Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет__Комп’ютерної інженерії та управління_____
(повна назва)
Кафедра____Автоматизації проектування обчислювальної техніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____
Моделі та інструментальні засоби для створення піксельних картин_____
_____
(тема)

Виконав:
здобувач __2025_____ року навчання,
групи ____СКСм-23-1_____
_____Мідіна Сергій _____
(прізвище, ініціали)
Спеціальність___123 – Комп’ютерна інженерія_____
(код і повна назва спеціальності)
Тип програми__ __освітньо-професійна __
(освітньо-професійна або освітньо-наукова)
Освітня програма__Спеціалізовані комп’ютерні системи_____
_____
(повна назва освітньої програми)
Керівник _доц. Ларченко Л.В._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С.В.
(підпис) (прізвище, ініціали)

2025 р.

`

Харківський національний університет радіоелектроніки

Факультет___Комп'ютерної інженерії та управління_____

Кафедра_____Автоматизації проектування обчислювальної техніки_____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність__123 – Комп'ютерна інженерія_____
(код і повна назва)

Тип програми __освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма___Спеціалізовані комп'ютерні системи_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві_____Мідіні Сергію Сергійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи ___Моделі та інструментальні засоби для створення піксельних картин_____
_____

затверджена наказом університету від "_08_"__11__2024 р. №_1189 Ст_____

2. Термін подання здобувачем роботи до екзаменаційної комісії ____23. 01.2025 р._____

3. Вихідні дані до роботи _____
Піксельна картина_____
Arduino UNO_____
Bluetooth модуль HC-06_____
Android_____
Мови програмування_Kotlin, C++_____
_____

4. Перелік питань, що потрібно опрацювати в роботі _____
Огляд піксельної графіки та графічних редакторів_____
Аналіз алгоритмів для створення піксельних картин_____
Вибір інструментальних засобів, розробка структури системи для створення моделей піксельних картин_____
Програмна реалізація та тестування розробленої системи_____
_____

`

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)__
18 слайдів

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання	02.09.2024 – 02.09.2024	
2	Аналіз проблемної галузі, постановка завдання	03.09.2024 – 10.09.2024	
3	Дослідження графічних редакторів	11.09.2024 – 18.09.2024	
4	Вибір інструментальних засобів та розробка структурної та функціональної схем системи	19.09.2024 – 10.10.2024	
5	Програмна реалізація мобільного застосунку	11.11.2024 – 25.11.2024	
6	Тестування розробленої системи	26.11.2024 – 09.12.2024	
7	Оформлення пояснювальної записки	10.12.2024 – 23.12.2024	
8	Оформлення графічного матеріалу	24.12.2024 – 02.01.25	
9	Перевірка виконаного проекту керівником	03.01.2025 – 10.01.2025	
10	Захист кваліфікаційної роботи	23.01.2025	

Дата видачі завдання __02 вересня_2024 р.

Здобувач _____
(підпис)

Керівник роботи ___ _____ ___доц. Ларченко Л.В.____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 67 сторінок, 35 рисунків, 3 таблиці, 1 лістингів, 15 джерел за переліком посилань.

ПІКСЕЛЬ, ГРАФІКА, ВІДЕОІГРИ, МИСТЕЦТВО, АНІМАЦІЯ, ГРАФІЧНІ РЕДАКТОРИ, МОДЕЛЬ

В кваліфікаційній роботі досліджено сучасні інструментальні засоби піксельної графіки та розроблено систему для створення піксельних картин. Основна увага була зосереджена на аналізі піксельної графіки та інструментів, які використовуються для її розробки, таких як Asperite, Piskel і Adobe Photoshop. Було досліджено ключові алгоритми, серед яких – алгоритм Брезенхена для малювання ліній, алгоритм серединної точки для малювання кіл та еліпсів, а також методи згладжування та заливки. Розглянуто процеси створення піксельних зображень і важливість алгоритмічної оптимізації для досягнення високої якості кінцевого продукту.

Розроблено систему для створення піксельних картин, компонентами якої є: мікроконтролер Arduino UNO, що забезпечує управління LED матрицею, Bluetooth модуль HC-06 для зв’язку з Android застосунком, LED матриця 16x16 для виведення піксельних зображень, а також Android застосунок для налаштування та передачі зображень.

`

ABSTRACT

Explanatory note contains 67 pages, 35 figures, 3 tables, 1 listings, 15 sources according to the list of references.

PIXEL, GRAPHICS, VIDEO GAMES, ART, ANIMATION, GRAPHICS EDITORS, MODEL

The purpose of thesis is to research modern means of pixel graphics and develop a system for creating pixel pictures based on a microcontroller, LED matrix and related components.

The main focus was on the analysis of pixel graphics and the tools used to develop them, such as Asperite, Piskel and Adobe Photoshop. In the course of the work, key algorithms were investigated, including the Bresenchen algorithm for drawing lines, the midpoint algorithm for drawing circles and ellipses, as well as smoothing and filling methods. As a result of the work, a clear idea of the processes of creating pixel images and the importance of algorithmic optimization to achieve high quality of the final product was obtained.

A system has been created, the components of which are an Arduino UNO microcontroller that provides control of the LED matrix, a Bluetooth module HC-06 for communication with an Android application, a 16x16 LED matrix for displaying pixel images, and an Android application for configuring and transferring images. The system also provides for the use of necessary materials for the physical implementation of a pixel picture. A block diagram shows the connections between components, while a functional diagram shows how data transfer and color control takes place in real time.

`

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8
ВСТУП 9
1 ОГЛЯД ПІКСЕЛЬНОЇ ГРАФІКИ ТА ГРАФІЧНИХ РЕДАКТОРІВ 11
1.1. Огляд піксельної графіки 11
1.2. Галузі використання засобів комп’ютерної графіки 13
1.2.1 Відеоігри 13
1.2.2 Мобільні застосунки та інтерфейси 14
1.2.3 Веб-дизайн 14
1.2.4 Рекламні та промоційні матеріали 15
1.2.5 Діджитал-арт анімація 16
1.3 Графічні редактори для піксельного мистецтва 16
1.3.1 Asperite 17
1.3.2 Piskel 18
1.3.3 Adobe Photoshop 20
1.4 Мета та постановка завдання 21
2 АЛГОРИТМІЧНІ РІШЕННЯ ДЛЯ ПІКСЕЛЬНОГО МИСТЕЦТВА 23
2.1 Алгоритм Брезенхенма для малювання ліній 23
2.2 Алгоритм серединної точки для малювання кіл та еліпсів 26
2.3 Алгоритм заливки областей 29
2.4 Алгоритм згладжування 30
3 СТРУКТУРНА ТА ФУНКЦІОНАЛЬНА СХЕМИ СИСТЕМИ ДЛЯ СТВОРЕННЯ МОДЕЛЕЙ ПІКСЕЛЬНИХ КАРТИН 33

3.1 Компоненти системи 33
3.1.1 Мікроконтролер Arduino UNO 33
3.1.2 Bluetooth модуль HC-06 35
3.1.3 Світлодіодна матриця 36
3.1.4 Компоненти системи управління-матрицею 37
3.1.5 Матеріали для фізичної піксельної картини 38
3.2 Структурна та функціональна схеми з'єднання компонентів системи 39
4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ СТВОРЕННЯ МОДЕЛЕЙ ПІКСЕЛЬНИХ КАРТИН 42
4.1 Android застосунок 42
4.1.1 Середовище розробки 42
4.1.2 Архітектура застосунку 44
4.1.3 Функціональність застосунку 47
4.1.4 Технічна складова застосунку 53
4.2 Arduino 59
4.2.1 Середовище розробки 60
4.2.2 Технічна складова контролера 61
ВИСНОВКИ 64
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 66
ДОДАТОК А 68
ДОДАТОК Б 77

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – набір правил і методів для взаємодії між програмами або сервісами.

CDN (Content Delivery Network) – мережа серверів, розташованих у різних регіонах світу, яка забезпечує швидку доставку контенту користувачам

IoT (Internet of Things) – мережа фізичних пристроїв, які підключені до Інтернету для збору, обміну та аналізу даних.

IPC (Inter-Process Communication) – механізм, який дозволяє процесам обмінюватися даними між собою.

MVVM (Model-View-ViewModel) – архітектурний патерн для розробки застосунків, який розділяє логіку програми, дані та представлення користувацького інтерфейсу.

NFT (Non-Fungible Token) – унікальний цифровий актив на блокчейні, який підтверджує право власності на об'єкт.

RX (Receiver) – один з каналів зв’язку, що забезпечує прийом даних, що передаються ззовні.

SDK (Software Development Kit) – набір інструментів, утиліт і документації для створення застосунків.

TX (Transmitter) – один з каналів зв’язку, який є джерелом передачі даних

UART (Universal Asynchronous Receiver-Transmitter) – вузол обчислювальних пристроїв, призначений для організації зв’язку з іншими цифровими пристроями.

URL (Uniform Resource Locator) – адреса, яка визначає розташування ресурсу в Інтернеті та спосіб доступу до нього.

ВСТУП

Актуальність теми кваліфікаційної роботи, яка пов’язана з розробкою моделей та інструментальних засобів для створення піксельних картин, полягає в тому, що піксельне мистецтво є важливою складовою цифрової культури, яка набуває подальшої популярності в багатьох сферах, включаючи відеоігри, дизайн та медіа. Піксельний стиль графіки дозволяє створювати унікальні ретро-візуалізації, що відзначаються простотою та високою естетичною виразністю.

З розвитком технологій і доступом до різноманітних програмних інструментів для створення піксельних зображень, як-от Aseprite, Piskel та інші, художники мають змогу легко втілювати свої ідеї.

Автоматизовані моделі штучного інтелекту, такі як генератори зображень на основі машинного навчання, дозволяють створювати піксельні картини швидше та точніше, знижуючи технічні обмеження для користувачів.

Актуальність теми також обумовлена тим, що піксельне мистецтво стало невід'ємною частиною культури інді-ігор, де художники можуть експериментувати з оригінальними стилями, які не залежать від ресурсів великих студій.

Дослідження та розвиток моделей і інструментів для створення піксельних зображень сприяють подальшому розвитку цієї форми мистецтва, а також надають можливості як для професійних художників, так і для аматорів створювати унікальні цифрові твори. Таким чином, тема роботи є актуальною як з погляду технологічного прогресу, так і з культурної точки зору, оскільки піксельна графіка є важливим елементом сучасної цифрової творчості.

Метою кваліфікаційної роботи є огляд алгоритмічних ріщень для піксельного мистецтва та розробка системи для створення моделей

піксельних картин за допомогою мікроконтролера, модуля Bluetooth, LED матриці та Android застосунку з використанням відповідних графічних засобів.

1 ОГЛЯД ПІКСЕЛЬНОЇ ГРАФІКИ ТА ГРАФІЧНИХ РЕДАКТОРІВ

1.1. Огляд піксельної графіки

Піксельна графіка є одним з найбільш поширених видів цифрової графіки, особливо в умовах обмеженої роздільної здатності або ресурсів апаратного забезпечення. Цей тип графіки базується на використанні пікселів – найменших структурних елементів зображення, які представляють собою одноколірні точки. З точки зору комп'ютерних технологій, піксельна графіка оперує матрицею пікселів, де кожен піксель займає певну позицію на сітці екрану та має визначений колір. Чіткість зображення, яка складається з пікселів, залежить від роздільної здатності, що визначає кількість пікселів по горизонталі та вертикалі. Наприклад, зображення з роздільною здатністю 800x600 складається з 480,000 пікселів. Чим більша кількість пікселів у зображенні, тим вищою є його деталізація [1].

Окрім позиціонування пікселів, піксельна графіка також базується на виборі кольорової палітри. Кольорова палітра визначає обмежену кількість кольорів, які можуть бути використані у створенні зображення. Одним з характерних аспектів класичної піксельної графіки є обмежена кількість кольорів, що сприяло розробці стилістично цілісних робіт з виразною візуальною мовою. У випадку обмежених ресурсів комп'ютера або консолі, класичні відеоігри використовували палітри, обмежені до 16 або 256 кольорів. Сучасні технології дозволяють використовувати значно більші палітри, але основні принципи залишаються незмінними – ефективне використання кожного пікселя для передачі змісту та атмосфери.

Проте, у порівнянні з векторною графікою, піксельна графіка має важливі обмеження: вона погано масштабується. При спробах збільшити зображення його чіткість втрачається, з’являється ефект пікселізації. Векторна ж графіка базується на математичних обчисленнях і легко масштабується без втрати

якості. Це робить її ідеальним вибором для створення логотипів, іконок та інших елементів дизайну, які мають залишатися чіткими незалежно від розміру.

Розглядаючи 3D-графіку треба зауважити, що вона суттєво відрізняється від піксельної за своєю природою. Якщо піксельна графіка працює у двовимірному просторі, то 3D-графіка використовує тривимірні об’єкти та простір, що дозволяє створювати реалістичні моделі і сцени. 3D-графіка активно застосовується у фільмах, відеоіграх та анімації, оскільки дає можливість створювати глибину, складні об’єкти і реалістичні ефекти освітлення. Однак, 3D-графіка потребує значних обчислювальних ресурсів для рендерингу та обробки, на відміну від піксельної, яка є більш простою і менш вимогливою до апаратних можливостей [2].

Ще одним важливим видом графіки є растрова графіка. Вона, як і піксельна, складається з пікселів, однак різниця полягає у деталізації. Растрова графіка використовується для фотографій та складних зображень, де кожен піксель може мати унікальний колір, створюючи більш реалістичну картину. Водночас, як і у випадку з піксельною графікою, растрові зображення також втрачають якість при збільшенні. Проте піксельна графіка, на відміну від растрової, має чіткі межі та обмежену кількість кольорів, що робить її менш реалістичною, але більш стилізованою.

Фрактальна графіка, у порівнянні з піксельною, є значно складнішою. Вона базується на математичних формулах, які дозволяють створювати нескінченні деталі при масштабуванні. Це зовсім інший підхід до створення зображень, де кожна частина зображення містить ту ж структуру, що й ціле, що робить фрактальну графіку дуже ефективною для створення природних об'єктів, таких як хмари, гори чи узори.

Отже, піксельна графіка має свої унікальні риси та стиль, що робить її відмінною від інших видів графіки. Вона проста, доступна і дає художникам великий контроль над кожним елементом зображення, однак має обмеження щодо масштабування та деталізації, що вигідно відрізняє її від векторної, 3D, растрової та фрактальної графіки [3].

1.2. Галузі використання засобів комп’ютерної графіки

Піксельна графіка, хоч і є відносно простим видом цифрового мистецтва, продовжує займати важливе місце в різних сферах і надалі розвивається та є затребуваною. Вона вирізняється своєю мінімалістичністю та високою виразністю, що робить її ефективним інструментом для художників, дизайнерів та розробників. Піксельна графіка знайшла своє застосування у багатьох ключових сферах[4].

1.2.1 Відеоігри

Однією з найвідоміших сфер використання піксельної графіки є відеоігри, особливо інді-ігри та ретро-ігри [5]. Піксельна графіка була основою візуальних стилів ранніх відеоігор у 1980-1990-х роках через технічні обмеження та продовжує використовуватися, навіть в епоху 3D-графіки, і на сьогоднішній день спостерігається оновлений інтерес до даного виду графіки [6]. Завдяки своїй простоті та естетичній привабливості, піксельна графіка часто використовується для створення ретро-естетики в таких сучасних іграх, як Undertale, Celeste (рис.1.1), Shovel Knight. Даний стиль дозволяє інді-розробникам створювати захоплюючі візуальні ефекти при мінімальних витратах на ресурси [7].

Рисунок 1.1 - Приклад піксельної відеогри “Celeste”

1.2.2. Мобільні застосунки та інтерфейси

Піксельна графіка широко використовується в мобільних застосунках та інтерфейсах завдяки своїй легкості та зрозумілості. У мобільних застосунках з обмеженими ресурсами вона може бути більш ефективною, оскільки займає менше пам’яті та швидше рендериться. Також її використовують для створення простих і чітких іконок, кнопок та інших елементів інтерфейсу, що дозволяє досягти високого рівня зручності для користувачів [8]. (рис.1.2).

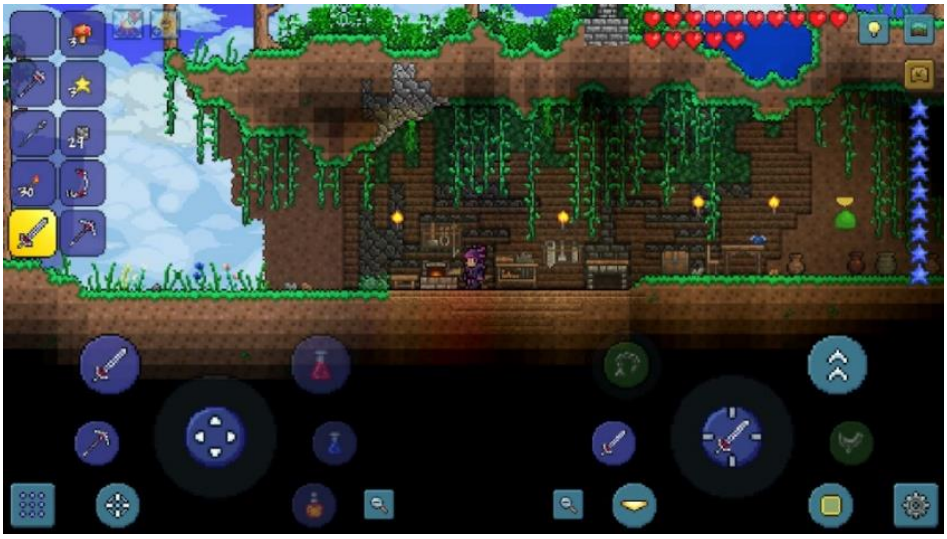

Рисунок 1.2 - Приклад мобільної піксельної відеогри “Terraria”

1.2.3. Веб-дизайн

Піксельна графіка також знаходить своє застосування у веб-дизайні, особливо в стилізованих або мінімалістичних вебсайтах, де важлива проста і чітка візуалізація. Деякі веб-сайти використовують піксельну графіку, щоб створити ретро-атмосферу або оригінальний вигляд, який відрізняється від сучасних трендів (рис1.3). Піксельні анімації або інтерактивні елементи можуть також ефективно впливати на користувацький досвід, роблячи сайт унікальним.

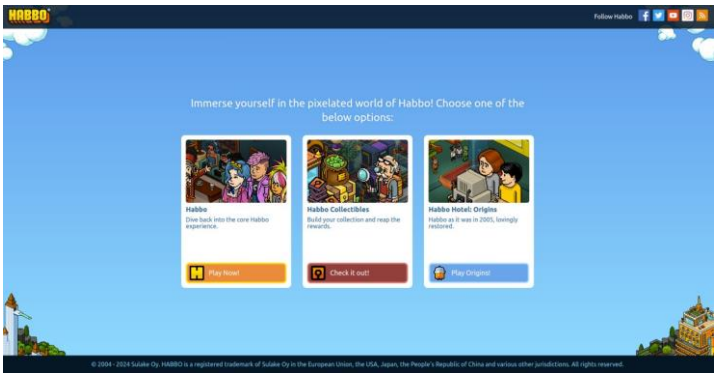

Рисунок 1.3 - Приклад піксельного веб-сайту “Hobbo Hotel”

1.2.4. Рекламні та промоційні матеріали

Завдяки своїй впізнаваній естетиці, піксельна графіка часто використовується в рекламних та промоційних матеріалах, особливо для продуктів чи кампаній, що орієнтовані на молодіжну або геймерську аудиторію.

Піксельний стиль допомагає брендам виділитися серед сучасних рекламних потоків, викликаючи почуття ностальгії та відсилаючи до епохи класичних відеоігор (рис1.4).

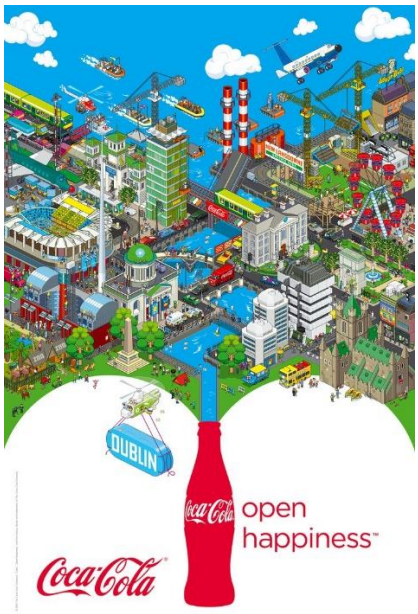

Рисунок 1.4 - Приклад піксельної реклами “Coca Cola”

1.2.5. Діджитал-арт та анімація

Піксельна графіка використовується багатьма сучасними цифровими художниками для створення стилізованих зображень, персонажів та анімацій. Вона є основою для таких жанрів, як "піксель-арт", де художники працюють з окремими пікселями для створення візуальних ефектів з використанням мінімальної кількості елементів.

Піксельна анімація стала популярною в інтернет-культурі, завдяки своїй простоті та доступності для початківців художників, оскільки створення таких анімацій не вимагає складних інструментів. NFT (non-fungible token) є важливою частиною діджитал-арту та анімації. NFT дозволяють художникам і аніматорам монетизувати свої цифрові твори, створюючи унікальні, перевірені записи про власність на цифрові активи, які можна купувати, продавати або обмінювати на блокчейні (рис.1.5).

Рисунок 1.5 - Приклад NFT картинки

1.3. Графічні редактори для піксельного мистецтва

У роботі розглянуто популярні інструменти для створення піксельного мистецтва, як Asperite, Piskel та Adobe Photoshop, використання яких значно полегшує процес створення піксельної графіки, дозволяючи працювати з

кольоровою палітрою, шарами та інструментами для автоматизації повторюваних завдань.

1.3.1. Asperite

Графічний редактор Aseprite створений спеціально для тих, хто працює з піксельною графікою. Всі функції та інструменти редактора зосереджені на точному контролі кожного пікселя, що робить його одним з найзручніших і найпотужніших редакторів у своїй ніші.

Одна з найбільших переваг Aseprite – це його інтуїтивний інтерфейс, який легко освоїти. Всі основні інструменти знаходяться під рукою, а налаштування та функції розроблені для того, щоб художники могли швидко та ефективно створювати піксельне мистецтво. Особливо корисним є інструмент "олівець", який дозволяє точно малювати окремі пікселі, а також можливість працювати з шарами та анімаціями. Шари – важливий інструмент для піксельного мистецтва в Aseprite, який дозволяє працювати над різними елементами зображення або анімації окремо. Це спрощує процес редагування і дозволяє художникам легко додавати чи змінювати окремі частини малюнка без ризику пошкодити інші елементи.

Для тих, хто працює з іграми чи анімацією, Aseprite пропонує потужні інструменти для створення спрайтів. Можливість створювати і редагувати анімаційні цикли безпосередньо у програмі дозволяє швидко експортувати результат у формати, зручні для використання в ігрових рушіях, таких як GIF або спрайт-листи.

Ще однією сильною стороною Aseprite є підтримка кастомізованих палітр кольорів, що дозволяє художникам створювати індивідуальні набори кольорів для своїх проектів. Це особливо корисно для ретро-естетики, де обмежені колірні палітри додають характеру піксельній графіці.

Aseprite також дозволяє працювати з прозорістю, що є важливою особливістю при створенні спрайтів для ігор або анімацій, де різні елементи повинні мати прозорий фон.

Однак є і недоліки, характерні для редактора Aseprite. Для деяких користувачів може бути важливим те, що, хоча програмне забезпечення редактора добре підходить для піксельної графіки, вона менш ефективна для інших типів цифрового мистецтва або великих зображень з високою роздільною здатністю. Крім того, Aseprite є платним програмним забезпеченням, хоча він доступний за досить низьку ціну.

Незважаючи на недоліки, Aseprite – це ідеальний інструмент для художників, які спеціалізуються на піксельному мистецтві та анімації. Він простий у використанні, підтримує всі необхідні функції для створення професійної піксельної графіки і забезпечує точний контроль над кожним пікселем, що робить його одним із найкращих рішень для цієї галузі (рис.1.6).

Рисунок 1.6 – Приклад піксельної картини, створеної в Asperite

1.3.2. Piskel

Piskel – це веб-застосунок, який дозволяє створювати піксельне мистецтво прямо в браузері, без необхідності завантажувати програмне забезпечення. Оскільки це онлайн-інструмент, він доступний з будь-якого пристрою, підключеного до інтернету, і не вимагає інсталяції або потужних апаратних ресурсів.

Однією з головних переваг Piskel є його простота та інтуїтивний інтерфейс. Усі основні функції легко доступні, що робить цей редактор привабливим для новачків, які тільки починають працювати з піксельною

графікою. Інструменти, такі як "олівець", "ластик" та "заливка", дозволяють легко малювати та редагувати кожен піксель.

Piskel також підтримує анімовані спрайти. Створення анімацій у ньому дуже просте: кожен кадр (frame) можна редагувати окремо, а програма автоматично створює плавний анімаційний цикл. Інтерфейс для роботи з анімаціями включає таймлайн, на якому користувачі можуть переглядати та редагувати свої кадри в режимі реального часу, що корисно для створення простих анімацій для ігор або веб-проектів.

Ще однією важливою функцією є можливість експорту в різні формати, включаючи PNG, GIF або як спрайт-листи для використання в ігрових проєктах. Крім того, Piskel дозволяє зберігати проєкти у форматі Piskel для подальшого редагування або в PNG з прозорістю для використання як спрайтів в іграх.

Однією з відмінних можливостей Piskel є підтримка публічних і приватних проєктів. Це дозволяє користувачам ділитися своїми роботами з іншими або працювати над приватними проєктами, якщо потрібно зберегти їх конфіденційність. Крім того, користувачі можуть зберігати свої роботи в хмарі або локально на пристрої, що додає гнучкості у використанні. Однак у недоліках можна зазначити, що Piskel обмежений у своїх можливостях порівняно з більш професійними інструментами, такими як Aseprite. Хоча він ідеально підходить для простих проєктів або швидких робіт, для великих чи складних проєктів він є недостатньо функціональним. Крім того, оскільки Piskel є веб-застосунком, для роботи потрібен доступ до інтернету, що може бути незручним у деяких ситуаціях.

Piskel – це чудовий інструмент для тих, хто шукає швидке і просте рішення для створення піксельної графіки та анімації та ідеально підходить для новачків та любителів, а також для тих, хто хоче створювати невеликі спрайти для ігор або веб-дизайну. Однак, для професійних художників або великих проєктів може бути потрібне більш функціональне програмне забезпечення (рис.1.7).

Рисунок 1.7 - Приклад піксельної картини, створеної в Piskel

1.3.3. Adobe Photoshop

Adobe Photoshop - це растровий графічний редактор, розроблений компанією Adobe Systems. Photoshop не найпоширеніший вибір для створення піксельного мистецтва, але його можна налаштувати для точного контролю пікселів, використовуючи шари та інструменти для редагування зображень.

Photoshop має кілька переваг, які роблять його потужним інструментом для створення піксельного мистецтва. По-перше, його основною перевагою є багатий набір інструментів для редагування зображень, який дозволяє створювати високоякісні графічні проєкти. Photoshop підтримує роботу з шарами, що надає велику гнучкість для створення складних піксельних картин або анімацій, таких як спрайти. Крім того, Photoshop є універсальним і може обробляти як низькоякісні піксельні зображення, так і великі файли з високою роздільною здатністю, що корисно, коли потрібно поєднувати піксельну графіку з іншими стилями. Це дозволяє художникам експериментувати та комбінувати різні типи зображень в одному проекті.

Однак у Photoshop також є і недоліки, особливо для тих, хто хоче працювати виключно з піксельним мистецтвом. Головний недолік полягає в

тому, що Photoshop не є спеціалізованим інструментом для піксельної графіки. Для виконання деяких операцій, таких як точне налаштування пікселів або збереження чіткості піксельних країв під час трансформацій, користувачеві потрібно уважно стежити за параметрами інструментів, такими як "Найближчий сусід", щоб уникнути розмитості або антиаліасингу. Це може бути складніше порівняно з простішими, спеціалізованими програмами для піксельного мистецтва, як-от Aseprite або Piskel, де названі налаштування вже інтегровані за замовчуванням.

Таким чином, Photoshop є потужним і універсальним інструментом для професійної роботи з графікою, але для створення чисто піксельного мистецтва інші редактори можуть бути більш інтуїтивними та зручними (рис.1.8).

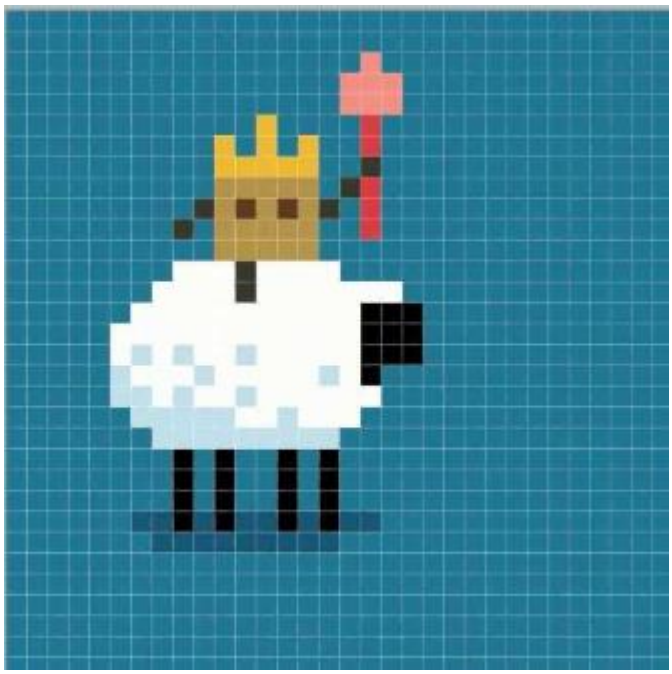

Рисунок 1.8 - Приклад піксельної картини, створеної в Adobe Photoshop

1.4 Мета та постановка завдання

Метою кваліфікаційної роботи є огляд інструментальних засобів та алгоритмічних рішень для піксельного мистецтва та розробка системи для створення моделей піксельних картин за допомогою мікроконтролера, модуля

Bluetooth, LED матриці та Android застосунку з використанням відповідних графічних засобів.

Основна ідея роботи – процес створення на мобільному пристрої цифрового зображення за допомогою растрового графічного редактора, в якому зображення редагуються на рівні пікселів (точок) з малою роздільною здатністю, що забезпечує їх чітку видимість.

Основнми задачами роботи є:

- аналіз предметної області та галузей використання засобів піксельної графіки;
- огляд графічних редакторів та алгоритмів для піксельного мистецтва;
- вибір мікроконтролерної платформи та компонентів для системи, призначеної для створення піксельних картин;
- розробка структури та функціоналу мікроконтролерної системи;
- вибір мов програмування та середовищ розробки системи;
- розробка програмного забезпечення Android-застосунку та опис процесу створення піксельної картини у мобільному пристрою;
- розробка програмної частини мікроконтролерної системи з описом обробки та передачі інформації з мікроконтролера на матрицю, яка забезпечує відображення картини.

2 АЛГОРИТМІЧНІ РІШЕННЯ ДЛЯ ПІКСЕЛЬНОГО МИСТЕЦТВА

У даному розділі розглянуто класичні алгоритми, на яких базується створення піксельного мистецтва, з акцентом на методах точного відтворення форм та заливки.

2.1 Алгоритм Брезенхенма для малювання ліній

Створення піксельних картин базується на декількох класичних алгоритмах, що включають малювання точок, ліній, геометричних фігур і заповнення областей, що буде розглянуто далі. Для реалізації алгоритмів існують спеціалізовані фреймворки та бібліотеки, які спрощують процес рендерингу.

Алгоритм Брезенхема - це один з основних алгоритмів комп'ютерної графіки, який використовується для малювання прямих ліній на піксельних екранах [9]. Головна задача, яку вирішує цей алгоритм, полягає в процесі перетворення ідеально прямої лінії з математичного погляду у відповідну послідовність пікселів на сітці дисплея. Важливо, що ця лінія повинна бути максимально схожа на справжню пряму, хоча вона складається з окремих пікселів, розташованих на дискретних позиціях (рис 2.1).

Для того, щоб краще зрозуміти необхідність алгоритму Брезенхема, потрібно згадати, що дисплей комп'ютера складається з сітки точок, де кожен піксель має свої цілі координати. Якщо спробувати просто намалювати лінію між двома точками, то відстань між ними не завжди буде ідеально кратною кроку сітки, тому завдання перетворюється на вибір таких пікселів, які найкраще апроксимують лінію.

Алгоритм Брезенхема вирізняється тим, що для малювання лінії використовує лише операції з цілими числами, уникаючи дорогих операцій з числами з плаваючою комою. Цей підхід залишається актуальним і сьогодні.

Даний спосіб дозволяє визначати, який піксель малювати на кожному кроці, таким чином, щоб лінія виглядала максимально гладкою і прямою.

Основна ідея алгоритму полягає в тому, що на кожному кроці виконується перевірка, наскільки лінія відхиляється від ідеальної прямої. Це відхилення коригується, і відповідний піксель на сітці обирається таким чином, щоб мінімізувати цю помилку. У процесі малювання лінії алгоритм поступово змінює координати пікселів, обираючи наступний піксель, який найближче розташований до математичної лінії.

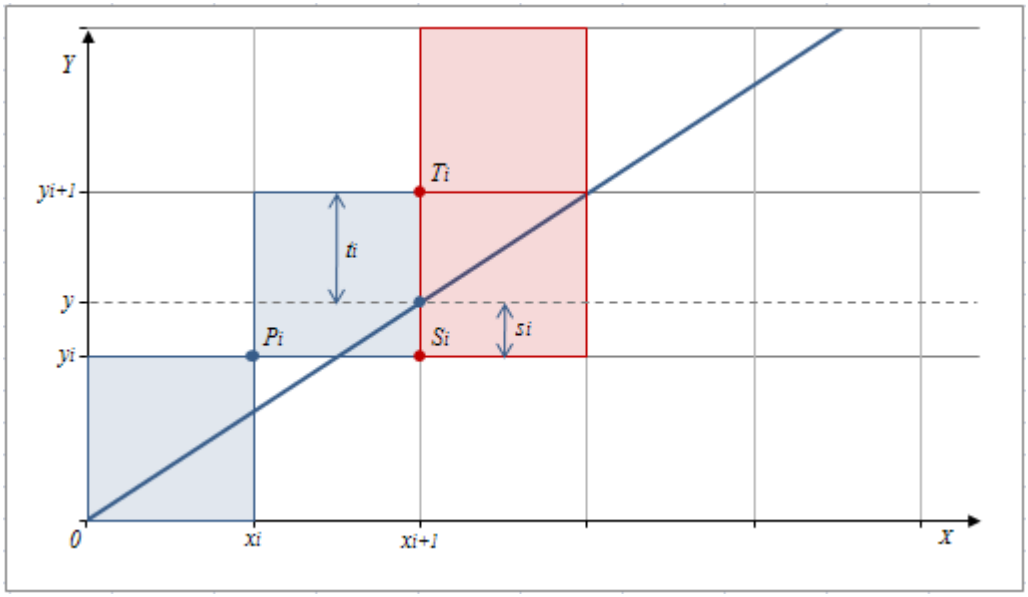

Рисунок 2.1 – Графічне представлення роботи алгоритму Брезенхема при растеризації відрізка прямої

Алгоритм працює наступним чином: на початку визначаються початкова і кінцева точки лінії (x_0y_0) і (x_1y_1) після визначається різниця між координатами кінцевої та початкової точок: $\Delta x=x_1-x_0$ і $\Delta y=y_1-y_0$.

Залежно від того, який із параметрів Δx чи Δy є більшим за абсолютним значенням, обирається основний напрямок малювання (по осі X або Y).

Алгоритм починається з ініціалізації поточної точки $(x,y)=(x_0y_0)$. Обчислюється початкове значення помилки D, яке визначає, наскільки

поточний піксель відхиляється від ідеальної лінії. Формула для початкового значення помилки:

$$D = 2\Delta y - \Delta x \quad (2.1)$$

Далі алгоритм поступово збільшує координату x (або y, залежно від напрямку), перевіряючи значення помилки D. Якщо D більше чи дорівнює нулю, значить, лінія "піднялася" над серединою між двома можливими пікселями, і треба збільшити y або x.

Формули оновлення помилки мають вигляд:

якщо $D \geq 0$, тоді $D=D+2(\Delta y-\Delta x)$ (2.2)

якщо $D<0$, тоді $D=D+2\Delta y$ (2.3)

Даний процес повторюється, доки не буде досягнута кінцева точка (x_1y_1). Алгоритм гарантує, що кожен вибраний піксель є найближчим до ідеальної лінії.

Головною перевагою цього алгоритму є його швидкість і ефективність. Завдяки використанню лише цілих чисел і простих арифметичних операцій, він значно швидший, ніж альтернативні методи, які залучають числа з плаваючою комою для обчислення позицій пікселів. Завдяки цьому алгоритм Брезенхема став основою для багатьох інших алгоритмів малювання графіки на дискретних екранах і використовується у багатьох програмах для створення піксельних зображень або векторних графічних редакторів.

Алгоритм Брезенхема також може бути адаптований для малювання не тільки прямих ліній, але й кіл, еліпсів та інших геометричних фігур, що робить його універсальним інструментом для роботи з комп'ютерною графікою

2.2 Алгоритм серединної точки для малювання кіл та еліпсів

Алгоритм серединної точки для малювання кіл та еліпсів є одним з фундаментальних методів у комп'ютерній графіці, який дозволяє ефективно відображати геометричні фігури у вигляді кіл та еліпсів на дискретних піксельних екранах. Він також відомий як алгоритм середньої точки Брезенхема для малювання кіл або еліпсів, оскільки є розширенням ідеї Брезенхема для малювання ліній. Цей алгоритм, як і алгоритм Брезенхема для ліній, використовує лише операції з цілими числами, що робить його швидким і придатним для малювання геометричних фігур на дискретних сітках.

Алгоритм серединної точки заснований на симетрії кіл і еліпсів. Коло можна повністю побудувати, малюючи лише одну його восьму частину, оскільки воно симетричне відносно обох осей та діагоналей. Малюючи лише одну октанту, можна легко відобразити її в інші сім частин за допомогою відповідних перетворень координат, що дозволяє значно скоротити кількість обчислень (рис 2.2.).

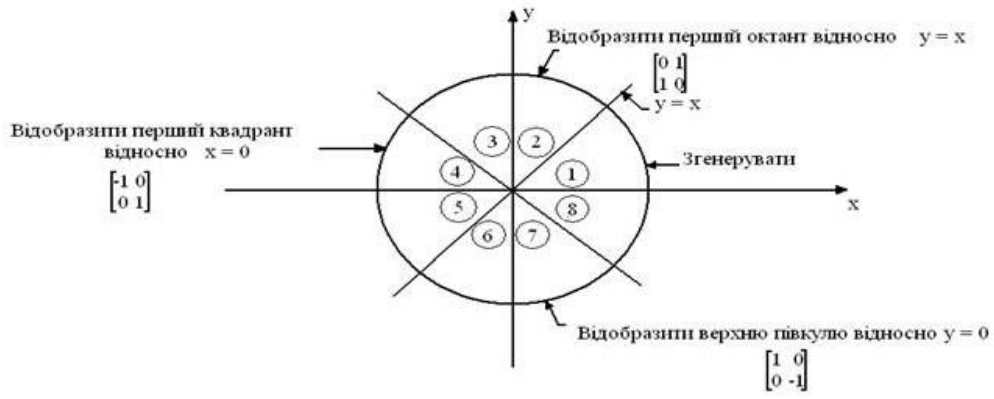

Рисунок 2.2 – Генерація повної окружності з дуги в першому октанті

Алгоритм середньої точки для малювання кола в першому квадранті з центром у точці (0,0) починається з початкової точки (0,r), де r – радіус кола (рис 2.3).

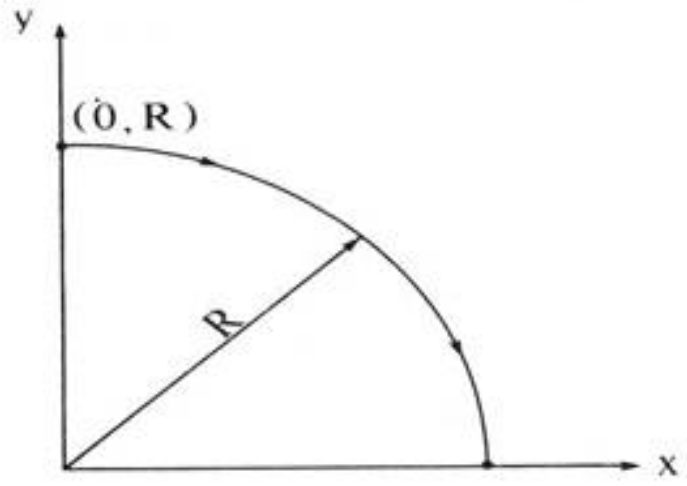

Рисунок 2.3 – Коло в першому квадранті

Під час побудови кола за годинниковою стрілкою, значення y монотонно зменшується, поки x збільшується в межах першого квадранта. Під час малювання кола необхідно вибирати найближчі пікселі, оскільки не можна безпосередньо відобразити неперервну дугу на растровому екрані.

Для вибору наступного пікселя використовується параметр рішення ppp, який визначає, чи вибирати піксель N(X+1,Y), чи піксель S(X+1,Y-1). Якщо значення p менше або дорівнює нулю, то вибирається N(X+1,Y), інакше вибирається точка S(X+1,Y-1).

Поперше, задаються вихідні параметри: радіус r та центр кола (X_C,Y_C). Початкове значення параметра рішення визначається формулою:

$$P_0=3 - 2r \tag{2.4}$$

Далі для кожної наступної позиції X_k, починаючи з k=0, проводиться обчислення параметра рішення P_k.

Якщо значення P_k менше нуля, вибирається точка X_{k+1},Y_k, і параметр P_k оновлюється за формулою:

$$P_{k+1} = P_k + 4X_k + 6 \quad (2.5)$$

Якщо значення P_k більше або дорівнює нулю, вибирається точка X_{k+1},Y_{k+1}, і параметр P_k змінюється за формулою:

$$P_{k+1} = P_k + 4(X_k - Y_k) + 10 \quad (2.6)$$

Після кожного обчислення координат, необхідно знайти симетричні точки для решти частин кола в інших октантах. Кожна точка (X,Y), яку вдалося отримати, переноситься на відповідну позицію кола, додаючи координати центра (X_C,Y_C), таким чином отримуються кінцеві координати X= X+ X_C і Y=Y+ Y_C .

Процес повторюється, поки не буде досягнуто завершення побудови кола. Алгоритм дозволяє точно малювати коло на растровому екрані, вибираючи найближчі пікселі для ідеальної дуги кола в першому квадранті та використовуючи симетрію для генерації решти частин.

Основна перевага алгоритму серединної точки - це використання цілих чисел для всіх обчислень, що робить його надзвичайно швидким і ефективним. Він уникає складних операцій з числами з плаваючою комою, що робить його ідеальним для застосування у графічних системах з обмеженими ресурсами, таких як сучасні вбудовані системи. Ще одна перевага алгоритму – це його простота в реалізації.

Завдяки використанню симетрії кола й еліпса, необхідно обчислювати лише частину пікселів, а потім просто відображати результати на інші частини. Це суттєво зменшує кількість необхідних обчислень.

2.3 Алгоритм заливки областей

Найчастіше в піксельному мистецтві використовують два основних підходи до заливки: заливка із затравкою та заливка меж. Однак є особливості, які роблять їх ще важливішими в контексті піксельної графіки.

Алгоритм заливки із затравкою, відомий також як Flood Fill, використовується для заповнення кольором області з однорідним кольором або для зміни кольору пікселів всередині певного замкнутого контуру. Принцип його роботи нагадує заливання фарбою на полотні: починаючи з певної точки (затравки), колір поступово поширюється на всі суміжні пікселі, поки не досягне межі або краю контуру (рис. 2.4).

Алгоритм починає роботу з початкової точки (затравки), яку користувач вказує всередині області, що потрібно залити. Після цього алгоритм перевіряє сусідні пікселі навколо цієї точки: якщо вони мають такий самий колір, як і затравка, вони замінюються на новий колір. Цей процес повторюється для кожного нового пікселя, на який "розливається" заливка, поки алгоритм не дійде до пікселів, що мають інший колір (наприклад, до межі фігури або до краю області). Цей підхід є ефективним для невеликих замкнутих областей і часто використовується в графічних редакторах для фарбування різних фігур.

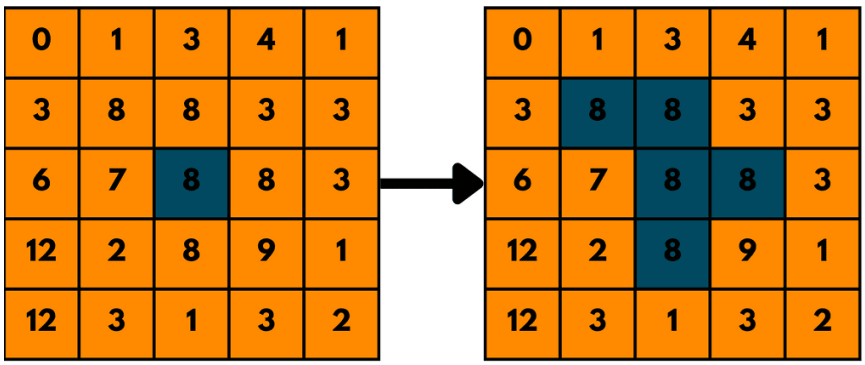

Рисунок 2.4 – Рекурсивний спосіб заповнення областей пікселів

Алгоритм заливки із затравкою може бути реалізований двома способами: рекурсивним й ітеративним. Рекурсивний метод простіший в реалізації, оскільки кожен виклик алгоритму здійснює пошук нових пікселів для заливки, занурюючись все глибше до країв області. Проте рекурсивний підхід може викликати проблеми зі стековою пам'яттю, якщо область для заливки дуже велика, тому іноді використовують ітеративний метод, який працює через використання черг або стеків для зберігання всіх пікселів, що потребують заливки.

Алгоритм заливки меж схожий на заливку із затравкою, але з однією відмінністю: замість заливки всіх пікселів одного кольору, він фокусується на заливці області до певної, заздалегідь визначеної, межі. Тобто алгоритм поширює заливку з точки-затравки доти, доки не зіткнеться з пікселем, який має інший, визначений як межа, колір. Цей метод ідеально підходить, коли необхідно залити область, оточену межами, які мають інший колір, наприклад, намальовану фігуру з контурами.

2.4 Алгоритм згладжування

Алгоритм згладжування, або антиаліасинг, в піксельному мистецтві виконує важливу функцію, хоч і має свої специфічні риси. Піксельне мистецтво відрізняється різкими й чіткими межами між пікселями, які формують малюнки на низькій роздільній здатності. Незважаючи на це, іноді художники використовують антиаліасинг для згладжування контурів і створення плавних переходів між кольорами, що допомагає зробити зображення більш реалістичним або м'яким, зберігаючи при цьому стилістичну чіткість піксельної графіки.

Антиаліасинг – це метод, що зменшує "зубчасті" краї або ефект сходинок, який виникає через низьку роздільну здатність [10]. У піксельному мистецтві цей ефект є нормальним, оскільки кожен піксель є важливим елементом, і зміна його кольору або форми може сильно вплинути на загальне сприйняття фігури.

Однак, у певних ситуаціях, коли потрібно зробити лінії або контури більш плавними, антиаліасинг стає корисним. Він дозволяє створювати ілюзію гладкості, додаючи проміжні пікселі зі змішаними кольорами.

У піксельному мистецтві антиаліасинг працює досить просто. Для контурів фігур, де є різкі переходи між кольорами, додаються пікселі з проміжними значеннями кольору. Наприклад, якщо є контур з чорного кольору на білому фоні, можуть додаватися сірі пікселі уздовж країв, щоб створити ілюзію плавного переходу. Це дозволяє контур виглядати менш "зубчастим" і більш м'яким. Замість того, щоб бачити чіткі "сходинки", око сприймає плавний перехід між кольорами.

Ручний антиаліасинг у піксельному мистецтві часто є найпоширенішим методом. Художник може вручну додавати потрібні пікселі для згладжування контурів або деталей. Такий підхід дає максимальний контроль над кожним пікселем, що особливо важливо для збереження чіткого стилю піксельного мистецтва. Художники можуть вирішувати, де саме необхідно згладити контури, не порушуючи при цьому характерного вигляду піксельної графіки.

Автоматичний антиаліасинг іноді використовується у програмах для малювання, які можуть автоматично згладжувати лінії під час створення зображення. Проте цей підхід менш точний і не завжди підходить для піксельного мистецтва, оскільки програма може додати надто багато згладжувальних пікселів, змінюючи стиль зображення. Відтак, хоча автоматичний антиаліасинг може спрощувати процес, художникам часто доводиться вручну коригувати результат для досягнення кращої точності та збереження оригінального стилю.

У певних випадках використовується й субпіксельний антиаліасинг, коли кольори пікселів змінюються залежно від їх розташування щодо сусідніх пікселів. Цей метод створює плавні переходи між кольорами на основі субпіксельних значень. В піксельному мистецтві субпіксельний антиаліасинг використовується рідко, оскільки тут переважає різкість та чіткість. Проте він

може бути корисним для створення м'яких тіней або плавних переходів світла на об'єктах.

Хоча антиаліасинг у піксельному мистецтві дозволяє зробити зображення плавнішим, важливо зберегти баланс між згладжуванням і різкістю. Зайвий антиаліасинг може зруйнувати характерну чіткість піксельного стилю, де кожен піксель має вагоме значення. Тому, навіть при використанні згладжування, важливо не порушувати основні елементи форми та деталей малюнка.

3 СТРУКТУРНА ТА ФУНКЦІОНАЛЬНА СХЕМИ СИСТЕМИ ДЛЯ СТВОРЕННЯ МОДЕЛЕЙ ПІКСЕЛЬНИХ КАРТИН

У розділі представлено опис та характеристики компонентів системи для створення моделей піксельних картин, а також розроблено структурну та функціональну схему запропонованої системи.

3.1 Компоненти системи

При розробці системи для створення моделей піксельних картин було проаналізовано та обрано основні компоненти, які становлять цю систему та розглядаються нижче.

1. Мікроконтролер Arduino UNO.
2. Bluetooth модуль HC-06.
3. Світлодіодна матриця 16x16.
4. Android смартфон із застосунком.

3.1.1 Мікроконтролер Arduino UNO

Arduino UNO є популярною платою мікроконтролера з відкритим кодом, що базується на мікроконтролері ATmega328P. яка використовується для створення електронних пристроїв, проектів Інтернету речей (IoT) та інтерактивних об'єктів. Arduino UNO програмується за допомогою мови програмування Arduino, яка ґрунтується на спрощеному C/C++. Мова програмування Arduino є простою у вивченні та використовує багато відомих понять, таких як змінні, цикли та умовні оператори.

Зовнішній вигляд та опис пінів Arduino UNO приведено на рис. 3.1.

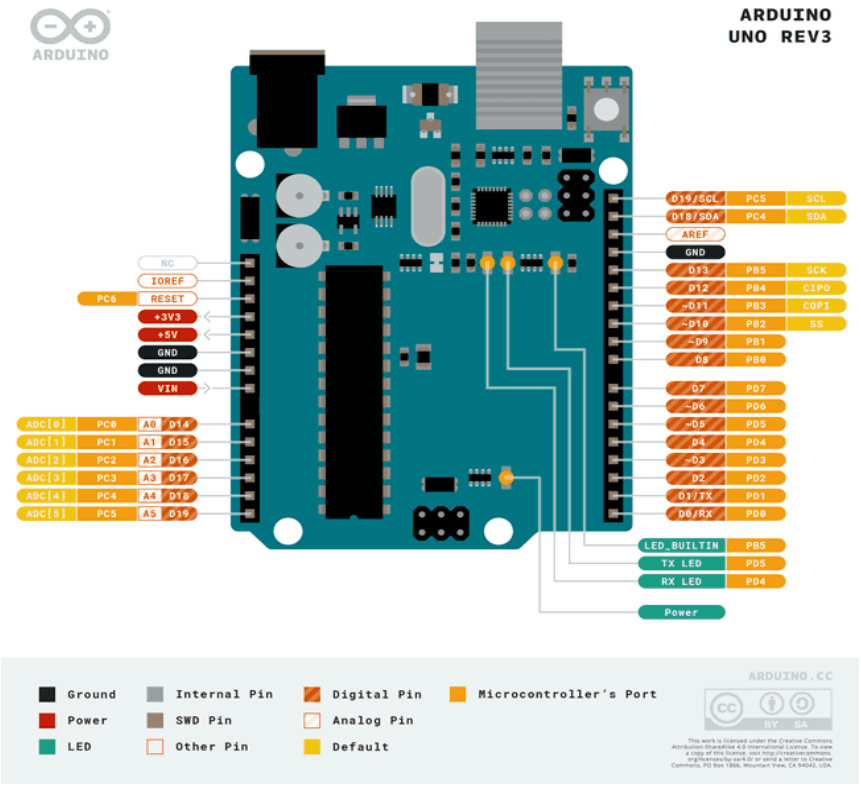

Рисунок 3.1 – Зовнішній вигляд та опис пінів Arduino UNO

Технічні характеристики мікроконтролера Arduino UNO наведено у таблиці 3.1.

Таблиця 3.1 – Технічні характеристики мікроконтролера Arduino Uno

Мікроконтролер	ATmega328
Робоча напруга	5В
Напруга живлення (рекомендоване)	7-12В
Напруга живлення (максимальне)	6-20В
Цифрові входи/виходи	14 (6 з них можуть використовуватись як ШІМ-виходів)
Аналогові входи	6
Максимальний струм одного виходу	40 мА
Максимальний вихідний струм виходу 3.3V	50 мА

Flash-пам’ять	32 КБ (ATmega328) з котрих 0.5 КБ використовується загрузчиком
SRAM	2 КБ
EEPROM	1 КБ
Тактова частота	16 МГц

3.1.2 Bluetooth модуль HC-06

HC-06 – це bluetooth модуль широкого застосування для з’єднання пристроїв через Bluetooth-з’єднання (рис 3.2.). Bluetooth модуль керується за допомогою UART, тобто, по суті, UART-to-Bluetooth є перетворювачем.

На відміну від модуля HC-06 може працювати як у режимі Master так і у Slave. Модуль Сумісний з контролерами Arduino / 51 / AVR / PIC / ARM / MSP430 і т.д.), плата також може бути підключена безпосередньо до 5V контролера.

Налаштування параметрів модуля проводиться за допомогою AT команд. Вхід в режим команд AT - по кнопці на платі. Дальність - до 10м [11].

Рисунок 3.2 – Зовнішній вигляд та опис пінів HC-06

Технічні характеристики bluetooth модуля HC-06 наведено у таблиці 3.2.

Таблиця 3.2 – Технічні характеристики HC-06

Апаратна підтримка	Bluetooth Specification v2.0 +RDR
Тип модуляції	GFSK (Gaussian Frequency Shift Keying)
Потужність відправки	≤ 4dBm, Class 2
Потужність прийому	≤ -84dBm at 0.1% BER
Швидкість	асинхронна 2.1Mbps(Max)/160 kbps синхронна 1Mbps/1Mbps
Безпека	Автентифікація та шифрування
Профіль	Bluetooth serial port
Живлення	+5V DC 50mA
Робочі температури	-20 ~ +75 C
Розміри	26.9мм x 13мм x 2.2мм
Частота	2,4 GHz
Напруга	3,6…6V
Струм	30…40mA

3.1.3 Світлодіодна матриця

Матриця світлодіодна 16x16 WS2812B RGB 256 світлодіодів (рис 3.3). Кожен піксель із трьох кольорів може змінювати яскравість у 256 значеннях й отримати 16777216 кольорів, модулі можуть об'єднуватися для передавання сигналів по одній лінії.

Інформація у світлодіодний модуль надходить послідовним каналом. Для керування можна використовувати спеціальні контролери, які можуть бути програмованими, світловузькими, автономними або підчиненими. Дана модель матриці працює з Arduino, Raspberry Pi, SP103E, SP105E, K1000C, T1000S та іншими контролерами.

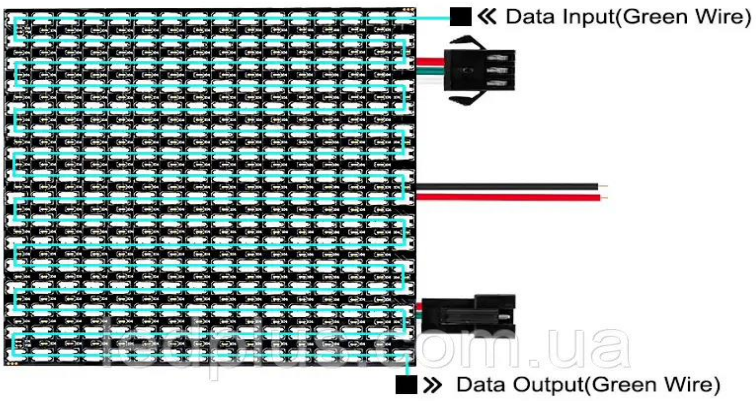

Рисунок 3.3 – Зовнішній вигляд світлодіодної матриці

Технічні характеристики світлодіодної матриці наведено у таблиці 3.3.

Таблиця 3.3 – Технічні характеристики світлодіодної матриці

Робоча напруга	5В DC
Споживана потужність	0.3 Вт/LED
Кількість світлодіодів	16 х 16 = 256 світлодіодів
Тип світлодіодів	WS2812B SMD5050
Потужність	75 Вт
Випромінюванні кольори	повноколірний (24 біти) RGB
Кут світіння	120°
Ресурс	до 50,000 годин
Пластина	м’яка мідь, підкладка чорного кольору
Габарити	160 х 160 х 3 мм

3.1.4 Компоненти системи управління-матрицею

Android застосунок потрібен для керування LED-матрицею за допомогою мікроконтролера та Bluetooth модуля. Застосунок має матрицю 16х16 для малювання, можливість підключення до Bluetooth модулю та передачі даних про матрицю на мікроконтролер.

3.1.5 Матеріали для фізичної піксельної картини

Серед необхідного матеріалу є решітка, матовий пластик, тоніровка та скло. Грати потрібні для того, щоб ізолювати кожен діод таким чином, щоб він давав світло в межах свого квадрату. Пластик, тоніровка та скло потрібні для розсіювального ефекту для забезпечення світла діоду у піксельному вигляді (рис.3.4).

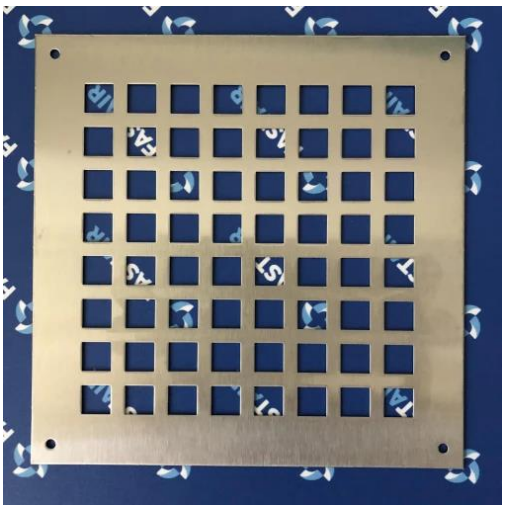

Рисунок 3.4 – Приклад решітки

Приклад моделі піксельної картини приведений на рис. 3.5.

Рисунок 3.5 – Приклад піксельної картини

3.2 Структурна та функціональна схеми з’єднання компонентів системи

У ході розробки моделі піксельної картини було розроблено структурну та функціональну схеми з’єднання компонентів системи.

Структурна схема демонструє способи зв’язку між компонентами, а функціональна демонструє з’єднання на фізичному рівні (рис.3.6).

З’єднання світлодіодної матриці, Arduino UNO та Bluetooth модуля відбувається за допомогою дротів, а мобільний застосунок під’єднується завдяки bluetooth.

В системі користувач створює піксельні картини у мобільному Android-застосунку/ Далі дані передаються на Arduino через Bluetooth модуль, а Arduino, у свою чергу, керує світлодіодною матрицею, відображаючи у реальному часі результат малюнку на LED-матриці, що створений раніше у мобільному застосунку.

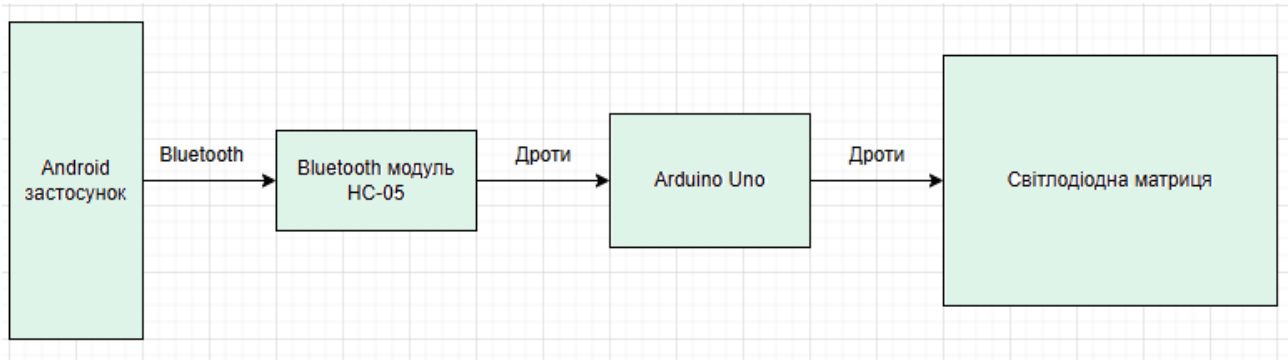

Рисунок 3.6 – Структурна схема розробленої системи

На рис. 3.7 зображено функціональну схему розробленої системи, а на рис. 3.8 зображено реалізацію системи.

Вся схема живиться через павербанк. Від штекера живлення VCC та GND підключаються до макетної плати. VCC та GND матриці підключається макетної плати відповідно до того як підключений штекер живлення. Вивід

DIN підключається до аналогового виходу контролера через резистор 300 Ом. Для підключення Bluetooth модуля HC-06 потрібно чотири дроти. VCC та GND Bluetooth модуля підключається до макетної плати відповідно того як підключений штекер живлення, RX підключається до TX, а TX до RX на контролері. Вивід GND Arduino підключено до макетної плати, а VCC з макетної плати підключена до VIN. З'єднання Android застосунку до контролеру відбувається за допомогою радіочастоти.

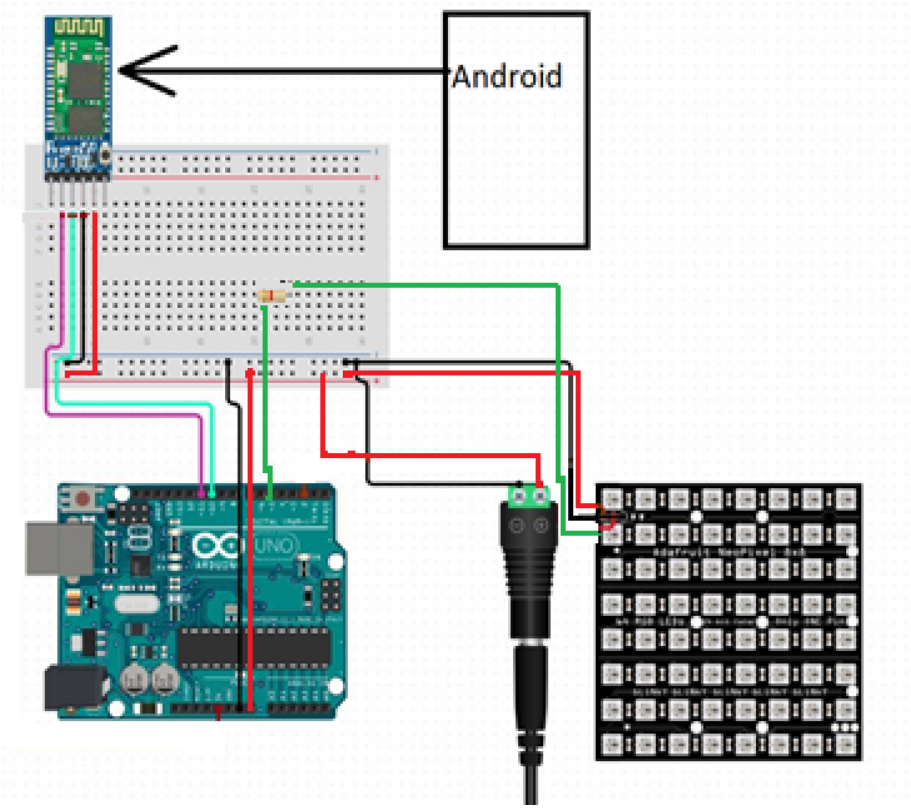

Рисунок 3.7 – Функціональна схема системи

На рисунку 3.8 приведено реалізацію розробленої системи для створення піксельних картин.

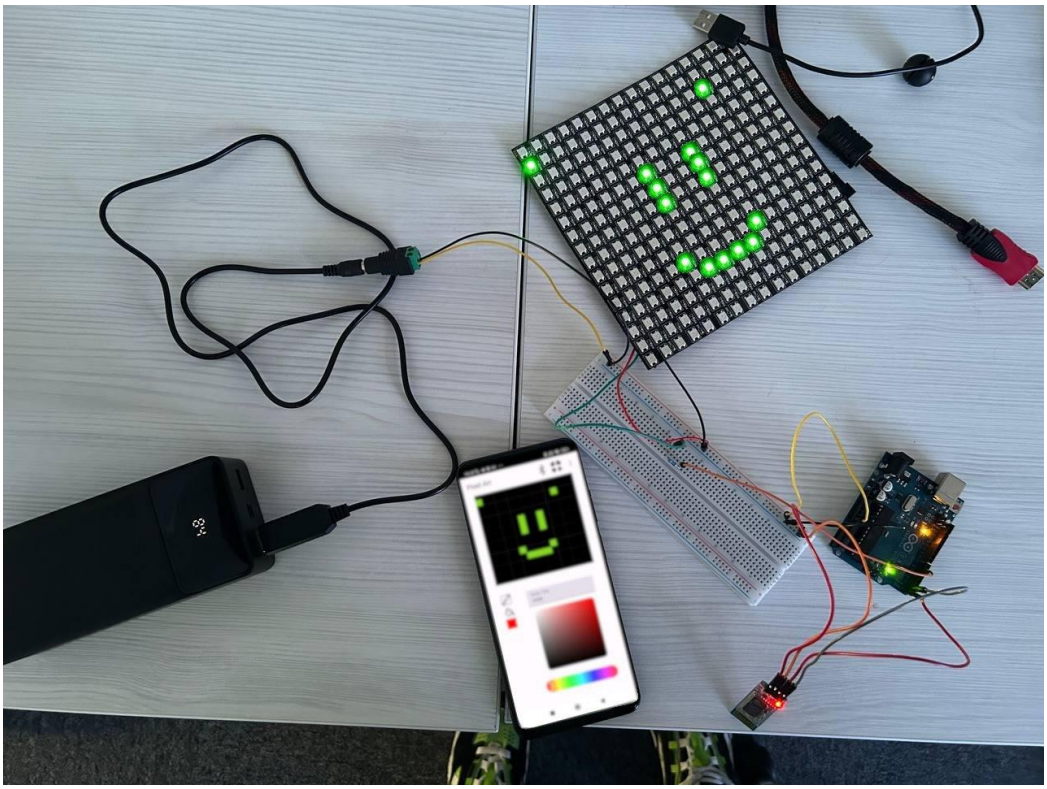

Рисунок 3.8 – Реалізація розробленої системи

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ СТВОРЕННЯ МОДЕЛЕЙ ПІКСЕЛЬНИХ КАРТИН

У розділі розглядаються деталі реалізації апаратно-програмної частини системи для створення піксельних картин. Основною метою є забезпечення інтеграції між мобільним застосунком на базі Android, модулем Arduino, Bluetooth-модулем HC-06 та світлодіодною матрицею, яка виступає візуальним засобом відображення створених зображень.

4.1 Android застосунок

Вибір апаратних компонентів та розробка програмного забезпечення обумовлені необхідністю забезпечити зручність для користувача, стабільність роботи системи та високу точність передачі даних. У системі передбачається, що користувач створює піксельні картини у мобільному застосунку, після чого дані передаються на Arduino через Bluetooth, а Arduino, у свою чергу, керує світлодіодною матрицею, відображаючи результат у реальному часі.

Android-застосунок є ключовим елементом у системі, що забезпечує створення піксельних картин користувачем. Він написаний мовою Kotlin та реалізований на основі сучасних технологій та підходів до розробки, зокрема Jetpack Compose, архітектурного патерну MVVM (Model-View-ViewModel) та принципів Clean Architecture. Це дозволяє забезпечити гнучкість, підтримуваність та тестованість коду.

4.1.1 Середовище розробки

Для створення застосунку використовувалося середовище розробки Android Studio, яке є офіційним IDE для розробки Android-застосунків. Це потужний інструмент, який надає всі необхідні функції для ефективної роботи з

Android SDK, включаючи підтримку таких мов програмування, як Kotlin і Java. Android Studio дозволяє зручно писати код, тестувати застосунки та налагоджувати їх на різних етапах розробки.

Основною перевагою Android Studio є тісна інтеграція з платформою Android. Застосунок підтримує Kotlin і Java як основні мови програмування для Android, а також включає все необхідне для роботи з Android SDK (Software Development Kit). Це забезпечує зручне налаштування середовища для кожного етапу розробки, від створення інтерфейсу користувача до інтеграції з базами даних або зовнішніми сервісами.

Android Studio має потужний редактор коду з підтримкою автодоповнення, підсвічування синтаксису та рефакторингу, що дозволяє розробникам швидко і безпечно працювати з кодом.

Вбудовані інструменти для налагодження дозволяють проводити детальну перевірку помилок у реальному часі, а також моніторити виконання застосунку через профайлери, щоб оцінити ефективність і використання ресурсів.

Android Studio також має Android Emulator, що дозволяє тестувати застосунки на різних пристроях та версіях Android без необхідності фізично підключати реальні гаджети.

Завдяки глибокій інтеграції з Google Play, Android Studio також підтримує легку публікацію застосунків, допомагаючи розробникам тестувати їх на різних пристроях і версіях Android. Крім того, Android Studio підтримує інтеграцію з системами контролю версій, такими як Git, що полегшує командну розробку.

Загалом, Android Studio є комплексним і потужним середовищем для розробки Android-застосунків, яке охоплює всі аспекти розробки, тестування, налагодження та публікації, роблячи процес створення мобільних додатків значно більш ефективним і зручним.

4.1.2 Архітектура застосунку

Архітектура застосунку зроблена за моделлю MVVM (рис. 4.1) та побудована на принципах Clean Architecture (рис. 4.2), що забезпечує чітке розділення відповідальностей між різними частинами коду (рис. 4.3).

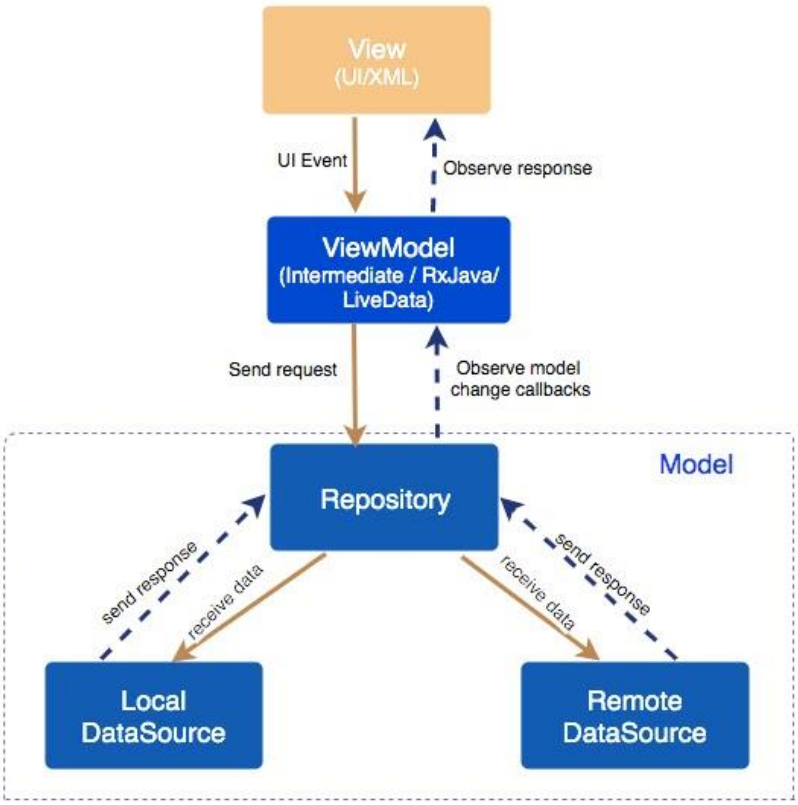

Рисунок 4.1 – Абстрактна схема моделі MVVM

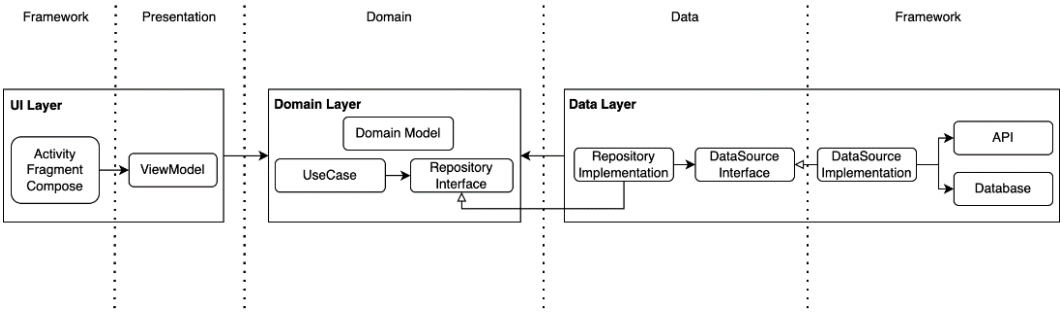

Рисунок 4.2 – Абстрактна схема Clean Architecture

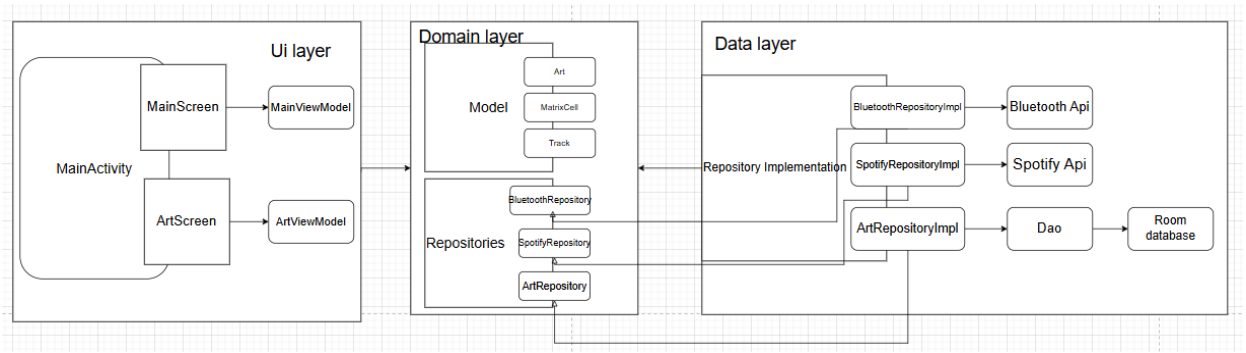

Рисунок 4.3 – Схема застосунку

Вибір архітектурного патерну MVVM, порівняно з іншими патернами, наприклад MVC або MVP, зумовлений його унікальними перевагами для Android-розробки. MVVM забезпечує чітке розділення між логікою представлення (View) та бізнес-логікою (ViewModel), використовуючи модель даних (Model) як сполучний елемент. У Jetpack Compose MVVM ідеально поєднується з декларативним підходом до створення інтерфейсів, оскільки ViewModel дозволяє легко управляти станом інтерфейсу користувача через потоки даних (Flow). Це усуває необхідність прямої взаємодії між View та Model, зменшує залежності та сприяє зручному тестуванню компонентів [12].

Інші патерни, такі як MVC, часто спричиняють перенавантаження View логікою або надмірну складність у масштабних проєктах. MVP частково вирішує цю проблему, але додає зайву складність у комунікацію між Presenter і View, що ускладнює підтримку. Натомість MVVM дозволяє уникнути цих недоліків завдяки використанню спостережуваних потоків даних, що автоматично оновлюють інтерфейс користувача за змін у ViewModel.

Принципи Clean Architecture доповнюють MVVM, забезпечуючи чітке розділення логіки за рівнями:

- верхній рівень (Presentation) відповідає за інтерфейс користувача та взаємодію з користувачем;
- середній рівень (Domain) містить бізнес-логіку, незалежну від зовнішніх залежностей;
- нижній рівень (Data) обробляє доступ до даних (API, бази даних) і

реалізує їхню інкапсуляцію.

Таке розділення забезпечує незалежність від фреймворків, легкість у внесенні змін, масштабованість та повторне використання компонентів у різних частинах застосунку. Крім того, Clean Architecture дозволяє спростити тестування: завдяки ізоляції логіки на різних рівнях можна тестувати кожен рівень окремо, не впливаючи на інші.

Таким чином, застосування MVVM разом із Clean Architecture дозволяє створити код, який легко підтримувати, розширювати й тестувати, що є критично важливим для довготривалих проєктів. Ці підходи сприяють реалізації масштабованих і стійких рішень, що відповідають сучасним стандартам розробки програмного забезпечення.

У Presentation-шарі використовується Jetpack Compose для побудови інтерфейсу користувача. Цей шар взаємодіє з ViewModel, яка відповідає за керування станом та логікою представлення. Інтерфейс повністю реактивний, тому будь-які зміни стану даних у ViewModel одразу відображаються в UI через механізми Kotlin Flow. Завдяки Compose інтерфейс забезпечує високу швидкість рендерингу та адаптивність навіть для складних операцій, таких як малювання на матриці або редагування імпортованих зображень.

Domain-шар є центральною частиною архітектури, яка інкапсулює всю бізнес-логіку. Тут реалізовані репорзиторії, що відповідають за виконання основних операцій застосунку, таких як малювання на матриці, завантаження зображень із файлової системи, інтеграція зі Spotify API та передача даних через Bluetooth. Domain-шар є незалежним від інших компонентів, що дозволяє легко тестувати його функціональність.

Data-шар відповідає за доступ до джерел даних. У ньому реалізована імплементація репозиторіїв, які взаємодіють із локальними джерелами, наприклад, файловою системою або базою даних та віддаленими API (Spotify API). Імплементація репозиторіїв передає чисті моделі даних до Domain-шару, що спрощує управління даними та підвищує надійність.

Окремо слід виділити Bluetooth Manager, який працює як окремий компонент у Data-шарі. Він відповідає за встановлення та підтримку з’єднання з Bluetooth модулем HC-06, передачу оброблених даних до апаратного рівня, а також обробку помилок зв’язку.

Комунікація між шарами здійснюється через чітко визначені інтерфейси, що відповідає принципам інверсії залежностей. Presentation-шар взаємодіє лише з Domain, а Domain-шар – із Data через абстракції. Завдяки цьому зміни в одному шарі мінімально впливають на інші.

Архітектура застосунку забезпечує баланс між складністю коду та його підтримуваністю. Завдяки Clean Architecture легко додавати нові функції, адаптувати існуючі модулі або інтегрувати нові технології. Крім того, дотримання архітектурних принципів забезпечує високий рівень тестованості всіх компонентів, що є важливим для стабільності та надійності роботи застосунку.

4.1.3 Функціональність застосунку

Інтерфейс застосунку складається з двох екранів та діалогового вікна. Один з екранів Main Screen має три різні варіації, котрі залежать від режиму вибраного з діалогового вікна. Застосунок має такі моди як “Малювання”, “Фото” та “Spotify”. Основний екран складається з Toolbar (рис. 4.4), (додаток Б, лістинг Б.1), матриці 16x16 для малювання (рис. 4.5), (додаток Б, лістинг Б.2) та SettingsView, що відрізняється від кожного моду. Toolbar головного екрану складається з трьох кнопок: Bluetooth, All Modes та More Actions Bluetooth кнопка відповідає за підключення та відключення Android пристрою до Arduino.

Рисунок 4.4 – Вигляд Toolbar

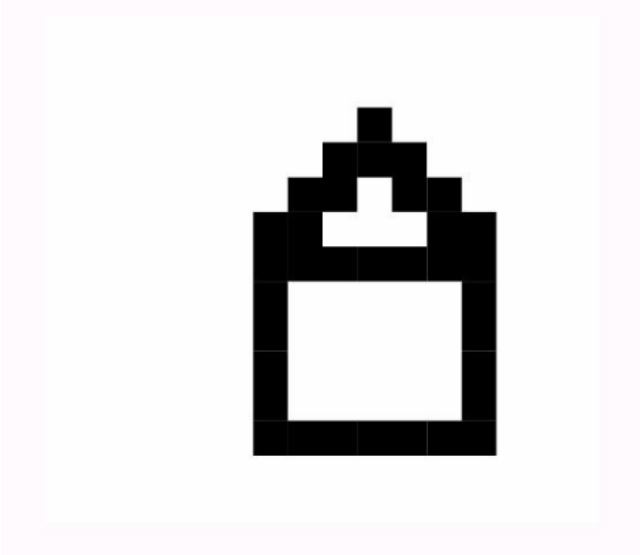

Рисунок 4.5 – Вигляд Матриці

В залежності від стану підключення Bluetooth змінюється ікона відповідної кнопки. Кнопка All Modes відповідає за відкриття діалогового вікна, де користувачу надається можливість обрати вищеперераховані моди (рис 4.6), (додаток Б, лістинг Б.3).

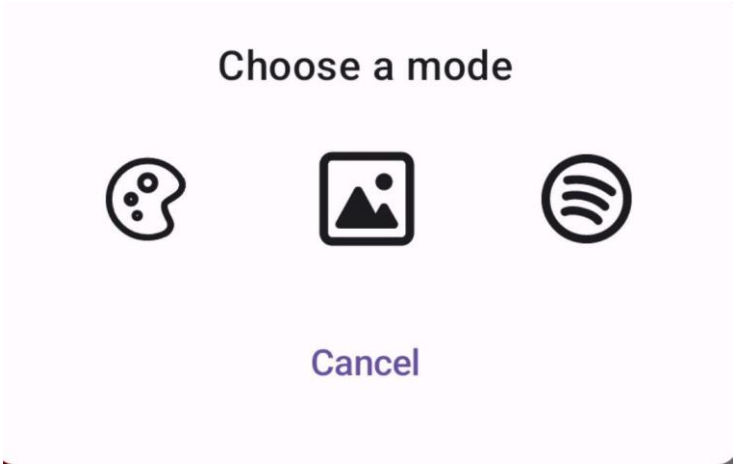

Рисунок 4.6 – Вигляд діалогу

Кнопка More Actions дає користувачу додаткові функції такі як “Почистити картину”, “Зберегти картину” та “Загрузити картину”. Дані функції відповідають своїм назвам “Почистити картину” – чистить картину, “Зберегти картину” – зберігає свою картину в базу даних , “Загрузити картину” – відкриває другий екран (Art Gallery Screen), де знаходяться всі збережені картини. База даних реалізована за допомогою фреймворку від Google – Room.

У режимі “Малювання” SettingsView має ColorPicker (додаток Б, лістинг Б.4) та кнопки для режиму малювання та заливки (рис 4.7).

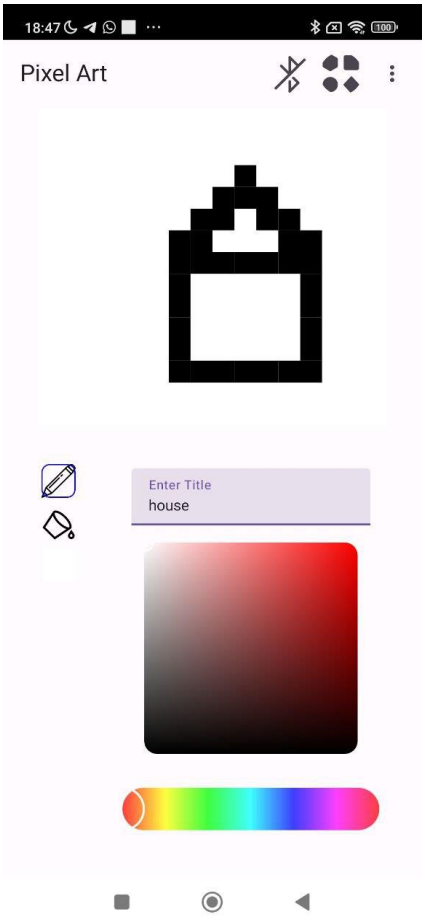

Рисунок 4.7 – Режим “Малювання”

Завдяки Jetpack Compose, використовуючи компонент Canvas, можна легко малювати, вибираючи кольори та редагуючи пікселі. Canvas надає зручний API для створення складних графічних елементів, таких як форми, лінії, кола та піксельні картини, даючи змогу гнучко змінювати кольори. Це

робить малювання зручним і ефективним навіть для складних візуальних завдань. Стан матриці синхронізується з ViewModel через реактивні механізми Kotlin Flow, що забезпечує плавну роботу інтерфейсу.

У режимі “Фото” SettingsView має лише кнопку “Вибрати фото”, яка дає користувачу можливість завантаження зображень із файлової системи (додаток Б, лістинг Б.5).

Завдяки бібліотеці Coil забезпечується швидке відображення та завантаження зображень файлової системи, а за допомогою бібліотеки ImageCropper користувач може обрізати фото до формату 16x16 (рис. 4.8). Це дозволяє зручно адаптувати будь-яку світлину до розміру матриці.

Після завантаження та обрізання, фото трансформується у піксельний формат й відображається на матриці (рис. 4.9).

Рисунок 4.8 – Процес обрізання фото

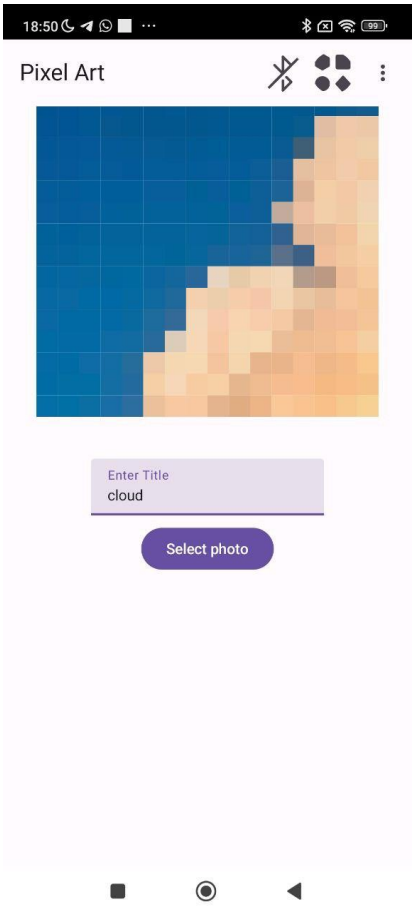

Рисунок 4.9 – Режим “Фото”

У режимі “Spotify” SettingsView дуже схожа на режим “Фото” єдина різниця в тексті кнопки. Кнопка має назву “Отримати постер пісні” (додаток Б, лістинг Б.6).

В застосунку реалізована інтеграція зі Spotify API, що дає змогу завантажувати обкладинки треків із музичного сервісу. Застосунок автоматично отримує фото та обробляє його, масштабуючи зображення до необхідного розміру, та відображує на матриці.

Перемикання пісень автоматично змінює постер. Це дає можливість відображати улюблені треки на піксельній картині (рис 4.10).

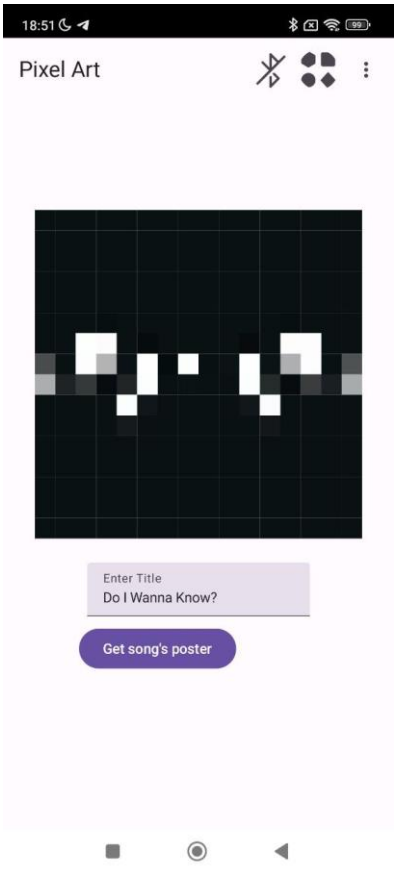

Рисунок 4.10 – Режим “Spotify”

Перейдемо до другого екрану Art Gallery Screen (додаток Б, лістинг Б.7). На екрані присутній прокручуваний список, який відображає збережені картини з бази даних

При натисканні на елемент списку відкривається можливість видалити цей елемент зі списку та бази даних (рис. 4.11).

Також застосунок вже має за замовчуванням деякі картини, картини сортуються за часом, коли були додані. Це дозволяє користувачам легко керувати своїм творчим доробком, створювати бібліотеку робіт і повертатися до них у будь-який час.

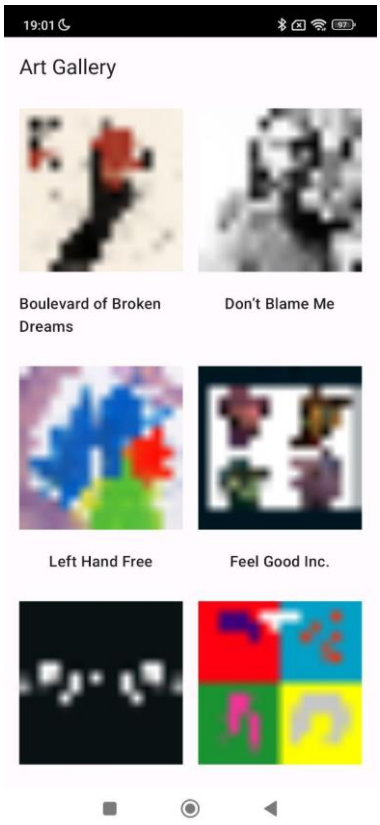

Рисунок 4.11– Режим “Spotify”

4.1.4 Технічна складова застосунку

Застосунок використовує Bluetooth для підключення до HC-06, яка є Bluetooth-модулем, призначеним для передачі даних між бездротовим пристроєм і Arduino.

Система працює наступним чином. Застосунок взаємодіє з Bluetooth-апаратним забезпеченням на смартфоні через Android Bluetooth API. HC-06 підтримує класичний Bluetooth (не BLE), тому процес підключення передбачає сканування доступних пристроїв. Android-пристрій знаходить HC-06, ідентифікуючи її за унікальною MAC-адресою. Основним компонентом є клас BluetoothAdapter, який дозволяє управляти Bluetooth-модулем на телефоні. За його допомогою можна вмикати або вимикати Bluetooth, а також починати процес сканування доступних пристроїв у межах радіусу дії [13].

Після процесу сканування відбувається спроба встановлення парного підключення (pairing). Коли пристрої стають «парою», між ними встановлюється стійке з’єднання (додаток Б, лістинг Б.8).

Коли з’єднання активне, застосунок за допомогою BluetoothSocket відкриває Bluetooth-сокет і створює канал для обміну даними. HC-06 працює як бездротовий міст, перетворюючи сигнали Bluetooth у послідовні UART-команди, які надсилаються на Arduino. Через BluetoothSocket відкриваються два потоки: вхідний і вихідний. Вхідний потік (getInputStream) дозволяє приймати дані, які надсилає HC-06, наприклад, з Arduino. Вихідний потік (getOutputStream) використовується для передачі даних з Android на HC-06. Arduino може читати ці команди через свої послідовні пін-контакти (TX і RX), виконувати відповідні дії (рис 4.12) (додаток Б, лістинг Б.9).

Рисунок 4.12 – Android в пошуку Bluetooth зв’язку

Для режиму “Фото” витягується світлина з файлової системи Android і обробляється за допомогою бібліотек Coil та ImageCropper.

Розглянемо бібліотеку Coil. Coil - це сучасна та зручна бібліотеки для завантаження фотографій, яка без проблем інтегрується із Jetpack Compose (рис.4.13). При завантаженні фото, отримується його URI.

ImageCropper бібліотека дозволяє обрізати фотографії з високою точністю, використовуючи як інтерактивний інтерфейс, так і програмний підхід. ImageCropper приймає вихідне зображення, визначає область обрізки і повертає зображення у новому форматі або розмірі. В нашому випадку обрізка

використана для фокусування на окремій ділянці фотографії у вигляді квадрату (додаток Б, лістинг Б.10).

Спочатку використовується бібліотека Coil для завантаження зображення з файлової системи. Після того як зображення завантажено, URI світлини передається до ImageCropper для обрізки. Після того як обрізка завершена, застосунок отримує URI, котре далі трансформується в бітмапу розміром 16x16, а згодом і з’являється на матриці.

Рисунок 4.13 – Бібліотека Coil

Застосунок інтегрує Spotify API, зокрема його SDK, для управління треками та отримання їхніх метаданих, забезпечуючи взаємодію з музичним плеєром Spotify. Основна інтеграція реалізована через Spotify App Remote SDK, який дозволяє не лише управляти відтворенням треків, але й отримувати актуальну інформацію про стан плеєра.

Першим важливим етапом є автентифікація користувача та встановлення з'єднання зі Spotify за допомогою класу SpotifyAppRemote (додаток Б, лістинг Б.11). Після вдалої автентифікації застосунок отримує можливість доступу до функціональності плеєра. Однією з ключових функцій є метод subscribeToPlayerState(), який дозволяє підписатися на оновлення стану плеєра. Завдяки цьому застосунок отримує дані про треки, які відтворюються, у режимі реального часу.

Коли з'єднання успішно встановлено, Spotify API надсилає інформацію про поточний трек через callback. Отримані дані містять назву треку, ім'я

виконавця, а також URI обкладинки треку. URI має формат spotify:image:<image_id> і вимагає перетворення в URL-адресу для завантаження зображення обкладинки з сервера Spotify CDN. Для цього формується повна URL-адреса, поєднуючи базову адресу Spotify CDN з ID зображення (додаток Б, лістинг Б.12).

Особливістю цієї інтеграції є використання міжпроцесної взаємодії (IPC) між створеним застосунком і застосунком Spotify (рис 4.14). Встановлення з'єднання, обробка запитів до плеєра та отримання метаданих здійснюються через IPC, оскільки відтворення музики контролюється безпосередньо застосунком Spotify. Застосунок, по суті, виступає клієнтом, який передає команди до Spotify через Spotify App Remote SDK і отримує відповіді у зворотньому напрямку. Це дозволяє уникати дублювання логіки плеєра у аплікації, але вимагає коректної обробки взаємодії між процесами.

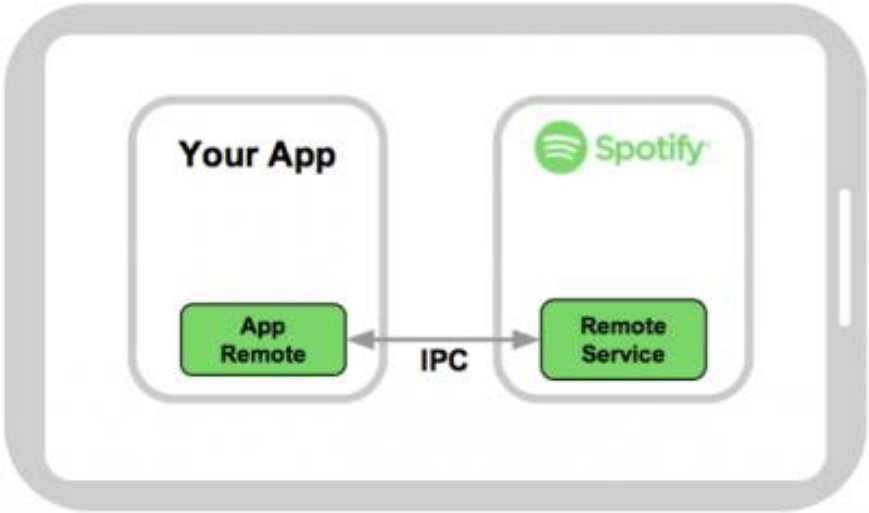

Рисунок 4.14 – Схема роботи Spotify API

Функції малювання та заливки реалізовані за допомогою Jetpack Compose, що забезпечує інтуїтивний і плавний досвід малювання.

Jetpack Compose – це новий підхід до створення інтерфейсів користувача в Android, який перевершує традиційний підхід завдяки своїй декларативності, простоті та гнучкості. На відміну від класичного підходу, де розробник повинен

був працювати з XML для опису інтерфейсу та Kotlin або Java для логіки, Compose дозволяє описувати інтерфейс безпосередньо в Kotlin-коді. Це забезпечує більш інтуїтивний процес розробки, позбавляючи необхідності синхронізувати код між XML та програмною частиною [14].

Однією з ключових переваг Jetpack Compose є декларативний підхід. У Compose розробник не описує послідовність дій для зміни стану інтерфейсу, а натомість лише визначає, як UI виглядає в конкретному стані. Це значно спрощує роботу: якщо стан змінюється, Compose автоматично оновлює відповідні елементи інтерфейсу без втручання розробника. Такий підхід особливо зручний для інтерактивних застосунків, де стан UI змінюється динамічно.

Окрім цього, Jetpack Compose дозволяє легко працювати з графікою. Для завдань, пов'язаних із малюванням і заливкою, можна використовувати Canvas API, інтегроване безпосередньо в Compose. Це робить можливим створення складних графічних елементів, зокрема піксельних картин, із мінімальними зусиллями. Легко інтегровано користувацькі функції малювання в загальний інтерфейс, що важко реалізувати з такою ж ефективністю у традиційному підході.

Крім того, Jetpack Compose значно полегшує тестування та підтримку коду. Ізольованість Composable-функцій дозволяє тестувати їх незалежно одна від одної. Інструменти на кшталт Compose Preview і Hot Reload дають змогу миттєво переглядати зміни в інтерфейсі прямо в IDE, що пришвидшує розробку та зменшує кількість помилок.

Таким чином, Jetpack Compose надає не лише більш сучасний і потужний підхід до створення UI, а й відкриває нові можливості для розробників. Його переваги в декларативності, інтеграції з Kotlin, зручності роботи з графікою та підтримці інструментів роблять його оптимальним вибором для створення сучасних Android-застосунків.

Функція «Малювання» дозволяє користувачеві змінювати колір окремих клітинок у матриці шляхом натискання або перетягування пальцем по екрану.

Це реалізовано через Canvas, компонент, що дозволяє малювати графічні елементи. Матриця ділиться на клітинки розміром, що залежить від ширини Canvas. Під час торкання чи перетягування виконується перевірка координат клітинки, щоб визначити, яка клітинка була змінена, і оновлює колір цієї клітинки.

Функція заливки працює за схожим принципом, але має більш складну логіку. Коли користувач торкається певної клітинки у режимі заливки, система перевіряє, який колір має ця клітинка, і виконує заливку всіх сусідніх клітинок із тим самим кольором. Логіка заливки реалізована за допомогою алгоритму "глибини пошуку", що використовує стек для перевірки сусідніх клітинок. Кожна клітинка, яка відповідає критеріям, додається до стека, після чого її колір оновлюється. Цей підхід гарантує, що всі з'єднані клітинки з однаковим кольором будуть замінені новим кольором.

Крім того, функція заливки оптимізована таким чином, щоб уникати перезапису клітинок, які вже мають потрібний колір. Для цього використовується окрема множина, яка відстежує, які клітинки вже були оброблені.

Додатково до цього застосунок інтегрує кольорову панель для вибору відтінків, реалізовану за допомогою окремого Canvas-компонента. Ця панель генерує градієнт кольорів залежно від обраного відтінку і дозволяє користувачеві вибрати насиченість і яскравість кольору. Коли користувач натискає на панель, її координати перетворюються у значення насиченості та яскравості й оновлює обраний колір.

У застосунку реалізовано збереження та управління даними за допомогою бази даних Room, що дозволяє працювати з арт-об'єктами (рис. 4.15). Кожен арт-об'єкт представлений як сутність, що містить основну інформацію, таку як назва, час створення та матриця кольорів. Всі ці дані зберігаються у таблиці бази даних, що дозволяє ефективно організувати доступ до них (додаток Б, лістинг Б.13).

Для роботи з базою даних використовуються DAO (Data Access Object) методи, які дозволяють зберігати, оновлювати, видаляти та отримувати арт-об'єкти (додаток Б, лістинг Б.14). Дані з бази повертаються у вигляді потоків, що забезпечує автоматичне оновлення інтерфейсу користувача, коли дані змінюються. Це дозволяє досягти інтерактивності, де зміни у базі даних миттєво відображаються у застосунку без потреби вручну оновлювати UI.

Операції з базою даних виконуються асинхронно, що дає можливість уникнути блокування основного потоку застосунку і забезпечує плавну роботу. Ця система дозволяє зберігати всю необхідну інформацію про арт-об'єкти, забезпечуючи при цьому зручний доступ до них, а також дозволяє ефективно працювати з великими обсягами даних.

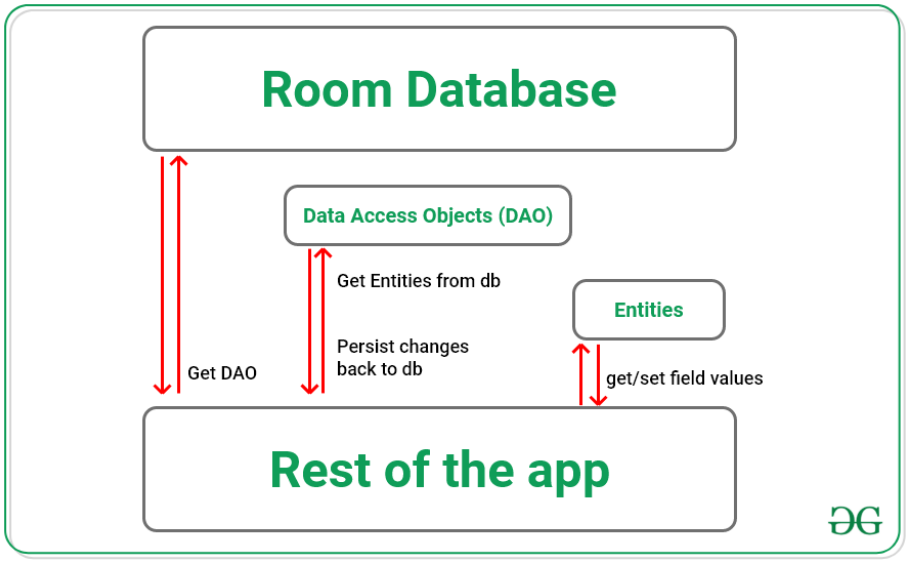

Рисунок 4.15 – Схема роботи бібліотеки Room

4.2 Arduino

Arduino відіграє ключову роль як посередник між HC-06 та світлодіодною матрицею 16x16. Він отримує команди від Android-пристрою через Bluetooth-з'єднання з HC-06, обробляє їх і відповідно управляє матрицею світлодіодів.

Arduino відповідає за інтерпретацію даних, що надходять від Android, і перетворення їх у сигнали, які контролюють кожен окремий піксель матриці.

4.2.1 Середовище розробки

Arduino IDE – це інтегроване середовище розробки, створене спеціально для програмування мікроконтролерів на базі платформ Arduino. Дане середовище забезпечує простий і зрозумілий інтерфейс, що підходить як для початківців, так і для досвідчених розробників. Arduino IDE підтримує написання, компіляцію та завантаження програмного коду на мікроконтролери, дозволяючи швидко створювати прототипи електронних пристроїв [15].

Основною мовою програмування в середовищі Arduino IDE є C++ із певними спрощеннями, що робить процес написання коду інтуїтивним навіть для тих, хто раніше не працював із мовами програмування. Програма, яку користувач створює, називається скетчем, а її структура містить дві основні функції: setup() для початкових налаштувань і loop() для циклічного виконання інструкцій.

Arduino IDE оснащено вбудованим редактором коду з підсвічуванням синтаксису, автодоповненням і базовими інструментами для дебагінгу. Важливою особливістю є підтримка бібліотек, які можна легко додавати до проєкту для розширення функціональності, наприклад, роботи з датчиками, дисплеями чи модулями зв'язку.

Після написання коду IDE забезпечує його компіляцію за допомогою вбудованого компілятора, а також завантаження прошивки на мікроконтролер через USB-з'єднання. Середовище автоматично визначає тип підключеної плати та відповідний порт, що значно спрощує взаємодію з обладнанням.

Крім цього, Arduino IDE підтримує монітор послідовного порту, який дозволяє переглядати дані, що надходять із плати в реальному часі. Це незамінний інструмент для налагодження проєктів і перевірки роботи датчиків або модулів.

Arduino IDE постійно оновлюється, що забезпечує підтримку нових моделей плат і вдосконалення інтерфейсу. Завдяки відкритому коду й активній спільноті розробників, середовище має численні ресурси для навчання та обміну досвідом. Це робить його універсальним вибором для створення проєктів у галузі робототехніки, автоматизації та IoT.

4.2.2 Технічна складова контролера

Arduino має дві основні функції: отримування даних з Bluetooth модуля та відправлення цих даних для світлодіодної матриці (додаток Б, лістинг Б.15).

Arduino використовує інтерфейс SoftwareSerial для встановлення Bluetooth-з'єднання з Android-пристроєм через модуль HC-06. Serial у контексті Arduino є одним із основних засобів комунікації між Arduino та іншими пристроями, включаючи комп’ютери, модулі Bluetooth або інші мікроконтролери. Цей інтерфейс базується на послідовному зв’язку (serial communication), який дозволяє передавати дані у вигляді послідовного потоку бітів.

Для використання послідовного зв’язку в Arduino необхідно викликати метод Serial.begin(baudRate), де baudRate визначає швидкість передачі даних (кількість бітів за секунду), у функції setup():

```
void setup() {
Serial.begin(9600);
}
```

Arduino постійно перевіряє чи є нові дані у вхідному буфері, використовуючи метод Serial.available(). Зчитування даних здійснюється методом Serial.read(), який отримує один байт даних за раз:

```
if (Serial.available()) {
     char incomingChar = Serial.read();
}
```

Дані мають вигляд “x,y,a,r,g,b;”, де x – рядок матриці, y – стовпчик матриці, a – альфа (alpha), r – червоний (red), g – зелений (green), b – синій (blue), які конвертуються в колір та присвоюються окремим клітинкам матриці (лістинг 4.1).

Лістинг 4.1 – Реалізація методу parseAndApplyColor

```
void parseAndApplyColor(String data) {

  Serial.println(data);
  int commas[5];
  int index = 0;

  for (int i = 0; i < data.length(); i++) {
    if (data[i] == ',') {
      commas[index++] = i;
    }
  }

  if (index == 5) {
    int row = data.substring(0, commas[0]).toInt();
    int col = data.substring(commas[0] + 1, commas[1]).toInt();
    int alpha = data.substring(commas[1] + 1, commas[2]).toInt();
    int red = data.substring(commas[2] + 1, commas[3]).toInt();
    int green = data.substring(commas[3] + 1, commas[4]).toInt();
    int blue = data.substring(commas[4] + 1).toInt();
  }
}
```

Для контролю матриці світлодіодів 16x16 Arduino використовує бібліотеку FastLED, яка є однією з найпопулярніших і найефективніших для роботи з адресованими світлодіодами. FastLED дозволяє Arduino керувати матрицею, індивідуально змінюючи колір і яскравість кожного світлодіода на основі переданих даних. Ця бібліотека оптимізує роботу з матрицею, забезпечуючи швидке оновлення екрану і підтримку різних ефектів, що дозволяє створювати анімації та інші візуальні ефекти на 16x16 матриці. Завдяки FastLED Arduino може коректно відображати кольори і візуальні патерни, отримані через Bluetooth-з'єднання з Android-пристроєм.

Для ініціалізації матриці світлодіодів використовується метод FastLED.addLeds<WS2812, DATA_PIN, RGB>(leds, NUM_LEDS), де WS2812 – тип світлодіодів, які використовуються. Також відомі як NeoPixel, які підтримують адресацію кожного пікселя, DATA_PIN – номер піну Arduino,

підключеного до вхідного порту даних матриці, RGB – порядок кольорів (Red, Green, Blue). Світлодіоди WS2812 зберігають колірні значення у форматі RGB, цей параметр визначає параметр у якому будуть відправлені кольори на світлодіоди. Цей метод пов'язує фізичні світлодіоди з масивом leds, де зберігаються значення їх кольорів. FastLED.setBrighness() задає глобальну яскравість для всіх світлодіодів у матриці. Значення передається у діапазоні від 0 до 255, де 0 – світлодіоди вимкнені, 255 – максимальна яскравість:

```
void setup() {
    FastLED.addLeds<WS2812, DATA_PIN, RGB>(leds, NUM_LEDS);
    FastLED.setBrightness(50);
}
```

Світлодіоди можуть споживати значну кількість енергії на максимальній яскравості. Наприклад, кожен піксель WS2812 може використовувати до 60 мА (20 мА на кожен канал кольору). Для матриці 16x16 це потенційно до 16 * 16 * 60 мА = 15.36 А. Обмеження яскравості знижує споживання енергії.

Для відображення кольору, що прийшов з Android застосунку, робиться конвертація 2D координат в 1D індекс. Встановлюється колір пікселя в масив світлодіодів та виконується команда FastLED.show():

```
int index = (row * MATRIX_SIZE) + col;
leds[index] = CRGB(red, green, blue);
FastLED.show();
```

Метод FastLED.show() є ключовим у бібліотеці FastLED, оскільки він відповідає за фактичне оновлення світлодіодної матриці або стрічки. Усе, що ви робите з масивом leds, не відображається на світлодіодах, поки не викликається цей метод. Ця функція генерує необхідний послідовний сигнал, який передається на пін даних. Усі значення кольорів з масиву діодів перетворюються на цифровий сигнал у формат, який властивий світлодіодам. Сигнал містить інформацію про кольори для кожного світлодіоду у визначеному порядку.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи, присвяченої дослідженню піксельної графіки та інструментів для її створення та розробки системи для створення піксельних картин, проведено аналіз сучасних інструментів і алгоритмів, що використовуються для розробки піксельного мистецтва.

У роботі проведено огляд піксельної графіки, розглянуто її сутність, особливості та основні принципи, що дозволило визначити ключові напрямки її використання. У процесі аналізу графічних редакторів було детально розглянуто такі популярні інструменти для створення піксельного мистецтва, як Asperite, Piskel та Adobe Photoshop, використання яких значно полегшує процес створення піксельної графіки, дозволяючи працювати з кольоровою палітрою, шарами та інструментами для автоматизації повторюваних завдань. Проведено аналіз алгоритмічних рішень, що використовуються для малювання піксельних зображень, які відіграють ключову роль у створенні високоякісних графічних зображень з точними геометричними формами.

В роботі розроблено структурну та функціональну схеми системи для створення моделей піксельних картин, здійснено вибір компонентів, які є складовими системи: мікроконтролера, Bluetooth модуля, LED матриці та Android застосунку. Основним компонентом є мікроконтролер Arduino UNO, який має всі необхідні технічні характеристики для створення моделей піксельних зображень. Bluetooth модуль HC-06 виконує важливу роль у передаванні даних з Android застосунку на мікроконтролер, після чого LED матриця змінює колір відповідних діодів відповідно до отриманих даних

У процесі розробки Android-застосунку було обрано архітектуру MVVM, що забезпечує чітке розділення відповідальностей між компонентами і полегшує тестування та підтримку. Використання Jetpack Compose замість традиційного XML дозволяє значно зменшити об’єм програмного коду та

спрощує інтеграцію з інструментами Android, такими як LiveData і Flow, що підвищує ефективність розробки та знижує ймовірність помилок.

Для збереження стану матриці та налаштувань було обрано локальну базу даних Room, що дозволяє ефективно працювати з великими об'ємами інформації. Асинхронна обробка даних здійснюється за допомогою Kotlin Coroutines та Flow, що забезпечує плавне оновлення інтерфейсу без блокування основного потоку.

У застосунку інтегровано Coil для ефективної обробки зображень та Spotify API для інтеграції з музичним контентом. Для передачі даних між пристроями використано Bluetooth, що дозволяє швидко обмінюватися інформацією між Android і Arduino.

Обрані технології створюють потужну екосистему, що забезпечує високу продуктивність, масштабованість та зручний досвід для користувачів.

Таким чином, результати роботи підтвердили важливість піксельної графіки як засобу створення візуального контенту у різних галузях, зокрема у відеоіграх, мобільних застосунках, веб-дизайні та рекламі. Розглянуті алгоритми і графічні редактори дозволяють не лише створювати унікальні візуальні образи, але й оптимізувати процес їх розробки, забезпечуючи високу якість зображень та анімацій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is pixel graphics? [Електронний ресурс]. – Режим доступу: https://www.mr-beam.org/en/blogs/news/was-ist-eine-pixelgrafik (дата звернення: 01.06.2024).

2. Демиденко М.А. Комп'ютерна графіка, дизайн та мультимедіа: навч. посіб./ М.А. Демиденко // Міністерство освіти і науки України, Нац. Техн ун-т «Дніпровська політехніка » – Д. : 2022.– 123 с.

3. Маценко В.Г. Комп'ютерна графіка [Електронний ресурс]. – Режим доступу: https://mmi.stu.cn.ua/wpcontent/uploads/2016/09/MatsenkoKompGrafyka.pdf

4. Мідіна С.С Моделі та інструментальні засоби для створення піксельних картин / С.С. Мідіна // Матеріали тез 28–го міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» – м. Харків, травень 2024. – С. 32–33.

5. Pixel Art is a Timeless Art Form That Has Seen a Comeback [Електронний ресурс]. – Режим доступу: https://www.kodrick.com/blog-post/pixel-art-is-a-timeless-art-form (дата звернення: 01.06.2024).

6. The Art of Pixel Graphics Reviving Retro Aesthetics in Modern Games [Електронний ресурс]. – Режим доступу: https://medium.com/@mindfig.llp/the-art-of-pixel-graphics-reviving-retro-aesthetics-in-modern-games-81dd1306c788 (дата звернення: 01.07.2024).

7. Pixel Art for Game Character Design [Електронний ресурс]. – Режим доступу https://www.researchgate.net/publication/362831872_Research_on_the_Application_of_Pixel_Art_in_Game_Character_Design (дата звернення: 30.11.2024).

8. Bresengams Line Algorithm [Електронний ресурс]. – Режим доступу: https://digitalbunker.dev/bresenhams-line-algorithm (дата звернення: 01.06.2024).

9. Anti-Aliasing Fundamentals for Pixel Artists [Електронний ресурс]. – Режим доступу: https://pixelparmesan.com/anti-aliasing-fundamentals-for-pixel-artists/ (дата звернення: 01.06.2024).

10. Guangzhou HC Information Technology Co., Ltd. HC-06 Datasheet [Електронний ресурс]. – Режим доступу: https://www.alldatasheet.com/datasheet-pdf/view/1179032/ETC1/HC-06.html (дата звернення: 10.12.2024).

11. Android Architecture [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/android-architecture/ (дата звернення: 30.11.2024).

12. Bluetooth overview [Електронний ресурс]. – Режим доступу: https://developer.android.com/develop/connectivity/bluetooth (дата звернення: 10.12.2024).

13. Introduction to Compose [Електронний ресурс]. – Режим доступу: https://medium.com/@ecemokan/introduction-to-jetpack-compose-c042e2544db6 (дата звернення: 30.11.2024).

14. Getting Started with Room Database in Android [Електронний ресурс]. – Режим доступу: https://amitraikwar.medium.com/getting-started-with-room-database-in-android-fa1ca23ce21e (дата звернення: 30.11.2024).

15. Everything you need to know about Arduino code [Електронний ресурс]. – Режим доступу: https://www.circuito.io/blog/arduino-code/ (дата звернення: 30.11.2024).