ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І.І.МЕЧНИКОВА
(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних систем та технологій
(повна назва кафедри (предметної, циклової комісії)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»
(освітньо-кваліфікаційний рівень)

« Розширення діапазону візуалізації алгоритмів комп'ютерної графіки »

« Expanding the range of visualization of computer graphics algorithms »

Виконав: здобувач денної форми навчання
спеціальності 122 Комп'ютерні науки
(код, назва спеціальності)
Освітня програма ОП - Комп'ютерні науки
(назва)
Антіпов Михайло Михайлович
(прізвище, ім'я, по-батькові здобувача)
Керівник канд. ф-м. н., доц. Шугайло Ю.Б. _____
(науковий ступінь, вчене звання, прізвище,ініціали) (підпис)
Рецензент канд. т. н., доц. Ларин Д.Г.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№ ___ від «____» _____ 2024 р.

Завідувач кафедри
_____ Юрій ГУНЧЕНКО
(підпис) (прізвище, ім'я)

Захищено на засіданні ЕК № _____
Протокол № ___ від «___» _____ 2024 р.
Оцінка _____ / ____ / _____
(за національною шкалою, шкалою ECTS, бали)

Голова ЕК
_____ Микола МАЛАКСІАНО
(підпис) (прізвище, ім'я)

Одеса - 2024

АНОТАЦІЯ

У дипломній роботі розробляється тема «Розширення діапазону візуалізації алгоритмів комп'ютерної графіки».

Мета роботи – розробка та дослідження можливостей застосування пристрою, здатного відображати тривимірну інформацію. Ідея пристрою полягає у створенні тривимірного масиву зі світлодіодів, які утворюють куб із розміром 8x8x8, при цьому кожен зі світлодіодів має можливість окремого керування.

В результаті роботи розроблено апаратну та програмну частину пристрою для візуалізації тривимірних даних. Також проаналізовані можливості використання пристрою у різних галузях.

ANNOTATION

This work explores the topic "Expanding the Range of Visualization for Computer Graphics Algorithms".

The objective of the work is to develop and investigate the capabilities of a device capable of displaying three-dimensional information. The concept of the device involves creating a three-dimensional array of LEDs forming an 8x8x8 cube, where each LED can be controlled individually.

As a result of this work, both the hardware and software components of the device for visualizing three-dimensional data have been developed. Additionally, the potential applications of the device in various fields have been analyzed.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

Скорочення

ОС – операційна система

ПЗ – програмне забезпечення

ЦАП – цифро-аналоговий перетворювач.

Умовні позначення

Бод – одиниця виміру швидкості передачі символьної інформації, рівний 1 символ / с.

Терміни

Парсинг – процес аналізу вхідної послідовності символів, з метою розбору граматичної структури згідно із заданою формальною граматикою.

Регістр зсуву – послідовний каскад тригерів, що мають один і той самий тактовий сигнал, в якому вихід кожного тригера сполучений з входом даних наступного тригера у каскаді, утворюючи ланцюг який зсуває на одну позицію бітову мапу що в ньому зберігається, при кожній подачі тактового сигналу.

ШІМ – широтно-імпульсна модуляція – метод модуляції сигналу, в якому ширина імпульсів сигналу змінюється в залежності від амплітуди сигналу, який модулюється.

HSV – колірна модель, заснована на трьох характеристиках кольору: колірному тоні (Hue), насиченості (Saturation) і значенні кольору (Value).

SDK – комплект для розробки програмного забезпечення.

UART – фізичний протокол передачі даних.

ВСТУП

У світі постійно зростає попит на нові методи візуалізації людини-оператора з графічними об'єктами, які можуть бути інтерфейсом виконавчих пристроїв. Візуалізація даних може дозволити побачити закономірності, які не завжди очевидні при роботі з текстовими, числовими, або двомірними графічними (схеми, графіки, діаграми) даними. Тривимірна візуалізація може допомогти краще уявити складні об’єкти та процеси. Візуалізація даних відіграє значну роль у багатьох сферах життя, адже вона дозволяє:

а. Ефективно доносити інформацію до користувача. Більшу частину інформації людина сприймає через зір, тому візуальні образи є більш зрозумілими та запам'ятовуються краще, ніж текст або цифри.
б. Підвищити аналітичні можливості. Візуалізація даних дає можливість побачити закономірності, які не завжди очевидні при роботі з текстовими або числовими даними.
в. Сприяти кращому розумінню складних концепцій. Тривимірна візуалізація може допомогти користувачам краще уявити собі складні об'єкти або процеси.

Актуальність цієї роботи полягає у потребі розвитку та вдосконалення засобів відображення об’єктів у тривимірному просторі. Сучасні пристрої для візуалізації даних, такі як монітори, телевізори, екрани, мають двомірну природу. Це обмежує їх можливості при візуалізації тривимірної інформації, адже вони можуть відображати лише двомірні проекції тривимірних об'єктів. Існує ряд досліджень та розробок для вирішення цієї проблеми, проте вони мають значні недоліки. Наприклад, основним підходом до вирішення проблеми відображення тривимірних об’єктів є стереоскопія, проте стереоскопічне зображення лише створює ілюзію тривимірного зображення. зображення за рахунок бінокулярного зору людини. При перегляді такого зображення під іншим кутом ефект глибини зникає.

Також для вирішення проблеми використовується голографія. Голографічне зображення відображає об’ємне зображення незалежно від куту зору. Наприклад, існує метод побудови голограми за допомогою фемтосекундного лазера, який змушує світитися матерію у точці фокусу. Проте такий спосіб вимагає дорогого обладнання та створює голограми дуже маленького розміру (1 $см^3$)[1].

Метою даної роботи є розробка пристрою, здатного відображати тривимірну інформацію, та дослідження можливостей його застосування. Для досягнення цієї мети необхідно вирішити декілька завдань:

а. Розробити апаратну та програмну частину пристрою.
б. Розробити інтерфейс для взаємодії з пристроєм.
в. Проаналізувати можливості застосування пристрою у різних галузях.

За основу пристрою візуалізації тривимірних даних було вирішено взяти ідею тривимірного масиву зі світлодіодів, які формують куб розміром 8x8x8 точок, з можливістю індивідуального керування кожним світлодіодом.

1. РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ ПРИСТРОЮ

Для реалізації задач роботи було вирішено взяти ідею тривимірного масиву зі світлодіодів, які формують куб розміром 8x8x8 точок. Ця ідея не нова, проте відомі реалізації мають значні обмеження. Пристрої розміром 8x8x8 або більше точок переважно виконуються з використанням світлодіодів, які здатні випромінювати світло лише одного кольору[2]. Рідше зустрічаються варіанти подібних пристроїв, які використовують трикольорові світлодіоди, проте вони значно менші за кількістю світлодіодів, зазвичай 5x5x5 точок, через що кількість світлодіодів менше у чотири рази. Це пов’язано з особливостями компонентів, за допомогою яких відбувається реалізація масиву зі світлодіодів.

За основу апаратної частини було вирішено взяти електронну платформу Arduino Nano. Ця платформа побудована на базі мікроконтролеру Atmel ATmega328P[3], який містить 32КБ флеш-пам’яті та 2КБ ОЗУ та працює на частоті 16МГц. Контролер має 14 цифрових виводів, 6 з яких можуть використовуватися для генерації ШІМ-сигналу, а також 8 аналогових входів. Ця платформа має доступну ціну, та має достатні обчислювальні можливості для реалізації поставлених задач.

Також для забезпечення можливості керування через Bluetooth було обрано готовий модуль JDY-33[4]. Цей модуль виконує обмін даними з мікроконтролером за протоколом UART на швидкості 115200 бод, який апаратно підтримується платформою Arduino Nano. Цей модуль підтримує як стандарт Bluetooth Classic 4.2, так і стандарт Bluetooth Low Energy, який надалі використовувався для комунікації з мобільним додатком.

1.1. ПОБУДОВА АРХІТЕКТУРИ ПРИСТРОЮ

Керування кожним світлодіодом за допомогою окремих виводів мікроконтролера є недосяжним у зв'язку з обмеженою кількістю доступних

виводів. Зазвичай для вирішення цієї проблеми застосовується схема підключення світлодіодів при якій контакти живлення світлодіоду об’єднуються у стовпці, а інші контакти світлодіодів об’єднуються у шари. Мікроконтролер задає у яких стовпцях та на якому шарі повинні увімкнутися світлодіоди та швидко перемикає шари по черзі. Швидке перемикання шарів завдяки особливостям зору створює ілюзію постійної роботи. Таке підключення дозволяє значно зменшити кількість необхідних з’єднань.

Проте, навіть при такому підключенні світлодіодів, для кубу розміром 8x8x8 кількість потрібних виводів становить $64 + 8 = 72$ виводи, що все ще значно більше ніж має мікроконтролер. У випадку однокольорових світлодіодів це вирішується за допомогою використання регістрів зсуву. Регістри зсуву отримують цифровий сигнал на контакт послідовного входу та розбивають послідовність бітів на паралельні вихідні контакти, що дозволяє вмикати та вимикати світлодіоди підключені до паралельних виводів регістру зсуву за допомогою лише одного керуючого контакту мікроконтролера.

У випадку застосування трикольорових світлодіодів, найрозповсюдженими є трикольорові світлодіоди з загальним виводом. Цей тип світлодіодів є збіркою з трьох окремих світлодіодів в одному корпусі. Таким чином такий світлодіод має 4 виводи: один – загальний для усіх вбудованих світлодіодів анод або катод, а усі інші – окремий другий вивід кожного із світлодіодів. Використання регістрів зсуву з таким типом трикольорових світлодіодів надає значне обмеження. Для задання кольору випромінювання світлодіода необхідно задавати яскравість кожного з трьох складових кольорових світлодіодів окремо. Регулювання яскравості світлодіодів досягається за допомогою ШІМ або регулювання напруги живлення.

Останній спосіб є достатньо складним у реалізації, адже мікроконтролер є цифровим пристроєм, тому для перетворення сигналу з мікроконтролера на

рівень необхідної напруги необхідно використовувати ЦАП, що значно ускладнює схему.

При використання ШІМ регулювання кольору одного трикольорового світлодіода із загальним виводом потребує три контакти мікроконтролера з підтримкою ШІМ. Вирішити виникаючу при цьому проблему з кількістю виводів мікроконтролера використання регістру зсуву вже не дозволяють. Регістри зсуву не можуть працювати із ШІМ-сигналом, тому їх використання можливо лише при умові, що світлодіоди будуть мати тільки стан «увімкнено» або «вимкнено», таким чином обмежуючи кількість кольорів, які можна відобразити, до восьми (включаючи чорний). Неможливість використання ШІМ та регістра зсуву одночасно пов’язано з обмеженою швидкістю перемикання стану виводів регістру зсуву.

Для забезпечення можливості окремого регулювання світлодіодів було вирішено використовувати інший тип світлодіодів – адресний. Адресні світлодіоди так само містять у корпусі три світлодіоди різних кольорів, проте додатково у корпус вбудована мікросхема-драйвер. Такі світлодіоди так само мають чотири виводи (рис. 1.1), але їх призначення інше. Виводи «+5V» та «GND» є контактами живлення, «DIN» – контакт для отримання керуючого цифрового сигналу, «DOUT» – контакт для передачі керуючого цифрового сигналу наступному світлодіоду. Вбудована мікросхема-драйвер приймає цифрові команди від контролера через вивід «DIN» і, відповідно до цих даних, регулює яскравість кожного з вбудованих світлодіодів, що дозволяє отримати будь-який загальний колір і яскравість. При послідовному з’єднанні декількох світлодіодів мікроконтролер посилає команди всім світлодіодам через контакт «DIN» першого світлодіода, драйвер першого світлодіода обробляє свою частину даних і задає колір свого світлодіода, а потім передає наступні дані на вхід наступного світлодіода і так до кінця ланцюга із адресних світлодіодів. Це дозволяє використовувати один вивід мікроконтролера для керування одразу багатьма світлодіодами. Додатковою перевагою такого типу

світлодіодів порівняно із трикольоровими світлодіодами з загальним виводом є відсутність необхідності підключати усі чотири виводи з кожного світлодіоду до плати: достатньо лише двох контактів живлення для кожного із світлодіодів та одного цифрового входу з першого світлодіода. Виходячи з цих переваг було обрано світлодіоди саме цього типу YF923-F5[5].

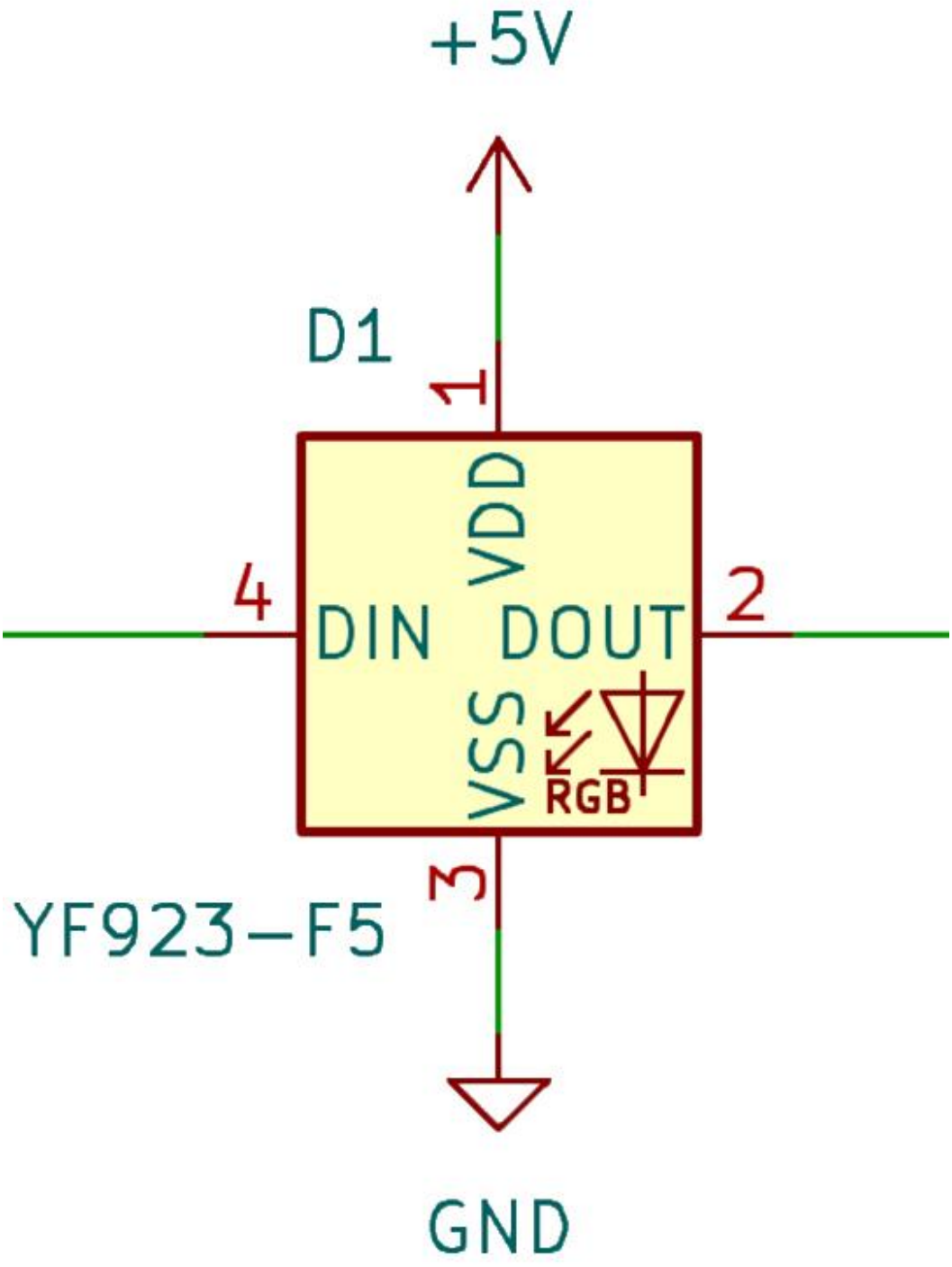


Рисунок 1.1 – Виводи адресного світлодіоду

1.2. РОЗРОБКА СХЕМИ З’ЄДНАННЯ

1.2.1. СХЕМА З’ЄДНАННЯ СВІТЛОДІОДІВ

Світлодіоди з’єднуються пошарово. Кожен шар розміром 8x8 підключається за схемою на рисунку 1.2. На схемі з’єднані виводи світлодіодів для передачі цифрового сигналу. Як можна побачити, при зміні напрямку підключення також змінюється розташування контактів живлення «+5V» та «GND». Це пов’язано з необхідністю фізично розгорнути світлодіод для підключення вхідного виводу до вихідного виводу попереднього світлодіоду. Таких шарів відповідно до розміру кубу вісім, кожен з яких з’єднується з окремим виводом мікроконтролера.

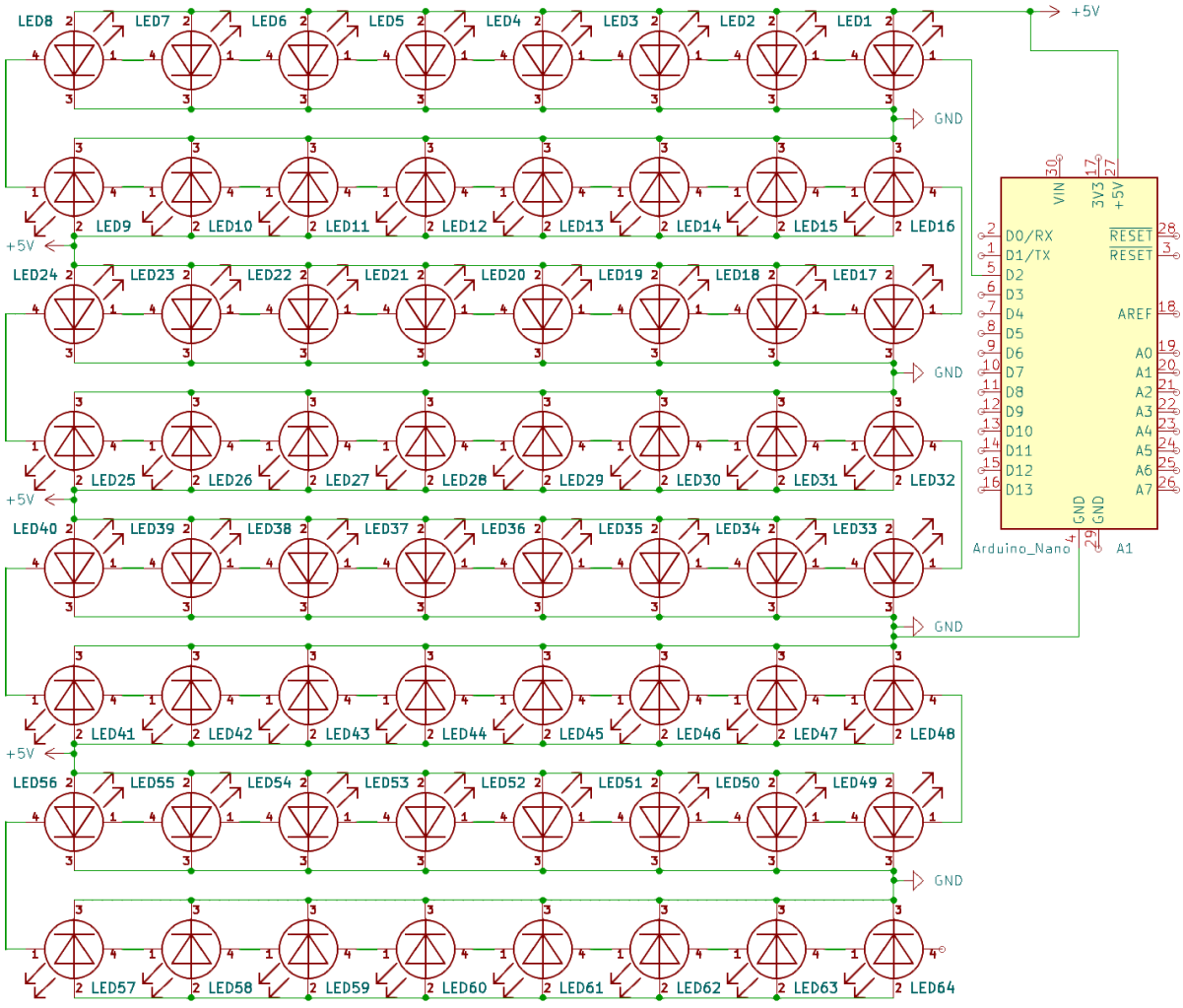


Рисунок 1.2 – Схема підключення світлодіодів у одному шарі

Підключення контактів «+5V» та «GND» відбувається за допомогою вертикальних колон з металевого дроту. Ці стовпці забезпечують як структурну підтримку шарів, так і є провідниками для підключення живлення світлодіодів. Змодельоване зображення цих стовпців наведено на рисунку 1.3.

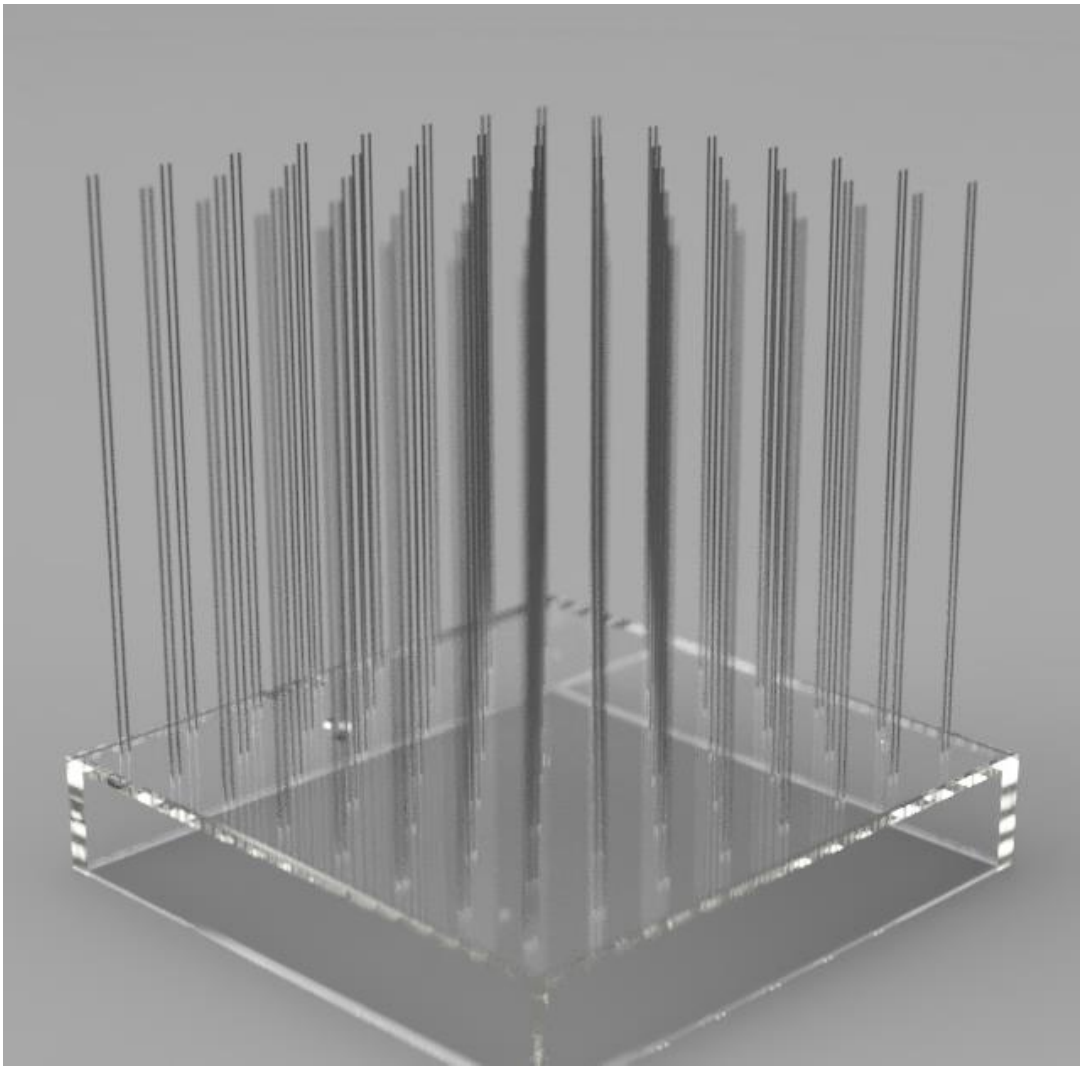


Рисунок 1.3 – Контактні колони

1.2.2. ЕЛЕКТРИЧНА СХЕМА ПРИСТРОЮ

Схема підключення компонентів є достатньо простою (рис 1.4). Виводи мікроконтролера D2-D9 через резистори R1-R8 з’єднуються з ввідними виводами перших світлодіодів у шарах (помічені як DATA-1 – DATA-8). Ці резистори бажано використовувати, бо у випадку відключення контакту

живлення від світлодіода він спробує отримати живлення через контакт входу цифрового сигналу, що може призвести до перенавантаження виводу контролера та виходу його з ладу. Таким чином ці резистори використовуються для зберігання працездатності Arduino у випадку нештатної ситуації.

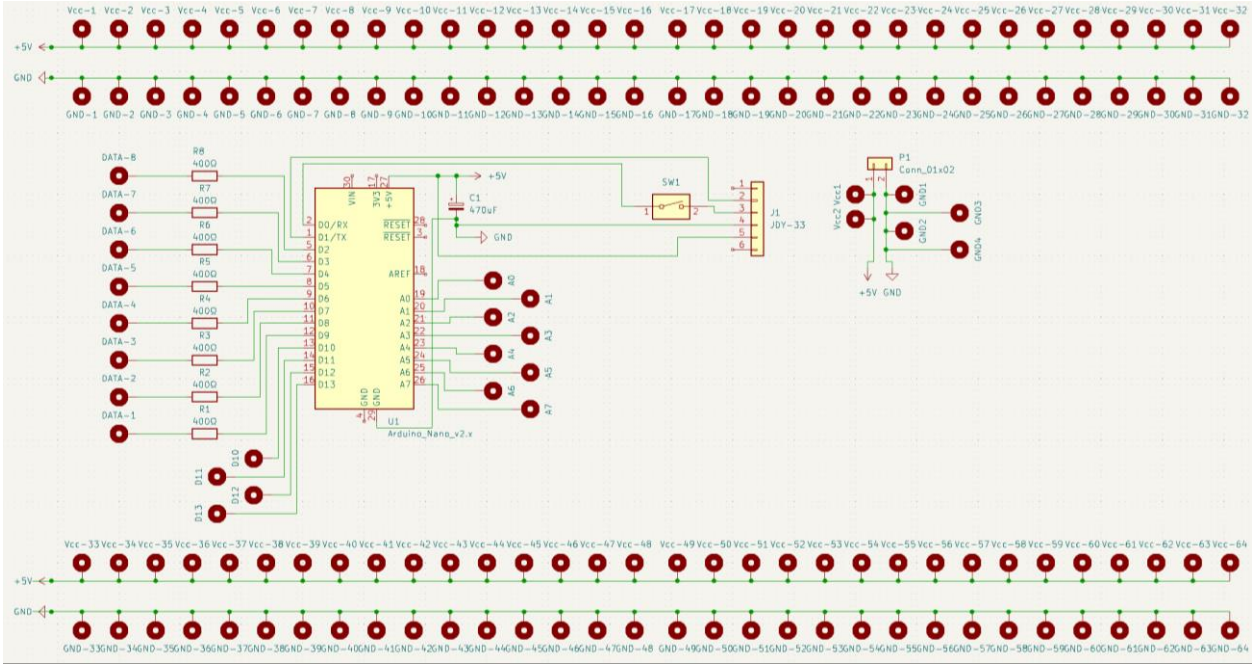


Рисунок 1.4 – Схема з’єднання компонентів

Мікроконтролер позначено на схемі як U1. Паралельно живленню мікроконтролера встановлено електролітичний конденсатор C1 для підвищення стабільності роботи контролера у випадку можливих пульсацій напруги. Усі цифрові та аналогові виводи мікроконтролера виведені на схемі для можливості підключення до них додаткових компонентів у майбутньому.

Bluetooth-модуль позначено як J1. Підключення його контакту Tx до контакту Rx контролера відбувається через перемичку. Ця перемичка дозволяє від’єднувати модуль від Arduino для здійснення прошивки.

Як P1 на схемі позначено конектор живлення плати. До живлення паралельно підключено по 64 контакти для «+5V» та «GND», до цих контактів під’єднуються стовпці зі світлодіодами. Також до живлення підключено

декілька додаткових виводів, на випадок якщо знадобиться під’єднати додаткові компоненти.

1.3 РОЗРОБКА ДРУКОВАНОЇ ПЛАТИ

Розробка друкованої плати почалась з визначення її розмірів та кількості провідних шарів. Розмір плати повинен бути якомога менше для забезпечення максимальної щільності тривимірного дисплею, проте залишити достатньо вільного місця для збирання пристрою. Виходячи з цих міркувань було обрано розмір 210х210мм. Кількість провідних шарів було обрано найменшим необхідним для розведення схеми, адже виготовлення плат з меншої кількістю шарів значно дешевше. Таким чином плата складається з двох шарів провідника.

Компоненти було вирішено розташувати знизу плати для забезпечення можливості їх легкої заміни при необхідності, а кріплення до корпусу пристрою буде здійснюватися за допомогою п’яти отворів для кріплення. На рисунку 1.5 наведено зовнішній вигляд нижньої сторони розробленої плати. У правому нижньому куту знаходяться контакти DATA-1…8 для підключення керуючих виводів світлодіодів до мікроконтролера. Після зборки ці виводи будуть розташовані у задньому правому куті готового пристрою. Поряд з цими контактами розташовано вісім кріплень для резисторів R1…R8. Місце для встановлення Arduino позначено U1, а Bluetooth-модуля – U2. Міткою P1 помічено місце для кріплення контактів живлення пристрою. C1 – місце для встановлення конденсатору. SW позначено перемичку з’єднуючу контакт Tx Bluetooth-модуля з контактом Rx мікроконтролера.

По усій платі на рівній відстані між собою розташовані групи по два контакти з позначенням 1…128. Ці контакти необхідні для підключення контактів живлення світлодіодів. Контакти 1…64 під’єднані до «+5V», а 65…128 до «GND». Також у верхній частині плати розведені усі невикористані

контакти Arduino для можливості додати компоненти у пристрій у майбутньому.

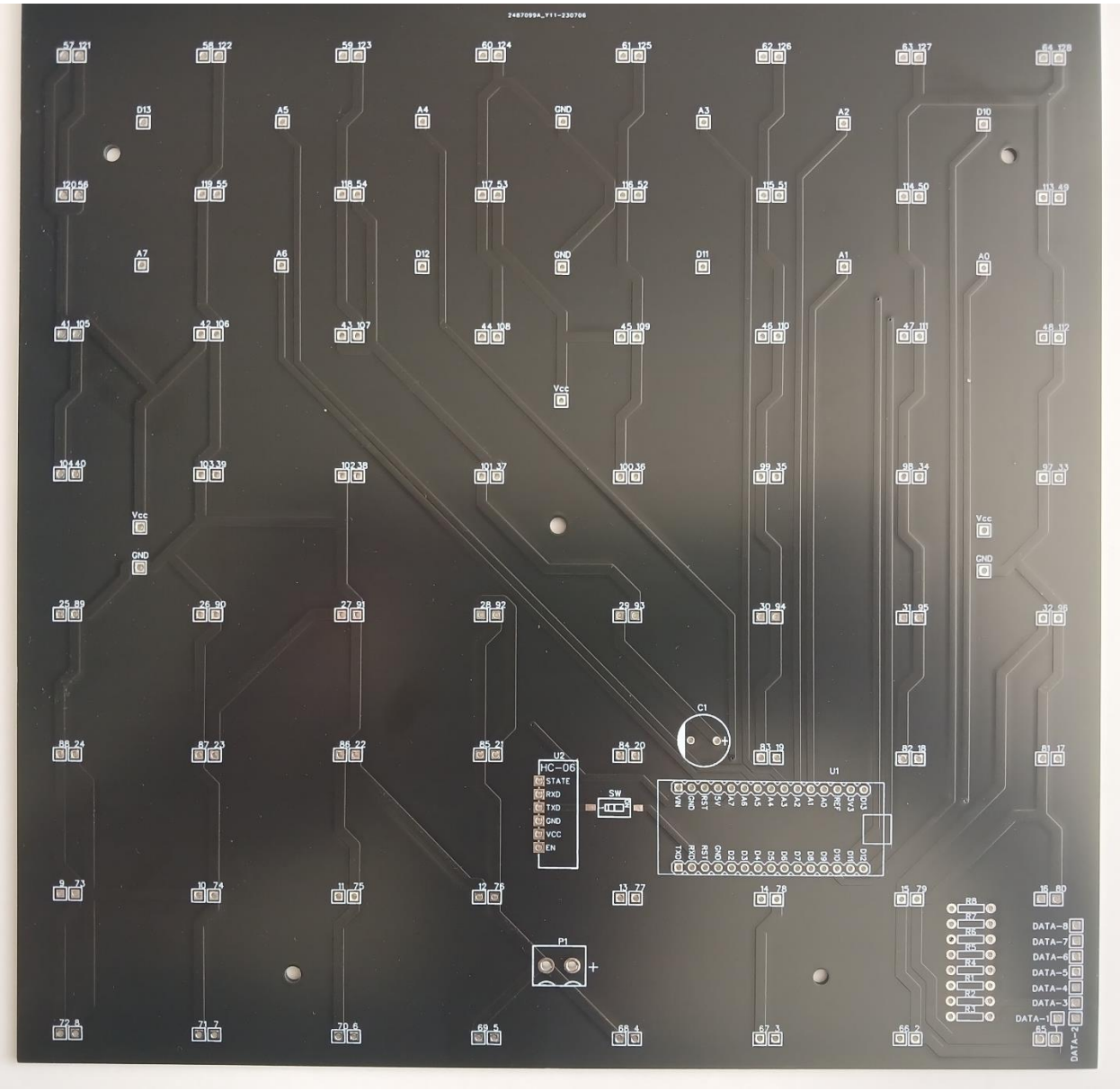


Рисунок 1.5 – Нижня сторона друкованої плати

Повністю зібраний пристрій наведено на рисунку 1.6.

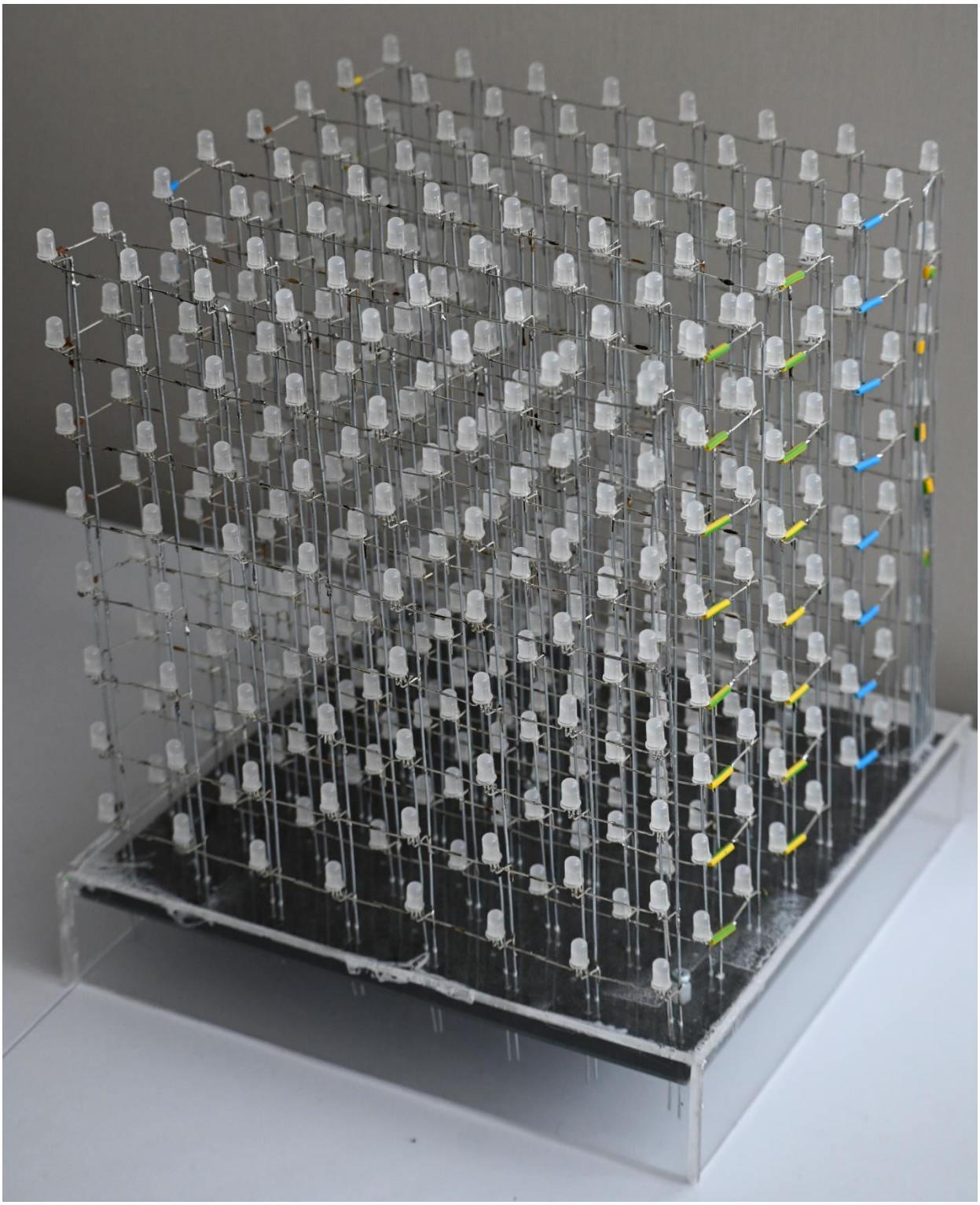


Рисунок 1.6 – Зібраний пристрій

2. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. ПРОТОКОЛ ОБМІНУ ДАНИХ

Першим кроком при розробці програмного забезпечення необхідно розробити протокол обміну даних між мобільним застосунком та пристроєм, що розроблюється.

Керуючі команди було поділено на три групи, кожній з яких присвоєно числове позначення:

- Налаштування – «0»;
- Ефекти - «1»;
- Ігри - «2».

Обмін даними через Bluetooth відбувається у форматі «*число_1,число_2,...,число_n;*», де перше число задає групу до якої відноситься команда, друге число – номер команди з цієї групи, а усі інші числа – параметри цієї команди. Для розділення чисел між собою використовується символ «,», а закінчення команди позначається символом «;».

2.1.1. КОМАНДИ НАЛАШТУВАННЯ

Команда налаштування наразі лише одна – увімкнути/вимкнути відображення та кодується числом «0». При цьому, якщо куб отримує цю команду без параметру, пристрій відправляє через Bluetooth поточний стан. Це дозволяє мобільному застосунку при підключенні к пристрою дізнатися його поточний стан та відповідно скоригувати інтерфейс. Якщо ж відправити команду «0,0,0;» відображення на пристрої вимкнеться, а командою «0,0,1;» – ввімкнеться. Розберемо запис «0,0,1;» більш детально: перший нуль позначає групу команд «Налаштування», другий – позначає команду увімкнути/вимкнути, а останнє число «1» – параметр команди який означає що

відображення необхідно увімкнути. Якщо останнє число замінити на «0» то команда буде вимикати відображення.

2.1.1. КОМАНДИ ЕФЕКТІВ

Пристрій може виводити декілька кольорових ефектів:

- Заливка – повністю заповнює усі світлодіоди обраним кольором;
- Малювання – дозволяє індивідуально керувати кожним із світлодіодів;
- Хвиля – виводить анімацію хвилі заданого кольору із заданою швидкістю;
- Дощ – виводить анімацію «дощу» заданого кольору із заданою швидкістю. Алгоритм задає колір декільком випадковим пікселям на верхньому шарі, після чого вони зміщаються донизу;
- Зміна кольорів – поступова зміна усіх можливих кольорів із заданою швидкістю.

Кодування та параметри кожного з ефектів наведено у таблиці 2.1. Літера R позначає значення для красного кольору, G – для зеленого, B – для синього, у дужках позначено діапазон допустимих значень цього параметру.

Таблиця 2.1 – Кодування ефектів

Код	Назва ефекту	№ параметру					
		1	2	3	4	5	6
0	Заливка	R (0-255)	G (0-255)	B (0-255)			
2	Малювання	Шар* (0-7, 9)	Рядок (0-7)	Стовпець (0-7)	R (0-255)	G (0-255)	B (0-255)

3	Хвиля	R (0-255)	G (0-255)	B (0-255)	Швидкість (0-255)		
4	Дощ	R (0-255)	G (0-255)	B (0-255)	Швидкість (0-255)		
5	Зміна кольорів	Швидкість (0-255)					

* - цей режим приймає параметр «9» для очищення заданого шару, наприклад команда «1,2,9,5;» відключить усі світлодіоди на шостому шарі.

Наприклад, для задання жовтого кольору світлодіоду у верхньому правому ближчому куті необхідні наступні значення параметрів:

а. Шар – 7;
б. Рядок – 0;
в. Стовпець – 7;
г. R – 255;
д. G – 255;
е. B – 0.

Також необхідно вказати номер групи команди – «1» та номер команди малювання – «2». Таким чином уся команда буде мати вигляд: «1,2,7,0,7,255,255,0;».

2.1.1. КОМАНДИ ІГОР

У пристрої реалізовано класичну гру «Змійка» з кодом «0». Для початку гри необхідно надіслати команду ініціалізації, параметри якої наведені у таблиці 2.2.

Таблиця 2 – Параметри команди ініціалізації гри «Змійка»

Колір голови змійки			Колір тіла			Колір яблука			Складність
R	G	B	R	G	B	R	G	B	0-2

В залежності від складності змінюється швидкість переміщення змійки. Після активації гри зміна напрямку руху змійки здійснюється командами наведеними у таблиці 3.

Таблиця 3 – Команди зміну руху

Вгору	Вниз	Ліворуч	Праворуч	Вперед	Назад
0	1	2	3	4	5

При цьому під час гри пристрій надсилає у мобільний застосунок бали, які отримав гравець та чи не програв він. Наприклад, пристрій надіслав команду «2,0,1,25;». Число «2» – група команди «Ігри», «0» – гра «Змійка». Третій параметр кодує статус гри: «1» – гра триває, «0» – гру закінчено. Останнє число є кількістю набраних балів. Таким чином, ця команда свідчить, що гравець набрав 25 балів та гра ще не закінчена. Команда ж «2,0,0,25;» означає що гра закінчена, а гравець набрав 25 балів.

2.2. РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ

Мобільний застосунок написано для ОС Android мовою програмування Java із використанням Android SDK[6]. Інтерфейс додатку задається за допомогою мови розмітки XML. Основними компонентами застосунку для ОС Android є файли з програмним кодом, файли з розміткою інтерфейсу та файли ресурсів, такі як відео, зображення та значення текстових змінних. Код застосунку розміщено у додатку А.

2.2.1. ЗАГАЛЬНА СТРУКТУРА МОБІЛЬНОГО ЗАСТОСУНКУ ANDROID

Кожен застосунок для ОС Android складається з трьох ключових елементів.

Першим ключовим елементом будь-якого застосунку на Android є «активність», яка задається класом Activity[7], та представляє собою екран користувача. Будь-яка активність у додатку має бути нащадком класу Activity та має власний життєвий цикл. Застосунок маже мати декілька активностей та переключатися між ними під час роботи. Кожний застосунок має принаймні одну активність, яка вважається активністю за замовчуванням, та запускається при відкритті застосунку.

Другим елементом є Manifest[8]. Файл AndroidManifest.xml – це основний конфігураційний файл застосунку Android, який міститься у каталозі «manifests». Він містить інформацію про застосунок, таку як його ім'я, версію, список активностей, служб, а також які привілеї повинен мати застосунок.

Останнім елементом є каталог ресурсів[9]. Основними ресурсами, які використовує застосунок, є макети екранів, зображення, рядки, кольори та кольорові теми. Використання файлів ресурсів дозволяє звертатися до них з коду застосунку, що полегшує їх управління та підтримку. Ресурси

зберігаються у підкаталогах, наприклад у каталозі «layout» розміщуються файли розмітки екранів.

2.2.2. ВИКОРИСТАНІ МОЖЛИВОСТІ ANDROID SDK

Під час розробки застосунку було використано деякі з можливостей, які вбудовані у Android SDK, для використання деяких можливостей операційної системи.

Android SDK дозволяє перевантажувати деякі методи вбудованих класів для виконання додаткового коду, що дозволяє змінювати функціонал вбудованих класів за потреби. Наприклад, при запуску будь-якого Activity викликається метод OnCreate, який буде викликатися на початку роботи активності. Перевантаження цього методу часто використовувалось під час розробки застосунку.

Android SDK надає інструмент для створення модульних елементів користувацького інтерфейсу, який має назву «фрагменти». Кожен фрагмент є нащадком класу Fragment[10]. Робота з фрагментами схожа на роботу з активністю, проте фрагмент має власні методи, розмітку інтерфейсу та життєвий цикл. Фрагмент повинен мати батьківську активність або фрагмент. Фрагменти використовувались про розробці застосунку для швидкого перемикання елементів інтерфейсу без довгого відкриття нових активностей.

Ще одним інструментом, який широко використовувався про розробці застосунку – «Налаштування застосунку». За допомогою інтерфейсу SharedPreferences[11] можливо зберігати та зчитувати дані під час роботи застосунку у вигляді пар «ключ»-«значення». Дані, збережені таким чином, зберігаються на пристрої навіть після завершення роботи застосунку та недоступні іншим застосункам на мобільному пристрої.

2.2.3. СТРУКТУРА РОЗРОБЛЕНОГО ЗАСТОСУНКУ

Розроблений застосунок складається з декількох активностей та фрагментів, які будуть розглянуті далі. Також біло розроблено декілька додаткових класів (додаток Б):

- DrawField – клас який відповідає за збереження сітки з кольорами для ефекту малювання;
- BluetoothConnectionHelper – клас, який відповідає за роботу з Bluetooth;
- Event<T> – клас, для зберігання значень, які надходять з пристрою.

Усі строкові значення зберігаються у файлі ресурсів strings.xml. Це дозволяє легко створювати переклади строкових значень на інші мови без змін у коді застосунку.

2.2.3.1 АКТИВНІСТЬ CHOOSEDEVICEACTIVITY

Першою активністю, яку побачить користувач при першому запуску застосунку, буде ChooseDeviceActivity (рисунок 2.1). Ця активність відповідає за обрання пристрою до якого необхідно підключитися за допомогою Bluetooth.

Рисунок 2.1 – Інтерфейс активності ChooseDeviceActivity

Активність для вибору пристрою при виконанні методу OnCreate першим кроком задає розмітку екрану з файлу activity_choose_device.xml та перевіряє наявність у застосунка дозволу на використання функції Bluetooth. Якщо використання Bluetooth не дозволено застосунок запрошує в користувача дозвіл на його використання.

Якщо доступ до Bluetooth дозволено перевіряється чи не має в налаштуваннях застосунку збереженого пристрою. Якщо він є ця активність

запускає головну активність застосунку та завершується. В іншому випадку застосунок проводить пошук пристроїв Bluetooth та виводить їх у список.

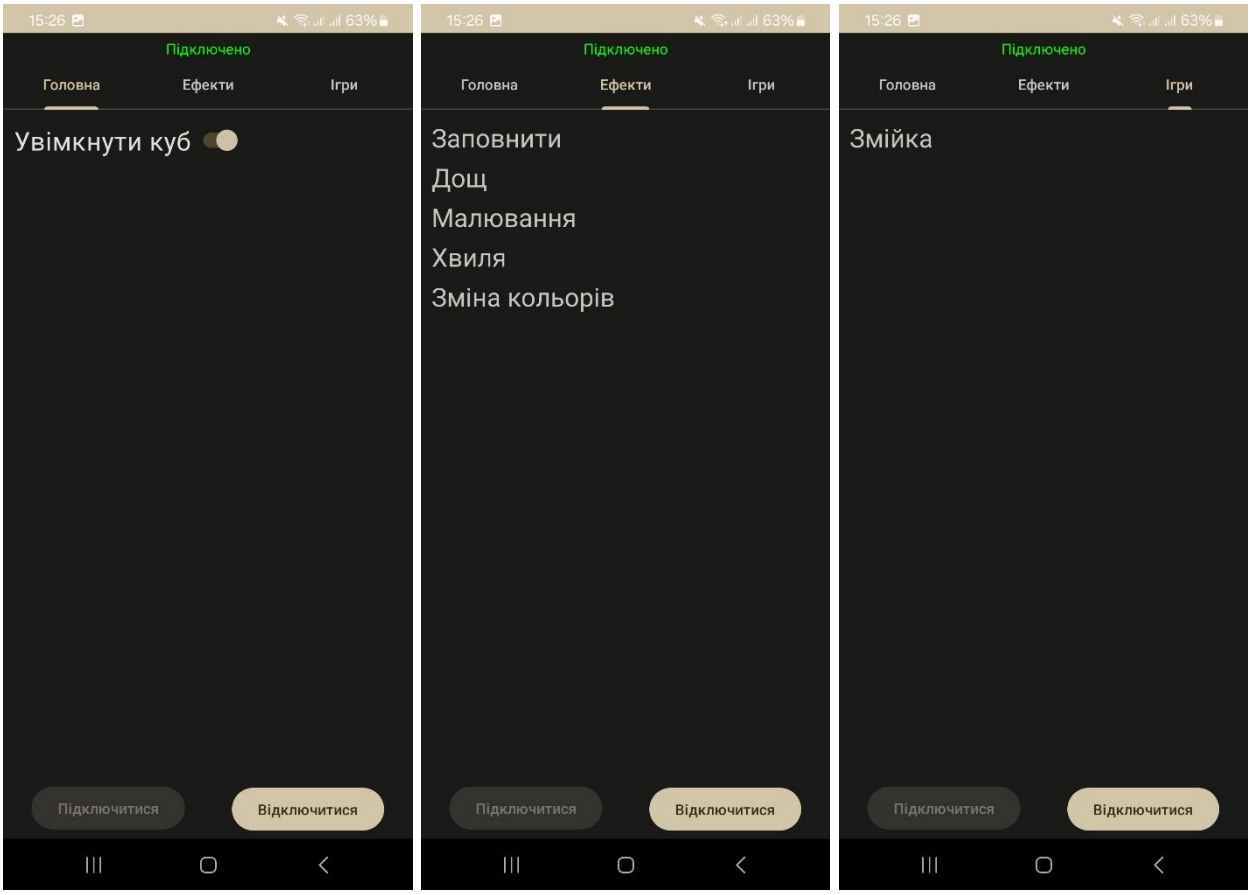
2.2.3.2 АКТИВНІСТЬ MAINACTVIVTY

Головна активність застосунку має назву MainActivity. Ця активність дозволяє користувачу обирати яку команду необхідно надіслати пристрою, а також дозволяє відключитися від пристрою та виводить статус підключення до пристрою. Оскільки команди для пристрою поділені на категорії «Налаштування», «Ефекти» та «Ігри», так само головний екран поділено на три вкладки з командами. Перемикатися між вкладками можна натискаючи на її заголовок або за допомогою жесту перелистування екрану. Кожна така вкладка є окремим фрагментом. Кожен фрагмент виконує обробку натискань на елементи, які він містить, та виконує надсилання відповідних команд пристрою.

При виконанні методу OnCreate встановлюється розмітка інтерфейсу користувача та виконується спроба встановлюється з’єднання з пристроєм. Результат з’єднання виводиться зверху інтерфейсу.

На вкладку «Головна» (рисунок 2.2а) виводиться інтерфейс фрагменту MainFragment. Він містить перемикач типу Switch зі стандартної бібліотеки елементів AndroidSDK, який дозволяє перемикати стан відображення пристрою.

На вкладку «Ефекти» (рисунок 2.2б) виводиться список ефектів. Натискання на елемент буде призводити до виклику нової активності, де можна налаштувати параметри обраного ефекту. Таку саму поведінку має вкладка «Ігри» (рисунок 2.2в).



а б в

Рисунок 2.2 – Інтерфейс головної активності

2.2.3.3 АКТИВНОСТІ ЕФЕКТІВ ТА ІГОР

Активності для кожного з ефектів та ігор мають приблизно однаковий інтерфейс. Для прикладну на рисунку 2.3 наведено інтерфейс налаштування ефекту «Дощ». Інтерфейс складається зі стандартного елементу SeekBar[12], який дозволяє задавати значення швидкості за допомогою переміщення повзунка. Задання кольору відбувається натисканням на круг вибору кольору, який представляє з себе звичайну кнопку зі стандартного набору елементів користувацького інтерфейсу. При її натисканні запускається діалог вибору кольору (рисунок 2.4), де можна задати колір у колірному просторі RGB або HSV.

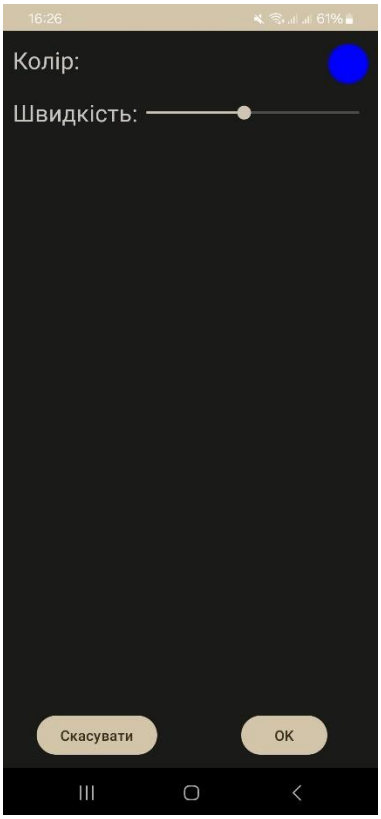


Рисунок 2.3 – Інтерфейс налаштування ефекту «Дощ»

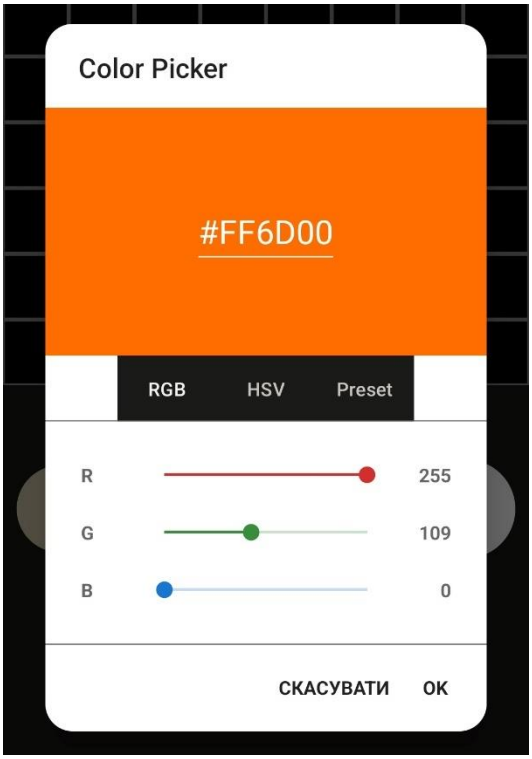


Рисунок 2.4 – Діалог вибору кольору

Окремої уваги заслуговує активність для ефекту малювання (рисунок 2.5). Для цієї активності було розроблено власні елемент інтерфейсу CubeDrawView та DrawFieldView.

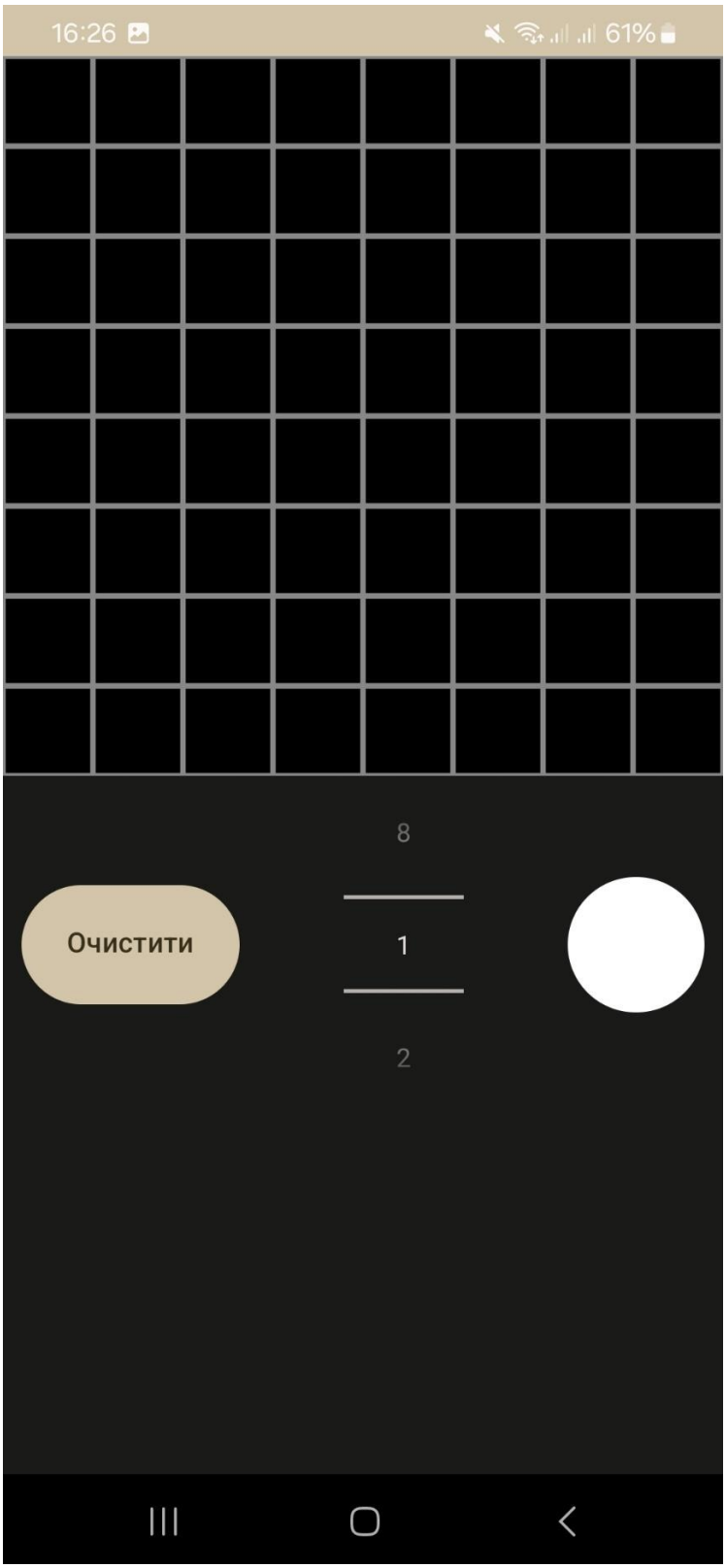


Рисунок 2.5 – Активність для ефекту малювання

Елемент DrawFieldView наслідує клас View[13] з Android SDK. Цей елемент відповідає за відображення сітки для малювання та обробку логіки натискання на клітини.

Елемент CubeDrawView наслідує клас ConstraintLayout[14] з Android SDK. ConstraintLayout – це клас елементу для створення розмітки, у яку додаються інші елементи інтерфейсу та їх обмеження за якими розраховується їх положення на екрані. Цей елемент складається з інших елементів:

- DrawFieldView – елемент з сіткою для малювання;
- Кнопка Button[15] для очищення сітки для малювання;
- NumberPicker[16] – стандартний елемент для обирання числа, яке задає шар на якому відбувається малювання;
- Кнопка Button для обирання кольору малювання.

Таким чином для задання кольору користувач обирає шар малювання за допомогою елементу обирання числа та колір, за допомогою кнопки праворуч, Натискаючи на елементи сітки вони будуть отримувати обраний колір та одразу надсилати його на пристрій для виведення.

2.3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРИСТРОЮ

2.3.1. РОБОТА З КОЛЬОРАМИ

Програмне забезпечення для мікроконтролера написано мовою програмування C++ (додаток В). Головним обмеження при розробці ПЗ стало обмеження на використання оперативної пам’яті. Оскільки зберігання кольору одного світлодіода потребує 3 байти пам’яті (по одному байту на червоний, зелений та синій колірний канал), загальне використання пам’яті лише на зберігання кольорів усього пристрою потрібно $512 \times 3 = 1536$ байт оперативної пам’яті. Таке використання пам’яті є завеликим, бо окрім цих даних необхідно зберігати змінні, які забезпечують роботу пристрою, а також необхідно залишити пам’ять для зберігання тимчасових значень при виконанні обчислень, наприклад, при парсингу даних отриманих з Bluetooth-модуля. Для вирішення цієї проблеми колір кожного з світлодіодів кодується лише двома байтами. Колір кожного світлодіода зберігається у змінній типу uint16_t, у якій можуть зберігатися цілі числа від 0 до 65 535.

Робота з кольором виконується у звичайному 24-бітному форматі. Для зберігання кольору у 16-бітну змінну виконується операція `(((r) & 0b11111000) << 8) | (((g) & 0b11111000) << 3) | (((b) & 0b11111000) >> 2)`, де змінні r, g та b – цілі числа розміром в 1 байт, які відповідають за значення червоного, зеленого та синього кольорів. Таким чином у змінну розміром два байти (16 біт) зберігається послідовно п’ять бітів значення червоного, зеленого та синього кольорів. Для того щоб отримати значення кольорів у 8-бітному форматі виконуються наступні операції, де color – 16-бітна змінна, у якій зберігається колір світлодіода:

- Для червоного – `(color & 0b1111100000000000) >> 11`;
- Для зеленого – `(color & 0b0000011111000000) >> 6`;
- Для синього – `(color & 0b0000000000111110) >> 1`.

Виконання перетворення з 16-бітної змінної дозволяє отримати числа в діапазоні від 0 до 31. Це зроблено з метою обмежити максимальну яскравість світлодіодів, адже для повної яскравості необхідно забезпечити живлення з силою току, що перевищує можливості блоку живлення. Таким чином одночасно виконується і обмеження споживання енергії пристрою, і економія оперативної пам’яті.

Виконання цих операцій дозволяє значно зменшити кількість необхідної оперативної пам’яті. Недоліком цього методу є зменшення кількості можливих кольорів, які може виводити пристрій, а також зменшення швидкості оновлення зображення. На практиці, зменшення кількості кольорів на реальному пристрої не помітно через низьку якість відображення кольорів світлодіодами. Зменшення швидкості оновлення також має достатньо незначний вплив, бо для перетворення значень кольорів використовуються лише побітові логічні операції ТА (&), АБО (|), зсуви ліворуч (<<) та праворуч (>>), які виконуються мікроконтролером дуже швидко.

2.3.2. БІБЛІОТЕКА LEDMATRIX

Для полегшення роботи зі світлодіодами було створено бібліотеку ledMatrix.h. У цьому файлі визначено функції для перетворення кольорів у 16-бітний формат та навпаки, а також визначено клас ledMatrix, який відповідає за роботу з матрицею світлодіодів. Код бібліотеки розміщено у додатку Г.

Мікроконтролер, який встановлено на платформі Arduino, для роботи з виводами використовує 8-бітні регістри[17]. Оскільки вбудовані у стандартний пакет Arduino функції роботи з виводами виконуються дуже повільно, працювати слід зі значеннями цих регістрів напряму. Мікроконтролер для роботи з цифровими виводами використовує порт D (виводи 0 – 7) та порт B (виводи 8 – 13). Для конфігурації напрямку передачі даних виводу необхідно у відповідний регістр необхідного порту встановити

значення біту, який відповідає за необхідний вивід, 0 – якщо вивід має бути вхідним, або 1 – якщо вивід має бути вихідним. Встановлення значення, яке повинно бути на виводі виконується аналогічним чином в іншому регістрі. Наприклад, за конфігурацію виводів порту D відповідає регістр DDRD (The Port D Data Direction Register). Якщо встановити значення цього регістру 11111110_2, то цифрові виводи Arduino 1-7 будуть сконфігуровані як вихідні, а вивід 8 – як вхідний. За вивід даних на виводи порту D відповідає регістр PORTD (The Port D Data Register), тому для задання високого рівня цифрового сигналу на виводах 1, 3 та 7 необхідно присвоїти регістру значення 10001010_2. Робота з виводами порту B відбувається аналогічно.

При створенні об’єкту класу задаються розміри матриці зі світлодіодів та номер виводу до якого підключено світлодіоди. У конструкторі класу викликається метод init(), у якій відбувається конфігурування портів (лістинг 2.1).

```
void init()
{
  pin_mask = digitalPinToBitMask(pin);
  pin_port = portOutputRegister(digitalPinToPort(pin));
  port_ddr = portModeRegister(digitalPinToPort(pin));
  *port_ddr |= pin_mask;
}
```

Лістинг 2.1 – Код методу init()

У змінну pin_mask записується бітова маска, у якій заданому параметром pin, виводу відповідає значення 1, а усі інші біти дорівнюють 0. У змінну pin_port записується поточний стан регістру даних порту, у якому розташован вивід pin. У змінну port_ddr записується стан регістру напряму передачі даних порту, у якому розташован вивід pin. В останньому рядку відбувається зміна стану регістру напряму передачі даних, таким чином щоб змінити стан виводу pin на 1, тобто на вихідний. Наприклад, розглянемо випадок коли значення pin

дорівнює 3. Значення pin_mask буде дорівнювати 00001000_2, оскільки цифровий вивід Arduino №3 відповідає четвертому біту регістру PORTD. Припустимо, що стан усіх виводів порту D дорівнює 0, тоді змінна pin_port отримає значення 00000000_2. Також припустимо, що усі виводи порту D налаштовані як вхідні, тоді змінна port_ddr отримає значення 00000000_2. Наступним кроком відбудиться операція зміни стану регістру напряму передачі порту D на значення: $00000000_2 \mid 00001000_2 = 00001000_2$. Операція логічного АБО (|) дозволяє змінити стан на вихідний для заданого виводу без впливу на усі інші виводи порту.

Для відображення даних на світлодіоди використовується метод show() (лістинг 2.2).

```
void show()
{
  mask_h = pin_mask | *pin_port;
  mask_l = ~pin_mask & *pin_port;
  for (uint8_t i = 0; i < width * height; i++)
    send(leds[i]);
}
```

Лістинг 2.2 – Код методу show()

Спочатку метод show() створює маску для запису у регістр даних значень високого та низького цифрового сигналу. Наприклад, використаємо значення для третього виводу отримані вище. У змінну mask_h буде записано: $00001000_2 \mid 01100000_2 = 01101000_2$, де 01100000_2 – деякий поточний стан регістру даних. Таким чином, у ця змінна буде мати таке саме значення як і регістр даних, окрім біту, що відповідає третьому виводу – він буде мати значення високого логічного сигналу (1). Так само у змінну mask_l буде записане значення $\sim 00001000_2 \,\&\, 01100000_2 = 11110111_2 \,\&\, 01100000_2 = 01100000_2$. Таким чином ми отримали таке саме значення яке записано в регістрі даних, крім необхідного виводу – він має значення 0. Далі у циклі

почергово для усіх світлодіодів викликається метод send(), який посилає на вивід мікроконтролера дані для відображення кольору поточного світлодіода.

Для виведення даних одного світлодіода використовується метод send(uint16_t color) (лістинг 2.3).

```
void send(uint16_t color)
{
  uint8_t data[3];
  data[0] = getR(color);
  data[1] = getG(color);
  data[2] = getB(color);
  cli();
  for (uint8_t i = 0; i < 3; i++)
    sendValue(data[i]);
  sei();
}
```

Лістинг 2.3 – Код методу send()

Цей метод перетворює значення змінної кольору з 16-бітного значення на 24-бітне. Далі функція cli()[18] вимикає переривання мікроконтролера на час виведення даних для світлодіоду, оскільки переривання може створити затримку у виведенні даних, що призведе до помилок відображення. Далі викликається метод sendValue() для кожного з кольорових каналів, який виводить дані для цього кольорового каналу на вивід мікроконтролера. Наприкінці, за допомогою функції sei() вмикається виконання переривань. Код методу sendValue() наведено у лістингу 2.4:

```
void sendValue(byte data)
{
  asm volatile(
      "LDI r24, 8         \n\t"
      "LOOP_START:        \n\t"
      "ST X, %[SET_H]     \n\t"
```

```
        "SBRS %[DATA], 7    \n\t"
        "ST X, %[SET_L]     \n\t"
        "LSL  %[DATA]       \n\t"
        "LDI r25, 4         \n\t"
        "DELAY_LOOP:        \n\t"
        "DEC r25            \n\t"
        "BRNE DELAY_LOOP    \n\t"
        "ST X, %[SET_L]     \n\t"
        "DEC r24            \n\t"
        "BRNE LOOP_START    \n\t"
        :
        : [DATA] "r"(data),
        [SET_H] "r"(mask_h),
        [SET_L] "r"(mask_l),
        "x"(pin_port)
        : "r24", "r25");
}
```

Лістинг 2.4 – Код методу sendValue()

У коді цього методу використано вставку мовою асемблера[19] для точного контролю за часом виведення даних на вивід мікроконтролера. Це важливо, адже кодування значень «0» та «1» для світлодіода відбувається завдяки різниці між тривалістю значень високого та низького логічного сигналу. У таблиці 4 наведено значення проміжків часу, необхідних для кодування сигналів «0» та «1», на рисунку 2.6 наведено схему отримання цих сигналів[5].

Таблиця 4 – Проміжки часу подачі сигналу

Назва	Опис	Значення	Похибка
T0H	«0», час високого сигналу	0,295μs	±0,05μs
T1H	«1», час високого сигналу	0,595μs	±0,05μs
T0L	«0», час низького сигналу	0,595μs	±0,05μs

T1L	«1», час низького сигналу	0,295μs	±0,05μs
TReset	Код скидання	≥ 80μs	

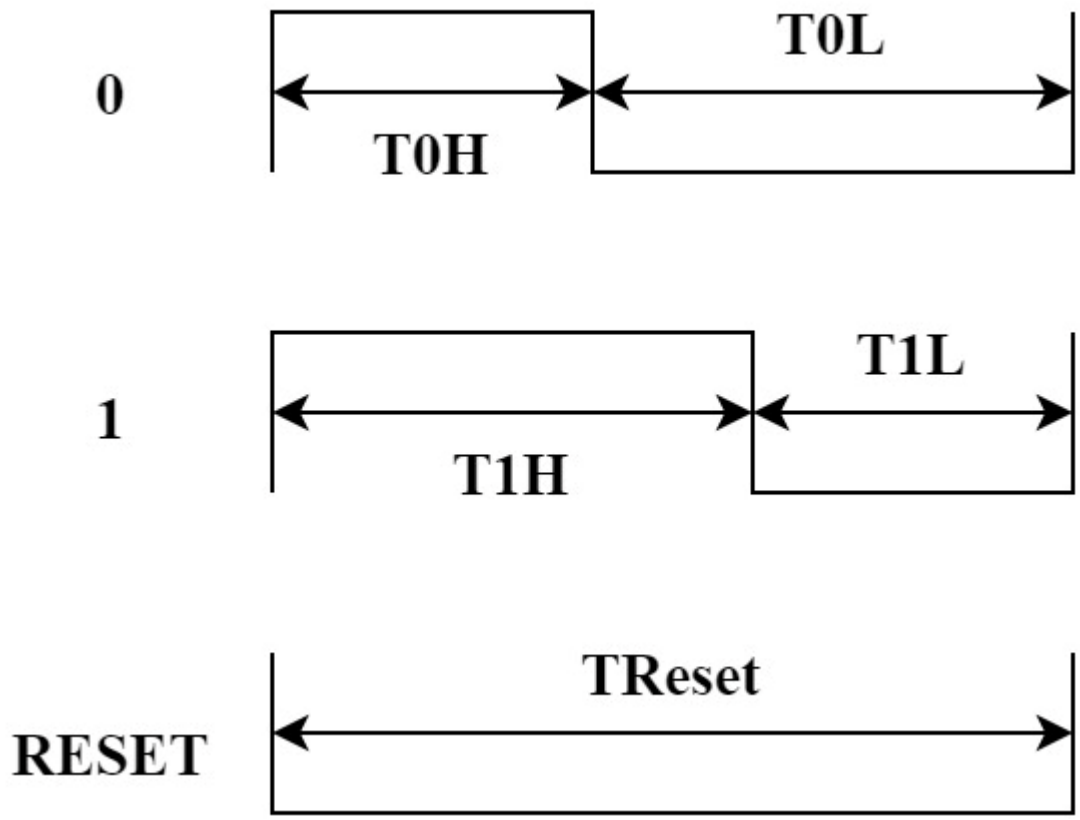


Рисунок 2.6 – Схема кодування сигналів світлодіодів

Розберемо код асемблерної вставки. Командою «LDI r24, 8» виконується запис числа 8 у регістр загального призначення r24. Це число є лічильником циклу виведення даних. У наступному рядку задається мітка переходу «LOOP_START» до початку циклу. Команда «ST X, %[SET_H]» виконує запис сигналу «1» у регістр X, при цьому регістр X використовується для запису даних у порт відповідного виводу. Далі виконується перевірка чому дорівнює останній біт даних, які потрібно вивести за допомогою команди «SBRS %[DATA], 7». Якщо останній біт дорівнює «1», наступний рядок не виконується, в іншому випадку виводиться низький сигнал. Команда «LSL %[DATA]» виконує лівий бітовий зсув даних. Наступні 4 рядка є циклом, мета якого створити затримку перед виведенням низького рівня сигналу. Команда

«DEC» виконує декремент вказаного регістру, а «BRNE» – перехід до вказаної мітки. В останніх рядках виконується декремент лічильника основного циклу та перехід його початку. Після символу «:» вказуються вихідні змінні, яких у нас немає, після наступного символу «:» вказані вхідні змінні. Наприкінці, вказуються регістри які використовуються у асемблерному коді.

2.3.3. АЛГОРИТМ РОБОТИ ПРОГРАМИ

Для задання параметрів, таких як коди Bluetooth-команд, швидкість комунікації мікроконтролера з Bluetooth-модулем, номери контактів до яких підключені шари зі світлодіодів та ін., використовується директива #define, наприклад, рядок `#define BLUETOOTH_SPEED 115200` задає швидкість обміну даними по шині UART у 115200 бод/с. При використанні директиви #define можливо одноразово визначити константи та використовувати їх у коді. При компіляції програми компілятор автоматично підставляє значення директиви #define там, де вони використовуються, завдяки чому ці константи не займають місця в пам’яті.

На початку виконання програми задаються наступні змінні:

```
bool state = true;
uint8_t currentEffect = EFFECT_NONE;
uint32_t timer;
```

Змінна state відповідає за перемикання відображення пристрою, currentEffect – зберігає поточний ефект, а timer – зберігає значення часу, яке пройшло від початку роботи мікроконтролера, ця змінна необхідна для коригування швидкості оновлення ефектів.

Основний цикл роботи програми наведено у лістингу 2.5.

```
for(;;){
    if(Serial.available()){
      parseCommand();
```

```
    }

    switch(currentEffect){
      case EFFECT_RAINBOW:
      {
        rainbow();
        break;
      }
      case EFFECT_SINUS:
      {
        sinusFill();
        break;
      }
      case EFFECT_RAIN:
      {
        rain();
        break;
      }
      case GAME_SNAKE:
      {
        snake();
        break;
      }
    }
  }
```

Лістинг 2.5 – Код основного циклу програми

На початку циклу за допомогою команди Serial.available()[20] перевіряється наявність нових команд, та якщо вони є викликається функція parseCommand(). В іншому випадку в залежності від обраного ефекту виконується функція оновлення відповідного ефекту. Замість оператора switch також можна використовувати декілька операторів if, проте це не бажано, адже при використанні оператора if необхідно виконати декілька операцій

порівняння. При використанні операції switch в залежності від значення змінної currentEffect одразу виконується необхідний блок коду.

2.3.4. ФУНКЦІЯ PARSECOMMAND

Функція parseCommand() відповідає за парсинг команд від Bluetooth-модуля. Початок цієї функції наведено у лістингу 2.6:

```
char buf[44];
uint8_t amount = Serial.readBytesUntil(EOL, buf, 44);
buf[amount] = 0;
uint8_t data[12];
uint8_t count = 0;
char* offset = buf;
while (true)
{
  data[count++] = atoi(offset);
  offset = strchr(offset, DELIMETER);
  if(offset) offset++;
  else break;
}
```

Лістинг 2.6 – Початок функції parseCommand()

Спочатку задається масив buf, у якому будуть зберігатися символи від Bluetooth-модуля, які необхідно інтерпретувати. На наступному кроці функція Serial.readBytesUntil()[21] записує у масив buf отриману команду та повертає кількість зчитаних символів у змінну amount. Далі створюється масив data у який будуть зберігатися числові значення отриманої команди. В залежності від значень у масиві data виконуються дії, необхідні для виконання цієї команди.

2.3.5. ФУНКЦІЯ ЕЕФЕКТУ «ДОЩ»

Принцип роботи усіх функцій ефектів приблизно однакове. Для прикладу розглянемо код функції ефекту «дощ» (лістинг 2.7).

```
void rain() {
  if (loading) {
    currentEffect = EFFECT_RAIN;
    clear();
    loading = false;
    timer = millis();
  }
  if (millis() - timer >= speed) {
    timer = millis();
    for(int i = 0; i < COLUMNS*ROWS; i++){
      layer1.leds[i] = layer2.leds[i];
      layer2.leds[i] = layer3.leds[i];
      layer3.leds[i] = layer4.leds[i];
      layer4.leds[i] = layer5.leds[i];
      layer5.leds[i] = layer6.leds[i];
      layer6.leds[i] = layer7.leds[i];
      layer7.leds[i] = layer8.leds[i];
      layer8.leds[i] = 0;
    }

    uint8_t numDrops = random(0, 5);
    for (uint8_t i = 0; i < numDrops; i++) {
      setPixel(7, random(0, 8), random(0, 8), effectColor);
    }
    draw();
  }
}
```

Лістинг 2.7 – Код функції ефекту «дощ»

На початку перевіряється значення глобальної змінної логічного типу loading. Якщо відбулась зміна ефекту, ця зміна буде мати значення true, таким чином вказуючи на необхідність виконання підготовки ефекту. Підготовка ефекту «дощ» полягає у заданні глобальної функції currentEffect значення, відповідного до цього ефекту, очищенню зображення за допомогою функції clear(). Також відбувається оновлення значення часу у змінній timer.

Далі функція rain() буде викликатися з основного циклу програми. Якщо різниця між поточним часом та збереженим у змінній timer буде більше ніж задана швидкість оновлення, відбувається зміщення усього зображення на один шар донизу, а на верхньому шарі випадково задається декілька нових точок. Наприкінці за допомогою функції draw() виконується оновлення зображення.

3. ПРАКТИЧНЕ ЗАСТОСУВАННЯ ПРИСТРОЮ

У сучасному світі стрімкий розвиток технологій постійно змінює підходи до візуалізації даних та взаємодії з ними. Зростання обсягів інформації та потреба у її швидкому та ефективному сприйнятті висувають нові вимоги до методів представлення даних. Візуалізація даних відіграє ключову роль у багатьох сферах життя, адже вона дозволяє не лише ефективно доносити інформацію до користувачів, але й підвищувати їх аналітичні можливості та сприяти кращому розумінню складних концепцій.

Тривимірна візуалізація є важливим кроком у розвитку методів представлення даних, оскільки вона дозволяє краще уявити складні об’єкти та процеси, що не завжди можливо при використанні двовимірних засобів. У порівнянні з текстовими, числовими або двовимірними графічними даними, 3D візуалізація забезпечує глибше розуміння інформації та сприяє більш інтерактивному досвіду користувачів.

Попри те, що сучасні пристрої для візуалізації даних, такі як монітори, телевізори та екрани, мають двовимірну природу, існує постійний попит на тривимірні методи відображення. Потреба у відображення об’єктів у тривимірному просторі реалізувалась у розвитку технологій віртуальної та доповненої реальності. Сучасні пристрої віртуальної та доповненої реальності мають значні недоліки. По-перше вони мають високу вартість, що значно обмежує їх розповсюдження. По-друге, такі пристрої є більшою мірою індивідуальними. У разі необхідності демонстрації якогось тривимірного зображення декільком людям одночасно, наприклад, коли інженер хоче порадитись з колегами стосовно нового механізму, необхідно забезпечити пристроєм для відтворення зображення, таким як шолом віртуальної реальності, кожного з людей. До того ж, додається необхідність синхронізації дій усіх користувачів в якомусь спільному віртуальному просторі.

Використання тривимірних дисплеїв дозволяє уникнути багатьох проблем, пов’язаних з використанням технології віртуальної реальності. Один такий дисплей дозволяє демонструвати зображення одночасно для багатьох користувачів. Також для його функціонування не потрібна велика кількість дорогих високоточних датчиків для відстеження переміщень користувача.

Реалізація дисплею, яка наведена у цій роботі має значні обмеження, перш за все – низька роздільна здатність. Вирішення цієї проблеми потенційно можливо з використанням вже існуючих технологій. Останні роки багато світових виробників електроніки демонстрували тонкі та прозорі дисплеї. Складання багатьох таких дисплеїв один над одним дозволить досягти високої роздільної здатності, при цьому провідники не заважатимуть відображенню.

3.1. ВИКОРИСТАННЯ 3D ДИСПЛЕЮ В МЕДИЦИНІ

Застосування тривимірних дисплеїв у медицині має великий потенціал для покращення якості діагностики, планування лікування, навчання медичного персоналу та проведення медичних досліджень.

Одним із основних потенційних застосувань тривимірних дисплеїв у медицині є візуалізація анатомічних структур. Завдяки тривимірним зображенням лікарі можуть детально розглядати внутрішні органи, судини, кістки та інші анатомічні елементи, що дозволяє більш точно діагностувати захворювання та планувати лікування. Медичні сканери, такі як МРТ та КТ, в результаті своєї роботи дають велику кількість плоских зображень. Виведення цих зображень на тривимірному дисплеї дозволить відновити зображення в тривимірному вигляді, завдяки чому лікар зможе детальніше розуміти внутрішню будову тканин.

Іншим застосуванням тривимірних дисплеїв планування хірургічних операцій. Дослідження показують, що використання віртуальної реальності дозволяє хірургам покращити просторове розуміння окремих анатомічних

структур і краще ідентифікувати анатомічних варіанти[22]. Тривимірна візуалізація дозволяє створювати тривимірні моделі пацієнтів, на яких хірурги можуть провести віртуальну операцію перед її виконанням у реальності. Це допомагає ідентифікувати потенційні проблеми та визначити найкращий підхід до операції, зменшуючи ризики.

Тривимірні дисплеї можуть використовуватися при навчанні медичного персоналу. Вони надають можливість студентам-медикам та лікарям-початківцям вивчати та розглядати анатомічні структури тривимірному вигляді.

Також такі дисплеї можуть допомогти у протезуванні та реконструктивній хірургії. За допомогою 3D дисплею можливо створювати індивідуальні моделі протезів та імплантатів. Хірург може детально розглянути як розроблений протез буде розміщуватися у тілі пацієнта та вносити відповідні коригування. Завдяки цьому можливо виготовляти імплантати, які ідеально підходять пацієнту.

Ще одним застосуванням пристрою може бути візуалізація процесів взаємодії клітин з різними речовинами, вірусами, бактеріями та іншими клітинами, що дозволить дослідникам вивчати складні біологічні процеси на молекулярному рівні у тривимірному просторі.

3.2. ВИКОРИСТАННЯ 3D ДИСПЛЕЮ У ПРОМИСЛОВОСТІ ТА ВИРОБНИЦТВІ

У промисловості та виробництві тривимірні дисплеї можуть забезпечити візуалізацію складних об'єктів і процесів, сприяючи інноваціям у проектуванні, виробництві та управлінні.

Завдяки тривимірній візуалізації інженери можуть створювати детальні моделі продуктів і компонентів, оцінюючи їх з різних кутів і в різних умовах,

що значно покращить процеси проектування та моделювання. Використання тривимірних дисплеїв дозволить виявляти потенційні проблеми на ранніх стадіях і вносити необхідні корективи, знижуючи ризик помилок і витрати на їх виправлення.

Також тривимірні дисплеї можуть застосовуються для контролю якості та управління. Їх використання дозволить операторам візуалізувати внутрішню структуру виробів, виявляючи дефекти та невідповідності ще до завершення виробництва. Наприклад, за допомогою 3D-сканування можна створювати точні моделі виробів і порівнювати їх з еталонними, виявляючи найменші відхилення.

У галузі технічного обслуговування та ремонту тривимірні дисплеї можуть покращити планування ремонтних робіт завдяки детальній візуалізації внутрішні компоненти обладнання, яке потребує обслуговування. Наочність тривимірної моделі обладнання також дозволить проводити інструктажі та навчання для нового персоналу більш ефективно.

Потенційно, такі пристрої можуть бути затребувані у системах управління виробництвом для візуалізації виробничих ліній, що дозволить точніше відстежувати стан обладнання та контролювати процеси в режимі реального часу. Це дозволить оптимізувати виробничі процеси та підвисити продуктивність.

3.3. ВИКОРИСТАННЯ 3D ДИСПЛЕЮ ДЛЯ ОСВІТИ

3D дисплеї мають значний потенціал для трансформації освітнього процесу, забезпечуючи більш інтерактивні та захоплюючі методи навчання. Вони сприяють кращому розумінню складних наукових концепцій, розширюють можливості візуалізації та створюють нові способи взаємодії з навчальним матеріалом.

Тривимірні дисплеї дозволяють відтворювати навчальні матеріали у тривимірному форматі, що робить їх більш наочними та зрозумілими. Наприклад, у вивченні біології учні можуть досліджувати тривимірні моделі клітин, органів та систем організму, що дозволяє краще зрозуміти їхню структуру та функції. Так само такі дисплеї можуть бути корисними для моделювання фізичних явищ, таких як електромагнітні поля, хвильові процеси або рух планет. Це дозволяє студентам краще бачити взаємодію різних елементів та досліджувати їх вплив один на одного.

Тривимірні дисплеї можуть використовуватись для проведення інтерактивних лабораторних робіт. Наприклад, вивчення хімії може включати тривимірне моделювання хімічних реакцій, де студенти можуть бачити, як взаємодіють молекули, та спостерігати за результатами реакцій. Це не тільки підвищує розуміння предмета, але й робить навчання більш цікавим та залучаючим.

У вивченні географії та історії тривимірні дисплеї також можуть знайти своє застосування. Вони дозволяють створювати тривимірні карти рельєфу, які допомагають краще зрозуміти географічні особливості. При вивченні історії можливо візуалізувати різноманітні історичні предмети, такі як одяг, прикраси, зброю.

Таким чином використання тривимірних дисплеїв у освіті відкриває нові можливості для покращення якості навчального процесу. Вони сприяють кращому розумінню матеріалу, роблять навчання більш інтерактивним та захоплюючим.

3.4. ВИКОРИСТАННЯ 3D ДИСПЛЕЮ В СФЕРІ РОЗВАГ

Тривимірні дисплеї можуть застосовуватися для створення нового досвіду у відеоіграх. Додавання третього виміру зображення дозволяє змінити підхід до розробки ігор. Наприклад, в розробленому пристрої, реалізовано

класичну гру «Змійка», але з додаванням третього виміру. На рисунку 3.1 можна побачити цю гру.

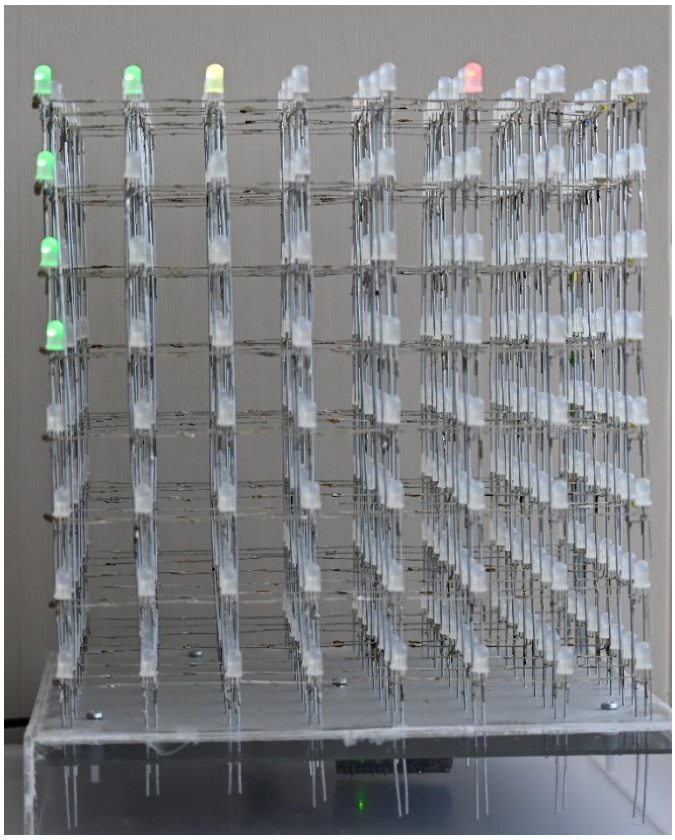


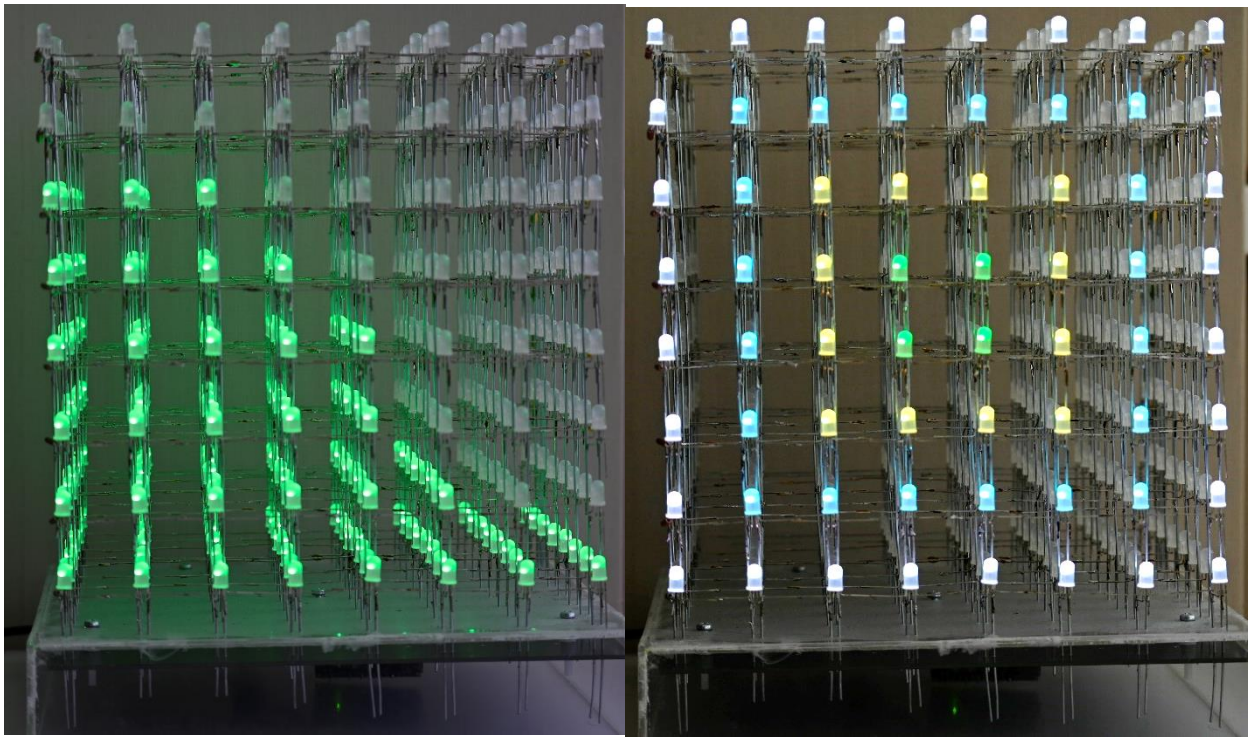
Рисунок 3.1 – Гра «Змійка»

Тривимірні дисплеї можуть знайти своє застосування у кінематографі. Кінофільми з використанням технологій для створення ефекту занурення вже давно набули популярності. Проте такі фільми використовують технологію стереографії для створення ефекту 3D, тому це зображення лише ілюзія тривимірного зображення. Однак створення тривимірних фільмів може бути складною задачею. Як для створення стереографічних фільмів було необхідно розробити нові засоби для зйомки, так і для зйомки тривимірних фільмів необхідно розробити нове обладнання.

Можливе використання тривимірних дисплеїв для проведенні концертів та інших розважальних заходів. Вже проводиться концерти віртуальних артистів, де ефект глибини створюється за допомогою стереографії.

Використання запропонованого пристрою дозволе глядачам насолоджуватися шоу з ефектом присутності, незалежно від їх куту зору.

Ще одним застосуванням пристрою, яке вже реалізовано на прототипі, може бути використання його для декоративних цілей. Пристрій може використовуватися як пристрій для освітлення. У світі розповсюджені «розумні» світлодіодні лампи, які можуть світитися різними кольорами. Розроблений пристрій також реалізує цю функцію. Приклади роботи світлових реалізованих ефектів можна побачити на рисунках 3.2а та 3.2б.



а б

Рисунок 3.2 а – Ефект «Хвиля», б – Ефект «Малювання»

ВИСНОВКИ

За результатами виконаної роботи було досягнуто поставлених цілей та вирішено ряд задач, що дозволило зробити кілька важливих висновків. По-перше, було успішно розроблено апаратну та програмну частини пристрою для відображення тривимірної інформації. Основою цього пристрою став тривимірний масив зі світлодіодів, які формують куб розміром 8x8x8 точок. Це дозволило візуалізувати різноманітні об'єкти у тривимірному просторі. Тестування пристрою показало, що він здатний відображати тривимірні зображення з достатньою якістю для базових застосувань, хоча й існують певні обмеження, зокрема низька роздільна здатність.

Можливості застосування тривимірних дисплеїв у різних галузях виявилися значними. В медицині тривимірні дисплеї можуть використовуватися для візуалізації тканин та органів, планування хірургічних втручань та навчанні медиків. В освітній сфері використання тривимірних дисплеїв може значно покращити освітній процес, забезпечуючи більш інтерактивне та захоплююче навчання. Це сприяє кращому розумінню складних наукових концепцій та створює нові способи взаємодії з навчальним матеріалом. У промисловості та виробництві тривимірні дисплеї можуть забезпечити візуалізацію складних об'єктів і процесів, сприяючи інноваціям у проектуванні, виробництві та управлінні. Вони дозволяють інженерам створювати детальні моделі продуктів і компонентів, оцінюючи їх з різних кутів та в різних умовах, що значно покращує процеси проектування та моделювання. В індустрії розваг тривимірних дисплеї відкривають нові можливості, надаючи користувачам більш захоплюючі враження. Це може бути застосовано у кінотеатрах, іграх, тематичних парках та інших розважальних заходах.

Рекомендації щодо практичного застосування розробленого пристрою включають підвищення його роздільної здатності та яскравості, що дозволить покращити якість відображуваних тривимірних зображень. Можливим

рішенням цих проблем може стати створення масивів з багатьох двомірних прозорих світлодіодних дисплеїв, розташованих один над одним для створення тривимірного дисплею.

Подальший розвиток та дослідження в області тривимірних дисплеїв можуть включати розробку методів для зменшення вартості виробництва, що зробить їх більш доступними для широкого кола користувачів. Наступним кроком може бути вивчення можливостей створення інтерактивних тривимірних дисплеїв, які дозволять користувачам взаємодіяти з відображуваними об'єктами у реальному часі.

За результатами даної роботи опубліковано тези на XI Всеукраїнській науково-практичної конференції здобувачів вищої освіти та молодих вчених з автоматичного управління[23], тези на XXI Всеукраїнській конференції студентів і молодих науковців «Інформатика, інформаційні системи та технології»[24] та тези на XII Міжнародній науково-технічній інтернет-конференції «Проблеми та перспективи розвитку автомобільного транспорту»[25].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fairy Lights in Femtoseconds [Електронний ресурс] / Yoichi Ochiai [та ін.] // ACM Transactions on Graphics. – 2016. – Т. 35, № 2. – С. 1–14. – Режим доступу: https://doi.org/10.1145/2850414.
2. 3D LED CUBE [Електронний ресурс] / Varun Tirumalasetty [та ін.] // International Research Journal of Engineering and Technology (IRJET). – 2020. – Т. 7, № 4. – Режим доступу: https://www.irjet.net/archives/V7/i4/IRJET-V7I4741.pdf
3. ATmega328P. 8-bit AVR Microcontroller Programmable Flash. Atmel Corporation. / Рев.: 7810D–AVR–01/15. 2015. [Електронний ресурс]. Режим доступу: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf.
4. JDY-33 Bluetooth Module Manual [Електронний ресурс] // FCC Report - Databases of FCC Filings. – Режим доступу: https://fcc.report/FCC-ID/2AXM8-JDY-33/4936885.pdf.
5. YF923-F5-F8-LED. iPixel LED LIGHT Corporation. / [Електронний ресурс]. Режим доступу: https://cdn.sparkfun.com/assets/a/b/1/e/1/DS-12877-LED_-_RGB_Addressable__PTH__8mm_Diffused__5_Pack_.pdf.
6. Android Studio & App Tools [Електронний ресурс] // Android Developers. – Режим доступу: https://developer.android.com/studio.
7. Activity [Електронний ресурс] // Android Developers. – Режим доступу: https://developer.android.com/reference/android/app/Activity.
8. App manifest overview [Електронний ресурс] // Android Developers. – Режим доступу: https://developer.android.com/guide/topics/manifest/manifest-intro.
9. App resources overview [Електронний ресурс] // Android Developers. – Режим доступу: https://developer.android.com/guide/topics/resources/providing-resources.

10. Fragments [Електронний ресурс] // Android Developers. – Режим доступу: https://developer.android.com/guide/fragments.
11. SharedPreferences [Електронний ресурс] // Android Developers. – Режим доступу: https://developer.android.com/reference/kotlin/android/content/SharedPreferences.
12. SeekBar [Електронний ресурс] // Android Developers. – Режим доступу: https://developer.android.com/reference/android/widget/SeekBar.
13. Create custom view components [Електронний ресурс] // Android Developers. – Режим доступу: https://developer.android.com/develop/ui/views/layout/custom-views/custom-components.
14. Build a responsive UI with ConstraintLayout [Електронний ресурс] // Android Developers. – Режим доступу: https://developer.android.com/develop/ui/views/layout/constraint-layout.
15. Add buttons to your app [Електронний ресурс] // Android Developers. – Режим доступу: https://developer.android.com/develop/ui/views/components/button.
16. NumberPicker [Електронний ресурс] // Android Developers. – Режим доступу: https://developer.android.com/reference/android/widget/NumberPicker.
17. PortManipulation [Електронний ресурс] // Arduino Docs. – Режим доступу: https://docs.arduino.cc/retired/hacking/software/PortManipulation/.
18. Khaled M. Arduino noInterrupts, interrupts, sei() & cli() Functions [Електронний ресурс] / Magdy Khaled // DeepBlueMbedded. – Режим доступу: https://deepbluembedded.com/arduino-nointerrupts-sei-cli-functions/.
19. AVR Microcontrollers. AVR Instruction Set Manual. / [Електронний ресурс]. Режим доступу:

http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf.

20. Serial.available() [Електронний ресурс] // Arduino Reference. – Режим доступу: https://www.arduino.cc/reference/tr/language/functions/communication/serial/available/.
21. Serial.readBytesUntil() [Електронний ресурс] // Arduino Reference. – Режим доступу: https://www.arduino.cc/reference/tr/language/functions/communication/serial/readbytesuntil/.
22. Virtual Reality for Surgical Planning – Evaluation Based on Two Liver Tumor Resections [Електронний ресурс] / Anke V. Reinschluessel [та ін.] // Frontiers in Surgery. – 2022. – Т. 9. – Режим доступу: https://doi.org/10.3389/fsurg.2022.821060.
23. Антіпов М. М. Розробка пристрою для відображення тривимірних даних [Електронний ресурс] / Антіпов М. М., Шугайло Ю. Б. // Матеріали XI Всеукраїнської науково-практичної конференції здобувачів вищої освіти та молодих вчених з автоматичного управління : 12 квіт. 2024 р. – Херсон-Хмельницький, 2024. – С. 81–84. – Режим доступу: https://drive.google.com/file/d/1VUQL4MemcRa8SNMcvUfJScj88QzB9EUS/view.
24. Антіпов М. М. Пристрій для відображення тривимірних даних. / Антіпов М. М., Шугайло Ю. Б. // Матеріали XXI Всеукраїнській конференції студентів і молодих науковців «Інформатика, інформаційні системи та технології» : 26 квіт. 2024 р. – Одеса, 2024. – С. 71–73.
25. Антіпов М. М. Розробка пристрою для відображення тривимірних даних [Електронний ресурс] / Антіпов М. М., Шугайло Ю. Б. // тези доповідей двадцять другої Міжнародної науково-технічної інтернет-

конференції «Проблеми та перспективи розвитку автомобільного транспорту»: 16-18 квіт. 2024 р. – Вінниця– С. 33–36. – Режим доступу: https://atmconf.vntu.edu.ua/materialy2024.pdf.

ДОДАТОК А

Код мобільного застосунку

Код класу MainActivity.java:

```
package com.mike.ledcube;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.lifecycle.Lifecycle;
import androidx.viewpager2.adapter.FragmentStateAdapter;
import androidx.viewpager2.widget.ViewPager2;

import com.google.android.material.tabs.TabLayout;
import com.google.android.material.tabs.TabLayoutMediator;

public class MainActivity extends AppCompatActivity {
    private TextView connectionTextView;
    private Button connectButton;
    public static BluetoothConnectionHelper BLHelper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        ViewPager2 viewPager =
findViewById(R.id.main_viewpager);
        viewPager.setAdapter(new
MainViewPagerAdapter(getSupportFragmentManager(),
getLifecycle()));
        TabLayout tabLayout = findViewById(R.id.main_tablayout);
        new TabLayoutMediator(tabLayout, viewPager, (tab,
position) -> {
            switch (position) {
                case 0:
                    tab.setText(getString(R.string.main));
                    break;
                case 1:
                    tab.setText(getString(R.string.effects));
                    break;
                case 2:
                    tab.setText(getString(R.string.games));
                    break;
            }
        }).attach();


        connectionTextView = findViewById(R.id.statusTextView);
        connectButton = findViewById(R.id.connect_button);
        connectButton.setOnClickListener((vw) ->
BLHelper.connect());
        Button disconnectButton =
findViewById(R.id.disconnect_button);
        disconnectButton.setOnClickListener((vw) -> {
            SharedPreferences sharedPref =
getSharedPreferences(getString(R.string.pref_file),
Context.MODE_PRIVATE);
            SharedPreferences.Editor editor = sharedPref.edit();

editor.putString(getString(R.string.bluetooth_device_address),
getString(R.string.string_null));
            editor.apply();
```

```
            Intent intent = new Intent(this,
ChooseDeviceActivity.class);
            startActivity(intent);
            finish();
        });


        BLHelper = new BluetoothConnectionHelper(this,
getIntent().getStringExtra("device_mac"));


        BLHelper.getConnectionStatus().observe(this,
this::onConnectionStatus);
        BLHelper.connect();


    }


    @Override
    protected void onDestroy() {
        super.onDestroy();
        BLHelper.disconnect();
    }


    private void
onConnectionStatus(BluetoothConnectionHelper.ConnectionStatus
connectionStatus) {
        switch (connectionStatus) {
            case CONNECTED:

connectionTextView.setText(R.string.status_connected);

connectionTextView.setTextColor(getColor(R.color.green));
                connectButton.setEnabled(false);
                break;


            case CONNECTING:

connectionTextView.setText(R.string.status_connecting);
```

```
connectionTextView.setTextColor(getColor(R.color.yellow));
                break;

            case DISCONNECTED:

connectionTextView.setText(R.string.status_disconnected);

connectionTextView.setTextColor(getColor(R.color.red));
                connectButton.setEnabled(true);
                break;
        }
    }

    public static class MainViewPagerAdapter extends
FragmentStateAdapter {
        private final Fragment[] fragments = new Fragment[3];

        public MainViewPagerAdapter(@NonNull FragmentManager
fragmentManager, @NonNull Lifecycle lifecycle) {
            super(fragmentManager, lifecycle);
            fragments[0] = new MainFragment();
            fragments[1] = new EffectsFragment();
            fragments[2] = new GamesFragment();
        }

        @Override
        public int getItemCount() {
            return fragments.length;
        }

        @NonNull
        @Override
        public Fragment createFragment(int position) {
            return fragments[position];
```

```
        }
    }
}
```

Код класу MainFragment.java:

```
package com.mike.ledcube;


import static
com.mike.ledcube.CubeCommunication.PreferencesCommands.PREFERENC
ES;
import static
com.mike.ledcube.CubeCommunication.PreferencesCommands.PREFERENC
E_STATE;
import static com.mike.ledcube.MainActivity.BLHelper;


import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CompoundButton;
import android.widget.Toast;


import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;


import
com.google.android.material.switchmaterial.SwitchMaterial;
import com.mike.ledcube.CubeCommunication.BluetoothCommands;


import java.util.Objects;


public class MainFragment extends Fragment implements
CompoundButton.OnCheckedChangeListener {


    private SwitchMaterial onOffSwitch;
```

```
    public MainFragment() {
        // Required empty public constructor
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_main,
container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable
Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        onOffSwitch =
requireView().findViewById(R.id.on_off_switch);
        onOffSwitch.setOnCheckedChangeListener(this);



BLHelper.getConnectionStatus().observe(getViewLifecycleOwner(),
this::onConnectionStatus);
        BLHelper.getMessages().observe(getViewLifecycleOwner(),
this::onMessage);
    }

    private void GetAllPreferences() {
        BLHelper.send(BluetoothCommands.getStateCommand());
```

```
    }


    private void onMessage(Event<char[]> message) {
        if (message.hasBeenHandled()) return;
        String[] msg =
BluetoothCommands.parseCommand(Objects.requireNonNull(message.ge
tContentIfNotHandled()));


        if (msg.length != 3) return;
        //pref
        if (msg[0].equals(String.valueOf(PREFERENCES))) {
            //state
            if (msg[1].equals(String.valueOf(PREFERENCE_STATE)))
{
                onOffSwitch.setChecked(msg[2].equals("1"));
            }
        }
    }


    private void
onConnectionStatus(BluetoothConnectionHelper.ConnectionStatus
connectionStatus) {
        switch (connectionStatus) {
            case CONNECTED:
                GetAllPreferences();
                break;
            case CONNECTING:
                break;
            case DISCONNECTED:
                break;
        }


    }


    @Override
```

```
    public void onCheckedChanged(CompoundButton compoundButton,
boolean b) {
        if (!BLHelper.send(b ? BluetoothCommands.onCommand() :
BluetoothCommands.offCommand())) {
            Toast.makeText(requireContext(),
R.string.error_not_send, Toast.LENGTH_LONG).show();
        }
    }
}
```

Код класу GamesFragment.java:

```
package com.mike.ledcube;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.RecyclerView;

import com.mike.ledcube.CubeCommunication.Games.GameTypes;
import com.mike.ledcube.Dialogs.SnakeGamePreferencesActivity;

import org.jetbrains.annotations.NotNull;

import java.util.ArrayList;
import java.util.List;
```

```
public class GamesFragment extends Fragment {

    private final ArrayList<GameState> games = new
ArrayList<>();


    public GamesFragment() {

    }


    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        SetGames();

    }


    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup
container,

                             Bundle savedInstanceState) {

        return inflater.inflate(R.layout.fragment_games,
container, false);

    }


    @Override

    public void onViewCreated(@NonNull View view, @Nullable
Bundle savedInstanceState) {

        super.onViewCreated(view, savedInstanceState);


        RecyclerView gamesRecyclerView =
requireView().findViewById(R.id.games_recyclerview);

        gamesRecyclerView.setAdapter(new GameAdapter(games));

    }


    private void SetGames() {

        GameState snake = new
GameState(getString(R.string.snake_game_name), GameTypes.Snake);

        games.add(snake);
```

```
    }

    private void startGameActivity(GameTypes game) {
        //PreferencesDialog
        //TODO:tetris game
        switch (game){
            case Snake:
                startActivity(new Intent(requireContext(),
SnakeGamePreferencesActivity.class));
                break;
        }
    }

    private static class GameState {
        private final String name;
        private final GameTypes game;
        private GameState(String name, GameTypes game) {
            this.name = name;
            this.game = game;
        }

        public String getName() {
            return name;
        }
        public GameTypes getGame() {
            return game;
        }
    }
    private class GameAdapter extends
RecyclerView.Adapter<GamesFragment.GameViewHolder> {
        private final List<GameState> games;

        private GameAdapter(List<GameState> games) {
            this.games = games;
```

```
        }

        @NotNull
        @Override
        public GamesFragment.GameViewHolder
onCreateViewHolder(@NotNull ViewGroup parent, int viewType) {
            return new
GameViewHolder(LayoutInflater.from(parent.getContext()).inflate(
R.layout.game_list_item, parent, false));
        }

        @Override
        public void onBindViewHolder(@NotNull
GamesFragment.GameViewHolder holder, int position) {
            GameState state = games.get(position);
            holder.nameTextView.setText(state.getName());
            holder.gameLayout.setOnClickListener((view) ->
startGameActivity(state.getGame()));
        }

        @Override
        public int getItemCount() {
            return games.size();
        }
    }
    private static class GameViewHolder extends
RecyclerView.ViewHolder {
        final TextView nameTextView;
        final ConstraintLayout gameLayout;
        GameViewHolder(View view) {
            super(view);
            nameTextView =
view.findViewById(R.id.game_name_textview);
            gameLayout = view.findViewById(R.id.game_layout);
        }
    }
```

```
}
```

Код класу EffectsFragment.java:

```
package com.mike.ledcube;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.RecyclerView;

import com.mike.ledcube.CubeCommunication.Effects.EffectTypes;
import com.mike.ledcube.Dialogs.FillEffectPreferencesActivity;
import com.mike.ledcube.Dialogs.RainEffectPreferencesActivity;
import com.mike.ledcube.Effects.DrawEffectActivity;
import
com.mike.ledcube.Dialogs.RainbowEffectPreferencesActivity;
import com.mike.ledcube.Dialogs.SinusEffectPreferencesActivity;

import org.jetbrains.annotations.NotNull;

import java.util.ArrayList;
import java.util.List;

public class EffectsFragment extends Fragment {
    private final ArrayList<EffectsFragment.EffectState> effects
= new ArrayList<>();
```

```
    public EffectsFragment() {
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        SetEffects();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_effects,
container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable
Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        RecyclerView effectsRecyclerView =
requireView().findViewById(R.id.effects_recyclerview);
        effectsRecyclerView.setAdapter(new
EffectsFragment.EffectAdapter(effects));
    }

    private void SetEffects() {
        EffectsFragment.EffectState fill = new
EffectsFragment.EffectState(getString(R.string.fill_effect_name)
, EffectTypes.Fill);
        EffectsFragment.EffectState fire = new
EffectsFragment.EffectState(getString(R.string.rain_effect_name)
, EffectTypes.Rain);
```

```
        EffectsFragment.EffectState draw = new
EffectsFragment.EffectState(getString(R.string.draw_effect_name)
, EffectTypes.Draw);

        EffectsFragment.EffectState sinus = new
EffectsFragment.EffectState(getString(R.string.sinus_effect_name
), EffectTypes.Sinus);

        EffectsFragment.EffectState rainbow = new
EffectsFragment.EffectState(getString(R.string.rainbow_effect_na
me), EffectTypes.Rainbow);

        effects.add(fill);

        effects.add(fire);

        effects.add(draw);

        effects.add(sinus);

        effects.add(rainbow);

    }


    private void startEffectActivity(EffectTypes effect) {

        //PreferencesDialog

        switch (effect){

            case Fill:

                startActivity(new Intent(requireContext(),
FillEffectPreferencesActivity.class));

                break;

            case Rain:

                startActivity(new Intent(requireContext(),
RainEffectPreferencesActivity.class));

                break;

            case Draw:

                startActivity(new Intent(requireContext(),
DrawEffectActivity.class));

                break;

            case Sinus:

                startActivity(new Intent(requireContext(),
SinusEffectPreferencesActivity.class));

                break;

            case Rainbow:
```

```
                startActivity(new Intent(requireContext(),
RainbowEffectPreferencesActivity.class));
                break;
        }
    }

    private static class EffectState {
        private final String name;
        private final EffectTypes effect;
        private EffectState(String name, EffectTypes Effect) {
            this.name = name;
            this.effect = Effect;
        }

        public String getName() {
            return name;
        }
        public EffectTypes getEffect() {
            return effect;
        }
    }
    private class EffectAdapter extends
RecyclerView.Adapter<EffectsFragment.EffectViewHolder> {
        private final List<EffectsFragment.EffectState> effects;

        private EffectAdapter(List<EffectsFragment.EffectState>
effects) {
            this.effects = effects;
        }

        @NotNull
        @Override
        public EffectsFragment.EffectViewHolder
onCreateViewHolder(@NotNull ViewGroup parent, int viewType) {
```

```
            return new
EffectsFragment.EffectViewHolder(LayoutInflater.from(parent.getC
ontext()).inflate(R.layout.effect_list_item, parent, false));
        }

        @Override
        public void onBindViewHolder(@NotNull
EffectsFragment.EffectViewHolder holder, int position) {
            EffectsFragment.EffectState state =
effects.get(position);
            holder.nameTextView.setText(state.getName());
            holder.effectLayout.setOnClickListener((view) ->
startEffectActivity(state.getEffect()));
        }

        @Override
        public int getItemCount() {
            return effects.size();
        }
    }
    private static class EffectViewHolder extends
RecyclerView.ViewHolder {
        final TextView nameTextView;
        final ConstraintLayout effectLayout;
        EffectViewHolder(View view) {
            super(view);
            nameTextView =
view.findViewById(R.id.effect_name_textview);
            effectLayout =
view.findViewById(R.id.effect_layout);
        }
    }
}
```

Код класу ChooseDeviceActivity.java:

```
package com.mike.ledcube;
```

```
import android.Manifest;
import android.annotation.SuppressLint;
import android.bluetooth.le.ScanResult;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.result.ActivityResultLauncher;
import
androidx.activity.result.contract.ActivityResultContracts;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.welie.blessed.BluetoothCentralManager;
import com.welie.blessed.BluetoothCentralManagerCallback;
import com.welie.blessed.BluetoothPeripheral;

import org.jetbrains.annotations.NotNull;
```

```
import java.util.ArrayList;
import java.util.List;


public class ChooseDeviceActivity extends AppCompatActivity {
    private final ActivityResultLauncher<String>
requestPermissionLauncher =
            registerForActivityResult(new
ActivityResultContracts.RequestPermission(), isGranted -> {
                if (!isGranted) {
                    Toast.makeText(this,
R.string.bluetooth_required, Toast.LENGTH_LONG).show();
                }
            });
    private final MutableLiveData<BluetoothPeripheral>
availableDeviceList = new MutableLiveData<>();
    private List<String> adresses = new ArrayList<>();


    private final BluetoothCentralManagerCallback
bluetoothCentralManagerCallback = new
BluetoothCentralManagerCallback() {
        @Override
        public void onDiscoveredPeripheral(@NonNull
BluetoothPeripheral peripheral, @NonNull ScanResult scanResult)
{
            if(adresses.stream().noneMatch((x) ->
x.equals(peripheral.getAddress()))) {
                availableDeviceList.postValue(peripheral);
                adresses.add((peripheral.getAddress()));
            }
        }
    };
    BluetoothCentralManager central;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_choose_device);

        if
(checkSelfPermission(android.Manifest.permission.BLUETOOTH_CONNE
CT) == PackageManager.PERMISSION_DENIED ||

checkSelfPermission(android.Manifest.permission.BLUETOOTH_SCAN)
== PackageManager.PERMISSION_DENIED) {

requestPermissionLauncher.launch(android.Manifest.permission.BLU
ETOOTH_CONNECT);

requestPermissionLauncher.launch(Manifest.permission.BLUETOOTH_S
CAN);
        }
        central = new
BluetoothCentralManager(getApplicationContext(),
bluetoothCentralManagerCallback, new
Handler(Looper.getMainLooper()));

        SharedPreferences sharedPref =
this.getSharedPreferences(getString(R.string.pref_file),
Context.MODE_PRIVATE);
        String deviceAddress =
sharedPref.getString(getString(R.string.bluetooth_device_address
), getString(R.string.string_null));
        if
(!deviceAddress.equals(getString(R.string.string_null))) {
            openMainActivity(deviceAddress);
        }

        RecyclerView deviceList =
findViewById(R.id.devices_recyclerView);

        deviceList.setLayoutManager(new
LinearLayoutManager(this));
        DeviceAdapter adapter = new DeviceAdapter();
        deviceList.setAdapter(adapter);
```

```
        getAvailableDeviceList().observe(this,
adapter::updateList);


        central.scanForPeripheralsWithNames(new
String[]{getString(R.string.bluetooth_device_name)});
    }


    private LiveData<BluetoothPeripheral>
getAvailableDeviceList() {
        return availableDeviceList;
    }


    private void openMainActivity(String address) {
        central.stopScan();
        Intent intent = new Intent(this, MainActivity.class);
        intent.putExtra("device_mac", address);
        startActivity(intent);
        finish();
    }


    private class DeviceViewHolder extends
RecyclerView.ViewHolder {


        private final RelativeLayout layout;
        private final TextView text1;
        private final TextView text2;


        DeviceViewHolder(View view) {
            super(view);
            layout = view.findViewById(R.id.list_item);
            text1 = view.findViewById(R.id.list_item_text1);
            text2 = view.findViewById(R.id.list_item_text2);
        }


        void setupView(BluetoothPeripheral device) {
```

```
            if
(ActivityCompat.checkSelfPermission(ChooseDeviceActivity.this,
Manifest.permission.BLUETOOTH_CONNECT) !=
PackageManager.PERMISSION_GRANTED) {
                finish();
            }
            text1.setText(device.getName());
            text2.setText(device.getAddress());
            layout.setOnClickListener(view -> {
                String deviceAddress = device.getAddress();
                SharedPreferences sharedPref =
ChooseDeviceActivity.this.getSharedPreferences(getString(R.strin
g.pref_file), Context.MODE_PRIVATE);
                SharedPreferences.Editor editor =
sharedPref.edit();

editor.putString(getString(R.string.bluetooth_device_address),
deviceAddress);
                editor.apply();
                openMainActivity(deviceAddress);
            });
        }
    }


    private class DeviceAdapter extends
RecyclerView.Adapter<DeviceViewHolder> {
        private final ArrayList<BluetoothPeripheral> deviceList
= new ArrayList<>();


        @NotNull
        @Override
        public DeviceViewHolder onCreateViewHolder(@NotNull
ViewGroup parent, int viewType) {
            return new
DeviceViewHolder(LayoutInflater.from(parent.getContext()).inflat
e(R.layout.list_item, parent, false));
        }
```

```
        @Override
        public void onBindViewHolder(@NotNull DeviceViewHolder
holder, int position) {
            holder.setupView(deviceList.get(position));
        }

        @Override
        public int getItemCount() {
            return deviceList.size();
        }

        @SuppressLint("NotifyDataSetChanged")
        void updateList(BluetoothPeripheral device) {
            deviceList.add(device);
            notifyDataSetChanged();
        }
    }
}
```

Код класу SnakeGameActivity.java:

```
package com.mike.ledcube.Games;

import static
com.mike.ledcube.CubeCommunication.Games.GameCommands.GAME_OVER;
import static
com.mike.ledcube.CubeCommunication.Games.GameCommands.SCORE;
import static com.mike.ledcube.MainActivity.BLHelper;

import android.content.Context;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.view.GestureDetector;
import android.view.KeyEvent;
import android.view.MotionEvent;
```

```
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import com.mike.ledcube.CubeCommunication.BluetoothCommands;
import com.mike.ledcube.CubeCommunication.Games.GameCommands;
import com.mike.ledcube.CubeCommunication.Games.GameTypes;
import com.mike.ledcube.Event;
import com.mike.ledcube.R;

import java.util.Objects;
import java.util.Timer;
import java.util.TimerTask;

public class SnakeGameActivity extends AppCompatActivity {
    private boolean backPressed = false;
    TextView scoreTextView;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_snake_game);

        if (savedInstanceState == null) {
            SharedPreferences pref =
getSharedPreferences(getString(R.string.snake_preferences),
MODE_PRIVATE);
```

```
            Color head =
Color.valueOf(Color.parseColor(pref.getString(getString(R.string
.snake_head_color), "#FF0000")));
            Color body =
Color.valueOf(Color.parseColor(pref.getString(getString(R.string
.snake_body_color), "#00FF00")));
            Color food =
Color.valueOf(Color.parseColor(pref.getString(getString(R.string
.snake_food_color), "#FFFF00")));
            int difficulty =
pref.getInt(getString(R.string.snake_difficulty), 0);


            if
(!BLHelper.send(BluetoothCommands.getGameInitializationCommand(G
ameTypes.Snake,
                    GameCommands.getSnakeInitCommandParams(head,
body, food, difficulty)))) {
                Toast.makeText(this, R.string.error_not_send,
Toast.LENGTH_LONG).show();
                finish();
            }
            scoreTextView =
findViewById(R.id.snake_score_textview);
            scoreTextView.setText(getString(R.string.score, 0));
        }

findViewById(R.id.snake_game_layout).setOnTouchListener(new
OnSwipeTouchListener(this) {
            public void onSwipeRight() {

BLHelper.send(BluetoothCommands.getGameCommand(GameTypes.Snake,
new char[]{GameCommands.RIGHT}));
            }


            public void onSwipeLeft() {

BLHelper.send(BluetoothCommands.getGameCommand(GameTypes.Snake,
new char[]{GameCommands.LEFT}));
            }
```

```
            public void onSwipeTop() {

BLHelper.send(BluetoothCommands.getGameCommand(GameTypes.Snake,
new char[]{GameCommands.FORWARD}));
            }


            public void onSwipeBottom() {

BLHelper.send(BluetoothCommands.getGameCommand(GameTypes.Snake,
new char[]{GameCommands.BACKWARD}));
            }
        });
        BLHelper.getMessages().observe(this, this::onMessage);
    }


    private void onMessage(Event<char[]> message) {
        if (message.hasBeenHandled()) return;
        String[] msg =
BluetoothCommands.parseCommand(Objects.requireNonNull(message.ge
tContentIfNotHandled()));
        if(msg.length != 4) return;
        if (msg[0].equals(String.valueOf(GameCommands.GAMES)) &&
msg[1].equals(String.valueOf(GameCommands.SNAKE_GAME))) {
            if (msg[2].equals(String.valueOf(GAME_OVER))) {
                int score = Integer.parseInt(msg[3]);
                new AlertDialog.Builder(this)

.setMessage(getString(R.string.snake_gameover, score))
                        .setPositiveButton(R.string.ok,
(dialogInterface, i) -> finish())
                        .setCancelable(false).create().show();
            }
            if (msg[2].equals(String.valueOf(SCORE))) {
                int score = Integer.parseInt(msg[3]);
                scoreTextView.setText(getString(R.string.score,
score));
            }
```

```
        }
    }

    @Override
    public void onBackPressed() {
        super.onBackPressed();
        BackPressed();
    }

    private void BackPressed() {
        if (backPressed) {

BLHelper.send(BluetoothCommands.getGameCommand(GameTypes.Snake,
new char[]{GameCommands.EXIT}));
            finish();
        } else {
            backPressed = true;
            Toast.makeText(SnakeGameActivity.this,
R.string.back_pressed, Toast.LENGTH_SHORT).show();
            (new Timer()).schedule(new TimerTask() {
                @Override
                public void run() {
                    backPressed = false;
                }
            }, 2000);
        }
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_VOLUME_DOWN) {

BLHelper.send(BluetoothCommands.getGameCommand(GameTypes.Snake,
new char[]{GameCommands.DOWN}));
        } else if (keyCode == KeyEvent.KEYCODE_VOLUME_UP) {
```

```
BLHelper.send(BluetoothCommands.getGameCommand(GameTypes.Snake,
new char[]{GameCommands.UP}));
        } else if (keyCode == KeyEvent.KEYCODE_BACK) {
            BackPressed();
        }
        return true;
    }

    private static class OnSwipeTouchListener implements
View.OnTouchListener {

        private final GestureDetector gestureDetector;

        public OnSwipeTouchListener(Context ctx) {
            gestureDetector = new GestureDetector(ctx, new
GestureListener());
        }

        @Override
        public boolean onTouch(View v, MotionEvent event) {
            v.performClick();
            return gestureDetector.onTouchEvent(event);
        }

        private final class GestureListener extends
GestureDetector.SimpleOnGestureListener {

            private static final int SWIPE_THRESHOLD = 100;
            private static final int SWIPE_VELOCITY_THRESHOLD =
100;

            @Override
            public boolean onDown(@NonNull MotionEvent e) {
                return true;
```

```
            }

            @Override
            public boolean onFling(MotionEvent e1, @NonNull
MotionEvent e2, float velocityX, float velocityY) {
                boolean result = false;
                try {
                    float diffY = e2.getY() - e1.getY();
                    float diffX = e2.getX() - e1.getX();
                    if (Math.abs(diffX) > Math.abs(diffY)) {
                        if (Math.abs(diffX) > SWIPE_THRESHOLD &&
Math.abs(velocityX) > SWIPE_VELOCITY_THRESHOLD) {
                            if (diffX > 0) {
                                onSwipeRight();
                            } else {
                                onSwipeLeft();
                            }
                            result = true;
                        }
                    } else if (Math.abs(diffY) > SWIPE_THRESHOLD
&& Math.abs(velocityY) > SWIPE_VELOCITY_THRESHOLD) {
                        if (diffY > 0) {
                            onSwipeBottom();
                        } else {
                            onSwipeTop();
                        }
                        result = true;
                    }
                } catch (Exception exception) {
                    exception.printStackTrace();
                }
                return result;
            }
        }
```

```
        public void onSwipeRight() {
        }

        public void onSwipeLeft() {
        }

        public void onSwipeTop() {
        }

        public void onSwipeBottom() {
        }
    }
}
```

Код класу DrawFieldView.java:

```
package com.mike.ledcube.Effects;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.res.TypedArray;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.RectF;
import android.util.AttributeSet;
import android.util.TypedValue;
import android.view.MotionEvent;
import android.view.View;

import androidx.annotation.Nullable;

import com.mike.ledcube.R;
```

```
interface OnCellMoveListener {
    void onCellMove(int row, int column, DrawField field);
}

public class DrawFieldView extends View {
    private final Context context;
    private final AttributeSet attrs;
    private int defStyleAttr;
    private int defStyleRes;

    private int rows;
    private int columns;
    private int gridColor;

    private DrawField drawField;
    private float cellSize;
    private final RectF fieldRect = new RectF(0, 0, 0, 0);

    public OnCellMoveListener cellMoveListener = null;

    public DrawFieldView(Context context) {
        super(context);
        this.context = context;
        attrs = null;
    }

    public DrawFieldView(Context context, @Nullable AttributeSet
attrs) {
        super(context, attrs);
        this.context = context;
        this.attrs = attrs;
        defStyleRes = R.attr.drawFieldStyle;
        initAttributes();
```

```
    }

    public DrawFieldView(Context context, @Nullable AttributeSet
attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        this.context = context;
        this.attrs = attrs;
        this.defStyleAttr = defStyleAttr;
        initAttributes();
    }

/*    public DrawFieldView(Context context, @Nullable
AttributeSet attrs, int defStyleAttr, int defStyleRes) {
        super(context, attrs, defStyleAttr, defStyleRes);
        this.context = context;
        this.attrs = attrs;
        this.defStyleAttr = defStyleAttr;
        this.defStyleRes = defStyleRes;
        initAttributes();
    }*/

    @SuppressLint("ClickableViewAccessibility")
    private void initAttributes() {
        if (attrs != null) {
            TypedArray typedArray =
context.obtainStyledAttributes(attrs, R.styleable.DrawFieldView,
defStyleAttr, defStyleRes);
            gridColor =
typedArray.getColor(R.styleable.DrawFieldView_gridColor,
Color.GRAY);
            typedArray.recycle();
        } else {
            gridColor = Color.GRAY;
        }
        setFocusable(true);
```

```
        setClickable(true);
    }


    /**
     * Use this method to set draw field
     *
     * @param drawfield draw field to draw
     */
    public void setDrawField(DrawField drawfield) {
        if (drawField != null)
drawField.listeners.remove(listener);
        this.drawField = drawfield;
        drawField.listeners.add(listener);
        rows = drawfield.getRows();
        columns = drawfield.getColumns();
        updateViewSizes();
        invalidate();
    }

    public void setGridColor(int color) {
        gridColor = color;
    }

    @Override
    protected void onAttachedToWindow() {
        super.onAttachedToWindow();
        if (drawField != null)
drawField.listeners.add(listener);
    }

    @Override
    protected void onDetachedFromWindow() {
        super.onDetachedFromWindow();
```

```
        if (drawField != null)
drawField.listeners.remove(listener);
    }


    @Override
    protected void onSizeChanged(int w, int h, int oldw, int
oldh) {
        updateViewSizes();
        super.onSizeChanged(w, h, oldw, oldh);
    }


    @Override
    protected void onMeasure(int widthMeasureSpec, int
heightMeasureSpec) {
        int width = widthMeasureSpec;
        int height = heightMeasureSpec;
        if ((MeasureSpec.getMode(widthMeasureSpec) ==
MeasureSpec.AT_MOST || MeasureSpec.getMode(widthMeasureSpec) ==
MeasureSpec.EXACTLY) &&
                (MeasureSpec.getMode(heightMeasureSpec) ==
MeasureSpec.AT_MOST || MeasureSpec.getMode(heightMeasureSpec) ==
MeasureSpec.EXACTLY)) {
            if (MeasureSpec.getSize(widthMeasureSpec) >
MeasureSpec.getSize(heightMeasureSpec)) {
                width =
MeasureSpec.makeMeasureSpec(heightMeasureSpec,
MeasureSpec.EXACTLY);
                setMeasuredDimension(width, heightMeasureSpec);
            } else {
                height =
MeasureSpec.makeMeasureSpec(widthMeasureSpec,
MeasureSpec.EXACTLY);
                setMeasuredDimension(widthMeasureSpec, height);
            }
        }
        super.onMeasure(width, height);
    }
```

```
    private void updateViewSizes() {
        if (drawField == null) return;
        int safeWidth = getWidth() - getPaddingLeft() -
getPaddingRight();
        int safeHeight = getHeight() - getPaddingTop() -
getPaddingBottom();

        float cellWidth = (float) safeWidth /
drawField.getColumns();
        float cellHeight = (float) safeHeight /
drawField.getRows();
        cellSize = Math.min(cellWidth, cellHeight);

        float fieldWidth = cellSize * drawField.getColumns();
        float fieldHeight = cellSize * drawField.getRows();

        fieldRect.left = getPaddingLeft();
        fieldRect.top = getPaddingTop();
        fieldRect.right = fieldRect.left + fieldWidth;
        fieldRect.bottom = fieldRect.top + fieldHeight;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        if (drawField == null) return;
        drawCells(canvas);
        drawGrid(canvas);
    }

    private void drawCells(Canvas canvas) {
        Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setColor(gridColor);
        paint.setStyle(Paint.Style.FILL);
```

```
        float xStart = fieldRect.left;
        float yStart = fieldRect.top;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                paint.setColor(drawField.getCell(i, j));
                canvas.drawRect(xStart + j * cellSize, yStart +
i * cellSize, xStart + (j + 1) * cellSize, yStart + (i + 1) *
cellSize, paint);
            }
        }
    }

    private void drawGrid(Canvas canvas) {
        Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setColor(gridColor);
        paint.setStyle(Paint.Style.STROKE);

paint.setStrokeWidth(TypedValue.applyDimension(TypedValue.COMPLE
X_UNIT_DIP, 3f, getResources().getDisplayMetrics()));

        float xStart = fieldRect.left;
        float xEnd = fieldRect.right;
        for (int i = 0; i <= drawField.getRows(); i++) {
            float y = fieldRect.top + cellSize * i;
            canvas.drawLine(xStart, y, xEnd, y, paint);
        }

        float yStart = fieldRect.top;
        float yEnd = fieldRect.bottom;
        for (int i = 0; i <= drawField.getColumns(); i++) {
            float x = fieldRect.left + cellSize * i;
            canvas.drawLine(x, yStart, x, yEnd, paint);
        }
    }
```

```
    @SuppressLint("ClickableViewAccessibility")
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (drawField == null) return false;
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                performClick();
            case MotionEvent.ACTION_MOVE:
                int row = getRow(event);
                int column = getColumn(event);
                if (row >= 0 && column >= 0 && row <
drawField.getRows() && column < drawField.getColumns()) {
                    if (cellMoveListener != null) {
                        cellMoveListener.onCellMove(row, column,
drawField);
                        return true;
                    }
                }
                break;
        }
        return false;
    }

    private int getRow(MotionEvent event) {
        return (int) ((event.getY() - fieldRect.top) /
cellSize);
    }

    private int getColumn(MotionEvent event) {
        return (int) ((event.getX() - fieldRect.left) /
cellSize);
    }

    private final OnFieldChangedListener listener = (view) ->
invalidate();
```

```
}
```

Код класу DrawEffectActivity.java:

```
package com.mike.ledcube.Effects;

import static com.mike.ledcube.MainActivity.BLHelper;

import android.graphics.Color;
import android.os.Bundle;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.mike.ledcube.CubeCommunication.BluetoothCommands;
import
com.mike.ledcube.CubeCommunication.Effects.EffectCommands;
import com.mike.ledcube.CubeCommunication.Effects.EffectTypes;
import com.mike.ledcube.R;

public class DrawEffectActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_draw_effect);

        CubeDrawView view = findViewById(R.id.draw_field);

        if
(!BLHelper.send(BluetoothCommands.getEffectCommand(EffectTypes.F
ill,

EffectCommands.getFillEffectCommandParams(Color.valueOf(0))))) {
            Toast.makeText(this, R.string.error_not_send,
Toast.LENGTH_LONG).show();
```

```
            finish();
        }


        view.setDimensions(8, 8, 8,
getSupportFragmentManager());
        view.cellChangedListener = (layer, row, column, field) -
> {
            if
(!BLHelper.send(BluetoothCommands.getEffectCommand(EffectTypes.D
raw,

EffectCommands.getDrawEffectCommandParams(layer, row, column,
                                Color.valueOf(field.getCell(row,
column))))) {
                Toast.makeText(this, R.string.error_not_send,
Toast.LENGTH_LONG).show();
                finish();
            }
        };
        view.fieldClearedListener = (layer) -> {
            if
(!BLHelper.send(BluetoothCommands.getEffectCommand(EffectTypes.D
raw,

EffectCommands.getDrawClearedEffectCommandParams(layer))) {
                Toast.makeText(this, R.string.error_not_send,
Toast.LENGTH_LONG).show();
                finish();
            }
        };
    }
}
```

Код класу CubeDrawView.java:

```
package com.mike.ledcube.Effects;


import android.content.Context;
import android.content.res.TypedArray;
```

```
import android.graphics.Color;
import android.os.Parcel;
import android.os.Parcelable;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.NumberPicker;

import androidx.annotation.Nullable;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.FragmentManager;

import com.mike.ledcube.R;
import com.mike.ledcube.databinding.CubeDrawLayoutBinding;

import me.jfenn.colorpickerdialog.dialogs.ColorPickerDialog;

interface OnCellChangedListener {
    void onCellChanged(int layer, int row, int column, DrawField
field);
}

interface OnFieldCleared {
    void onFieldCleared(int layer);
}

public class CubeDrawView extends ConstraintLayout implements
View.OnClickListener, NumberPicker.OnValueChangeListener {
    private final Context context;
    private final AttributeSet attrs;
    private int defStyleAttr;
    private int defStyleRes;
    private CubeDrawLayoutBinding binding;
    private FragmentManager fragmentManager;
```

```
    public OnCellChangedListener cellChangedListener = null;
    public OnFieldCleared fieldClearedListener = null;
    private int currentColor;
    private int currentLayer;

    private int layers;
    private int rows;
    private int columns;

    private DrawField[] drawFields;

    public CubeDrawView(Context context) {
        super(context);
        this.context = context;
        attrs = null;
    }

    public CubeDrawView(Context context, @Nullable AttributeSet
attrs) {
        super(context, attrs);
        this.context = context;
        this.attrs = attrs;
        defStyleRes = R.attr.drawFieldStyle;
        initAttributes();
    }

    public CubeDrawView(Context context, @Nullable AttributeSet
attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        this.context = context;
        this.attrs = attrs;
        this.defStyleAttr = defStyleAttr;
        initAttributes();
```

```
    }


   /* public CubeDrawView(Context context, @Nullable
AttributeSet attrs, int defStyleAttr, int defStyleRes) {
        super(context, attrs, defStyleAttr, defStyleRes);
        this.context = context;
        this.attrs = attrs;
        this.defStyleAttr = defStyleAttr;
        this.defStyleRes = defStyleRes;
        initAttributes();
    }*/


    private void initAttributes() {
        LayoutInflater inflater = LayoutInflater.from(context);
        inflater.inflate(R.layout.cube_draw_layout, this, true);
        binding = CubeDrawLayoutBinding.bind(this);
        int gridColor;
        if (attrs != null) {
            TypedArray typedArray =
context.obtainStyledAttributes(attrs, R.styleable.CubeDrawView,
defStyleAttr, defStyleRes);
            gridColor =
typedArray.getColor(R.styleable.CubeDrawView_cubeGridColor,
Color.GRAY);
            typedArray.recycle();
        } else {
            gridColor = Color.GRAY;
        }
        binding.clearButton.setOnClickListener(this);
        binding.colorButton.setOnClickListener(this);
        binding.layerPicker.setOnValueChangedListener(this);
        binding.drawFieldView.setGridColor(gridColor);
        binding.drawFieldView.cellMoveListener = (row, column,
field) -> {
            field.setCell(row, column, currentColor);
```

```
            if (cellChangedListener != null) {
                cellChangedListener.onCellChanged(currentLayer,
row, column, field);
            }
        };
    }

    public void setDimensions(int layers, int rows, int columns,
FragmentManager fragmentManager) {
        this.fragmentManager = fragmentManager;
        this.layers = layers;
        this.rows = rows;
        this.columns = columns;
        currentLayer = 0;
        currentColor = Color.WHITE;
        drawFields = new DrawField[layers];
        for (int i = 0; i < layers; i++)
            drawFields[i] = new DrawField(rows, columns);
        binding.layerPicker.setMinValue(1);
        binding.layerPicker.setMaxValue(layers);
        binding.layerPicker.setValue(1);
        binding.colorButton.setBackgroundColor(currentColor);
        binding.drawFieldView.setDrawField(drawFields[0]);
    }

    @Override
    public void onClick(View v) {
        if (v == binding.clearButton) {
            drawFields[currentLayer].clear();
            if (fieldClearedListener != null) {

fieldClearedListener.onFieldCleared(currentLayer);
            }
        } else if (v == binding.colorButton) {
```

```
            int[] colorPallet = new int[16];
            for (int i = 0; i < colorPallet.length - 2; i++) {
                colorPallet[i] = Color.HSVToColor(new
float[]{360f / (colorPallet.length - 2) * i, 1, 1});
            }
            colorPallet[colorPallet.length - 2] = Color.BLACK;
            colorPallet[colorPallet.length - 1] = Color.WHITE;
            new ColorPickerDialog()
                    .withAlphaEnabled(false)
                    .withPresets(colorPallet)
                    .withColor(currentColor)
                    .withCornerRadius(20)
                    .withListener((dialog, color) -> {
                        currentColor = color;

binding.colorButton.setBackgroundColor(color);
                    })
                    .show(fragmentManager, "drawColorPicker");
        }
    }


    @Override
    public void onValueChange(NumberPicker picker, int oldVal,
int newVal) {
        currentLayer = newVal - 1;

binding.drawFieldView.setDrawField(drawFields[currentLayer]);
        invalidate();
    }


    @Nullable
    @Override
    protected Parcelable onSaveInstanceState() {
        Parcelable superState = super.onSaveInstanceState();
        SavedState savedState = new SavedState(superState);
```

```
        savedState.layers = layers;
        savedState.rows = rows;
        savedState.columns = columns;
        savedState.currentLayer = currentLayer;
        savedState.currentColor = currentColor;
        int[][][] colors = new int[layers][rows][columns];
        for (int i = 0; i < layers; i++) {
            for (int j = 0; j < rows; j++) {
                for (int k = 0; k < columns; k++) {
                    colors[i][j][k] = drawFields[i].getCell(j,
k);
                }
            }
        }
        savedState.colors = colors;
        return savedState;
    }

    @Override
    protected void onRestoreInstanceState(Parcelable state) {
        SavedState savedState = (SavedState) state;

super.onRestoreInstanceState(savedState.getSuperState());
        this.layers = savedState.layers;
        this.rows = savedState.rows;
        this.columns = savedState.columns;
        this.currentLayer = savedState.currentLayer;
        this.currentColor = savedState.currentColor;
        drawFields = new DrawField[layers];
        for (int i = 0; i < layers; i++)
            drawFields[i] = new DrawField(rows, columns);
        for (int i = 0; i < layers; i++) {
            for (int j = 0; j < rows; j++) {
                for (int k = 0; k < columns; k++) {
```

```
                    drawFields[i].setCell(j, k,
savedState.colors[i][j][k], false);
                }
            }
        }
        binding.colorButton.setBackgroundColor(currentColor);
        binding.layerPicker.setValue(currentLayer + 1);

binding.drawFieldView.setDrawField(drawFields[currentLayer]);
    }

    static class SavedState extends BaseSavedState {
        private int layers;
        private int rows;
        private int columns;
        private int currentLayer;
        private int currentColor;
        private int[][][] colors;

        public SavedState(Parcelable superState) {
            super(superState);
        }

        public SavedState(Parcel parcel) {
            super(parcel);
            layers = parcel.readInt();
            rows = parcel.readInt();
            columns = parcel.readInt();
            currentLayer = parcel.readInt();
            currentColor = parcel.readInt();
            colors = new int[layers][rows][columns];
            for (int x = 0; x < layers; x++) {
                for (int y = 0; y < rows; y++) {
                    parcel.readIntArray(colors[x][y]);
```

```
                }
            }
        }

        @Override
        public void writeToParcel(Parcel out, int flags) {
            super.writeToParcel(out, flags);
            out.writeInt(layers);
            out.writeInt(rows);
            out.writeInt(columns);
            out.writeInt(currentLayer);
            out.writeInt(currentColor);
            for (int x = 0; x < layers; x++) {
                for (int y = 0; y < rows; y++) {
                    out.writeIntArray(colors[x][y]);
                }
            }
        }

        public static final Creator<SavedState> CREATOR = new
Creator<>() {
            @Override
            public SavedState createFromParcel(Parcel in) {
                return new SavedState(in);
            }

            @Override
            public SavedState[] newArray(int size) {
                return new SavedState[size];
            }
        };
    }
}
```

Код класу FillEffectPreferencesActivity.java:

```
package com.mike.ledcube.Dialogs;

import static com.mike.ledcube.MainActivity.BLHelper;

import android.content.Context;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.widget.Button;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.mike.ledcube.CubeCommunication.BluetoothCommands;
import
com.mike.ledcube.CubeCommunication.Effects.EffectCommands;
import com.mike.ledcube.CubeCommunication.Effects.EffectTypes;
import com.mike.ledcube.R;

import me.jfenn.colorpickerdialog.dialogs.ColorPickerDialog;

public class FillEffectPreferencesActivity extends
AppCompatActivity {
    Button fillButton;
    Button cancelButton;
    Button startButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fill_dialog_layout);
```

```
        fillButton = findViewById(R.id.fill_button);


        Setup();
        cancelButton = findViewById(R.id.fill_cancel_button);
        cancelButton.setOnClickListener((view) -> finish());
        startButton = findViewById(R.id.fill_ok_button);
        startButton.setOnClickListener((view) ->{
            SharedPreferences pref =
getSharedPreferences(getString(R.string.fill_preferences),
MODE_PRIVATE);
            Color fill =
Color.valueOf(Color.parseColor(pref.getString(getString(R.string
.fill_color), "#FFFFFF")));
            if
(!BLHelper.send(BluetoothCommands.getEffectCommand(EffectTypes.F
ill, EffectCommands.getFillEffectCommandParams(fill)))) {
                Toast.makeText(this, R.string.error_not_send,
Toast.LENGTH_LONG).show();
            }
            finish();
        });
    }


    private void Setup() {
        SharedPreferences sharedPref =
getSharedPreferences(getString(R.string.fill_preferences),
Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPref.edit();



fillButton.setBackgroundColor(Color.parseColor(sharedPref.getStr
ing(getString(R.string.fill_color), "#FFFFFF")));
        fillButton.setOnClickListener((view -> new
ColorPickerDialog()

.withColor(Color.parseColor(sharedPref.getString(getString(R.str
ing.fill_color), "#FFFFFF")))
                .withAlphaEnabled(false)
```

```
                .withCornerRadius(20)
                .withListener((dialog, color) -> {

editor.putString(getString(R.string.fill_color),
String.format("#%06X", (0xFFFFFF & color)));
                    editor.apply();
                    fillButton.setBackgroundColor(color);
                    if
(!BLHelper.send(BluetoothCommands.getEffectCommand(EffectTypes.F
ill,
EffectCommands.getFillEffectCommandParams(Color.valueOf(color)))
)) {
                        Toast.makeText(this,
R.string.error_not_send, Toast.LENGTH_LONG).show();
                    }
                })
                .show(getSupportFragmentManager(),
"fillColorPicker")));
    }
}
```

Код класу RainbowEffectPreferencesActivity.java:

```
package com.mike.ledcube.Dialogs;

import static com.mike.ledcube.MainActivity.BLHelper;

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.mike.ledcube.CubeCommunication.BluetoothCommands;
```

```
import
com.mike.ledcube.CubeCommunication.Effects.EffectCommands;
import com.mike.ledcube.CubeCommunication.Effects.EffectTypes;
import com.mike.ledcube.R;


public class RainbowEffectPreferencesActivity extends
AppCompatActivity {
    SeekBar speedSeekbar;
    Button cancelButton;
    Button startButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.rainbow_dialog_layout);


        speedSeekbar = findViewById(R.id.rainbow_speed);


        Setup();
        cancelButton = findViewById(R.id.rainbow_cancel_button);
        cancelButton.setOnClickListener((view) -> finish());
        startButton = findViewById(R.id.rainbow_ok_button);
        startButton.setOnClickListener((view) ->{
            if
(!BLHelper.send(BluetoothCommands.getEffectCommand(EffectTypes.R
ainbow,

EffectCommands.getRainbowEffectCommandParams(speedSeekbar.getPro
gress()))))) {
                Toast.makeText(this, R.string.error_not_send,
Toast.LENGTH_LONG).show();
            }
            finish();
        });
    }


    private void Setup() {
```

```
        SharedPreferences sharedPref =
getSharedPreferences(getString(R.string.rainbow_preferences),
Context.MODE_PRIVATE);

        SharedPreferences.Editor editor = sharedPref.edit();

        int speed =
sharedPref.getInt(getString(R.string.rainbow_speed), 127);


        speedSeekbar.setProgress(speed);

        speedSeekbar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {

            @Override

            public void onProgressChanged(SeekBar seekBar, int
i, boolean b) {

                editor.putInt(getString(R.string.rain_speed),
seekBar.getProgress());

                editor.apply();

            }


            @Override

            public void onStartTrackingTouch(SeekBar seekBar) {


            }


            @Override

            public void onStopTrackingTouch(SeekBar seekBar) {


            }

        });

    }

}
```

Код класу RainEffectPreferencesActivity.java:

```
package com.mike.ledcube.Dialogs;


import static com.mike.ledcube.MainActivity.BLHelper;
```

```
import android.content.Context;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.mike.ledcube.CubeCommunication.BluetoothCommands;
import
com.mike.ledcube.CubeCommunication.Effects.EffectCommands;
import com.mike.ledcube.CubeCommunication.Effects.EffectTypes;
import com.mike.ledcube.R;

import me.jfenn.colorpickerdialog.dialogs.ColorPickerDialog;

public class RainEffectPreferencesActivity extends
AppCompatActivity {
    Button colorButton;
    SeekBar speedSeekbar;
    Button cancelButton;
    Button startButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.rain_dialog_layout);

        colorButton = findViewById(R.id.rain_color_button);
        speedSeekbar = findViewById(R.id.rain_speed);

        Setup();
```

```
        cancelButton = findViewById(R.id.rain_cancel_button);
        cancelButton.setOnClickListener((view) -> finish());
        startButton = findViewById(R.id.rain_ok_button);
        startButton.setOnClickListener((view) ->{
            SharedPreferences pref =
getSharedPreferences(getString(R.string.rain_preferences),
MODE_PRIVATE);
            Color color =
Color.valueOf(Color.parseColor(pref.getString(getString(R.string
.rain_color), "#FFFF00")));
            if
(!BLHelper.send(BluetoothCommands.getEffectCommand(EffectTypes.R
ain,

EffectCommands.getRainEffectCommandParams(color,
speedSeekbar.getProgress()))))) {
                Toast.makeText(this, R.string.error_not_send,
Toast.LENGTH_LONG).show();
            }
            finish();
        });
    }

    private void Setup() {
        SharedPreferences sharedPref =
getSharedPreferences(getString(R.string.rain_preferences),
Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPref.edit();
        String color =
sharedPref.getString(getString(R.string.rain_color), "#FFFF00");
        int speed =
sharedPref.getInt(getString(R.string.rain_speed), 127);

        colorButton.setBackgroundColor(Color.parseColor(color));
        colorButton.setOnClickListener((view -> new
ColorPickerDialog()
                .withColor(Color.parseColor(color))
                .withAlphaEnabled(false)
```

```
                .withCornerRadius(20)
                .withListener((dialog, c) -> {

editor.putString(getString(R.string.rain_color),
String.format("#%06X", (0xFFFFFF & c)));
                    editor.apply();
                    colorButton.setBackgroundColor(c);
                })
                .show(getSupportFragmentManager(),
"colorPicker")));
        speedSeekbar.setProgress(speed);
        speedSeekbar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int
i, boolean b) {
                editor.putInt(getString(R.string.rain_speed),
seekBar.getProgress());
                editor.apply();
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });
    }
}
```

Код класу SinusEffectPreferencesActivity.java:

```
package com.mike.ledcube.Dialogs;
```

```
import static com.mike.ledcube.MainActivity.BLHelper;

import android.content.Context;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.mike.ledcube.CubeCommunication.BluetoothCommands;
import
com.mike.ledcube.CubeCommunication.Effects.EffectCommands;
import com.mike.ledcube.CubeCommunication.Effects.EffectTypes;
import com.mike.ledcube.R;

import me.jfenn.colorpickerdialog.dialogs.ColorPickerDialog;

public class SinusEffectPreferencesActivity extends
AppCompatActivity {
    Button colorButton;
    SeekBar speedSeekbar;
    Button cancelButton;
    Button startButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.sinus_dialog_layout);

        colorButton = findViewById(R.id.sinus_color_button);
        speedSeekbar = findViewById(R.id.sinus_speed);
```

```
        Setup();
        cancelButton = findViewById(R.id.sinus_cancel_button);
        cancelButton.setOnClickListener((view) -> finish());
        startButton = findViewById(R.id.sinus_ok_button);
        startButton.setOnClickListener((view) ->{
            SharedPreferences pref =
getSharedPreferences(getString(R.string.sinus_preferences),
MODE_PRIVATE);
            Color color =
Color.valueOf(Color.parseColor(pref.getString(getString(R.string
.sinus_color), "#FFFF00")));
            if
(!BLHelper.send(BluetoothCommands.getEffectCommand(EffectTypes.S
inus,

EffectCommands.getSinusEffectCommandParams(color,
speedSeekbar.getProgress()))))) {
                Toast.makeText(this, R.string.error_not_send,
Toast.LENGTH_LONG).show();
            }
            finish();
        });
    }


    private void Setup() {
        SharedPreferences sharedPref =
getSharedPreferences(getString(R.string.sinus_preferences),
Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPref.edit();
        String color =
sharedPref.getString(getString(R.string.sinus_color),
"#FFFF00");
        int speed =
sharedPref.getInt(getString(R.string.sinus_speed), 127);


        colorButton.setBackgroundColor(Color.parseColor(color));
        colorButton.setOnClickListener((view -> new
ColorPickerDialog()
```

```
                .withColor(Color.parseColor(color))
                .withAlphaEnabled(false)
                .withCornerRadius(20)
                .withListener((dialog, c) -> {

editor.putString(getString(R.string.sinus_color),
String.format("#%06X", (0xFFFFFF & c)));
                    editor.apply();
                    colorButton.setBackgroundColor(c);
                })
                .show(getSupportFragmentManager(),
"colorPicker")));
        speedSeekbar.setProgress(speed);
        speedSeekbar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int
i, boolean b) {
                editor.putInt(getString(R.string.sinus_speed),
seekBar.getProgress());
                editor.apply();
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });
    }
}
```

Код класу SnakeGamePreferencesActivity.java:

```
package com.mike.ledcube.Dialogs;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.Spinner;

import androidx.appcompat.app.AppCompatActivity;

import com.mike.ledcube.Games.SnakeGameActivity;
import com.mike.ledcube.R;

import me.jfenn.colorpickerdialog.dialogs.ColorPickerDialog;

public class SnakeGamePreferencesActivity extends
AppCompatActivity {
    Spinner difficultySpinner;
    Button headButton;
    Button bodyButton;
    Button foodButton;
    Button cancelButton;
    Button startButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.snake_dialog_layout);
```

```
        difficultySpinner =
findViewById(R.id.snake_difficulty_spinner);
        headButton = findViewById(R.id.snake_head_button);
        bodyButton = findViewById(R.id.snake_body_button);
        foodButton = findViewById(R.id.snake_food_button);


        Setup();
        cancelButton = findViewById(R.id.snake_cancel_button);
        cancelButton.setOnClickListener((view) -> finish());
        startButton = findViewById(R.id.snake_ok_button);
        startButton.setOnClickListener((view) ->{
            startActivity(new Intent(this,
SnakeGameActivity.class));
            finish();
        });
    }


    private void Setup() {
        SharedPreferences sharedPref =
getSharedPreferences(getString(R.string.snake_preferences),
Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPref.edit();
        String headColor =
sharedPref.getString(getString(R.string.snake_head_color),
"#FF0000");
        String bodyColor =
sharedPref.getString(getString(R.string.snake_body_color),
"#00FF00");
        String foodColor =
sharedPref.getString(getString(R.string.snake_food_color),
"#FFFF00");
        int difficulty =
sharedPref.getInt(getString(R.string.snake_difficulty), 0);


headButton.setBackgroundColor(Color.parseColor(headColor));
```

```
bodyButton.setBackgroundColor(Color.parseColor(bodyColor));

foodButton.setBackgroundColor(Color.parseColor(foodColor));
        difficultySpinner.setSelection(difficulty);

        //createListeners
        headButton.setOnClickListener((view -> new
ColorPickerDialog()
                .withColor(Color.parseColor(headColor))
                .withAlphaEnabled(false)
                .withCornerRadius(20)
                .withListener((dialog, color) -> {

editor.putString(getString(R.string.snake_head_color),
String.format("#%06X", (0xFFFFFF & color)));
                    editor.apply();
                    headButton.setBackgroundColor(color);
                })
                .show(getSupportFragmentManager(),
"snakeHeadColorPicker")));
        bodyButton.setOnClickListener((view -> new
ColorPickerDialog()
                .withColor(Color.parseColor(bodyColor))
                .withAlphaEnabled(false)
                .withCornerRadius(20)
                .withListener((dialog, color) -> {

editor.putString(getString(R.string.snake_body_color),
String.format("#%06X", (0xFFFFFF & color)));
                    editor.apply();
                    bodyButton.setBackgroundColor(color);
                })
                .show(getSupportFragmentManager(),
"snakeBodyColorPicker")));
        foodButton.setOnClickListener((view -> new
ColorPickerDialog()
```

```
                .withColor(Color.parseColor(foodColor))
                .withAlphaEnabled(false)
                .withCornerRadius(20)
                .withListener((dialog, color) -> {

editor.putString(getString(R.string.snake_food_color),
String.format("#%06X", (0xFFFFFF & color)));
                    editor.apply();
                    foodButton.setBackgroundColor(color);
                })
                .show(getSupportFragmentManager(),
"snakeFoodColorPicker")));
        difficultySpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent,
View view, int position, long id) {

editor.putInt(getString(R.string.snake_difficulty), position);
                editor.apply();
            }


            @Override
            public void onNothingSelected(AdapterView<?>
adapterView) {


            }
        });
    }
}
```

Код класу PreferencesCommands.java:

```
package com.mike.ledcube.CubeCommunication;


public class PreferencesCommands {
    public static final char PREFERENCES = '0';
```

```
    public static final char PREFERENCE_STATE = '0';
    public static final char PREFERENCE_STATE_OFF = '0';
    public static final char PREFERENCE_STATE_ON = '1';
}
```

Код класу PreferencesCommands.java:

```
package com.mike.ledcube.CubeCommunication;


import static
com.mike.ledcube.CubeCommunication.Effects.EffectCommands.DRAW_E
FFECT;
import static
com.mike.ledcube.CubeCommunication.Effects.EffectCommands.EFFECT
S;
import static
com.mike.ledcube.CubeCommunication.Effects.EffectCommands.FILL_E
FFECT;
import static
com.mike.ledcube.CubeCommunication.Effects.EffectCommands.RAINBO
W_EFFECT;
import static
com.mike.ledcube.CubeCommunication.Effects.EffectCommands.RAIN_E
FFECT;
import static
com.mike.ledcube.CubeCommunication.Effects.EffectCommands.SINUS_
EFFECT;
import static
com.mike.ledcube.CubeCommunication.Games.GameCommands.GAMES;
import static
com.mike.ledcube.CubeCommunication.Games.GameCommands.SNAKE_GAME
;
import static
com.mike.ledcube.CubeCommunication.PreferencesCommands.PREFERENC
ES;
import static
com.mike.ledcube.CubeCommunication.PreferencesCommands.PREFERENC
E_STATE;
import static
com.mike.ledcube.CubeCommunication.PreferencesCommands.PREFERENC
E_STATE_OFF;
```

```
import static
com.mike.ledcube.CubeCommunication.PreferencesCommands.PREFERENC
E_STATE_ON;


import com.mike.ledcube.CubeCommunication.Effects.EffectTypes;

import com.mike.ledcube.CubeCommunication.Games.GameTypes;


import java.util.ArrayList;


public class BluetoothCommands {

    public static final char EOL = ';';

    public static final char DELIMITER = ',';


    // -- Preferences --

    public static char[] offCommand() {

        return new char[]{PREFERENCES, DELIMITER,
PREFERENCE_STATE, DELIMITER,

                PREFERENCE_STATE_OFF, EOL};

    }


    public static char[] onCommand() {

        return new char[]{PREFERENCES, DELIMITER,
PREFERENCE_STATE, DELIMITER,

                PREFERENCE_STATE_ON, EOL};

    }


    public static char[] getStateCommand() {

        return new char[]{PREFERENCES, DELIMITER,
PREFERENCE_STATE, EOL};

    }


    // -- Effects --

    public static char[] getEffectCommand(EffectTypes effect,
char[] params) {

        char[] res = new char[params.length + 5];
```

```
    int i = 0;
    res[i++] = EFFECTS;
    res[i++] = DELIMITER;
    switch (effect) {
        case Fill:
            res[i++] = FILL_EFFECT;
            res[i++] = DELIMITER;
            break;
        case Rain:
            res[i++] = RAIN_EFFECT;
            res[i++] = DELIMITER;
            break;
        case Draw:
            res[i++] = DRAW_EFFECT;
            res[i++] = DELIMITER;
            break;
        case Rainbow:
            res[i++] = RAINBOW_EFFECT;
            res[i++] = DELIMITER;
            break;
        case Sinus:
            res[i++] = SINUS_EFFECT;
            res[i++] = DELIMITER;
            break;
    }
    for (char p : params) {
        res[i++] = p;
    }
    res[i] = EOL;
    return res;
}

// -- Games --
```

```
    public static char[] getGameInitializationCommand(GameTypes
game, char[] params) {
        char[] res = new char[params.length + 5];
        int i = 0;
        res[i++] = GAMES;
        res[i++] = DELIMITER;
        if (game == GameTypes.Snake) {
            res[i++] = SNAKE_GAME;
            res[i++] = DELIMITER;
        }
        for (char p : params) {
            res[i++] = p;
        }
        res[i] = EOL;
        return res;
    }

    public static char[] getGameCommand(GameTypes game, char[]
params){
        char[] res = new char[params.length + 5];
        int i = 0;
        res[i++] = GAMES;
        res[i++] = DELIMITER;
        if (game == GameTypes.Snake) {
            res[i++] = SNAKE_GAME;
            res[i++] = DELIMITER;
        }
        for (char p : params) {
            res[i++] = p;
        }
        res[i] = EOL;
        return res;
    }
```

```
    public static String[] parseCommand(char[] command){
        ArrayList<String> res = new ArrayList<>();
        StringBuilder tmp = new StringBuilder();
        for (char s: command) {
            if(s == DELIMITER || s == EOL){
                res.add(tmp.toString());
                tmp = new StringBuilder();
                continue;
            }
            tmp.append(s);
        }
        return res.toArray(new String[0]);
    }


    public static String getColorComponent(float value) {
        return String.valueOf((int) Math.floor(value == 1 ? 255
: value * 256));
    }
}
```

Код класу GameCommands.java:

```
package com.mike.ledcube.CubeCommunication.Games;


import static
com.mike.ledcube.CubeCommunication.BluetoothCommands.DELIMITER;
import static
com.mike.ledcube.CubeCommunication.BluetoothCommands.getColorCom
ponent;


import android.graphics.Color;


public class GameCommands {
    public static final char GAMES = '2';
    public static final char SNAKE_GAME = '0';
```

```
    public static final char UP = '0';
    public static final char DOWN = '1';
    public static final char LEFT = '2';
    public static final char RIGHT = '3';
    public static final char FORWARD = '4';
    public static final char BACKWARD = '5';
    public static final char EXIT = '6';
    public static final char GAME_OVER = '0';
    public static final char SCORE = '1';
    public static char[] getSnakeInitCommandParams(Color
headColor, Color bodyColor, Color foodColor, int difficulty){
        String res = "";
        res += getColorComponent(headColor.red());
        res += DELIMITER;
        res += getColorComponent(headColor.green());
        res += DELIMITER;
        res += getColorComponent(headColor.blue());
        res += DELIMITER;
        res += getColorComponent(bodyColor.red());
        res += DELIMITER;
        res += getColorComponent(bodyColor.green());
        res += DELIMITER;
        res += getColorComponent(bodyColor.blue());
        res += DELIMITER;
        res += getColorComponent(foodColor.red());
        res += DELIMITER;
        res += getColorComponent(foodColor.green());
        res += DELIMITER;
        res += getColorComponent(foodColor.blue());
        res += DELIMITER;
        res += String.valueOf(difficulty);
        return res.toCharArray();
    }
```

```
}
```

Код класу GameTypes.java:

```
package com.mike.ledcube.CubeCommunication.Games;


public enum GameTypes {
    Snake
}
```

Код класу EffectCommands.java:

```
package com.mike.ledcube.CubeCommunication.Effects;


import static
com.mike.ledcube.CubeCommunication.BluetoothCommands.DELIMITER;
import static
com.mike.ledcube.CubeCommunication.BluetoothCommands.getColorCom
ponent;


import android.graphics.Color;


public class EffectCommands {
    public static final char EFFECTS = '1';
    public static final char FILL_EFFECT = '0';
    public static final char RAIN_EFFECT = '4';
    public static final char RAINBOW_EFFECT = '5';
    public static final char SINUS_EFFECT = '3';
    public static final char DRAW_EFFECT = '2';
    public static final char DRAW_EFFECT_CLEAR = '9';
    public static char[] getFillEffectCommandParams(Color
color){
        String res = "";
        res += getColorComponent(color.red());
        res += DELIMITER;
        res += getColorComponent(color.green());
        res += DELIMITER;
        res += getColorComponent(color.blue());
```

```
        return res.toCharArray();
    }
    public static char[] getRainEffectCommandParams(Color color,
int speed){
        String res = "";
        res += getColorComponent(color.red());
        res += DELIMITER;
        res += getColorComponent(color.green());
        res += DELIMITER;
        res += getColorComponent(color.blue());
        res += DELIMITER;
        res += String.valueOf(speed);
        return res.toCharArray();
    }

    public static char[] getDrawEffectCommandParams(int layer,
int row, int column, Color color){
        String res = "";
        res += String.valueOf(layer);
        res += DELIMITER;
        res += String.valueOf(column);
        res += DELIMITER;
        res += String.valueOf(7 - row);
        res += DELIMITER;
        res += getColorComponent(color.red());
        res += DELIMITER;
        res += getColorComponent(color.green());
        res += DELIMITER;
        res += getColorComponent(color.blue());
        return res.toCharArray();
    }

    public static char[] getDrawClearedEffectCommandParams(int
layer){
```

```
        String res = "";
        res += DRAW_EFFECT_CLEAR;
        res += DELIMITER;
        res += String.valueOf(layer);
        return res.toCharArray();
    }

    public static char[] getRainbowEffectCommandParams(int
speed) {
        String res = "";
        res += String.valueOf(speed);
        return res.toCharArray();
    }

    public static char[] getSinusEffectCommandParams(Color
color, int speed) {
        String res = "";
        res += getColorComponent(color.red());
        res += DELIMITER;
        res += getColorComponent(color.green());
        res += DELIMITER;
        res += getColorComponent(color.blue());
        res += DELIMITER;
        res += String.valueOf(speed);
        return res.toCharArray();
    }
}
```

Код класу EffectTypes.java:

```
package com.mike.ledcube.CubeCommunication.Effects;
public enum EffectTypes {
    Fill,
    Rain,
    Sinus, Rainbow, Draw
```

```
}
```

Код файлу розмітки navigation_bar_menu.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/mainFragment"
        android:title="@string/main" />
    <item android:id="@+id/effectsFragment"
        android:title="@string/effects"/>
<!--         android:icon="" />-->
    <item android:id="@+id/gamesFragment"
        android:title="@string/games"/>
<!--         android:icon="" />-->
</menu>
```

Код файлу розмітки strings.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Led Cube</string>
    <string name="games">Ігри</string>
    <string name="effects">Ефекти</string>
    <string name="main">Головна</string>
    <string name="bluetooth_required">Помилка: потребується
Bluetooth</string>
    <string name="choose_device">Виберіть пристрій</string>
    <string name="disconnect">Відключитися</string>
    <string name="status_connected">Підключено</string>
    <string name="status_connecting">Підключення…</string>
    <string name="status_disconnected">Відключено</string>
    <string name="cube_on">Увімкнути куб</string>
    <string name="connect">Підключитися</string>
    <string name="snake_game_name">Змійка</string>
    <string name="choose_snake_head_color">Колір
голови:</string>
```

```
    <string name="choose_snake_body_color">Колір тіла:</string>
    <string name="choose_snake_food_color">Колір їжі:</string>
    <string name="choose_snake_difficulty">Складність:</string>
    <string name="start_game">Почати</string>
    <string name="cancel_game">Скасувати</string>
    <string name="easy">Легкий</string>
    <string name="medium">Середній</string>
    <string name="hard">Складний</string>
    <string name="title_activity_snake_game">Змійка</string>
    <string name="error_not_send">Помилка: не надіслано</string>
    <string name="back_pressed">Щоб вийти, натисніть назад ще
раз</string>
    <string name="snake_gameover">Гра закінчена! Ваш рахунок:
%1$d</string>
    <string name="score">Ваш рахунок: %1$d</string>
    <string name="fill_effect_name">Заповнити</string>
    <string name="rain_effect_name">Дощ</string>
    <string name="choose_fill_color">Колір:</string>
    <string name="choose_rain_color">Колір:</string>
    <string name="choose_rain_speed">Швидкість:</string>
    <string name="draw_effect_name">Малювання</string>
    <string name="clear">Очистити\n</string>
    <string name="sinus_effect_name">Хвиля</string>
    <string name="rainbow_effect_name">Зміна кольорів</string>
    <string name="choose_sinus_speed">Швидкість:</string>
    <string name="choose_sinus_color">Колір:</string>
    <string name="choose_rainbow_speed">Швидкість:</string>
</resources>
```

Код файлу розмітки preferences_strings.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools">
    <string name="pref_file"
translatable="false">preferences</string>
```

```
    <string name="bluetooth_device_name"
translatable="false">Cube</string>

    <string name="bluetooth_device_address"
translatable="false">bluetooth_device_address</string>

    <string name="string_null"
translatable="false">null</string>


    <!-- GAMES -->

    <!-- Snake -->

    <string name="snake_preferences"
translatable="false">snake_pref</string>

    <string name="snake_body_color"
translatable="false">snake_body_color</string>

    <string name="snake_head_color"
translatable="false">snake_head_color</string>

    <string name="snake_food_color"
translatable="false">snake_food_color</string>

    <string name="snake_difficulty"
translatable="false">snake_difficulty</string>


    <!-- EFFECTS -->

    <!-- Fill -->

    <string name="fill_preferences"
translatable="false">fill_pref</string>

    <string name="fill_color"
translatable="false">fill_color</string>

    <!-- Fire -->

    <string name="rain_preferences"
translatable="false">rain_pref</string>

    <string name="rain_color"
translatable="false">rain_color</string>

    <string name="rain_speed"
translatable="false">rain_speed</string>

    <!-- Rainbow -->

    <string name="rainbow_preferences"
translatable="false">rainbow_pref</string>

    <string name="rainbow_color"
translatable="false">rainbow_color</string>
```

```
    <string name="rainbow_speed"
translatable="false">rainbow_speed</string>

    <!-- Sinus -->

    <string name="sinus_preferences"
translatable="false">sinus_pref</string>

    <string name="sinus_color"
translatable="false">sinus_color</string>

    <string name="sinus_speed"
translatable="false">sinus_speed</string>


    <string name="SERVICE_UUID" tools:ignore="TypographyDashes"
translatable="false">0000FFE0-0000-1000-8000-
00805F9B34FB</string>

    <string name="WRITE_CHARACTERISTIC_UUID"
tools:ignore="TypographyDashes" translatable="false">0000FFE2-
0000-1000-8000-00805F9B34FB</string>

    <string name="READ_CHARACTERISTIC_UUID"
tools:ignore="TypographyDashes,Typos"
translatable="false">0000FFE1-0000-1000-8000-
00805F9B34FB</string>

</resources>
```

Код файлу розмітки activity_choose_device.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    android:layout_width="match_parent"

    android:layout_height="match_parent">


    <TextView

        android:id="@+id/textView"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_marginStart="@dimen/margin"

        android:layout_marginTop="@dimen/margin"

        android:layout_marginEnd="@dimen/margin"

        android:text="@string/choose_device"
```

```
        android:textSize="@dimen/title"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/devices_recyclerView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_marginStart="@dimen/margin"
        android:layout_marginTop="@dimen/margin"
        android:layout_marginEnd="@dimen/margin"
        android:layout_marginBottom="@dimen/margin"
        android:fadeScrollbars="true"
        android:scrollbars="vertical"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код файлу розмітки activity_draw_effect.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Effects.DrawEffectActivity">

    <com.mike.ledcube.Effects.CubeDrawView
        android:id="@+id/draw_field"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код файлу розмітки activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/statusTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/margin"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <com.google.android.material.tabs.TabLayout
        android:id="@+id/main_tablayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintTop_toBottomOf="@+id/statusTextView" />

    <androidx.viewpager2.widget.ViewPager2
        android:id="@+id/main_viewpager"
```

```
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintBottom_toTopOf="@+id/disconnect_button"
app:layout_constraintTop_toBottomOf="@+id/main_tablayout" />

    <Button
        android:id="@+id/disconnect_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="@dimen/margin"
        android:layout_marginBottom="4dp"
        android:text="@string/disconnect"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="@+id/guideline1"
/>

    <Button
        android:id="@+id/connect_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="@dimen/margin"
        android:layout_marginBottom="@dimen/margin"
        android:text="@string/connect"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/guideline1"
        app:layout_constraintStart_toStartOf="parent" />
    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline1"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.5"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код файлу розмітки activity_snake_game.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/snake_game_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/score"
        android:textSize="24sp"
        android:textColor="@color/white"
        android:id="@+id/snake_score_textview"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код файлу розмітки cube_draw_layout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<merge
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"

tools:parentTag="androidx.constraintlayout.widget.ConstraintLayo
ut"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.mike.ledcube.Effects.DrawFieldView
        android:id="@+id/drawFieldView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/drawFieldView">

        <Button
            android:id="@+id/colorButton"
            android:layout_width="73dp"
            android:layout_height="80dp"
            android:layout_marginEnd="@dimen/marginBig"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <NumberPicker
```

```
            android:id="@+id/layerPicker"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toStartOf="@+id/colorButton"
app:layout_constraintStart_toEndOf="@+id/clearButton"
            app:layout_constraintTop_toTopOf="parent" />

        <Button
            android:id="@+id/clearButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:paddingTop="20dp"
            android:text="@string/clear"
            android:textSize="15sp"
            android:layout_marginStart="@dimen/marginBig"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</merge>
```

Код файлу розмітки effect_list_item.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/effect_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
```

```
        android:id="@+id/effect_name_textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="24sp"
        android:layout_marginStart="@dimen/margin"
        android:layout_marginTop="@dimen/margin"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код файлу розмітки fill_dialog_layout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="@dimen/marginBig"
        android:layout_marginTop="@dimen/marginBig"
        android:layout_marginEnd="@dimen/marginBig"
        android:orientation="horizontal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textSize="24sp"
            android:text="@string/choose_fill_color" />
```

```
        <Button
            android:id="@+id/fill_button"
            android:layout_width="41dp"
            android:layout_height="48dp"
            app:cornerRadius="48dp"/>
    </LinearLayout>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginBottom="@dimen/marginBig"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintBottom_toBottomOf="parent">
        <androidx.constraintlayout.widget.Guideline
            android:id="@+id/fill_guideline"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            app:layout_constraintGuide_percent="0.5" />

        <Button
            android:id="@+id/fill_cancel_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/cancel_game"
            android:textSize="15sp"

app:layout_constraintEnd_toStartOf="@+id/fill_guideline"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
```

```
        <Button
            android:id="@+id/fill_ok_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/ok"
            android:textSize="15sp"
            app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="@+id/fill_guideline"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код файлу розмітки fragment_effects.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".EffectsFragment">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/effects_recyclerview"

app:layoutManager="androidx.recyclerview.widget.LinearLayoutMana
ger"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginStart="@dimen/margin"
        android:layout_marginEnd="@dimen/margin"
        android:layout_marginTop="@dimen/margin"
        android:layout_marginBottom="@dimen/margin"
        app:layout_constraintBottom_toBottomOf="parent"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код файлу розмітки fragment_games.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".GamesFragment">



    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/games_recyclerview"

app:layoutManager="androidx.recyclerview.widget.LinearLayoutMana
ger"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginStart="@dimen/margin"
        android:layout_marginEnd="@dimen/margin"
        android:layout_marginTop="@dimen/margin"
        android:layout_marginBottom="@dimen/margin"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код файлу розмітки fragment_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainFragment">

    <com.google.android.material.switchmaterial.SwitchMaterial
        android:id="@+id/on_off_switch"
        android:text="@string/cube_on"
        android:textSize="24sp"
        android:checked="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/margin"
        android:layout_marginStart="@dimen/marginBig"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код файлу розмітки game_list_item.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/game_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/game_name_textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:textSize="24sp"
        android:layout_marginStart="@dimen/margin"
        android:layout_marginTop="@dimen/margin"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код файлу розмітки list_item.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/list_item"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:minHeight="?android:attr/listPreferredItemHeight"

android:paddingStart="?android:attr/listPreferredItemPaddingStar
t"

android:paddingEnd="?android:attr/listPreferredItemPaddingEnd"
    android:clickable="true"
    android:focusable="true"
    android:background="?android:attr/selectableItemBackground"
>

    <TextView
        android:id="@+id/list_item_text1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"

android:textAppearance="?android:attr/textAppearanceListItem" />

    <TextView
        android:id="@+id/list_item_text2"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_below="@id/list_item_text1"
        android:layout_marginBottom="8dp"
        android:layout_alignStart="@id/list_item_text1"

android:textAppearance="?android:attr/textAppearanceListItemSeco
ndary" />
</RelativeLayout>
```

Код файлу розмітки rain_dialog_layout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="@dimen/marginBig"
        android:layout_marginTop="@dimen/marginBig"
        android:layout_marginEnd="@dimen/marginBig"
        android:orientation="horizontal"
        android:id="@+id/rain_color_layout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textSize="24sp"
            android:text="@string/choose_rain_color" />
        <Button
```

```
            android:id="@+id/rain_color_button"
            android:layout_width="41dp"
            android:layout_height="48dp"
            app:cornerRadius="48dp"/>
    </LinearLayout>
    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="@dimen/marginBig"
        android:layout_marginTop="@dimen/marginBig"
        android:layout_marginEnd="@dimen/marginBig"
        android:orientation="horizontal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/rain_color_layout">

        <TextView
            android:id="@+id/textView2"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/choose_rain_speed"
            android:textSize="24sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="1.0" />

        <SeekBar
            android:layout_width="250dp"
            android:layout_height="match_parent"
            android:id="@+id/rain_speed"
```

```
            android:min="0"
            android:max="255"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toEndOf="@+id/textView2"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginBottom="@dimen/marginBig"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintBottom_toBottomOf="parent">
        <androidx.constraintlayout.widget.Guideline
            android:id="@+id/fill_guideline"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            app:layout_constraintGuide_percent="0.5" />

        <Button
            android:id="@+id/rain_cancel_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/cancel_game"
            android:textSize="15sp"

app:layout_constraintEnd_toStartOf="@+id/fill_guideline"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
```

```
        <Button
            android:id="@+id/rain_ok_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/ok"
            android:textSize="15sp"
            app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="@+id/fill_guideline"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код файлу розмітки rainbow_dialog_layout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="@dimen/marginBig"
        android:layout_marginTop="@dimen/marginBig"
        android:layout_marginEnd="@dimen/marginBig"
        android:orientation="horizontal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView
```

```
            android:id="@+id/textView2"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/choose_rainbow_speed"
            android:textSize="24sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="1.0" />

        <SeekBar
            android:layout_width="250dp"
            android:layout_height="match_parent"
            android:id="@+id/rainbow_speed"
            android:min="0"
            android:max="255"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toEndOf="@+id/textView2"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginBottom="@dimen/marginBig"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintBottom_toBottomOf="parent">
        <androidx.constraintlayout.widget.Guideline
            android:id="@+id/fill_guideline"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            app:layout_constraintGuide_percent="0.5" />

        <Button
            android:id="@+id/rainbow_cancel_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/cancel_game"
            android:textSize="15sp"

app:layout_constraintEnd_toStartOf="@+id/fill_guideline"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <Button
            android:id="@+id/rainbow_ok_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/ok"
            android:textSize="15sp"
            app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="@+id/fill_guideline"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код файлу розмітки sinus_dialog_layout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="@dimen/marginBig"
    android:layout_marginTop="@dimen/marginBig"
    android:layout_marginEnd="@dimen/marginBig"
    android:orientation="horizontal"
    android:id="@+id/sinus_color_layout"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textSize="24sp"
        android:text="@string/choose_sinus_color" />
    <Button
        android:id="@+id/sinus_color_button"
        android:layout_width="41dp"
        android:layout_height="48dp"
        app:cornerRadius="48dp"/>
</LinearLayout>
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="@dimen/marginBig"
    android:layout_marginTop="@dimen/marginBig"
    android:layout_marginEnd="@dimen/marginBig"
    android:orientation="horizontal"
    app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@id/sinus_color_layout">

        <TextView
            android:id="@+id/textView2"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/choose_sinus_speed"
            android:textSize="24sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="1.0" />

        <SeekBar
            android:layout_width="250dp"
            android:layout_height="match_parent"
            android:id="@+id/sinus_speed"
            android:min="0"
            android:max="255"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toEndOf="@+id/textView2"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginBottom="@dimen/marginBig"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintBottom_toBottomOf="parent">
        <androidx.constraintlayout.widget.Guideline
            android:id="@+id/fill_guideline"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            app:layout_constraintGuide_percent="0.5" />

        <Button
            android:id="@+id/sinus_cancel_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/cancel_game"
            android:textSize="15sp"

app:layout_constraintEnd_toStartOf="@+id/fill_guideline"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <Button
            android:id="@+id/sinus_ok_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/ok"
            android:textSize="15sp"
            app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="@+id/fill_guideline"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Код файлу розмітки snake_dialog_layout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:id="@+id/snake_head_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="@dimen/marginBig"
        android:layout_marginTop="@dimen/marginBig"
        android:layout_marginEnd="@dimen/marginBig"
        android:orientation="horizontal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textSize="24sp"
            android:text="@string/choose_snake_head_color" />
        <Button
            android:id="@+id/snake_head_button"
            android:layout_width="41dp"
            android:layout_height="48dp"
            app:cornerRadius="48dp"/>
    </LinearLayout>
    <LinearLayout
        android:id="@+id/snake_body_layout"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_marginStart="@dimen/marginBig"
        android:layout_marginTop="@dimen/marginBig"
        android:layout_marginEnd="@dimen/marginBig"
        android:orientation="horizontal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/snake_head_layout">
        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textSize="24sp"
            android:text="@string/choose_snake_body_color" />
        <Button
            android:id="@+id/snake_body_button"
            android:layout_width="41dp"
            android:layout_height="48dp"
            app:cornerRadius="48dp"/>
    </LinearLayout>
    <LinearLayout
        android:id="@+id/snake_food_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="@dimen/marginBig"
        android:layout_marginTop="@dimen/marginBig"
        android:layout_marginEnd="@dimen/marginBig"
        android:orientation="horizontal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/snake_body_layout">
        <TextView
```

```
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textSize="24sp"
            android:text="@string/choose_snake_food_color" />
        <Button
            android:id="@+id/snake_food_button"
            android:layout_width="41dp"
            android:layout_height="48dp"
            app:cornerRadius="48dp" />
    </LinearLayout>
    <LinearLayout
        android:id="@+id/snake_difficulty_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="@dimen/marginBig"
        android:layout_marginTop="@dimen/marginBig"
        android:layout_marginEnd="@dimen/marginBig"
        android:orientation="horizontal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/snake_food_layout">
        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textSize="24sp"
            android:text="@string/choose_snake_difficulty" />
        <Spinner
            android:id="@+id/snake_difficulty_spinner"
            android:layout_height="wrap_content"
            android:layout_width="150dp"
```

```
        android:entries="@array/difficulties"
        />
</LinearLayout>
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="@dimen/marginBig"
    android:orientation="horizontal"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintBottom_toBottomOf="parent">
    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.5" />

    <Button
        android:id="@+id/snake_cancel_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cancel_game"
        android:textSize="15sp"
        app:layout_constraintEnd_toStartOf="@+id/guideline2"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/snake_ok_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/start_game"
```

```
            android:textSize="15sp"
            app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="@+id/guideline2"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

ДОДАТОК Б

Коди додаткових класів

Код класу DrawField:

```
package com.mike.ledcube.Effects;

import android.graphics.Color;

import java.util.HashSet;
import java.util.Set;

interface OnFieldChangedListener {
    void onFieldChange(DrawField field);
}

public class DrawField {
    private final int rows;
    private final int columns;
    private final int[][] cells;

    public DrawField(int rows, int columns) {
        this.rows = rows;
        this.columns = columns;
        cells = new int[rows][];
        for (int i = 0; i < rows; i++) {
            cells[i] = new int[columns];
            for (int j = 0; j < columns; j++) {
                cells[i][j] = Color.BLACK;
            }
        }
    }
```

```
    public Set<OnFieldChangedListener> listeners = new
HashSet<>();

    public int getCell(int row, int column) {

        if (row < 0 || row >= rows || column < 0 || column >=
columns) return 0;

        return cells[row][column];

    }


    public void setCell(int row, int column, int cell) {

        if (row < 0 || row >= rows || column < 0 || column >=
columns) return;

        if (cells[row][column] != cell) {

            cells[row][column] = cell;

            for (OnFieldChangedListener listener : listeners)

                listener.onFieldChange(this);

        }

    }


    public void setCell(int row, int column, int cell, boolean
notify) {

        if (notify) setCell(row, column, cell);

        else {

            if (row < 0 || row >= rows || column < 0 || column
>= columns) return;

            if (cells[row][column] != cell) {

                cells[row][column] = cell;

            }

        }

    }


    public int getColumns() {

        return columns;

    }


    public int getRows() {
```

```
        return rows;
    }
    public void clear() {
        for (int i = 0; i < rows; i++) {
            cells[i] = new int[columns];
            for (int j = 0; j < columns; j++) {
                cells[i][j] = Color.BLACK;
            }
        }
        for (OnFieldChangedListener listener : listeners)
            listener.onFieldChange(this);
    }
}
```

Код класу BluetoothConnectionHelper:

```
package com.mike.ledcube;

import android.bluetooth.BluetoothGattCharacteristic;
import android.content.Context;
import android.os.Handler;
import android.os.Looper;

import androidx.annotation.NonNull;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;

import com.welie.blessed.BluetoothCentralManager;
import com.welie.blessed.BluetoothCentralManagerCallback;
import com.welie.blessed.BluetoothPeripheral;
import com.welie.blessed.BluetoothPeripheralCallback;
import com.welie.blessed.GattStatus;
import com.welie.blessed.HciStatus;
import com.welie.blessed.WriteType;
```

```
import java.nio.charset.StandardCharsets;
import java.util.UUID;


public class BluetoothConnectionHelper {
    private final UUID SERVICE_UUID;


    private final MutableLiveData<Event<char[]>> messagesData =
new MutableLiveData<>();
    private final MutableLiveData<ConnectionStatus>
connectionStatusData = new MutableLiveData<>();


    private final String deviceAddress;
    private final UUID WRITE_CHARACTERISTIC_UUID;
    private final BluetoothPeripheralCallback peripheralCallback
= new BluetoothPeripheralCallback() {
        @Override
        public void onCharacteristicUpdate(@NonNull
BluetoothPeripheral peripheral, @NonNull byte[] value,
                                           @NonNull
BluetoothGattCharacteristic characteristic, @NonNull GattStatus
status) {
            super.onCharacteristicUpdate(peripheral, value,
characteristic, status);
            if (status == GattStatus.SUCCESS) {
                messagesData.postValue(new Event<>(new
String(value, StandardCharsets.US_ASCII).toCharArray()));
            }
        }
    };
    BluetoothCentralManager central;


    enum ConnectionStatus {
        DISCONNECTED,
        CONNECTING,
        CONNECTED
```

```
    }

    public BluetoothConnectionHelper(Context context, String
deviceAddress) {

        this.deviceAddress = deviceAddress;

        BluetoothCentralManagerCallback
bluetoothCentralManagerCallback = new
BluetoothCentralManagerCallback() {

            @Override

            public void onConnectedPeripheral(@NonNull
BluetoothPeripheral peripheral) {

                super.onConnectedPeripheral(peripheral);


connectionStatusData.postValue(ConnectionStatus.CONNECTED);

                BluetoothGattCharacteristic readChar =
peripheral.getCharacteristic(UUID.fromString(context.getString(R
.string.SERVICE_UUID)),


UUID.fromString(context.getString(R.string.READ_CHARACTERISTIC_U
UID)));

                if (readChar != null) {

                    peripheral.setNotify(readChar, true);

                }

            }


            @Override

            public void onConnectingPeripheral(@NonNull
BluetoothPeripheral peripheral) {

                super.onConnectingPeripheral(peripheral);


connectionStatusData.postValue(ConnectionStatus.CONNECTING);

            }


            @Override

            public void onConnectionFailed(@NonNull
BluetoothPeripheral peripheral, @NonNull HciStatus status) {

                super.onConnectionFailed(peripheral, status);


connectionStatusData.postValue(ConnectionStatus.DISCONNECTED);
```

```
            }
        };
        central = new BluetoothCentralManager(context,
bluetoothCentralManagerCallback, new
Handler(Looper.getMainLooper()));
        SERVICE_UUID =
UUID.fromString(context.getString(R.string.SERVICE_UUID));
        WRITE_CHARACTERISTIC_UUID =
UUID.fromString(context.getString(R.string.WRITE_CHARACTERISTIC_
UUID));
        connect();

connectionStatusData.postValue(ConnectionStatus.DISCONNECTED);
    }

    public boolean send(char[] data) {
        return
central.getPeripheral(deviceAddress).writeCharacteristic(SERVICE
_UUID, WRITE_CHARACTERISTIC_UUID,
                new String(data).getBytes(),
WriteType.WITH_RESPONSE);//maybe without_response
    }

    public void connect() {

central.autoConnectPeripheral(central.getPeripheral(deviceAddres
s), peripheralCallback);
    }

    public void disconnect() {

central.cancelConnection(central.getPeripheral(deviceAddress));
    }
    public LiveData<Event<char[]>> getMessages() {
        return messagesData;
    }
    public LiveData<ConnectionStatus> getConnectionStatus() {
        return connectionStatusData;
```

```
    }
}
```

Код класу Event<T>:

```
package com.mike.ledcube;

import androidx.annotation.Nullable;

public class Event<T> {

    private final T mContent;

    private boolean hasBeenHandled = false;

    public Event(T content) {
        if (content == null) {
            throw new IllegalArgumentException("null values in
Event are not allowed.");
        }
        mContent = content;
    }

    @Nullable
    public T getContentIfNotHandled() {
        if (hasBeenHandled) {
            return null;
        } else {
            hasBeenHandled = true;
            return mContent;
        }
    }

    public boolean hasBeenHandled() {
        return hasBeenHandled;
```

```
        }
    }
```

ДОДАТОК В

Код мікроконтролера

```
#define PREFERENCES_CODE 0
#define PREFERENCE_STATE_CODE 0
#define EFFECTS_CODE 1
#define EFFECT_FILL_CODE 0
#define EFFECT_DRAW_CODE 2
#define EFFECT_SINUS_FILL_CODE 3
#define EFFECT_RAIN_CODE 4
#define EFFECT_RAINBOW_CODE 5

#define GAMES_CODE 2
#define GAME_SNAKE_CODE 0

#define EOL ';'
#define DELIMETER ','

#define BLUETOOTH_SPEED 115200

#define LAYERS 8
#define ROWS 8
#define COLUMNS 8

// continues effects/games
#define EFFECT_NONE 0
#define EFFECT_DRAW 2
#define EFFECT_SINUS 3
#define EFFECT_RAIN 4
#define EFFECT_RAINBOW 5
#define GAME_SNAKE 6

#define SNAKE_START 4
```

```
#define D_UP 0
#define D_DOWN 1
#define D_LEFT 2
#define D_RIGHT 3
#define D_FORWARD 4
#define D_BACKWARD 5

// pins
#define LAYER1_PIN 9
#define LAYER2_PIN 8
#define LAYER3_PIN 7
#define LAYER4_PIN 6
#define LAYER5_PIN 5
#define LAYER6_PIN 4
#define LAYER7_PIN 3
#define LAYER8_PIN 2

#include <ledMatrix.h>

bool state = true;
uint8_t currentEffect = EFFECT_NONE;
uint32_t millTimer;

ledMatrix<ROWS, COLUMNS, LAYER1_PIN> layer1;
ledMatrix<ROWS, COLUMNS, LAYER2_PIN> layer2;
ledMatrix<ROWS, COLUMNS, LAYER3_PIN> layer3;
ledMatrix<ROWS, COLUMNS, LAYER4_PIN> layer4;
ledMatrix<ROWS, COLUMNS, LAYER5_PIN> layer5;
ledMatrix<ROWS, COLUMNS, LAYER6_PIN> layer6;
ledMatrix<ROWS, COLUMNS, LAYER7_PIN> layer7;
ledMatrix<ROWS, COLUMNS, LAYER8_PIN> layer8;

void clear()
```

```
{
  layer1.fill(0);
  layer2.fill(0);
  layer3.fill(0);
  layer4.fill(0);
  layer5.fill(0);
  layer6.fill(0);
  layer7.fill(0);
  layer8.fill(0);
}

uint16_t effectColor;
boolean loading = true;
void fillColor()
{
  layer1.fill(effectColor);
  layer2.fill(effectColor);
  layer3.fill(effectColor);
  layer4.fill(effectColor);
  layer5.fill(effectColor);
  layer6.fill(effectColor);
  layer7.fill(effectColor);
  layer8.fill(effectColor);
  draw();
}

void drawEffect(uint8_t layer, uint8_t row, uint8_t column,
uint16_t color)
{
  if (loading && currentEffect != EFFECT_DRAW)
  {
    clear();
    draw();
```

```
  currentEffect = EFFECT_DRAW;
  loading = false;
}
switch (layer)
{
case 0 ... 7:
  setPixel(layer, row, column, color);
  break;
case 9:
  switch (column)
  {
  case 0:
    layer1.fill(0);
    break;
  case 1:
    layer2.fill(0);
    break;
  case 2:
    layer3.fill(0);
    break;
  case 3:
    layer4.fill(0);
    break;
  case 4:
    layer5.fill(0);
    break;
  case 5:
    layer6.fill(0);
    break;
  case 6:
    layer7.fill(0);
    break;
  case 7:
```

```
      layer8.fill(0);
      break;
    }
  }
  draw();
}

void setup()
{
  Serial.begin(BLUETOOTH_SPEED);
  off();
}

void loop()
{
  for (;;)
  {
    if (Serial.available())
    {
      parseCommand();
    }

    switch (currentEffect)
    {
    case EFFECT_RAINBOW:
    {
      rainbow();
      break;
    }
    case EFFECT_SINUS:
    {
      sinusFill();
      break;
```

```
    }
    case EFFECT_RAIN:
    {
      rain();
      break;
    }
    case GAME_SNAKE:
    {
      snake();
      break;
    }
    }
  }
}

void sendState()
{
  char *temp = new char[7];
  temp[0] = PREFERENCES_CODE + '0';
  temp[1] = DELIMETER;
  temp[2] = PREFERENCE_STATE_CODE + '0';
  temp[3] = DELIMETER;
  temp[4] = state ? '1' : '0';
  temp[5] = EOL;
  temp[6] = '\0';
  Serial.write(temp);
  delete[] temp;
}

void setPixel(int layer, int row, int column, uint16_t color)
{
  switch (layer)
  {
```

```
  case 0:
    layer1.set(column, row, color);
    break;
  case 1:
    layer2.set(column, row, color);
    break;
  case 2:
    layer3.set(column, row, color);
    break;
  case 3:
    layer4.set(column, row, color);
    break;
  case 4:
    layer5.set(column, row, color);
    break;
  case 5:
    layer6.set(column, row, color);
    break;
  case 6:
    layer7.set(column, row, color);
    break;
  case 7:
    layer8.set(column, row, color);
    break;
  }
}

void sendScore(uint8_t score, boolean isGameOver = false)
{
  uint8_t size;
  switch (score)
  {
  case 0 ... 9:
```

```
    size = 1;
    break;
  case 10 ... 99:
    size = 2;
    break;
  case 100 ... 255:
    size = 3;
    break;
  }
  char *temp = new char[size + 8];
  temp[0] = GAMES_CODE + '0';
  temp[1] = DELIMETER;
  temp[2] = GAME_SNAKE_CODE + '0';
  temp[3] = DELIMETER;
  temp[4] = isGameOver ? '0' : '1';
  temp[5] = DELIMETER;
  for (int i = size - 1; i >= 0; --i)
  {
    temp[i + 6] = (score % 10) + '0';
    score /= 10;
  }
  temp[size + 6] = EOL;
  temp[size + 7] = '\0';
  Serial.write(temp);
  delete[] temp;
}

int8_t snakeArray[SNAKE_START + 100][3];
uint8_t snakeLength;
bool generateApple;
uint8_t apple[3];
enum direction
{
```

```
  UP,
  DOWN,
  LEFT,
  RIGHT,
  FORWARD,
  BACKWARD,
  STOP
} dir;
uint16_t appleColor;
uint16_t headColor;
uint16_t snakeColor;
uint16_t speed = 8000;

void snake()
{
  if (loading)
  {
    clear();
    draw();
    loading = false;
    dir = RIGHT;
    currentEffect = GAME_SNAKE;
    snakeLength = SNAKE_START;
    generateApple = true;
    for (uint8_t i = 0; i < SNAKE_START; i++)
    {
      snakeArray[i][0] = SNAKE_START - i - 1;
      snakeArray[i][1] = 0;
      snakeArray[i][2] = 7;
      setPixel(7, 0, SNAKE_START - i - 1, (i == 0 ? headColor :
snakeColor));
    }
    millTimer = millis();
```

```
    draw();
  }

  while (generateApple)
  {
    apple[0] = random(0, 8);
    apple[1] = random(0, 8);
    apple[2] = random(0, 8);
    for (uint8_t i = 0; i < snakeLength; i++)
    {
      if (snakeArray[i][0] == apple[0] && snakeArray[i][1] ==
apple[1] && snakeArray[i][2] == apple[2])
        break;
      if (i == snakeLength - 1)
      {
        generateApple = false;
        setPixel(apple[2], apple[1], apple[0], appleColor);
        draw();
      }
    }
  }

  if (millis() - millTimer >= speed)
  {
    millTimer = millis();
    uint8_t xHeadPrev = snakeArray[0][0];
    uint8_t yHeadPrev = snakeArray[0][1];
    uint8_t zHeadPrev = snakeArray[0][2];
    switch (dir)
    {
    case UP:
      snakeArray[0][2]++;
      break;
```

```
case DOWN:
  snakeArray[0][2]--;
  break;
case LEFT:
  snakeArray[0][0]--;
  break;
case RIGHT:
  snakeArray[0][0]++;
  break;
case FORWARD:
  snakeArray[0][1]++;
  break;
case BACKWARD:
  snakeArray[0][1]--;
  break;
}

// check apple
if (snakeArray[0][0] == apple[0] &&
    snakeArray[0][1] == apple[1] &&
    snakeArray[0][2] == apple[2])
{
  snakeLength++;
  for (uint8_t i = snakeLength - 1; i > 1; i--)
  {
    snakeArray[i][0] = snakeArray[i - 1][0];
    snakeArray[i][1] = snakeArray[i - 1][1];
    snakeArray[i][2] = snakeArray[i - 1][2];
  }

  snakeArray[1][0] = xHeadPrev;
  snakeArray[1][1] = yHeadPrev;
  snakeArray[1][2] = zHeadPrev;
```

```
      generateApple = true;
    }
    else
    {
      setPixel(snakeArray[snakeLength - 1][2],
snakeArray[snakeLength - 1][1], snakeArray[snakeLength - 1][0],
0);
      for (uint8_t i = snakeLength - 1; i > 1; i--)
      {
        snakeArray[i][0] = snakeArray[i - 1][0];
        snakeArray[i][1] = snakeArray[i - 1][1];
        snakeArray[i][2] = snakeArray[i - 1][2];
      }
      snakeArray[1][0] = xHeadPrev;
      snakeArray[1][1] = yHeadPrev;
      snakeArray[1][2] = zHeadPrev;
    }

    // check collide with walls
    if (snakeArray[0][0] < 0 || snakeArray[0][0] > 7 ||
        snakeArray[0][1] < 0 || snakeArray[0][1] > 7 ||
        snakeArray[0][2] < 0 || snakeArray[0][2] > 7)
    {
      currentEffect = EFFECT_NONE;
      sendScore(snakeLength - SNAKE_START, true);
      loading = true;
      off();
      delay(500);
      draw();
      delay(500);
      off();
      delay(500);
      draw();
      delay(500);
```

```
  off();
  delay(500);
  draw();
  return;
}

// check self collide
for (uint8_t i = 2; i < snakeLength; i++)
{
  if (snakeArray[0][0] == snakeArray[i][0] &&
      snakeArray[0][1] == snakeArray[i][1] &&
      snakeArray[0][2] == snakeArray[i][2])
  {
    currentEffect = EFFECT_NONE;
    sendScore(snakeLength - SNAKE_START, true);
    loading = true;
    off();
    delay(500);
    draw();
    delay(500);
    off();
    delay(500);
    draw();
    delay(500);
    off();
    delay(500);
    draw();
    return;
  }
}

// win
if (snakeLength == SNAKE_START + 100)
```

```
    {
      currentEffect = EFFECT_NONE;
      sendScore(snakeLength - SNAKE_START, true);
      loading = true;
      off();
      delay(500);
      draw();
      delay(500);
      off();
      delay(500);
      draw();
      delay(500);
      off();
      delay(500);
      draw();
      return;
    }


    setPixel(snakeArray[0][2], snakeArray[0][1],
snakeArray[0][0], headColor);
    setPixel(snakeArray[1][2], snakeArray[1][1],
snakeArray[1][0], snakeColor);
    sendScore(snakeLength - SNAKE_START);
    draw();
  }
}


int8_t pos = 0;
void sinusFill()
{
  if (loading)
  {
    currentEffect = EFFECT_SINUS;
    clear();
```

```
    loading = false;
    millTimer = millis();
  }
  if (millis() - millTimer >= speed)
  {
    millTimer = millis();
    clear();
    if (++pos > 10)
      pos = 0;
    for (uint8_t i = 0; i < 8; i++)
    {
      for (uint8_t j = 0; j < 8; j++)
      {
        int8_t sinZ = 4 + ((float)sin((float)(i + pos) / 2) *
3);
        for (uint8_t y = 0; y < sinZ; y++)
        {
          setPixel(y, j, i, effectColor);
        }
      }
    }
    draw();
  }
}

void rain()
{
  if (loading)
  {
    currentEffect = EFFECT_RAIN;
    clear();
    loading = false;
    millTimer = millis();
```

```
  }
  if (millis() - millTimer >= speed)
  {
    millTimer = millis();
    for (int i = 0; i < COLUMNS * ROWS; i++)
    {
      layer1.leds[i] = layer2.leds[i];
      layer2.leds[i] = layer3.leds[i];
      layer3.leds[i] = layer4.leds[i];
      layer4.leds[i] = layer5.leds[i];
      layer5.leds[i] = layer6.leds[i];
      layer6.leds[i] = layer7.leds[i];
      layer7.leds[i] = layer8.leds[i];
      layer8.leds[i] = 0;
    }

    uint8_t numDrops = random(0, 5);
    for (uint8_t i = 0; i < numDrops; i++)
    {
      setPixel(7, random(0, 8), random(0, 8), effectColor);
    }
    draw();
  }
}

void rainbow()
{
  if (loading)
  {
    currentEffect = EFFECT_RAINBOW;
    loading = false;
    millTimer = millis();
    clear();
```

```
    draw();
    pos = 0;
  }
  if (millis() - millTimer > speed)
  {
    millTimer = millis();
    uint16_t color = getHSV(pos, 255, 255);
    if (pos >= 255)
      pos = 0;
    pos += 1;
    layer1.fill(color);
    layer2.fill(color);
    layer3.fill(color);
    layer4.fill(color);
    layer5.fill(color);
    layer6.fill(color);
    layer7.fill(color);
    layer8.fill(color);
  }
  draw();
}

void parseCommand()
{
  char buf[44];
  uint8_t amount = Serial.readBytesUntil(EOL, buf, 44);
  buf[amount] = 0;
  uint8_t data[12];
  uint8_t count = 0;
  char *offset = buf;

  while (true)
  {
```

```
  data[count++] = atoi(offset);
  offset = strchr(offset, DELIMETER);
  if (offset)
    offset++;
  else
    break;
}

// getData
if (count == 2)
{
  switch (data[0])
  {
  case PREFERENCES_CODE:
    switch (data[1])
    {
    case PREFERENCE_STATE_CODE:
      sendState();
      break;
    }
    break;
  }
}
else // setData
{
  switch (data[0])
  {
  case PREFERENCES_CODE:
    switch (data[1])
    {
    case PREFERENCE_STATE_CODE:
      state = data[2];
      if (state)
```

```
      draw();
    else
      off();
    break;
  default:
    return;
  }
  break;
case EFFECTS_CODE:
  switch (data[1])
  {
  case EFFECT_FILL_CODE:
    loading = true;
    effectColor = getRGB(data[2], data[3], data[4]);
    fillColor();
    break;
  case EFFECT_RAINBOW_CODE:
    loading = true;
    speed = 510 - (data[2] << 1);
    rainbow();
    break;
  case EFFECT_SINUS_FILL_CODE:
    effectColor = getRGB(data[2], data[3], data[4]);
    speed = 127 - (data[5] >> 1);
    loading = true;
    sinusFill();
    break;
  case EFFECT_RAIN_CODE:
    effectColor = getRGB(data[2], data[3], data[4]);
    speed = 255 - data[5];
    loading = true;
    rain();
    break;
```

```
      case EFFECT_DRAW_CODE:
        loading = true;
        drawEffect(data[2], data[4], data[3], getRGB(data[5],
data[6], data[7]));
      default:
        return;
      }
      break;
    case GAMES_CODE:
      switch (data[1])
      {
      case GAME_SNAKE_CODE:
      {
        if (count == 3)
        {
          switch (data[2])
          {
          case D_UP:
            if (dir != DOWN)
            {
              dir = UP;
            }
            break;
          case D_DOWN:
            if (dir != UP)
            {
              dir = DOWN;
            }
            break;
          case D_LEFT:
            if (dir != RIGHT)
            {
              dir = LEFT;
```

```
    }
    break;
  case D_RIGHT:
    if (dir != LEFT)
    {
      dir = RIGHT;
    }
    break;
  case D_BACKWARD:
    if (dir != FORWARD)
    {
      dir = BACKWARD;
    }
    break;
  case D_FORWARD:
    if (dir != BACKWARD)
    {
      dir = FORWARD;
    }
    break;
  }
  break;
}
loading = true;
headColor = getRGB(data[2], data[3], data[4]);
snakeColor = getRGB(data[5], data[6], data[7]);
appleColor = getRGB(data[8], data[9], data[10]);
switch (data[11])
{
case 0:
  speed = 1000;
  break;
case 1:
```

```
          speed = 700;
          break;
        case 2:
          speed = 500;
          break;
        }
        snake();
        break;
      }
      default:
        return;
      }
      break;
    default:
      return;
    }
  }
}

void off()
{
  layer1.show(0);
  layer2.show(0);
  layer3.show(0);
  layer4.show(0);
  layer5.show(0);
  layer6.show(0);
  layer7.show(0);
  layer8.show(0);
}

void draw()
{
```

```
  if (!state)
    return;
  layer1.show();
  layer2.show();
  layer3.show();
  layer4.show();
  layer5.show();
  layer6.show();
  layer7.show();
  layer8.show();
}
```

ДОДАТОК Г

Код бібліотеки ledMatrix.h

```
#ifndef _ledMatrix_h
#define _ledMatrix_h

#include <Arduino.h>

// convert from 16-bit color value to RGB values
#define getR(x) (((x) & 0b1111100000000000) >> 11)
#define getG(x) (((x) & 0b0000011111000000) >> 6)
#define getB(x) (((x) & 0b0000000000111110) >> 1)

// convert RGB values to 16-bit color value
#define getRGB(r, g, b) ((((r) & 0b11111000) << 8) | (((g) & 0b11111000) << 3) | (((b) & 0b11111000) >> 2))
// convert HSV values to 16-bit color value
uint16_t getHSV(uint8_t h, uint8_t s, uint8_t v)
{
    float r, g, b;

    float H = h / 255.0;
    float S = s / 255.0;
    float V = v / 255.0;

    int i = int(H * 6);
    float f = H * 6 - i;
    float p = V * (1 - S);
    float q = V * (1 - f * S);
    float t = V * (1 - (1 - f) * S);

    switch (i % 6)
    {
```

```
    case 0:
        r = V, g = t, b = p;
        break;
    case 1:
        r = q, g = V, b = p;
        break;
    case 2:
        r = p, g = V, b = t;
        break;
    case 3:
        r = p, g = q, b = V;
        break;
    case 4:
        r = t, g = p, b = V;
        break;
    case 5:
        r = V, g = p, b = q;
        break;
    }
    r *= 255.0;
    g *= 255.0;
    b *= 255.0;
    return getRGB((byte)r, (byte)g, (byte)b);
}

// led matrix class
template <uint8_t width, uint8_t height, int8_t pin>
class ledMatrix
{
public:
    uint16_t leds[width * height]; // array to store color
values for each led in matrix
```

```
    // prepare pin
    void init()
    {
        pin_mask = digitalPinToBitMask(pin);                    //
get bit mask of pin
        pin_port = portOutputRegister(digitalPinToPort(pin)); //
get register of port states
        port_ddr = portModeRegister(digitalPinToPort(pin));   //
get register of port I/O configurations
        *port_ddr |= pin_mask;                                  //
set pin for output (0 - in, 1 - out)
    }

    // constructor
    ledMatrix()
    {
        init();
    }

    // fill all leds with color
    void fill(uint16_t color)
    {
        for (uint8_t i = 0; i < width * height; i++)
            leds[i] = color;
    }

    // get number of pixel in array by its coordinates
    uint8_t getNumberOfPixel(uint8_t x, uint8_t y)
    {
        if (y % 2)
            return (height * width - y * width - x - 1);    //
if even row
        else
            return (height * width - y * width - width + x);//
if odd row
```

```
    }

    // set color for pixel
    void set(uint8_t x, uint8_t y, uint16_t color)
    {
        if (x < 0 || y < 0 || x >= width || y >= height) //
check if coordinates are out of range
            return;
        leds[getNumberOfPixel(x, y)] = color;
    }

    // get color by coordinates
    uint16_t get(uint8_t x, uint8_t y)
    {
        return leds[getNumberOfPixel(x, y)];
    }

    // show all pixels
    void show()
    {
        mask_h = pin_mask | *pin_port;  // mask of high level
        mask_l = ~pin_mask & *pin_port; // mask of low level
        for (uint8_t i = 0; i < width * height; i++)
            send(leds[i]);  // send led
    }

    void show(uint16_t color)
    {
        mask_h = pin_mask | *pin_port;  // mask of high level
        mask_l = ~pin_mask & *pin_port; // mask of low level
        for (uint8_t i = 0; i < width * height; i++)
            send(color); // send led
    }
```

```
    // send led color
    void send(uint16_t color)
    {
        // convert from 16-bit color value to RGB
        uint8_t data[3];
        data[0] = getR(color);
        data[1] = getG(color);
        data[2] = getB(color);

        cli(); // turn off interrupts for sending
        // send RGB
        for (uint8_t i = 0; i < 3; i++)
            sendValue(data[i]);
        sei(); // turn от interrupts
    }

    // send color value
    void sendValue(byte data)
    {
        asm volatile(
            "LDI r24, 8         \n\t" // write to register r24
value 8 - loop counter
            "LOOP_START:        \n\t" // label of loop start
            "ST X, %[SET_H]     \n\t" // set high signal on pin

            "SBRS %[DATA], 7    \n\t" // if last bit is equal 1
skip setting pin to low
            "ST X, %[SET_L]     \n\t" // set low signal on pin -
-4 ticks = 250ns
            "LSL  %[DATA]       \n\t" // shift data by 1

            // delay 4 ticks: 1 cycle of 3 ticks, write value of
1 tick
```

```
            "LDI r25, 4         \n\t" // write to register r25
value 1
            "DELAY_LOOP:        \n\t" // label of delay loop
            "DEC r25            \n\t" // decrement - 1 tick
            "BRNE DELAY_LOOP    \n\t" // return to label - 2
ticks

            "ST X, %[SET_L]     \n\t" // set low signal on pin -
-9tics = 562.5ns
            "DEC r24            \n\t" // decrement loop counter
            "BRNE LOOP_START    \n\t" // return to loop start
            :                         // no output variabless
            : [DATA] "r"(data),       // input variables
              [SET_H] "r"(mask_h),
              [SET_L] "r"(mask_l),
              "x"(pin_port)           // write to X port for
data output
            : "r24", "r25");          // used registers
    }

private:
    volatile uint8_t *pin_port, *port_ddr;
    uint8_t pin_mask;
    uint8_t mask_h, mask_l;
};
#endif
```