += gcnew System::EventHandler(this, &Form1::button1_Click);
    }

```

```

//
// button2
//
this->button2->BackColor = System::Drawing::Color::LightBlue;
this->button2->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button2->Location = System::Drawing::Point(10, 51);
this->button2->Margin = System::Windows::Forms::Padding(1);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(191, 62);
this->button2->TabIndex = 2;
this->button2->Text = L"Двигун";
this->button2->UseVisualStyleBackColor = false;
this->button2->Click += gcnew System::EventHandler(this, &Form1::button2_Click);
//
// button3
//
this->button3->BackColor = System::Drawing::Color::LightBlue;
this->button3->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button3->Location = System::Drawing::Point(10, 115);
this->button3->Margin = System::Windows::Forms::Padding(1);
this->button3->Name = L"button3";
this->button3->Size = System::Drawing::Size(191, 62);
this->button3->TabIndex = 3;
this->button3->Text = L"Сервоприводи";
this->button3->UseVisualStyleBackColor = false;
this->button3->Click += gcnew System::EventHandler(this, &Form1::button3_Click);
//
// button4
//
this->button4->BackColor = System::Drawing::Color::LightBlue;
this->button4->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button4->Location = System::Drawing::Point(10, 243);
this->button4->Margin = System::Windows::Forms::Padding(1);
this->button4->Name = L"button4";
this->button4->Size = System::Drawing::Size(191, 62);
this->button4->TabIndex = 4;
this->button4->Text = L"Додаткові Функції";
this->button4->UseVisualStyleBackColor = false;
this->button4->Click += gcnew System::EventHandler(this, &Form1::button4_Click);
//
// label3
//
this->label3->AutoSize = true;
this->label3->BackColor = System::Drawing::Color::Transparent;
this->label3->Location = System::Drawing::Point(499, 37);
this->label3->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(118, 13);
this->label3->TabIndex = 8;
this->label3->Text = L"Налаштування пульта";
//
// button6
//
this->button6->BackColor = System::Drawing::Color::LightBlue;
this->button6->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button6->Location = System::Drawing::Point(426, 51);
this->button6->Margin = System::Windows::Forms::Padding(1);
this->button6->Name = L"button6";
this->button6->Size = System::Drawing::Size(191, 62);
this->button6->TabIndex = 9;
this->button6->Text = L"Радіозв'язок";
this->button6->UseVisualStyleBackColor = false;
this->button6->Click += gcnew System::EventHandler(this, &Form1::button6_Click);
//
// button7
//
this->button7->BackColor = System::Drawing::Color::LightBlue;
this->button7->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button7->Location = System::Drawing::Point(426, 115);
this->button7->Margin = System::Windows::Forms::Padding(1);
this->button7->Name = L"button7";
this->button7->Size = System::Drawing::Size(191, 62);
this->button7->TabIndex = 10;
this->button7->Text = L"Налаштування підключення";
this->button7->UseVisualStyleBackColor = false;
this->button7->Click += gcnew System::EventHandler(this, &Form1::button7_Click);
//
// openFileDialog1

```

```

//
this->openFileDialog1->FileName = L"FileDialog";
this->openFileDialog1->FileOk += gcnew System::ComponentModel::CancelEventHandler(this,
&Form1::openFileDialog1_FileOk);
//
// menuStrip1
//
this->menuStrip1->BackColor = System::Drawing::Color::CadetBlue;
this->menuStrip1->BackgroundImageLayout = System::Windows::Forms::ImageLayout::None;
this->menuStrip1->GripStyle = System::Windows::Forms::ToolStripGripStyle::Visible;
this->menuStrip1->ImageScalingSize = System::Drawing::Size(14, 14);
this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {
    this->файлToolStripMenuItem,
    this->прошитиToolStripMenuItem, this->довідкаToolStripMenuItem
});
this->menuStrip1->Location = System::Drawing::Point(0, 0);
this->menuStrip1->Name = L"menuStrip1";
this->menuStrip1->Size = System::Drawing::Size(624, 24);
this->menuStrip1->TabIndex = 17;
this->menuStrip1->Text = L"menuStrip1";
this->menuStrip1->ItemClicked += gcnew System::Windows::Forms::ToolStripItemClickedEventHandler(this,
&Form1::menuStrip1_ItemClicked);
//
// файлToolStripMenuItem
//
this->файлToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^ >(4) {
    this->новийToolStripMenuItem,
    this->відкритиToolStripMenuItem, this->зберегтиToolStripMenuItem, this-
>зберегтиЯкToolStripMenuItem
});
this->файлToolStripMenuItem->Name = L"файлToolStripMenuItem";
this->файлToolStripMenuItem->Size = System::Drawing::Size(48, 20);
this->файлToolStripMenuItem->Text = L"Файл";
//
// новийToolStripMenuItem
//
this->новийToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"новийToolStripMenuItem.Image")));
this->новийToolStripMenuItem->Name = L"новийToolStripMenuItem";
this->новийToolStripMenuItem->Size = System::Drawing::Size(139, 22);
this->новийToolStripMenuItem->Text = L"Новий ";
this->новийToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::новийToolStripMenuItem_Click);
//
// відкритиToolStripMenuItem
//
this->відкритиToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"відкритиToolStripMenuItem.Image")));
this->відкритиToolStripMenuItem->Name = L"відкритиToolStripMenuItem";
this->відкритиToolStripMenuItem->Size = System::Drawing::Size(139, 22);
this->відкритиToolStripMenuItem->Text = L"Відкрити";
this->відкритиToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::відкритиToolStripMenuItem_Click);
//
// зберегтиToolStripMenuItem
//
this->зберегтиToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"зберегтиToolStripMenuItem.Image")));
this->зберегтиToolStripMenuItem->Name = L"зберегтиToolStripMenuItem";
this->зберегтиToolStripMenuItem->Size = System::Drawing::Size(139, 22);
this->зберегтиToolStripMenuItem->Text = L"Зберегти";
this->зберегтиToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::зберегтиToolStripMenuItem_Click);
//
// зберегтиЯкToolStripMenuItem
//
this->зберегтиЯкToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"зберегтиЯкToolStripMenuItem.Image")));
this->зберегтиЯкToolStripMenuItem->Name = L"зберегтиЯкToolStripMenuItem";
this->зберегтиЯкToolStripMenuItem->Size = System::Drawing::Size(139, 22);
this->зберегтиЯкToolStripMenuItem->Text = L"Зберегти як";
this->зберегтиЯкToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::зберегтиЯкToolStripMenuItem_Click);
//
// прошитиToolStripMenuItem
//

```

```

        this->прошитиToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^ >(1) { this->налаштуванняCOMPORTToolStripMenuItem });
        this->прошитиToolStripMenuItem->Name = L"прошитиToolStripMenuItem";
        this->прошитиToolStripMenuItem->Size = System::Drawing::Size(47, 20);
        this->прошитиToolStripMenuItem->Text = L"COM";
        //
        // налаштуванняCOMPORTToolStripMenuItem
        //
        this->налаштуванняCOMPORTToolStripMenuItem->Image =
(cli::safe_cast<System::Drawing::Image^>(resources->GetObject(L"налаштуванняCOMPORTToolStripMenuItem.Image")));
        this->налаштуванняCOMPORTToolStripMenuItem->Name =
L"налаштуванняCOMPORTToolStripMenuItem";
        this->налаштуванняCOMPORTToolStripMenuItem->Size = System::Drawing::Size(220, 22);
        this->налаштуванняCOMPORTToolStripMenuItem->Text = L"Налаштування COM-PORT";
        this->налаштуванняCOMPORTToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::налаштуванняCOMPORTToolStripMenuItem_Click);
        //
        // довідкаToolStripMenuItem
        //
        this->довідкаToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^ >(5) {
            this->технічніВимогиToolStripMenuItem,
            this->cOMPORTToolStripMenuItem, this->застереженняToolStripMenuItem, this-
>розробникToolStripMenuItem, this->aboutHacerioPilotToolStripMenuItem
        });
        this->довідкаToolStripMenuItem->Name = L"довідкаToolStripMenuItem";
        this->довідкаToolStripMenuItem->Size = System::Drawing::Size(61, 20);
        this->довідкаToolStripMenuItem->Text = L"Довідка";
        //
        // технічніВимогиToolStripMenuItem
        //
        this->технічніВимогиToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"технічніВимогиToolStripMenuItem.Image")));
        this->технічніВимогиToolStripMenuItem->Name = L"технічніВимогиToolStripMenuItem";
        this->технічніВимогиToolStripMenuItem->Size = System::Drawing::Size(175, 22);
        this->технічніВимогиToolStripMenuItem->Text = L"Технічні вимоги";
        this->технічніВимогиToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::технічніВимогиToolStripMenuItem_Click);
        //
        // cOMPORTToolStripMenuItem
        //
        this->cOMPORTToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"cOMPORTToolStripMenuItem.Image")));
        this->cOMPORTToolStripMenuItem->Name = L"cOMPORTToolStripMenuItem";
        this->cOMPORTToolStripMenuItem->Size = System::Drawing::Size(175, 22);
        this->cOMPORTToolStripMenuItem->Text = L"COM-PORT";
        this->cOMPORTToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::cOMPORTToolStripMenuItem_Click);
        //
        // застереженняToolStripMenuItem
        //
        this->застереженняToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"застереженняToolStripMenuItem.Image")));
        this->застереженняToolStripMenuItem->Name = L"застереженняToolStripMenuItem";
        this->застереженняToolStripMenuItem->Size = System::Drawing::Size(175, 22);
        this->застереженняToolStripMenuItem->Text = L"Застереження";
        this->застереженняToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::застереженняToolStripMenuItem_Click);
        //
        // розробникToolStripMenuItem
        //
        this->розробникToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^ >(2) {
            this->проРозробникаToolStripMenuItem,
            this->githubToolStripMenuItem
        });
        this->розробникToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"розробникToolStripMenuItem.Image")));
        this->розробникToolStripMenuItem->Name = L"розробникToolStripMenuItem";
        this->розробникToolStripMenuItem->Size = System::Drawing::Size(175, 22);
        this->розробникToolStripMenuItem->Text = L"Розробник";
        //
        // проРозробникаToolStripMenuItem
        //
        this->проРозробникаToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"проРозробникаToolStripMenuItem.Image")));
        this->проРозробникаToolStripMenuItem->Name = L"проРозробникаToolStripMenuItem";
        this->проРозробникаToolStripMenuItem->Size = System::Drawing::Size(166, 22);

```

```

        this->проРозробникаToolStripMenuItem->Text = L"Про розробника";
        this->проРозробникаToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::проРозробникаToolStripMenuItem_Click);
        //
        // githubToolStripMenuItem
        //
        this->githubToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"githubToolStripMenuItem.Image"))));
        this->githubToolStripMenuItem->Name = L"githubToolStripMenuItem";
        this->githubToolStripMenuItem->Size = System::Drawing::Size(166, 22);
        this->githubToolStripMenuItem->Text = L"Github";
        this->githubToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::githubToolStripMenuItem_Click);
        //
        // aboutHacerioPilotToolStripMenuItem
        //
        this->aboutHacerioPilotToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"aboutHacerioPilotToolStripMenuItem.Image"))));
        this->aboutHacerioPilotToolStripMenuItem->Name = L"aboutHacerioPilotToolStripMenuItem";
        this->aboutHacerioPilotToolStripMenuItem->Size = System::Drawing::Size(175, 22);
        this->aboutHacerioPilotToolStripMenuItem->Text = L"About HacerioPilot";
        this->aboutHacerioPilotToolStripMenuItem->Click += gcnew System::EventHandler(this,
&Form1::aboutHacerioPilotToolStripMenuItem_Click);
        //
        // button5
        //
        this->button5->BackColor = System::Drawing::Color::LightBlue;
        this->button5->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
        this->button5->Location = System::Drawing::Point(513, 260);
        this->button5->Name = L"button5";
        this->button5->Size = System::Drawing::Size(104, 45);
        this->button5->TabIndex = 18;
        this->button5->Text = L"Очистити дані";
        this->button5->UseVisualStyleBackColor = false;
        this->button5->Click += gcnew System::EventHandler(this, &Form1::button5_Click);
        //
        // panel1
        //
        this->panel1->BackColor = System::Drawing::Color::LightBlue;
        this->panel1->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;
        this->panel1->Controls->Add(this->label6);
        this->panel1->Controls->Add(this->label2);
        this->panel1->ForeColor = System::Drawing::SystemColors::ControlText;
        this->panel1->Location = System::Drawing::Point(0, 343);
        this->panel1->Name = L"panel1";
        this->panel1->Size = System::Drawing::Size(624, 29);
        this->panel1->TabIndex = 19;
        //
        // label6
        //
        this->label6->AutoSize = true;
        this->label6->Location = System::Drawing::Point(425, 6);
        this->label6->Name = L"label6";
        this->label6->Size = System::Drawing::Size(191, 13);
        this->label6->TabIndex = 21;
        this->label6->Text = L"Лише для цивільного використання!";
        //
        // label2
        //
        this->label2->AutoSize = true;
        this->label2->Location = System::Drawing::Point(3, -1);
        this->label2->Name = L"label2";
        this->label2->Size = System::Drawing::Size(210, 26);
        this->label2->TabIndex = 20;
        this->label2->Text = L"© 2024 Hacerio. All rights reserved.\r\nCompatible with HacerioPilot-like firmwares"
        L"";
        this->label2->DoubleClick += gcnew System::EventHandler(this, &Form1::label2_DoubleClick);
        //
        // label4
        //
        this->label4->AutoSize = true;
        this->label4->BackColor = System::Drawing::Color::Transparent;
        this->label4->Location = System::Drawing::Point(12, 327);
        this->label4->Name = L"label4";
        this->label4->Size = System::Drawing::Size(64, 13);
        this->label4->TabIndex = 20;
        this->label4->Text = L"Opened file:";
        //

```



```

// label5
//
this->label5->AutoSize = true;
this->label5->BackColor = System::Drawing::Color::Transparent;
this->label5->Location = System::Drawing::Point(82, 327);
this->label5->Name = L"label5";
this->label5->Size = System::Drawing::Size(35, 13);
this->label5->TabIndex = 21;
this->label5->Text = L"label5";
//
// Form1
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->BackColor = System::Drawing::Color::DeepSkyBlue;
this->ClientSize = System::Drawing::Size(624, 372);
this->Controls->Add(this->label5);
this->Controls->Add(this->label4);
this->Controls->Add(this->panel1);
this->Controls->Add(this->button5);
this->Controls->Add(this->button7);
this->Controls->Add(this->button6);
this->Controls->Add(this->label3);
this->Controls->Add(this->button4);
this->Controls->Add(this->button3);
this->Controls->Add(this->button2);
this->Controls->Add(this->button1);
this->Controls->Add(this->label1);
this->Controls->Add(this->menuStrip1);
this->ForeColor = System::Drawing::SystemColors::ControlText;
this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;
this->Icon = (cli::safe_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));
this->MainMenuStrip = this->menuStrip1;
this->Margin = System::Windows::Forms::Padding(1);
this->MaximizeBox = false;
this->MinimizeBox = false;
this->Name = L"Form1";
this->Text = L"HacerioPilot";
this->TopMost = true;
this->Load += gcnew System::EventHandler(this, &Form1::Form1_Load);
this->menuStrip1->ResumeLayout(false);
this->menuStrip1->PerformLayout();
this->panel1->ResumeLayout(false);
this->panel1->PerformLayout();
this->ResumeLayout(false);
this->PerformLayout();
}

#pragma endregion
#pragma region MainMenu
#pragma endregion
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e);
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void listBox2_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void button4_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void button9_Click(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void openFileDialog1_FileOk(System::Object^ sender, System::ComponentModel::CancelEventArgs^ e) {
}
private: System::Void відкритиToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void зберегтиЯкToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void button6_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void button7_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void новийToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void налаштуванняCOMPORTToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void button5_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void menuStrip1_ItemClicked(System::Object^ sender, System::Windows::Forms::ToolStripItemClickedEventArgs^ e) {
}
private: System::Void зберегтиToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void технічніВимогиToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e) {
    MessageBox::Show("БПЛА повинен бути обладнаним радіомодулем NRF24L01, контролером двигуна(при використанні ДВЗ сервоприводом). Сервоприводами рульових поверхонь. Сумісно з Arduino UNO, NANO, MEGA, LEONARDO сімейство контролерів AVR MEGA. Потенційно може працювати з копіями(не гарантується) мікроконтролерів AVR MEGA. AVR TINY не підтримується");
}

```

```

        MessageBox::Show("Не рекомендується підключення більше 2-х сервоприводів до пінів DC-DC перетворювача встановленого в контролер двигуна. Використовуйте окремо DC-DC перетворювач для підключення сервоприводів");
        MessageBox::Show("Використовуйте спеціалізований DC-DC перетворювач для NRF24L01, а не штатний ARDUINO");
        MessageBox::Show("Не рекомендується підключення живлення до ARDUINO напряму з акумулятора. Натомість використайте вихід DC-DC перетворювача з контролера двигуна. Це може запобігти не одночасного запуску");
    }
    private: System::Void cOMPORTToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e) {
        MessageBox::Show("Підтримується використання як віртуального послідовного порта на платах ARDUINO, використання встроєного в материнську плату, BLUETOOTH-порта та використання перехідників USB-COM");
    }
    private: System::Void githubToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e);
    private: System::Void проРозробникаToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e) {
        MessageBox::Show("Розробник Віктор hacerio Боднарчук. Email: hacerio@dnmx.org");
    }
    private: System::Void label2_DoubleClick(System::Object^ sender, System::EventArgs^ e) {
        MessageBox::Show("Ніколи не використовуйте HacerioPilot для військових БПЛА та для ведення війни!", "Застереження");
    }
    private: System::Void застереженняToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e) {
        MessageBox::Show("Забороняється використовувати HacerioPilot для військових цілей. Не намагайтесь використати HacerioPilot для ведення війни, це небезпечно для здоров'я та життя");
        MessageBox::Show("HacerioPilot не пристосований для військових цілей");
        MessageBox::Show("HacerioPilot дозволяється для використання в цивільних БПЛА");
        MessageBox::Show("Завжди дотримуйтесь авіаційних правил вашого регіону");
        MessageBox::Show("Не запускайте БПЛА в зону де працює глушіння сигналу, БПЛА при втраті сигналу падає!");
        MessageBox::Show("Будьте особливо обережні з гвинтом БПЛА. Гвинт крутиться на високих оборотах і може нанести травми");
        MessageBox::Show("Ніколи не спрямовуйте БПЛА в сторону людей, тварин, будівель, транспортних засобів, літальних апаратів. Ви можете нанести травми і збитки!");
        MessageBox::Show("Дотримуйтесь правил ТБ при роботі з Li-PO акумуляторами. При неправильному використанні можуть загорітись!");
        MessageBox::Show("НІКОЛИ НЕ НАТИСКАЙТЕ КНОПКУ ПЕРЕЗАВАНТАЖЕННЯ!!! ЦЕ ВИКЛИЧЕ ПОВТОРНУ ІНІЦІАЛІЗАЦІЮ ДВИГУНА І ОБОРОТИ ВИЙДУТЬ НА МАКСИМАЛЬНІ! Є РИЗИК ТРАВМ. ОБМЕЖЕННЯ ОБОРОТІВ НЕ ВПЛИВАЄ НА ОБОРОТИ ЯКІ МОЖУТЬ БУТИ ДОСЯГНУТІ ПРИ НЕ ПРАВИЛЬНІЙ ІНІЦІАЛІЗАЦІЇ!!!", "Особливе застереження!");
        MessageBox::Show("Не перенавантажуйте сервоприводи. Це несе ризик як згорання самих сервоприводів і втрату керування");
        MessageBox::Show("Дотримуйтесь вимог стосовно потужності радіозв'язку вашого регіону. Потужність можна змінити");
        MessageBox::Show("Дозволяється створювати прошивки на основі HacerioPilot, а також прошивки сумісні з ПЗ HacerioPilot. Єдина умова - вказувати в сирцевому коді, або іншими способами 'Powered by HacerioPilot'");
        MessageBox::Show("Насолоджуйтесь польотом");
    }
    private: System::Void aboutHacerioPilotToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e) {
        MessageBox::Show("HacerioPilot - система, яка складається з Програми і Прошивок. HacerioPilot 0.9 зроблена в якості дипломної роботи Боднарчука Віктора(Hacerio). Орієнтована на сам перед на авіамоделістів та аматорів. Не для військового, або воєнного призначення");
    }
};
}

```

## otherFeathuresForm.h

```
#pragma once
```

```

namespace HacerioPilot {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Summary for otherFeathuresForm
    /// </summary>
    public ref class otherFeathuresForm : public System::Windows::Forms::Form
    {
    public:
        otherFeathuresForm(void)
        {
            InitializeComponent();
            //
            //TODO: Add the constructor code here
            //
        }

    protected:
        /// <summary>
        /// Clean up any resources being used.

```

```

    /// </summary>
    ~otherFeaturesForm()
    {
        if (components)
        {
            delete components;
        }
    }

private: System::Windows::Forms::Label^ label1;
protected:
private: System::Windows::Forms::CheckBox^ checkBox1;

private: System::Windows::Forms::Label^ label2;
private: System::Windows::Forms::CheckBox^ checkBox2;

private: System::Windows::Forms::Button^ button1;
private: System::Windows::Forms::Label^ label3;
private: System::Windows::Forms::Label^ label4;
private: System::Windows::Forms::TextBox^ textBox1;
private: System::Windows::Forms::TextBox^ textBox2;
private: System::Windows::Forms::Button^ button2;
private: System::Windows::Forms::Label^ label5;
private: System::Windows::Forms::Label^ label6;

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    void InitializeComponent(void)
    {
        System::ComponentModel::ComponentResourceManager^ resources = (gcnew
System::ComponentModel::ComponentResourceManager(otherFeaturesForm::typeid));
        this->label1 = (gcnew System::Windows::Forms::Label());
        this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());
        this->label2 = (gcnew System::Windows::Forms::Label());
        this->checkBox2 = (gcnew System::Windows::Forms::CheckBox());
        this->button1 = (gcnew System::Windows::Forms::Button());
        this->label3 = (gcnew System::Windows::Forms::Label());
        this->label4 = (gcnew System::Windows::Forms::Label());
        this->textBox1 = (gcnew System::Windows::Forms::TextBox());
        this->textBox2 = (gcnew System::Windows::Forms::TextBox());
        this->button2 = (gcnew System::Windows::Forms::Button());
        this->label5 = (gcnew System::Windows::Forms::Label());
        this->label6 = (gcnew System::Windows::Forms::Label());
        this->SuspendLayout();
        //
        // label1
        //
        this->label1->AutoSize = true;
        this->label1->BackColor = System::Drawing::Color::Transparent;
        this->label1->Location = System::Drawing::Point(12, 31);
        this->label1->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
        this->label1->Name = L"label1";
        this->label1->Size = System::Drawing::Size(114, 13);
        this->label1->TabIndex = 0;
        this->label1->Text = L"Додаткова функція 1";
        //
        // checkBox1
        //
        this->checkBox1->AutoSize = true;
        this->checkBox1->BackColor = System::Drawing::Color::Transparent;
        this->checkBox1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
        this->checkBox1->Location = System::Drawing::Point(13, 56);
        this->checkBox1->Margin = System::Windows::Forms::Padding(1);
        this->checkBox1->Name = L"checkBox1";
        this->checkBox1->Size = System::Drawing::Size(117, 17);
        this->checkBox1->TabIndex = 1;
        this->checkBox1->Text = L"Використовується";
        this->checkBox1->UseVisualStyleBackColor = false;
    }

```

```

//
// label2
//
this->label2->AutoSize = true;
this->label2->BackColor = System::Drawing::Color::Transparent;
this->label2->Location = System::Drawing::Point(277, 31);
this->label2->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(114, 13);
this->label2->TabIndex = 3;
this->label2->Text = L"Додаткова функція 2";
//
// checkBox2
//
this->checkBox2->AutoSize = true;
this->checkBox2->BackColor = System::Drawing::Color::Transparent;
this->checkBox2->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->checkBox2->Location = System::Drawing::Point(280, 56);
this->checkBox2->Margin = System::Windows::Forms::Padding(1);
this->checkBox2->Name = L"checkBox2";
this->checkBox2->Size = System::Drawing::Size(117, 17);
this->checkBox2->TabIndex = 4;
this->checkBox2->Text = L"Використовується";
this->checkBox2->UseVisualStyleBackColor = false;
//
// button1
//
this->button1->BackColor = System::Drawing::Color::LightBlue;
this->button1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button1->Location = System::Drawing::Point(13, 116);
this->button1->Margin = System::Windows::Forms::Padding(1);
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(117, 37);
this->button1->TabIndex = 6;
this->button1->Text = L"Зберегти";
this->button1->UseVisualStyleBackColor = false;
this->button1->Click += gcnew System::EventHandler(this, &otherFeaturesForm::button1_Click);
//
// label3
//
this->label3->AutoSize = true;
this->label3->BackColor = System::Drawing::Color::Transparent;
this->label3->Location = System::Drawing::Point(12, 84);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(23, 13);
this->label3->TabIndex = 7;
this->label3->Text = L"Піи";
//
// label4
//
this->label4->AutoSize = true;
this->label4->BackColor = System::Drawing::Color::Transparent;
this->label4->Location = System::Drawing::Point(277, 84);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(23, 13);
this->label4->TabIndex = 8;
this->label4->Text = L"Піи";
//
// textBox1
//
this->textBox1->Location = System::Drawing::Point(59, 81);
this->textBox1->MaxLength = 2;
this->textBox1->Name = L"textBox1";
this->textBox1->Size = System::Drawing::Size(71, 20);
this->textBox1->TabIndex = 9;
//
// textBox2
//
this->textBox2->Location = System::Drawing::Point(326, 81);
this->textBox2->MaxLength = 2;
this->textBox2->Name = L"textBox2";
this->textBox2->Size = System::Drawing::Size(71, 20);
this->textBox2->TabIndex = 10;
//
// button2
//
this->button2->BackColor = System::Drawing::Color::LightBlue;
this->button2->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

```

```

this->button2->Location = System::Drawing::Point(280, 116);
this->button2->Margin = System::Windows::Forms::Padding(1);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(117, 37);
this->button2->TabIndex = 11;
this->button2->Text = L"Вийти";
this->button2->UseVisualStyleBackColor = false;
this->button2->Click += gcnew System::EventHandler(this, &otherFeaturesForm::button2_Click);
//
// label5
//
this->label5->AutoSize = true;
this->label5->BackColor = System::Drawing::Color::Transparent;
this->label5->Location = System::Drawing::Point(136, 88);
this->label5->Name = L"label5";
this->label5->Size = System::Drawing::Size(35, 13);
this->label5->TabIndex = 12;
this->label5->Text = L"label5";
//
// label6
//
this->label6->AutoSize = true;
this->label6->BackColor = System::Drawing::Color::Transparent;
this->label6->Location = System::Drawing::Point(403, 88);
this->label6->Name = L"label6";
this->label6->Size = System::Drawing::Size(35, 13);
this->label6->TabIndex = 13;
this->label6->Text = L"label6";
//
// otherFeaturesForm
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->BackColor = System::Drawing::SystemColors::ScrollBar;
this->ClientSize = System::Drawing::Size(455, 176);
this->ControlBox = false;
this->Controls->Add(this->label6);
this->Controls->Add(this->label5);
this->Controls->Add(this->button2);
this->Controls->Add(this->textBox2);
this->Controls->Add(this->textBox1);
this->Controls->Add(this->label4);
this->Controls->Add(this->label3);
this->Controls->Add(this->button1);
this->Controls->Add(this->checkBox2);
this->Controls->Add(this->label2);
this->Controls->Add(this->checkBox1);
this->Controls->Add(this->label1);
this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
this->Icon = (cli::safe_cast<System::Drawing::Icon^>)(resources->GetObject(L"$this.Icon"));
this->Margin = System::Windows::Forms::Padding(1);
this->Name = L"otherFeaturesForm";
this->Text = L"Додаткові Функції";
this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this,
&otherFeaturesForm::otherFeaturesForm_FormClosing);
this->Load += gcnew System::EventHandler(this, &otherFeaturesForm::otherFeaturesForm_Load);
this->ResumeLayout(false);
this->PerformLayout();

}

#pragma endregion
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void otherFeaturesForm_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e) {
    Owner->Show();
}
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
    Owner->Show();
    this->Hide();
}
private: System::Void otherFeaturesForm_Load(System::Object^ sender, System::EventArgs^ e);
};
}

```

## Radio1Form.h

```
#pragma once
```

```
namespace HacerioPilot {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Summary for radio1Form
    /// </summary>
    public ref class radio1Form : public System::Windows::Forms::Form
    {
    public:
        radio1Form(void)
        {
            InitializeComponent();
            //
            //TODO: Add the constructor code here
            //
        }

    protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ~radio1Form()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::Label^ label1;

    private: System::Windows::Forms::Label^ label2;

    private: System::Windows::Forms::Button^ button1;
    private: System::Windows::Forms::ComboBox^ comboBox1;
    private: System::Windows::Forms::ComboBox^ comboBox2;
    private: System::Windows::Forms::Label^ label3;
    private: System::Windows::Forms::Label^ label4;
    private: System::Windows::Forms::Button^ button2;

    protected:

    private:
        /// <summary>
        /// Required designer variable.
        /// </summary>
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        void InitializeComponent(void)
        {
            System::ComponentModel::ComponentResourceManager^ resources = (gcnew
System::ComponentModel::ComponentResourceManager(radio1Form::typeid));
            this->label1 = (gcnew System::Windows::Forms::Label());
            this->label2 = (gcnew System::Windows::Forms::Label());
            this->button1 = (gcnew System::Windows::Forms::Button());
            this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());
            this->comboBox2 = (gcnew System::Windows::Forms::ComboBox());
            this->label3 = (gcnew System::Windows::Forms::Label());
            this->label4 = (gcnew System::Windows::Forms::Label());
            this->button2 = (gcnew System::Windows::Forms::Button());
            this->SuspendLayout();
            //
            // label1
```

```

//
this->label1->AutoSize = true;
this->label1->BackColor = System::Drawing::Color::Transparent;
this->label1->Location = System::Drawing::Point(19, 43);
this->label1->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(99, 13);
this->label1->TabIndex = 0;
this->label1->Text = L"Канал для зв'язку";
//
// label2
//
this->label2->AutoSize = true;
this->label2->BackColor = System::Drawing::Color::Transparent;
this->label2->Location = System::Drawing::Point(19, 69);
this->label2->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(166, 13);
this->label2->TabIndex = 2;
this->label2->Text = L"Потужність підсилення сигналу";
//
// button1
//
this->button1->BackColor = System::Drawing::Color::LightBlue;
this->button1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button1->Location = System::Drawing::Point(22, 113);
this->button1->Margin = System::Windows::Forms::Padding(1);
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(163, 38);
this->button1->TabIndex = 4;
this->button1->Text = L"Зберегти";
this->button1->UseVisualStyleBackColor = false;
this->button1->Click += gcnew System::EventHandler(this, &radio1Form::button1_Click);
//
// comboBox1
//
this->comboBox1->FormattingEnabled = true;
this->comboBox1->Location = System::Drawing::Point(200, 40);
this->comboBox1->Name = L"comboBox1";
this->comboBox1->Size = System::Drawing::Size(121, 21);
this->comboBox1->TabIndex = 5;
//
// comboBox2
//
this->comboBox2->FormattingEnabled = true;
this->comboBox2->Location = System::Drawing::Point(200, 66);
this->comboBox2->Name = L"comboBox2";
this->comboBox2->Size = System::Drawing::Size(121, 21);
this->comboBox2->TabIndex = 6;
//
// label3
//
this->label3->AutoSize = true;
this->label3->BackColor = System::Drawing::Color::Transparent;
this->label3->Location = System::Drawing::Point(345, 43);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(35, 13);
this->label3->TabIndex = 7;
this->label3->Text = L"label3";
//
// label4
//
this->label4->AutoSize = true;
this->label4->BackColor = System::Drawing::Color::Transparent;
this->label4->Location = System::Drawing::Point(345, 69);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(35, 13);
this->label4->TabIndex = 8;
this->label4->Text = L"label4";
//
// button2
//
this->button2->BackColor = System::Drawing::Color::LightBlue;
this->button2->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;
this->button2->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button2->Location = System::Drawing::Point(217, 113);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(163, 38);

```

```

        this->button2->TabIndex = 9;
        this->button2->Text = L"Вийти";
        this->button2->UseVisualStyleBackColor = false;
        this->button2->Click += gcnew System::EventHandler(this, &radio1Form::button2_Click);
        //
        // radio1Form
        //
        this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
        this->BackColor = System::Drawing::SystemColors::ScrollBar;
        this->ClientSize = System::Drawing::Size(392, 163);
        this->ControlBox = false;
        this->Controls->Add(this->button2);
        this->Controls->Add(this->label4);
        this->Controls->Add(this->label3);
        this->Controls->Add(this->comboBox2);
        this->Controls->Add(this->comboBox1);
        this->Controls->Add(this->button1);
        this->Controls->Add(this->label2);
        this->Controls->Add(this->label1);
        this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
        this->Icon = (cli::safe_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));
        this->Margin = System::Windows::Forms::Padding(1);
        this->Name = L"radio1Form";
        this->Text = L"Радіозв'язок";
        this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this,
&radio1Form::radio1Form_FormClosing);
        this->Load += gcnew System::EventHandler(this, &radio1Form::radio1Form_Load);
        this->ResumeLayout(false);
        this->PerformLayout();

    }
#pragma endregion
    private: System::Void radio1Form_Load(System::Object^ sender, System::EventArgs^ e);
    private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void radio1Form_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e) {
    Owner->Show();
}
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
    Owner->Show();
    this->Hide();
}
};
}

```

## Radio2Form.h

```
#pragma once
```

```

namespace HacerioPilot {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Summary for radio2Form
    /// </summary>
    public ref class radio2Form : public System::Windows::Forms::Form
    {
    public:
        radio2Form(void)
        {
            InitializeComponent();
            //
            //TODO: Add the constructor code here
            //
        }

    protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ~radio2Form()
        {

```



```

        if (components)
        {
            delete components;
        }
    }
private: System::Windows::Forms::Button^ button1;
protected:

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Label^ label1;
private: System::Windows::Forms::Button^ button2;
private: System::Windows::Forms::ComboBox^ comboBox1;
private: System::Windows::Forms::ComboBox^ comboBox2;
private: System::Windows::Forms::Label^ label3;
private: System::Windows::Forms::Label^ label4;

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    void InitializeComponent(void)
    {
        System::ComponentModel::ComponentResourceManager^ resources = (gcnew
System::ComponentModel::ComponentResourceManager(radio2Form::typeid));
        this->button1 = (gcnew System::Windows::Forms::Button());
        this->label2 = (gcnew System::Windows::Forms::Label());
        this->label1 = (gcnew System::Windows::Forms::Label());
        this->button2 = (gcnew System::Windows::Forms::Button());
        this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());
        this->comboBox2 = (gcnew System::Windows::Forms::ComboBox());
        this->label3 = (gcnew System::Windows::Forms::Label());
        this->label4 = (gcnew System::Windows::Forms::Label());
        this->SuspendLayout();
        //
        // button1
        //
        this->button1->BackColor = System::Drawing::Color::LightBlue;
        this->button1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
        this->button1->Location = System::Drawing::Point(13, 112);
        this->button1->Margin = System::Windows::Forms::Padding(1);
        this->button1->Name = L"button1";
        this->button1->Size = System::Drawing::Size(163, 38);
        this->button1->TabIndex = 9;
        this->button1->Text = L"Зберегти";
        this->button1->UseVisualStyleBackColor = false;
        this->button1->Click += gcnew System::EventHandler(this, &radio2Form::button1_Click);
        //
        // label2
        //
        this->label2->AutoSize = true;
        this->label2->BackColor = System::Drawing::Color::Transparent;
        this->label2->Location = System::Drawing::Point(10, 70);
        this->label2->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
        this->label2->Name = L"label2";
        this->label2->Size = System::Drawing::Size(166, 13);
        this->label2->TabIndex = 7;
        this->label2->Text = L"Потужність підсилення сигналу";
        //
        // label1
        //
        this->label1->AutoSize = true;
        this->label1->BackColor = System::Drawing::Color::Transparent;
        this->label1->Location = System::Drawing::Point(10, 43);
        this->label1->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
        this->label1->Name = L"label1";
        this->label1->Size = System::Drawing::Size(99, 13);
        this->label1->TabIndex = 5;
        this->label1->Text = L"Канал для зв'язку";
    }

```

```

//
// button2
//
this->button2->BackColor = System::Drawing::Color::LightBlue;
this->button2->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;
this->button2->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button2->Location = System::Drawing::Point(196, 112);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(163, 38);
this->button2->TabIndex = 10;
this->button2->Text = L"Вийти";
this->button2->UseVisualStyleBackColor = false;
this->button2->Click += gcnew System::EventHandler(this, &radio2Form::button2_Click);
//
// comboBox1
//
this->comboBox1->FormattingEnabled = true;
this->comboBox1->Location = System::Drawing::Point(196, 40);
this->comboBox1->Name = L"comboBox1";
this->comboBox1->Size = System::Drawing::Size(105, 21);
this->comboBox1->TabIndex = 11;
//
// comboBox2
//
this->comboBox2->FormattingEnabled = true;
this->comboBox2->Location = System::Drawing::Point(196, 67);
this->comboBox2->Name = L"comboBox2";
this->comboBox2->Size = System::Drawing::Size(105, 21);
this->comboBox2->TabIndex = 12;
//
// label3
//
this->label3->AutoSize = true;
this->label3->BackColor = System::Drawing::Color::Transparent;
this->label3->Location = System::Drawing::Point(324, 43);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(35, 13);
this->label3->TabIndex = 13;
this->label3->Text = L"label3";
//
// label4
//
this->label4->AutoSize = true;
this->label4->BackColor = System::Drawing::Color::Transparent;
this->label4->Location = System::Drawing::Point(324, 70);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(35, 13);
this->label4->TabIndex = 14;
this->label4->Text = L"label4";
//
// radio2Form
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->BackColor = System::Drawing::SystemColors::ScrollBar;
this->ClientSize = System::Drawing::Size(371, 162);
this->ControlBox = false;
this->Controls->Add(this->label4);
this->Controls->Add(this->label3);
this->Controls->Add(this->comboBox2);
this->Controls->Add(this->comboBox1);
this->Controls->Add(this->button2);
this->Controls->Add(this->button1);
this->Controls->Add(this->label2);
this->Controls->Add(this->label1);
this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
this->Icon = (cli::safe_cast<System::Drawing::Icon^>)(resources->GetObject(L"$this.Icon"));
this->Name = L"radio2Form";
this->Text = L"Радіозв'язок Пульта";
this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this,
&radio2Form::radio2Form_FormClosing);
this->Load += gcnew System::EventHandler(this, &radio2Form::radio2Form_Load);
this->ResumeLayout(false);
this->PerformLayout();

}

#pragma endregion
private: System::Void radio2Form_Load(System::Object^ sender, System::EventArgs^ e);

```

```

        private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void radio2Form_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e) {
    Owner->Show();
}
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
    Owner->Show();
    this->Hide();
}
};
}

```

## ServoForm.h

```
#pragma once
```

```

namespace HacerioPilot {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Summary for servoForm
    /// </summary>
    public ref class servoForm : public System::Windows::Forms::Form
    {
    public:
        servoForm(void)
        {
            InitializeComponent();
            //
            //TODO: Add the constructor code here
            //
        }

    protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ~servoForm()
        {
            if (components)
            {
                delete components;
            }
        }

private:
        System::Windows::Forms::TextBox^ textBox4;
        System::Windows::Forms::Label^ label6;
        System::Windows::Forms::Panel^ panel1;
        System::Windows::Forms::Label^ label7;
        System::Windows::Forms::Label^ label8;
        System::Windows::Forms::Label^ label11;
        System::Windows::Forms::TextBox^ textBox5;

        System::Windows::Forms::TextBox^ textBox7;
        System::Windows::Forms::TextBox^ textBox6;
        System::Windows::Forms::Label^ label10;
        System::Windows::Forms::Panel^ panel2;
        System::Windows::Forms::Label^ label1;
        System::Windows::Forms::Label^ label2;
        System::Windows::Forms::TextBox^ textBox1;
        System::Windows::Forms::Label^ label3;
        System::Windows::Forms::Label^ label4;
        System::Windows::Forms::TextBox^ textBox2;

        System::Windows::Forms::TextBox^ textBox3;
        System::Windows::Forms::TextBox^ textBox8;
        System::Windows::Forms::Label^ label12;

```

```

private: System::Windows::Forms::CheckBox^ checkBox1;
private: System::Windows::Forms::Panel^ panel3;
private: System::Windows::Forms::CheckBox^ checkBox2;
private: System::Windows::Forms::Label^ label13;
private: System::Windows::Forms::Label^ label14;
private: System::Windows::Forms::TextBox^ textBox9;
private: System::Windows::Forms::Label^ label15;
private: System::Windows::Forms::Label^ label16;
private: System::Windows::Forms::TextBox^ textBox10;

```

```

private: System::Windows::Forms::TextBox^ textBox11;
private: System::Windows::Forms::TextBox^ textBox12;
private: System::Windows::Forms::Label^ label18;

```

```

private: System::Windows::Forms::Panel^ panel5;
private: System::Windows::Forms::Label^ label22;
private: System::Windows::Forms::Label^ label25;
private: System::Windows::Forms::TextBox^ textBox15;
private: System::Windows::Forms::Label^ label26;
private: System::Windows::Forms::Label^ label27;
private: System::Windows::Forms::TextBox^ textBox17;

```

```

private: System::Windows::Forms::TextBox^ textBox18;
private: System::Windows::Forms::TextBox^ textBox19;
private: System::Windows::Forms::Label^ label29;

```

```

private: System::Windows::Forms::Button^ button1;

```

```

private: System::Windows::Forms::Button^ button4;
private: System::Windows::Forms::Button^ button5;
private: System::Windows::Forms::Button^ button8;
private: System::Windows::Forms::Button^ button6;

```

```

private: System::Windows::Forms::Button^ button2;
private: System::Windows::Forms::Label^ label41;
private: System::Windows::Forms::Label^ label40;
private: System::Windows::Forms::Label^ label39;
private: System::Windows::Forms::Label^ label38;
private: System::Windows::Forms::Label^ label50;
private: System::Windows::Forms::Label^ label49;
private: System::Windows::Forms::Label^ label48;
private: System::Windows::Forms::Label^ label46;
private: System::Windows::Forms::Label^ label47;
private: System::Windows::Forms::Label^ label45;
private: System::Windows::Forms::Label^ label44;
private: System::Windows::Forms::Label^ label17;

```

```

private: System::Windows::Forms::Label^ label37;
private: System::Windows::Forms::Label^ label36;
private: System::Windows::Forms::Label^ label35;
private: System::Windows::Forms::Label^ label33;

```

```

private: System::IO::Ports::SerialPort^ serialPort1;

```

```

protected:

protected:
private: System::ComponentModel::IContainer^ components;

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    void InitializeComponent(void)
    {
        this->components = (gcnew System::ComponentModel::Container());
        System::ComponentModel::ComponentResourceManager^ resources = (gcnew
System::ComponentModel::ComponentResourceManager(servoForm::typeid));
        this->textBox4 = (gcnew System::Windows::Forms::TextBox());
        this->label6 = (gcnew System::Windows::Forms::Label());
        this->panel1 = (gcnew System::Windows::Forms::Panel());
        this->label41 = (gcnew System::Windows::Forms::Label());
        this->label40 = (gcnew System::Windows::Forms::Label());
        this->label39 = (gcnew System::Windows::Forms::Label());
        this->label38 = (gcnew System::Windows::Forms::Label());
        this->button5 = (gcnew System::Windows::Forms::Button());
        this->label7 = (gcnew System::Windows::Forms::Label());
        this->label8 = (gcnew System::Windows::Forms::Label());
        this->label11 = (gcnew System::Windows::Forms::Label());
        this->textBox5 = (gcnew System::Windows::Forms::TextBox());
        this->textBox7 = (gcnew System::Windows::Forms::TextBox());
        this->textBox6 = (gcnew System::Windows::Forms::TextBox());
        this->label10 = (gcnew System::Windows::Forms::Label());
        this->panel2 = (gcnew System::Windows::Forms::Panel());
        this->label50 = (gcnew System::Windows::Forms::Label());
        this->label49 = (gcnew System::Windows::Forms::Label());
        this->label48 = (gcnew System::Windows::Forms::Label());
        this->button8 = (gcnew System::Windows::Forms::Button());
        this->label46 = (gcnew System::Windows::Forms::Label());
        this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());
        this->label1 = (gcnew System::Windows::Forms::Label());
        this->label2 = (gcnew System::Windows::Forms::Label());
        this->textBox1 = (gcnew System::Windows::Forms::TextBox());
        this->label3 = (gcnew System::Windows::Forms::Label());
        this->label4 = (gcnew System::Windows::Forms::Label());
        this->textBox2 = (gcnew System::Windows::Forms::TextBox());
        this->textBox3 = (gcnew System::Windows::Forms::TextBox());
        this->textBox8 = (gcnew System::Windows::Forms::TextBox());
        this->label12 = (gcnew System::Windows::Forms::Label());
        this->panel3 = (gcnew System::Windows::Forms::Panel());
        this->label47 = (gcnew System::Windows::Forms::Label());
        this->label45 = (gcnew System::Windows::Forms::Label());
        this->label44 = (gcnew System::Windows::Forms::Label());
        this->label17 = (gcnew System::Windows::Forms::Label());
        this->button6 = (gcnew System::Windows::Forms::Button());
        this->checkBox2 = (gcnew System::Windows::Forms::CheckBox());
        this->label13 = (gcnew System::Windows::Forms::Label());
        this->label14 = (gcnew System::Windows::Forms::Label());
        this->textBox9 = (gcnew System::Windows::Forms::TextBox());
        this->label15 = (gcnew System::Windows::Forms::Label());
        this->label16 = (gcnew System::Windows::Forms::Label());
        this->textBox10 = (gcnew System::Windows::Forms::TextBox());
    }

```

```

this->textBox11 = (gcnew System::Windows::Forms::TextBox());
this->textBox12 = (gcnew System::Windows::Forms::TextBox());
this->label18 = (gcnew System::Windows::Forms::Label());
this->panel5 = (gcnew System::Windows::Forms::Panel());
this->label37 = (gcnew System::Windows::Forms::Label());
this->label36 = (gcnew System::Windows::Forms::Label());
this->label35 = (gcnew System::Windows::Forms::Label());
this->label33 = (gcnew System::Windows::Forms::Label());
this->button4 = (gcnew System::Windows::Forms::Button());
this->label22 = (gcnew System::Windows::Forms::Label());
this->label25 = (gcnew System::Windows::Forms::Label());
this->textBox15 = (gcnew System::Windows::Forms::TextBox());
this->label26 = (gcnew System::Windows::Forms::Label());
this->label27 = (gcnew System::Windows::Forms::Label());
this->textBox17 = (gcnew System::Windows::Forms::TextBox());
this->textBox18 = (gcnew System::Windows::Forms::TextBox());
this->textBox19 = (gcnew System::Windows::Forms::TextBox());
this->label29 = (gcnew System::Windows::Forms::Label());
this->button1 = (gcnew System::Windows::Forms::Button());
this->button2 = (gcnew System::Windows::Forms::Button());
this->serialPort1 = (gcnew System::IO::Ports::SerialPort(this->components));
this->panel1->SuspendLayout();
this->panel2->SuspendLayout();
this->panel3->SuspendLayout();
this->panel5->SuspendLayout();
this->SuspendLayout();
//
// textBox4
//
this->textBox4->Location = System::Drawing::Point(82, 60);
this->textBox4->Margin = System::Windows::Forms::Padding(1);
this->textBox4->MaxLength = 3;
this->textBox4->Name = L"textBox4";
this->textBox4->Size = System::Drawing::Size(58, 20);
this->textBox4->TabIndex = 8;
//
// label6
//
this->label6->AutoSize = true;
this->label6->Location = System::Drawing::Point(15, 107);
this->label6->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label6->Name = L"label6";
this->label6->Size = System::Drawing::Size(43, 13);
this->label6->TabIndex = 9;
this->label6->Text = L"Верхняя";
//
// panel1
//
this->panel1->BackColor = System::Drawing::Color::Transparent;
this->panel1->Controls->Add(this->label41);
this->panel1->Controls->Add(this->label40);
this->panel1->Controls->Add(this->label39);
this->panel1->Controls->Add(this->label38);
this->panel1->Controls->Add(this->button5);
this->panel1->Controls->Add(this->label6);
this->panel1->Controls->Add(this->label7);
this->panel1->Controls->Add(this->textBox4);
this->panel1->Controls->Add(this->label8);
this->panel1->Controls->Add(this->label11);
this->panel1->Controls->Add(this->textBox5);
this->panel1->Controls->Add(this->textBox7);
this->panel1->Controls->Add(this->textBox6);
this->panel1->Controls->Add(this->label10);
this->panel1->Location = System::Drawing::Point(14, 36);
this->panel1->Margin = System::Windows::Forms::Padding(1);
this->panel1->Name = L"panel1";
this->panel1->Size = System::Drawing::Size(218, 205);
this->panel1->TabIndex = 10;
//
// label41
//
this->label41->AutoSize = true;
this->label41->Location = System::Drawing::Point(144, 107);
this->label41->Name = L"label41";
this->label41->Size = System::Drawing::Size(41, 13);
this->label41->TabIndex = 25;
this->label41->Text = L"label41";
//

```

```

// label40
//
this->label40->AutoSize = true;
this->label40->Location = System::Drawing::Point(144, 85);
this->label40->Name = L"label40";
this->label40->Size = System::Drawing::Size(41, 13);
this->label40->TabIndex = 24;
this->label40->Text = L"label40";
//
// label39
//
this->label39->AutoSize = true;
this->label39->Location = System::Drawing::Point(144, 63);
this->label39->Name = L"label39";
this->label39->Size = System::Drawing::Size(41, 13);
this->label39->TabIndex = 23;
this->label39->Text = L"label39";
//
// label38
//
this->label38->AutoSize = true;
this->label38->Location = System::Drawing::Point(79, 25);
this->label38->Name = L"label38";
this->label38->Size = System::Drawing::Size(41, 13);
this->label38->TabIndex = 22;
this->label38->Text = L"label38";
//
// button5
//
this->button5->BackColor = System::Drawing::Color::LightBlue;
this->button5->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button5->Location = System::Drawing::Point(41, 151);
this->button5->Name = L"button5";
this->button5->Size = System::Drawing::Size(112, 31);
this->button5->TabIndex = 13;
this->button5->Text = L"Текст";
this->button5->UseVisualStyleBackColor = false;
this->button5->Click += gcnew System::EventHandler(this, &servoForm::button5_Click);
//
// label7
//
this->label7->AutoSize = true;
this->label7->Location = System::Drawing::Point(7, 2);
this->label7->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label7->Name = L"label7";
this->label7->Size = System::Drawing::Size(75, 13);
this->label7->TabIndex = 0;
this->label7->Text = L"Лівий Елерон";
//
// label8
//
this->label8->AutoSize = true;
this->label8->Location = System::Drawing::Point(7, 24);
this->label8->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label8->Name = L"label8";
this->label8->Size = System::Drawing::Size(23, 13);
this->label8->TabIndex = 1;
this->label8->Text = L"Пін";
//
// label11
//
this->label11->AutoSize = true;
this->label11->Location = System::Drawing::Point(15, 85);
this->label11->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label11->Name = L"label11";
this->label11->Size = System::Drawing::Size(65, 13);
this->label11->TabIndex = 7;
this->label11->Text = L"Нормальна";
//
// textBox5
//
this->textBox5->Location = System::Drawing::Point(41, 21);
this->textBox5->Margin = System::Windows::Forms::Padding(1);
this->textBox5->MaxLength = 2;
this->textBox5->Name = L"textBox5";
this->textBox5->Size = System::Drawing::Size(34, 20);
this->textBox5->TabIndex = 2;
//

```

```

// textBox7
//
this->textBox7->Location = System::Drawing::Point(82, 104);
this->textBox7->Margin = System::Windows::Forms::Padding(1);
this->textBox7->MaxLength = 3;
this->textBox7->Name = L"textBox7";
this->textBox7->Size = System::Drawing::Size(58, 20);
this->textBox7->TabIndex = 6;
//
// textBox6
//
this->textBox6->Location = System::Drawing::Point(82, 82);
this->textBox6->Margin = System::Windows::Forms::Padding(1);
this->textBox6->MaxLength = 3;
this->textBox6->Name = L"textBox6";
this->textBox6->Size = System::Drawing::Size(58, 20);
this->textBox6->TabIndex = 4;
//
// label10
//
this->label10->AutoSize = true;
this->label10->Location = System::Drawing::Point(15, 63);
this->label10->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label10->Name = L"label10";
this->label10->Size = System::Drawing::Size(41, 13);
this->label10->TabIndex = 5;
this->label10->Text = L"Нижня";
//
// panel2
//
this->panel2->BackColor = System::Drawing::Color::Transparent;
this->panel2->Controls->Add(this->label50);
this->panel2->Controls->Add(this->label49);
this->panel2->Controls->Add(this->label48);
this->panel2->Controls->Add(this->button8);
this->panel2->Controls->Add(this->label46);
this->panel2->Controls->Add(this->checkBox1);
this->panel2->Controls->Add(this->label1);
this->panel2->Controls->Add(this->label2);
this->panel2->Controls->Add(this->textBox1);
this->panel2->Controls->Add(this->label3);
this->panel2->Controls->Add(this->label4);
this->panel2->Controls->Add(this->textBox2);
this->panel2->Controls->Add(this->textBox3);
this->panel2->Controls->Add(this->textBox8);
this->panel2->Controls->Add(this->label12);
this->panel2->Location = System::Drawing::Point(243, 254);
this->panel2->Margin = System::Windows::Forms::Padding(1);
this->panel2->Name = L"panel2";
this->panel2->Size = System::Drawing::Size(218, 203);
this->panel2->TabIndex = 11;
//
// label50
//
this->label50->AutoSize = true;
this->label50->Location = System::Drawing::Point(146, 114);
this->label50->Name = L"label50";
this->label50->Size = System::Drawing::Size(41, 13);
this->label50->TabIndex = 35;
this->label50->Text = L"label50";
//
// label49
//
this->label49->AutoSize = true;
this->label49->Location = System::Drawing::Point(146, 90);
this->label49->Name = L"label49";
this->label49->Size = System::Drawing::Size(41, 13);
this->label49->TabIndex = 34;
this->label49->Text = L"label49";
//
// label48
//
this->label48->AutoSize = true;
this->label48->Location = System::Drawing::Point(146, 67);
this->label48->Name = L"label48";
this->label48->Size = System::Drawing::Size(41, 13);
this->label48->TabIndex = 33;
this->label48->Text = L"label48";

```



```

//
// button8
//
this->button8->BackColor = System::Drawing::Color::LightBlue;
this->button8->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button8->Location = System::Drawing::Point(41, 155);
this->button8->Name = L"button8";
this->button8->Size = System::Drawing::Size(112, 31);
this->button8->TabIndex = 16;
this->button8->Text = L"Тест";
this->button8->UseVisualStyleBackColor = false;
this->button8->Click += gcnew System::EventHandler(this, &servoForm::button8_Click);
//
// label46
//
this->label46->AutoSize = true;
this->label46->Location = System::Drawing::Point(39, 42);
this->label46->Name = L"label46";
this->label46->Size = System::Drawing::Size(41, 13);
this->label46->TabIndex = 32;
this->label46->Text = L"label46";
//
// checkBox1
//
this->checkBox1->AutoSize = true;
this->checkBox1->Location = System::Drawing::Point(89, 23);
this->checkBox1->Margin = System::Windows::Forms::Padding(1);
this->checkBox1->Name = L"checkBox1";
this->checkBox1->Size = System::Drawing::Size(98, 17);
this->checkBox1->TabIndex = 10;
this->checkBox1->Text = L"Використання";
this->checkBox1->UseVisualStyleBackColor = true;
//
// label1
//
this->label1->AutoSize = true;
this->label1->Location = System::Drawing::Point(15, 110);
this->label1->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(39, 13);
this->label1->TabIndex = 9;
this->label1->Text = L"Права";
//
// label2
//
this->label2->AutoSize = true;
this->label2->Location = System::Drawing::Point(7, 2);
this->label2->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(85, 13);
this->label2->TabIndex = 0;
this->label2->Text = L"Рুলъ Напрямку";
//
// textBox1
//
this->textBox1->Location = System::Drawing::Point(84, 64);
this->textBox1->Margin = System::Windows::Forms::Padding(1);
this->textBox1->MaxLength = 3;
this->textBox1->Name = L"textBox1";
this->textBox1->Size = System::Drawing::Size(58, 20);
this->textBox1->TabIndex = 8;
//
// label3
//
this->label3->AutoSize = true;
this->label3->Location = System::Drawing::Point(7, 24);
this->label3->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(23, 13);
this->label3->TabIndex = 1;
this->label3->Text = L"Піи";
//
// label4
//
this->label4->AutoSize = true;
this->label4->Location = System::Drawing::Point(15, 88);
this->label4->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label4->Name = L"label4";

```

```

this->label4->Size = System::Drawing::Size(65, 13);
this->label4->TabIndex = 7;
this->label4->Text = L"Нормальна";
//
// textBox2
//
this->textBox2->Location = System::Drawing::Point(41, 21);
this->textBox2->Margin = System::Windows::Forms::Padding(1);
this->textBox2->MaxLength = 2;
this->textBox2->Name = L"textBox2";
this->textBox2->Size = System::Drawing::Size(34, 20);
this->textBox2->TabIndex = 2;
//
// textBox3
//
this->textBox3->Location = System::Drawing::Point(84, 87);
this->textBox3->Margin = System::Windows::Forms::Padding(1);
this->textBox3->MaxLength = 3;
this->textBox3->Name = L"textBox3";
this->textBox3->Size = System::Drawing::Size(58, 20);
this->textBox3->TabIndex = 6;
//
// textBox8
//
this->textBox8->Location = System::Drawing::Point(84, 110);
this->textBox8->Margin = System::Windows::Forms::Padding(1);
this->textBox8->MaxLength = 3;
this->textBox8->Name = L"textBox8";
this->textBox8->Size = System::Drawing::Size(58, 20);
this->textBox8->TabIndex = 4;
//
// label12
//
this->label12->AutoSize = true;
this->label12->Location = System::Drawing::Point(15, 71);
this->label12->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label12->Name = L"label12";
this->label12->Size = System::Drawing::Size(29, 13);
this->label12->TabIndex = 5;
this->label12->Text = L"Jiba";
//
// panel3
//
this->panel3->BackColor = System::Drawing::Color::Transparent;
this->panel3->Controls->Add(this->label47);
this->panel3->Controls->Add(this->label45);
this->panel3->Controls->Add(this->label44);
this->panel3->Controls->Add(this->label17);
this->panel3->Controls->Add(this->button6);
this->panel3->Controls->Add(this->checkBox2);
this->panel3->Controls->Add(this->label13);
this->panel3->Controls->Add(this->label14);
this->panel3->Controls->Add(this->textBox9);
this->panel3->Controls->Add(this->label15);
this->panel3->Controls->Add(this->label16);
this->panel3->Controls->Add(this->textBox10);
this->panel3->Controls->Add(this->textBox11);
this->panel3->Controls->Add(this->textBox12);
this->panel3->Controls->Add(this->label18);
this->panel3->Location = System::Drawing::Point(243, 36);
this->panel3->Margin = System::Windows::Forms::Padding(1);
this->panel3->Name = L"panel3";
this->panel3->Size = System::Drawing::Size(218, 205);
this->panel3->TabIndex = 12;
//
// label47
//
this->label47->AutoSize = true;
this->label47->Location = System::Drawing::Point(133, 106);
this->label47->Name = L"label47";
this->label47->Size = System::Drawing::Size(41, 13);
this->label47->TabIndex = 32;
this->label47->Text = L"label47";
//
// label45
//
this->label45->AutoSize = true;
this->label45->Location = System::Drawing::Point(133, 85);

```

```

this->label45->Name = L"label45";
this->label45->Size = System::Drawing::Size(41, 13);
this->label45->TabIndex = 31;
this->label45->Text = L"label45";
//
// label44
//
this->label44->AutoSize = true;
this->label44->Location = System::Drawing::Point(133, 63);
this->label44->Name = L"label44";
this->label44->Size = System::Drawing::Size(41, 13);
this->label44->TabIndex = 30;
this->label44->Text = L"label44";
//
// label17
//
this->label17->AutoSize = true;
this->label17->Location = System::Drawing::Point(39, 42);
this->label17->Name = L"label17";
this->label17->Size = System::Drawing::Size(41, 13);
this->label17->TabIndex = 29;
this->label17->Text = L"label17";
//
// button6
//
this->button6->BackColor = System::Drawing::Color::LightBlue;
this->button6->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button6->Location = System::Drawing::Point(41, 151);
this->button6->Name = L"button6";
this->button6->Size = System::Drawing::Size(112, 31);
this->button6->TabIndex = 14;
this->button6->Text = L"Тест";
this->button6->UseVisualStyleBackColor = false;
this->button6->Click += gcnew System::EventHandler(this, &servoForm::button6_Click);
//
// checkBox2
//
this->checkBox2->AutoSize = true;
this->checkBox2->Location = System::Drawing::Point(84, 24);
this->checkBox2->Margin = System::Windows::Forms::Padding(1);
this->checkBox2->Name = L"checkBox2";
this->checkBox2->Size = System::Drawing::Size(98, 17);
this->checkBox2->TabIndex = 10;
this->checkBox2->Text = L"Використання";
this->checkBox2->UseVisualStyleBackColor = true;
//
// label13
//
this->label13->AutoSize = true;
this->label13->Location = System::Drawing::Point(7, 107);
this->label13->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label13->Name = L"label13";
this->label13->Size = System::Drawing::Size(43, 13);
this->label13->TabIndex = 9;
this->label13->Text = L"Верхня";
//
// label14
//
this->label14->AutoSize = true;
this->label14->Location = System::Drawing::Point(7, 2);
this->label14->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label14->Name = L"label14";
this->label14->Size = System::Drawing::Size(85, 13);
this->label14->TabIndex = 0;
this->label14->Text = L"Правий Елерон";
//
// textBox9
//
this->textBox9->Location = System::Drawing::Point(71, 82);
this->textBox9->Margin = System::Windows::Forms::Padding(1);
this->textBox9->MaxLength = 3;
this->textBox9->Name = L"textBox9";
this->textBox9->Size = System::Drawing::Size(58, 20);
this->textBox9->TabIndex = 8;
//
// label15
//
this->label15->AutoSize = true;

```

```

this->label15->Location = System::Drawing::Point(7, 24);
this->label15->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label15->Name = L"label15";
this->label15->Size = System::Drawing::Size(23, 13);
this->label15->TabIndex = 1;
this->label15->Text = L"Пин";
//
// label16
//
this->label16->AutoSize = true;
this->label16->Location = System::Drawing::Point(4, 85);
this->label16->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label16->Name = L"label16";
this->label16->Size = System::Drawing::Size(65, 13);
this->label16->TabIndex = 7;
this->label16->Text = L"Нормальна";
//
// textBox10
//
this->textBox10->Location = System::Drawing::Point(41, 21);
this->textBox10->Margin = System::Windows::Forms::Padding(1);
this->textBox10->MaxLength = 2;
this->textBox10->Name = L"textBox10";
this->textBox10->Size = System::Drawing::Size(34, 20);
this->textBox10->TabIndex = 2;
//
// textBox11
//
this->textBox11->Location = System::Drawing::Point(71, 60);
this->textBox11->Margin = System::Windows::Forms::Padding(1);
this->textBox11->MaxLength = 3;
this->textBox11->Name = L"textBox11";
this->textBox11->Size = System::Drawing::Size(58, 20);
this->textBox11->TabIndex = 6;
//
// textBox12
//
this->textBox12->Location = System::Drawing::Point(71, 103);
this->textBox12->Margin = System::Windows::Forms::Padding(1);
this->textBox12->MaxLength = 3;
this->textBox12->Name = L"textBox12";
this->textBox12->Size = System::Drawing::Size(58, 20);
this->textBox12->TabIndex = 4;
//
// label18
//
this->label18->AutoSize = true;
this->label18->Location = System::Drawing::Point(4, 63);
this->label18->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label18->Name = L"label18";
this->label18->Size = System::Drawing::Size(41, 13);
this->label18->TabIndex = 5;
this->label18->Text = L"Нижня";
//
// panel5
//
this->panel5->BackColor = System::Drawing::Color::Transparent;
this->panel5->Controls->Add(this->label37);
this->panel5->Controls->Add(this->label36);
this->panel5->Controls->Add(this->label35);
this->panel5->Controls->Add(this->label33);
this->panel5->Controls->Add(this->button4);
this->panel5->Controls->Add(this->label22);
this->panel5->Controls->Add(this->label25);
this->panel5->Controls->Add(this->textBox15);
this->panel5->Controls->Add(this->label26);
this->panel5->Controls->Add(this->label27);
this->panel5->Controls->Add(this->textBox17);
this->panel5->Controls->Add(this->textBox18);
this->panel5->Controls->Add(this->textBox19);
this->panel5->Controls->Add(this->label29);
this->panel5->Location = System::Drawing::Point(14, 254);
this->panel5->Margin = System::Windows::Forms::Padding(1);
this->panel5->Name = L"panel5";
this->panel5->Size = System::Drawing::Size(218, 203);
this->panel5->TabIndex = 11;
//
// label37

```

```

//
this->label37->AutoSize = true;
this->label37->Location = System::Drawing::Point(144, 110);
this->label37->Name = L"label37";
this->label37->Size = System::Drawing::Size(41, 13);
this->label37->TabIndex = 21;
this->label37->Text = L"label37";
//
// label36
//
this->label36->AutoSize = true;
this->label36->Location = System::Drawing::Point(144, 86);
this->label36->Name = L"label36";
this->label36->Size = System::Drawing::Size(41, 13);
this->label36->TabIndex = 20;
this->label36->Text = L"label36";
//
// label35
//
this->label35->AutoSize = true;
this->label35->Location = System::Drawing::Point(144, 63);
this->label35->Name = L"label35";
this->label35->Size = System::Drawing::Size(41, 13);
this->label35->TabIndex = 19;
this->label35->Text = L"label35";
//
// label33
//
this->label33->AutoSize = true;
this->label33->Location = System::Drawing::Point(79, 25);
this->label33->Name = L"label33";
this->label33->Size = System::Drawing::Size(41, 13);
this->label33->TabIndex = 18;
this->label33->Text = L"label33";
//
// button4
//
this->button4->BackColor = System::Drawing::Color::LightBlue;
this->button4->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button4->Location = System::Drawing::Point(53, 155);
this->button4->Name = L"button4";
this->button4->Size = System::Drawing::Size(112, 31);
this->button4->TabIndex = 13;
this->button4->Text = L"Тест";
this->button4->UseVisualStyleBackColor = false;
this->button4->Click += gcnew System::EventHandler(this, &servoForm::button4_Click);
//
// label22
//
this->label22->AutoSize = true;
this->label22->Location = System::Drawing::Point(15, 110);
this->label22->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label22->Name = L"label22";
this->label22->Size = System::Drawing::Size(43, 13);
this->label22->TabIndex = 9;
this->label22->Text = L"Верхняя";
//
// label25
//
this->label25->AutoSize = true;
this->label25->Location = System::Drawing::Point(7, 2);
this->label25->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label25->Name = L"label25";
this->label25->Size = System::Drawing::Size(69, 13);
this->label25->TabIndex = 0;
this->label25->Text = L"Руль высоты";
//
// textBox15
//
this->textBox15->Location = System::Drawing::Point(82, 60);
this->textBox15->Margin = System::Windows::Forms::Padding(1);
this->textBox15->MaxLength = 3;
this->textBox15->Name = L"textBox15";
this->textBox15->Size = System::Drawing::Size(58, 20);
this->textBox15->TabIndex = 8;
//
// label26
//

```

```

this->label26->AutoSize = true;
this->label26->Location = System::Drawing::Point(7, 24);
this->label26->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label26->Name = L"label26";
this->label26->Size = System::Drawing::Size(23, 13);
this->label26->TabIndex = 1;
this->label26->Text = L"Пий";
//
// label27
//
this->label27->AutoSize = true;
this->label27->Location = System::Drawing::Point(15, 90);
this->label27->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label27->Name = L"label27";
this->label27->Size = System::Drawing::Size(65, 13);
this->label27->TabIndex = 7;
this->label27->Text = L"Нормална";
//
// textBox17
//
this->textBox17->Location = System::Drawing::Point(41, 21);
this->textBox17->Margin = System::Windows::Forms::Padding(1);
this->textBox17->MaxLength = 2;
this->textBox17->Name = L"textBox17";
this->textBox17->Size = System::Drawing::Size(34, 20);
this->textBox17->TabIndex = 2;
//
// textBox18
//
this->textBox18->Location = System::Drawing::Point(82, 107);
this->textBox18->Margin = System::Windows::Forms::Padding(1);
this->textBox18->MaxLength = 3;
this->textBox18->Name = L"textBox18";
this->textBox18->Size = System::Drawing::Size(58, 20);
this->textBox18->TabIndex = 6;
//
// textBox19
//
this->textBox19->Location = System::Drawing::Point(82, 83);
this->textBox19->Margin = System::Windows::Forms::Padding(1);
this->textBox19->MaxLength = 3;
this->textBox19->Name = L"textBox19";
this->textBox19->Size = System::Drawing::Size(58, 20);
this->textBox19->TabIndex = 4;
//
// label29
//
this->label29->AutoSize = true;
this->label29->Location = System::Drawing::Point(15, 67);
this->label29->Margin = System::Windows::Forms::Padding(1, 0, 1, 0);
this->label29->Name = L"label29";
this->label29->Size = System::Drawing::Size(41, 13);
this->label29->TabIndex = 5;
this->label29->Text = L"Нижня";
//
// button1
//
this->button1->BackColor = System::Drawing::Color::LightBlue;
this->button1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button1->Location = System::Drawing::Point(293, 494);
this->button1->Margin = System::Windows::Forms::Padding(1);
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(171, 54);
this->button1->TabIndex = 14;
this->button1->Text = L"Вийти";
this->button1->UseVisualStyleBackColor = false;
this->button1->Click += gcnew System::EventHandler(this, &servoForm::button1_Click);
//
// button2
//
this->button2->BackColor = System::Drawing::Color::LightBlue;
this->button2->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button2->Location = System::Drawing::Point(14, 494);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(171, 54);
this->button2->TabIndex = 15;
this->button2->Text = L"Зберегти";
this->button2->UseVisualStyleBackColor = false;

```

```

        this->button2->Click += gcnew System::EventHandler(this, &servoForm::button2_Click);
        //
        // servoForm
        //
        this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
        this->BackColor = System::Drawing::SystemColors::ScrollBar;
        this->ClientSize = System::Drawing::Size(474, 560);
        this->ControlBox = false;
        this->Controls->Add(this->button2);
        this->Controls->Add(this->button1);
        this->Controls->Add(this->panel5);
        this->Controls->Add(this->panel3);
        this->Controls->Add(this->panel2);
        this->Controls->Add(this->panel1);
        this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedDialog;
        this->Icon = (cli::safe_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));
        this->Margin = System::Windows::Forms::Padding(1);
        this->Name = L"servoForm";
        this->Text = L"Налаштування сервоприводів";
        this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this,
&servoForm::servoForm_FormClosing);
        this->Load += gcnew System::EventHandler(this, &servoForm::servoForm_Load);
        this->panel1->ResumeLayout(false);
        this->panel1->PerformLayout();
        this->panel2->ResumeLayout(false);
        this->panel2->PerformLayout();
        this->panel3->ResumeLayout(false);
        this->panel3->PerformLayout();
        this->panel5->ResumeLayout(false);
        this->panel5->PerformLayout();
        this->ResumeLayout(false);
    }

#pragma endregion
private: void SendCOMTest(array<Byte>^ byteArray) {
    try {
        if (serialPort1 != nullptr) {
            if (!serialPort1->IsOpen)
                serialPort1->Open();
            serialPort1->Write(byteArray, 0, byteArray->Length);
            serialPort1->Close();
            MessageBox::Show("Дані відправлено успішно");
        }
        else {
            MessageBox::Show("Порт зайнятий. Можливо на цей порт вже відправляються дані, або інша програма
використовує його");
        }
    }
    catch (Exception^ ex) {
        MessageBox::Show("Виникла помилка під час відправки даних в порт: " + ex->Message);
    }
}

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
    this->Hide();
    Owner->Show();
}

private: System::Void servoForm_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e) {
    Owner->Show();
}

private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void servoForm_Load(System::Object^ sender, System::EventArgs^ e);
private: System::Void button5_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void button6_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void button4_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void button8_Click(System::Object^ sender, System::EventArgs^ e);
};
}

```

## SettingsForm.h

```
#pragma once
```

```

namespace HacerioPilot {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;

```

```

using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;

/// <summary>
/// Summary for settingsForm
/// </summary>
public ref class settingsForm : public System::Windows::Forms::Form
{
public:
    settingsForm(void)
    {
        InitializeComponent();
        //
        //TODO: Add the constructor code here
        //
    }

protected:
    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    ~settingsForm()
    {
        if (components)
        {
            delete components;
        }
    }

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::CheckBox^ checkBox1;
private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::TextBox^ textBox5;
private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::Label^ label6;
private: System::Windows::Forms::TextBox^ textBox8;
private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::Label^ label9;
private: System::Windows::Forms::TextBox^ textBox12;
private: System::Windows::Forms::Label^ label10;

private: System::Windows::Forms::CheckBox^ checkBox2;

private: System::Windows::Forms::Label^ label12;
private: System::Windows::Forms::TextBox^ textBox16;

private: System::Windows::Forms::CheckBox^ checkBox5;

private: System::Windows::Forms::CheckBox^ checkBox7;
private: System::Windows::Forms::CheckBox^ checkBox8;

private: System::Windows::Forms::CheckBox^ checkBox9;
private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::Button^ button2;

```



```

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Label^ label8;
private: System::Windows::Forms::Label^ label11;
private: System::Windows::Forms::Label^ label13;

protected:

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    void InitializeComponent(void)
    {
        System::ComponentModel::ComponentResourceManager^ resources = (gcnew
System::ComponentModel::ComponentResourceManager(settingsForm::typeid));
        this->label1 = (gcnew System::Windows::Forms::Label());
        this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());
        this->label3 = (gcnew System::Windows::Forms::Label());
        this->textBox5 = (gcnew System::Windows::Forms::TextBox());
        this->label5 = (gcnew System::Windows::Forms::Label());
        this->label6 = (gcnew System::Windows::Forms::Label());
        this->textBox8 = (gcnew System::Windows::Forms::TextBox());
        this->label7 = (gcnew System::Windows::Forms::Label());
        this->label9 = (gcnew System::Windows::Forms::Label());
        this->textBox12 = (gcnew System::Windows::Forms::TextBox());
        this->label10 = (gcnew System::Windows::Forms::Label());
        this->checkBox2 = (gcnew System::Windows::Forms::CheckBox());
        this->label12 = (gcnew System::Windows::Forms::Label());
        this->textBox16 = (gcnew System::Windows::Forms::TextBox());
        this->checkBox5 = (gcnew System::Windows::Forms::CheckBox());
        this->checkBox7 = (gcnew System::Windows::Forms::CheckBox());
        this->checkBox8 = (gcnew System::Windows::Forms::CheckBox());
        this->checkBox9 = (gcnew System::Windows::Forms::CheckBox());
        this->button1 = (gcnew System::Windows::Forms::Button());
        this->button2 = (gcnew System::Windows::Forms::Button());
        this->label2 = (gcnew System::Windows::Forms::Label());
        this->label8 = (gcnew System::Windows::Forms::Label());
        this->label11 = (gcnew System::Windows::Forms::Label());
        this->label13 = (gcnew System::Windows::Forms::Label());
        this->SuspendLayout();
        //
        // label1
        //
        this->label1->AutoSize = true;
        this->label1->BackColor = System::Drawing::Color::Transparent;
        this->label1->Location = System::Drawing::Point(14, 53);
        this->label1->Name = L"label1";
        this->label1->Size = System::Drawing::Size(74, 13);
        this->label1->TabIndex = 0;
        this->label1->Text = L"Вісь Двигуна";
        //
        // checkBox1
        //
        this->checkBox1->AutoSize = true;
        this->checkBox1->BackColor = System::Drawing::Color::Transparent;
        this->checkBox1->Location = System::Drawing::Point(17, 69);
        this->checkBox1->Name = L"checkBox1";
        this->checkBox1->Size = System::Drawing::Size(78, 17);
        this->checkBox1->TabIndex = 5;
        this->checkBox1->Text = L"Джойстик";
        this->checkBox1->UseVisualStyleBackColor = false;
        //
        // label3
        //
        this->label3->AutoSize = true;
        this->label3->BackColor = System::Drawing::Color::Transparent;
        this->label3->Location = System::Drawing::Point(14, 129);
        this->label3->Name = L"label3";
        this->label3->Size = System::Drawing::Size(60, 13);
        this->label3->TabIndex = 6;
    }

```

```

this->label3->Text = L"Вісь крену";
//
// textBox5
//
this->textBox5->Location = System::Drawing::Point(103, 86);
this->textBox5->Name = L"textBox5";
this->textBox5->Size = System::Drawing::Size(55, 20);
this->textBox5->TabIndex = 9;
//
// label5
//
this->label5->AutoSize = true;
this->label5->BackColor = System::Drawing::Color::Transparent;
this->label5->Location = System::Drawing::Point(14, 89);
this->label5->Name = L"label5";
this->label5->Size = System::Drawing::Size(83, 13);
this->label5->TabIndex = 10;
this->label5->Text = L"Мертва зона %";
//
// label6
//
this->label6->AutoSize = true;
this->label6->BackColor = System::Drawing::Color::Transparent;
this->label6->Location = System::Drawing::Point(14, 148);
this->label6->Name = L"label6";
this->label6->Size = System::Drawing::Size(83, 13);
this->label6->TabIndex = 13;
this->label6->Text = L"Мертва зона %";
//
// textBox8
//
this->textBox8->BackColor = System::Drawing::SystemColors::Window;
this->textBox8->Location = System::Drawing::Point(103, 145);
this->textBox8->Name = L"textBox8";
this->textBox8->Size = System::Drawing::Size(55, 20);
this->textBox8->TabIndex = 14;
//
// label7
//
this->label7->AutoSize = true;
this->label7->BackColor = System::Drawing::Color::Transparent;
this->label7->Location = System::Drawing::Point(264, 129);
this->label7->Name = L"label7";
this->label7->Size = System::Drawing::Size(74, 13);
this->label7->TabIndex = 15;
this->label7->Text = L"Вісь Тангажу";
//
// label9
//
this->label9->AutoSize = true;
this->label9->BackColor = System::Drawing::Color::Transparent;
this->label9->Location = System::Drawing::Point(264, 148);
this->label9->Name = L"label9";
this->label9->Size = System::Drawing::Size(83, 13);
this->label9->TabIndex = 20;
this->label9->Text = L"Мертва зона %";
//
// textBox12
//
this->textBox12->Location = System::Drawing::Point(353, 145);
this->textBox12->Name = L"textBox12";
this->textBox12->Size = System::Drawing::Size(51, 20);
this->textBox12->TabIndex = 21;
//
// label10
//
this->label10->AutoSize = true;
this->label10->BackColor = System::Drawing::Color::Transparent;
this->label10->Location = System::Drawing::Point(264, 53);
this->label10->Name = L"label10";
this->label10->Size = System::Drawing::Size(80, 13);
this->label10->TabIndex = 22;
this->label10->Text = L"Вісь Рискання";
//
// checkBox2
//
this->checkBox2->AutoSize = true;
this->checkBox2->BackColor = System::Drawing::Color::Transparent;

```

```

this->checkBox2->Location = System::Drawing::Point(267, 69);
this->checkBox2->Name = L"checkBox2";
this->checkBox2->Size = System::Drawing::Size(166, 17);
this->checkBox2->TabIndex = 25;
this->checkBox2->Text = L"Використання осі рискання";
this->checkBox2->UseVisualStyleBackColor = false;
//
// label12
//
this->label12->AutoSize = true;
this->label12->BackColor = System::Drawing::Color::Transparent;
this->label12->Location = System::Drawing::Point(264, 89);
this->label12->Name = L"label12";
this->label12->Size = System::Drawing::Size(83, 13);
this->label12->TabIndex = 28;
this->label12->Text = L"Мертва зона %";
//
// textBox16
//
this->textBox16->Location = System::Drawing::Point(353, 86);
this->textBox16->Name = L"textBox16";
this->textBox16->Size = System::Drawing::Size(64, 20);
this->textBox16->TabIndex = 29;
//
// checkBox5
//
this->checkBox5->AutoSize = true;
this->checkBox5->BackColor = System::Drawing::Color::Transparent;
this->checkBox5->Location = System::Drawing::Point(267, 212);
this->checkBox5->Name = L"checkBox5";
this->checkBox5->Size = System::Drawing::Size(70, 17);
this->checkBox5->TabIndex = 40;
this->checkBox5->Text = L"Тумблер";
this->checkBox5->UseVisualStyleBackColor = false;
//
// checkBox7
//
this->checkBox7->AutoSize = true;
this->checkBox7->BackColor = System::Drawing::Color::Transparent;
this->checkBox7->Location = System::Drawing::Point(267, 189);
this->checkBox7->Name = L"checkBox7";
this->checkBox7->Size = System::Drawing::Size(205, 17);
this->checkBox7->TabIndex = 37;
this->checkBox7->Text = L"Використання додаткової функції 1";
this->checkBox7->UseVisualStyleBackColor = false;
//
// checkBox8
//
this->checkBox8->AutoSize = true;
this->checkBox8->BackColor = System::Drawing::Color::Transparent;
this->checkBox8->Location = System::Drawing::Point(17, 212);
this->checkBox8->Name = L"checkBox8";
this->checkBox8->Size = System::Drawing::Size(70, 17);
this->checkBox8->TabIndex = 44;
this->checkBox8->Text = L"Тумблер";
this->checkBox8->UseVisualStyleBackColor = false;
//
// checkBox9
//
this->checkBox9->AutoSize = true;
this->checkBox9->BackColor = System::Drawing::Color::Transparent;
this->checkBox9->Location = System::Drawing::Point(17, 189);
this->checkBox9->Name = L"checkBox9";
this->checkBox9->Size = System::Drawing::Size(205, 17);
this->checkBox9->TabIndex = 41;
this->checkBox9->Text = L"Використання додаткової функції 2";
this->checkBox9->UseVisualStyleBackColor = false;
//
// button1
//
this->button1->BackColor = System::Drawing::Color::LightBlue;
this->button1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button1->Location = System::Drawing::Point(267, 240);
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(199, 43);
this->button1->TabIndex = 45;
this->button1->Text = L"Вийти";
this->button1->UseVisualStyleBackColor = false;

```

```

this->button1->Click += gcnew System::EventHandler(this, &settingsForm::button1_Click);
//
// button2
//
this->button2->BackColor = System::Drawing::Color::LightBlue;
this->button2->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->button2->Location = System::Drawing::Point(12, 240);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(199, 43);
this->button2->TabIndex = 47;
this->button2->Text = L"Зберегти";
this->button2->UseVisualStyleBackColor = false;
this->button2->Click += gcnew System::EventHandler(this, &settingsForm::button2_Click);
//
// label2
//
this->label2->AutoSize = true;
this->label2->BackColor = System::Drawing::Color::Transparent;
this->label2->Location = System::Drawing::Point(164, 89);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(35, 13);
this->label2->TabIndex = 48;
this->label2->Text = L"label2";
//
// label8
//
this->label8->AutoSize = true;
this->label8->BackColor = System::Drawing::Color::Transparent;
this->label8->Location = System::Drawing::Point(164, 148);
this->label8->Name = L"label8";
this->label8->Size = System::Drawing::Size(35, 13);
this->label8->TabIndex = 50;
this->label8->Text = L"label8";
//
// label11
//
this->label11->AutoSize = true;
this->label11->BackColor = System::Drawing::Color::Transparent;
this->label11->Location = System::Drawing::Point(425, 89);
this->label11->Name = L"label11";
this->label11->Size = System::Drawing::Size(41, 13);
this->label11->TabIndex = 51;
this->label11->Text = L"label11";
//
// label13
//
this->label13->AutoSize = true;
this->label13->BackColor = System::Drawing::Color::Transparent;
this->label13->Location = System::Drawing::Point(409, 147);
this->label13->Name = L"label13";
this->label13->Size = System::Drawing::Size(41, 13);
this->label13->TabIndex = 52;
this->label13->Text = L"label13";
//
// settingsForm
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->BackColor = System::Drawing::SystemColors::ScrollBar;
this->ClientSize = System::Drawing::Size(478, 301);
this->ControlBox = false;
this->Controls->Add(this->label13);
this->Controls->Add(this->label11);
this->Controls->Add(this->label8);
this->Controls->Add(this->label2);
this->Controls->Add(this->button2);
this->Controls->Add(this->button1);
this->Controls->Add(this->checkBox8);
this->Controls->Add(this->checkBox9);
this->Controls->Add(this->checkBox5);
this->Controls->Add(this->checkBox7);
this->Controls->Add(this->textBox16);
this->Controls->Add(this->label12);
this->Controls->Add(this->checkBox2);
this->Controls->Add(this->label10);
this->Controls->Add(this->textBox12);
this->Controls->Add(this->label9);
this->Controls->Add(this->label7);

```

```

        this->Controls->Add(this->textBox8);
        this->Controls->Add(this->label6);
        this->Controls->Add(this->label5);
        this->Controls->Add(this->textBox5);
        this->Controls->Add(this->label3);
        this->Controls->Add(this->checkBox1);
        this->Controls->Add(this->label1);
        this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
        this->Icon = (cli::safe_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));
        this->Name = L"settingsForm";
        this->Text = L"Налаштування пульта";
        this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this,
&settingsForm::settingsForm_FormClosing);
        this->Load += gcnew System::EventHandler(this, &settingsForm::settingsForm_Load);
        this->ResumeLayout(false);
        this->PerformLayout();

    }

#pragma endregion
    private: System::Void settingsForm_Load(System::Object^ sender, System::EventArgs^ e);
    private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
        this->Hide();
        Owner->Show();
    }
    private: System::Void settingsForm_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e) {
        Owner->Show();
    }
    private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e);
};
}

```

## Додаток Б

### Сирцевий код прошивки БПЛА

#### DyplomeUAV.ino

```

тї#include <EEPROM.h>
#include <SPI.h>
#include <Servo.h>
#include "RF24.h"
const uint64_t pipe = 0x4ACE910; //Назва труби
RF24 radio(9, 10);
byte left_min;
byte left_max;
byte left_nor;
bool right_isuse;
byte right_min;
byte right_nor;
byte right_max;
byte back_min;
byte back_nor;
byte back_max;
bool rysk_isuse;
byte rysk_min;
byte rysk_nor;
byte rysk_max;
byte radiochanel;
byte radiopower;
byte engine_limit;
bool engine_is_servo;
bool other1_isuse;
byte other1_pin;
byte other2_isuse;
bool other2_pin;

byte mass[6];
byte power;
byte kren;
byte tang;
byte rysk;
byte dod1;

```

```

byte dod2;

void ReadSerial() { //Функція для Читання з порта і запису в EEPROM
  Serial.println("Powered by HacerioPilot 0.9");
  while(!Serial){}
  while(true){
    if(Serial.available() > 0){
      if(Serial.read() == 1){
        if(Serial.read() == 0){ // Запис в файл
          byte receivedData[27];
          for (int i = 0; i < 27; i++) {
            receivedData[i] = Serial.read() + '0';
            delay(1);
          }
          for (int i = 0; i < 27; i++) {
            EEPROM.write(i, receivedData[i]);
            delay(1);
          }
          EEPROM.write(27, 128);
        }
      }
    }
  }
}

void setup() {
  Serial.begin(9600);
  delay(2);
  pinMode(0, INPUT);
  if(digitalRead(0)){ReadSerial();} //Якщо натиснута відлагодна кнопка
  else { //Тест серво
    if(Serial.read() == 1){
      Servo test;
      byte testpin = Serial.read() + '0';
      test.attach(testpin);
      byte testmin = Serial.read() + '0';
      byte testnor = Serial.read() + '0';
      byte testmax = Serial.read() + '0';
      test.write(testmin);
      delay(1000);
      test.write(testmax);
      delay(1000);
      test.write(testnor);
    }
  }
  Servo left;
  Servo right;
  Servo back;
  Servo rysk;
  Servo engine;
  if(EEPROM.read(27) == 128){
    engine.attach(EEPROM.read(0)); //Ініціалізація мотора
    engine_limit = EEPROM.read(1);
    engine_is_servo = EEPROM.read(2);
    if(engine_is_servo == 0){
      engine.writeMicroseconds(2300);
      delay(3000);
      engine.writeMicroseconds(800);
      delay(3000);
    }
    else{
      engine.write(90);
      delay(500);
      engine.write(0);
      delay(500);
    }
    left.attach(EEPROM.read(3)); //Лівий серво
    left_min = EEPROM.read(4);
    left_nor = EEPROM.read(5);
    left_max = EEPROM.read(6);
    if(EEPROM.read(7) == 1){ //Правий серво
      right_isuse = 1;
      right.attach(EEPROM.read(8));
      right_min = EEPROM.read(9);
      right_nor = EEPROM.read(10);
      right_max = EEPROM.read(11);
    }
    else right_isuse = 0;
    back.attach(EEPROM.read(12)); //Задній серво
  }
}

```

```

back_min = EEPROM.read(13);
back_nor = EEPROM.read(14);
back_max = EEPROM.read(15);
if(EEPROM.read(16) == 1){//Серво рискання
  rysk_isuse = 1;
  rysk.attach(EEPROM.read(17));
  rysk_min = EEPROM.read(18);
  rysk_nor = EEPROM.read(19);
  rysk_max = EEPROM.read(20);
}
else rysk_isuse = 0;
radiochannel = EEPROM.read(21);//Радіозв'язок
radiopower = EEPROM.read(22);
if(EEPROM.read(24) == 1){//Дод1
  other1_isuse = 1;
  other1_pin = EEPROM.read(23);
  pinMode(other1_pin, OUTPUT);
}
else other1_isuse = 0;
if(EEPROM.read(26 == 1)){//Дод2
  other2_isuse = 1;
  other2_pin = EEPROM.read(25);
}
else other2_isuse = 0;

back.write(back_min); //Перевірка сервоприводів
left.write(left_min);
if(right_isuse) right.write(right_min);
if(rysik_isuse) rysik.write(rysik_min);
delay(500);

back.write(back_max);
left.write(left_max);
if(right_isuse) right.write(right_max);
if(rysik_isuse) rysik.write(rysik_max);
delay(500);

back.write(back_nor);
left.write(left_nor);
if(right_isuse) right.write(right_nor);
if(rysik_isuse) rysik.write(rysik_nor);

delay(200);
radio.begin(); //ініціалізація радіоприймача
delay(2);
radio.setChannel(radiochannel);
radio.setDataRate(RF24_250KBPS);
if(radiopower == 0){//Налаштування потужності LNA підсилювача
  radio.setPALevel(RF24_PA_MIN);
}
else if(radiopower == 1){
  radio.setPALevel(RF24_PA_LOW);
}
else if(radiopower == 2){
  radio.setPALevel(RF24_PA_HIGH);
}
else{radio.setPALevel(RF24_PA_MAX);}
radio.openReadingPipe(1,pipe);
radio.startListening();

} else{
  ReadSerial();
}
delay(2);
}

void loop() {
  if (radio.available()){
    radio.read(&mass, sizeof(mass));
    power = mass[0];
    kren = mass[1];
    tang = mass[2];
    rysk = mass[3];
    dod1 = mass[4];
    dod2 = mass[5];
    engine.writeMicroseconds(map(power, 0, 100, 800, 2300));
    if(right_isuse == 1){
      byte left_pos = kren;

```

```

    byte right_pos = 100 - kren;
    SetServoPos(left, left_min, left_max, left_nor, left_pos);
    SetServoPos(right, right_min, right_max, right_nor, right_pos);
  }
  else{
    SetServoPos(left, left_min, left_max, left_nor, kren);
  }
  SetServoPos(back, back_min, back_max, back_nor, tang);
  if(rysk_isuse == 1){
    SetServoPos(rysk, rysk_min, rysk_max, rysk_nor, rysk);
  }
  if(other1_isuse == 1){
    digitalWrite(other1_pin , dod1);
  }
  if(other2_isuse == 1){
    digitalWrite(other2_pin, dod2);
  }
}

delay(1);
}

void SetServoPos(Servo servo, byte minimum, byte maximum, byte normal, byte pos){
  if(pos <= 48){
    servo.write(map(pos, 0, 100, minimum, maximum));
  }
  else if(pos >= 52){
    servo.write(map(pos, 0, 100, minimum, maximum));
  }
  else{
    servo.write(normal);
  }
}
}

```

## Додаток В

### Сирцевий код прошивки пульта керування

#### DyplomePult.ino

```

#include <EEPROM.h>
#include <SPI.h>
#include "RF24.h"
byte enginejoy_is_joy;
byte engine_deadzone;
byte engine_value;
byte tang_deadzone;
byte tang_value;
byte rysk_isuse;
byte rysk_value;
byte rysk_deadzone;
byte kren_value;
byte kren_deadzone;
byte mass[6];
byte radiopower;
byte radiochanel;
const uint64_t pipe = 0x4ACE910; //Назва труби
RF24 radio(9, 10);
void setup() {
  radio.begin();
  delay(2);
  radio.setDataRate(RF24_250KBPS);
  if (EEPROM.read(8) != 128){
    ReadSerial();
  }
  enginejoy_is_joy = EEPROM.read(0);
  engine_deadzone = EEPROM.read(1);
  kren_deadzone = EEPROM.read(2);
  rysk_isuse = EEPROM.read(3);
  rysk_deadzone = EEPROM.read(4);
  tang_deadzone = EEPROM.read(5);
  radiochanel = EEPROM.read(6);
  radiopower = EEPROM.read(7);
  radio.setChannel(radiochanel);
}

```



```

radio.setDataRate(RF24_250KBPS);
if(radiopower == 0){//Налаштування потужності LNA підсилювача
  radio.setPALevel(RF24_PA_MIN);
}
else if(radiopower == 1){
  radio.setPALevel(RF24_PA_LOW);
}
else if(radiopower == 2){
  radio.setPALevel(RF24_PA_HIGH);
}
else{radio.setPALevel(RF24_PA_MAX);}
radio.openWritingPipe(pipe);
radio.stopListening();
delay(2);
}
void ReadSerial(){ //Функція для Читання з порта і запису в EEPROM
  Serial.println("Powered by HacerioPilot 0.9");
  while(!Serial){}
  while(true){
    if(Serial.available() > 0){ // Запис в файл
      byte receivedData[8];
      for (int i = 0; i < 8; i++) {
        receivedData[i] = Serial.read() + '0';
        delay(1);
      }
      for (int i = 0; i < 8; i++) {
        EEPROM.write(i, receivedData[i]);
        delay(1);
      }
      EEPROM.write(8, 128);
    }
  }
}
void loop() {
  engine_value = analogRead(A0);
  tang_value = analogRead(A1);
  if(rysk_isuse){
    rysk_value = analogRead(A2);
  }
  else{
    rysk_value = 512;
  }
  kren_value = analogRead(A3);
  if(enginejoy_is_joy){
    engine_value = map(engine_value, 512 + engine_deadzone, 1023 - engine_deadzone, 0, 100);
  }
  else{
    engine_value = map(engine_value, 0 + engine_deadzone, 1023 - engine_deadzone, 0, 100);
  }
  tang_value = map(tang_value, 0 + tang_deadzone, 1023 - tang_deadzone, 0, 100);
  kren_value = map(kren_value, 0 + kren_deadzone, 1023 - kren_deadzone, 0, 100);
  rysk_value = map(rysk_value, 0 + rysk_deadzone, 1023 - rysk_deadzone, 0, 100);
  mass[0] = engine_value;
  mass[1] = kren_value;
  mass[2] = tang_value;
  mass[3] = rysk_value;
  mass[4] = 0;
  mass[5] = 0;
  radio.write(mass, sizeof(mass));
}

```