Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“ Дослідження та реалізація апаратно-програмних засобів отримання фотозображення, його стилізації та відтворення за допомогою реалізації пристрою IoT плоттеру”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-24М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Радін Д.О.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Дрєєв О.М.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

АНОТАЦІЯ

Радін Д.О. Дослідження та реалізація апаратно-програмних засобів отримання фотозображення, його стилізації та відтворення за допомогою реалізації пристрою IoT плоттеру. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено системи отримання фотозображення, його стилізації та відтворення за допомогою пристрою IoT плоттеру.

Метою роботи є отримання та відтворення попередньо стилізованого фотографічного зображення з використанням можливостей IoT на плотері.

Об'єкт дослідження є процес стилізації растрового зображення, перетворення його у векторну графіку й відтворення пристроєм.

Предмет дослідження є методи, програмні засоби й алгоритми отримання, стилізації та векторизації зображення.

Методи дослідження базуються на методах системного аналізу, методи цифрової обробки зображень та алгоритми машинного навчання.

Результат роботи – програмна реалізація отримання та відтворення попередньо стилізованого фотографічного зображення з використанням можливостей IoT на плотері.

У процесі проектування програмної моделі проведено аналіз сучасних апаратних рішень та програмних засобів для керування плоттером.

Розроблено інтерфейс користувача для зручного управління процесом обробки та відтворення зображень.

Програма може використовуватися на ОС Linux.

Програму розроблено в середовищі Visual Studio Code.

Ключові слова: комп'ютерна інженерія, IoT, плоттер, стилізація.

ABSTRACT

Radin D.O. Research and implementation of hardware and technical means for obtaining a photo image, its stylization and reproduction using an IoT plotter device. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the second (master's) level of higher education, software has been developed that is designed for a system for obtaining photographic images, their stylization and reproduction using an IoT plotter device.

The aim of the work is to obtain and reproduce a pre-stylized photographic image using the capabilities of IoT on a plotter.

The object of research is the process of stylizing a raster image, converting it into vector graphics, and reproducing it with a device.

The subject of research is methods, software tools, and algorithms for obtaining, stylizing, and vectorizing images.

The research methods are based on system analysis methods, digital image processing methods, and machine learning algorithms.

The result of the work is the software implementation of obtaining and reproducing a pre-stylized photographic image using the capabilities of IoT on a plotter.

In the process of designing the software model, an analysis of modern hardware solutions and software tools for controlling peripheral devices was carried out. The work describes the architecture and functional purpose of all components of the developed software for interaction with the IoT plotter.

A user interface has been developed for convenient control of the image processing and reproduction process.

The program can be used on Linux OS.

The program was developed in the Visual Studio Code environment.

Keywords: computer engineering, IoT, plotter, stylization.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

AIoT	- Artificial Intelligence of Things
API	- Application Programming Interface
ПЗ	- Програмне забезпечення
ЧПК	- Числове програмне керування
USB	- Universal Serial Bus
SPA	- Single Page Application
ОС	- Операційна система
ШІ	- Штучний інтелект

ВСТУП

Актуальність теми. Стрімкий розвиток інформаційних технологій, зокрема штучного інтелекту, суттєво вплинув на різні галузі. Паралельно продовжує активно розвиватися Інтернет речей, який охоплює мільярди фізичних пристроїв та розширює сфери свого застосування. Поєднання двох цих напрямів стало відоме як Інтелект речей(AIoT).

Поява генеративних моделей, зробила можливою трансформацію зображень в різноманітні стилі. Однак більшість існуючих рішень реалізують або програмну обробку зображень або їх відтворення.

Проблема полягає у відсутності повноцінних систем, які поєднували отримання зображення, його подальшу обробку та відтворення пристроєм.

Мета й завдання дослідження. Метою роботи є отримання та відтворення попередньо стилізованого фотографічного зображення з використанням можливостей IoT на плотері.

Для досягнення мети було поставлено такі завдання:

– провести огляд існуючих методів векторизації растрових зображень;

– порівняти існуючі моделі стилізації зображень на базі архітектури Stable Diffusion;

– програмно реалізувати захоплення та передачу зображення, стилізацію, подальше перетворення у дані придатні для відображення плотером.

Об'єкт дослідження. Процес стилізації растрового зображення, перетворення його у векторну графіку й відтворення пристроєм.

Предмет дослідження. Методи, програмні засоби й алгоритми отримання, стилізації та векторизації зображення.

Методи дослідження. Застосовуються наступні методи, а саме: методи системного аналізу, методи цифрової обробки зображень та алгоритми машинного навчання.

Наукова новизна отриманих результатів. Запропоновано архітектуру системи, поєднуючу засоби IoT, генеративні моделі й алгоритми векторизації. Удосконалено метод підготовки зображення для виведення шляхом засобів Stable Diffusion, в результаті отримуючи більш чіткий результат.

Практична цінність отриманих результатів. Реалізовано систему, що автоматично створює портрети або малюнки у довільному стилі. Рішення може бути використане на різноманітних заходах, для навчальних або рекламних стендів, а також як основа для комерційних продуктів.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Зазначена система представляє рішення класу AIoT. Традиційний Інтернет Речей фокусується на зборі та передачі даних, але з використанням засобів штучного інтелекту, з'являється шар прийняття рішень та обробки. Це дозволяє компенсувати обмежені ресурси мікроконтролерів обчислювальною потужністю хмарних сервісів, що в свою чергу забезпечить більш високу якість кінцевого результату при збереженні компактності пристрою. Система усуває проблему неповноцінних комплексів поєднуючи всі низчезазначені етапи створення кінцевого продукту.

У подібних системах пристрої часто виступають крайовими елементами. Завдяки функціоналу на базі мікроконтролерів з підтримкою Wi-Fi, пристрій плотеру може отримувати зображення з хмарних сервісів чи веб серверів. Це дозволяє уникнути прямого кабельного підключення, що робить плотер самостійним вузлом мережі. Зі свого боку алгоритми на стороні сервера виконують корекцію деталей: зміна стилю, яскравості, контрасту або видалення шумів.

Через те що пристрій захоплення також не володіє достатньою обчислювальною потужністю, після отримання даних відбувається їхня передача до потужних вузлів або хмарних сервісів. Необхідний параметр в описуваній системі – використання мікроконтролеру з вбудованою можливістю фіксувати зображення або під'єднання зовнішньої камери. Потрібно забезпечити достатню кількість пам'яті для збереження перед перетворенням у формат придатний для передачі по мережі з метою відправки до найбільш потужних вузлів. Отримані дані проходять трансформацію в дані необхідні для плотера, а потім передаються безпосередньо плотеру.

Важливою характеристикою для пристрою фізичного

відтворення – забезпечення чіткості ліній та передачі дрібних деталей. Це означає, що необхідно забезпечити швидкість, плавність та неможливість коливань. Стабільність під час створення складних контурів, використання якісних двигунів та відповідних драйверів з підтримкою мікрокрокування дозволить досягти точності близько 0.1 мм. Аби розширити можливості плотера застосовується швидка заміна малюючих інструментів без складного переналаштування за допомогою адаптивної робочої головки. В результаті окрім стандартних маркерів або ручок з'явиться підтримка інших інструментів таких як: пензлі, олівці, модулі для гравіювання. Таким чином досягається нанесення зображень на матеріали шкіри, дерева, пластику й інших поверхонь відмінних від паперу.

1.2 Область застосування

Освіта. Вищезазначений IoT плотер – лабораторна платформа для навчання. Студенти, що вивчають основи робототехніки та схемотехніки, здобувають знання у процесі складання, налагодження та експлуатації пристрою. Для засвоєння роботи електромеханічних систем, не слід забувати, що програмна складова включає в себе роботу з низькорівневим G-кодом(необхідним для роботи плотера), інтерполяцію руху. Взаємодія з API сервісами генеративних моделей допомагає краще розвинути навички та розуміння роботи сучасних застосунків.

Індустрія розваг – одна з найбільш комерційно успішних застосувань IoT плотерів. Попит на сферу організації заходів постійно зростає. Роботизовані художники трансформують нудний процес фотографування в захопливий захід. Рух маніпулятора, який крок за кроком малює портрет гостя – привертає увагу людини біля рекламного стенду, шляхом так званого ефекту "технологічного зачарування". Таке спостереження за роботою підсилює емоційний зв'язок з брендом.

За допомогою алгоритмів векторизації інтегрують елементи брендингу безпосередньо в структуру малюнку. Як приклад логотип компанії може виступати

як частина фону або вписаний в сам портрет. Отриманий сувенір носить подвійний характер. Він виступає як рекламний носій, але на відміну від листівки, одержувач зберігає його та демонструє іншим.

Відкриваються нові джерела доходу для розважальних центрів, музеїв та парків атракціонів. Автоматичне виконання процесу – від прийому платежу до завершення малюнку – зменшує кількість обслуговуючого персоналу.

Промисловість. Хоча плотери більше асоціюються з мистецтвом, технології в їх основі мають місце й в промисловості. Застосовується у невеликих майстернях або навіть лабораторіях для автоматизації задач подібних різанню або кресленню.

Існують системи настінних плотерів, наприклад Scribit. Така система перетворює приміщення на інформаційну панель. Пристрої можуть наносити на стіни будь-які написи або малюнки. “Програмований інтер’єр” підлаштовується під потреби користувача, й за можливості підключення до Інтернету, здатний виводити графіки, новини або погоду.

Промислові вертикальні UV-принтери – фактично потужні IoT-протери. Вони наносять зображення безпосередньо на скло, бетон й інші матеріали. Прогрес виконання замовлення або моніторинг рівня чорнил такої системи відбувається за допомогою модулів на базі IoT.

Медицина та соціальна складова. У медицині подібні роботи використовуються для допомоги дітям з обмеженими можливостями. Керування плотером поглядом або голосом дозволяє паралізованим пацієнтам займатися малюванням.

Невеликі роботи, наприклад Line-us, передають малюнки в реальному часі через Інтернет. Застосовуються для створення відчуття “спільної присутності”. З одного боку користувач малює на екрані свого телефону, а з іншого боку робот відтворює малюнок в будь-якому куточку світу.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

GRBL Plotter

ПЗ з відкритим кодом, розроблене на C#, орієнтоване на операційну систему Windows, відповідно використовуючи її технології Windows Forms та GDI+ для графіки. Може працювати без потужних дискретних відеокарт. Але при рендерингу складної графіки сильно навантажує центральний процесор. Програма не підтримує Unix-подібні системи. Незважаючи на це, все ж можливий запуск програми використовуючи рішення сумісності. Розглядаючи Linux, запуск відбувається за допомогою Wine або Proton. Wine призначений для запуску програм розроблених для Windows систем, емулюючи їх системні виклики. А Proton, який розроблений на його основі, додає оптимізацію й додаткові компоненти для роботи з графікою. Щодо macOS, ситуація складніша. Політика безпеки не дозволяє непідписані драйвери та доступ до портів. Через це рекомендується використовувати віртуальну машину.

Підримує розширення можливостей через механізм HTML Applications. Це доволі специфічна, але ефективна технологія Windows. Головна ідея – це об’єднати можливості HTML та JavaScript та надати повний доступ до системних ресурсів ОС. За допомогою нього ми можемо створити модулі без модифікації самого програмного забезпечення. Оскільки ми маємо доступ до системних ресурсів, ми отримуємо доступ до гілки GRBL Plotter в реєстрі та можемо задати конфігурацію, що буде використовуватися для всіх сеансів роботи. Або використовуючи генератори G- коду, копіювати його в буфер обміну, після чого програма виявляє вміст, а потім імпортує його як дані для керування. Інше

застосування – це генерація G-коду на основі введених нами даних скриптом.

Робота з обладнанням. Якщо більшість програм вимагає прямого переналаштування великої кількості для цього параметрів, то з Use Cases ми можемо створити конфігурацію під окрему задачу, а також зберегти її для повторного використання. Налаштування задаються у вигляді сценарію. До нього входять: налаштування імпорту, логіка підйому або опускання, швидкість, специфічні команди старту та завершення. Налаштування імпорту описує правила перетворень векторних даних при завантаженні файлів. Різні формати файлів мають різні системи координат або одиниці вимірювання. Цей процес масштабує розміри зображення. Логіка підйому або опускання відподає за налаштування контакту між інструментом та робочою поверхнею. Через те, що може використовуватися різне обладнання, нам потрібно напряму задати глибину обробку та параметрів накшталт потужності. Ігнорування параметру призведе до зайвих ліній або навіть пошкодження матеріалу. Плавність та гальмування при зміні напрямку руху задається параметрами швидкості та прискорення. Некоректно задані параметри проявляються у зниженні точності. Специфічні команди старту та завершення додаються в кінець та початок G-коду. Команди початку відповідальні за режими роботи контролеру, а також за підготовку інструмента до завдання. Команди завершення безпечно переводять інструмент у початкове положення.

Інтерфейс програми трохи ускладнений, що може відлякати недосвідченого користувача. Розглянемо найцікавіші з його функцій. Програма має в арсеналі інстумент 2D View. Цей інстумент відображає траєкторії руху плотера за завантаженим G-кодом. G-код – це мова програмування, що описує команди для руху плотера та керування його механізмами. За допомогою функції ми можемо не тільки попередньо переглянути результат, але й редагувати ці траєкторії. Ми можемо за допомогою миші виділяти окремі сегменти або групи, й геометрично перетворювати, переміщувати, масштабувати, обертати тощо. Інстумент додатково виконує вирівнювання об’єктів за сіткою координат, що

також включена до нього. Для оптимізації роботи плотера ми можемо переглянути всі переміщення. Наприклад холостих переміщення позначаються пунктирними лініями, а робочі рухи – суцільними. На основі цього ми можемо оцінити наскільки наш G-код оптимізований.

Інстумент Control Pad дозволяє нам ручним режимом керувати апаратною частиною плотера. Тобто ми можемо задати йому позицію, змінити калібрування й підготувати пристрій для завдання. Він має деякі підфункції. Virtual Joystick – здійснює покроковий рух або безперервний, що задається мишею. Рух відбувається як у двовимірному просторі так і в тривимірному. При бажанні, ми можемо його перетворити на повноцінний пульт керування, через підключення зовнішніх пристроїв через шину. Для швидкого переміщення існують гарячі клавіши для кожної функції. Кожен користувач може змінити конкретну клавішу на свій розсуд. Відображення швидкості, частоти обертань інструменту, сигналів контролера забезпечує Digital Read Out. Крім цього відображає поточне положення в двух осях.

Ще більше контролю. Для відчуття більшого контролю, ми можемо інтегрувати камеру спрямовану вниз під робочу поверхню та відобразити її зображення на координати сітки у області overlay. Ми можемо співставити зображення та її G-code траєкторії. Здебільшого використовується для функції корекції. Коли виявлені деформації, програма використовує зображення камери для корекції.

Камера, як засіб отримання зображення та стилізація. Камера не передбачена, як засіб откримання вхідного зображення. Відсутня підтримка його перетворення у траєкторії малювання. Використовується тільки для описаних вище цілей. Це стосується і засобів стилізації. Програма орієнтована тільки на роботу з плотером. Тому не містить вбудованих для цього засобів. Будь-яке перетворення зображень ми маємо виконати у сторонніх програмах до імпорту даних у програму.

CNCjs

Це ПЗ для роботи через браузер для керування ЧПК-пристоями з відкритим вихідним кодом. Якщо попередня програма була створена для конкретної ОС й прив’язана до локального обладнання, то дана програма використовує модель клієнт-сервер. Модель реалізована на Node.js. Це середовище, що виконує JavaScript код за межами браузера.

Модель клієнт-сервер. Взаємодія з пристроєм відтворення відбувається через сервер. Архітектура Node.js паралельно обробляє команди, отримує підтвердження від плотера та обробляє дії користувача не блокуючи виконання. Сервер взаємодіє з контролером плотера за механізмом послідовного обміну даними. Для цього використовується бібліотека низькорівневого доступу до інтерфейсів USB. Використовується власний механізм буферизації та виконання G-коду. Під час роботи, сервер враховує буфер для команд мікроконтролерів та не переповнює його, формується керований потік даних.

Для роботи з клієнтом необхідна наявність API. Через нього програма приймає запити від застосунків клієнта, а потім в асинхронному режимі передає інформацію про різні стани через мережеві протоколи. Ми можемо отримати доступ до керування виконання G-коду. Через застосунок можна запускати, зупиняти, передавати нові команди.

Клієнт побудований у вигляді SPA. Працює в браузері і через нього ми можемо взаємодіяти з сервером. Основа застосунку це бібліотека React. Тому зникає необхідність перезавантаження сторінки при кожному новому запиті. Підтримується режим багатьох користувачів. З сервером може взаємодіяти декілька клієнтів одночасно. Сервер формує та синхронізує між клієнтами стан системи. Наприклад якщо ми натиснемо кнопку зупинення виконання програми ця дія відобразиться іншим. Комунікація використовує технологію WebSockets. Забезпечується канал обміну даними, при якому на відміну від HTTP запитів, сервер передає інформацію клієнту без попереднього запиту з браузера. Створюється тунель через який ми завжди маємо актуальну інформацію та

передаємо її без затримок.

Коли немає потреби в клієнт-серверній моделі, програма надається у вигляді застосунку. Реалізований за допомогою Electron. Це фреймворк, який використовується для побудови десктопної програми мовою програмування JavaScript за межами браузера. Ми фактично запускаємо застосунок, як окремий браузер. Саме це робить програму незалежною від конкретної платформи. Таким чином серверна частина запускається локально, а клієнт – це браузерний застосунок. Така версія програми надає доступ до ОС. Стає доступною файлова система чи графічне середовище користувача.

Інтерфейс користувача працює за принципом модулів. Складається з незалежних компонентів у вигляді віджетів. Інтерфейс підлаштовується як під невеликий екран смартфону так і великі монітори. Кожен віджет представляє окрему взаємодію з системою й не зв'язаний з іншими інструментами. Система підтримує зміну інтерфейсу без впливу на логіку керування обладнання, тому ми можемо підлаштувати його під свій смак. Розглянемо найголовніші з цих віджетів.

Connection Widget – виконує функцію керування між серверною частиною та контроллером пристрою. Через нього ми ініціюємо з'єднання та контролюємо з'єднання. Ми можемо задати швидкість передачі даних, сумісну з технічними характеристиками мікроконтролера. Взагалі рекомендується вимикати функцію керування потоком. Як вже було згадано, програма використовує власне рішення для керування буфером мікроконтролера та автоматично регулює швидкість через запит-відповідь з контролером. При модифікації параметрів цієї функції треба бути обережним, через те що можна викликати затримки при низькій швидкості та нерівномірну роботу плотера при великій швидкості. До функцій також входить вибір порту пристроїв, автоматично виявлених сервером. Відбувається сканування інтерфейсів вводу-виводу і формується список активних портів. Від нас вимагається чітко обрати порт до якого фізично підключено плотер.

Ручне керування інструментом відображення ми здійснюємо віджетом Axes Widget. Використовується для роботи з координатною системою. Він постійно

відображає їх у двох системах відліку, у машинній та робочій. За першою ми контролюємо абсолютне положення, тобто визначаємо межі робочого поля, а за другою, локальні координати конкретного завдання, наприклад лівий кут аркуша. Точність відображення до трьох знаків після коми.

Інструмент Jogging використовується для ручного переміщення за допомогою якого ми можемо рухати плотер без повноцінної програми G-коду. Ми можемо точно переміщувати його між різними ділянками та задавати режими роботи, подібні до швидкості. Також реалізована підтримка гарячих клавіш для рутинних задач. За допомогою них ми можемо не взаємодіяти з графічним представленням елементів інтерфейсу.

Для графічного відображення траєкторій руху плотера ми можемо використати Visualizer Widget. Візуалізатор працює на стороні клієнта. Це зменшує навантаження на серверну частину, але комп'ютер без графічного адаптеру знижує продуктивність, в таких випадках рекомендується вимикати функцію. При ввімкненні ми отримуємо тривимірне представлення даних G-коду, та можемо візуально контролювати процес обробки на будь-якому етапі. Передбачена інтерактивність. Перед запуском програми запускається перегляд траєкторій. G-код інтерпретується клієнтом та відображається у просторі тому ми можемо заздалегідь виявлити помилки, наприклад вихід за робочу межу або неправильне налаштування. Має апаратне прискорення графіки. Через це навіть при складних траєкторіях руху зберігається плавність при використанні. Коли ми переглянули результат та впевнились у відсутності помилок й запустили виконання, віджет починає позначати поточне положення інструмента. Позначається віртуальним маркером, який синхронізується з рухом пристрою. Після початкового перегляду, ми контролюємо хід виконання програми в будь-якому місці.

У разі коли нам потрібно виконати окрему команду G-коду без запуску програми, ми можемо використати **консоль**. Всі введені команди передаються на сервер, а з нього до контролера. Ми можемо переміщувати перо, змінювати або

встановлювати режими. Після виконання команди, контролер повертає відповідь, яку можна переглянути у консолі.

Підтримка камери та стилізація. Як і в GRBL Plotter, камера застосовується лише для моніторингу процесів. Цю функцію виконує окремий віджет. Щодо стилізації, вбудованих для цього інструментів немає. Ми можемо користуватися програмою лише як дуже зручним відправником G-коду.

LightBurn

Комерційне ПЗ для керування лазерними гравірувальниками. Зазвичай такі програми орієнтуються на ОС Windows, але дана програма доступна як для Windows так і macOS та Linux. Вимагає наявність ліцензії. Для використання нам потрібно здійснити одноразову покупку або спробувати пробний період, що надає доступ на 30 днів. Одна ліцензія дозволяє активувати до 3 комп'ютерів. Для роботи програми не потрібен сервер або мережева взаємодія, всі процеси відбуваються локально. У цій програмі нас цікавить взаємодія з камерою та обробка растрових зображень.

Якщо попередні програми використовували камеру для візуального контролю, то тут це повноцінне джерело даних. Коли ми налаштовуємо камеру відносно робочого столу, програма встановлює відповідність між зображенням та фізичним обладнанням. Зображення накладається у вигляді фону, по якому буде працювати інструмент. Після початку роботи, ми маємо можливість оновлювати зображення з камери у будь-який момент. В цей момент змінюється фоновий малюнок, а всі розміщені траєкторії залишаються.

Для растрових зображень ми можемо використати вбудовані інструменти стилізації для гравіювання. За допомогою цих інструментів виконується: базова корекція зображення, його яскравість, контраст, гама тощо. Також використовуються інструменти підвищення якості зображення. До них входить підсилення різкості та зменшення шуму. Для відображення лазером, зображення нам необхідно спочатку привести у відповідну форму. Хоча програма не має повноцінних методів стилізації, вона пропонує кілька варіантів:

– Бінаризація. Один з найпростіших методів. Кожен піксель порівнюється за заздалегідь заданим пороговим значенням яскравості. Якщо піксель перевищує поріг, то перетворюється на білий, якщо навпаки, то на чорний;

– Стохастичне дизерування. Імітує напівтони через розподіл чорних та білих точок залежно від локальної яскравості. Відрізняється від бінаризації збереженням переходів сірого;

– Контурний ескіз. Перетворює зображення на контури. Результат схожий на малюнок рукою, зі зменшеною кількістю деталей та чіткою формою основного об’єкта.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Перед відтворенням зображення плотером, нам необхідно правильно підібрати генеративну модель. Архітектура моделей хоч і має однакову основу, але мають різні показники втрати інформації, складність зображення на його візуальний стиль. Складні текстури, градієнти та шум растрових зображень не придатні для плотер, потому що він працює лише з контурними зображеннями.

Для запуску моделей нам знадобиться відповідне програмне середовище. **Stable Diffusion WebUI (AUTOMATIC1111)** – це графічний інтерфейс з відкритим кодом для керуванням та запуском моделей Stable Diffusion у браузері. Однак наша система включає взаємодію з API сервісом для автоматизії процесу стилізації. Для досягнення цілі, нам необхідно запустити цей GUI у вигляді сервісу. Для цього ми відредагуємо файл конфігурації та додамо відповідні параметри запуску. Після цього інтерфейс готовий отримувати дані та команди через запити свої кінцеві точки.

Ми виконаємо порівняльний аналіз доступних генеративних моделей на базі Stable Diffusion. **Stable Diffusion** – це нейромережа глибокого навчання для перетворення тексту у вображення з відкритим кодом. Використовується зазвичай

для модифікацій існуючих зображень. Вибір моделі зумовлений тим, що код та ваги моделі можна знайти у відкритому доступі, а також можливістю використовувати на локальному ком'пютері. Ми не сплачуємо за кожну генерацію зображення, як при використанні оглянутих пропрієтарних аналогів. А також забезпечити незалежність від інтернет з'єднання.

При порівнянні були використані наступні моделі: RealisticVision v5.1, Anything v5, DreamShaper v8. Ми розглянемо кожну з них, виділимо сильні та слабкі сторони, а також визначимо їх придатність для відображення плотером.

RealisticVision v5.1

Вузькоспеціалізована модель для генерації реалістичних зображень, особливо портретів, з метою імітації фотографічної якості. Створює портрети з керуванням навколишнього середовища а такой імітацією параметрів камери. Щодо використання моделі у програмному середовищі, модель підтримує налаштування параметрів генерування, тобто є можливість задавати розміри зображення, кількість кроків виведення та шкалу відповідності запиту або додатково створюється запит англійською мовою для більшого контролю над вихідним зображенням. Для запитів іншою мовою необхідно використовувати інше програмне середовище, що перекладає запит на англійську. Текстурна складність таких реалістичних зображень високої якості є проблемою для подальших етапів виділення контурів та векторизації. Через її високий рівень деталізації та м'які переходи кольорів, вихідний малюнок складно перетворити на чіткий контурний, що може призводити до втрати важливих деталей чи рис або великої кількості небажаного шуму в векторному файлі. Також модель не підходить для генерування зображень інших стилів.

Anything v5

Створена шляхом об'єднання багатьох моделей мультсеріального стилю для досягнення високої якості ілюстрацій. На відміну від попередньої моделі, навчена на великій кількості ілюстрацій, більшість з яких мають чіткі контури, що спрощує подальшу обробку вихідного зображення. Але для досягнення саме

контурного малюнку, вимагає точного та детального текстового запиту, інакше буде отримано растрове зображення. При точному запиті результат характеризується чистими лініями та визначеними областями, що набагато краще підходить для перетворення на контурний малюнок для відтворення на плоттері.

Важливий вибір версії. Версія V5-Prt (pruned) є найбільш рекомендованою через її точність та сумісність з моделями LoRA(Low-Rank Adaptation), які дозволяють швидко й ефективно донавчати великі нейронні мережі без зміни всієї моделі. З іншої сторони, старіша версія V3.2++ має проблеми сумісності, які можуть призвести до низької якості зображень.

DreamShaper v8

Універсальний інструмент. Створює як фотореалістичні, так і художні стилі. Альтернатива комерційним моделям, таким як MidJourney. Існує кілька версій моделі, серед яких версія 8 найкраще обробляє фотореалізм. Пропонує збалансований підхід. Модель стилізує зображення з певним рівнем реалізму, але її також можна налаштувати для створення більш художнього результату, який легше відтворити на плотері. Через це може бути складно досягти чистого реалізму або ілюстраційного стилю порівняно з вузькоспеціалізованими моделями, але гнучкість дозволить нам отримати різноманітні результати від одного інструменту.

Апаратні обмеження

Основним апаратним обмеженням для запуску переглянутих моделей є обсяг відеопам'яті графічного процесора. Для моделей на базі Stable Diffusion 1.5 мінімальна вимога це 4 ГБ, а рекомендований діапазон становить 4-6 ГБ для генерації зображень розміром 512x512 пікселів

При використанні функцій масштабування або розширень, таких як ControlNet, рекомендується 8-12 VRAM на графічному процесорі серії NVIDIA RTX.

Крім відеопам’яті необхідно мати достатній обсяг системної оперативної пам'яті – рекомендовано 16-32 ГБ та SSD-накопичувач обсягом 20-50 ГБ для моделей та програмного забезпечення.

Таблиця 2.1 – Порівняння генеративних моделей

Модель	Ключові особливості	Сценарій використання	Рекомендації по VRAM
RealisticVision v5.1	Спеціалізована на реалістичних зображеннях. Висока текстурна складність.	Стилізація з подальшим перетворенням на контурний малюнок. Вимагає значної обробки.	6-8 ГБ+
Anything v5	Можливість генерування як растрового так і контурного зображення художнього стилю.	Створення персонажів та ілюстрацій, які легко перетворюються на контурні лінії.	4-6 ГБ+
DreamShaper v8	Здатна змішувати різні стилі.	Експерименти зі стилізацією. Пошук балансу між деталізацією та чистотою ліній.	4-8 ГБ+

Невідповідність цим умовам може спричинити тривалий час генерації або неможливість запуску моделі локально. Хоча за допомогою параметрів AUTOMATIC1111 Stable Diffusion Web UI можна запускати модель у режимі низької або середньої VRAM, але як правило, знижує якість зображення або призводить до нестабільної роботи моделі.

Stable Diffusion WebUI підтримує розширення. Ми будемо використовувати розширення **ControlNet**. Оглянуті нейромережі мають суттєвий недолік: вони не забезпечують контроль над структурою зображення. Через це можуть бути присутні зміни у вихідному зображенні, такі як зміна положення або рис об’єкту. ControlNet виключає цей недолік, використовуючи архітектуру мережі, що розуміє

контури об’єкта.

Нас цікавить модель **lineart_anime** цього модуля. По-перше модель дозволяє нам робити стилізацію у художньому стилі. По-друге вона ігнорує текстури або тіні, та стилізує саме риси об’єкта. Результатом є чіткі контури об’єкта на білому фоні. Визначимо головні переваги цього методу:

– зменшення часу відображення плотером за рахунок неперервних ліній, що впливає на кількість холостих ходів плотера;

– на основі таких ліній краще будуються вектори без фільтрації;

– лінії рівномірної товщини, що краще відображаються плотером.

Python – це інтерпретована, об’єктно-орієнтована мова програмування високого рівня загального призначення. Вона має неймовірно велику кількість бібліотек та готових інструментів. Наша система одночасно працює з відеопотоком USB, відправляє запити на REST API та відправляє команди мікроконтролеру плотера. Python має унікальну здатність об’єднати ці абсолютно різні процеси в один, при цьому не ускладнюючи архітектуру нашої системи.

По-друге використання мови скорочує час розробки через свій висоий рівень абстракції та динамічній типізації. Існують оптимізовані бібліотеки обчислювальних оптимізацій для задач. Ми зможемо зосередитися на експериментальній перевірці алгоритмів, а не оптимізації низькорівневих задач. По-третє програмне забезчення не залежить від ОС. Якщо ми написали один раз програму, то можемо використати її на різних платформах за умови, що не пишемо код використовуючи конкретні функції ОС.

2.3 Розгорнута постановка завдання

Для досягнення мети необхідно виконати наступні завдання:

– проаналізувати програмні рішення для захоплення зображень, їх обробки та відтворення плотером;

– дослідити стилізацію зображень із використанням генеративних моделей

штучного інтелекту класу Stable Diffusion та методів керування генерацією;

– розробити архітектуру системи, що об’єднує захоплення зображення, графічний інтерфейс користувача, обробку зображень та контролер плоттера;

– реалізувати програмно захоплення зображення з камери та інтегрувати в користувацький інтерфейс;

– реалізувати програмно взаємодію з Stable Diffusion WebUI для стилізації зображень;

– забезпечити перетворення стилізованих растрових зображень у векторне представлення, придатне для подальшої генерації керуючих команд;

– розробити механізм формування G-коду на основі векторизованих даних та реалізувати передачу команд керування до плоттера;

– забезпечити візуалізацію проміжних та кінцевих результатів роботи системи у графічному інтерфейсі користувача;

– розробити рекомендації по впровадженню в промислову експлуатацію;

– визначити економічну ефективність системи;

– розробити заходи по охороні праці та цивільного захисту;

– сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

3.1 Опис функціонування системи

Функціонування запропонованої системи представляє собою циклічний конвеєр обробки даних, який об’єднує методи отримання первинної візуальної інформації, її візуальну трансформацію та механічне відтворення. Загальний алгоритм роботи системи розділено на чотири етапи:

– Захоплення зображення.

– Стилізація зображення.

– Векторизація.

– Відображення.

На першому етапі ми здійснюємо ініціалізацію модуля камери. Вона, функціонуючи як окремий вузол, виконує захоплення растрового зображення. Ми можемо звертатися до вбудованої камери обчислювальної станції або зовнішньої USB-периферії через системні драйвери та інтерфейси API засобами бібліотеки OpenCV.

Другий етап полягає в обробці отриманого кадру нейромережами. Програмне забезпечення сервера, реалізоване мовою Python, відправляє отримані дані до API Automatic1111. На цьому етапі ми використовуємо модель Stable Diffusion 1.5 у поєднанні з додатковими модулями керованої генерації:

– ControlNet.

– LoRA-моделі.

LoRA-моделі фокусують генерацію на отриманні специфічного стилю “Line Art”, що важливо для роботи плотера, тому що вихідне зображення виходить з чіткими лініями та мінімальною кількістю шумів, які легше відобразити.

На третьому етапі виконуємо постобробку згенерованого стилізованого зображення. За допомогою бібліотеки OpenCV здійснюється бінаризація та

фільтрація дефектів. Отриманий растр піддається алгоритму векторизації, що перетворює масив пікселів у набір ліній та кривих. В результаті формується файл з розширенням .gcode, який містить послідовність команд для переміщення пера плоттера у тривимірному просторі.

На фінальному етапі ми передаємо згенеровані команди на контролер плоттера. Контролер інтерпретує G-код, керує кроковими двигунами для точного фізичного відтворення стилізованого малюнка на носії.

3.2 Розробка структурної схеми

Структурної схема вказує взаємозв'язки між компонентами, розподіл обчислювального навантаження та протоколи інформаційного обміну. Відповідно до поставленого завдання, архітектура системи побудована за модульним принципом, для стабільності передачі даних між різнорідними середовищами. Структурна схема системи зображена на малюнку 3.1. Розглянемо детальніше її елементи.

Блок отримання візуальних даних

Здійснює роль сенсорного входу системи. Схема передбачає універсальність інтерфейсу підключення. Це дозволить нам використовувати як автономні IoT-модулі, наприклад, ESP32-CAM, так і стаціонарні пристрої відеозахоплення, як інтегровані камери. Єдине завдання цього блоку – це формувати стабільний кадровий потік або знімків та їх передача у центральний вузол обробки. Відео транслюється одразу в оперативну пам'ять керуючого комп'ютера, минаючи проміжні буфери. Через це ми зможемо мати можливість інтерактивно змінювати композицію кадру, параметри експозиції та позицію об'єкта ще до моменту фіксації зображення, нівелює вірогідність отримання некоректних вхідних даних. Вихідними даними блоку буде растрове зображення у форматі високої чіткості для подальшої стилізації.

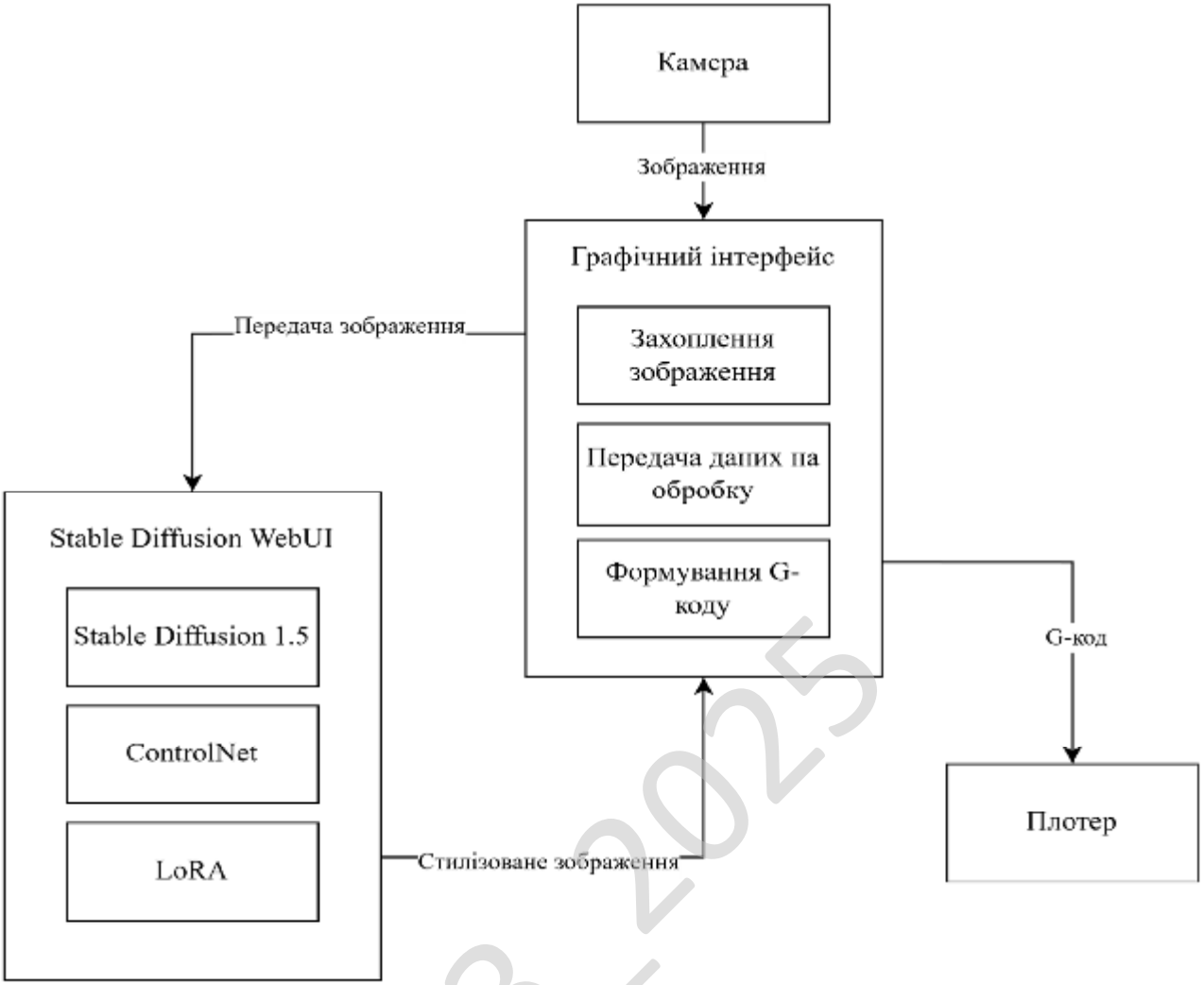

Рисунок 3.1 – Структурна схема системи

Графічний інтерфейс

Посідає центральне місце у структурній ієрархії. Це вузол графічного інтерфейсу та керування, який виступає архітектурним ядром системи. Якщо традиційні клієнтські додатки виконують лише роль терміналу введення-виведення, то даний вузол – це повноцінний оркестратор асинхронних процесів. Внутрішня логіка керуючого вузла включає підсистему попередньої обробки відеосигналу, модуль векторизації та драйвер комунікації з периферійними пристроями. Тобто цей блок керує:

– Захопленням зображення.

– Передача даних на обробку.

– Формування G-коду.

Таким рішенням ми можемо інкапсулювати складність керування різнорідним обладнанням всередині однієї програми, надаючи користувачу інтуїтивно зрозумілий інтерфейс високого рівня абстракції. Через реалізацію багатопотокової моделі обчислень, керуючий вузол гарантує швидкодію інтерфейсу навіть під час тривалих операцій.

Модуль інтелектуальної трансформації

Цей блок винесено в окремий блок через високу обчислювальну складність операцій. Крім того така архітектура дозволяє нам розвантажити основний потік виконання та забезпечити масштабованость системи. Він представляє собою серверне середовище, де відбувається дифузійна обробка зображення.

Внутрішня архітектура блока складається з трьох інтегрованих рівнів. Найголовніший рівень – це базова генеративна модель Stable Diffusion 1.5, яка відповідає за формування загальної текстури та композиції на основі ймовірнісної дифузії. Для адаптації цієї моделі до задач інженерної графіки застосовано адаптер ControlNet. Додатково застосовується стилізаційний модуль LoRA, що застосовує специфічний художній стиль для моделі без повної модифікації ваг нейромережі.

Плотер

Останній блок структурної схеми, для фізичного втілення початкового растрового зображення. Вхідні дані для цього блоку – це послідовність команд G-коду. Плотер має у собі мікроконтролер керування двигунами, драйвери силової частини та безпосередньо механіку. Має архітектурною особливість. Це наявність буфера команд із функцією попереднього перегляду. Це дозволяє нам згладити надходження даних та забезпечити неперервність руху інструменту. Результатом функціонування цього блоку є точне відтворення графічних ліній на фізичному носії.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Дана схема вказує окремі етапи виконання від захоплення зображення до його відтворення, їх залежності від попередніх етапів. Кожен блок виконує свою конкретну роль у функціонуванні системи. Для розуміння всіх функцій, розглянемо кожен з них.

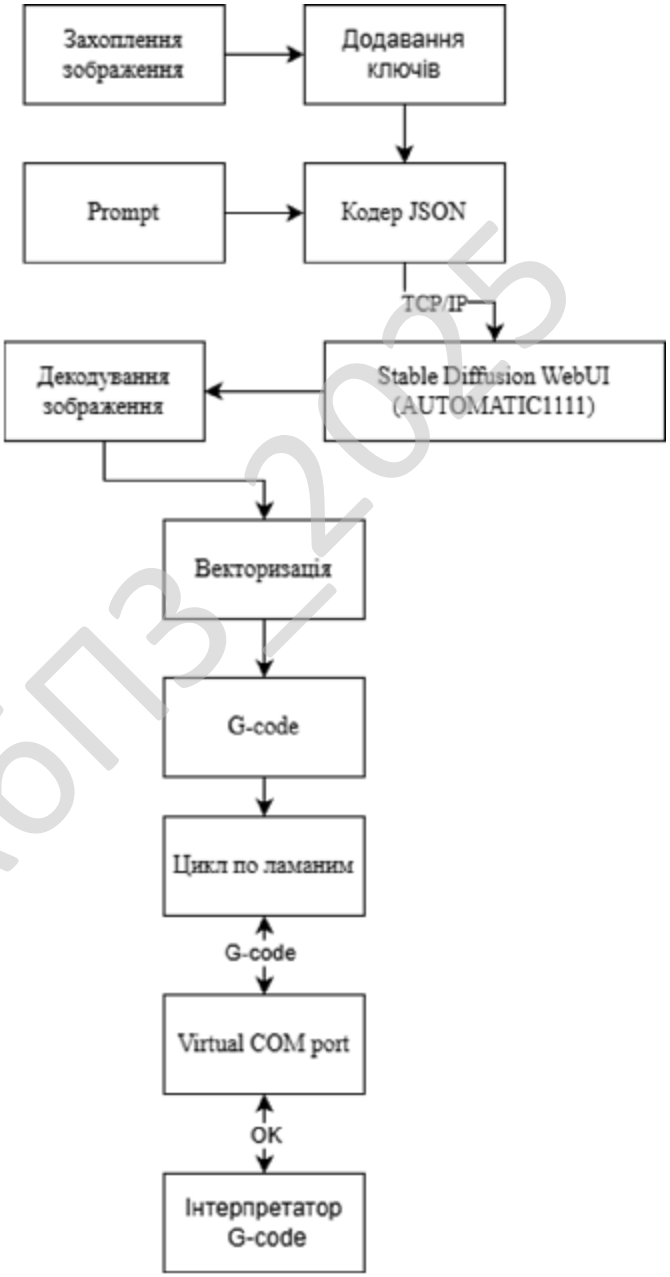

Рисунок 3.2 – Функціональна схема системи

Захоплення зображення. Натискання на кнопку ініціює знімок за допомогою IoT камери. Після цього зображення зберігається у відповідній директорії. В процесі використання пристрою ми забезпечуємо відображення з камери на екрані комп'ютера.

Prompt. Нам необхідно створити текстовий опис для стилізації зображення. Для задовільного результату вигамає чітко вказаного Positive та Negative prompt. Positive prompt вказує, з якими характеристиками буде вихідне зображення. Negative prompt, навпаки, описує чого не має бути у вихідному зображенні. Складається з фраз або слів англійською мовою. Різні моделі можуть ігнорувати деякі складні або суперечливі фрази.

Додавання ключів. Отримані дані з першого кроку, представляють бінарні дані, які не можуть безпечно передаватися по мережі. Нам потрібно перетворити ці бінарні дані у формат придатний для передачі. Stable Diffusion WebUI Automatic1111 API чекає на вхід зображення у форматі Base64.

Base64 – це метод кодування двійкових даних за допомогою 64 символів ASCII. Алфавіт складається з великих та малих латинських літер a-z, чисел від 0 до 9, а також знаків = та /. HTTP протокол орієнтований на текст. Через це деякі байти бінарного файлу можуть означати деякі команди, наприклад новий рядок. Кодування за допомогою метода нівелює можливість пошкодження файлу або переривання передачі. Збільшує об'єм даних приблизно на 33%.

Після кодування зображення ми маємо сформувати словник параметрів для API сервісу. В список ключів входять:

– закодоване зображення та вищезазначені запити;
– ключі, що відповідають за налаштування стилізації;
– параметри модулів розширення.

Кодер JSON. Коли словник сформований, ми перетворюємо дані словника у текстовий формат для передачі через протокол HTTP. Для досягнення цієї мети нам потрібно всі елементи словника перетворити у відповідні JSON типи даних.

JSON (JavaScript Object Notation) – текстовий формат обміну даних, зрозумілий як комп'ютеру так і людині. Складається з пар ключ-значення. Є стандартом для взаємодії с API сервісами. Незважаючи на свою назву, підтримується багатьма мовами програмування.

Трансформація відбувається наступним чином:

– рядки зберігають свій тип даних;

– числа перетворюються у відповідний формат JSON;

– якщо логічні значення задані як 1 або 0, вони перетворюються у true або false відповідно;

– якщо задається модуль розширення, він представляє окремий словник та буде перетворений у вкладений JSON-об'єкт;

В результаті ми отримуємо тіло майбутнього POST запиту.

Stable Diffusion WebUI (Automatic1111). Ми використовуємо це середовище для запуску моделей Stable Diffusion. Програму необхідно попередньо підготувати для роботи в API режимі, вказавши параметри запуску. Якщо POST запит сформовано правильно та налаштована функція API, відбувається зворотнє перетворення base64, та відновлюється початкове зображення. Після цього процесу починається стилізація. В іншому разі викликається виключення та повідомлення про помилку в консоль середовища.

Коли процес стилізації завершено, ми повинні отримати його у HTTP відповіді, як JSON-об'єкт. Сервер відповідає статусом 200, а саме зображення знаходиться у масиві images, закодоване у base64.

Декодування зображення. Ми транформуємо формат передачі даних у формат зображення. По-перше ми перетворюємо JSON у структуру даних мови програмування. Програма звертається до ключа, що містить зображення та дістає його. По-друге ми виконуємо декодування base64 й відновлюємо байти зображення. В результаті ми отримуємо відновлений файл стилізованого зображення та зберігаємо його у директорію.

Векторизація. На цьому етапі відбувається перетворення пікселів

растрового зображення у векторні криві. Ми маємо описати рух плотера. Для цього ми спочатку виконуємо фільтрацію шумів та інших артефактів. Наша мета, щоб система не вважала дефекти, як об’єкти для відтворення. Далі ми ділимо малюнок на об’єкти та фон. Після виділення об’єкту, щоб уникнути подвійних ліній, виконуємо пошук центрів ліній. В результаті замість широких кривих, ми отримаємо товщину в одну криву, тобто, що проходить по центру вхідного штриха. З кінцевих ліній формується векторний формат SVG, який описує ці криві.

G-code. Наше завдання перетворити векторні лінії у команди для плоттера. По-перше ми маємо підлаштувати координати малюнка, відповідно до розмірів робочого поля плотера, щоб зображення повністю вмістилося на аркуш. По-друге наша система має аналізувати лінії і розташовувати їх, щоб кінець однієї лінії був найближче до початку наступної. В кінці всі лінії перетворюються в команди для плотера та зберігаються у файл.

Цикл по ламаним. Ініціюємо цикл считування G-коду. Оскільки плотер реалізований на базі Arduino, ми не можемо передати одразу весь файл. Ми вибираємо одну команду та готуємо її до відправки. Відправка відбувається лише за умови, що плотер відповів “OK”. Після відправки команди цикл повторюється.

Virtual COM port. Через те, що плотер під’єднується за допомогою USB, нам треба реалізувати обмін даних між хостом та плотером. Однак для мікроконтролера це складний протокол, тому необхідно використаємо спрощену передачу даних. Цього можна досягти за допомогою Virtual COM порту.

Virtual COM порт – це програмне втілення фізичного послідовного порту, де дані передаються біт за бітом. Він емулює роботу застарілого порту RS-232 використовуючи USB. Наше завдання це використовувати цей порт, як стандартний файл I/O. Запис означає відправку даних, тоді як читання – отримання підтвердження від плотера.

Інтерпретатор “G-code”. Необхідно реалізувати розбиття вхідних рядків на лексеми. Виділяємо тип команди та координати для відображення. Далі вони передаються в модулі керування рухом та після перевірки формує послідовності

для двигунів. Коли лінія відображена, інтерпретатор має відправити “ОК” на комп’ютер, який відправить наступну команду та буде чекати наступного підтвердження.

3.4 Розробка діаграми процесів

Розробка діаграми процесів – це фінальний етап логічної архітектури ПЗ. Якщо попередні схеми визначали статичні компоненти, то діаграма процесів показує поведінку системи. Вони ініціюються зовнішніми або внутрішніми подіями. При проєктуванні я звертав увагу на сучасні графічні інтерфейси користувача. Там лінійна модель виконання алгоритму є неприйнятною. Будь-яка тривала операція, наприклад, очікування відповіді від моделі або передача даних на плотер, призводить до блокування інтерфейсу. На мою думку архітектура повинна розбивати алгоритми на атомарні стани, перехід між якими здійснюється асинхронно.

Діаграма побудована за радіальним принципом, а центральним вузлом виступає стан “Вікно програми”. Цей вузол виконує роль управлінця всією системою. Діаграма процесів зображена на рисунку 3.3. Центральний вузол постійно опитує ОС, оновлює елементи інтерфейсу та перевіряє статус підключених апаратних драйверів. Саме тут реалізовано механізм обробки переривань. Коли ми натискаємо кнопку, генерується сигнал, який перехоплюється і йде до одного з активних робочих станів. Відбувається автоматичне повернення до цього центрального стану після будь-якого циклу.

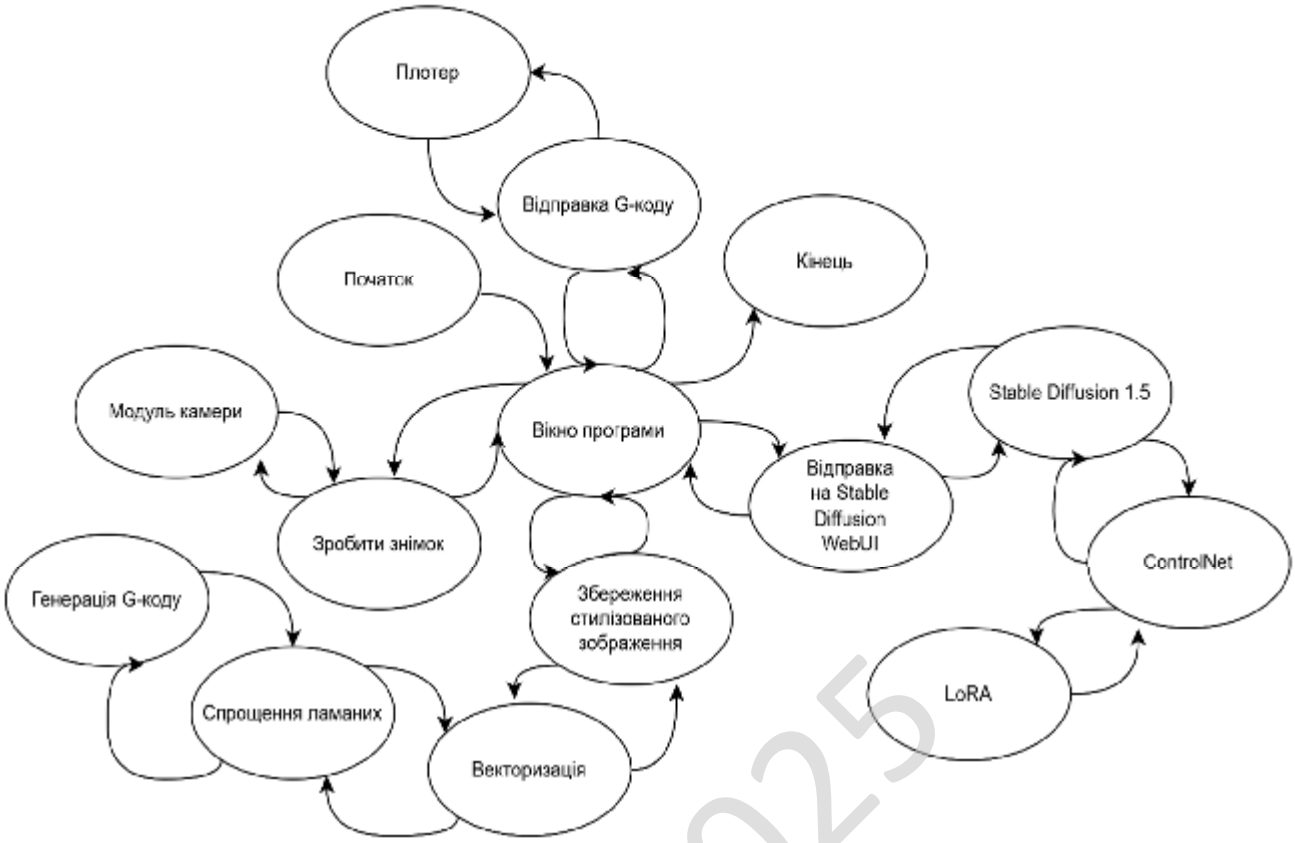

Рисунок 3.3 – Діаграма процесів

Перший процес діаграми – це отримання вхідних даних, який переходить системою зі стану очікування до стану “Модуль камери”. Цей перехід ініціалізує драйвер пристрою захоплення фото через інтерфейс, залежно від операційної системи. На низькому рівні виділяється кільцевий буфер в оперативній пам'яті для зберігання потокових даних.

Стан “Модуль камери” підтримує режим попереднього перегляду, транслюючи відеопотік на екран у реальному часі. В цьому циклі є команда “Зробити знімок”. Коли вона надходить то система копіює актуального кадр з відеопам'яті у локальну пам'ять процесу. Вимагає синхронізації доступу до даних для уникнення дефектів кадрів. Отримане зображення проходить обов'язкову конвертацію колірних просторів. Формат BGR, який є стандартним для бібліотек комп'ютерного зору типу OpenCV, перетворюється у формат RGB для правильного відображення в інтерфейсі та подальшої обробки. Коли растровий масив успішно

збережено, потік керування повертається до центрального вузла, сигналізуючи про готовність даних до наступного етапу.

Найбільш ресурсомістким етапом функціонування системи це ШІ генерація, котру ініціюємо через асинхронну мережеву транзакцію за протоколом HTTP. На етапі підготовки система виконує серіалізацію растрових даних у формат Base64 та формування JSON-структури з параметрами генерації, що дозволяє нам перекласти обчислення з клієнта на віддалений сервер без блокування графічного інтерфейсу. Після завершення генерації та десеріалізації відповіді розпочинається цикл пост-обробки. Головне завданням цього етапу – це зміна типу даних у модулі «Векторизація». Якщо ми видалимо надлишкові вершини, відхилення яких не перевищує задане порогове значення, ми отримаємо плавні векторні криві замість піксельні траєкторії. В кінці відбувається синхронний потік команд G-коду з реалізацією механізму Flow Control. Це означає, що кожна наступна інструкція руху надсилається лише після отримання від плотера повідомлення про готовність.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 зображена блок-схема алгоритму роботи розробленої програми. Схема відображає логіку та послідовність виконання функцій, а також переходи між ними. Побудована за модульним принципом, розділена обробка на блоки. Кожен з них відповідає за виконання конкретного етапу. Завдяки цьому ми можемо простежити весь цикл. Розглянемо та опишемо кожен з етапів.

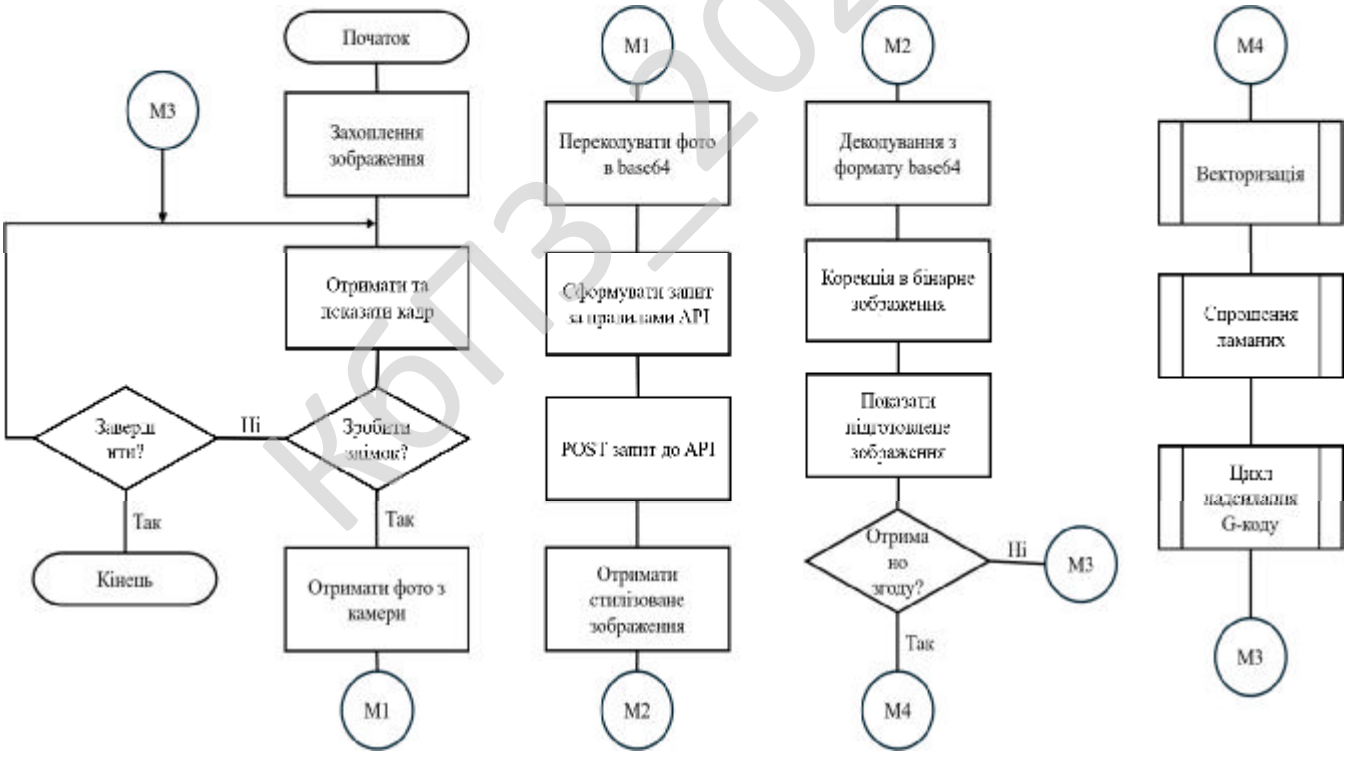

Рисунок 4.1 – Блок схема алгоритму роботи програми

Ми ініціалізуємо процес захоплення зображення з камери підключеної через USB. Починається неперервне захоплення кадрів. Зображення попередньо

транслюється на екран як відео. На його основі ми підбираємо вхідне зображення. Відео насправді це багато одинарних знімків частотою в 30 кадрів на секунду. Коли ми натискаємо на кнопку фотографування, програма обирає кадр, який був в цей момент та відображає його на екрані для того, щоб ми могли його оглянути та оцінити його якість. В цей момент відеопотік переривається й програма чекає на наші дії. Якщо якість зображення незадовільна, то відеопотік відновлюється для захоплення наступного. Перед цим програма запитає у нас чи бажаємо ми завершити роботу. У цьому випадку алгоритм підходить до кінцевого стану. Якщо нас задовільняє якість, то відбувається фіксація кадру та його збереження у растровому вигляді. Далі це зображення передається як вхідні дані в подальші етапи алгоритму.

Коли ми отримали зображення ми кодуємо його в формат base64, через те що на даному етапі зображення в бінарному представленні. Після цього ми формуємо запит за правилами API Stable Diffusion WebUI (AUTOMATIC1111). Згідно правил ми формуємо пари ключ-значення у форматі JSON. До нього входять:

– заздалегідь створені текстові запити, що описують обраний нами стиль вихідного зображення та його характеристики;

– кодоване зображення;

– розміри вихідного зображення в пікселях;

– об’єкт для налаштування ControlNet з необхідними для нього аргументами;

– параметри, що визначають як саме буде оброблятися зображення.

Після створення словника с ключами-значеннями та перетворення його у JSON дані готові для надсилання до API. Ми створюємо POST запит до відповідної кінцевої точки. Запит відбувається за стандартним механізмом клієнт-серверної архітектури.

Stable Diffusion WebUI отримує запит, виймає та декодує дані. Далі використовуючи вказану нами модель Stable Diffusion стилізує зображення

відповідно до наших параметрів. В результаті формується нове растрове зображення. Після цього він повертає відповідь з отриманими зображеннями також закодованими в base64 та форматі JSON.

Коли ми отримали відповідь, ми виймаємо массив зображень з JSON формату. Далі декодуємо зображення та отримуємо нове растрове зображення готове для подальших операцій.

Отримане на попередньому етапі зображення проходить бінаризацію. Це спрощує нам задачу аналізу структури зображення. Цей процес кожному пікселю зображення просвоює в залежності від яскравості одне з двох можливих значень. Працює наступним чином. Спочатку задається пороговий рівень. Якщо яскравість перевищує цей поріг, то піксель відноситься до фону та отримує значення білого кольору, навпаки, піксель отримує значення чорного кольору та відноситься до об’єкта зображення.

Після бінаризації, програма відображає нам його та чекає нашого вироку. Якщо результат обробки незадовільний, то алгоритм повертає нас до етапу захоплення зображення. Неякісне зображення відтворено не буде.

Коли отримано згоду, ми розпочинаємо підготовку зображення до відображення плотером. Алгоритм перетворює його із растрового в G-код. Але для початку ми маємо перетворити його у векторний вигляд. Тобто виділити контури з бінарного зображення та сформувати відповідні вектори. Було розроблено блок-схему алгоритму векторизації. На малюнку 4.2 зображено стандартний алгоритм векторизації. Розглянемо його більш детально.

Стандартний алгоритм векторизації

На вхід векторизації подається бінарне растрове зображення з попереднього етапу. Воно представляє собою двовимірну матрицю пікселів *img* та має визначені розміри ширини та висоти. Кожен елемент цієї матриці приймає лише два можливі значення – це або фон зображення або об’єкт який ми маємо векторизувати.

Далі нам потрібно знайти всі контури. Наш алгоритм переходить до етапу

повного обходу зображення. Обхід здійснюється послідовним способом. Зовнішній цикл змінює значення координати x, що відповідає за ширину зображення, а внутрішній – координату y, що відповідає за висоту зображення. Тепер для кожної знайденої пари (x, y), ми виконуємо перевірку значення їх пікселів. Якщо піксель білий, то він позначається як фон й алгоритм переходить до наступної пари, якщо піксель чорний, то він представляє елемент контура. Коли знайдений чорний піксель, починається побудова ламаної, тому що це може бути початок вектора.

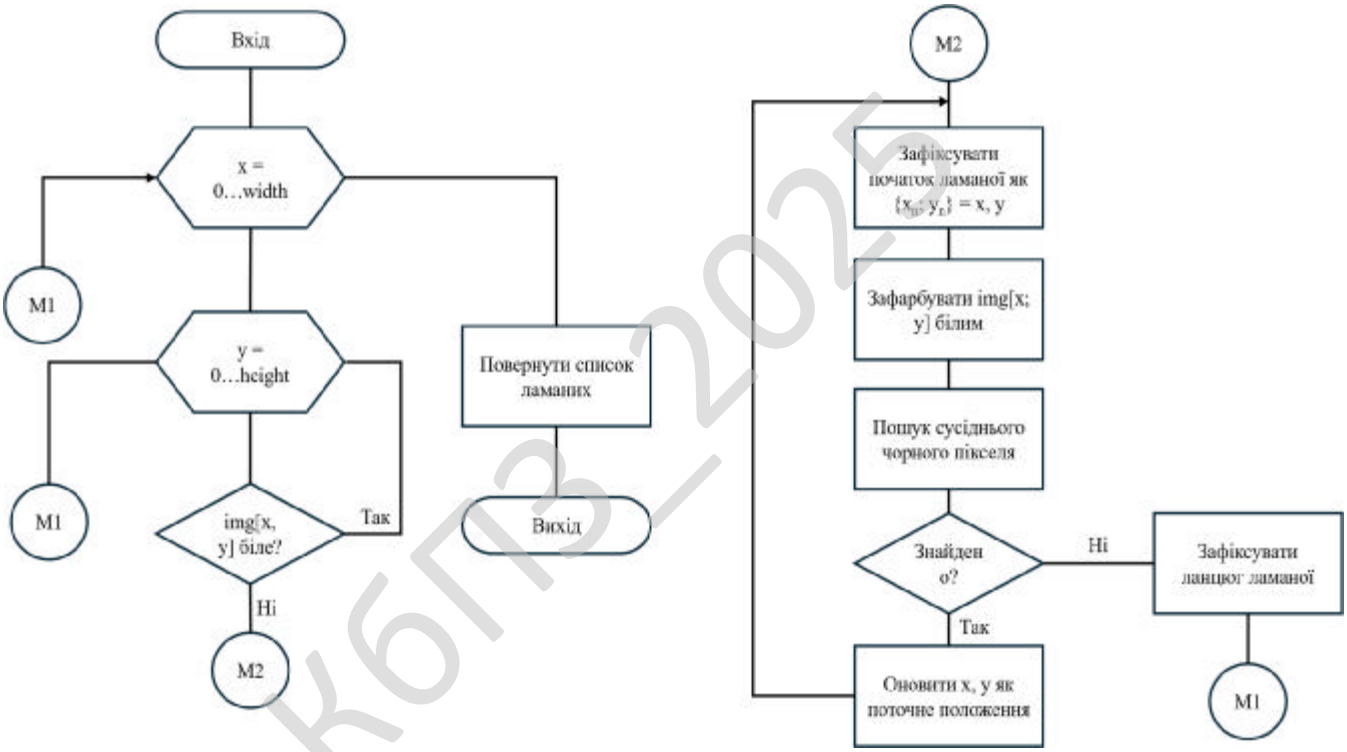

Рисунок 4.2 – Стандартний алгоритм векторизації

Коли ми виявили чорний піксель, алгоритм переходить до етапу створення ламаної. На цьому кроці створюється новий вектор. Координата (x, y) позначається початковою вершиною цього вектора. Ми створюємо структуру даних для збереження послідовності точок ламаної, до якої додається зафіксована координата як перший елемент.

Після цього оброблений піксель змінює колір з чорного на білий в матриці

img. Це зроблено для того, щоб виключити повторну обробку вже оброблених пікселів на подальших ітераціях обходу зображення. Одразу після зміни кольору виконуємо перевірку сусідніх пікселів від зафарбованої координати, щоб знайти чорного пікселя, що буде продовженням цього контуру. Здійснюємо пошук зі всіх сторін координати. Якщо ми знаходимо такий піксель його координати становлюються як нові положення, після чого ми повторюємо процес починаючи з додавання точки до ламаної.

Процес створення ламаної завершується для ділянки, коли біля поточної координати не виявлено чорних пікселів. Тепер ми фіксуємо завершену ламану на основі послідовності точок та додаємо до списку, який використовуємо для зберігання всіх виявлених ламаних. Коли ламану збережно, алгоритм повертається до обходу зображення та продовжує пошук наступних чорних пікселів. Після завершення обходу всього зображення виконання алгоритму припиняється, а список ламаних передається далі.

Недоліки стандартного алгоритму векторизації

Під час практичного використання базового алгоритму було встановлено, що відсутність контролю напряму та жорсткий порядок обходу зображення призводять до формування ламаних із великою кількістю дрібних елементів та різких змін напрямку. Загальні недоліки:

– Якщо на кожному кроці шукати будь-якого чорного пікселя, то алгоритм робить зайві повороти, утворювати злами там, де контур мав бути плавним;

– Коли контур товстий або є перетини, випадковий вибір сусіда веде до передчасного обриву однієї ламаної і народження багатьох коротких, або до невизначених переходів між гілками. Такі особливості негативно впливають на генерацію G-коду. Багато ламаних породжує багато команд. Вони ускладнюють керування процесом малювання хаотичним рухом робочого інструмента;

– Наш алгоритм знаходить наступний старт ламаної далеко від попередньої. У результаті після перетворення в G-код зростає час малюванная та кількість холостих переміщень.

Покращений алгоритм векторизації

Покращений алгоритм зберігає основні етапи стандартного алгоритму. Ми так само перевіряємо значення пікселя, позначаємо вже оброблені елементи, щоб унеможливити повторну обробку одних і тих самих ділянок та проводимо обхід бінарного зображення з векторизацією. Але він змінює порядок обходу зображення та вводить врахування напрямку руху під час прокладання контуру. Ці зміни прибирають обриви ламаних та як наслідок покращить нам результат малюнок плотером. Розглянемо детально кожен крок нового алгоритму.

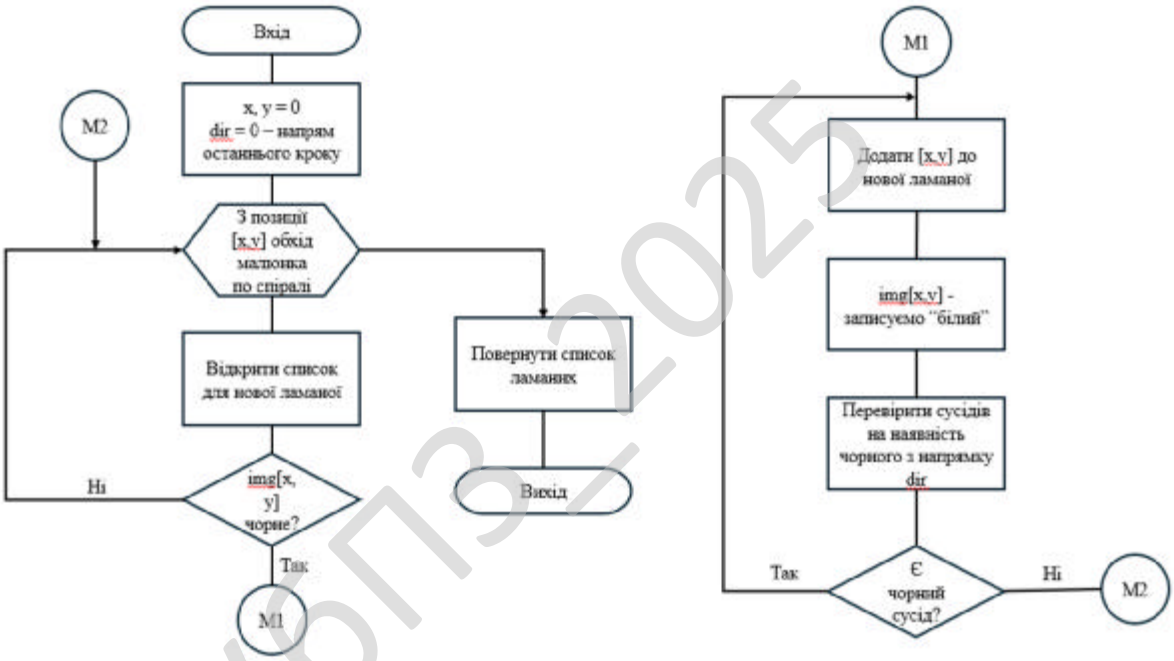

Рисунок 4.3 – Покращений алгоритм векторизації

Ініціалізація алгоритму

На початку виконання алгоритму встановлюється початкове положення обходу зображення через ініціалізацію координат $x = 0$ та $y = 0$. Додатково вводиться змінна *dir*, що встановлюється нулем, та означає напрям останнього виконаного кроку під час відстеження ламаної. Змінна по суті буде зберігати напрямок переходу між пікселями, тому у нас не буде формуватися ламана з хаотичними змінами напряму на кожному кроці.

Обхід зображення по спіралі

Замість класичного повного перебору координат зображення за рядками та стовпцями у покращеному алгоритмі ми використаємо обхід по спіралі з позиції (x, y). Це покращення необхідне нам, щоб усунути проблему старту ламаної далеко від попередньої. Ми поступово розширюємо область пошуку навколо поточного положення, тому нові чорні пікселі будуть знаходитися біля вже оброблених. Як результат отримуємо меншу кількість холостих переходів між частинами зображення при відображенні плотером.

Відкриття нової ламаної та додавання точки до неї

Коли в позиції (x, y) з'явиться чорний піксель, алгоритм створить нову ламану. Для неї є список для збереження її точок, які будуть додаватися у процесі прокладання контуру. На початку роботи логіка співпадає зі стандартним, але в наступних кроках ламана формується за іншими правилами. Координата поточного пікселя (x, y) додається до списку точок ламаної. Далі кожна нова знайдена точка контура буде додаватися в цей список.

Маркування пікселя як обробленого

Після додавання координати до ламаної відповідний піксель у матриці зображення позначається як оброблений шляхом зміни його значення на біле. Цей крок відповідає стандартному алгоритму й відповідно забезпечує завершення процесу обходу зображення.

Перевірка сусідів з урахуванням напрямку руху

Це покращення для послідовності руху та зменшенні кількості змін напряму. Наш покращений алгоритм шукає наступний чорний піксель з використанням змінної *dir*. Коли перевіряється сусідній піксель використовується впорядкований список напрямків. Спочатку алгоритм перевіряє значення напряму в *dir*. Якщо він не виявив чорного пікселя у цьому напрямку, то продовжує перевірку для інших напрямів.

Оновлення стану та завершення ламаної

Якщо алгоритм під час перевірки знайшов чорний сусідній піксель,

координати *x* та *y* оновлюються відповідно до його положення, а змінна *dir* до напрямку переходу. Після цього процес додавання точки повторюється. Якщо немає чорного сусіднього пікселя, ламана вважається завершеною. Вона додається до загального списку. Алгоритм повертається до спірального обходу зображення та шукає наступну стартову точки.

Повернення результату

Після обходу всієї області зображення алгоритм припиняє виконання й повертає список знайдених ламаних. Отриманий список ламаних передається у наступний етап – спрощення ламаних.

Спрощення ламаних

Поки що ми маємо лише велику кількость послідовних точок. Ламана повторює форму контуру, але має непотрібні вершини. Нам вони не потрібні, адже тільки впливають на геометрію лінії. Ігнорування збільшує обсяг G-коду, від чого залежить кількість коротких сегментів та змін руху. Тому ціль цього етапу – зменшити кількість вершин, але зберегти форму в межах допустимої похибки. Блок-схема алгоритма зображена на малюнку

Для реалізації нам потрібно визначити відстань від довільної точки, що представляє ламану, до прямої на площині. Нехай у декартовій площині задано 3 точки. Це P0(x0,y0), P1(x1,y1), P2(x2,y2). Точки P0 і P2 представляють пряму, а точка P1 відносно розташована відносно неї.

Позначимо d, як відстань від точки P1 до прямої, що проходить через точки P0 та P1. d – це перпендикуляр, опущений з точки P1 на пряму. Ми можемо звести задачу до знаходження спочатку висоти трикутника, які утворені цими точками.

Спочатку ми виводимо довжину сторін утвореного трикутника. Довжини сторін обчислюємо в декартовій системі координат за теоремою Піфагора перетворену для аналітичної геометрії. Позначимо відрізок між точками P0 та P2, як a, між P0 та P1, як b, й P1 та P2 через c. Відстань між цими точками розраховується за формулою:

$$(P_iP_j) = \sqrt{\left((x_j - x_i)^2 + (y_j - y_i)^2\right)}$$

Знаходимо за формулою значення для a, b та c:

$$a = \sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2}$$

$$b = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$

$$c = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Далі обчислюємо площу трикутника з довжиною його висоти та основи. Якщо довжина основи позначена a, а висота це опущений перпендикуляр на пряму, як d, то формула набуває наступного вигляду:

$$S = \frac{ad}{2}$$

Ми використаємо цей вираз площі трикутника разом з його альтернативним способом обчислення, та виведемо шукану відстань d, яка одночпсно є висотою трикутника до сторони a.

Тепер нам потрібно обчислити півпериметр трикутника для обчислення площі на основі його довжин. Півпериметр – це половина суми довжин усіх сторін трикутника. Позначимо його як l:

$$l = \frac{(a + b + c)}{2}$$

Коли ми отримали підпериметр ми можемо обчислити площу трикутника за довжинами його сторін. Для цього використаємо формулу Герона, яка визначає площу без тригонометричних функцій або знаходження кутів. Для відомих сторін нашого трикутника та півпериметра, формула виглядає наступним чином:

$$S = \sqrt{\big(l(l-a)(l-b)(l-c)\big)},$$

$$S = \frac{ad}{2}.$$

Визначивши двома різними способами площу трикутника ми можемо прирівняти їх між собою:

$$\frac{ad}{2} = \sqrt{l(l-a)(l-b)(l-c)}$$

Ми встановили зв’язок між шуканою відстанню d та довжинами сторін

трикутника. Далі ми виводимо формулу, що обчислює відстан від точки до прямої без додаткових геометричних обчислень.

Виконуємо алгебраїчне перетворення отриманого рівняння відносно d. Для цього помножимо обидві частини на 2 й поділимо на a та отримуємо:

$$d = \frac{2}{a}\sqrt{l(l-a)(l-b)(l-c)}$$

Отриманий вираз – це замкнена формула обчислення відстані від довільної точки до прямої, яким ми вирахували d, та за можемо використовувати для спрощення ламаних та оптимізувати роботу плотера. Розглянемо блок-схему алгоритму спрощення зображену на малюнку 4.4.

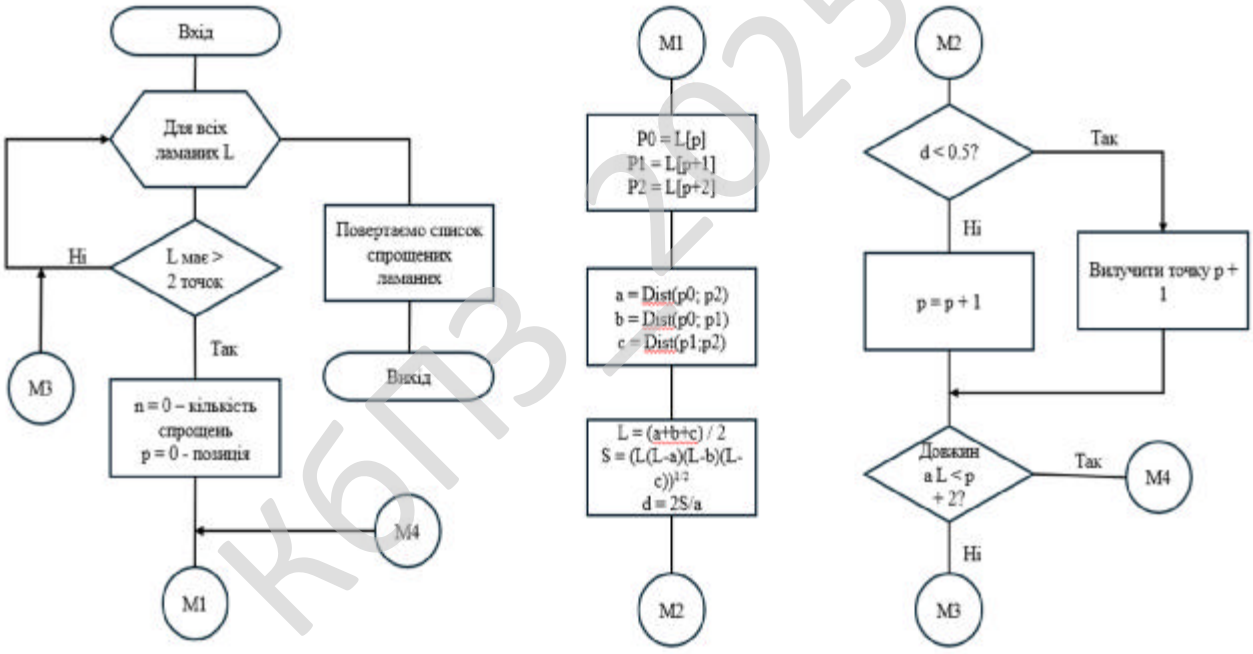

Рисунок 4.4 – Алгоритм спрощення ламаних

Алгоритм спрощення ламаних запускається після завершення етапу векторизації зображення. На вхід поступає список ламаних, кожна з яких є послідовністю точок у декартовій системі координат. Кожна ламана відповідає

окремому фрагменту контуру або штриха зображення й алгоритм одну за одною обробляє кожну ламану зі списку. Для кожної ламаної ми перевіряємо кількість її вершин. Якщо ламана містить менше трьох точок, ми її не спрощуємо, тому що для визначення відхилення точки від прямої має бути хоча б три точки. Якщо менше трьох точок – ламана передається до вихідного списку без змін й алгоритм переходить до обробки наступної.

Для ламаних, що складаються з трьох або більше точок, ініціалізуються змінні:

– p = 0. Це індекс поточної позиції ламаної, який визначає початкову точку трійки;

– n = 0. Це лічильник виконаних операцій спрощення.

Після ініціалізації алгоритм переходить до аналізу геометрії ламаної. На кожній ітерації алгоритм вибирає три послідовні точки ламаної:

$$P_0 = L[p]$$
$$P_1 = L[p+1]$$
$$P_2 = L[p+2]$$

Точки P_0 та P_2 розглядаються як крайні вершини ламаної, тоді як точка P_1 як проміжна, яка може бути вилучена без істотної зміни форми ламаної. Для обраної трійки ми обчислюємо довжини відрізків між кожною парою. Для цього ми використаємо функцію Dist, що рахує евклідову відстань між двома точками в декартовій площині. Функція зображена на малюнку 4.5.

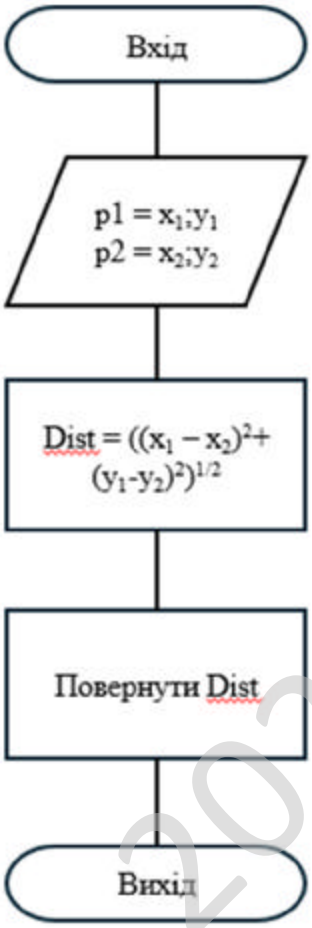

Рисунок 4.5 – Блок-схема функції Dist

Нехай задано дві точки площини $P_1(x_1, y_1)$ та $P_2(x_2, y_2)$. Відстань між цими точками визначається як довжина відрізка, що їх з'єднує. Функція Dist має вигляд:

$$\text{Dist}(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

Ми використовуємо функцію, щоб вирахувати довжини сторін утвореного трикутника P_0, P_1та P_2. Це знадобиться нам для розрахування значень a, b, c для обчислення відстані від точки до прямої.

Тепер коли ми маємо d, можемо дізнатися потрібна нам точка чи ні наступним чином:

Якщо d < 0.5. Це означає, що точка P1 майже на прямій між P0 і P2. Точку можна прибрати, адже тільки дробить траєкторію. При такій умові алгоритм виконує дії: видаляє L[p + 1]. Після видалення точки ламана зменшується і на позиції p розташовується нова трійка. Якщо d ≥ 0.5. Точка важлива. Виконуємо p = p + 1, інакше кажучи переходимо до наступної трійки точок.

Цикл надсилання G-коду

Це останній етап нашого алгоритму. На вхід надходить набір спрощених ламаних у вигляді впорядкованих списків точок. Кожна ламана – це траєкторія пера, яку залишилось фізично відтворити. Перед надсиланням ці траєкторії вже були приведені до робочої системи координат плотера, а також оптимізовані спрощенням, щоб зменшити кількість точок і команд.

Щоб почати роботи, ми маємо налаштувати параметри серійного з'єднання:

– Порт підключення (у нашому випадку /dev/ttyACM0);

– Швидкість обміну (115200);

– Час очікування відповіді від плотера.

Тепер для кожної ламаної алгоритм формує список G-code команд:

– G21. Встановлення одиниць у міліметрах;

– G90. Абсолютне позиціонування;

– G1 F1500. Швидкість переміщень;

– G0 Z1.5. Підняти перо;

– G0 X... Y... Швидкий перехід у стартову точку ламаної;

– G0 Z0 – опустити перо і почати малювати.

На основі цих інструкцій для кожної ламаної створюємо команди. Далі ці команди по черзі передаються пристрою. Передача відбувається у синхронному режимі. Після кожного рядка чекаємо на відповідь "OK" від контролера. Коли всі команди відправлені, серійне з'єднання закривається, а алгоритм переходить до фінального стану. Це або новий запуск або завершення роботи програми.

4.2 Захист розробленого програмного забезпечення

Оббфускація вихідного коду

Python – це інтерпретована мова програмування, отже зберігає вихідні коди програми у форматі байт-коду. Мені потрібно було додатково захистити алгоритми від методів зворотного проектування. Щоб зменшити ризики несанкціонованого доступу та копіювання файлів програми, особливо унікального алгоритму спірального пошуку та трансформації контурів, я модифікував вихідний код на рівні абстрактного синтаксичного дерева, тобто здійснив трансформацію логіки програми, але без зміни її кінцевого функціоналу. Це включає:

– Перейменував усі назви змінних, внутрішніх функцій та класів у модулях векторизації та обробки сигналів. Замінено на нечитабельні буквено-цифрові послідовності. Стороннім особам дуже важко провести семантичний аналіз без спеціалізованих програм;

– Вніс непотрібний код для протидії автоматизованим засобам аналізу до файлів обробки зображень та додав фрагменти коду, які нічого не роблять, але ускладнює знаходження реальних алгоритмів обробки серед масиву непотрібних операцій;

– Зіпсував візуальне представлення програми в декомпільованому виді. Перетворив лінійну структуру логічних переходів та циклів у модулях керування камерою та інтерфейсом користувача на складну архітектуру з центральним диспетчером станів;

– Усі текстові параметри, шляхи до API та налаштування нейромережевих моделей у файлах зашифрував. Дешифруються динамічно в оперативній пам’яті лише під час виконання відповідної операції.

Ізоляції середовища виконання та обмеження доступу

Потенційних загрози цілісності операційної системи, можуть виникнути через вразливості сторонніх бібліотек (OpenCV, PyQt6) вихідного коду. Тому я ізолював виконання та суворого розставив права доступу. Ізоляція включає:

– Взаємодію з камерою та плотером здійснив тільки через прокидання конкретних файлів пристроїв (/dev/ttyACM0 –для передачі G-коду) у середовище контейнера. У разі вразливості в біблиотеці шкідливого коду не зможе отримати доступ до інших пристроїв хост-систем;

– Дав право додатку записувати лише у директорію output_images, яка монтується як окремий розділ. Решта файлової системи залишається тільки для читання;

– Зв’язок з сервером Stable Diffusion працює через внутрішню мережу Docker. Додаток може надсилати HTTP-запити лише на визначену адресу SD_URL. Спроба несанкціонованої передачі даних на зовнішній ресурс блокується миттєво.

Для керування правами доступу здійснив:

– Обробка зображень та векторизація виконуються від імені системного користувача з обмеженими правами, який не має доступу до команд sudo;

– Доступ до послідовного порту для відправки G-коду реалізовав через групу plotter, для взаємодії з пристроєм /dev/ttyACM0 без надання root-прав. Таким же чином реалізував роботу з відеопотоком через групу video.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Головне вікно програми поділене на дві частини. Зображене на рисунку 5.1. У лівій частині розміщена область для відео з камери. Знизу знаходиться кнопка захопити кадр, яка фіксує поточне зображення без зупинки відео.

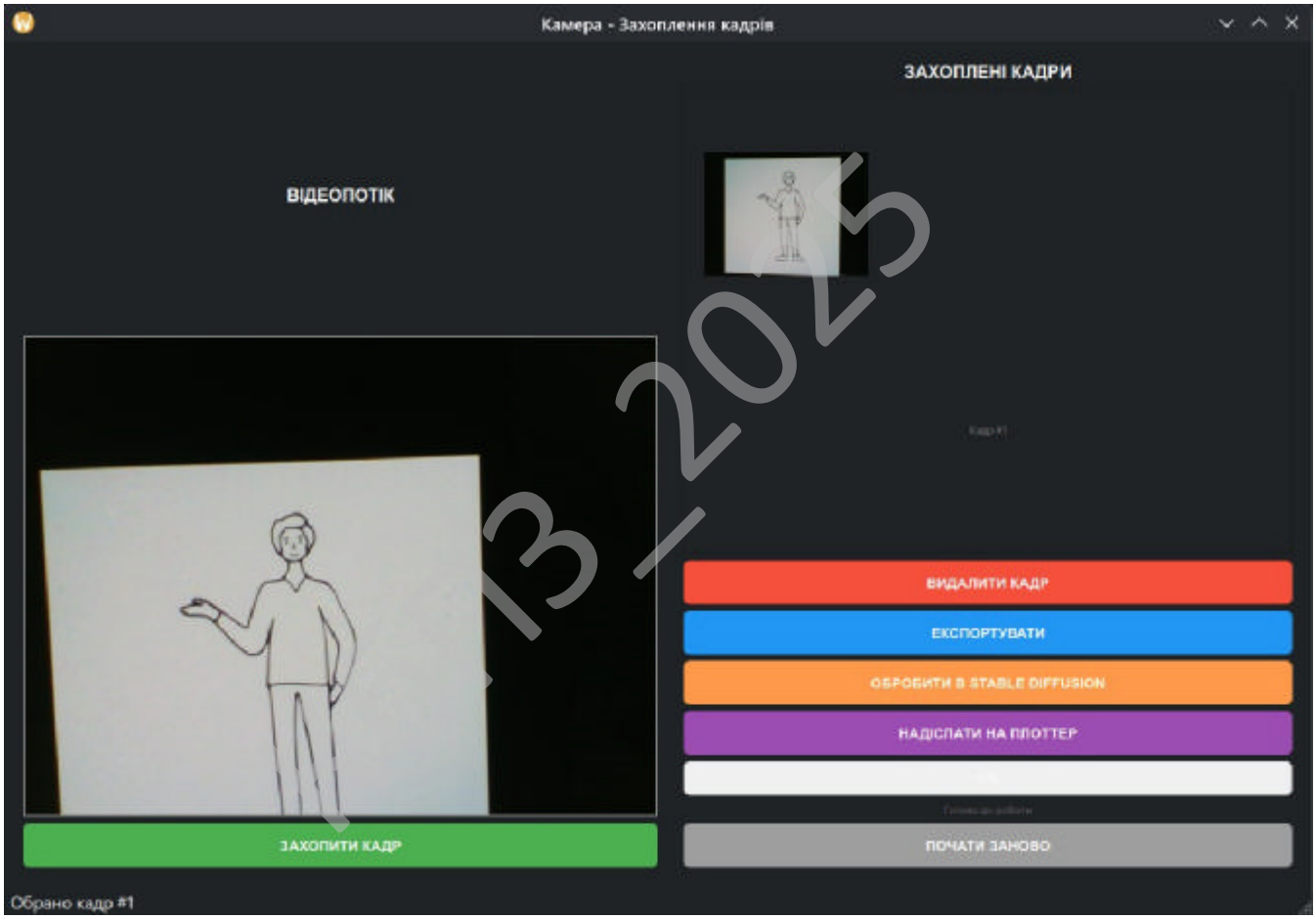

Рисунок 5.1 – Головне вікно програми

Права частина призначена для роботи із захопленими кадрами. Тут відображається галерея зменшених копій зображень. Ми можемо швидко обрати потрібний нам кадр та виконати над ним наступні операції:

– Видалити кадр.

– Експорт у файлову систему.

– передати на обробку в Stable Diffusion WebUI.

– надіслати на плотер.

– поки що неактивна кнопка почати заново(стане доступна при обробці).

Внизу розміщені елементи стану системи. Це смуга прогресу та текстове повідомлення, які показують нам поточний етап виконання стилізації, векторизації, передавання G-коду.

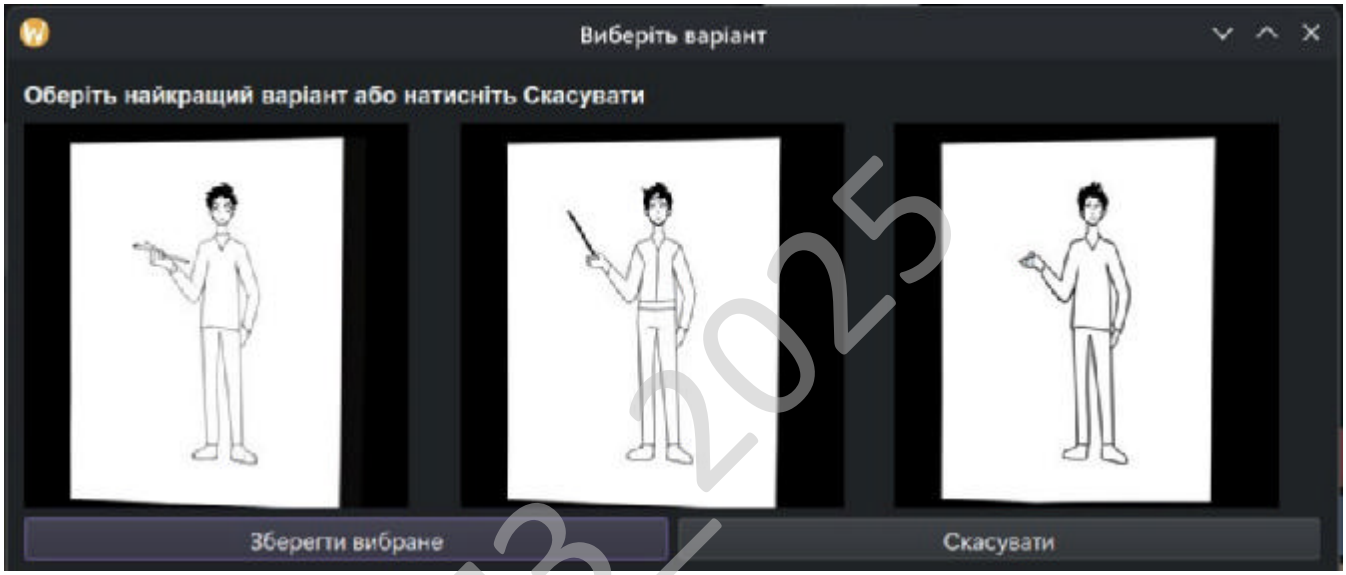

Рисунок 5.2 – Вікно вибору результату після стилізації

Ми підходимо до завершальної фази життєвого циклу розробки, а саме впровадження розробленого програмно-апаратного комплексу. Нам потрібно адаптувати програму під конкретні експлуатаційні умови та підготувати її до повноцінного функціонування. Це дуже важливий процес трансформації програмного продукту з тестової версії у стан використання кінцевим споживачем. Для моєї програми цикл розгортання наступний:

– Підготовати реліз та включити логіку обробки зображень та конфігураційні файли для API Stable Diffusion WebUI;

– Розгорнути середовище Python, встановити залежності та налаштувати

локальний сервер моделей;

– Скоректувати параметри векторизації та масштабу відповідно до габаритів плотера;

– Забезпечити стабільну взаємодію між камерою та контролером керування рухом пера;

– Контролювати оновлення моделей ControlNet та версій API, від яких залежить якість генерації лінійного малюнка.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для отримання фотозображення, його стилізації та відтворення за допомогою реалізації пристрою IoT плоттеру.

Метою розробки є створення програми отримання фотозображення, його стилізації та відтворення за допомогою реалізації пристрою IoT плоттеру.

Об'єкт дослідження є процес стилізації растрового зображення, перетворення його у векторну графіку й відтворення пристроєм.

Предмет дослідження є методи, програмні засоби й алгоритми отримання, стилізації та векторизації зображення.

Методи дослідження є методи системного аналізу, методи цифрової обробки зображень та алгоритми машинного навчання.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– удосконалено алгоритм векторизації растрових зображень;

– удосконалено алгоритм спрощення ламаних.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Визначення цільової аудиторії розробленого апаратно-програмного комплексу базується на аналізі потреб ринку в автономних та бюджетних засобах кіберзахисту. Оскільки система використовує технології граничних обчислень на базі мікроконтролера, вона орієнтована на користувачів, для яких критичними є швидка реакція на загрози та незалежність від хмарних сервісів.

Основним споживачем продукту є промисловий сектор, де підприємства потребують захисту мереж від несанкціонованого втручання без передачі даних за межі локальної мернежі. Локальний аналіз трафіку нейромережею дозволяє оперативно ізолювати загрози, що важливо для операторів критичної інфраструктури, які мають підвищені вимоги до безпеки та конфіденційності.

В сегменті споживчої електроніки та систем «Розумного будинку» розроблений комплекс виступає як захисний шлюз. Він забезпечує безпеку для великої кількості пристроїв різних виробників, які часто не мають вбудованих засобів захисту та рідко отримують оновлення безпеки. Компанії які інтегрують такі системи можуть використовувати даний продукт для підвищення загального рівня захищеності мереж своїх клієнтів.

Також система приваблива для компаній, що надають послуги з моніторингу кібербезпеки. Завдяки підтримці протоколу MQTTS та наявності адміністративної панелі на Python, комплекс легко стає частиною великих центрів моніторингу як доступний периферійний датчик виявлення аномалій. Комерційний успіх проєкту забезпечується поєднанням низької вартості апаратної частини та високої ефективності інтелектуального аналізу даних.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Щоб перевірити, наскільки розроблений комплекс буде затребуваним на реальному ринку, я провів аналіз його характеристик за допомогою методу експертних оцінок. Це дозволило поглянути на систему не просто як на наукову розробку, а як на потенційний продукт, оцінюючи його очима фахівців з кібербезпеки та автоматизації. В основу оцінювання лягли такі критерії, як технічна новизна, надійність захисту, простота впровадження та, що дуже важливо для комерційного сектору, економічна вигідність.

Найвищі бали експерти поставили за використання концепції Edge AI, оскільки запуск нейронної мережі безпосередньо на мікроконтролері ESP32-S3 дозволяє системі працювати автономно. Фахівці відзначили, що адаптація алгоритмів під векторні інструкції архітектури Xtensa LX7 робить обробку трафіку дуже швидкою, що критично для виявлення атак у реальному часі. Крім того, професійну зацікавленість викликав той факт, що в системі з самого початку закладено апаратні функції захисту, такі як Secure Boot V2 та шифрування пам’яті за стандартом AES-256 XTS, що робить розробку стійкою до фізичного втручання та підміни коду.

Окремо обговорювалася практична цінність механізму дистанційної ізоляції вузлів через MQTT-команди. Експерти зійшлися на думці, що можливість миттєво заблокувати зловмисника через зручну адмін-панель на Python є вагомою перевагою для системного адміністратора. При цьому використання протоколу MQTTS із захистом TLS 1.3 знімає питання щодо безпеки самих керуючих команд.

З точки зору економіки, мій проєкт виглядає виграшно, бо використання недорогої платформи ESP32-S3 дозволяє створити потужний захисний шлюз за ціною, значно нижчою, ніж у класичних промислових IDS. Загальний підсумок експертного аналізу підтвердив, що продукт вдало заповнює вільну нішу на ринку безпеки IoT, поєднуючи в собі інтелектуальні можливості великих систем із

компактністю та низькою ціною вбудованих рішень.

7.3 Вибір методу оцінки вартості ПЗ

Щоб вибрати метож оцінки вартості ПЗ необхідно врахувати не лише витрати на написання коду, а й на апаратну реалізацію периферійних обчислень. Для обґрунтування інвестиційної доцільності впровадження системи обрано метод повної вартості володіння. Даний підхід дозволяє комплексно проаналізувати всі прямі та непрямі витрати, що виникають протягом повного життєвого циклу експлуатації ситеми – від розробки архітектури TinyML до етапу технічної підтримки в промислових умовах.

Використання цього методу для цього проєкту зумовлене можливістю деталізації наступних видатків:

— Капітальні витрати: придбання мікроконтролерів ESP32-S3, розробка спеціалізованих друкованих плат, а також часові витрати на навчання нейромережевих моделей у середовищі TensorFlow Lite.

— Операційні витрати: мінімальне енергоспоживання архітектури Xtensa LX7 (що критично для автономних вузлів), витрати на адміністрування MQTT-брокера та забезпечення безпечного оновлення прошивки за технологією OTA.

Застосування даної методики дозволяє чітко знайти потенційні зони оптимізації бюджету. Зокрема, перенесення процесу обробки трафіку безпосередньо сам мікроконтролер радикально знижує витрати на мережу та оренду хмарних серверних потужностей, які зазвичай необхідні для аналізу великих масивів даних.

Інтеграція методу ТСО з аналізом вигід забезпечує зіставлення вкладених ресурсів із отриманими результатами: підвищенням рівня стійкості локальної мережі до DDoS-атак та зменшенням часу реакції на інциденти. Такий підхід робить оцінку вартості прозорою для управлінських структур, підкреслюючи перевагу використання бюджетної, але потужної платформи ESP32-S3 над

дорогими промисловими аналогами. У результаті, обраний метод дозволяє довести, що економічна ефективність системи досягається не лише за рахунок низької ціни компонентів, а й завдяки автономності та енергоефективності програмних рішень.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Перенесення інференсу нейронної мережі безпосередньо на мікроконтролер дозволяє замінити дорогі серверні рішення автономними вузлами з низьким енергоспоживанням. Обґрунтування привабливості проєкту проведено на прикладі локальної мережі промислового об’єкта з 50 точками моніторингу трафіку.

Для розрахунку використано наступні показники:

— Зниження витрат на хмарні сервіси: використання локального аналізу замість API-запитів до Cloud AI дозволяє заощадити близько 4 500 грн на місяць для одного вузла.

— Економія мережевого ресурсу: фільтрація 98% надлишкового трафіку на рівні Edge мінімізує витрати на пропускну здатність каналів зв’язку.

— Енергоефективність: сумарна потужність 50 модулів ESP32-S3 не перевищує 25 Вт, що в десятки разів менше за споживання типового сервера аналізу даних (від 250 Вт), забезпечуючи річну економію електроенергії обсягом понад 15 000 грн.

Фінансові результати впровадження системи демонструють високу рентабельність, а саме після вирахування витрат на адміністрування MQTT-інфраструктури становить 420 000 грн для досліджуваної мережі завдяки низькій собівартості апаратної частини інвестиції повертаються протягом 3,5 місяців експлуатації; коефіцієнт ROI складає 285%, що підтверджує високу фінансову стійкість проєкту.

Порівняльні характеристики систем наведено у таблиці 7.1.

Таблиця 7.1 – Порівняння витрат та характеристик систем захисту

Показник	Хмарна IDS (традиційна)	Edge AI шлюз (ESP32-S3)
Вартість апаратної точки	12 000 грн	950 грн
Щомісячна абонентська плата	3 500 грн	0 грн
Затримка аналізу	150–400 мс	15–30 мс
Енергоспоживання	1800 кВт·год	12 кВт·год
Ризик компрометації даних	Середній	Мінімальний

Поряд із прямими грошовими вигодами, впровадження комплекс забезпечує і нефінансові ефекти. Локальна обробка даних нейромережею зводить затримку на реакцію до інциденту до мінімуму. Відсутність передачі конфіденційних пакетів за межі локального контуру автоматично підвищує рівень захисту від перехоплення даних.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Оюрана стратегія виходу на ринок базується на демонстрації технічної переваги TinyML-алгоритмів над хмарними аналогами. Просування розділено на три логічні етапи, що дозволяє поступово нарощувати довіру професійної спільноти та замовників.

Першочерговим кроком є запуск пілотних майданчиків (MVP) у реальних умовах. Візуалізація того, як мікроконтролер ESP32-S3 самостійно ідентифікує атаку та миттєво ізолює вузол, є ключовим фактором переконання технічних директорів (CTO). Прямий показ результатів роботи нейромережі на периферії

виключає скептицизм щодо обчислювальних можливостей вбудованих систем.

На другому етапі акцент зміщується на цифрову присутність у професійних мережах. Публікація оптимізованого коду на GitHub та опис архітектурних рішень у LinkedIn дозволяють залучити увагу системних адміністраторів та розробників IoT-рішень. Участь у профільних конференціях (наприклад, з питань кібербезпеки критичної інфраструктури) закріплює статус розробки як надійного вітчизняного продукту.

Заключна стадія передбачає налагодження B2B-партнерств із постачальниками обладнання для «розумних будинків» та системними інтеграторами Industrial IoT. Створення готових модулів, що легко інтегруються в наявні шафи керування, дозволяє реалізувати модель продажу «під ключ». Такий підхід мінімізує витрати на рекламу, фокусуючись на прямому вирішенні проблем безпеки конкретних підприємств.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Пріоритетним каналом збуту буде співпраця з компаніями, що займаються автоматизацією промислових об’єктів, де розроблена система впроваджується як готовий автономний модуль захисту. Це дозволяє використовувати наявні мережі збуту партнерів для швидкого охоплення ринку без значних витрат на власну логістику та маркетинг. Окремим шляхом реалізації є ліцензування програмного стека TinyML для сторонніх виробників електроніки, що дозволяє вбудовувати розроблені алгоритми безпосередньо в їхні апаратні платформи.

Оптимізація впровадження передбачає перехід до гібридної сервісної моделі, де одноразовий продаж обладнання поєднується з підпискою на оновлення нейромережевих моделей. Дистанційна доставка прошивок через захищений механізм OTA гарантує актуальність бази загроз на периферійних вузлах без необхідності фізичного доступу спеціаліста до пристроїв. Цифрова присутність у репозиторіях професійного спрямування та кейси в мережі LinkedIn допомагають

залучати технічних фахівців і скорочувати цикл ухвалення рішення про закупівлю. Такий підхід забезпечує сталий попит і стабільний розвиток проєкту завдяки низькій вартості масштабування та високій автономності розроблених рішень.

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключовий успіх розробленого комплексу полягає у поєднанні інтелектуальної автономності Edge AI та апаратної стійкості мікроконтролера ESP32-S3. Завдяки використанню векторних інструкцій архітектури Xtensa LX7 обробка мережевого трафіку відбувається практично миттєво, що дозволяє ідентифікувати загрози в реальному часі без звернення до хмари.

Надійність розробки підсилюється вбудованими функціями Secure Boot V2 та шифруванням пам'яті AES-256 XTS, які запобігають фізичному втручанню, тоді як протокол MQTTS із захистом TLS 1.3 гарантує конфіденційність керуючих команд. Висока економічна вигідність платформи у поєднанні з можливістю автономної ізоляції атак робить систему конкурентоспроможною, заповнюючи вільну нішу доступних інтелектуальних засобів захисту для IoT.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Регулювання трудових відносин у межах вітчизняного ІТ-сектору нерозривно пов’язане з виконанням вимог Закону України «Про охорону праці», що зобов’язує кожне підприємство до впровадження комплексних захисних заходів. Процес управління безпекою персоналу реалізується через систематизацію інструкцій за професіями, розробку внутрішніх положень, наказів та ведення журналів реєстрації інструктажів. Роботодавець забезпечує створення профільного відділу, функціонування якого повністю відповідає державній політиці та типовим нормативам центральних органів виконавчої влади. Ігнорування встановлених законодавчих правил тягне за собою адміністративну відповідальність у вигляді штрафів, а виникнення випадків травмування працівників призводить до притягнення відповідальних осіб до кримінальної відповідальності.

Основоположним інструментом деталізації безпекових норм є Наказ Міністерства соціальної політики № 207, що регламентує захист здоров’я під час роботи з екранними пристроями. Професійна діяльність інженерів-програмістів супроводжується специфічним негативним впливом на органи зору, високою розумовою напругою та інтенсивним нервово-емоційним навантаженням. Постійна взаємодія з периферійними пристроями введення створює значне статичне напруження м'язів та суглобів рук. Додаткову загрозу в процесі експлуатації апаратної частини обчислювальних систем становлять високочастотні електромагнітні випромінювання та погіршення складу повітря через виділення шкідливих газів.

Формування безпечних умов праці базується на положеннях ДСанПіН 3.3.2-007-98 та вимогах НПАОП 0.00-7.15-18, що передбачають ретельний

контроль параметрів мікроклімату, освітленості та ефективності вентиляції приміщень. Системний аналіз цих факторів дозволяє вчасно ідентифікувати потенційні причини виникнення професійних захворювань. Своєчасне визначення чинників шкідливого впливу є необхідним підґрунтям для розробки дієвої стратегії захисту організму розробника та підтримки його високої працездатності протягом усього життєвого циклу проєкту.

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Експлуатація апаратної платформи та супутнього обладнання створює умови для виникнення низки потенційно небезпечних факторів, серед яких пріоритетним визначено ризик ураження електричним струмом у разі порушення цілісності ізоляції блоків живлення. Оскільки проектування архітектури нейромережевих моделей та подальший аналіз мережевого трафіку супроводжуються інтенсивним зоровим навантаженням, особливої ваги набуває організація належної освітленості робочої зони. Робоче середовище інженера-розробника TinyML-рішень повинно відповідати встановленим параметрам мікроклімату, регулювання яких здійснюється згідно з нормами ДСанПіН 3.3.2.007-98 для забезпечення стабільної працездатності організму.

Процес розробки та валідації автономних шлюзів безпеки характеризується впливом специфічних високочастотних електромагнітних випромінювань, що генеруються радіомодулями Espressif у режимах активного сканування ефіру та передачі телеметрії. Поряд із технологічними ризиками, до яких належить імовірність виникнення пожежі через перегрів електронних компонентів або перевантаження силових контурів, існують значні психофізіологічні загрози. Інтелектуальне навантаження зумовлене необхідністю постійного розрізнення аномалій у складних часових рядах даних, що потребує високого ступеня концентрації уваги при мінімальних часових затримках. Невідповідність ергономічних показників робочого місця актуальним стандартам призводить до

виникнення статичних перевантажень кістково-м’язового апарату, зокрема в області суглобів кистей рук під час інтенсивного введення коду.

Діяльність фахівця з комп'ютерної інженерії класифікується як психічна форма праці з високим ступенем напруженості, що пов’язано з безперервним відстеженням динаміки інференсу нейромережі та читанням технічної документації. Монотонність виконання окремих операцій у поєднанні з акустичним шумом від систем охолодження серверного обладнання створює умови для накопичення нервово-емоційної втоми. Будь-яка активність у межах реалізації даного проєкту супроводжується мобілізацією вищих психічних функцій, оскільки ціна помилки при налаштуванні порогів чутливості нейродетектора може призвести до компрометації всієї IoT-інфраструктури. Таким чином, сукупний вплив кількох десятків шкідливих чинників вимагає впровадження дієвих заходів захисту для нейтралізації професійних ризиків.

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Оцінювання гігієнічної адекватності робочого простору базується на аналізі геометричних характеристик приміщення, де здійснюється експлуатація обчислювальної техніки та розробка нейромережевих моделей. Досліджувана робоча зона має ширину 5 метрів, довжину 6,2 метра та висоту 3,4 метра, що сумарно забезпечує загальну площу 31 м² та корисний об’єм 105,4 м³. Враховуючи одночасне перебування у приміщенні двох фахівців, на кожного працюючого припадає 15,5 м² площі та 52,7 м³ об’єму. Такі показники суттєво перевищують мінімальні пороги, встановлені ДСанПіН 3.3.2-007-98 та Наказом Міністерства соціальної політики № 207, що гарантує відсутність ефекту обмеженості простору та сприяє нормальному газообміну в процесі трудової діяльності.

Термічний режим середовища формується під впливом внутрішніх тепловиділень від апаратної частини ESP32-S3, системних блоків ЕОМ,

освітлювальних приладів та безпосередньо працюючого персоналу. Процес розробки TinyML-рішень класифікується як робота категорії Іа, що характеризується низькими енерговитратами організму на рівні до 120 ккал/год. Підтримка оптимальних параметрів мікроклімату реалізується через функціонування систем кондиціонування у теплий період та штучного опалення у холодну пору року. Фактична температура повітря у діапазоні 22–25 °С при відносній вологості 40–65 % повністю корелює з нормативними вимогами, забезпечуючи стабільну когнітивну активність розробника. Мінімізація запиленості досягається шляхом регламентованого провітрювання та проведення вологого прибирання робочих поверхонь.

Акустичний фон у приміщенні створюється переважно аеродинамічним шумом систем охолодження комп'ютерної техніки та періодичною роботою принтера. Сумарний рівень звукового тиску не виходить за межі допустимих значень, що дозволяє уникнути нервового виснаження та підтримувати високу концентрацію уваги при аналізі великих масивів даних. Окремим критичним чинником виступає світлотехнічне забезпечення робочої зони, що регулюється нормами ДБН В.2.5-28:2018. Робота програміста ідентифікована як діяльність V розряду зорової роботи підрозряду В, що вимагає штучної освітленості на рівні 300 Лк та коефіцієнта природної освітленості не менше 1,5 %. Рівномірний розподіл світлового потоку та ідентичність яскравості екранів з навколишнім фоном мінімізують напруження периферійного зору та запобігають розвитку передчасної втоми в умовах тривалого виконання складних інженерних завдань.

8.4 Розробка заходів з поліпшення умов охорони праці

Проведений аналіз санітарно-гігієнічного стану лабораторії дозволяє констатувати відповідність геометричних параметрів приміщення, показників мікроклімату та акустичного фону встановленим державним стандартам. Основним чиником потенційного зниження працездатності дослідника визначено

психофізіологічний фактор, зумовлений високою когнітивною складністю квантування нейромережевих моделей та налагодження низькорівневого коду для архітектури Xtensa LX7. Мінімізація негативного впливу інтелектуального перенапруження досягається через суворе доотримання регламентованого режиму праці та відпочинку, а також створення сприятливої психологічної атмосфери в інженерному колективі.

Організація робочого місця розробника Edge AI рішень базується на принципах ергономіки, що передбачає раціональне розміщення лабораторних стендів з модулями та контрольно-вимірювальної апаратури. Важливою складовою безпеки є регулярне наочне ознайомлення персоналу зі шляхами евакуації згідно з затвердженим планом, який повинен бути розміщений на доступному для огляду місці. Програма цивільного захисту передбачає включення до складу аптечок першої допомоги засобів для проведення штучного дихання, зокрема масок-клапанів, що є критичним при наданні допомоги у разі виникнення непередбачуваних ситуацій у закритих приміщеннях обчислювальних центрів.

Електротехнічна безпека при роботі з налагоджувальними платами та силовими блоками живлення забезпечується шляхом регулярної перевірки параметрів захисного заземлення та занулення. Вимірювання опору заземлювального ланцюга проводиться з періодичністю, встановленою нормами ПТЕЕС, для запобігання накопиченню статичного заряду на корпусах пристроїв. Розподільні щити лабораторії оснащуються спеціалізованими розетками з інтегрованими заземлюючими контактами, а всі прилади, що функціонують під напругою понад 36 В, підлягають обов’язковому заземленню.

З огляду на специфічний ризик виникнення фібриляції шлуночків серця при випадковому контакті з відкритими струмоведучими частинами обладнання, доцільним є оснащення лабораторії автоматичним зовнішнім дефібрилятором. Ефективність даного заходу гарантується лише за умови попередньої підготовки персоналу до роботи з реанімаційним обладнанням. Запропонований комплекс заходів дозволяє нейтралізувати дію ідентифікованих шкідливих чинників та

забезпечити високу надійність функціонування системи в умовах тривалої інженерної експлуатації.

8.5 Розрахункова частина

Розрахунок параметрів штучного освітлення лабораторії TinyML виконано за методом коефіцієнта використання світлового потоку. Обрана методика дозволяє врахувати не лише пряме випромінювання світильників, а й відбиття енергії від внутрішніх поверхонь приміщення шириною 5 м, довжиною 6,2 м та висотою 3,4 м. Для забезпечення комфортних умов при монтажі мікросхем ESP32-S3 та розробці програмного коду встановлено нормовану мінімальну освітленість на рівні 300 Лк.

Розрахункову площу об’єкта визначено як добуток його лінійних розмірів, що становить 31 м². Оскільки в процесі експлуатації спостерігається поступове зниження ефективності випромінювачів через запилення та деградацію світлодіодів, у розрахунок введено коефіцієнт запасу 1,5 та поправковий коефіцієнт нерівномірності 1,1. Параметри відбиття стін та стелі прийняті на рівні 50%, що відповідає типовому світлому оздобленню сучасних обчислювальних центрів.

Для визначення інтегральних характеристик середовища обчислено індекс приміщення, який при висоті підвісу світильників 3 м складає 0,43. Відповідно до отриманого індексу та обраного типу ламп встановлено коефіцієнт використання світлового потоку, що дорівнює 0,23. Загальний розрахунковий світловий потік, необхідний для створення нормованого середовища у приміщенні, де працює 5 осіб, становить 66717 Лм.

За апаратну основу системи освітлення обрано світлодіодні панелі PL PFM 600 потужністю 30 Вт із номінальним світловим потоком 3000 Лм кожна. Шляхом зіставлення сумарної потреби об’єкта в освітленні з одиничною потужністю пристрою визначено необхідну кількість одиниць обладнання. Результат

обчислень вказує на потребу у 22,18 джерелах світла, що при округленні у більшу сторону вимагає встановлення 23 світлодіодних світильників.

Висновки до розділу

Комплексний аналіз умов праці інженера-програміста підтвердив критичну роль ергономічних та санітарно-гігієнічних факторів у забезпеченні стабільності розробки ІТ-проєктів. Ідентифікація шкідливих чинників, таких як електромагнітне випромінювання мікроконтролерів ESP32-S3 та зорова напруга, дозволила сформувати дієву стратегію захисту персоналу. Математичне обґрунтування системи освітлення гарантує підтримку зорового комфорту, тоді як розроблені заходи з електробезпеки мінімізують технічні ризики при експлуатації обладнання. Реалізація запропонованих рішень сприяє не лише збереженню здоров'я спеціалістів, а й підвищенню загальної продуктивності праці за рахунок оптимізації робочого середовища.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, розроблене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначене для функціонування апаратно-технічних засобів отримання фотозображення, його стилізації та відтворення за допомогою реалізації пристрою IoT плоттеру.

В межах сучасного ринку спостерігається дефіцит комплексних рішень, здатних забезпечити повний цикл трансформації візуального образу від фотографії до фізичного малюнка без участі оператора, що зумовлює актуальність даного дослідження.

У випускній кваліфікаційній роботі наведено теоретичне узагальнення та розв'язання науково-технічного завдання щодо інтеграції нейромережевих алгоритмів обробки зображень у системи числового програмного керування (ЧПК).

Досягнення поставленої мети базувалося на послідовному виконанні таких етапів:

– Проведено аналітичний огляд існуючих методів векторизації растрових зображень та архітектурних рішень для CNC-систем;

– Досліджено методи оптимізації генеративних моделей Stable Diffusion за допомогою адаптерів ControlNet для збереження геометричної точності контурів;

– На основі отриманих результатів створено програмну реалізацію комплексу стилізації та векторизації зображень безпосередньо на керуючому вузлі (Host-системі);

– Розроблені алгоритми спірального пошуку та спрощення ламаних, адаптовані під специфіку плотерів, дозволяють успішно вирішувати завдання перетворення піксельних масивів у векторні траєкторії з мінімальною кількістю холостих ходів.

У ході аналізу предметної галузі було ідентифіковано ключові об’єкти взаємодії, визначено їхні функціональні характеристики та побудовано математичну модель перетворення графічної інформації.

Розроблене програмне забезпечення вирізняється інтуїтивно зрозумілим графічним інтерфейсом, реалізованим на базі бібліотеки PyQt6, що забезпечує легкість освоєння продукту користувачем без специфічної підготовки у сфері комп’ютерної графіки чи робототехніки.

При створенні системи застосовано об’єктно-орієнтований підхід та сучасні парадигми проектування подіє-орієнтованих систем.

Програмний комплекс реалізовано з використанням мови високого рівня Python для керуючого ядра та стандартизованого протоколу G-code для взаємодії з виконавчим механізмом. Це дозволило досягти максимальної ефективності обробки графічних даних, мінімізувати терміни розробки та забезпечити кросплатформеність рішення.

Система призначена для роботи у зв’язці з персональним комп’ютером (сервером інференсу) та безпосередньо апаратним контролером плотера через послідовний інтерфейс.

Для забезпечення безкомпромісного рівня надійності в роботі запропоновано та реалізовано комплекс програмних засобів контролю передачі даних, зокрема механізми підтвердження виконання команд (Handshaking) та буферизації потоку. Це унеможливлює переповнення пам’яті мікроконтролера та забезпечує плавність руху механічних вузлів, запобігаючи аварійним ситуаціям під час друку.

В цілому створене апаратно-програмне забезпечення підтверджує коректність обраних проектних рішень та повністю відповідає вимогам технічного завдання.

Розробка має значний потенціал для подальшого масштабування, включаючи можливість інтеграції нових стилізаційних моделей LoRA та підтримку віддаленого керування через веб-інтерфейс.

Проведене маркетингове та економічне обґрунтування підтвердило високу інвестиційну привабливість проєкту та визначило ключові фактори його ринкового успіху у сферах розваг та автоматизованого дизайну.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Радін Д.О. Дослідження та реалізація апаратно-технічних засобів отримання фотозображення, його стилізації та відтворення за допомогою реалізації пристрою IoT плоттеру // Збірник праць молодих науковців ЦНТУ. – Вип. 15. – Кропивницький: ЦНТУ, 2025.

2. Peter Hoddie, Lizzie Prader «IoT Development for ESP32 and ESP8266 with JavaScript: A Practical Guide to XS and the Moddable SDK» ISBN-13 (pbk): 978-1-4842-5069-3 ISBN-13 (electronic): 978-1-4842-5070-9

3. STM32CubeMX for STM32 configuration and initialization C code generation. User manual. June 2022. 397 p.

4. І.В.Чихіра, А.Г. Микитишин Конспект лекцій з дисципліни «Програмування систем реального часу» / Укладачі : Чихіра І.В., Микитишин А.Г., – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя , 2016. – 76 с.

5. Jack Ganssle and Michael Barr. 2003. Embedded Systems Dictionary. CMP Books.

6. Технології інтернету речей. Навчальний посібник [Електронний ресурс]: / Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 12,5 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 271 с.

7. Greg Dunko, Joydeep Misra, Josh Robertson, Tom Snyder “A reference guide to the Internet of Things” / 2017 Bridgera LLC, RIoT.

8. Donald Norris “Programming with STM32. Getting started with the Nucleo Board and C/C++” 416 p. 2018.

9. Neil Kolban “Kolban’s book on ESP32”. Texas, USA. 951 p.

10. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical systems». Security and Privacy of Cyber Physical Systems Emerging Trends

Technologies and Applications, 2025, pp. 193–224.

11. Kuznetsov, O., Atzeni, G., Arnesano, M., Randieri, C., Smirnov, O. «Secure IoT-based smart wheelchair system: From implementation to security enhancement strategy». Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications, 2025, pp. 225–257.

12. Kuznetsov, O., Smirnov, O., Kuznetsova, T., Shaikhanova, A., Svatowsky, I. «Privacy-utility trade-offs in IoT networks: A comparative analysis of differential privacy mechanisms for sensor data aggregation». Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications, 2025, pp. 589–622.

13. Вінтенко Б., Смірнов О., Миронець І., Смірнова Т., Смірнов С. «Імітаційна модель шляхів вхідних даних комп'ютерної інтелектуальної системи підтримки оператора енергоблоку АЕС». Комбінаторні конфігурації та їхні застосування: Матеріали XXVII Міжнародного науково-практичного семінару, присвяченого 125-річчю Національного університету «Запорізька політехніка» (Запоріжжя-Кропивницький-Київ, 4-6 червня 2025 р.). Запоріжжя: НУ «Запорізька політехніка», 2025. С.82-91.

14. Al-Azzeh, J., Ayyoub, B., Mesleh, A., Smirnova, T., Gnatyuk, S., Drieiev, O., Smirnov, O., Dorenskyi, O. «Cloud-Based Information System for Evaluating Caverns in the Process of Blasting Metal Surfaces of Details». International Review on Modelling and Simulations 18 (1), 2025. pp. 32-42.

15. Вінтенко Б.Ю., Смірнов О.А., Миронець І.В., Смірнова Т.В., Коваленко О.В., Мацуй А.М. «Модель шляхів отримання вхідних даних комп'ютерної інтелектуальної системи підтримки оперативного персоналу АЕС». Центральноукраїнський науковий вісник. Технічні науки. 2025. Вип. 11(42), ч. II. С.52-62.

16. Вінтенко Б.Ю., Смірнов О.А., Миронець І.В., Смірнова Т.В. «Методи забезпечення відмовостійкості інтелектуальних систем підтримки оператора». VIII міжнародна науково-практична конференція "Інформаційна

безпека та комп'ютерні технології", м. Кропивницький. 24-25 квітня 2025 р. – Кропивницький: ЦНТУ. – 2025. – С. 44-46.

17. Вінтенко, Б., Миронець, І., Смірнов, О., Коваленко, А., Конопліцька-Слободенюк, О., Смірнова, Т., Константинова, Л. «Дослідження застосування систем підтримки оперативного персоналу об'єкту критичної інфраструктури при керуванні енергоблоком АЕС з реактором типу ВВЕР-1000». Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 2024. № 2(26), С. 6-26.

18. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». Кібербезпека: освіта, наука, техніка. 2024. №4(24), С. 6-27.

19. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». Кібербезпека: освіта, наука, техніка. 2024. №3(23), С. 111-131.

20. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». Advanced Information Systems, 2023, 7(2), pp. 49-56.

21. Kuznetsov, O., Kuznetsova, Y., Smirnov, O., Kostenko, O., Zvieriev, V. «Evaluating Hashing Algorithms in the Age of ASIC Resistance». CEUR Workshop Proceedings, 2023, 3628, pp. 93-105.

22. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». CEUR Workshop Proceedings, Volume 3530, 2023, pp. 256-265.

23. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

24. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 3(73), С. 155-166.

25. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.

26. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

27. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». Проблеми інформатизації та управління, № 2(70). 2022. С. 28-37.

28. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» Системи управління, навігації та зв'язку, 2022, № 3(69). С. 93-98.

29. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» Вісник Хмельницького національного університету. Серія: «Технічні науки», № 2 (307). С. 46-52. 2022.

30. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу

запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» Системи управління, навігації та зв'язку, 2022, № 1(67). С. 84-89.

31. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». Communications in Computer and Information Science, 2021, vol 1486. Springer, Cham. pp 169-184.

32. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

33. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». Journal of theoretical and applied information technology Vol.98. No 21, 2020, P. 3334-3346.

34. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

35. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

36. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». CEUR Workshop Proceedings Volume 2616, 2020, Pages 125-136.

37. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». CEUR Workshop Proceedings Volume 2616, 2020, Pages 366-379.

38. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S.,

Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

39. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

40. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

41. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

42. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

43. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

44. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

45. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019

IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 353-358.

46. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

47. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», CEUR Workshop Proceedings Volume 2353, CEUR Workshop Proceedings 2019, Pages 618-629.

48. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», Telecommunications and Radio Engineering. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

49. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». Сучасні інформаційні системи. 2021. Т. 5, № 4. С. 79-95

50. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

51. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.