Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Інтерактивна гра для мікроконтролерів STM32
з використанням графічного дисплея

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-3
Ілля ДАНИЛОВ
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

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

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: доц. Олексій ПІСКАРЬОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

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

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Данилову Іллі Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Інтерактивна гра для мікроконтролерів STM32 з використанням графічного дисплея

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи 1) мікроконтролер: STM32 серії F3; 2) обсяг пам’яті: 64 кБ; 3) роздільна здатність дисплея: 320 на 240 точок; 4) протокол передачі даних: SPI.

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз існуючих рішень
2) визначення вимог до пристрою
3) вибір та обґрунтування електронних компонентів
4) розробка програмної частини системи
5) перевірка працездатності пристрою
6) аналіз результатів

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 18 слайдів

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	27.05.25-30.05.25	
2	Вибір та обґрунтування електронних модулів	31.05.25-01.06.25	
3	Розробка програмного забезпечення	02.06.25-05.06.25	
4	Перевірка працездатності пристрою та аналіз	07.06.25-08.06.25	
	результатів		
5	Оформлення матеріалів кваліфікаційної роботи	10.06.25-11.06.25	
	Подання кваліфікаційної роботи керівникові та	12.06.25-13.06.25	
	її попередній захист		
	Подання кваліфікаційної роботи на рецензу-	14.06.25-16.06.25	
	вання		

Дата видачі завдання " 26 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Олексій ПІСКАРЬОВ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 62 с., 14 рис., 2 дод., 16 джерел.

МІКРОКОНТРОЛЕР, ДИСПЛЕЙ, ГРА, STM32, SPI, DMA, ГРАФІКА, СЕНСОРНИЙ ЕКРАН, ЗВУК.

Метою кваліфікаційної роботи є розробка інтерактивної електронної гри для мікроконтролерів STM32 з використанням сенсорного графічного дисплея.

У ході виконання кваліфікаційної роботи було здійснено повний цикл проєктування пристрою: проаналізовано предметну область, сформульовано вимоги до системи з урахуванням її цільового призначення. Здійснено обґрунтований вибір основних апаратних компонентів.

Програмну частину реалізовано мовою програмування C з використанням інструментів та бібліотек, наданих компанією STMicroelectronics, зокрема STM32CubeIDE та STM HAL. Розроблено алгоритм роботи гри та реалізовано ключові функції пристрою. Застосовано методи оптимізації обміну даними з дисплеєм, що дозволило досягти вимог щодо швидкодії системи.

Апаратну реалізацію проєкту здійснено на базі відлагоджувальної плати NUCLEO-F303RE. Після складання прототипу проведено тестування працездатності пристрою, перевірено відповідність функціоналу технічним вимогам, а також виконано аналіз результатів експериментальної перевірки.

ABSTRACT

Bachelor’s thesis: 62 pages, 14 figures, 2 appendices, 16 sources.

MICROCONTROLLER, DISPLAY, GAME, STM32, SPI, DMA, GRAPHICS, TOUCHSCREEN, SOUND.

The major goal of this thesis is to develop an interactive electronic game for STM32 microcontrollers with a touch-sensitive graphical display.

In order to achieve this, a full cycle of device design was carried out: the subject area was analyzed, the requirements for the system were formulated taking into account its intended purpose. A reasoned choice of the main hardware components was made.

The software component was implemented using the C programming language with tools and libraries provided by STMicroelectronics, in particular STM32CubeIDE and STM HAL. The game operation algorithm was developed and the key functions of the device were implemented. Methods for optimizing the display data transfer were applied, which allowed achieving the performance requirements.

The hardware implementation of the project was carried out on the basis of the NUCLEO-F303RE development board. After assembling the prototype, the device's performance was tested, the functionality was checked for compliance with the technical requirements, and the results of experimental testing were analyzed.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ 7
ВСТУП 8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 10
1.1 Огляд існуючих технічних рішень 10
1.2 Визначення функціональних вимог до пристрою 13
1.3 Інтерфейс користувача та взаємодії з пристроєм 15
2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ 18
2.1 Вибір та обґрунтування мікроконтролера 18
2.2 Вибір дисплея та допоміжних модулів 20
2.3 Живлення та безпека пристрою 22
2.4 Структурна схема пристрою 22
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 24
3.1 Огляд мови програмування та засобів розробки 24
3.2 Алгоритм роботи програми 25
3.3 Вибір бібліотек, драйверів та стеків протоколів 27
3.4 Реалізація ключових функцій 28
4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА РЕЗУЛЬТАТИ 41
4.1 Перевірка працездатності пристрою та гри 41
4.2 Аналіз результатів роботи пристрою 42
4.3 Порівняння з аналогами 44
4.4 Потенційні напрями оптимізації 45
ВИСНОВКИ 46
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 47
ДОДАТОК А ГРАФІЧНИЙ МАТЕРІАЛ КВАЛІФІКАЦІЙНОЇ РОБОТИ 49
ДОДАТОК Б ПУБЛІКАЦІЯ ЗА ТЕМОЮ РОБОТИ 59

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПДП – прямий доступ до пам’яті

ЦАП – цифро-аналоговий перетворювач

FPS – кадри на секунду (англ., Frames per Second)

GPIO – введення/виведення загального призначення (англ., General-Purpose Input/Output)

PCM - імпульсно-кодова модуляція (англ., Pulse Code Modulation)

SoC – система на кристалі (англ., System on Chip)

SRAM – статична пам’ять з довільним доступом (англ., Static Random Access Memory)

ВСТУП

Сучасний світ неможливо уявити без мікроконтролерів – компактних, енергоефективних і надзвичайно гнучких пристроїв, які є основою більшості вбудованих систем. Вони знаходять широке застосування у побутовій техніці, автомобільній електроніці, медичному обладнанні, системах автоматизації та, звісно, в інтерактивних пристроях, включно з іграми та навчальними модулями. Стрімкий розвиток мікроконтролерів дозволяє реалізовувати проєкти все більшої складності на основі доступних апаратних платформ із довершеним інструментарієм

Одним із популярних представників таких платформ є серія STM32 від компанії STMicroelectronics. Мікроконтролери цієї серії вирізняються високою продуктивністю, широкими комунікаційними можливостями, наявністю апаратної підтримки сучасних протоколів, а також доступною вартістю. Поєднання цих факторів робить STM32 оптимальним вибором для побудови пристроїв, які потребують не лише обчислювальної потужності, а й ефективної взаємодії з користувачем через деталізований графічний інтерфейс.

Реалізація гри як прикладу інтерактивного пристрою – це не лише спосіб продемонструвати технічні можливості мікроконтролера, а й зручний інструмент для набуття навичок з ефективності обробки графіки, роботи з швидкісними периферійними пристроями та оптимізації програмного коду в умовах обмежених апаратних ресурсів. Крім того, створення ігрового застосунку для апаратного забезпечення вбудованих систем дозволяє на практиці випробувати концепції реального часу, обробки подій, таймерів і взаємодії між різними модулями системи.

Метою даної кваліфікаційної роботи є розробка інтерактивної гри з використанням сенсорного дисплея та мікроконтролера STM32. Для забезпечення графічного виводу в даній роботі використовується кольоровий

TFT-дисплей з контролером ILI9341. Програмна частина реалізована мовою C з урахуванням особливостей розробки програмного забезпечення для вбудованих систем.

У межах роботи проведено аналіз існуючих технічних рішень, сформульовано вимоги до пристрою, обґрунтовано вибір апаратних засобів, розроблено програмну частину системи, а також проведено експериментальні дослідження для підтвердження працездатності.. Практична значущість проєкту полягає у демонстрації можливостей сучасних мікроконтролерів у сфері створення нескладних ігор. Отримані при розробці системи навички можуть бути адаптовані для розробки інших вбудованих систем, що потребують обробки подій в реальному часі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих технічних рішень

У сучасному світі мікроконтролерні системи стали основою для розробки широкого спектру електронних пристроїв, зокрема інтерактивних ігор та розважальних застосунків. Стрімкий розвиток вбудованих систем, ринку різноманітних електронних модулів і доступність мікроконтролерів, таких як Arduino, PIC, STM32 та інших, значно розширили можливості розробників у створенні інтерактивних пристроїв із графічним інтерфейсом, зокрема, ігрових. У цьому підрозділі розглянуто існуючі технічні рішення, на яких реалізуються інтерактивні ігрові пристрої.

У витоків портативних цифрових розваг знаходяться електронні ігри. Це пристрої, обладнані LCD-дисплеєм та елементами керування, такими як кнопки. Використовувані дисплеї зазвичай не дозволяли відображати довільні зображення, натомість вони складалися з попередньо визначених піктограм [1]. Через це обмеження ці пристрої були призначені для відтворення лише однієї гри. Історичним прикладом подібних електронних ігор можна вважати лінійку Nintendo Game & Watch (приклад гри показано на рисунку 1.1).

Рисунок 1.1 – Портативна електронна гра Nintendo Game & Watch Ball

Наступним етапом еволюції ігрових пристроїв можна вважати портативні ігрові консолі. На відміну від попередників, ці пристрої були

більш продуктивними, мали матричні LCD-дисплеї та дозволяли відтворювати довільний ігровий контент, що завантажувався із зовнішніх носіїв, таких як картриджі. Найвідомішим прикладом є випущений у 1989 році Nintendo Game Boy (рисунок 1.2). В основі даного пристрою лежить спеціальна SoC, що має у своєму складі 8-бітний процесор Sharp SM83 із тактовою частотою 4.19 МГц та пристрій обробки зображень (PPU), обладнаний 8 кБ відеопам’яті [2].

Рисунок 1.2 – Портативна ігрова консоль Nintendo Game Boy

Оскільки портативні пристрої, такі як мобільні телефони, стають більш потужними та дозволяють використовувати довільне програмне забезпечення, подібні ігрові пристрої відходять у минуле. Сучасні портативні ігрові консолі, такі як Nintendo Switch або Steam Deck, можна вважати портативними ПК, оскільки вони обладнуються потужними процесорами із архітектурою ARM або x86, дисплеями високої роздільної здатності, а також підтримкою безпровідних мереж, таких як Bluetooth та Wi-Fi. Окрім власне ігрових застосунків, ці пристрої надають можливість відтворення відео та перегляду веб-сайтів в мережі Інтернет.

Розглядаючи електронні ігрові пристрої в контексті сучасності, слід виділити категорію портативних ігрових консолей, створених на основі доступних мікроконтролерів. Вони є доступними в ціновому плані та простими з точки зору апаратури. Метою подібних проєктів є надання платформи, яка може бути застосована як для розваг, так і для навчання

програмуванню та електроніці. Фокус робиться на доступності продукту та легкості відтворення, що може бути корисно у навчальних цілях. Це досягається за рахунок використання вільного вихідного коду та відкритого ПЗ для розробки. Розглянемо деякі приклади таких проєктів.

Arduboy – мініатюрна консоль на основі мікроконтролера ATmega32u4 (платформа Arduino), що має OLED-дисплей з роздільною здатністю 128x64 точок [3]. Цей пристрій має обмежену обчислювальну потужність, однак є яскравим прикладом реалізації повноцінної ігрової платформи з простим графічним інтерфейсом. Екземпляр пристрою зображено на рисунку 1.3.

Рисунок 1.3 – Портативна ігрова консоль Arduboy

Pokitto – мікроконтролерна консоль з ARM Cortex-M0+, обладнана кольоровим дисплеєм, має вищу продуктивність, ніж Arduboy, і дає змогу реалізовувати більш довершені ігри (рисунок 1.4). Також пристрій має вільний порт GPIO на корпусі, що дозволяє створювати на його основі власні електронні проєкти [4].

Рисунок 1.4 – Портативна ігрова консоль Pokitto

Gamebuino – ще один проєкт на базі Arduino, що позиціонується як платформа для навчання програмуванню [5]. Консоль забезпечує базову підтримку графіки, звуку та взаємодії з користувачем за допомогою тактових кнопок. На рисунку 1.5 показано зовнішній вигляд Gamebuino.

Рисунок 1.5 – Портативна ігрова консоль Gamebuino

Усі ці пристрої демонструють, що навіть бюджетні мікроконтролери здатні реалізовувати інтерактивні розваги. Проте їх функціональні можливості часто обмежені через низьку тактову частоту, малий об’єм пам’яті та низьку роздільну здатність дисплея.

1.2 Визначення функціональних вимог до пристрою

Для реалізації інтерактивної гри на базі мікроконтролера STM32 необхідно чітко визначити набір функціональних вимог до апаратної та програмної частин пристрою. Ці вимоги є основою для подальшого вибору компонентів, архітектури системи та розробки програмного забезпечення. Оскільки мова йде про динамічну гру, ключовими характеристиками пристрою мають стати чітке, плавне відображення графіки, достатня швидкодія в обробці ігрової логіки, а також точна обробка введення з боку користувача.

Однією з основних функціональних вимог є забезпечення якісного графічного виводу на дисплей. Використання кольорового TFT-дисплея з роздільною здатністю 240 на 320 пікселів надає широкі можливості для

відображення різноманітних графічних елементів гри. Проте, щоб не викликати візуального дискомфорту у гравця, оновлення зображення має відбуватися з якнайбільшою частотою [6]. Це особливо важливо для аркадних ігор, де зміна зображення повинна відповідати динаміці ігрового процесу. Затримка між дією користувача та відгуком на екрані має бути мінімальною, інакше це негативно вплине на ігровий досвід. Отже, мікроконтролер повинен бути здатним з достатньою швидкістю формувати зображення та передавати його на дисплей без затримок.

Ще одним важливим аспектом є обробка введення. В більшості модулів TFT-дисплеїв (із контролером ILI9341 або подібними) використовується окремий контролер сенсорної панелі, наприклад XPT2046, який працює за SPI-протоколом. Завдяки цьому можна реалізувати точне позиціонування дотиків на екрані, але для отримання плавного і точного відгуку необхідно періодично опитувати координати сенсора з високою частотою. У грі, де керування реалізується через сенсорні жести або дотики, це особливо важливо. Інтервал між оновленнями координат дотику не повинен перевищувати 20–30 мс, щоб гравець не відчував "запізнення" керування.

Мікроконтролер повинен мати достатній обсяг оперативної пам’яті для збереження графічного буфера, а також для обробки логіки гри. Навіть якщо використовується часткове оновлення зображення, що є бажаним у обмежених за обчислювальною потужністю системах, потрібно мати буфер для хоча б частини кадру. У випадку використання подвійної буферизації, обсяг необхідної пам’яті зростає, але це дає змогу уникнути мерехтіння та забезпечити більш плавну зміну кадрів. З огляду на це, обираючи мікроконтролер, доцільно віддати перевагу моделям з не менше ніж 64 кБ оперативної пам’яті, а також з апаратною підтримкою ПДП, що дозволяє прискорити передачу графічних даних на дисплей.

Щодо обчислювальної потужності, мікроконтролер має забезпечувати виконання циклу оновлення гри за час, що не перевищує 40-50 мс, що відповідає частоті кадрів 20-25 FPS. Це включає оновлення позицій об’єктів,

обробку колізій, логіки гри, а також передачу зображення на екран. Таким чином, слід обрати мікроконтролер із тактовою частотою не менше 72 МГц, бажано з апаратними модулями SPI високої швидкості, підтримкою апаратного множення та обчислень з рухомою крапкою, наявністю таймерів та інших необхідних периферійних пристроїв.

Крім основних вимог до швидкодії та графіки, слід враховувати також енергоспоживання. У випадку, якщо передбачається автономність роботи пристрою, необхідна підтримка живлення від акумулятора, та, можливо, його зарядки, а також стабільна схема живлення для забезпечення належної роботи дисплея та сенсорної панелі.

Загалом, основні функціональні вимоги до пристрою можна сформулювати так: підтримка стабільної частоти оновлення дисплея не нижче 20 кадрів на секунду, точна обробка введення з мінімальними затримками, наявність достатнього обсягу пам’яті для графіки та логіки гри, а також відповідність усіх компонентів вимогам до сумісності, енергоспоживання та швидкодії.

Більш детально використання мікроконтролера STM32 для створення електронної гри з графічним дисплеєм, пов’язані вимоги, обмеження та рішення для їх подолання розглядаються у публікації [7].

1.3 Інтерфейс користувача та взаємодії з пристроєм

Інтерфейс користувача є важливою складовою будь-якого інтерактивного пристрою, особливо коли йдеться про гру в жанрі аркада, що вимагає оперативної взаємодії з користувачем. Основною метою є створення інтуїтивно зрозумілого, візуально привабливого та зручного у використанні інтерфейсу користувача, який би забезпечував комфортну взаємодію із пристроєм та лаконічно інформував гравця про події в ігровому процесі.

Оскільки пристрій має бути оснащений сенсорним TFT-дисплеєм, основний спосіб взаємодії – це дотики до екрана. Використання дотику

дозволяє реалізувати гнучкий інтерфейс користувача без потреби у додаткових апаратних кнопках. У межах гри сенсор може використовуватись як для вибору пунктів меню, так і для прямого керування персонажем або діями в реальному часі. Наприклад, дотик до певної області екрана може відповідати за переміщення, стрільбу або активацію певної функції. Таким чином, точне визначення координат дотику та швидка обробка введення є критично важливими.

Інтерфейс гри має бути адаптованим до фізичних обмежень дисплея – зокрема, до його розмірів та роздільної здатності. При роздільній здатності 240 на 320 пікселів варто уникати дрібних елементів керування, оскільки це ускладнює точне натискання. Кнопки, області взаємодії та графічні об’єкти повинні бути достатньо великими для зручності їх натискання Також важливо забезпечити візуальний зворотний зв’язок: наприклад, зміна кольору елемента або його зовнішнього вигляду при взаємодії. Це підвищує відчуття керованості та покращує користувацький досвід [8].

Крім того, користувацький інтерфейс має оперативно інформувати гравця про різні події, а також забезпечувати відображення очок, здоров’я та іншої статусної інформації. Це може досягатися за рахунок використання тексту та піктограм в окремій області дисплея (статусний рядок), або, наприклад, поруч із ігровим персонажем.

Окрім використання графічної інформації, сигналізувати про окремі події можна за допомогою звуку. Короткі звукові сигнали різної частоти можуть бути використані, наприклад, аби повідомити користувача, про втрату ігровим персонажем здоров’я або набір додаткових очок. Генерація примітивних звукових сигналів з використанням мікроконтролера є простим завданням, що не вимагає значних апаратних витрат, проте дозволяє покращити досвід користувача від взаємодії з пристроєм. При використанні мікроконтролера із вбудованим ЦАП можливо реалізувати відтворення більш довершених звукових ефектів та навіть музики [9].

Для введення гравцем окремих команд, окрім використання сенсорного

дисплея, можливо застосовувати фізичні кнопки та перемикачі. Зокрема, подібні елементи можуть бути використані для встановлення гри на паузу та її продовження, керування живленням, відтворенням звуку тощо Використання фізичних кнопок дозволяє розвантажити дисплей від великої кількості елементів керування, а допомагає забезпечити тактильний зворотній зв’язок для окремих взаємодій.

Із програмного боку, інтерфейс повинен бути реалізований із урахуванням обмежень продуктивності системи. Наприклад, надмірне використання складних графічних операцій (анімацій, прозорості, згладжування) може суттєво вплинути на швидкодію пристрою.

Таким чином, інтерфейс користувача в проєкті має бути логічно структурованим, адаптованим до сенсорного керування, а також максимально легким для графічної обробки, щоб не знижувати загальну продуктивність і не створювати перешкод для плавного відтворення гри. Використання різних засобів інформування та керування (графіка, звук, дотик) дозволяють покращити загальне враження від взаємодії із пристроєм. Незважаючи на загальну простоту системи, ці елементи створюють завершену структуру гри, яка сприймається як повноцінний програмно-апаратний продукт.

2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ

2.1 Вибір та обґрунтування мікроконтролера

У межах цієї роботи було обрано мікроконтролер STM32F303RE, що встановлений на відлагоджувальній платі NUCLEO-F303RE (рисунок 2.1). Цей мікроконтролер належить до лінійки STM32F3, яка поєднує високу швидкодію ядра ARM Cortex-M4 із широкою підтримкою периферії та доступною вартістю.

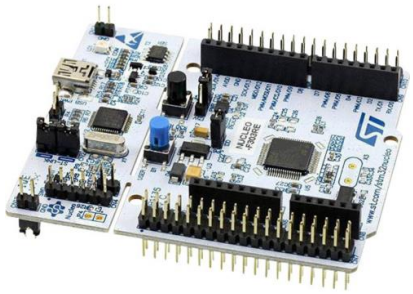

Рисунок 2.1 – Плата розробки STM32 NUCLEO-F303RE

Однією з причин вибору STM32F303RE є його тактова частота – до 72 МГц, що є оптимальним значенням для реалізації ігрової логіки, яка включає постійне оновлення графіки, обробку подій і передачу даних до дисплея. Крім того, ядро Cortex-M4 підтримує апаратне виконання операцій з плаваючою комою (FPU), що вигідно вирізняє його на фоні рішень без підтримки даної технології.

Обсяг оперативної пам’яті SRAM мікроконтролера, що становить 64 кБ, дозволяє зберігати графічні елементи, звукові ефекти, змінні ігрової логіки та інші дані, не створюючи дефіциту ресурсів. Також контролер має

512 кБ флеш-пам’яті, чого цілком достатньо для зберігання даних гри навіть із кількома рівнями, графікою у вигляді спрайтів, а також дозволяє не обмежуватися при виборі бібліотек та драйверів.

Окрему увагу слід приділити периферійним можливостям. STM32F303RE має декілька апаратних SPI-інтерфейсів, що дозволяє підключати TFT-дисплей із контролером ILI9341 та сенсорний контролер XPT2046 без втрати швидкодії. Завдяки наявності ПДП-контролера можливо реалізувати передачу даних до дисплея та інших периферійних пристроїв у фоновому режимі, без завантаження основного ядра, що особливо корисно при досягненні високої плавності відтворення зображення [10].

Важливою перевагою обраного мікроконтролера лінійки STM32F3 є наявність великої кількості таймерів. У порівнянні з більш бюджетними лінійками STM32, дане сімейство має, окрім базових таймерів та таймерів загального призначення, також і таймери з розширеним управлінням, а також високоточний таймер [9]. Дані таймери надають широкі можливості для налаштування та отримання часових показників, що може бути використано не тільки для тактування обробки ігрових кадрів, а і, наприклад, для генерації ШІМ.

Також до переваг мікроконтролера STM32F303RE можна віднести наявність продуктивного ЦАП, що є незамінним для генерації звукових сигналів. ЦАП в STM32 тактується за допомогою одного з таймерів та підтримує роботу з використанням механізму ПДП, що дозволяє організувати відтворення звуку, не перериваючи при цьому основне ядро ARM. Даний сценарій використання є оптимальним для відтворення звукових ефектів та музики в ігровому застосунку.

Обрана для роботи з мікроконтролером плата розробки NUCLEO-F303RE є зручною, оскільки містить вбудований відлагоджувач ST-LINK/V2-1, сумісна із популярними середовищами розробки (STM32CubeIDE, Keil, PlatformIO), має простий доступ до всіх виводів мікроконтролера через роз’єми Arduino і Morpho, а також підтримує живлення як через USB, так і

ззовні. Мікроконтролер на платі виконано у форм-факторі LQFP64, що дозволяє не обмежуватися задіяною кількістю виводів GPIO при підключенні периферійних пристроїв. Перелічені особливості плати роблять її зручною для швидкого прототипування.

2.2 Вибір дисплея та допоміжних модулів

Для відображення графіки було обрано модуль, зображений на рисунку 2.2. Цей модуль засновано на основі контролера дисплея ILI9431, що також оснащений резистивним сенсорним екраном, реалізованим із застосуванням мікросхеми XPT2046. Дисплей має TFT-матрицю з роздільною здатністю 320 на 240 пікселів та підтримує 16-бітну кольорову палітру. Це дозволяє відображати графіку відносно високої якості, що позитивно впливає на сприйняття гри користувачем.

Рисунок 2.2 – Модуль TFT дисплея з контролером ILI9341

Для взаємодії із мікроконтролером в обраному модулі застосовуються інтерфейс SPI. Модулі з використанням цього інтерфейсу є доволі розповсюдженими та мають порівняно низьку ціну. З одного боку, використання SPI дозволяє значно спростити підключення дисплея, оскільки не потребує великої кількості виводів. З іншого боку, оскільки для передачі даних використовується послідовний інтерфейс, даний аспект може негативно впливати на продуктивність виведення у порівнянні з паралельною

шиною. Швидкість оновлення дисплея в такому випадку обмежуватиметься тактовою частотою ядра використовуваного мікроконтролера. Для пришвидшення обміну даними із дисплеєм можна задіяти наявний у STM32 механізм ПДП, що дозволяє здійснювати взаємодію між пам’яттю та периферійним пристроєм в обхід ядра мікропроцесора.

З-поміж особливостей даного модуля можна навести напругу логічних рівнів у 3.3 В, що дозволяє підключати дисплей до STM32 напряму без використання додаткових перетворювачів рівнів. Дисплей має світлодіодне підсвічування, яким можна керувати за допомогою окремого виводу. Також на зворотній стороні модуля знаходиться роз’єм для карти пам’яті формату microSD, що, подібно до дисплея, використовує SPI для взаємодії із мікроконтролером та може бути використаний для розширення функціоналу пристрою (наприклад, для збереження прогресу гравця чи набраних очок).

Для відтворення звуку обрано мініатюрний модуль динаміка, зображений на рисунку 2.3.

Рисунок 2.3 – Обраний модуль динаміка

Подібний модуль не забезпечує якісного відтворення звуку на усьому діапазоні частот, проте добре підходить для подання простих звукових сигналів із застосуванням прямокутних імпульсів. До плюсів даного модуля можна віднести низький опір динаміка (15 Ом), що дозволяє обійтися без підсилення сигналу.

2.3 Живлення та безпека пристрою

Підсистема живлення обраної плати розробки, подібно до усіх плат серії NUCLEO-64, дозволяє здійснювати живлення від джерел із різною напругою, що забезпечується наявними регуляторами напруги. При розробці та тестуванні у якості джерела живлення може виступати USB-порт ПК, до якого підключається вбудований у плату програматор ST-LINK/V2-1. Якщо необхідно використати зовнішнє джерело живлення, можливі наступні варіанти:

- пін «VIN» дозволяє живити плату розробки від джерела постійної напруги 7-12 В із максимальним струмом 800 мА;
- пін «E5V» передбачає використання зовнішнього джерела із напругою від 4.75 В до 5.25 В із максимальним струмом 500 мА;
- пін «+3.3V» може бути використаний для живлення від джерела напругою від 3 В до 3.6 В.

Для того щоб забезпечити живлення плати NUCLEO від одного із зовнішніх джерел, достатньо відповідним чином встановити положення перемички JP5 на платі [11]. Таким чином, можливість вибору джерел живлення у широкому діапазоні напруг дозволяє живити пристрій як від джерела USB, так і, наприклад, від батареї або спеціального мережевого адаптера.

2.4 Структурна схема пристрою

Було розроблено структурну схему пристрою із застосуванням САПР Proteus Design Suite 8. На схемі відображені компоненти пристрою та з’єднання між ними (рисунок 2.4).

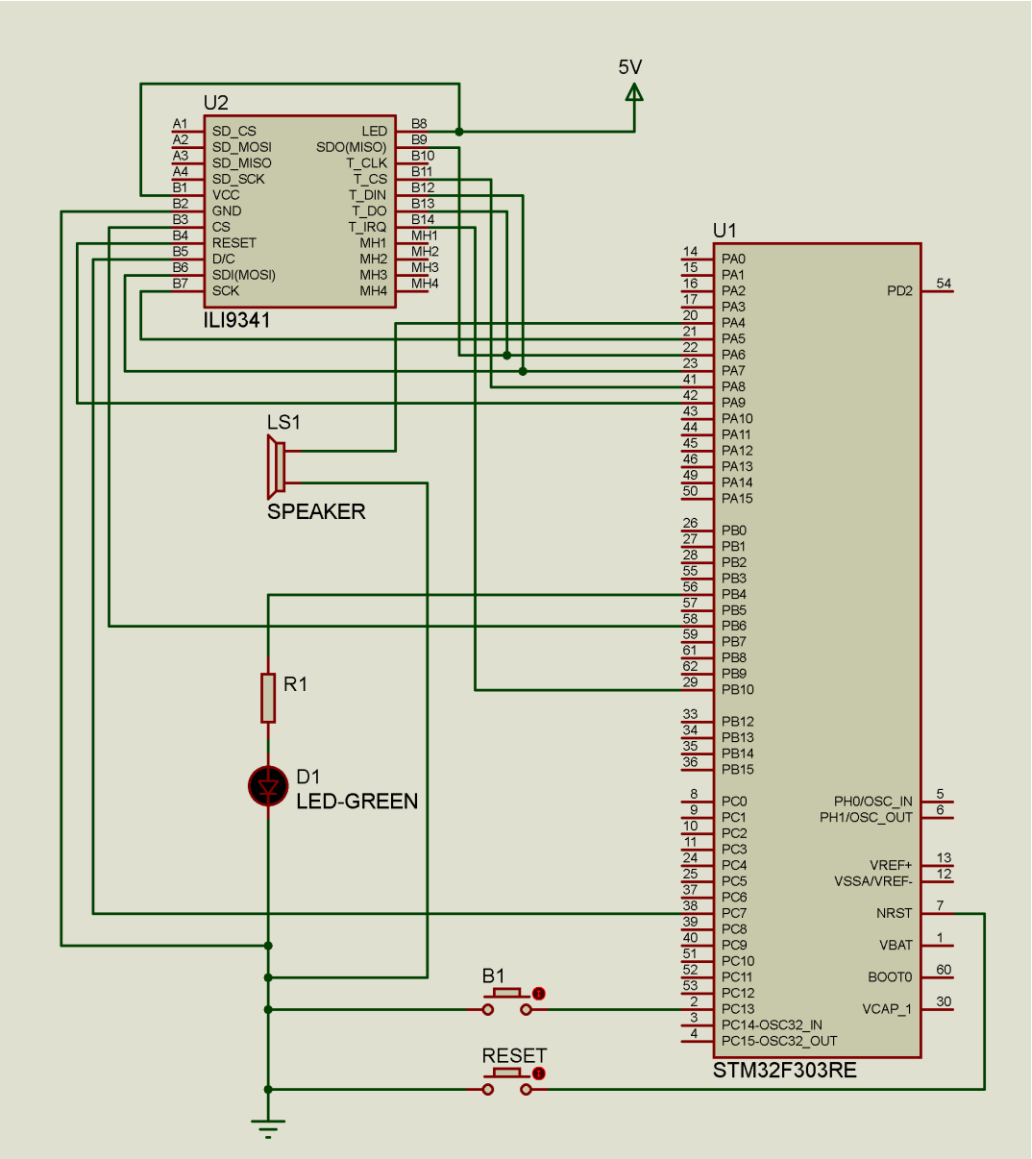

Рисунок 2.4 – Схема пристрою

Пристрій складається з плати розробки STM32 NUCLEO, модуля дисплея на основі контролера ILI9341, модуля динаміка, а також світлодіода для відображення налагоджувальної інформації.. Для з’єднання контролерів ILI9341 та XPT2046 із керуючим STM32 використовується спільна шина SPI із окремими лініями CS для перемикання між ними. Пристрій також має кнопки B1 та RESET, що розташовані на платі NUCLEO та використовуються для встановлення гри на паузу та скидання стану мікроконтролера відповідно.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Огляд мови програмування та засобів розробки

Для написання програмного забезпечення електронної гри було обрано мову програмування С. Дана компільована мова є однією із найпоширеніших у галузі розробки вбудованого програмного забезпечення завдяки своїй високій ефективності та гнучкості. Мова надає можливості для тонкого контролю над пам’яттю та організацією структур даних. Це дозволяє оптимізувати роботу програми в умовах обмеженості апаратних ресурсів, а відсутність складних мовних конструкцій мінімізує накладні витрати при компіляції, що забезпечує максимальну швидкодію.

В якості середовища розробки було обрано інтегроване середовище розробки STM32CubeIDE. Це офіційне середовище від компанії STMicroelectronics, засноване на фреймворку Eclipse®/CDT та інструментарії GCC (GNU Compiler Collection), що надає засоби для компіляції та побудови програм мовами C та C++ [12]. Також середовище надає інструменти для відлагодження, засновані на GDB (GNU Debugger).

Середовище STM32CubeIDE дозволяє проводити відлагодження безпосередньо на цільовому мікроконтролері під час його роботи. Надаваний відлагоджувач GDB підтримує методологію віддаленого налагодження, що дозволяє керувати роботою процесора ARM через послідовний порт, який використовується для завантаження програми. Середовище надає графічний інтерфейс до налагоджувача, що дозволяє встановлювати точки зупинки під час роботи програми та відслідковувати її виконання покроково, а також переглядати вміст змінних та регістрів процесора. Використання цих засобів підвищує наочність при тестуванні програми та значно пришвидшує процес знаходження помилок.

Для конфігурування апаратної частини мікроконтролера

використовується інструмент STM32CubeMX – графічна утиліта, що поставляється у складі STM32CubeIDE. Вона дозволяє зручно налаштовувати периферійні пристрої (SPI, DMA, таймери тощо), джерела тактування, а також призначати піни введення-виведення через графічний інтерфейс (рисунок 3.1). STM32CubeMX автоматично генерує початкову структуру проєкту мовою C на основі створеної конфігурації. Отримана структура включає код ініціалізації для обраного набору периферійних пристроїв та заготовки важливих функцій, таких як обробники переривань та помилок.

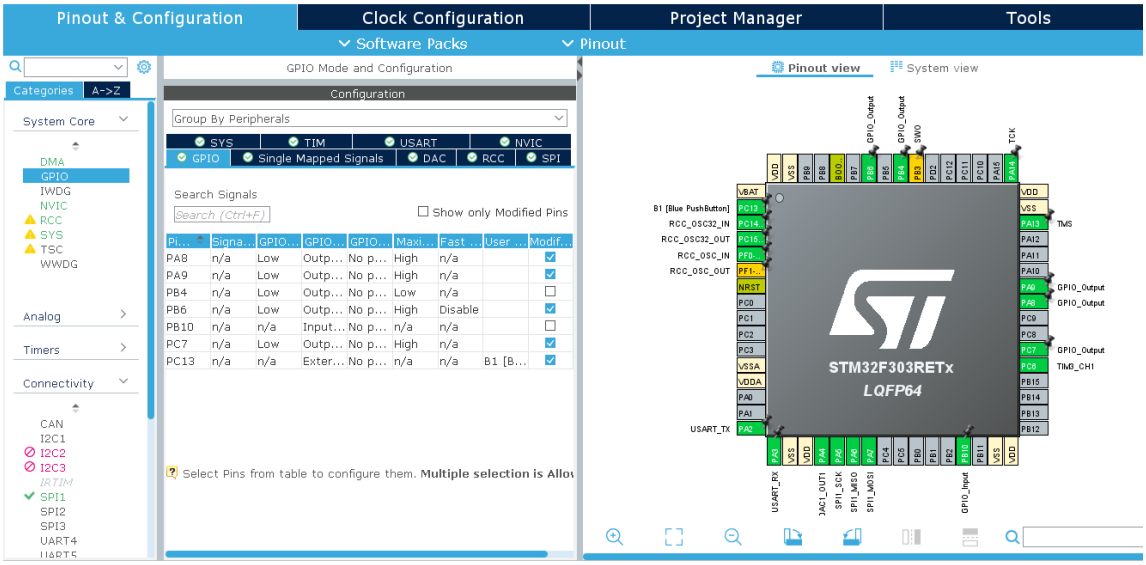

Рисунок 3.1 – Налаштування проєкту у STM32CubeMX

3.2 Алгоритм роботи програми

В основі алгоритму гри лежить ітерація ігрового циклу, що відповідає за зчитування даних введення, оновлення положення об’єктів та значень змінних, а також візуалізацію кадру гри. Щоб забезпечити постійну частоту кадрів, ця функція викликається з обробника переривання одного з таймерів.

При розробці динамічних ігрових застосунків прийнято використовувати шаблон «ігровий цикл». Його застосування має на увазі постійне відтворення ігрової логіки у безкінечному циклі. Для отримання

стабільної швидкості відтворення гри в такому шаблоні зазвичай використовується деякий спосіб синхронізації періоду обробки кадру, наприклад, введення часу простою, що обчислюється як різниця між часом, затраченим на обробку кадру та цільовим періодом [13]. Оскільки платформа STM32 надає декілька гнучких апаратних таймерів, для забезпечення постійної частоти кадрів оновлення стану гри викликається з обробника переривання одного з них, без використання циклу в основній частині програми.

Розглянемо алгоритм роботи пристрою. Після увімкнення живлення першою йде стадія ініціалізації периферійних пристроїв мікроконтролера. Потім відбувається генерація звукових сигналів та їхнє збереження у пам’ять, після чого на дисплей виводиться початкова заставка та пристрій очікує натискання гравцем сенсорної кнопки початку гри.

Після натискання кнопки відбувається ініціалізація гри: встановлюються початкові значення змінних та запускається таймер відліку кадрів гри. Після цього керування переходить до обробника переривання таймера, що відповідає за відтворення ігрової логіки. По завершенні гри таймер зупиняється та на екран виводяться набрані гравцем очки. Після цього пристрій очікує натискання кнопки «Зіграти знову», що заново ініціалізує гру та перезапускає таймер.

Обробник ігрової логіки, що викликається для кожного нового кадру гри, відповідає за наступні функції:

- обробка сенсорного введення;
- оновлення положень об’єктів;
- створення об’єктів;
- обробка зіткнень;
- запуск відтворення звукових ефектів;
- вибіркове очищення дисплея;
- виведення на дисплей.

Аби дозволити належну швидкість відтворення гри, у програмі

застосовується метод вибіркового оновлення зображення, що передбачає перемальовування лише тих частин кадру, які змінилися у порівнянні з попереднім кадром. Це дозволяє уникнути пересилання усього кадру на дисплей, що значно пришвидшує час оновлення зображення. Алгоритм більш детально розглянуто у пункті 3.4.

3.3 Вибір бібліотек, драйверів та стеків протоколів

Для програмної взаємодії із ресурсами мікроконтролера було використано STM32 HAL — рівень абстракції апаратного забезпечення (Hardware Abstraction Layer). Це набір бібліотек та драйверів, розроблених STMicroelectronics, які забезпечують уніфікований і зручний інтерфейс до периферійних пристроїв без необхідності прямої роботи з регістрами. Працювати з різними елементами мікроконтролера можна через простий у використанні та інтуїтивно зрозумілий API. Функції HAL працюють на усіх мікроконтролерах STM32 за рахунок використання різних реалізацій драйверів для різних моделей серії, що дозволяє створювати переносимий код. Таким чином, використання HAL значно спрощує розробку та підвищує читабельність коду.

Для керування дисплеєм обрано бібліотеку ILI9341-STM32-HAL [14]. Вона надає можливості для роботи з дисплеями, що засновані на контролері ILI9341 через протокол SPI, а також із контролерами сенсорного дисплея XPT2046. Дана бібліотека використовує механізм ПДП для передачі даних, що дозволяє значно підвищити швидкість виведення графіки. До переваг даної бібліотеки можна віднести наступні особливості:

- підтримка STM HAL;
- висока швидкість оновлення зображення у порівнянні з аналогами;
- зручна модульна конфігурація інтерфейсів дисплея, що, з-поміж іншого, дозволяє працювати із декількома модулями дисплея одночасно;
- два методи для калібрації сенсорного екрана;

- можливість встановити функції зворотного виклику для початку та кінця натискання на сенсорний екран;

- функції для малювання точок, прямокутників та кіл, монохромних растрових зображень, а також текстових рядків з використанням декількох вбудованих шрифтів;

- вільна ліцензія MIT, що дозволяє копіювання та модифікацію вихідного коду.

3.4 Реалізація ключових функцій

Програмну частину системи було реалізовано в проєкті середовища STM32CubeIDE, доступному на сервісі GitHub за посиланням [15]. Проєкт включає файли вихідного коду мовою C та файл із налаштуваннями периферійних пристроїв мікроконтролера із розширенням «.ioc». Нижче більш детально розглядається код, що відповідає за основну функціональність пристрою

Ключовим для роботи системи є функціонування таймера TIM6. Даний таймер викликає переривання, що відповідає за підготовку ігрових кадрів. Для розрахунку частоти, з якою таймер викликатиме оновлення, використовуються значення двох регістрів: PSC (регістр попереднього дільника) та ARR (регістр перезавантаження). Розрахувати частоту роботи таймера можна за формулою 3.1 [10].

$$f_{таймера} = \frac{f_{тактова}}{(PSC + 1) \times (ARR + 1)} \tag{3.1}$$

Значення регістрів PSC (35999) та ARR (79) підібрано таким чином, щоб досягти періоду спрацювання таймера у 25 Гц (3.2). Це відповідає необхідній частоті оновлення кадрів гри:

$$f_{таймера} = \frac{7.2 \times 10^7 \ Гц}{(35999 + 1) \times (79 + 1)} = 25 \ Гц \quad (3.2)$$

Обробник переривання від таймера TIM6 (лістинг 3.1) викликає функцію UpdateGame, що містить код, відповідальний за підготовку та візуалізацію нового кадру гри. Також в цьому обробнику міститься спеціальний код для тестування продуктивності системи. Він перевіряє, чи не встановлений прапор TIM_FLAG_UPDATE після завершення обробки кадру. Якщо прапор встановлений, це означає, що таймер викликав нову подію до того, як обробник попереднього переривання встигнув завершити свою роботу. В такому випадку у послідовний порт виводиться текстове попередження, оскільки подібна ситуація свідчить про недостатню швидкодію системи.

Лістинг 3.1 – Функція зворотного виклику обробки переривання таймера

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if(htim->Instance == TIM6)
  {

    UpdateGame();

    if (__HAL_TIM_GET_FLAG(htim, TIM_FLAG_UPDATE))
    {
      UART_Printf("Timer overrun - UpdateGame() too slow!\r\n");
    }
  }
```

Розглянемо обробник переривання від зовнішнього джерела (лістинг 3.2). Якщо джерелом переривання є кнопка користувача (пін PC13), таймер TIM6 перемикає свій стан на вимкнений або навпаки, а також програється звуковий сигнал. Таким чином реалізовано встановлення гри на паузу.

Лістинг 3.2 – Функція обробки переривання від зовнішнього джерела

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if (GPIO_Pin == GPIO_PIN_13)
  {
    if (gameOver)
      return;
    pause = !pause;
    if (pause)
    {
      HAL_TIM_Base_Stop_IT(&htim6);
      PlaySound(soundPause);
    }
    else
      HAL_TIM_Base_Start_IT(&htim6);

  }
  else
  {
    __NOP();
  }
}
```

Елементами гри, що підлягають обробці, є динамічні об’єкти, які характеризуються типом, положенням у просторі, швидкістю та станом. У лістингу 3.3 наведено структуру Object, що слугує для представлення даних про ігрові об’єкти (ворогів, гравця, частинки тощо).

Лістинг 3.3 – Структура Object

```
struct Object {
  int16_t x;
  int16_t y;
  int16_t w;
  int16_t h;
  int16_t old_x;
  int16_t old_y;
  int8_t vx;
  int8_t vy;
  int8_t alive;
  int8_t state;
  enum ObjectType type;
};
```

Поля даної структури мають наступне призначення:

- x, y, w та h – позиція об’єкта по осях x та y, його ширина та висота відповідно;
- old_x, old_y – позиція об’єкта у минулому кадрі, що використовується при локальному оновленні зображення функцією ClearObject;
- vx та vy – складові вектора швидкості об’єкта;
- alive – чи є об’єкт активним, неактивні об’єкти ніяк не обробляються;
- state – стан об’єкта, що використовується, зокрема, для поведінки ворогів та відображення анімації гравця;
- type – поле, що визначає тип об’єкта з переліку можливих типів.

Функція UpdateGame оперує над масивом об’єктів гри, оновлюючи їхній стан відповідно до типу кожного об’єкта. У лістингу 3.4 наведено фрагмент функції UpdateGame, що відповідає за керування гравцем.

Лістинг 3.4 – Обробка сенсорного введення та керування гравцем

```
if (playerLife > 0 && ili9341_touch_coordinate(lcd, &touch_x,
&touch_y) == itpPressed)
{
  touch_y = 320 - touch_y;
  touch_x -= 48;
  touch_x -= 12;

  int16_t spd_x = touch_x - player->x;
  int16_t spd_y = touch_y - player->y;

  spd_x /= (1 + abs(spd_x) / 4);
  spd_y /= (1 + abs(spd_y) / 4);

  player->vx = spd_x;
  player->vy = spd_y;

  if (spd_x > 0) player->state = 2;
  else if (spd_x < 0) player->state = 1;
```

Розглянемо наведений вище фрагмент більш детально. Для початку перевіряється наявність натискання та отримуються його координати. Далі здійснюється корекція координат, оскільки за замовчуванням вони є віддзеркаленими по вертикалі відносно зображення та дещо зміщеними у по горизонталі у порівнянні із дійсною точкою натискання. Після цього на основі отриманих координат розраховується нова швидкість гравця, що є пропорційною відстані від точки дотику до ігрового персонажа. В залежності від того, в який бік рухається гравець, встановлюється значення його поля state, що відповідає одній з анімацій нахилу корабля убік.

Лістинг 3.5 – Обробка сенсорного введення та здійснення пострілу

```
  if (frameCounter % 13 == 0 && !playerInvul)
  {
    CreateObject(player->x + (player->w / 2) - 2, player->y - 8,
4, 8, 0, -8, typeBullet);

    PlaySound(soundShot);
  }

}
else
{
  player->state = 0;
  player->vx = 0;
  player->vy = 0;
}
```

При натисканні кожні 13 кадрів здійснюється постріл (лістинг 3.5). Для цього створюється новий об’єкт снаряду та програється відповідний звуковий ефект. У випадку, якщо натискання не зареєстроване, швидкість гравця встановлюється в нуль та його анімація скидається.

Для усіх інших об’єктів оновлення відбувається в циклі, що перебирає усі активні об’єкти та викликає різні функції оновлення в залежності від їхніх типів. Як приклад розглянемо функцію UpdateEnemy (лістинг 3.6), що відповідає базовому типу ворогів:

Лістинг 3.6 – Функція оновлення об’єкта-ворога

```
void UpdateEnemy(struct Object* obj)
{
  obj->vx = enemySpeeds[((frameCounter) / 10) % 8];
}
```

Як можна побачити, дана функція циклічно встановлює горизонтальну швидкість ворогів з масиву в залежності від номера поточного кадру. Значення швидкості в масиві (лістинг 3.7) підібрані таким чином, щоб забезпечити періодичний рух ворогів з боку в бік:

Лістинг 3.7 – Масив швидкостей об’єкта-ворога

```
const int8_t enemySpeeds[8] = {
  -2, -1, 0, 0, 1, 2, 0, 0
};
```

Для інших типів ворогів оновлення реалізовано аналогічним чином, з використанням більш довершеної логіки для досягнення складнішої поведінки (рух по колу та прицільна стрільба).

Після оновлення усіх динамічних об’єктів їхнє положення оновлюється з урахуванням встановленої раніше швидкості (лістинг 3.8). Також запам’ятовується старе положення об’єктів, що є важливим для роботи алгоритму оновлення зображення.

Лістинг 3.8 – Збереження старого положення об’єкта

```
obj->old_x = obj->x;
obj->old_y = obj->y;

obj->y += obj->vy;
obj->x += obj->vx;
```

Для досягнення оптимальної швидкодії системи важливу роль має запроваджений алгоритм вибіркового оновлення зображення Розглянемо роботу функції ClearObject, що реалізує зональне очищення фону (лістинг 3.9). Дана функція визначає, чи зсунувся об’єкт по кожній з осей та на основі цієї інформації обчислює координати та розміри regX, regY, regW та regH областей, що мають бути оновлені. Області додатково коригуються таким чином, щоб не накладатися на елементи інтерфейсу користувача у верхній частині екрану. Для вертикальної області обчислення виконується аналогічно, тому відповідний фрагмент коду опущено.

Лістинг 3.9 – Фрагмент функції ClearObject, формування горизонтальної області оновлення

```
void ClearObject(struct Object *obj, ili9341_color_t color)
{
  int16_t regX, regW, regY, regH;
  int8_t skipX = 0;

  if (obj->x < obj->old_x)
  {
      regX = obj->x + obj->w;
      regW = obj->old_x - obj->x;
  }
  else if (obj->x > obj->old_x)
  {
      regX = obj->old_x;
      regW = obj->x - obj->old_x;
  }
  else
      skipX = 1;

  if (regW > obj->w)
  {
      regX = obj->old_x;
      regW = obj->w;
  }
```

Після цього обчислені значення використовуються для заповнення фону потрібним кольором У випадку, якщо у порівнянні з минулим кадром об’єкт не змістився, функція не робить нічого. Можливі результати роботи

функції ClearObject показані на рисунку 3.2, горизонтальна область оновлення виділена синім кольором, вертикальна – блакитним

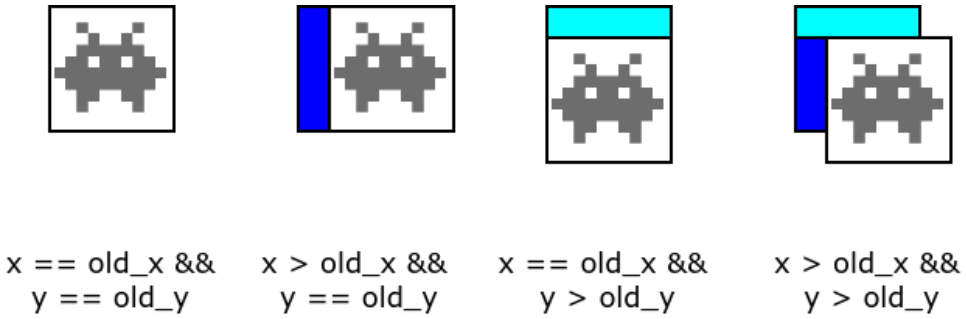

Рисунок 3.2 – Можливі результати роботи функції ClearObject

Результатом роботи функції є заповнення областей кольором фону (лістинг 3.10).

Лістинг 3.10 – Фрагмент функції ClearObject, виведення областей

```
  if ((!skipX))
    ili9341_fill_rect(lcd, color, regX, (obj->y < 20 ? 20 : obj-
>y), regW, obj->h);
  if ((!skipY))
    ili9341_fill_rect(lcd, color, skipX ? obj->x : obj->old_x,
(regY < 20 ? 20 : regY), obj->w, regH);
}
```

Обрана бібліотека для виведення на дисплей надає функцію ili9341_draw_bitmap_1b для малювання монохромних зображень в упакованому форматі 1 біт на піксель. На жаль, ця функція містить ряд помилок, що робить виведення некоректним при використанні деяких положень та розмірів зображень [16].

Для відтворення даного функціоналу було реалізовано власну функцію ili9341_my_draw_bmp_2b, що виводить на дисплей зображення у форматі 2 біти на піксель (лістинг 3.11). Це дозволяє використовувати до чотирьох

кольорів при виведенні одного зображення (спрайта), що є достатнім для відображення ігрових об’єктів.

Лістинг 3.11 – Функція ili9341_my_draw_bmp_2b, початок

```
void ili9341_my_draw_bmp_2b(ili9341_t *lcd, ili9341_color_t
color, ili9341_color_t color2, ili9341_color_t color3,
ili9341_color_t color4, int16_t x, int16_t y, uint16_t w,
uint16_t h, const uint8_t *bmp, uint8_t score_area)
{

  int16_t  _x = x, _y = y;
  uint16_t _w = w, _h = h;

  if (score_area)
  {
    if (ibNOT(ili9341_clip_rect(lcd, &_x, &_y, &_w, &_h)))
      { return; }
  }
  else
  {
    if (ibNOT(ili9341_my_clip_rect(lcd, &_x, &_y, &_w, &_h)))
      { return; }
  }

  uint32_t num_pixels = _w * _h;
  uint32_t rect_wc    = num_pixels;

  uint32_t block_wc = rect_wc;
  if (block_wc > __SPI_TX_BLOCK_MAX__)
    { block_wc = __SPI_TX_BLOCK_MAX__; }

  uint16_t color_le = __LEu16(&color);
  uint16_t color2_le = __LEu16(&color2);
  uint16_t color3_le = __LEu16(&color3);
  uint16_t color4_le = __LEu16(&color4);

  ili9341_spi_tft_set_address_rect(lcd, _x, _y, (_x + _w - 1),
(_y + _h - 1));
  ili9341_spi_tft_select(lcd);

  HAL_GPIO_WritePin(lcd->data_command_port, lcd-
>data_command_pin, __GPIO_PIN_SET__);
```

Функція дозволяє здійснювати малювання в ігровій області або в області відображення очок (використовується для малювання сердечок). Спочатку здійснюється корекція розмірів зображення таким чином, щоб воно

поміщалося у межі екрана. Надалі робота ведеться із відредагованими значеннями. Потім обчислюється кількість точок зображення для подальшої обробки та здійснюється переведення кольорів у підтримуваний дисплеєм вигляд.

Код у лістингу 3.12 декодує упаковане зображення та переводить його у формат, що підходить для передачі на дисплей через SPI:

Лістинг 3.12 – Функція ili9341_my_draw_bmp_2b, продовження

```
uint16_t byte = 0;
uint16_t b = 0;
b = ((_y - y) * _w + (_x - x)) / 4;
while (rect_wc > 0) {
for (uint16_t i = 0; i < block_wc; ++i)
{
   if (i % _w == 0)
   {
       b = b + (w - _w) / 4;
   }
  if (i & 3)
  {
  byte <<= 2;
  }
  else
  {
  byte = bmp[b];
  b++;
  }
  uint8_t bits = byte & 0xC0;
  switch (bits)
  {
  case 0x00:
  spi_tx_block[i] = color2_le;
  break;
  case 0xC0:
  spi_tx_block[i] = color_le;
  break;
  case 0x80:
  spi_tx_block[i] = color3_le;
  break;
  case 0x40:
  spi_tx_block[i] = color4_le;
  break;
  }
}
```

Лістинг 3.13 – Функція ili9341_my_draw_bmp_2b, кінець

```
  uint32_t curr_wc;

    curr_wc = rect_wc;
    if (curr_wc > block_wc)
      { curr_wc = block_wc; }
    ili9341_transmit_color(lcd, curr_wc * 2/*16-bit words*/,
spi_tx_block, ibYes);
    rect_wc -= curr_wc;
  }

  ili9341_spi_tft_release(lcd);
}
```

Нарешті, код, наведений у лістингу 3.13, заповнює відповідними кольорами буфер для передачі та виводить зображення на дисплей.

Ще однією важливою функцією пристрою є відтворення звукових сигналів. Для виконання цієї задачі у розроблюваній системі використовується вбудований ЦАП DAC1. Для керування перетворенням використовується таймер TIM7, налаштований на створення події із частотою 8 кГц.

Оскільки використання ПДП передбачає пересилання даних з пам’яті, звукові сигнали для відтворення генеруються під час ініціалізації системи та зберігаються в буферах для подальшого відтворення. Розглянемо функцію GenerateWaveform (лістинг 3.14).

У даному алгоритмі реалізується генерація прямокутної звукової хвилі у форматі PCM з частотою дискретизації 8 кГц на основі масиву частот, кожне значення якого відповідає інтервалу тривалістю 25 мс (лістинг 3.15).

Для кожного значення масиву tune обчислюється відповідна частота прямокутного сигналу, що генерується шляхом чергування високого (4095) та низького (0) рівнів. Зміна рівня відбувається з періодичністю, що відповідає половині періоду заданої частоти (тобто 8000 / freq / 2). Таким чином, формується послідовність семплів, яка імітує прямокутну хвилю заданої частоти, або тишу, якщо значення частоти дорівнює нулю.

Лістинг 3.14 – Функція GenerateWaveform

```
void GenerateWaveform(const uint16_t *tune, uint16_t *waveform,
unsigned int length)
{

  for (int i = 0; i < length; i++)
  {
      int idx = i / 200;
      int freq = tune[idx];

      if (freq == 0) {
       waveform[i] = 0;
       continue;
      }
      if ((i / (8000 / freq / 2) ) % 2 == 0)
      {
          waveform[i] = 4095;
      }
      else
       waveform[i] = 0;
  }

}
```

Лістинг 3.15 – Приклад масиву частот для генерації звукового сигналу

```
const uint16_t tunePause[6] = {
  800, 600, 0, 0, 800, 600,
};
```

Розглянутий вище алгоритм дозволяє перетворювати компактне представлення мелодії у вигляді частотного ряду на-дані в форматі PCM, придатні для виводу через ЦАП.

Для відтворення звуку у грі наявна функція PlaySound (лістинг 3.16). Вона дозволяє відлік таймера TIM7 та викликає функцію HAL_DAC_Start_DMA, що запускає цифро-аналогове перетворення з використанням ПДП. В якості джерела даних для ПДП вказується буфер, що містить згенерований раніше звуковий сигнал.

Лістинг 3.16 – Функція PlaySound (фрагмент)

```
void PlaySound(enum SoundType sound)
{
  if (!(sound == soundShot || previousSound ==
soundPlayerExplosion))
  {
    HAL_DAC_Stop_DMA(&hdac1, DAC_CHANNEL_1);
    HAL_TIM_Base_Stop(&htim7);
  }
  switch(sound)
  {
  case soundShot:
    HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1,
(uint32_t*)soundWaveformShot, SOUND_SHOT_LENGTH,
DAC_ALIGN_12B_R);
    break;

  case soundPause:
    HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1,
(uint32_t*)soundWaveformPause, SOUND_PAUSE_LENGTH,
DAC_ALIGN_12B_R);
    break;
  }
  HAL_TIM_Base_Start(&htim7);
  previousSound = sound;
}
```

Лістинг 3.17 – Функція зворотного виклику HAL_DAC_ConvCpltCallbackCh1

```
void HAL_DAC_ConvCpltCallbackCh1(DAC_HandleTypeDef* hdac)
{
  HAL_TIM_Base_Stop(&htim7);
}
```

Для того, щоб забезпечити коректну роботу ЦАП, важливо зупинити роботу таймера по завершенні перетворення. Це досягається за рахунок перевизначеної вбудованої функції зворотнього виклику HAL_DAC_ConvCpltCallbackCh1, наведеної у лістингу 3.17.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА РЕЗУЛЬТАТИ

4.1 Перевірка працездатності пристрою та гри

Для того, щоб перевірити роботу пристрою, перевіримо працездатність окремих компонентів системи. Для отримання інформації про роботу системи використовується виведення текстових повідомлень у послідовний порт USART2.

Для перевірки загальної продуктивності системи достатньо впевнитися у тому, що основна функція підготовки та візуалізації кадру UpdateGame має змогу виконуватися із необхідною швидкістю. Для цього у обробнику переривання таймера TIM6 було запроваджено код для перевірки, розглянутий у підрозділі 3.4 (лістинг 3.1).

Важливо також перевірити правильне налаштування та коректну роботу ЦАП при відтворенні аудіо. Для цього в коді програми перевизначені дві функції зворотного виклику HAL_DAC_ErrorCallbackCh1 та HAL_DAC_DMAUnderrunCallbackCh1, що викликаються драйверами HAL при виникненні помилки ЦАП та при недостатній швидкості перетворення, відповідно (лістинг 4.1)

Лістинг 4.1 – Функції обробки помилок ЦАП

```
void HAL_DAC_ErrorCallbackCh1(DAC_HandleTypeDef *hdac)
{
  UART_Printf("DAC error! Error: %d, DAC state: %d\r\n", hdac-
>ErrorCode, hdac->State);
}

void HAL_DAC_DMAUnderrunCallbackCh1(DAC_HandleTypeDef *hdac)
{
  UART_Printf("DAC underrun! Error: %d, DAC state: %d\r\n",
hdac->ErrorCode, hdac->State);
}
```

Важливим аспектом роботи пристрою є реалізований у ньому алгоритм вибіркового оновлення дисплея. Для перевірки його роботи модифікуємо функцію ClearObject таким чином, щоб відображати оновлювані області кольором, відмінним від кольору фону.

Після збірки пристрою на макетній платі та завантаження програмного забезпечення, впевнюємося що система функціонує та відповідає заданим вимогам.

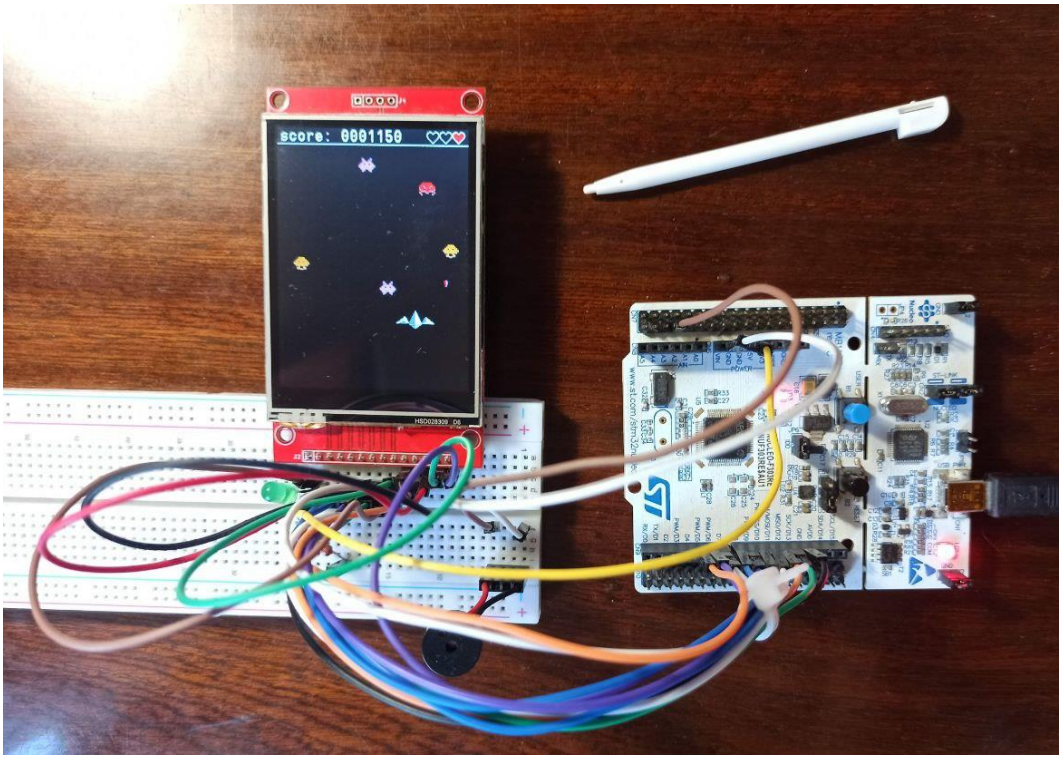

Рисунок 4.1 – Зовнішній вигляд пристрою під час роботи.

4.2 Аналіз результатів роботи пристрою

Під час роботи пристрою у консолі послідовного порту не бачимо повідомлень про недостатню швидкодію або помилки перетворення ЦАП, що свідчить про задовільну продуктивність системи та правильне налаштування периферійних пристроїв (рисунок 4.2).

```
Game initialized!
Game over! Score: 0000200
Restarting game...
Game initialized!
Game over! Score: 0001950
```

Рисунок 4.2 – Інформація, отримана у послідовному порту упродовж двох ігрових сесій

Добре себе показав алгоритм вибіркового оновлення дисплея. Його використання дозволило відображати динамічну картинку із декількома об’єктами, які рухаються одночасно із частотою оновлення 25 кадрів на секунду. Подібного було б неможливо досягти при використанні повного оновлення зображення із даним модулем дисплея. На рисунку 4.3 продемонстровано результат роботи даного алгоритму. Як можна побачити, рухомі об’єкти залишають за собою характерний слід, що відповідає перемальованим областям екрана (горизонтальна область оновлення показана синім кольором, вертикальна – блакитним).

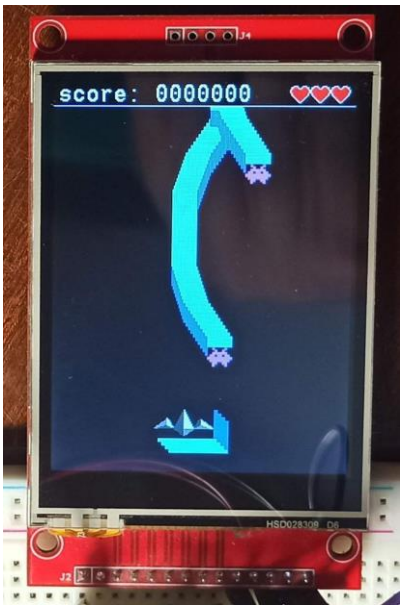

Рисунок 4.3 – Демонстрація роботи алгоритму часткового оновлення зображення

В рамках тестування було оцінено також і ігровий процес. Усі його аспекти, такі як створення та переміщення ворогів, керування та стрільба, зміна та відображення набраних очок та здоров’я а також відтворення звукових ефектів працюють належним чином.

До недоліків пристрою можна віднести дещо низьку чутливість сенсорної панелі через використання бюджетного модуля дисплея. Це викликає неточності введення, що не значно не впливають на ігровий процес. Оскільки сенсорна панель є резистивною, для надійного розпізнавання натискання слід використовувати наявний у комплекті із модулем стилус або інший загострений предмет. Це викликає незручності у порівнянні із ємнісними сенсорними панелями, що використовуються у сучасних споживчих пристроях.

В цілому отримані результати тестування є задовільними, оскільки вони свідчать про досягнення поставлених цілей стосовно продуктивності системи, виведення графіки та відтворення звукових ефектів. Наявні недоліки сенсорної панелі не мають значного впливу на ігровий процес.

4.3 Порівняння з аналогами

У порівнянні з електронними ігровими пристроями, що розглядалися в підрозділі 1.1 даної роботи, реалізований пристрій надає зображення кращої якості за рахунок використання кольорового дисплея з відносно високою роздільною здатністю. Також особливістю пристрою у порівнянні з аналогами можна назвати сенсорне керування.

До недоліків пристрою можна віднести можливість відтворення лише однієї запрограмованої гри. Даний аспект системи може бути виправлено, однак це потребуватиме значних змін до структури та програмної частини пристрою.

4.4 Потенційні напрями оптимізації

Після проведеної роботи над тестуванням пристрою можна заявити, що його слабким місцем є використаний модуль дисплея. Вбудована резистивна сенсорна панель є не надто зручною для натискання та забезпечує посередню точність введення у порівнянні з сучасними ємнісними дисплеями. Використання модуля дисплея із більш якісною сенсорною панеллю дозволило би покращити точність сенсорного введення. Також для покращення точності можливо використати алгоритм інтерполяції координат точок натискання, що дозволить зробити керування більш плавним.

Ще одним напрямком поліпшення пристрою могло би стати пришвидшення передачі даних на дисплей, що в свою чергу відкриває можливості для використання більш довершених візуальних ефектів. Це може бути досягнуто декількома способами. Використання інтерфейсу SPI з більшою тактовою частотою передбачає заміну мікроконтролера на більш продуктивний, оскільки використана модель STM32F303RE обмежена максимальною тактовою частотою протоколу SPI у 18 МГц. Кращим варіантом було б використання модуля дисплея з паралельним інтерфейсом, але це ускладнює підключення та програмну частину системи, а також передбачає більші фінансові витрати.

ВИСНОВКИ

В результаті виконання даної кваліфікаційної роботи було розроблено електронну гру на основі мікроконтролера STM32 з використанням сенсорного графічного дисплея. Було виконано усі стадії розробки, а саме аналіз вимог до пристрою, підбір електронних компонентів, створення програмного забезпечення для пристрою, збірка та тестування прототипу. Проведено дослідження результатів тестування, виявлено сильні та слабкі сторони системи, а також запропоновано можливі напрямки з її поліпшення.

У рамках роботи над проєктом було отримано та закріплено навички роботи з різними аспектами вбудованих систем. Зокрема, було охоплено роботу з периферійними пристроями через протокол SPI, опрацювання сенсорного введення, використання таймерів для обробки даних в режимі реального часу та застосування ЦАП для відтворення звуку. Усі ці навички можуть бути використані для роботи над іншими проєктами в галузі вбудованих систем, Інтернету речей тощо.

В рамках подальшого розвитку проєкту можливо як поліпшення апаратної частини, так і подальша робота над ігровим процесом. Це може включати покращення автономної роботи системи, встановлення більш довершеного модуля дисплея, додавання засобів збереження інформації, а також розробку нових ігрових рівнів, типів ворогів та режимів гри, поліпшення графіки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Handheld electronic game [Електронний ресурс] // Wikipedia : веб-сайт. – Режим доступу: https://en.wikipedia.org/wiki/Handheld_electronic_game (дата звернення: 09.05.2025).

2. Copetti R. Game Boy / Color Architecture, 2019 [Електронний ресурс]. – Режим доступу: https://www.copetti.org/writings/consoles/game-boy/ (дата звернення: 09.05.2025).

3. Arduboy [Електронний ресурс] : офіційний сайт. – Режим доступу: https://www.arduboy.com (дата звернення: 09.05.2025).

4. Pokitto - The Pokitto Guidebook [Електронний ресурс]. – Режим доступу: https://pokitto.github.io/pokitto (дата звернення: 09.05.2025).

5. Gamebuino – play & make games, quick & easy! [Електронний ресурс]. – Режим доступу: https://gamebuino.com (дата звернення: 09.05.2025).

6. Frame rate [Електронний ресурс] // Wikipedia : веб-сайт. – Режим доступу: https://en.wikipedia.org/wiki/Frame_rate (дата звернення: 12.05.2025).

7. Данилов І. М., Піскарьов О.М. Створення інтерактивної гри для мікроконтролерів STM32 з використанням графічного дисплея. // ITSBTU25 Інформаційні технології в сучасному світі [Електронний ресурс]: матеріали Міжнародної науково-практичної конференції здобувачів вищої освіти і молодих учених, 29 квітня 2025 р. / Державний біотехнологічний ун-т. – Харків, 2025. – С. 45-46.

8. Oshana R., Kraeling M. Software Engineering for Embedded Systems. – Amsterdam : Elsevier, 2013. – 1150 p. – Режим доступу: https://doi.org/10.1016/c2011-0-04586-6 (дата звернення: 12.05.2025).

9. Noviello C. Mastering STM32 : A Step-by-Step Guide to the Most Complete ARM Cortex-M Platform, Using a Free and Powerful Development Environment Based on Eclipse and GCC. – Victoria, British Columbia : Leanpub, 2016. – 849 p.

10. Martin T. The Insider’s Guide to the STM32 ARM®-Based Microcontroller. – Warwick : Hitex (UK) Ltd., 2009. – Режим доступу: https://www.hitex.com/fileadmin/assets/download/insiders-guides/stm32/isg-stm32-v18d-scr.pdf (дата звернення: 12.05.2025).

11. UM1724 User Manual. STM32 Nucleo-64 boards (MB1136). 2025. – 91 p. – Режим доступу: https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf (дата звернення: 14.05.2025)

12. STM32CubeIDE - Integrated Development Environment for STM32 – STMicroelectronics [Електронний ресурс] : офіційний сайт. – Режим доступу: https://www.st.com/en/development-tools/stm32cubeide.html (дата звернення: 15.05.2025).

13. Nystrom R. Game Programming Patterns. – Genever Benning, 2014. – 354 p.

14. GitHub - ardnew/ILI9341-STM32-HAL: ILI9341 color TFT display and touchscreen driver for STM32 using HAL SPI with DMA [Електронний ресурс]. – Режим доступу: https://github.com/ardnew/ILI9341-STM32-HAL (дата звернення: 20.05.2025)

15. GitHub - nure-danylov-i/stm32-ili9341-game: Інтерактивна гра для мікроконтролерів STM32 з використанням графічного дисплея [Електронний ресурс]. – Режим доступу: https://github.com/nure-danylov-i/stm32-ili9341-game (дата звернення: 05.06.2025)

16. Distorted bitmaps when not covering entire LCD · Issue #4 · ardnew/ILI9341-STM32-HAL [Електронний ресурс]. – Режим доступу: https://github.com/ardnew/ILI9341-STM32-HAL/issues/4 (дата звернення: 20.05.2025)