Хмельницький національний університет
Факультет інформаційних технологій
Кафедра автоматизації та комп'ютерно-інтегрованих технологій

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Система керування безпілотним літальним апаратом
Назва теми

КвРАКІТ.021032.01.05.ПЗ
Шифр

Галузь знань *15 «Автоматизація та приладобудування»*
Шифр, назва

Спеціальність *151 «Автоматизація та комп'ютерно-інтегровані технології»*
Шифр, назва

Освітня програма *«Автоматизація та комп'ютерно-інтегровані технології»*
Назва

Виконав: студент IV курсу, група АКІТ-21-К _____ Андрій ЄРОХІН
Підпис Ініціали, прізвище

Керівник _____ Микола ФЕДУЛА
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ Людмила КОРЕЦЬКА
Підпис, дата Ініціали, прізвище

До захисту допускаю:
зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Валерій МАРТИНЮК
Підпис Ініціали, прізвище

«19» червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра АВТОМАТИЗАЦІЇ ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ ТЕХНОЛОГІЙ

Освітній рівень БАКАЛАВР

Галузь знань 15 АВТОМАТИЗАЦІЯ ТА ПРИЛАДОБУДУВАННЯ

Спеціальність 151 АВТОМАТИЗАЦІЯ ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ

Освітня програма «АВТОМАТИЗАЦІЯ ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ»

ЗАТВЕРДЖУЮ

Завідувач кафедри АКІТтаР

Валерій МАРТИНЮК

07 лютого 2025р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Андрію ЄРОХІНУ
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Система керування безпілотним літальним апаратом
Керівник проекту (роботи) Микола ФЕДУЛА, к. т. н. доцент кафедри
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання
Затверджена наказом ректора університету від 07.02.2025 р. № 5
2. Строк подання студентом проекту (роботи) на кафедру 02.06.2025 р.
3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Система керування безпілотним літальним апаратом та постановка задачі щодо її удосконалення. Проєктування системи обробки інформації в системі керування безпілотним літальним апаратом. Програмно-апаратна реалізація системи керування безпілотним літальним апаратом.
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Загальна схема системи керування. Загальна блок-схема системи керування. Загальний код програми для системи керування
6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Людмила КОРЕЦЬКА, доцент кафедри АКІТтаР		
Антиплагіат	Микола ФЕДУЛА, доцент кафедри АКІТтаР		

7. Дата видачі завдання « 07 » лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	01.12-15.12.2024	виконано
2	Ознайомлення з предметною областю. формулювання мети та задач дослідження. визначення об'єкта та предмета дослідження	15.12-31.12.2024	виконано
3	Робота над розділом 1 – аналіз відомих методів керування	01.01-20.02.2025	виконано
4	Робота над розділом 2 – розробка алгоритму керування об'єкту	21.02-20.03.2025	виконано
5	Робота над розділом 3 – розробка принципової схеми та програмного забезпечення керування БПЛА	21.03-30.05.2025	виконано
6	Оформлення пояснювальної записки згідно вимог	11.05-20.05.2025	виконано
7	Перевірка на плагіат, нормоконтроль. Здача КвР на кафедру	21.05-30.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2025 року	

Студент _____ Андрій ЄРОХІН
Підпис Ім'я, прізвище

Керівник роботи _____ Микола ФЕДУЛА
Підпис Ім'я, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Система керування безпілотним літальним апаратом та постановка задачі щодо її удосконалення».

Автор роботи: Андрій ЄРОХІН.

Керівник роботи: Микола ФЕДУЛА.

Пояснювальна записка: 67 с., 47 рисунків, 5 табл., 41 джерело

Графічна частина: 15 презентаційних слайдів

БПЛА, СИСТЕМА КЕРУВАННЯ, МІКРОКОНТРОЛЕР, ПОВІТРЯНА ШВИДКІСТЬ, ОБРОБКА ДАНИХ.

Мета роботи: розробити та дослідити систему керування безпілотним літальним апаратом, а також виконати аналіз підходів до реалізації програмно-апаратної частини з урахуванням особливостей побудови систем автоматичного керування БПЛА для забезпечення стабільного польоту та ефективного виконання завдань. Об'єктом дослідження є процес керування безпілотним літальним апаратом у режимі реального часу. Предметом дослідження є апаратні та програмні методи реалізації системи керування БПЛА. Під час виконання дослідження було використано метод аналізу літературних джерел, нормативних документів та технічної документації для вивчення принципів побудови систем керування безпілотними літальними апаратами, а також порівняння сучасних підходів до реалізації керуючих алгоритмів.

Підпис студента

17.06.2025

Дата

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

СПШ – Сенсор повітряної швидкості

БПЛА – безпілотний літальний апарат

АЦП – аналого-цифровий перетворювач

В – Вольт

А – Ампер

Вт – Ват

GND – загальний провід («земля», англ. ground)

CAN_L – сигнальна лінія низького рівня CAN-шини

CAN_H – сигнальна лінія високого рівня CAN-шини

СВПШ – сенсор вимірювання повітряної швидкості

ШІМ – широтно імпульсна модуляція

CAN – Controller Area Network (протокол зв’язку)

I2C – Inter-Integrated Circuit (протокол зв’язку)

SPI – Serial Peripheral Interface (протокол зв’язку)

UART – Universal Asynchronous Receiver-Transmitter (протокол зв’язку)

GPS – Global Positioning System

ESC – Electronic Speed Controller

ВСТУП

У сучасному світі безпілотні системи все частіше використовуються для різних завдань у повітрі. Їх застосовують як у цивільних цілях, наприклад, для доставки їжі, товарів або ліків у важкодоступні райони, так і для моніторингу інфраструктури, аерофотозйомки та спостереження за станом навколишнього середовища. Крім того, такі системи знаходять застосування у військових та спеціалізованих сферах – для патрулювання, розвідки або логістичної підтримки.

Технології, описані в цій роботі, можна використовувати не тільки на БПЛА, а й на звичайних літаках або інших повітряних платформах. Для вимірювання швидкості повітря використовується трубка Піто, яка забезпечує точні дані для системи керування. Автоматичне і стабільне регулювання швидкості двигуна на основі цих даних допомагає зробити роботу літального апарата більш безпечною, ефективною й економною.

Метою дипломного проєкту є створення простої та ефективної системи автоматичного керування для БПЛА. У роботі розглядається процес зчитування даних з сенсорів, їх обробка, а також побудова алгоритмів, які дозволяють на основі цих даних керувати польотом апарата. Було враховано інтеграцію розробленої системи для різних типів БПЛА та повітряних платформ.

Для досягнення ключової мети потрібно виконати такі завдання дипломного проєкту:

- розробити та зібрати апаратну частину системи;
- розробити алгоритми автоматичного керування двигуном залежно від швидкості повітря;
- реалізувати програмне забезпечення для зчитування, обробки та аналізу даних із сенсорів.

Таким чином, розробка та впровадження ефективних систем автоматичного керування для безпілотних і пілотованих літальних апаратів є

актуальним завданням сьогодення, яке відповідає потребам ринку та технологічному розвитку авіації.

Об’єктом дослідження є процес керування безпілотним літальним апаратом.

Предметом дослідження є методи, алгоритми та структурні рішення реалізації системи керування безпілотним літальним апаратом.

Розроблена система може застосовуватись для керування швидкістю двигуна в різних типах БПЛА, включаючи фіксованокрилі та багатогвинтові апарати.

1 ОГЛЯД ВІДОМИХ ТЕХНОЛОГІЙ

1.1 Загальні відомості про сенсори повітряної швидкості

У системах керування безпілотними літальними апаратами сенсори вимірювання повітряної швидкості відіграють важливу роль для стабільного польоту. Особливо у випадках, коли точність і стабільність польоту мають критичне значення. На відміну від GPS, який показує швидкість відносно землі, сенсори типу СВПШ вимірюють реальну повітряну швидкість, тобто швидкість відносно потоку повітря. Це особливо важливо при польотах проти вітру, з вітром або при складних погодних умовах.

Система вимірювання повітряної швидкості складається з двох основних частин – трубки Піто та електронного сенсора. Трубка Піто – це металева трубка, зазвичай алюмінієва, яка ловить потік повітря під час руху. Вона направляє це повітря до сенсора, який вимірює тиск. Електронний сенсор обчислює різницю між статичним і динамічним тиском, і за допомогою формули обчислюється повітряна швидкість. Ці дані передаються на польотний контролер, який використовує їх для керування польотом. Крім того, багато сучасних сенсорів можуть також вимірювати інші параметри, наприклад температуру, та вологість.

Статичний тиск – це атмосферний тиск навколишнього середовища, який вимірюється в стані спокою, тому його вимірюють через отвори, які розташовані перпендикулярно потоку повітря.

Динамічний тиск – це різниця між загальним та статичним тиском. Формула динамічного тиску наведена нижче (1.1) [15].

$$P\text{дин} = \frac{1}{2} * \ p \ * \ v^2 \qquad (1.1)$$

Загальний тиск – це сума статичного та динамічного тиску.

Визначається через отвір, який розташований в напрямку руху. Формула загального тиску наведена нижче (1.2).

$$Pзаг = Pстат + Pдин \quad (1.2)$$

Диференційний тиск – це різниця між статичним та динамічним тиском. Використовується для визначення повітряної швидкості за формулою Бернуллі наведеної нижче (1.3) [17].

$$Pдин = Pзаг - Pстат \quad (1.3)$$

Протокол зв'язку – інтерфейс, який використовується для обміну даними між модулями та польотним контролером. У цій роботі використовуються такі протоколи: I2C, SPI, CAN, UART.

Трубка Піто має два отвори – для вимірювання загального тиску(поєднує динамічний та статичний тиск) використовується фронтальний отвір (тиск виникає від швидкості набігаючого потоку) та для статичного тиску використовуються бокові отвори (тиск відповідає атмосферному). Різниця між статичним та динамічним тиском дозволяє визначити повітряну швидкість за формулою наведеною нижче (1.4) [16].

$$V = \frac{2(Pd-Ps)}{\rho} \quad (1.4)$$

де $V -$ швидкість, $\rho -$ густина повітря.

Нижче наведено конструкцію трубки Піто (рисунок 1.1).

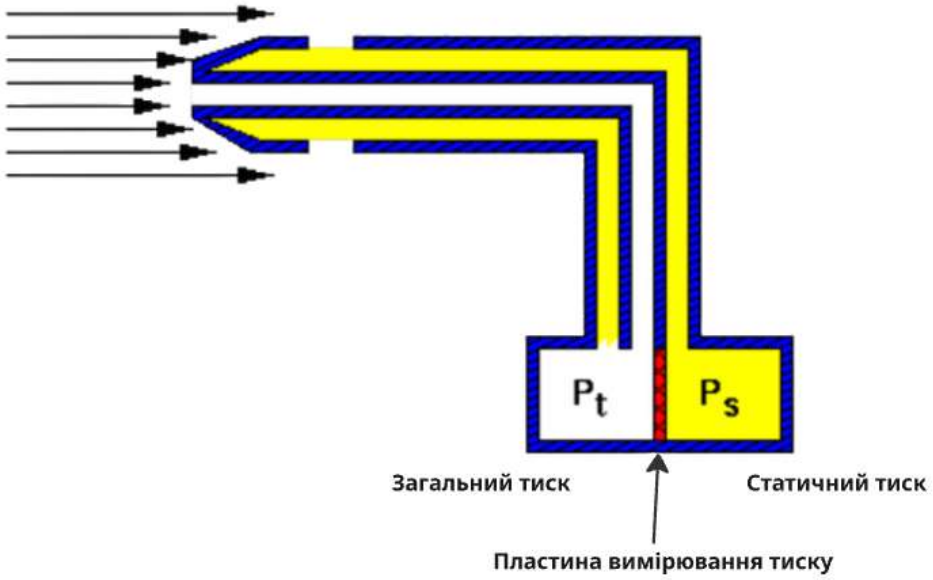

Рисунок 1.1 – Конструкція трубки Піто

Використання СВПШ дозволяє літальному апарату точно оцінювати свою аеродинамічну поведінку та приймати коригувальні рішення в реальному часі. Наприклад, якщо швидкість повітря занадто низька є ризик аеродинамічного зриву, тобто втрати підйомної сили. За допомогою СВПШ система керування може вчасно підвищити тягу або змінити кут атаки, запобігаючи аварійним ситуаціям, а також повітряна швидкість є головним параметром для підтримки крейсерської швидкості.

Сучасні сенсори вимірювання повітряної швидкості зазвичай реалізовані у вигляді трубки Піто, виготовленої з алюмінієвого матеріалу, у поєднанні з диференційним датчиком тиску, який вимірює різницю тисків та автоматично обчислює повітряну швидкість. Отримані дані передаються до польотного контролера через інтерфейси CAN, I2C, UART, SPI. Якщо сенсор використовує протокол CAN, дані передаються за допомогою програмного протоколу DroneCAN [23].

DroneCAN [1] – це програмний протокол цифрового зв'язку основою якого є апаратний CAN. Протокол розроблений для БПЛА та інших систем, який має розширенні функції для зручного обміну даними. Він розроблений на основі протоколу CAN, який використовується в автомобільній галузі та авіації.

Протокол забезпечує високу швидкість та стійкість до електромагнітних перешкод [27]

DroneCAN підтримують польотні контролери та більшість сенсорів та модулів БПЛА, наприклад GPS модулі, ESC, барометри, СВПШ та інші.

1.2 Класифікація сенсорів повітряної швидкості

СВПШ класифікуються за кількома критеріями, які залежать від діапазону вимірювань, типу вимірювань, типу обчислень, способу передачі даних, та додаткові функції сенсора.

Сенсори повітряної швидкості бувають різних типів залежно від того, з якими швидкостями вони можуть працювати.

Низькошвидкісні сенсори – вони підходять для повільних БПЛА, які літають до 30 м/с. Універсальні сенсори можуть вимірювати швидкість до 50 - 100 м/с і використовуються в більшості звичайних БПЛА. Для швидкісних БПЛА, наприклад реактивних, потрібні високошвидкісні сенсори, які можуть фіксувати швидкість понад 150 м/с.

Сенсори поділяються на два типи – ті, що самостійно обчислюють швидкість повітря та передають готове значення, і ті, що видають лише «сирі» значення тиску, які передаються на польотний контролер для подальшого обчислення швидкості, який обробляє ці дані та визначає реальну швидкість БПЛА [24].

Сенсори можуть передавати дані двома способами – через аналоговий інтерфейс або цифровий. Аналогові сенсори видають сигнал у вигляді напруги, і контролер сам розраховує значення. Це простий варіант, але менш точний, через різні зовнішні завади та втрати. Цифрові сенсори передають вже готові значення через спеціальні інтерфейси, такі як I2C, UART, SPI або CAN [24].

Сенсори повітряної швидкості можуть мати багато корисних додаткових функцій, які покращують точність. Наприклад, автоматичне калібрування тиску, коли БПЛА знаходиться на землі, сенсор сам обнуляє значення. Також

важливою є температурна компенсація, коли сенсор зчитує температуру повітря і враховує її при обчисленнях, тому що від температури залежить густина повітря. Також сенсори мають захист від електричних завад, вони фільтрують сигнал, щоб зменшити вплив шуму з інших частин електроніки.

Деякі сенсори можуть самі визначати помилки, наприклад, неправильний тиск або перегрів, і попереджати про це контролер.

1.3 Огляд існуючих сенсорів повітряної швидкості

На сучасному ринку представлено багато сенсорів повітряної швидкості які відрізняються між собою за багатьма параметрами. Ці сенсори активно використовуються в БПЛА, де важливим є точне вимірювання швидкості відносно повітряного потоку. Найбільш поширеними є сенсори розроблені на базі трубки Піто, які вимірюють динамічний та статичний тиск і на основі їх різниці визначають повітряну швидкість.

Розглядатимуться продукти, що активно використовуються в БПЛА. Будуть порівняні їх основні технічні характеристики та додаткові функції. Також для кращого рзуміння, нижче наведено основну термінологію, що буде використовуватися при аналізі СВПШ [22].

Загальна похибка (Total Error) – похибка, яка враховує все, що може вплинути на точність вимірювання (відхилення температури, тиску, гістерезис, різні завади на сигнал).

Діапазон тиску (Pressure Rage) – діапазон значень, яке може виміряти давач без пошкодження (наприклад, від -6 кПа до 6 кПа відносно атмосферного тиску) [20].

1.3.1. Airspeed sensor CUAV SKYE

Airspeed sensor CUAV SKYE (рисунок 1.2) використовується в професійних системах БПЛА, а також у різних військових та спеціалізованих

цілях, де важлива точність та надійність системи в екстремальних умовах (підвищена температура, турбулентність та інші умови). Завдяки використанню протоколу CAN, давач захищений від зовнішніх завад, що покращує стабільність. СВПШ призначений для вимірювань динамічного та статичного тиску, температури, а також інших атмосферних показників. Завдяки вбудованому мікроконтролеру та алгоритмам попередньої обробки даних, сенсор може здійснювати обчислення повітряної швидкості та компенсуватитиск за допомогою додаткових вимірювань вологості і температури повітря й передавати результати у цифровому вигляді до польотного контролера. Конструкція давача включає вбудований нагрівач трубки Піто, що дозволяє уникнути обмерзання під час польотів за низьких температур або у вологому середовищі. Також CUAV SKYE підтримує температурну компенсацію, що забезпечує стабільні вимірювання у широкому діапазоні температур. Пристрій підтримує протокол DroneCAN, тому сумісний з багатьма польотними контролерами. Біля основного давача додано резистивний нагрівач, який забезпечує підігрів диференційного сенсора, для уникнення замерзання давача та покращених показників [26, 2].

Рисунок 1.2 – Airspeed sensor CUAV SKYE

Модуль СВПШ від компанії CUAV SKYE використовується для вимірювання повітряної швидкості в БПЛА та інших літальних апаратах. Розроблений на базі мікроконтролера STM32F401, який керує обробкою сигналів і передачею даних. Для зв’язку з іншими елементами системи застосовується протокол DroneCAN, що працює через апаратний інтерфейс

CAN.

У модулі встановлено прецензійний диференційний сенсор тиску MS5525, який вимірює різницю між повним і статичним тиском для обчислення швидкості.

Для роботи в умовах низьких температур пристрій має систему нагріву з резистивним елементом потужністю 35 Вт, яка запобігає обмерзанню трубки Піто, а також є підігрів основного давача тиску для коректної роботи. Модуль працює від джерела постійної напруги в межах 12 до 36 В.

Основний сенсор тиску, що використовується в модулі, розроблений компанією Honeywell. Він вимірює повітряну швидкість до ±500 км/год, що дозволяє застосовувати його в швидкісних системах керування. Сенсор підтримує зчитування даних через інтерфейси I2C, SPI та аналоговий вихід. Загальна похибка вимірювань становить лише ±2.5% від повної шкали, що є достатнім для більшості застосувань у безпілотних літальних апаратах.

Діапазон робочих температур становить від 0 до +85 °C. Він здатен вимірювати тиск у межах від 0.07 до 2.49 PSI.

Живлення сенсора здійснюється при напрузі від 1.8 до 3.6 В. Додатково сенсор має вбудовану функцію автоматичного калібрування тиску та температурну компенсацію, що дозволяє зберігати точність вимірювань при змінних умовах середовища.

СВПШ який був використаний в CUAV SKYE зображено на рисунку 1.3.

Рисунок 1.3 – Диференційний сенсор MS5525

Далі, на рисунку 1.4, показано, як розташовані основні частини модуля СВПШ CUAV SKYE. У центрі знаходиться головний диференційний сенсор тиску, який вимірює різницю між статичним і повним тиском. Щоб уникнути обмерзання, в модулі є система нагріву трубки Піто, яка працює на керамічних резисторах, а також окремий підігрів для самого сенсора тиску. Усім цим керує мікроконтролер STM32F401, який збирає дані з сенсорів і передає їх у польотний контролер. Додатково в модулі є сенсор температури та вологості, щоб враховувати вплив навколишнього середовища на вимірювання [14].

Рисунок 1.4 – Схема розташування компонентів на модулі СВПШ CUAV SKYE

1.3.2. Airspeed sensor CUAV SKYE 2

Airspeed sensor CUAV SKYE 2 – це оновлена версія СВПШ CUAV SKYE, яка отримала технічні покращення для підвищення точності вимірювання, надійності роботи та зменшення розмірів конструкції. У новій модифікації було змінено головний сенсор тиску – замість MS5525 використано покращений SM5391 [4], який має вищу точність і стабільність показників, це важливо для критичних польотних місій. Одним з важливих покращень стала заміна

вбудованого стабілізатора живлення на зовнішній перетворювач. Завдяки цьому вдалося зменшити розміри плати сенсора та зробити сам пристрій легшим. Це також покращило обтікання повітрям під час польоту, зменшивши опір. Це позитивно вплинуло на аеродинамічні характеристики модуля, зменшивши опір повітрю під час польоту. Крім цього, оновлений зовнішній перетворювач живлення має збільшену потужність, що дозволяє забезпечити ефективніший обігрів трубки Піто. Це значно знижує ризик її обмерзання під час польотів у холодному кліматі та підвищує надійність роботи сенсора в складних погодних умовах.

Рисунок 1.5 – Airspeed sensor CUAV SKYE 2

Модуль системи вимірювання повітряної швидкості (СВПШ) від компанії CUAV SKYE побудований на основі мікроконтролера STM32F401 і використовує протокол DroneCAN, що працює через апаратну CAN-шину.

У якості основного сенсора застосовується прецизійний диференційний давач тиску SM5391, який забезпечує точні вимірювання повітряної швидкості на основі різниці тисків.

Для захисту трубки Піто та сенсора від обмерзання в модулі реалізовано резистивну систему нагріву з потужністю 35 Вт.

Стандартна робоча напруга складає 16 В, проте за наявності зовнішнього джерела живлення модуль може працювати від 16 до 68 В.

Сенсор тиску від Honeywell використовується для вимірювання повітряної швидкості та тиску в різних умовах. Він може вимірювати швидкість до 380 км/год. Підключити сенсор можна через цифровий інтерфейс I2C та

через аналоговий вихід.

Похибка сенсора при використані протоколу I2C становить ±1% від повної шкали, а при аналоговому зчитуванні ±1.5%. Сенсор працює в широкому температурному діапазоні від -20 °C до +125 °C. Він також може вимірювати тиск від 0.07 до 2.49 PSI.

Також є автоматичне калібрування та температурна компенсація, тобто сенсор підлаштовується під умови.

Нижче наведено розташування компонентів модуля СВПШ CUAV SKYE 2, яке за структурою майже не відрізняється від попередньої версії (рисунок 1.6). Основні елементи розташовані подібним чином, у центрі знаходиться диференційний сенсор тиску, що виконує основні вимірювання [34]. Система нагріву побудована на керамічних резисторах. Замість STM32F401 у цій версії використовується мікроконтролер на базі ARM Cortex-M4, який відповідає за обробку та передачу даних.

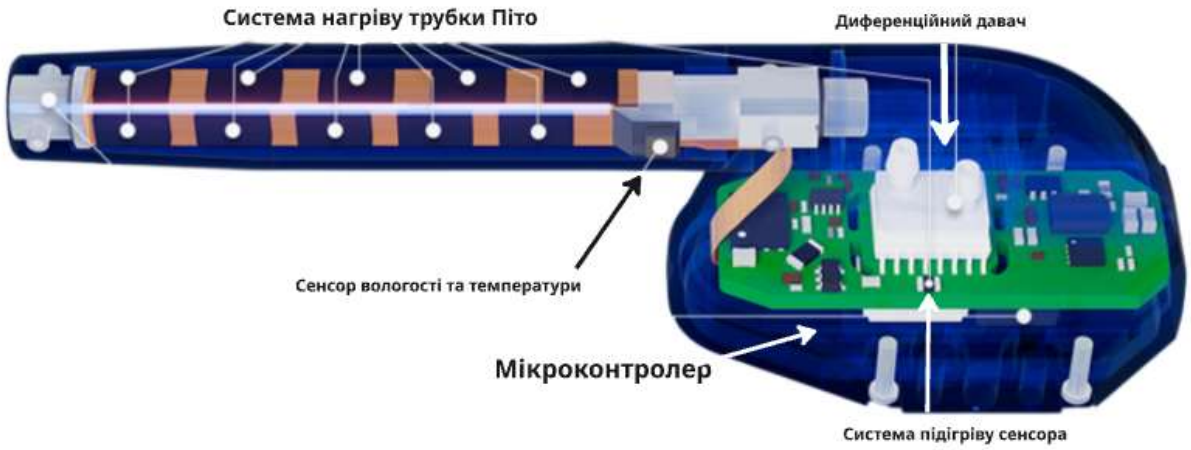

Рисунок 1.6 – Схема розташування компонентів на модулі СВПШ CUAV SKYE 2

1.3.3. Matek ASPD 4525

Matek ASPD 4525 – це цифровий сенсор, який розроблений для використання в безпілотних системах аматорського рівня, де є менша потреба

в надійності та точності, ніж в професійних рішеннях.

СВПШ поставляється без захисного корпусу, тому є можливість інтегрувати його в зручне місце на літальному апараті. Для обміну даними з польотним контролером використовується інтерфейс I2C, що немає захисту від електромагнітних завад, на відміну професійного CAN, який захищений від сильних перешкод.

СВПШ побудований на основі сенсору MS4525DO, який забезпечує гарний показник точності та діапазон виміру для аматорських цілей.

Датчик має два окремі отвори для вимірювання тиску. Один з них призначений для статичного тиску, тобто атмосферного, який не залежить від руху повітря. Другий отвір для повного тиску, що включає як статичний, так і динамічний тиск, який виникає внаслідок руху повітряного потоку [2].

На рисунку 1.7 зображено готовий вигляд сенсора у зібраному стані. До нього під'єднано трубку Піто за допомогою прозорих гнучких трубок, які з'єднують відповідні отвори на трубці та сенсорі.

Рисунок 1.7 – Зібраний стан сенсора Matek ASPD 4525

Модуль СВПШ від компанії Matek Systems побудований на основі прецизійного сенсора Honeywell MS4525DO. Для передачі даних використовується цифровий інтерфейс I2C.

Живлення модуля здійснюється в межах 4-5.5 В. Вбудовані функції автоматичного калібрування тиску та компенсації температури дозволяють сенсору працювати стабільно та точно навіть при зміні погодних умов чи висоти польоту.

Сенсор тиску MS4525DO підтримує два цифрові інтерфейси I2C та SPI.. Максимальний діапазон вимірювання швидкості становить ±360 км/год. Загальна похибка ±1% від повної шкали. Сенсор може працювати в температурному діапазоні від -20 °C до +125 °C, тому підходить для використання в різних кліматичних умовах. Діапазон вимірювання тиску від 0.07 до 2.49 PSI, що дозволяє використовувати його в широкому спектрі застосувань, пов'язаних із вимірюванням повітряного потоку.

1.3.4. Matek ASPD-DLVR

Matek ASPD-DLVR СВПШ призначений для використання в різних системах БПЛА. На відміну від попередніх моделей, цей сенсор підтримує як I2C, так і CAN інтерфейси, тому його можна використовувати для аматорських та професійних польотних контролерах. Модуль побудований на основі давача тиску Honeywell DLVR. Завдяки використанню інтерфейсу CAN, модуль має кращий захист від перешкод [5, 7].

Давач, як і попередній Honeywell MS4525DO, немає корпусу, тому є можливість встановити в зручне місце на БПЛА.

На рисунку 1.8 показано готовий вигляд сенсора з під'єднаною трубкою Піто.

Рисунок 1.8 – Зібраний стан Matek ASPD DLVR

Модуль СВПШ від Matek Systems оснащений сенсором тиску DLVR-L10D. Для передачі даних підтримуються два інтерфейси – I2C та DroneCAN, який працює через апаратну CAN-шину. Живлення модуля здійснюється від

джерела напругою 4–5.5 В. Також модуль має функції автоматичного калібрування тиску та компенсації температури. Прошивка модуля включає можливість налаштування фільтрації та інтервалів калібрування через стандартні команди I2C або специфікації DroneCAN, що спрощує інтеграцію з популярними автопілотами та контролерами польоту. Крім того, виробник надає готові бібліотеки для середовищ ArduPilot та PX4, що дозволяє швидко реалізувати алгоритми автоматичного утримання висоти, планування траєкторії та логування польотних даних у наземних станціях керування. При роботі в комплексі з іншими датчиками модуль СВПШ від Matek Systems гарантує стабільний і точний контроль навколишнього середовища навіть у складних метеоумовах.

Сенсор тиску виробництва Honeywell DLVR-L10D використовується для вимірювання повітряної швидкості та тиску. Він дозволяє вимірювати швидкість повітря в межах ±250 км/год. Для зчитування даних доступні два цифрові інтерфейси – I2C та SPI.

Загальна похибка вимірювань не перевищує ±1% від повної шкали. Сенсор працює в температурному діапазоні від -20 °C до +85 °C. Діапазон вимірювання тиску становить ±2.5 кПа.

1.4 Висновки до першого розділу

У цьому розділі було розглянуто основні типи сенсорів вимірювання повітряної швидкості (СВПШ), які активно застосовуються в безпілотних літальних апаратах. Проведено аналіз їх технічних характеристик, принципів роботи, особливостей конструкції, протоколів зв’язку та умов використання. Проаналізувавши кілька моделей, можна зробити висновок, що сенсори типу CUAV SKYE та SKYE 2 більше підходять для професійних і військових застосувань завдяки точності (компенсації тиску), надійності (наявність обігріву трубки для запобігання обмерзанню), а також використанню протоколу

CAN, що забезпечує захист від завад та підтримку інших корисних функцій. Натомість Matek ASPD 4525 є бюджетним варіантом, який підходить для аматорських проєктів з меншими вимогами до точності та надійності [2].

Також було розглянуто конструкцію трубки Піто та принцип роботи диференційного давача тиску для визначенні повітряної швидкості.

У результаті можна зробити висновок, що вибір СВПШ залежить від цілей, умов експлуатації та бюджету, а також правильна інтеграція в систему керування БПЛА [21].

2 ОГЛЯД СИСТЕМ КЕРУВАННЯ ДВИГУНАМИ БПЛА

Основним завданням проєкту є розробка системи керування двигунами БПЛА на основі даних, отриманих з диференційного сенсора тиску, який використовується для визначення повітряної швидкості. Проте цей сенсор є лише одним з багатьох модулів, що визначає повітряну швидкість та інші параметри. Для повноцінного функціонування БПЛА також враховується інформація з GPS, гіроскопів, акселерометрів, барометрів, магнітометрів та інших давачів. Польотний контролер або мікроконтролер обробляє ці дані, виконує обчислення та порівнює їх із показниками з інших сенсорів, щоб підвищити точність та надійність керування [17].

2.1 Принципи керування швидкістю моторів у БПЛА

Для керування двигунами в БПЛА використовується ESC та польотний контролер. ESC (електронний регулятор швидкості) – це електронний модуль, який приймає команди від польотного контролера, і регулює оберти двигуна.

ESC складається з мікроконтролера, силових транзисторів (MOSFET), драйверів і вбудованих фільтрів. Він отримує команду від польотного контролера (наприклад, ШІМ сигнал, згенерований на основі ПІД регулятора) і регулює напругу чи частоту, що подається на двигун, тим самим змінюючи оберти [14, 15, 16].

В ESC для керування використовується кілька видів керуючих сигналів. Найпоширенішим є ШІМ, він використовується майже в усіх типах БПЛА, оскільки простий у реалізації. Керування відбувається за допомогою імпульсів тривалістю від 1000 до 2000 мікросекунд (чим довший імпульс, тим вищі оберти двигуна). Для швидкісних БПЛА, де важлива мінімальна затримка, використовуються сигнали OneShot і MultiShot. Це теж ШІМ, але з коротшими

імпульсами – OneShot має тривалість близько 250 мкс, а MultiShot 25 мкс. Завдяки цьому ESC отримує дані швидше.

Найновіший тип сигналу це DShot. Це цифровий протокол, який передає дані у вигляді бітів, а не імпульсів. Він точніший, стійкий до шумів і не потребує калібрування, як ШІМ. DShot працює на високих швидкостях, наприклад, 600 або 1200 кбіт/с, і може передавати дані назад від ESC, якщо такий зворотний канал підтримується.

У порівняні різні типи протоколів мають свої переваги й недоліки. ШІМ простіший у реалізації, але менш точний. OneShot і MultiShot покращують передачу даних, але вимагають сумісного ESC. DShot є найсучаснішим, але потребує програмного оновлення для польотного контролера та ESC.

ESC підтримують два типи двигунів – це щіткові та безщіткові. Вони відрізняються методом регулювання швидкості через їхню конструкцію. У щіткових двигунах керування швидкістю здійснюється зміною напруги, що подається на двигун. Зі зростанням напруги зростає й частота обертання. Це просте рішення, але для серйозних проектів є недоліком через наявність щіток, які механічно контактують із рухомими частинами. Через постійне тертя щітки поступово зношуються, що призводить до зниження надійності та скорочення терміну служби двигуна. Тому щіткові двигуни рідко використовуються в сучасних БПЛА [36].

Безщіткові двигуни не мають щіток, а обертання досягається за рахунок електронного перемикання фаз обмоток статора. Цей процес здійснюється спеціальним контролером – ESC, який формує послідовність імпульсів струму відповідно до положення ротора. Завдяки такому способу керування досягається висока ефективність, точність регулювання обертів та довговічність двигуна.[17]

Також сучасні ESC можуть підтримувати деякі додаткові функції. Наприклад, зворотній зв’язок – контролер може отримувати дані про температуру, струм та інші параметри в реальному часі для запобігання можливим несправностям. Також корисною функцією є плавний старт. Завдяки

цьому двигун не запускається різко, а оберти збільшуються поступово, щоб зменшити механічне навантаження.

Також багато ESC мають функцію реверсу, тобто можуть змінювати напрям обертання двигуна, що зручно для деяких типів БПЛА або наземної техніки. Крім того, деякі ESC можуть повідомляти користувача про помилки, такі як перегрів або коротке замикання.

На рисунку 2.1 приведено модуль ESC з вбудованою схемою вирівнювання напруги.

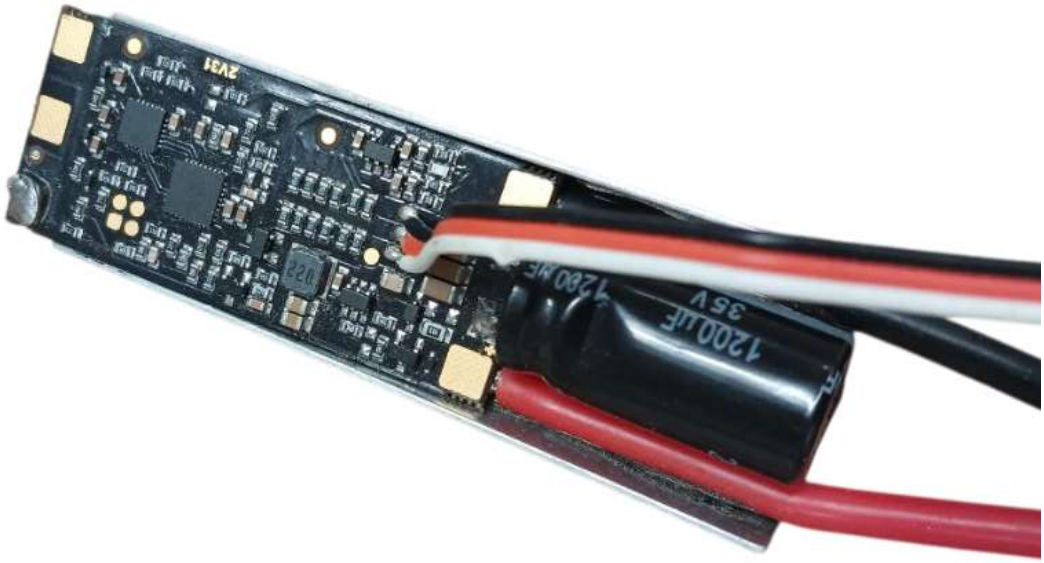

Рисунок 2.1 – Модуль ESC

2.2 Застосування сенсорів повітряної швидкості в автоматичному регулюванні швидкості моторів

Сенсори повітряної швидкості є невід’ємною частиною автоматизованих систем керування двигунами БПЛА, оскільки вони забезпечують точні дані для регулювання тяги в реальному часі. У розробленій системі, як описано в розділі 1.1, сенсор із трубкою Піто (наприклад, MS4525DO) вимірює різницю між динамічним (Pd) і статичним тиском (Ps), обчислюючи швидкість за формулою

$V = \sqrt{\frac{2*(Pd-Ps}{\rho}}$ (ρ – густина повітря). Ці дані передаються через інтерфейс I2C до мікроконтролера, який використовує їх для автоматичного коригування обертів двигунів через електронний регулятор швидкості (ESC), як зазначено в розділі 2.1.

Автоматичне регулювання швидкості базується на принципі зворотного зв'язку: мікроконтролер порівнює виміряну швидкість із заданим значенням (наприклад, 20 м/с для крейсерського режиму) і застосовує алгоритм ПІД-регулятора для обчислення необхідної корекції. ПІД-регулятор – це пристрій, який, отримуючи команди від основної системи керування (регулятора), змінює параметри в системі керування, такі як стан двигуна, компресора чи інших виконавчих пристроїв. Він може виконувати функції фільтрації сигналу, зміни рівня напруги, підсилення сигналу, а також забезпечувати плавне регулювання системи. Наприклад, в системі БПЛА, якщо швидкість падає до 15 м/с через зустрічний вітер, мікроконтролер генерує сигнал (PWM або DShot, залежно від ESC), щоб збільшити оберти безщіткових двигунів, підтримуючи стабільність польоту завдяки регулювання ПІД-регулятора. Це дозволяє уникнути аеродинамічного зриву, що особливо важливо для малих БПЛА, де низька швидкість може призвести до втрати підіймальної сили [26, 15].

Інтеграція сенсорів повітряної швидкості з іншими датчиками, описаними в розділі 1.3, такими як гіроскопи, акселерометри (для вимірювання прискорення) та барометри (для контролю висоти), забезпечує комплексний підхід до автоматизації. Наприклад, якщо барометр фіксує зниження висоти через зміну густини повітря, мікроконтролер може скоригувати тягу, спираючись на дані від давача диференційного тиску. Для підвищення точності застосовуються алгоритми обробки, такі як фільтр Калмана [9], який зчитує дані з шумом та іншими завадами протягом певного часу та фільтрує їх для подальшої обробки, це особливо важливо при турбулентності чи перепадах температури, це може впливати на густину повітря та інші показники [21].

Нижче на наведено приклад даних до та після фільтрування (жовтий графік – вхідні значення, синій графік – дані після фільтрування) (рисунок 2.2).

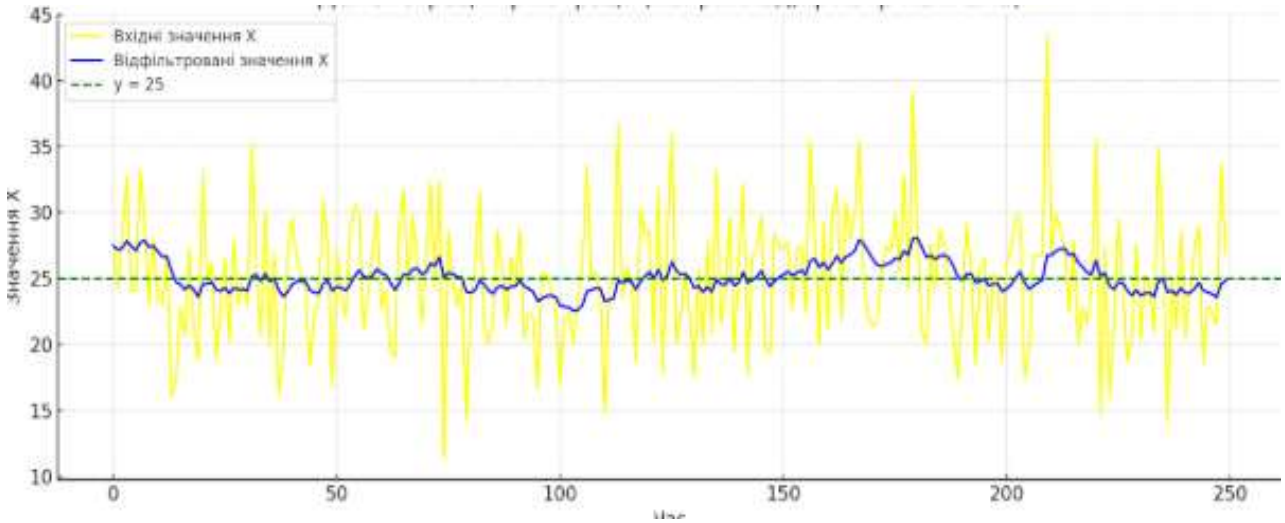

Рисунок 2.2 – Фільтр Калмана

2.3 Огляд апаратних платформ для реалізації керування

Для ефективного керування потрібно використовувати платформи, які здатні швидко обробляти дані з СВПШ та передавати дані керування на ESC для керування двигуном. Це важливо для забезпечення стабільного польоту та контрольованого польоту.

В сучасних БПЛА для цього використовують професійні польотні контролери. Контролер обробляє дані, щоб автоматично змінювати оберти двигунів через ESC. Від вибору платформи залежить, наскільки швидко й точно система реагуватиме на зміни. Також важливо, щоб система була здатна працювати з сучасними протоколами такими як PWM та DShot. Крім того, важлива швидка обробка основних та додаткових завдань такі як отримавння.

У цьому розділі буде розглянуто кілька популярних апаратних та програмних платформ для керування польотом.

Arduino (рисунок 2.3) – це одна з найпопулярніших і доступних платформ, яку широко використовують для створення прототипів різноманітних

електронних пристроїв, включаючи безпілотні літальні апарати. Наприклад, Arduino Uno – це компактна плата з мікроконтролером ATmega328P, що працює на частоті 16 МГц і має 32 КБ пам'яті для зберігання програм. Для більш складних завдань також існує плата Arduino Mega, яка має потужніший мікроконтролер ATmega2560 та 256 КБ пам'яті. У польотних системах Arduino може використовуватися для зчитування даних з сенсорів швидкості через відповідні протоколи. Після цього система може обчислювати реальну швидкість польоту за допомогою формули, наведеної в розділі 2.2, та передавати команди на ESC (електронний регулятор швидкості) у вигляді PWM-сигналів. Такі сигнали мають діапазон від 1000 до 2000 мікросекунд і дозволяють регулювати швидкість обертання безщіткових двигунів, що в свою чергу допомагає підтримувати потрібну швидкість польоту. Arduino є дуже зручним інструментом для початківців і професіоналів завдяки простоті використання.

Існує велика кількість готових бібліотек для роботи з протоколами I2C чи PWM, а програмування здійснюється через просте і зручне середовище Arduino IDE. Плата має доступну ціну – близько 20 доларів. Проте, як і будь-яка інша платформа, Arduino має свої обмеження. Вона не є дуже потужною, тому не завжди може швидко обробляти складні алгоритми або великі об'єми даних. Наприклад, при роботі з кількома сенсорами одночасно, такими як гіроскопи або барометри, може бути деяке уповільнення. Крім того, Arduino не підтримує сучасні протоколи передачі даних, такі як DShot, без додаткових бібліотек. Таким чином, Arduino є чудовим вибором для створення простих систем керування, але для більш вимогливих проектів, таких як обробка складних даних з кількох сенсорів чи швидка передача команд, можуть знадобитись більш потужні платформи [39].

Рисунок 2.3 – Мікроконтролер Arduino UNO

Нижче у таблиці 2.1 наведено основні технічні характеристики Arduino UNO.

Таблиця 2.1 – Технічні характеристики Arduino UNO

Мікроконтролер	ATmega328P
Тактова частота роботи	16 МГц
Пам’ять	Програмна: 32 КБ Оперативна: 2 КБ EEPROM: 1 КБ
Піни входу та виходу	14 цифрових входів та виходів (6 з них підтримує ШІМ) 6 підтримує АЦП
Інтерфейси	UART, SPI, I2C
Живлення	7-12 В
Розрядність АЦП	10 біт (0 – 1023)
Розрядність ШІМ	8 біт (0 – 255)

Raspberry Pi 5 – це невеликий комп'ютер, який часто використовують для різних складних завдань (рисунок 2.4). Він має потужний процесор ARM Cortex-A76 з тактовою частотою 2,4 ГГц. Ці характеристики дозволяють Raspberry Pi виконувати обчислювальні задачі і працювати з різними сенсорами, що важливо для створення БПЛА. Raspberry Pi також може генерувати PWM-сигнали через свої GPIO-контакти, що дозволяє використовувати його для керування ESC і регулювання обертів двигунів. Проте, Raspberry Pi працює під управлінням операційної системи Linux, що може призвести до затримок в обробці сигналів. Ці затримки можуть досягати 10 мілісекунд, що є занадто довгим для завдань, де важлива миттєва реакція. Наприклад, для коректного та точного управління двигунами в реальному часі потрібна дуже швидка обробка сигналів [30].

У випадку наявності затримок керування двигунами може бути некоректним, що вплине на стабільність польоту. Raspberry Pi чудово підходить для інших завдань, де не потрібно такої миттєвої реакції. Його можна використовувати для запису даних з сенсорів під час польоту, для обробки відео з камер або для виконання складних обчислювальних задач. Наприклад, можна використовувати його для збору і аналізу інформації про політ, або для роботи з картами та планування маршруту.

Слід зазначити, що для основного керування двигунами та обробки сигналів у реальному часі важлива кожна мілісекунда, і Raspberry Pi може бути занадто повільним. Для точного і швидкого керування БПЛА, зокрема для регулювання швидкості двигунів, краще використовувати спеціалізовані платформи. Вони здатні обробляти сигнали швидше і з меншими затримками, що робить їх набагато ефективнішими для завдань, де потрібна миттєва реакція на зміни.

Рисунок 2.4 – Мікроконтролер Raspberry Pi 5

Нижче у таблиці 2.2 наведено основні технічні характеристики Raspberry Pi 5.

Таблиця 2.2 – Технічні характеристики Raspberry Pi 5

Процесор	ARM Cortex-A76
Графічний процесор	VideoCore VII
Тактова частота роботи	2.4 ГГц
Пам’ять	Оперативна: 4 ГБ EEPROM: Підтримка SD-карти з швидкісним протоколом (UHS-1, UHS-2)
Піни входу та виходу	40 пінів
Інтерфейси	USB, Ethernet, CSI, DSI, HDMI, I2C, UART, SPI, WIFI
Живлення	5 В
Розрядність ШІМ	12 біт (4096)

STM32 – це мікроконтролер, який займає проміжне місце між платформами такими як Arduino та професійними рішеннями (рисунок 2.5).

Вона дає змогу будувати більш потужні та точні системи керування, але вимагає трохи більше знань від розробника. Одним із популярних представників цієї серії є STM32F103, який працює на частоті 72 МГц, має 64 КБ пам’яті для коду (Flash) та 20 КБ оперативної пам’яті (RAM). Це дозволяє обробляти дані значно швидше, ніж Arduino, особливо коли потрібно працювати з кількома сенсорами одночасно або реалізовувати точне керування двигунами. STM32 підтримує багато сучасних інтерфейсів, таких як I2C, PWM, DShot, UART, SPI, а також CAN-шину, яка часто використовується у більш професійних проектах для надійної передачі даних. Це робить її дуже універсальною – можна підключати різні сенсори, ESC [35]. Ще однією перевагою STM32 є її доступність – такі плати коштують всього близько 10 доларів. Це робить її привабливою для студентських проєктів. Однак у STM32 є й певні мінуси. Програмувати її складніше, ніж Arduino, тому що потрібно використовувати спеціальне середовище, наприклад, STM32CubeIDE, а також розуміти, як працює апаратна частина – таймери, регістри, частоти, периферія тощо. Також потрібно знати мову C або C++ на досить хорошому рівні. Крім того, налаштування периферії часто вимагає більше часу та розуміння, ніж у платформах із готовими бібліотеками, як Arduino.

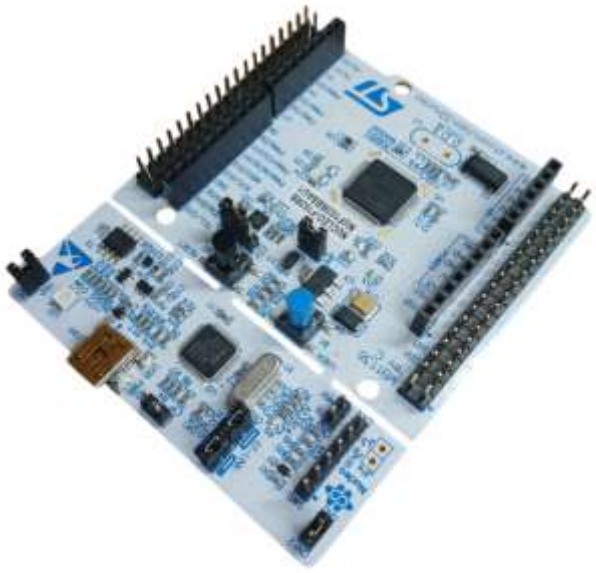

Рисунок 2.5 – Плата розробника з мікроконтролером STM32F103C8T6

Нижче у таблиці 2.3 наведено основні технічні характеристики STM32F103C8T6.

Таблиця 2.3 – Технічні характеристики STM32F103C8T6

Мікроконтролер	Cortex-M3 (ARM)
Тактова частота роботи	72 МГц
Пам’ять	Програмна: 128 КБ Оперативна: 20 КБ
Піни входу та виходу	Всього 47 пінів 10 каналів АЦП 4 канали ШІМ
Інтерфейси	USART, SPI, I2C, CAN, USB
Живлення	2 – 3.6 В
Розрядність АЦП	12 біт (0 – 4096)
Розрядність ШІМ	16 біт (0 – 65536)

Pixhawk [22] – це професійна платформа, яка широко використовується для розробки складних БПЛА (рисунок 2.6). Наприклад, Pixhawk 4 має потужний мікроконтролер STM32F427, який працює на частоті 168 МГц і оснащений 256 КБ оперативної пам'яті та 2 МБ пам'яті для програм. Така потужність дозволяє обробляти велику кількість даних і виконувати складні алгоритми керування, що особливо важливо для стабільного польоту та точного контролю безпілотного апарата.

Pixhawk підтримує роботу з різними сенсорами, такими як гіроскопи, акселерометри, GPS та інші пристрої, що дає змогу зібрати велику кількість даних для оцінки стану апарата під час польоту. Це дає можливість враховувати різні параметри, такі як вплив турбулентності, щоб забезпечити стабільність польоту навіть у складних погодних умовах. За допомогою цієї платформи можна регулювати поведінку БПЛА в реальному часі і коригувати траєкторії на основі отриманих даних.

Платформа підтримує сучасний протокол DShot, який дозволяє швидко та точно передавати команди на ESC для регулювання обертів безщіткових

двигунів. Це важливо для того, щоб зберігати стабільний контроль над апаратом під час польоту, особливо коли потрібно миттєво реагувати на зміни умов чи необхідність корекції курсу. Завдяки цій функціональності Pixhawk дозволяє досягати високої точності та надійності в керуванні двигунами [27].

Pixhawk також має можливість передавати дані про політ, такі як швидкість, висота, струм та інші параметри на комп'ютер через систему телеметрії. Це дозволяє здійснювати моніторинг стану апарата в реальному часі, що особливо важливо для віддаленого керування та корекції польоту.

Основною перевагою Pixhawk є його потужність та можливість виконувати складні завдання, такі як планування маршрутів, автоматичне коригування траєкторій і взаємодія з іншими системами на борту. Однак, з огляду на вартість цієї платформи (близько 200 доларів), вона є досить дорогою в порівнянні з простішими варіантами, такими як Arduino. Також, для роботи з Pixhawk необхідні знання програмування, зокрема мови C++, що може ускладнити роботу з цією платформою для новачків [29].

Проте, Pixhawk є ідеальним вибором для більш складних і професійних проєктів, де потрібна висока точність, стабільність і можливість обробки великої кількості даних. Для початківців, які лише знайомляться з розробкою БПЛА, робота з цією платформою може здатися складною, але для досвідчених інженерів або тих, хто працює над великими проєктами, Pixhawk стане потужним інструментом для реалізації різноманітних задач [25, 12, 15].

Рисунок 2.6 – Польотний контролер Pixhawk 4

Нижче у таблиці 2.4 наведено основні технічні характеристики системи Pixhawk 4.

Таблиця 2.4 – Технічні характеристики Pixhawk 4

Мікроконтролер	Cortex-M7 (ARM)
Тактова частота роботи	216 МГц
Пам’ять	Програмна: 2 МБ Оперативна: 512 КБ
Піни входу та виходу	Всього 26 пінів 4 каналів АЦП 8 канали ШІМ
Інтерфейси	UART, SPI, I2C, CAN, USB, RSSI, SBus, PPM, DSM
Живлення	4.3 – 5.4 В
Розрядність АЦП	12 біт (0 – 4096)
Розрядність ШІМ	16 біт (0 – 65536)

CubePilot (рисунок 2.7) – це одна з найсучасніших платформ, що часто

використовується для розробки складних БПЛА, наприклад, у комерційних проектах [23]. Однією з найпопулярніших моделей є Cube Orange+, яка має потужний мікроконтролер STM32H757 з тактовою частотою 480 МГц, 1 МБ оперативної пам'яті та 2 МБ для програм. Це дозволяє обробляти великі обсяги даних і виконувати керування в реальному часі [12, 15].

CubePilot може працювати з різними сенсорами, такими як гіроскопи та акселерометри, а також з іншими пристроями, наприклад, барометри чи магнітометри, що дозволяє зібрати велику кількість даних для підтримки стабільного польоту. Платформа також підтримує DShot (до DShot1200), що дає можливість швидко передавати команди до ESC і регулювати оберти двигунів з високою точністю.

Однією з основних переваг CubePilot є те, що він має вбудовані алгоритми, які допомагають підтримувати стабільність навіть у випадку неполадок з одним із сенсорів. Наприклад, фільтр Калмана дозволяє компенсувати помилки сенсорів, що забезпечує більшу надійність і точність польоту. Також ця платформа може працювати з різними додатковими модулями, такими як GPS або камери, що дозволяє використовувати її для складніших завдань, наприклад, для аерофотозйомки або доставки.

CubePilot також підтримує телеметрію, що дозволяє отримувати важливі дані про стан БПЛА в реальному часі, такі як температуру, струм чи напругу. Це зручно для віддаленого моніторингу і контролю за роботою БПЛА.

Однак, CubePilot коштує від 250 до 300 доларів, і для роботи з цією платформою потрібні хороші знання програмування. Вона не підходить для початківців, оскільки її налаштування і інтеграція потребують більш складного підходу. Але якщо ви працюєте з професійними БПЛА для складних завдань, таких як доставка чи зйомка, CubePilot стане відмінним вибором завдяки своїй потужності та розширюваності [27, 25, 15].

Рисунок 2.7 – Польотний контролер Cube Orange

Нижче у таблиці 25 наведено основні технічні характеристики Cube Orange [12, 15, 13].

Таблиця 2.5 – Технічні характеристики Cube Orange

Мікроконтролер	Cortex-M7 (ARM)
Тактова частота роботи	480 МГц
Пам’ять	Програмна: 2 МБ Оперативна: 1 МБ
Піни входу та виходу	Всього 80 пінів 6 каналів АЦП 14 канали ШІМ
Інтерфейси	UART, SPI, I2C, CAN, USB, SBus, PPM, DSM
Живлення	4.3 – 5.4 В
Розрядність АЦП	12 біт (0 – 4096)
Розрядність ШІМ	16 біт (0 – 65536)

Крім апаратних платформ, не менш важливим є програмне забезпечення, яке на них працює. Саме воно визначає, як ефективно буде здійснюватися керування БПЛА. Професійні платформи, такі як Pixhawk та Cube, використовують спеціалізовані програми, які називаються програмними платформами або польотними системами керування. Вони виконують важливі завдання, такі як обробка даних від різних сенсорів (гіроскопів, акселерометрів, барометрів тощо), контроль стабільності апарата, а також передавання команд для регулювання роботи двигунів [17, 15].

Програмне забезпечення на цих платформах дозволяє автоматично коригувати траєкторію польоту, враховувати зміну умов навколишнього середовища (наприклад, турбулентність або зміна напрямку вітру) і підтримувати стабільний політ навіть у складних умовах. Важливою частиною таких систем є вбудовані алгоритми, які дозволяють ефективно працювати з помилками сенсорів, наприклад, через фільтри Калмана [9] чи інші методи корекції.

Такі платформи можуть також підтримувати інтеграцію з різними додатковими модулями, наприклад, GPS для точного позиціонування або камери для збору візуальних даних. Вся ця інформація обробляється в реальному часі і дозволяє забезпечити стабільний і точний політ БПЛА. Тому правильний вибір програмного забезпечення є таким же важливим, як і вибір апаратної платформи, і впливає на ефективність роботи БПЛА.

ArduPilot – це безкоштовне програмне забезпечення з відкритим кодом, яке працює на таких платформах, як Pixhawk і Cube. В основному його використовують для управління різними типами БПЛА – дрони, літаки або гелікоптери. ArduPilot дозволяє налаштувати багато функцій (стабілізації польоту в сильний вітер, планування маршруту БПЛА, та інше).

Програма може працювати з різними сенсорами через інтерфейс I2C, CAN, а також надсилати команди на ESC через протокол DShot, щоб точно регулювати швидкість двигунів. Завдяки використанню програми Mission Planner, можна налаштувати всі необхідні параметри, зокрема калібрувати

сенсори і налаштувати правильне керування двигунів.

На Cube програма ArduPilot здатна швидко обробляти дані про швидкість і висоту, а також коригувати оберти двигунів в реальному часі. ArduPilot є потужним та надійним рішенням для керування БПЛА, але його налаштування потребує навичок [17, 20].

PX4 – це ще одна програмна платформа з відкритим кодом, яку використовують на платформах Pixhawk і Cube. Вона більше підходить для комерційних дронів. PX4 може здатен обробляти команди швидко ніж ArduPilot (до 1000 команд на секунду), що дає змогу БПЛА точно і швидко реагувати на зміни під час польоту [18, 25].

Програма підтримує роботу з різними сенсорами і може передавати команди керування двигунами через сучасні протоколи, наприклад, DShot. PX4 використовує різні алгоритми, щоб утримувати дрон стабільним, навіть якщо є сильний вітер чи інші перешкоди.

Налаштовувати PX4 зручно через програму QGroundControl – у програмі можна виставити всі потрібні параметри для сенсорів і двигунів. Хоча PX4 швидший за ArduPilot, він не такий універсальний і краще підходить для конкретних завдань, де потрібна швидкість і точність, а не широка підтримка різних типів дронів та інших БПЛА.

У порівнянні з іншими апаратними платформами, Arduino є найпростішим і найбільш доступним варіантом для розробки на початкових етапах, але вона має обмежену потужність. Pixhawk і Cube є значно більш потужними і універсальними рішеннями, при цьому Cube має кращі характеристики, що дозволяє йому ефективно справлятися з більш складними завданнями. Raspberry Pi не підходить для швидкої обробки команд керування двигунами через високу латентність, що обмежує його використання у задачах реального часу. STM32 знаходиться між Arduino та професійними платформами, поєднуючи доступність та достатню потужність для більш складних розробок.

Якщо порівнювати програмні платформи, ArduPilot є більш

універсальним для різних типів БПЛА, що дозволяє працювати з багатьма сенсорами і керувати різними системами. PX4 має більш високу швидкість обробки команд і є більш оптимальним складних застосувань завдяки здатності обробляти більше даних за менший проміжок часу [20, 16].

2.4 Висновки до другого розділу

У другому розділі були розглянуті основні принципи роботи систем керування двигунами БПЛА, що забезпечують стабільність польоту. Було описано, як працюють ESC (електронні регулятори обертів двигунів), які допомагають регулювати швидкість обертів двигунів на БПЛА. Порівняно різні типи сигналів, які передаються до ESC, зокрема стандартний PWM, а також швидші варіанти, такі як OneShot і MultiShot, та DShot, який забезпечує більшу точність і швидкість передачі даних. Також зазначено, що для БПЛА зазвичай використовують безщіткові двигуни, оскільки вони довговічніші і легші, і це важливо для БПЛА [24, 23].

Також були розглянуті різні апаратні платформи для керування БПЛА, серед яких Arduino, Pixhawk, Cube, Raspberry Pi та STM32. Для складних проєктів потрібно використовувати Pixhawk або Cube, оскільки вони забезпечують більшу швидкість обробки даних і більшу точність керування.

3 РОЗРОБКА СИСТЕМИ КЕРУВАННЯ ШВИДКІСТЮ МОТОРІВ

У цьому розділі описано розроблення системи автоматичного регулювання швидкості двигунів БПЛА залежно від швидкості польоту. Для реалізації цієї системи буде використано сенсор повітряної швидкості, перетворювач CAN–I2C, мікроконтролер Arduino, ESC, та двигун, для якого буде здійснюватися керування.

3.1 Вибір апаратного забезпечення

Мікроконтролер: Для керування системою безпілотного літального апарата обрана платформа Arduino UNO. Вона є популярною та доступною завдяки простоті використання, невисокій вартості та великій кількості готових бібліотек і документації. Arduino часто застосовують для створення прототипів, вона дозволяє швидко підключати різні сенсори й пристрої без глибоких знань схемотехніки.

Проте, Arduino має обмежену обчислювальну потужність у порівнянні з більш сучасними платформами, такими як STM32, Pixhawk чи Cube, її можливостей цілком достатньо для реалізації базових функцій керування в БПЛА.

У цьому проєкті Arduino UNO використовується для збору даних з сенсорів, обробки вхідної інформації та формування керуючих сигналів до ESC для регулювання обертів двигунів [25].

Сенсор вимірювання повітряної швидкості: У цьому проєкті буде використовуватися саморобний модуль сенсора повітряної швидкості на основі давача тиску SP160, який вимірює різницю тиску між повітрям, що подається в трубку Піто, та атмосферним. Плата модуля була розроблена в програмному забезпеченні Altium Designer, програмна частина написана в STM32CubeIDE, а сам сенсор протестований у реальних умовах на професійному БПЛА. Основні компоненти які розміщено на платі це: сенсор,

мікроконтролер STM32L431 та CAN-трансивер на мікросхемі TJA1051, який дозволяє передавати дані на інші модулі або системи в реальному часі. Мікроконтролер отримує дані з сенсора через інтерфейс I2C, виконує обробку даних та формування CAN сигналу. Через те, що точність вимірювання тиску сильно залежить від температури навколишнього середовища, було реалізовано резистивний підігрів основного давача. Нагрів контролюється мікроконтролером через MOSFET, а керування базується на температурному сенсорі, який вбудовано в сенсор тиску.

Також окремо була розроблена гнучка плата для підігріву трубки Піто, яка встановлюється безпосередньо на саму трубку. На ній розміщено температурний сенсор (цифровий сенсор з інтерфейсом I2C), який постійно передає значення температури мікроконтролеру. Завдяки цьому регулюється подача потужності на нагрівач і не допускається перегрів трубки. Окрім контролю температури, реалізовано контроль струму через нагрівальний елемент, для цього використовується шунтовий резистор разом з операційним підсилювачем, що дозволяє визначати реальну потужність, яка споживається нагрівачем, і відповідно. Також у програмі реалізовано функції обмеження максимальної температури, калібрування сенсора, а також перевірку на обрив або коротке замикання в ланцюгах підігріву. Крім цього, система підтримує автоматичну перевірку справності сенсора при старті, а також видає службові повідомлення через CAN у разі виявлення помилки (наприклад, вихід температури за межі, перевищення струму або обрив нагрівального елементу) [25, 14, 13].

Нижче наведено зображення основної плати модуля з обох боків – лицьова сторона (рисунок 3.1) та зворотна сторона (рисунок 3.2), а також зображення гнучкої плати з обох боків – лицьова сторона (рисунок 3.3) та зворотна сторона (рисунок 3.4).

Рисунок 3.1 – Лицьова сторона основної плати

Рисунок 3.2 – Зворотна сторона основної плати

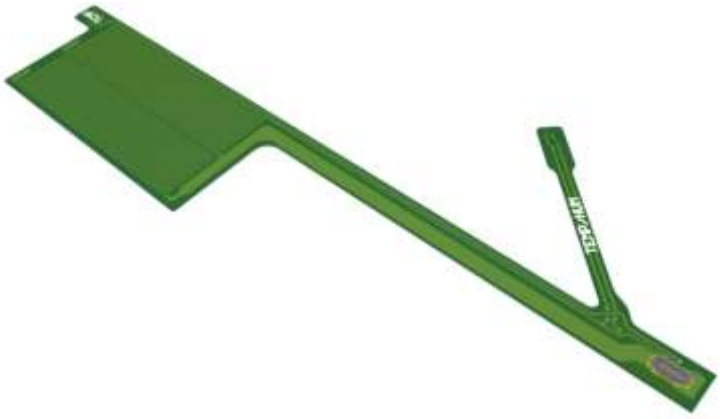

Рисунок 3.3 – Лицьова сторона гнучкої плати

Рисунок 3.4 – Зворотна сторона гнучкої плати

Оскільки мікроконтролер Arduino UNO (рисунок 3.5) не має апаратної підтримки CAN-протоколу, для забезпечення взаємодії з основним сенсором який працює через CAN-шину, буде використовуватися модуль на базі мікросхеми MCP2515 у поєднанні з трансивером TJA1050. MCP2515 є повноцінним контролером CAN-протоколу, який реалізує обробку повідомлень, фільтрацію ID, пріоритезацію та контроль помилок відповідно до стандарту CAN 2.0B. Він забезпечує логічну реалізацію протоколу, працюючи через інтерфейс SPI з мікроконтролером, але MCP2515 не може безпосередньо підключатися до фізичної CAN-шини, оскільки його виходи CAN_RX і CAN_TX працюють на рівні цифрової логіки. Саме тому використовується CAN-трансивер TJA1051, який перетворює логічні сигнали з MCP2515 у диференціальний сигнал стандарту CAN, що передається по лініях CAN_H та CAN_L. Це перетворення необхідне, адже CAN-протокол розроблений на основі передаванні даних як різниці потенціалів між CAN_H і CAN_L, що забезпечує високу завадостійкість та надійність зв’язку. Передача даних між MCP2515 і Arduino UNO здійснюється через стандартний SPI-інтерфейс, що підтримується апаратно в UNO. Таким чином, Arduino може приймати і передавати дані через справжню CAN-шину, хоча сам по собі не має CAN-контролера. У проєкті завдяки цьому модуль буде зчитуватися дані з сенсора повітряної швидкості, який передає значення через CAN-шину [15, 17].

Рисунок 3.5 – Модуль перетворювача CAN-SPI

Для керування безколекторним двигуном буде використано електронний регулятор швидкості iFlight BLITZ E55, який працює на основі прошивки BLHeli_32. Це сучасний 32-бітний ESC, який забезпечує високу точність керування, швидку реакцію на команди та підтримку цифрових протоколів, таких як DShot150/300/600, PWM та OneShot125. Завдяки прошивці BLHeli_32 користувач має можливість налаштовувати параметри ESC через спеціальні утиліти, і це дозволяє адаптувати його роботу до конкретних умов. Модуль підтримує напругу живлення до 6S (максимум 25.2 В) та забезпечує струм до 55 А, що дозволяє використовувати його з потужними двигунами. ESC оснащений вбудованими захистами від перевантаження, перегріву, короткого замикання та перенапруги, це підвищує його надійність під час роботи в умовах високих навантажень, тому цей модуль ідеально підходить для застосування в безпілотних літальних апаратах, де важлива стабільна робота двигуна [24, 16].

Нижче наведено дві сторони модуля (рисунок 3.6 – лицьова сторона, рисунок 3.7 – зворотна сторона) без радіатора, але в комплекті він передбачений. Радіатор використовується для відведення тепла від силових ключів (MOSFET), які під час роботи можуть сильно нагріватися внаслідок високих струмів і перемикань на високій частоті. Надмірне тепло може

призвести до перегріву, зниження ефективності або навіть виходу з ладу. Тому використання системи охолодження є необхідним для забезпечення стабільної та надійної роботи ESC під навантаженням.

На рисунку 3.6 показано шість MOSFET, на цій стороні плати розташовується радіатор для ефективного тепловідведення від MOSFET. Також на цій стороні розташований шунтуючий резистор, який використовується для вимірювання струму, що проходить через двигун, завдяки цьому контролюється навантаження і захист системи від перевантажень.

Рисунок 3.6 – Лицьова сторона модуля

На рисунку 3.7 показана основна частина електроніки модуля, де розташовані основні компоненти для керування та обробки сигналів.

Рисунок 3.7 – Зворотна сторона модуля

Двигун: у проекті буде використано безколекторний двигун BLDCM

(Brushless DC Motor) з параметрами 45A, 15V, 1000KV, розроблений для БПЛА. Безколекторні двигуни не мають щіток, що знижує знос і підвищує надійність, а також забезпечує більш плавну і тиху роботу.

Параметр 1000KV означає кількість обертів ротора за хвилину (об/хв) при подачі 1 вольта живлення без навантаження. Тобто, при напрузі 15В теоретично двигун може розкручуватися до 15 000 об/хв (1000 об/хв × 15 В).

Максимальний струм 45A визначає максимальне навантаження, яке двигун може витримувати без перегріву чи пошкоджень. Завдяки цьому параметру підбирається електронний регулятор швидкості (ESC), який повинен витримувати подачу такого струму.

Двигун ідеально підходить для БПЛА завдяки високій потужності, ефективності та надійності. Також він має невелику вагу та компактні розміри, що сприяє зменшенню загальної маси безпілотника [25].

Нижче зображено безколекторний двигун BLDC 45A, 15V, 1000KV якйи використовується в проєкті (рисунок 3.8).

Рисунок 3.8 – Безколекторний двигун BLDC 45A, 15V, 380KV

3.2 Розробка схеми підключення

Блок-схема загальної структури системи зображена на рисунку 3.9. У цій схемі показано взаємозв'язки між основними модулями, що використовуються у проєкті. Далі буде детально описано підключення кожного з компонентів (додаток А) між собою, принцип їхньої взаємодії та передача даних у системі.

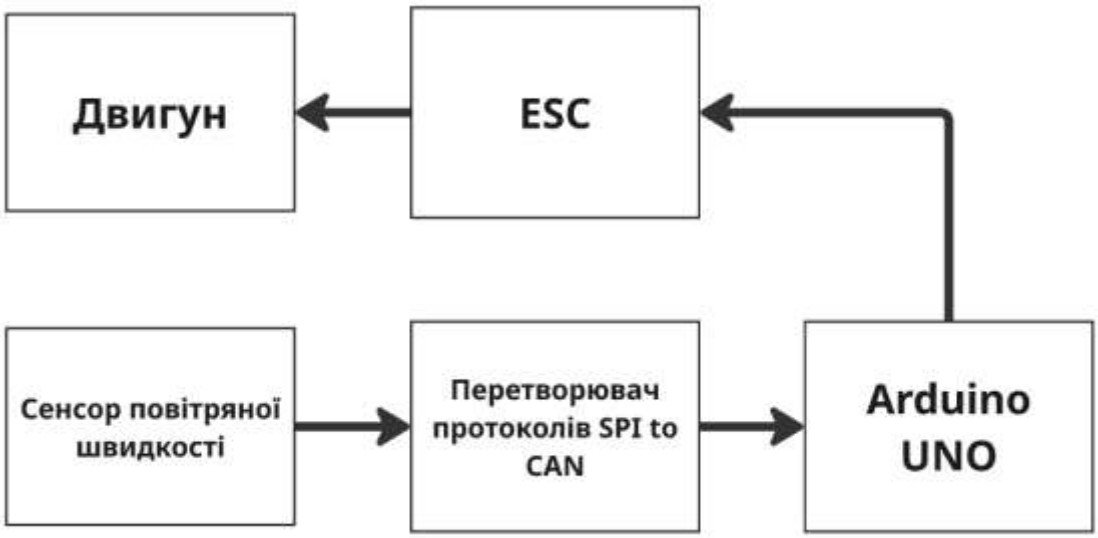

Рисунок 3.9 – Блок-схема системи керування

Підключення сенсора повітряної швидкості до модуля MCP2515 виконується за схемою, зображеною на рисунку 3.9. Модуль сенсора повітряної швидкості працює в діапазоні вхідної напруги від 5 до 12 В, тому під час тестування використовується стабілізоване джерело живлення на 5 В.

При увімкненому нагрівачі струм споживання становить до 3 А, у той час як без нагрівача – не перевищує 100 мА. Для підключення сенсора застосовується вбудований роз'єм типу JST на 8 контактів, через який подається живлення та передаються сигнали CAN-шини. З'єднання із мікроконтролером Arduino реалізовано за допомогою модуля MCP2515, який підключається до сенсора через лінії CAN_H та CAN_L. Оскільки передавання сигналів у протоколі CAN здійснюється диференційним способом, у тестовій конфігурації

GND не підключався [33, 40, 41].

Підключення здійснювалось за таким маркуванням: червоний – напруга живлення (VCC), чорний – загальний провід (GND), жовтий – сигнальна лінія CAN_H, синій – сигнальна лінія CAN_L. Схема зображена на рисунку 3.10.

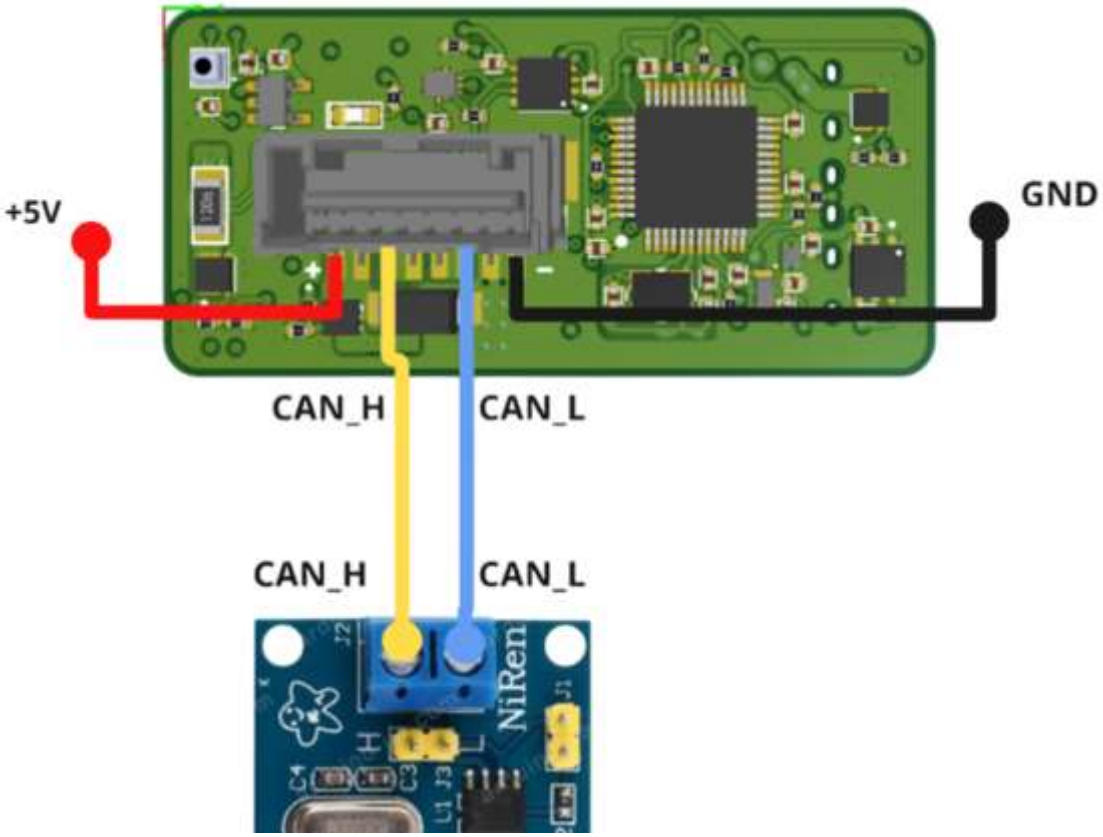

Рисунок 3.10 – Схема підключення модуля СВПШ до перетворювача CAN

На схемі (рисунок 3.11) наведено підключення перетворювача CAN-SPI на базі MCP2515 до мікроконтролера Arduino через інтерфейс SPI. Живлення модуля здійснюється від 5 В, а для передавання даних використовуються стандартні SPI-лінії: SCK (темно-зелений) підключено до піну D13 Arduino, MOSI (темно-червоний) – до D11, MISO (коричневий) – до D12, та CS (зелений) – до D9.

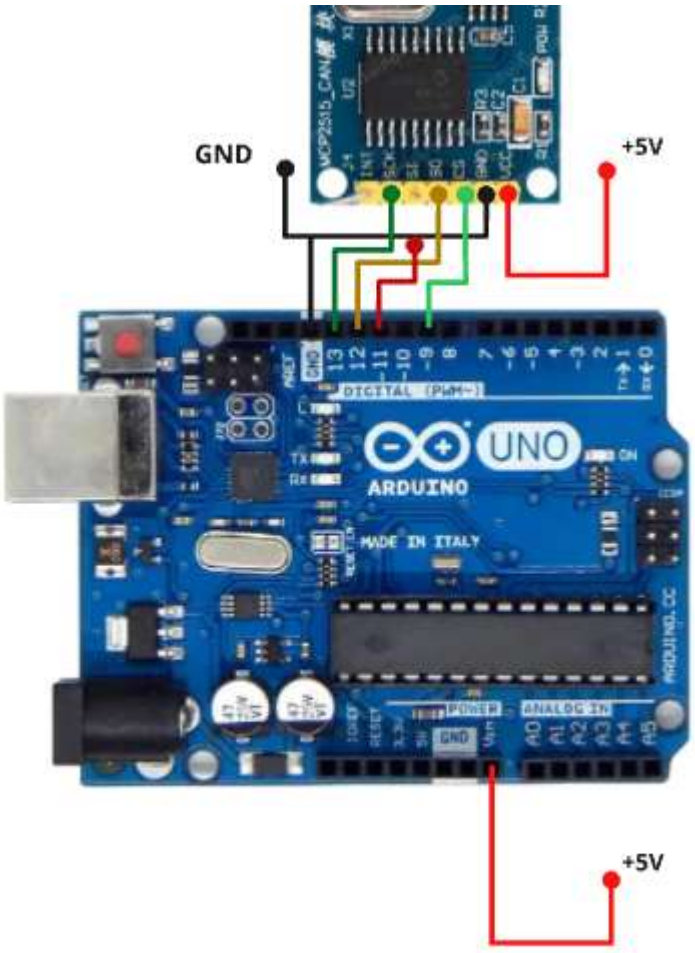

Рисунок 3.11 – Схема підключення перетворювача CAN до Arduino UNO

На схемі, зображеній на рисунку 3.12, наведено підключення електронного регулятора швидкості (ESC) iFlight BLITZ E55 до мікроконтролера Arduino. Для з'єднання використовуються три основні контакти: G (GND) – земля, підключається до загального мінуса мікроконтролера, S (Signal) – сигнальний пін, підключається до PWM-виходу (наприклад, D3), та T (Telemetry) – вихід телеметрії, який при необхідності можна з'єднати з UART-входом мікроконтролера для зчитування [36]

інформації про оберти, температуру або споживання струму. Якщо телеметрія не використовується, контакт T залишається не підключеним. Через сигнальний контакт Arduino передає керуючі PWM-імпульси до ESC для керування двигуном.

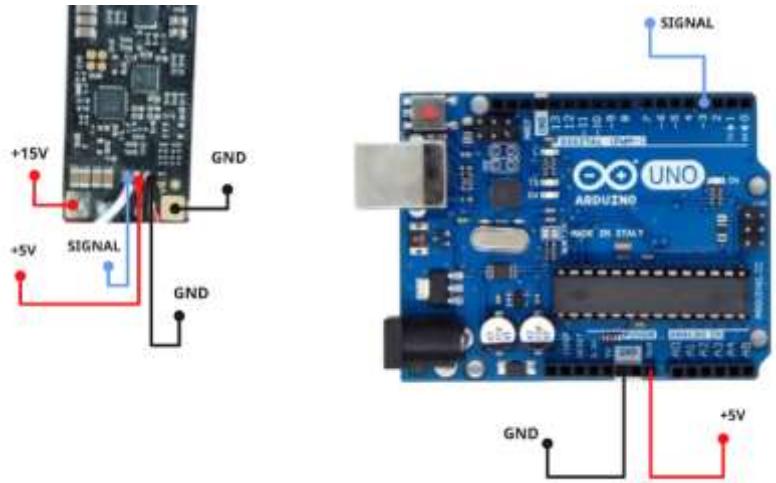

Рисунок 3.12 – Схема підключення Arduino UNO до модуля ESC

На рисунку 3.13 показано підключення ESC до безколекторного двигуна (BLDC) у послідовності, що забезпечує обертання за годинниковою стрілкою (CW). ESC має три силові виходи, які з'єднуються з трьома фазами двигуна. Напрямок обертання визначається порядком підключення цих фаз. У стандартному підключенні (без перехрестя проводів) двигун обертається у напрямку CW. Якщо необхідно змінити напрямок обертання на протилежний – проти годинникової стрілки (CCW) – достатньо поміняти місцями будь-які два з трьох дротів. Це наведено на схемі (рисунок 3.14), де вказано, які дроти необхідно перехрестити для зміни напрямку обертання [40].

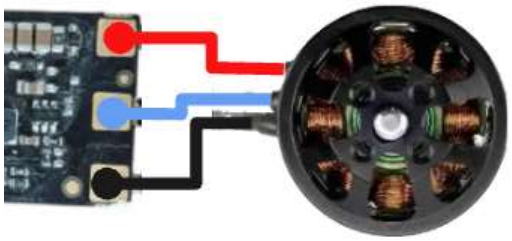

Рисунок 3.13 – Схема підключення ESC до двигуна за стандартною схемою підключення CW

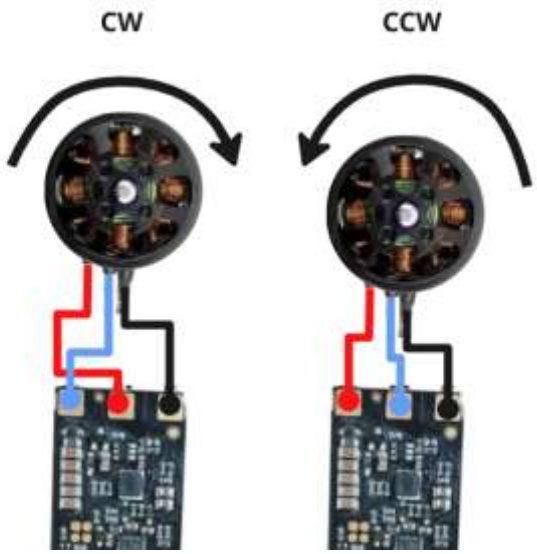

Рисунок 3.14 – Приклад схеми підключення ESC до двигуна за стандартною схемою підключення CW та за схемою CCW для зміни напрямку обертання

3.3 Алгоритм керування швидкістю двигунів БПЛА

У цьому розділі представлено алгоритм роботи системи керування, розробленої на основі платформи Arduino UNO. Алгоритм реалізує основні функції взаємодії з периферійними модулями, такими як сенсор повітряної швидкості, CAN-шина та електронний регулятор швидкості (ESC). Блок-схема основного алгоритму роботи наведена далі (рисунок 3.15), вона відображає загальну логіку роботи мікроконтролера – запуск системи та ініціалізація всіх модулів, зчитування даних із сенсора повітряної швидкості, їх обробка та передача керуючого сигналу на ESC для автоматизованого керування безколекторним двигуном (додаток Б).

Для кращого розуміння роботи системи, загальна блок-схема була розділена на кілька підсхем, які детально описують окремі функціональні частини алгоритму [12, 14].

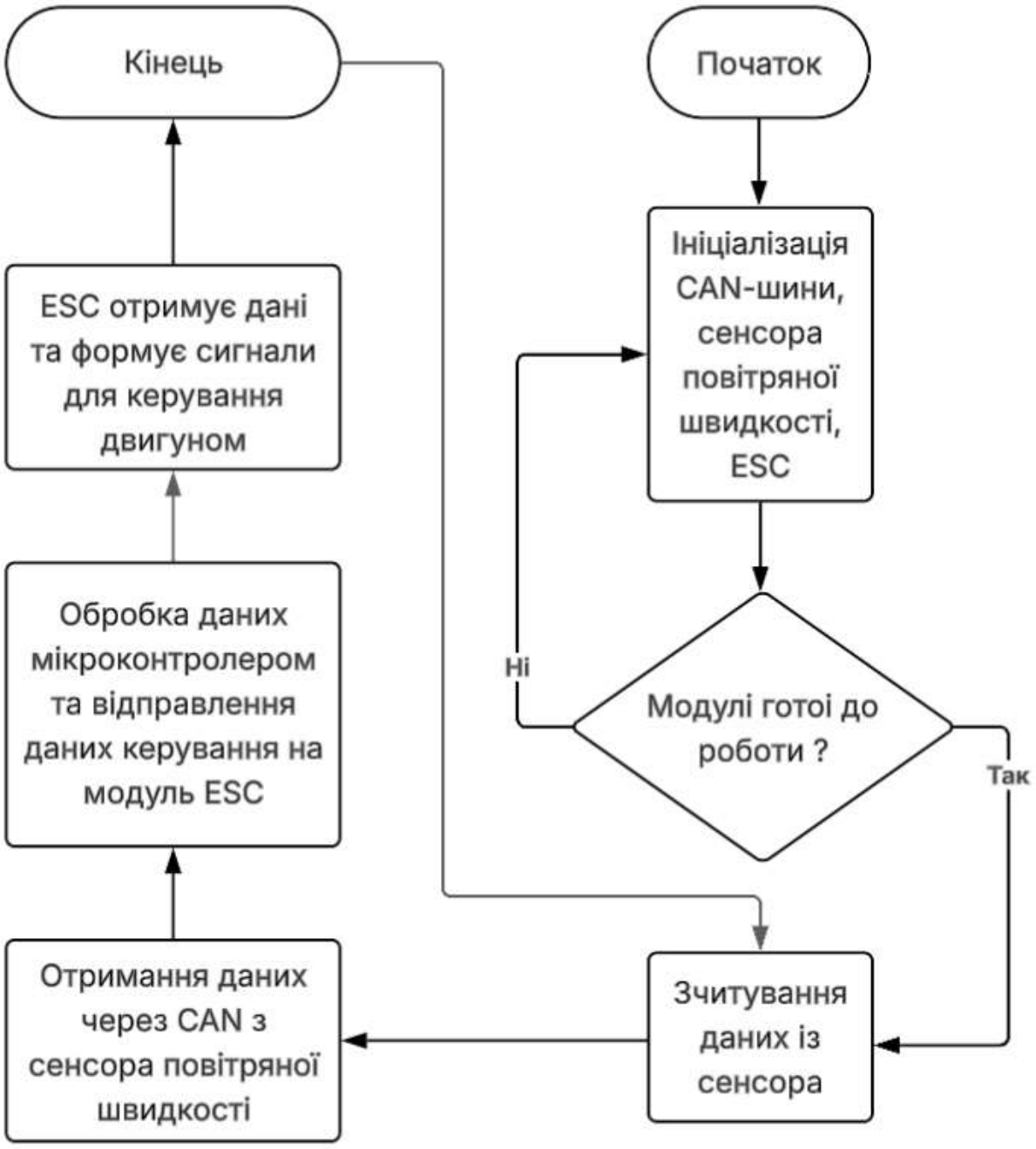

Рисунок 3.15 – Блок-схема основного алгоритму роботи

У цьому підрозділі описано детальну блок-схему (рисунок 3.16), яка описує процес передачі даних від сенсора повітряної швидкості (трубки Піто) до мікроконтролера Arduino UNO через CAN-драйвер. В модулі також є перевірка контролю суми та довжини для коректності даних.

Рисунок 3.16 – Блок-схема обробки даних CAN-модулем

Нижче, на рисунку 3.17 зображено блок-схему, яка описує процес формування керуючого сигналу на електронний регулятор швидкості (ESC) на основі даних, отриманих мікроконтролером Arduino UNO. Після перевірки та обробки значення швидкості, дані перетворюються у відповідний PWM-сигнал, який передається на ESC для керування обертами безколекторного двигуна.

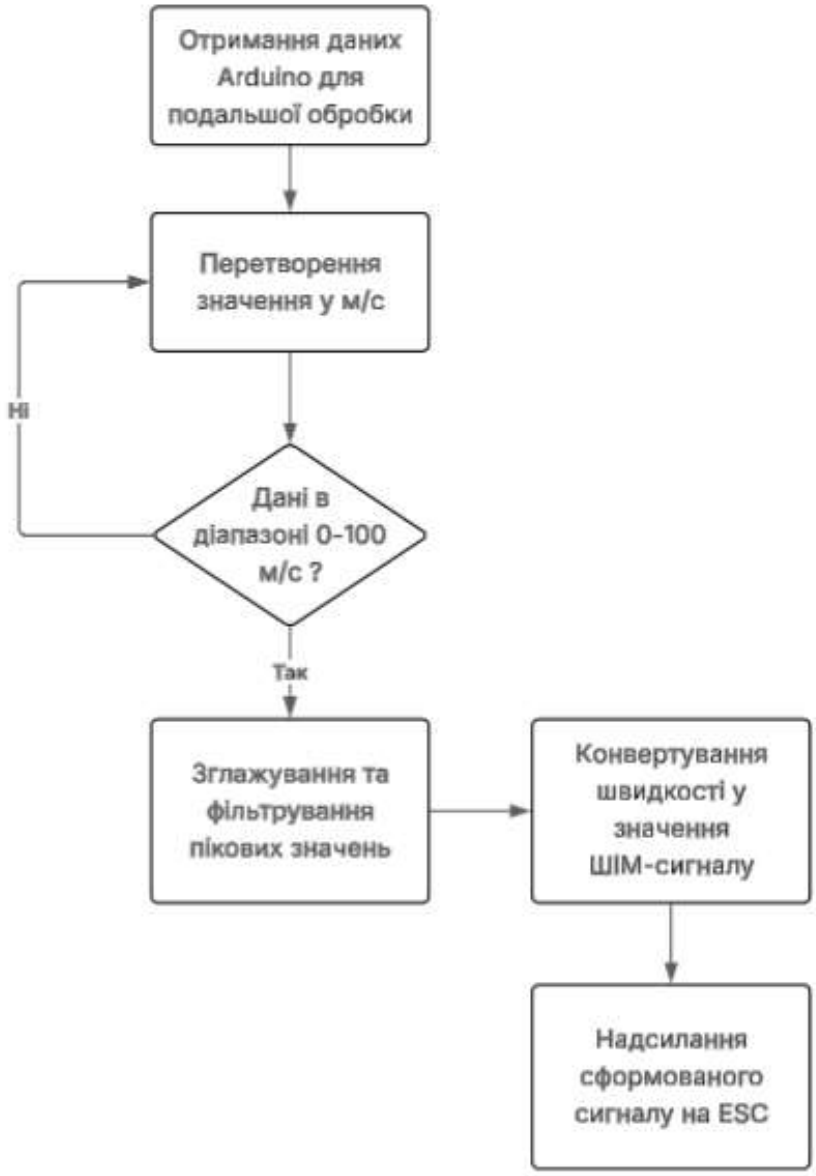

Рисунок 3.17 – Блок-схема обробки даних Arduino

У підрозділі зображено блок-схему (рисунок 3.18), яка описує процес прийому, обробки та реалізації керуючого сигналу модулем електронного регулятора швидкості (ESC). ESC отримує PWM-сигнал від мікроконтролера Arduino UNO, перевіряє його на відповідність допустимому діапазону, після чого перетворює його у команду швидкості для безколекторного двигуна. Далі сигнал обробляється вбудованим мікроконтролером ESC, який формує необхідну послідовність комутації для силових ключів, що керують живленням

фаз BLDC-двигуна. У разі виявлення перевищення струму або перегріву, система виконує захисні дії

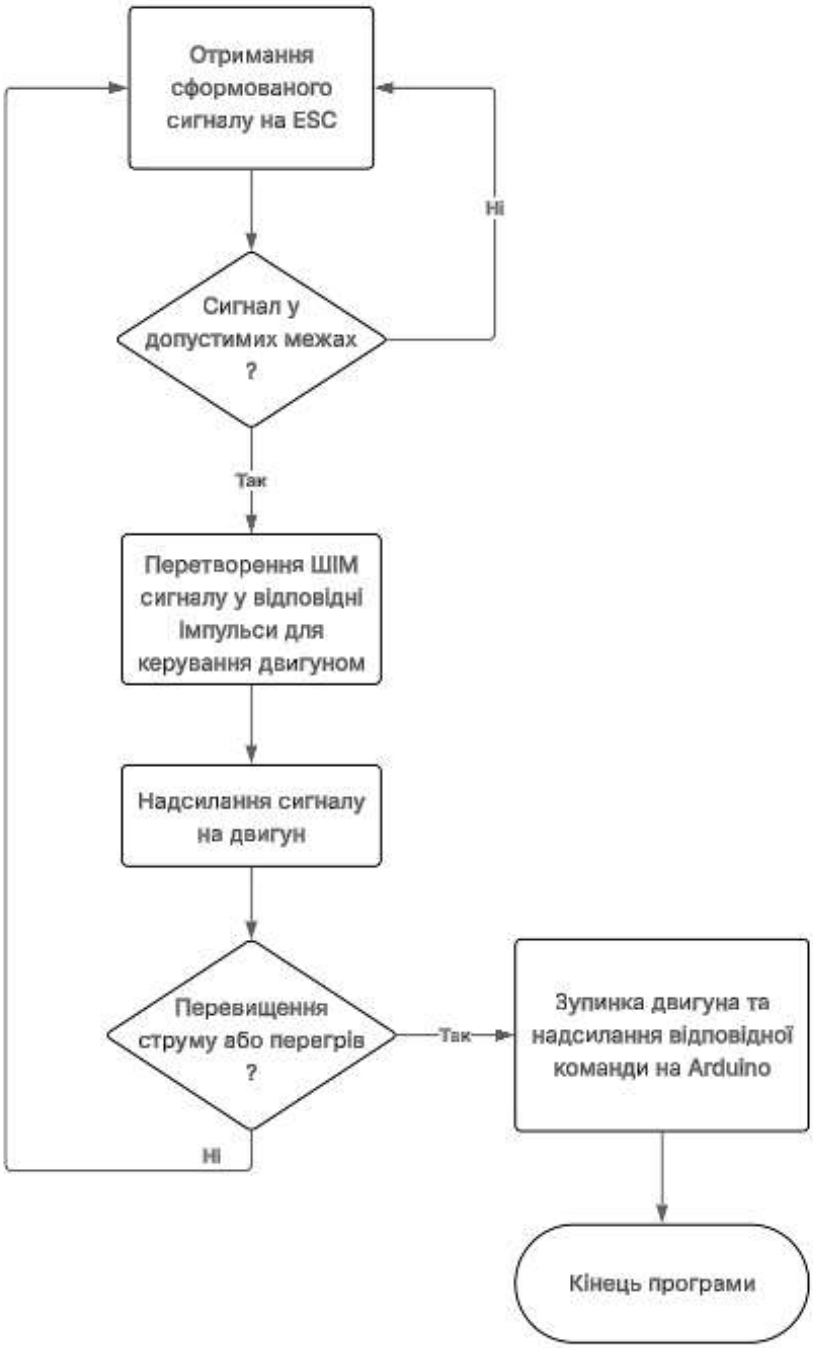

Рисунок 3.18 – Блок-схема обробки модулем ESC

У результаті побудови та аналізу трьох ключових блок-схем було детально описано послідовність обміну даними між сенсором повітряної швидкості, мікроконтролером та електронним регулятором швидкості.

Передача даних починається з того, що сенсор повітряної швидкості формує цифрове повідомлення у форматі CAN та передає його через диференціальну шину CAN_H і CAN_L. Це повідомлення надходить до CAN-трансивера TJA1051, який забезпечує перетворення сигналів фізичного рівня у логічні рівні, придатні для обробки CAN-контролером MCP2515.

Далі MCP2515 приймає це повідомлення та передає його до мікроконтролера Arduino UNO через інтерфейс SPI. Arduino виконує кілька перевірок: перевірку ID повідомлення, довжини пакету, контрольної суми (CRC) та допустимого діапазону отриманого значення. Якщо дані є коректними, вони проходять фільтрацію для усунення шуму, після чого на їх основі розраховується ширина керуючого ШІМ-сигналу.

Сформований сигнал передається на ESC, який декодує ширину ШІМ-імпульсу та відповідно до неї формує сигнали для силових ключів. Ці сигнали керують подачею струму на обмотки безколекторного двигуна, змінюючи його оберти відповідно до заданої швидкості.

3.4 Програмна реалізація системи керування швидкістю двигуна

У цьому підпункті розглянуто програмну реалізацію алгоритму керування швидкістю безколекторного двигуна з використанням мікроконтролера Arduino UNO, модуля MCP2515 для прийому даних із сенсора повітряної швидкості, та електронного регулятора швидкості (ESC). Програма написана мовою C++ у середовищі Arduino IDE з використанням бібліотек SPI, mcp_can та Servo. Програмне забезпечення реалізує зчитування даних з CAN-шини, перевірку їх коректності, обробку значення швидкості, масштабування до відповідного діапазону PWM, а також передачу сигналу на ESC (додаток В). Алгоритм також передбачає елементи базової діагностики – виявлення помилок прийому або аномальних значень [15, 13].

Оголошення бібліотек, констант і глобальних змінних (рисунок 3.19).

```
#include <SPI.h>
#include <mcp_can.h>
#include <Servo.h>

// --- Піни та параметри ---
const int SPI_CS_PIN = 10;
const int ESC_PIN = 9;
const unsigned long SENSOR_ID = 0x100;

// --- Об'єкти ---
MCP_CAN CAN(SPI_CS_PIN);
Servo esc;

// --- Таймери та безпека ---
unsigned long lastMessageTime = 0;
const unsigned long timeout = 1000;

// --- PID-регулятор ---
float targetSpeed = 20.0;
float Kp = 30.0;
float Kd = 5.0;
float previousError = 0;
float output;

// --- Фільтрація ковзним середнім ---
#define FILTER_SIZE 5
float speedBuffer[FILTER_SIZE];
int bufferIndex = 0;

// --- Програмний фільтр ---
int previousPWM = 1500;
const int pwmStepLimit = 30;

// --- Діагностика ---
int lostSignalCount = 0;
int recoveredSignalCount = 0;

// --- Стан системи ---
enum SystemStatus { OK, SENSOR_LOST, VALUE_OUT_OF_RANGE };
SystemStatus currentStatus = OK;
```

Рисунок 3.19 – Оголошення бібліотек, констант і глобальних змінних

У цій частині коду відбувається підключення основних бібліотек, що забезпечують роботу з ключовими апаратними модулями системи.

Підключається бібліотека SPI, яка дозволяє мікроконтролеру обмінюватися даними із зовнішнім CAN-контролером MCP2515 за допомогою послідовного інтерфейсу. Бібліотека mcp_can.h надає функціонал для прийому і передачі повідомлень по CAN-шині, що дає змогу отримувати цифрові дані з сенсора повітряної швидкості. Бібліотека Servo.h використовується для генерації ШІМ-сигналу, необхідного для керування електронним регулятором швидкості безколекторного двигуна.

Також визначаються параметри ПІД-регулятора, буфер для згладжування вимірюваної швидкості та змінні для контролю втрати сигналу й діагностики роботи системи.

Ініціалізація модулів у функції void setup() (рисунок 3.20).

```
void setup() {
  Serial.begin(9600);
  esc.attach(ESC_PIN);

  // Калібрування ESC (одноразово)
  Serial.println("Калібрування ESC...");
  esc.writeMicroseconds(1900); delay(1000);
  esc.writeMicroseconds(1100); delay(1000);

  // Ініціалізація CAN
  while (CAN_OK != CAN.begin(CAN_500KBPS)) {
    Serial.println("CAN не ініціалізовано, повторна спроба...");
    delay(500);
  }
  Serial.println("CAN ініціалізовано успішно");
}
```

Рисунок 3.20 – Ініціалізація модулів у функції void setup()

У цій частині програми здійснюється підготовка всієї апаратної частини системи до роботи. Спочатку ініціалізується серійний порт для виведення діагностичних повідомлень, підключається електронний регулятор швидкості (ESC) до ШІМ-виводу Arduino та проводиться процедура калібрування ESC.

Це калібрування необхідне для того, щоб регулятор швидкості міг коректно розпізнати межі керуючих сигналів, що дозволяє забезпечити правильну роботу системи у всьому робочому діапазоні обертів двигуна. Далі здійснюється ініціалізація CAN-контролера MCP2515 із встановленням швидкості обміну, а також перевіряється підключення до CAN-шини. У випадку невдалої спроби ініціалізації система повторює спроби до досягнення успіху. Після завершення цієї частини налаштувань система готова до прийому та обробки даних із сенсора повітряної швидкості.

Функція згладжування даних зображена на рисунку 3.21.

```
float getFilteredSpeed(float newValue) {
  speedBuffer[bufferIndex] = newValue;
  bufferIndex = (bufferIndex + 1) % FILTER_SIZE;

  float sum = 0;
  for (int i = 0; i < FILTER_SIZE; i++) {
    sum += speedBuffer[i];
  }
  return sum / FILTER_SIZE;
}
```

Рисунок 3.21 – Функція згладжування даних

Функція згладжування даних реалізує фільтрацію, при якій кожне нове значення швидкості додається до спеціального буфера, що зберігає кілька останніх вимірів. Далі обчислюється середнє арифметичне цих значень, і це усереднене значення використовується для подальшої обробки та керування двигуном. Це дозволяє значно зменшити вплив випадкових стрибків чи шуму в показниках сенсора, зробити реакцію системи більш плавною для керування.

Зчитування повідомлень із CAN-шини виконується за допомогою команд, зображених на рисунку 3.22

```
71   long unsigned int rxId;
72   unsigned char len = 0;
73   unsigned char rxBuf[8];
74
75   if (CAN_MSGAVAIL == CAN.checkReceive()) {
76     CAN.readMsgBuf(&rxId, &len, rxBuf);
```

Рисунок 3.22 – Зчитування повідомлень із CAN-шини

На початку кожної ітерації основного циклу програма здійснює перевірку, чи з'явилися нові дані від сенсора повітряної швидкості на CAN-шині. Для цього використовується спеціальна функція бібліотеки, яка сигналізує про наявність нового повідомлення в буфері прийому. Якщо таке повідомлення дійсно є, система переходить до етапу зчитування: отримує ідентифікатор повідомлення (ID), визначає довжину отриманих даних та копіює самі дані у спеціальний буфер для подальшої обробки.

Перевірка і декодування повідомлення показані на рисунку 3.23.

```
if (rxId == SENSOR_ID && len >= 2) {
  int rawSpeed = (rxBuf[0] << 8) | rxBuf[1];
  float airspeed = rawSpeed / 100.0;
```

Рисунок 3.23 – Перевірка і декодування повідомлення

Після зчитування нового повідомлення програма перевіряє, чи ідентифікатор отриманого пакету збігається з ID сенсора повітряної швидкості, тобто чи справді це ті дані, які необхідні для подальшої обробки. Додатково аналізується довжина пакету, щоб переконатися, чи міститься у повідомленні достатньо байтів для правильного декодування інформації про швидкість. Лише у разі, якщо обидві ці умови виконуються, система переходить до наступного кроку – декодування цифрових даних із буфера у числове значення швидкості, яке вже можна використовувати для розрахунків та керування двигуном.

Перевірка діапазону виміряної швидкості зображена на рисунку 3.24.

```
if (airspeed >= 0 && airspeed <= 30.0) {
```

Рисунок 3.24 – Перевірка діапазону виміряної швидкості

На цьому етапі система здійснює додатковий контроль якості отриманих даних, щоб уникнути впливу випадкових шумів або некоректних вимірювань, які можуть виникати через електромагнітні завади або помилки передачі по CAN-шині. Для цього перевіряється, чи потрапляє отримане значення швидкості у заздалегідь визначений фізично обґрунтований діапазон. Обробці підлягають лише ті дані, що знаходяться у межах реальних експлуатаційних параметрів, притаманних даному типу літального апарата. Якщо ж значення виходить за допустимі межі, воно ігнорується або система переходить у захищений режим роботи.

Фільтрація швидкості показана на рисунку 3.25.

```
float filteredSpeed = getFilteredSpeed(airspeed);
```

Рисунок 3.25 – Фільтрація швидкості

Щоб уникнути різких коливань у роботі системи, які можуть виникати через короткочасні шуми або незначні помилки сенсора, використовується метод згладжування вимірюваних даних. З цією метою програма зберігає кілька останніх отриманих значень швидкості у спеціальному буфері, після чого розраховується їх середнє арифметичне. Саме це усереднене значення застосовується для подальших розрахунків та керування двигуном.

Обрахунок параметрів для ПІД-регулятора приведений на рисунку 3.26.

```
// PID-регулятор
float error = targetSpeed - filteredSpeed;
float dError = error - previousError;
output = 1500 + Kp * error + Kd * dError;
previousError = error;
```

Рисунок 3.26 – Обрахунок параметрів для ПІД-регулятора

У цій частині алгоритму для керування швидкістю двигуна застосовується ПІД-регулятор, який дозволяє точно та плавно підтримувати задане значення швидкості. ПІД-регулятор враховує як поточну похибку між цільовим і виміряним значенням, так і швидкість зміни цієї похибки, завдяки чому система швидко реагує на зміни умов, але уникає різких стрибків і коливань у керуючому сигналі.

Обмеження та згладжування ШІМ-сигналу показано на рисунку 3.27.

```
// Обмеження PWM
output = constrain(output, 1100, 1900);

// Фільтр
if (abs(output - previousPWM) > pwmStepLimit) {
  output = previousPWM + (output > previousPWM ? pwmStepLimit : -pwmStepLimit);
}
```

Рисунок 3.27 – Обмеження та згладжування ШІМ

Для забезпечення плавної та безпечної роботи системи в алгоритмі реалізовано кілька рівнів захисту від різких змін керуючого сигналу. Застосовується апаратне обмеження діапазону значень ШІМ-сигналу, яке не дозволяє виходити за межі безпечних для ESC і двигуна значень. Окрім цього, використовується програмний фільтр, який додатково обмежує швидкість зміни ШІМ між сусідніми циклами роботи програми. Завдяки цьому, навіть якщо сенсор або регулятор в один момент надішле нестандартне або помилкове значення, система не передасть різкий імпульс на двигун.

Надсилання керуючого сигналу на ESC зображено на рисунку 3.28.

```
esc.writeMicroseconds(output);
previousPWM = output;
lastMessageTime = millis();
currentStatus = OK;
```

Рисунок 3.28 – Надсилання керуючого сигналу на ESC

Після завершення всіх необхідних розрахунків та перевірок система

формує керуючий ШІМ-сигнал, який передається на електронний регулятор швидкості (ESC). Саме цей сигнал визначає, з якою швидкістю буде обертатися двигун. Одночасно з передачею керуючого сигналу програма оновлює відповідні змінні стану, такі як останнє відправлене значення ШІМ та час останнього успішного прийому даних.

Команди виводу інформації для діагностики зображено на рисунку 3.29.

```
Serial.print("PWM: ");
Serial.print(output);
Serial.print(" | SPD: ");
Serial.println(filteredSpeed);
else {
currentStatus = VALUE_OUT_OF_RANGE;
Serial.println("ПОМИЛКА: Швидкість поза межами");
```

Рисунок 3.29 – Вивід інформації для діагностики

Якщо під час перевірки виявляється, що отримане значення швидкості виходить за межі допустимого діапазону або ідентифікатор повідомлення не відповідає очікуваному, система розпізнає це як помилку. У такому випадку програма виводить попереджувальне повідомлення в серійний порт для оператора або розробника. При цьому керуючий сигнал на ESC не надсилається, що дозволяє уникнути небезпечних чи непередбачуваних дій з боку двигуна.

Захист від втрати сигналу показаний на рисунку 3.30.

```
// Перевірка втрати сигналу
if (millis() - lastMessageTime > timeout) {
  if (currentStatus != SENSOR_LOST) {
    lostSignalCount++;
  }
  currentStatus = SENSOR_LOST;
  esc.writeMicroseconds(1100);
}
```

Рисунок 3.30 – Захист від втрати сигналу

Якщо система не отримує жодних нових даних із сенсора повітряної швидкості протягом більше однієї секунди, це розцінюється як втрата зв’язку або несправність у роботі сенсора чи каналу зв’язку. У такій ситуації система автоматично переходить у захищений режим, надсилаючи на електронний регулятор швидкості (ESC) мінімальний керуючий сигнал. Це призводить до зупинки або переходу двигуна у безпечний режим роботи, що дозволяє уникнути аварійних ситуацій та пошкоджень як самого апарата, так і навколишнього середовища.

Контроль відновлення зв’язку приведено на рисунку 3.31.

```
// Відновлення зв'язку
if (currentStatus == SENSOR_LOST && millis() - lastMessageTime <= timeout) {
  recoveredSignalCount++;
  currentStatus = OK;
}
```

Рисунок 3.31 – Контроль відновлення зв’язку

Якщо після перерви дані від сенсора знову починають надходити і відповідають усім необхідним критеріям коректності, система автоматично визначає факт відновлення зв’язку. У цьому випадку статус змінюється із аварійного на нормальний, і система продовжує роботу у штатному режимі: керуючі сигнали знову передаються на ESC відповідно до поточних вимірів швидкості.

Вивід статусу системи показано на рисунку 3.32, де приведено дії системи у випадках різних нештатних систуацій із даним сенсором, зокрема, втрата сигналу від сенсора, а також значення швидкості поза допустимим діапазоном.

```
  // Статус
  switch (currentStatus) {
    case OK:
      break;
    case SENSOR_LOST:
      Serial.println("⚠️ Стан: втрачено сигнал сенсора");
      break;
    case VALUE_OUT_OF_RANGE:
      Serial.println("⚠️ Стан: швидкість поза допустимим діапазоном");
      break;
  }

  delay(10);
}
```

Рисунок 3.32 – Вивід статусу системи

Вивід статусу роботи системи у серійний порт дозволяє в реальному часі відслідковувати основні події та стан кожного з компонентів. Завдяки таким діагностичним повідомленням оператор або розробник може швидко визначити, чи працює система у нормальному режимі, чи виникли проблеми, наприклад, втрата зв'язку із сенсором, некоректні дані або аварійне вимкнення двигуна.

3.5 Висновки до третього розділу

У цьому розділі було здійснено практичну реалізацію системи керування швидкістю двигуна безпілотного літального апарата з використанням платформи Arduino UNO. Розроблено електричні схеми підключення основних модулів: сенсора повітряної швидкості, CAN-контролера, електронного регулятора швидкості (ESC) та безколекторного двигуна. Надано пояснення щодо принципу підключення та взаємодії кожного модуля, а також способу зміни напрямку обертання двигуна.

Було побудовано логічні блок-схеми, що ілюструють процес обробки даних від сенсора до двигуна, включаючи передавання даних через CAN-шину, перевірку коректності, обробку значень та формування керуючого сигналу.

Програмна реалізація виконана на мові C++ у середовищі Arduino IDE. Вона охоплює ініціалізацію модулів, зчитування та перевірку повідомлень з CAN-шини, фільтрацію та згладжування швидкості, обробку даних за допомогою ПІД-регулятора, формування ШІМ-сигналу та його передачу на ESC. Також реалізовано діагностичні функції, захист від втрати зв’язку з сенсором та контроль за стабільністю роботи системи.

ВИСНОВОК

У кваліфікаційній роботі було проведено аналіз сучасних підходів до автоматичного керування швидкістю двигунів літальних апаратів, а також розглянуто можливості застосування сенсорів повітряної швидкості для підвищення ефективності та безпеки роботи БПЛА та інших повітряних платформ. На основі вивчених рішень і технологій було розроблено власну систему автоматичного керування, яка використовує дані з трубки Піто та сенсора тиску, обробляє їх із використанням фільтрації та ПІД-регулювання, та формує відповідний сигнал для електронного регулятора швидкості двигуна.

У процесі виконання роботи були обґрунтовані вибір апаратної платформи, методика підключення модулів, а також програмна реалізація основних алгоритмів збору й обробки даних. Окрему увагу приділено питанням забезпечення енергоефективності, стабільності та захисту від аварійних ситуацій, що підтверджується результатами експериментального тестування системи на реальному БПЛА. Було реалізовано такі етапи:

- розроблено та зібрано апаратну частину системи керування безпілотним літальним апаратом;
- реалізовано підключення та інтеграцію сенсорів для вимірювання параметрів польоту;
- розроблено алгоритми автоматичного керування двигуном залежно від швидкості повітря;
- створено програмне забезпечення для зчитування, обробки та аналізу даних із сенсорів;
- забезпечено обмін даними між мікроконтролером та іншими компонентами системи;
- проведено тестування системи та перевірку її роботи в реальних умовах.

Розроблена система відзначається універсальністю й можливістю інтеграції як у безпілотні, так і у пілотовані літальні апарати.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DroneCAN URL: https://dronecan.github.io/ (дата звернення: 02.03.2025).

2. CUAV SKYE URL: https://doc.cuav.net/others/skye/en/ (дата звернення: 09.03.2025).

3. MS5525DSO URL:https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FMS5525DSO%7FC1%7Fpdf%7FEnglish%7FENG_DS_MS5525DSO_C1.pdf%7FCAT-BLPS0003 (дата звернення: 11.03.2025).

4. SM5391 URL: https://www.te.com/commerce/DocumentDelivery/DDEController?Action=srchrtrv&DocNm=SM7000SM6000SM5000&DocType=Data%20Sheet&DocLang=English&DocFormat=pdf&PartCntxt=5391-BCE-S-012-001 (дата звернення: 05.03.2025).

5. ASPD-4525 URL: https://flymod.net/en/item/matek_aspd-4525?srsltid=AfmBOoriNl5s43gmazFfPeUBacglhr35OGAjH9FGsLX_sGXoYCj5hzQq (дата звернення: 09.03.2025).

6. MS4515DO URL: https://www.te.com/commerce/DocumentDelivery/DDEController?Action=srchrtrv&DocNm=MS4515DO&DocType=Data%20Sheet&DocLang=English&DocFormat=pdf&PartCntxt=4525DO-DS5AI001DP (дата звернення: 04.03.2025).

7. ASPD-DLVR URL: https://dronehub.in.ua/product/matek-aspd-dlvr-digital-airspeed-sensor/ (дата звернення: 09.03.2025).

8. DLVR-L10D URL: https://www.mouser.com/datasheet/2/18/1/DS-0300_Rev_E-1628655.pdf?srsltid=AfmBOoovhodHCJAt0SO-wdme0yMkDacxP9QlcXyPkz_1Ra-x7GIJnorb (дата звернення: 03.03.2025).

9. Фільтр Калмана URL: https://arduino.ua/art250-prostii-filtr-kalmana-priklad-z-arduino-ta-bmp280 (дата звернення: 08.03.2025).

10. Штіфзон О. Й., Новіков П. В., Бунь В. П. Теорія автоматичного

управління. КПІ ім. Ігоря Сікорського. Київ : КПІ ім. Ігоря Сікорського, 2020. 144 с.

11. Лук'янов П. В. Аеродинаміка літальних апаратів. КПІ ім. Ігоря Сікорського. Київ : КПІ ім. Ігоря Сікорського, 2024. 188 с.

12. Попович М. Г., Ковальчук О. В. Теорія автоматичного керування. К.: Либідь, 2007. 656 с.

13. Бондаренко І.М., Бородін О.В., Карнаушенко В.П. Мікропроцесорні системи контролю та керування. Харків: ХНУРЕ. 2020. 244 с.

14. Корчемний М.О., Клендій П.Б., Потапенко М.В. Теоретичні основи автоматики. Тернопіль: Навчальна книга. Богдан, 2021. 304 с.

15. Ямненко Ю. С., Осипенко К. С. Основи теорії автоматичного регулювання. Київ : КПІ ім. Ігоря Сікорського, 2017. 70 с.

16. Fahlstrom P.G., Gleason T.G. Introduction to UAV Systems, Fourth Edition. 2012. 280 p.

17. Ren Z. Air Data Estimating of A Small UAV based on Micro Pressure Sensors. 2023 35th Chinese Control and Decision Conference (CCDC). Yichang, China, 2023. P. 4842-4846

18. Видмиш А. А., Ярошенко Л. В. Основи електропривода. Теорія та практика. Частина 1. Вінниця: ВНАУ, 2020. 387 с.

19. Болюх В. Ф., Данько В. Г. Основи електроніки і мікропроцесорної техніки. Харків: НТУ «ХПІ», 2011. 257 с.

20. Anderson J.D. Introduction to Flight. New York: McGraw-Hill, 2017. 912p.

21. Spitzer C.R. Avionics: Elements, Software and Functions. New York: CRC Press, 2018. 448 p.

22. Titterton D., Weston J. Strapdown Inertial Navigation Technology. London: The Institution of Engineering and Technology, 2004. 560 p.

23. Jacob Fraden. Handbook of Modern Sensors: Physics, Designs, and Applications. New York: Springer, 2016. 758 p.

24. Bolton W. Mechatronics: Electronic Control Systems in Mechanical and

Electrical Engineering. Boston: Pearson, 2018. 712 p.

25. Dorf R. C., Bishop R. H. Modern Control Systems. Boston: Pearson, 2016. 1104 p.

26. Petru J., Singh S. Sensor Technology Handbook. Oxford: Elsevier, 2005. 608 p.

27. Incicco S., Giribet J. Colombo L. Enhanced UAV Navigation Systems Through Sensor Fusion with Trident Quaternions. 2025. P. 809-816.

28. Pratt R. Flight Control Systems: Practical Issues in Design and Implementation. IET. 2000. 382 p.

29. Khadraoui S., Fareh R., Baziyad M., Elbeltagy B. M., Bettayeb M., A Comprehensive Review and Applications of Active Disturbance Rejection Control for Unmanned Aerial Vehicles. IEEE Access. 2024. Vol. 12. P. 185851-185868.

30. Курдеча В.В., Іщенко І.О., Захарчук А.Г. Метод обробки даних в розподіленій мережі інтернету речей. Young Scientist. 2017. № 10 (50). С. 75-80.

31. Цирульник С. М., Азаров О. Д., Крупельницький Л. В., Трояновська Т. І. Програмування мікроконтролерів AVR. Вінниця: ВНТУ, 2018. 111 с.

32. Ligerko, R. Algorithm of autonomous drone path search for monitoring critical infrastructure objects. Computer-integrated technologies: education, science, production. 2025. Vol. 57. P. 168-173.

33. Співак В.М., Гуржий А.М., Нельга А.Т., Ітякін О.С. Загальна електротехніка і основи електроніки: навчальний посібник. К.: КПІ, 2020. 266с.

34. Peng D., Zhang H., Weng J., Li H., Xia F. Research and design of embedded data acquisition and monitoring system based on PowerPC and CAN Bus. 8th World Congress on Intelligent Control and Automation, Jinan, China, 2010, P. 4147-4151.

35. Ledin J., Embedded control systems in C/C++. CMP Books. 2004. 252 p.

36. Биков М. М., Черв’яков В. Д. Дискретний аналіз і теорія автоматів. Суми : Сумський державний університет, 2016. 354 с.

37. Unguritu M. G., Nichiţelea T. C.. Adaptive Real-Time Operating System in Automotive Multicore Embedded Systems. 2021 25th International Conference on

System Theory, Control and Computing (ICSTCC). Iasi, Romania, 2021. P. 150-153.

38. Артюхов В.Г., Бритов О.А., Гіоргізова-Гай В.Ш., Кирюща Б.А. Архітектура обчислювальних систем. Київ: КПІ ім. І. Сікорського, 2023. 85 с.

39. Xu J., Guo Q., Li Z. Dynamic Selection Method for Cooperative Decision-Making Center of Multi-UAV System based on Cloud Trust Model. 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). Chongqing, China, 2018, P. 922-926.

40. Трет'як А. В., Кльон А. М. Основи робототехніки. Полтава: видавництво національного університету «Полтавська політехніка імені Юрія Кондратюка», 2024. 135 с.

41. Blum J. Exploring Arduino. Tools and Technoques for engineering wizardry. Wiley, 2019. 512 p.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Єрохін Андрій Сергійович

Тема: Система керування безпілотними літальними апаратами на основі даних сенсора повітряної швидкості

Спеціальність: 151 «Автоматизація та комп'ютерно-інтегровані технології»

Обсяг кваліфікаційної роботи:

Кількість сторінок записки 67

1. Короткий зміст роботи та прийнятих рішень: метою роботи є розробка системи керування безпілотними літальними апаратами на основі даних сенсора повітряної швидкості

2. Висновок про відповідність роботи дипломному завданню: робота повністю відповідає поставленому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: у першому розділі проведено аналітичний огляд систем керування безпілотними літальними апаратами, розглянуто типи сенсорів, методи вимірювання швидкості та основи побудови автоматизованих систем. У другому розділі виконано проєктування апаратної частини системи на основі сучасного мікроконтролера STM32, сенсора повітряної швидкості та електронного регулятора обертів. У третьому розділі реалізовано алгоритми керування, обробки сигналів з використанням фільтра Калмана, ПІД-регулятора та цифрових методів фільтрації даних. Розроблено повноцінну систему керування на основі CAN-протоколу, яка може адаптувати керування двигуном в залежності від повітряної швидкості в реальному часі.

4. Позитивні сторони роботи: висока практична цінність роботи, сучасний підхід до побудови системи керування, використання реального обладнання та перевірених алгоритмів стабілізації

5. Негативні сторони роботи: у роботі недостатньо уваги приділяється аналізу альтернативних методів побудови систем керування, а також комерційних готових рішень.

6. Оцінка графічного оформлення та пояснювальної записки роботи: пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації

7. Відгук про роботу в цілому: робота виконана на належному науково-технічному рівні.

8. Інші зауваження: відсутні

9. Оцінка дипломної роботи: відмінно (4,75/A)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи)
Чешун Віктор Миколайович
канд. техн. наук, доц.
доцент кафедри кібербезпеки

"19" червня 2025 р. (підпис)

Завідувачу кафедри АКІТтаР
д-ру техн.наук, проф. Мартинюку В.В.

Єрохін Андрій Сергійович
ПІБ здобувача вищої освіти

ФІТ, 4 курс, групи АКІТ-21-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність плагіату ознайомлений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (StrikePlagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

02.06.2025р.
дата

підпис

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Андрій ЄРОХІН

Співавтор:

Назва: Єрохін антиплагіат

Експерт:

Підрозділ: Кафедра автоматизації, комп'ютерно-інтегрованих технологій та робототехніки

Коефіцієнт подібності 1:0.5%

Коефіцієнт подібності 2:0%

Мікропробіли: 6

Заміна букв: 17

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-20 04:14:20.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2025-06-20 Доцент Микола Федула

Дата експерт

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 0.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 10%**

ID: 247087 Title: БКР Система керування безпілотним літальним апаратом Added in a DB: 2025-06-19 Authors: Андрій ЄРОХІН Heads: Микола ФЕДУЛА Consultants: Opponents:	Document		Sum coincidence on the DB	
	Symbols	Lexemes	Symbols	Lexemes
	70002	603	1004 (1%)	19 (3%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ АВТОМАТИЗАЦІЇ, КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:
Назва: Система керування безпілотним літальним апаратом
Автор: Єрохін Андрій Сергійович
Спеціальність: 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітня програма: Освітньо-професійна програма «Автоматизація та комп'ютерно-інтегровані технології»
Науковий керівник: Федула Микола Васильович, доктор технічних наук, професор
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:
Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:
1) у тексті кваліфікаційної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів тощо, у назвах публікацій у переліку джерел посилання;
2) усі запозичення є фрагментарними або мають належним чином оформленні посилання;
3) виявлені модифікації тексту не впливають на відсоток схожості.
Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 0,55% і адресується до 40 джерела, що, з урахуванням наведених обгрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Завідувач кафедри — Валерій МАРТИНЮК
Гарант освітньої програми — Юрій ФОРКУН
Керівник кваліфікаційної роботи — Микола ФЕДУЛА