

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ВИКОРИСТАННЯ
ПРОГРАМОВАНИХ МАКРОСІВ НА ПЛАТФОРМІ ARDUINO
DEVELOPMENT OF AN APPLICATION FOR PROGRAMMABLE
MACROS ON ARDUINO PLATFORM

Виконав(ла): студент(ка) 4 курсу, групи КНТ-210СП

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ЯВОРСЬКИЙ Б.В.

(ПРИЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

Сергій СУББОТІН

« » 20 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЯВОРСЬКОГО Богдана Вікторовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка застосунку для використання програмованих макросів на платформі Arduino. Development of an Application for Programmable Macros on Arduino Platform

керівник проєкту (роботи) к.т.н., доцент, СТЕПАНЕНКО Олександр Олексійович

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «04» квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 01 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1 Аналіз проблем та постановка завдань дослідження. 2 Матеріали і методи. 3 Опис проєкту. 4 Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтроль	БЄЛОВА А. В., асистент		

7. Дата видачі завдання « 04 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки	2 тиждень	Розділ 2
4	Розробка архітектури програми	2 тиждень	Розділ 3
5	Розробка програми	3–4 тижні	Розділ 3,4
6	Тестування та експериментальне Дослідження програмного забезпечення	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та Документів до неї	6 тиждень	Додатки
8	Нормоконтроль та рецензування	7 тиждень	
9	Захист роботи	8 тиждень	

Студент(ка)

_____ Богдан ЯВОРСЬКИЙ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Олександр СТЕПАНЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 88 с., 32 рис., 5 табл., 4 дод., 27 джерел.

СКЕТЧ, C++, ARDUINO, ESP, MICROCONTROLLER, BLUETOOTH,
МАКРОС, ГЕЙМІНГ, ПРИСТРІЙ ВВОДУ, ЕЛЕКТРОНІКА.

Об'єкт дослідження – програмовані мікроконтролери.

Предмет дослідження – методи та інструменти розробки застосунків на платформі мікроконтролерів.

Мета роботи – розробка програмного продукту що дозволить використовувати макроси на платформі мікропроцесора на платі у якості допоміжного пристрою вводу.

Матеріали, методи та технічні засоби: інтернет-браузер Google Chrome середовище розробки Arduino IDE, мікропроцесор ESP32 на платі, дошка контактів, клавіатура на чотири кнопки.

Актуальність розробки обумовлена сучасними тенденціями до більшого розповсюдження роботи на персональному комп'ютері, стрімінгу та геймінгу, у чому сильно допомагають такі пристрої як стрім-дека що дозволяє швидко виконувати програмовані дії, для яких не достатньо одної клавіатури.

Вищеназвані проблеми можуть бути вирішені за допомогою самостійного створення пристрою на гнучкій платформі Arduino. Платформа мікропроцесорів може охопити дуже широкий спектр можливих потреб у людей у їх домівках, таких як: управління пристроями, управління розумним домом, система безпеки, невеликий веб сервер.

В результаті роботи було розроблено пристрій на основі мікропроцесору, який служить прикладом пристроїв, які можуть забезпечити різноманітні запити просунутих користувачів комп'ютерів, простий у

використанні та адаптований до використання на портативних пристроях через дистанційне керування.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 88 pages, 32 figures, 5 tables, 27 references, 4 appendixes.

SKETCH, C++, ARDUINO, ESP, MICROCONTROLLER, BLUETOOTH, MACROS, GAMING, INPUT DEVICE, ELECTRONICS.

The object of research is programmable microcontrollers. The subject of the research is methods and tools for developing applications on the microcontroller platform.

The purpose of the work is the development of a software product that will allow the use of macros on the microprocessor platform on the board as an auxiliary input device. Materials, methods and technical means: Google Chrome Internet browser, Arduino IDE development environment, ESP32 microprocessor on the board, contact board, four-button keyboard.

The relevance of the development is due to modern trends towards greater distribution of work on a personal computer, streaming and gaming. In this, such devices as a stream deck, which allows you to quickly perform programmed actions for which one keyboard is not enough, help a lot.

The above-mentioned problems can be solved by creating a device yourself on the flexible Arduino platform. The microprocessor platform can cover a very wide range of possible needs of people in their homes, such as: device management, smart home, security system, small web server.

The work resulted in a microprocessor-based device that exemplifies devices that can meet the diverse demands of advanced computer users, is easy to use, and is adaptable for use on portable devices through remote use.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	9
Вступ.....	10
1 Аналіз проблем та постановка завдань дослідження	12
1.1 Сучасність користувачів ПК та Стрімінг	12
1.2 Актуальність геймінгу	13
1.3 Огляд існуючих рішень	16
1.4 Постановка завдання.....	19
1.5 Висновки за розділом 1	19
2 Матеріали та методи	20
2.1 Вибір та обґрунтування мікропроцесору.....	20
2.2 Вибір та обґрунтування середовища розробки.....	23
2.3 Вибір та обґрунтування мови програмування	28
2.4 Висновки за розділом 2	31
3 Опис програми.....	32
3.1 Проектування та реалізація апаратної частини.....	32
3.2 Налаштування середовища розробки.....	36
3.3 Розробка програмної частини проекту (скетч)	42
3.4 Висновки за розділом 3	47
4 Експлуатація, тестування та експериментальне дослідження проекту	48
4.1 Призначення та умови застосування пристрою.....	48
4.2 Інструкція користувача по експлуатації	48
4.3 Тестування продукту та результати роботи пристрою	49
4.4 Висновки за розділом 4	55
Висновки	56
Перелік джерел посилання	57
Додаток А Технічне завдання	60
Додаток Б Опис проекту.....	65

Додаток В Текст програми 69

Додаток Г Слайди презентації 81

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

ESP-IDF– Espressif IoT Development Framework;

GPIO - General Purpose Input-Output;

IDE – Integrated Development Environment;

ПЗ – програмний застосунок;

ПК – персональний комп'ютер;

СР – середовище розробки.

ВСТУП

У сучасному світі технології все більше проникають у різні сфери нашого життя. Однією з таких областей, де технології відіграють ключову роль, є персональне використання комп'ютерів. Просунуті користувачі, стрімери та геймери постійно шукають нові способи для покращення своєї ефективності та зручності. Один із способів досягнення цієї мети - використання програмованих макросів.

Мікроконтролери, зокрема платформа Arduino, можуть надати велику допомогу в цьому. Arduino - це відкрита платформа, що базується на простому та зручному у використанні обладнанні та програмному забезпеченні, які дозволяють створювати різноманітні проекти, від простих до складних.

Об'єктом дослідження є процес розробки застосунку для використання програмованих макросів на платформі Arduino. Предметом дослідження є методи та інструменти розробки застосунків на платформі мікроконтролерів.

Метою даної дипломної роботи є розробка функціонального та ефективного застосунку для використання програмованих макросів на платформі Arduino для просунутих користувачів, геймерів та стрімерів.

Особливу актуальність темі надає той факт, що розробка застосунку для використання програмованих макросів на платформі Arduino може значно полегшити процес використання комп'ютера для просунутих користувачів, геймерів та стрімерів. Враховуючи це, результати цієї роботи можуть бути корисними не лише для наукової спільноти, але і для широкого кола потенційних користувачів.

В кінці роботи було розроблено застосунок, який буде не лише функціональним та ефективним, але й простим у використанні для користувачів, що не мають глибоких знань у програмуванні та роботі з мікроконтролерами.

Робота складається з чотирьох основних розділів кожен з яких вирішує певні завдання та спрямований на досягнення поставленої мети. По завершенні роботи будуть сформульовані висновки, які відображають основні результати дослідження та їх значення для потенційних користувачів.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

В першому розділі буде проведено аналіз предметної області. Буде досліджено поняття макросу в використанні ПК та мобільних пристроїв у стрімінгу.

За результатами роботи буде сформовано задачі для проектного пристрою.

1.1 Сучасність користувачів ПК та стрімінг

Стрімінг, або трансляція в реальному часі, стає все більш популярним засобом споживання контенту. Використовуючи платформи, такі як Twitch, YouTube, Facebook Live та інші, люди з усього світу мають можливість надавати та переглядати контент у режимі реального часу. Є кілька ключових факторів, які сприяють зростаючій популярності стрімінгу:

- вплив геймінгу. Геймінг став одним з основних каталізаторів популярності стрімінгу. За допомогою стрімінгу геймери можуть показувати свої навички, навчати інших, або гарно проводити час разом зі своєю аудиторією;

- поява нових технологій. Покращення в технологіях передачі даних та доступ до швидкісного інтернету дозволяють транслювати відео високої якості без значних затримок, що поліпшило досвід користувача;

- зміна споживання медіа. Все більше людей віддають перевагу перегляду контенту на вимогу або в реальному часі, замість традиційного телебачення. Стрімінг дозволяє глядачам відстежувати події в реальному часі та більш живо взаємодіяти з творцями контенту;

- монетизація і кар'єра. Для творців контенту стрімінг став важливим засобом монетизації та навіть професійної кар'єри. Через підписки, рекламу,

спонсорство та донати стрімери можуть заробляти достатньо, щоб робити стрімінг своєю основною роботою.

Arduino є однією з найбільш популярних платформ для розробки на базі мікроконтролерів. Ця популярність обумовлена рядом ключових чинників:

- Arduino плати доступні по всьому світу за розумними цінами, що робить їх доступними для широкого кола людей, включаючи студентів, хобістів, митців і інженерів;
- Arduino має простий та інтуїтивний інтерфейс. Arduino IDE, основна середовище розробки для Arduino, було спеціально розроблене для того, щоб зробити процес програмування якомога простішим, особливо для новачків;
- навколо мікропроцесорів сформувалась величезна глобальна спільнота користувачів, яка надає величезну кількість ресурсів, включаючи навчальні матеріали, бібліотеки, проекти та форуми для допомоги та обговорення;
- Arduino часто використовується в освітніх цілях для викладання програмування та електроніки. Багато шкіл та університетів використовують Arduino у своїх курсах;
- програмне забезпечення та апаратні схеми Arduino є відкритими, що дозволяє користувачам легко модифікувати та адаптувати їх до своїх потреб.

Завдяки цим перевагам, Arduino стала ключовою платформою для електроніки, створюваною своїми руками і мікроконтролерних проектів.

1.2 Актуальність геймінгу

Геймінг – це присвячення часу гравню в відео ігри на пристрої типу персонального комп'ютера або телефона.

За останні п'ятнадцять років геймінг перетворився з нішевої галузі розваг у цілу культуру, покоління молодих геймерів вже виросло до середнього віку і мають теплі почуття до речей що давали їм хороший настрій

у дитинстві.

Такий бурний ріст геймінгу пов'язаний з швидким розвитком технологій та програмного забезпечення. Розвиток потужностей сучасних компонентів для комп'ютерів та їх відносна дешевизна через розповсюдженість роблять цю галузь квітучою поляною для створення нових медіа технологій таких як комп'ютерна графіка та 3D моделювання.

Геймінг-індустрія розвивається шаленими темпами. Ще нещодавно вершиною технологічної думки було відкриття ери HD у консолях PlayStation 3. А зараз геймерам світу пропонуються всі можливості віртуальної реальності та штучного інтелекту. Гравці можуть спробувати себе у розробці ігор завдяки відкритому коду таких ігрових рушіїв як Unity та Unreal Engine, стрімити на сервіси на зразок Twitch чи YouTube, заробляючи соціальний та грошовий капітал. Онлайн-ігри зараз – це не доля інтровертів-одиначок. Це чудова можливість для розвитку у медіасфері [1].

Геймінг також став одним з шляхів самореалізації людей у мережі, особливо у час пандемії COVID-19.

З появою нових технологій з'являються нові професії. Кібер-спорт перетворив вчорашніх школярів на зірок геймінгу, які заробляють немалі гроші. Самі геймери стверджують, що прекрасно розуміють, що ця професія не назавжди, і паралельно займаються іншими справами, саморозвиваються. Психологи ж кажуть, що колись складно було б уявити собі таку картину, але тепер заробіток на відеоіграх – цілком нормальна ситуація [2].

Багато людей зараз пов'язують своє життя зараз з геймінгом, деякі люди вибрали своєю професією стрімінг ігор, а деякі працюють у кіберспорті і стають тренерами професійних гравців або коментаторами.

За даними дослідження крупного партнера в області цифрової комерції PayPal цифрові звички швидко змінилися через карантинні заходи та соціальне дистанціювання під час пандемії, в результаті чого на live-стрімінг платформах різко зросла кількість користувачів [3].

62% респондентів повідомили, що очікують зростання доходів від стрімів після того, як мине криза, і тільки 11% вважають, що їхні доходи зменшаться. 51% повідомили, що вони заробляють в цій сфері менше ніж рік, що говорить про те, що прямі трансляції почали стрімко зростати ще до появи пандемія COVID-19. Щодо тих, хто заробляє на стрімінгових платформах, 38% очікували, що їх прибуток збільшиться під час кризи, а 25% думали, що будуть заробляти приблизно стільки ж [3].

З урахуванням цих факторів, популярність стрімінгу має тенденцію зростання. Важливо зауважити, що, хоча стрімінг почався як явище, пов'язане переважно з геймінгом, він тепер розширюється на багато інших областей, включаючи музику, спорт, освіту та інше [3].

Так як стрімінг та технології все набирають популярність, таке явище як геймінг продовжить рости і збільшувати розповсюдженість.

1.3 Огляд існуючих рішень

Програмовані клавіатури представлені у доволі широкому асортименті на таких платформах як aliexpress та ebay. Частіше за все їх роблять своїми руками із підручних матеріалів та електроніки що можна купити у інтернет та звичайних магазинах.

На рисунку 1.1 розглянуті існуючі рішення на платформі інтернет магазину aliexpress.

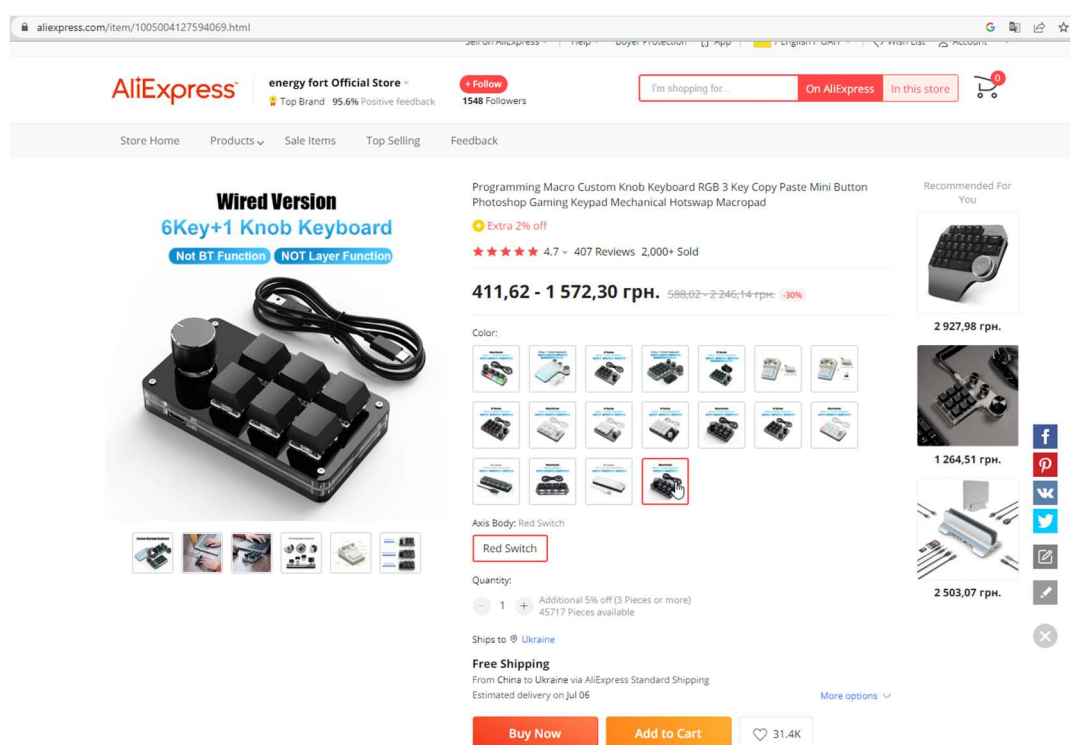


Рисунок 1.1 – Програмована клавіатура у різних конфігураціях [4]

На рисунку 1.2 показаний інтерфейс забезпечення Oscar від a4tech ексклюзивно для мишки X7.



Рисунок 1.2 – Інтерфейс програми Oscar a4tech [5]

На рисунку 1.3 показаний інтерфейс забезпечення для китайської програмованої клавіатури.

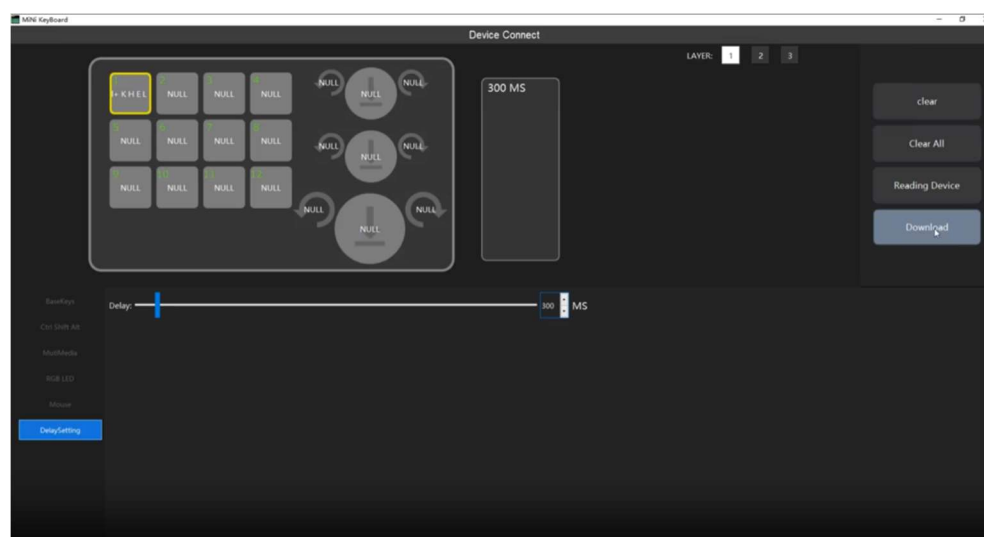


Рисунок 1.3 – Інтерфейс ПЗ для китайської клавіатури

На рисунку 1.4 показаний інтерфейс забезпечення для клавіатури a4tech Bloody.



Рисунок 1.4 – Інтерфейс забезпечення для клавіатури a4tech Bloody [6]

Слід виділити кілька факторів принципіальної різниці між існуючими товарами та проєктованим пристроєм:

- існуючі товари це відполірований комерційний вузько направлєне забезпечення для фірмового пристрою;
- для створення були застосовані виробничі потужності та задіяні спеціалісти програмування.

Проектований пристрій має переваги у насамперед ціні, а також можливості адаптації виробу під себе, має майже необмежені можливості по застосуванню у побуті. На платформі Arduino люди без особливих складностей проєктують такі забезпечення як розумний дім, систему безпеки, систему контролю приладів, полив рослин та інше. Для цього не потрібно мати навичок професійного розробника, досить мати наполегливість і вміння знаходити інформацію в інтернеті.

Таблиця 1.1. – Порівняння існуючих аналогів ПЗ

Забезпечення	Ціна	Гнучкість	Зручність	Потенціал	Надійність
Саморобний пристрій	+	+	-	+	-/+
Покупна клавіатура	-/+	+	-/+	-	+
Oscar a4tech	-	-	+	-	+
Bloody a4tech	-	-	-	-	+

1.4 Постановка завдання

Проаналізувавши все вищезазначене було прийняте рішення розробки пристрою, що може імітувати роботу пристрою вводу на основі мікропроцесора на платі що дозволить під час розробки тонко налаштувати потреби користувача і дослідити технології втілення подібного пристрою.

1.5 Висновки за розділом 1

В першому розділі було проведено аналіз предметної області, дослідження актуальності стрімінгу у часи пандемії. Було досліджено поняття макросу та його застосування у стрімінгу.

За результатами роботи буде сформовано задачі для проектованого пристрою.

2 МАТЕРІАЛИ ТА МЕТОДИ

Другий розділ буде присвячено висвітленню вибору апаратної та програмної частини проєкту. Буде проведено дослідження основних переваг та недоліків серед можливих апаратних рішень та середовища розробки і мови програмування.

2.1 Вибір та обґрунтування мікроконтролеру

Серед мікропроцесорів кандидатів для проєкту підходять такі платформи як Arduino Micro, Arduino Leonardo, Arduino Uno, ESP8266 та ESP32.

Розглянемо характеристики кожної платформи почавши з Arduino Uno.

Arduino Uno - це широко використовувана плата мікроконтролерів з відкритим кодом на базі мікроконтролера ATmega328P. У його склад входить все необхідне для зручної роботи з мікроконтролером: 14 цифрових входів/виходів (з них 6 можуть використовуватися в якості ШІМ-виходів), 6 аналогових входів, кварцовий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм для програмування всередині схеми (ICSP) і кнопка скидання. На відміну від всіх попередніх плат Arduino Uno в якості перетворювача інтерфейсів USB-UART використовує мікроконтролер ATmega16U2 (ATmega8U2 до версії R2) замість мікросхеми FTDI [7].

Arduino Leonardo- це пристрій на базі мікроконтролера ATmega32U4). До його складу входить все необхідне для роботи з цим мікроконтролером: 20 цифрових входів/виходів (7 з яких можуть працювати як ШІМ-виходи, 12 - як аналогові входи), кварцовий резонатор на 16 МГц, роз'єм мікро-USB, роз'єм живлення, роз'єм для внутрішньосхемного програмування ICSP (In-Circuit Serial Programming) та кнопка скидання. Для початку роботи як з Leonardo так і з Uno/Micro досить просто подати живлення від AC/DC-

адаптера або батареї або підключити їх до комп'ютера за допомогою USB-кабелю [7].

Arduino Micro - це пристрій на основі мікроконтролера ATmega32u4 (datasheet), розроблений спільно з Adafruit. До його складу входить все необхідне для зручної роботи з мікроконтролером: 20 цифрових входів/виходів (з них 7 можуть використовуватися як ШИМ-виходи, 12 - як аналогові входи), кварцовий резонатор на 16 МГц, роз'єм micro-USB, роз'єм ICSP для внутрішньосхемного програмування та кнопка скидання. Пристрій розроблений таким чином, щоб його можна було зручно розміщувати на макетній платі. В платах Leonardo та Micro використовується мікропроцесор ATmega32u4 з вбудованим контролером USB. Таке рішення виключає використання додаткового контролера, і при підключенні до комп'ютера дозволяє Arduino визначатися в системі як звичайна миша, клавіатура або віртуальний COM-порт [7].

ESP8266 та ESP32 - це недорогі Wi-Fi модулі, що ідеально підходять для аматорських проектів в області інтернету речей та домашньої автоматизації. Обидва чіпи мають 32-бітний процесор, ESP8266 – це одноядерний процесор, що працює на частоті 80 МГц тоді як ESP32 – двоядерний процесор із частотою 160...240 МГц. Ці модулі поставляються з виводами GPIO (інтерфейс вводу/виводу), що підтримують широкий спектр протоколів, таких як SPI, I2C, UART, ADC, DAC, PWM [8].

Найприємніше те, що ці плати поставляються з інтегрованою бездротовою мережею, що відрізняє їх від інших мікроконтролерів, таких як Arduino. Це означає, що ви можете легко керувати та контролювати пристрої віддалено через Wi-Fi за дуже низькою ціною. Це дуже корисно у таких проектах як розумний дім чи система безпеки. Крім того, якщо вам не потрібно використовувати його бездротові можливості, ви можете використовувати ESP32/ESP8266 для управління входами та виходами, як це було б з Arduino [8].

ESP32 є наступником ESP8266. Він має додаткове ядро процесора, швидший Wi-Fi, більше GPIO і підтримує Bluetooth 4.2 і Bluetooth з низьким енергоспоживанням. Крім того, ESP32 поставляється із сенсорними контактами, які можна використовувати для пробудження ESP32 із глибокого сну. Обидві плати дуже дешеві, але ESP32 коштує трохи дорожче. У той час як ESP32 може коштувати від 6 до 12 доларів, ESP8266 може коштувати від 4 до 6 доларів [8].

Приведено таблицю 2.1 з основними характеристиками мікропроцесорів на платах.

Таблиця 2.1. – Порівняння мікроконтролерів

Плата	Чіп	Т.частота	Пам'ять	Порти	Bluetooth	Wi-Fi
Uno	ATmega328	16 МГц	32 КБ	14	-	-
Leonardo	ATmega32u4	16 МГц	32 КБ	20	-	-
Micro	ATmega32u4	16 МГц	32 КБ	20	-	-
ESP8266	1Core 32-bit L106	80 МГц	16 МБ	17	-	+
ESP32	2Cores 32-bit LX6 600	160 МГц	16 МБ	36	+	+

Порівнявши характеристики приведених плат, було вибрано ESP32 з декількох причин:

- вбудований Bluetooth і Wi-Fi: ESP32 має вбудовані Bluetooth та Wi-Fi, що дозволяє йому з'єднуватися безпосередньо з іншими пристроями без необхідності додаткового обладнання. Це відрізняє його від таких мікроконтролерів як Arduino Uno, Arduino Nano та Arduino Leonardo, які не мають цих можливостей за замовчуванням;
- двоядерний процесор. ESP32 має двоядерний процесор, що робить його більш потужним, ніж Arduino Uno, Nano, Leonardo та ESP8266.

Це дозволяє ESP32 обробляти більш складні задачі та швидше працювати у багатозадачності;

- більше GPIO портів. ESP32 має більше портів введення/виведення (GPIO), ніж Arduino Uno, Nano або Leonardo. Це дає більше гнучкості для підключення додаткових сенсорів, кнопок, світло діодів тощо;

- сумісність з Arduino IDE. ESP32 сумісний з Arduino IDE, що дозволяє розробникам використовувати вже знайоме середовище програмування. Однак, він також має підтримку інших середовищ, таких як ESP-IDF, що робить його більш гнучким для розробки;

- відносно низька вартість. На відміну від більшості Arduino плат, ESP32 має відносно низьку вартість, що робить його доступним для більшості проєктів та не потребує таких додаткових адаптерів як Bluetooth та Wi-Fi, так як вони вже вбудовані.

З урахуванням всіх цих чинників, ESP32 є відмінним вибором для розробки проєктів, які вимагають високої продуктивності, бездротового з'єднання, економії коштів та великої кількості портів GPIO що дозволить мати простір для покращення або переробки проєкту.

2.2 Вибір та обґрунтування середовища розробки

Середовищем розробки для сімейств плат Ардуїновиступає Arduino IDE.

Arduino IDE—інтегроване середовище розробки для Windows, Mac OS та Linux, розроблене на C та C++, призначене для створення та завантаження програм на Arduino-сумісні плати, а також на плати інших виробників.

Вихідний код середовища випущений під загальнодоступною ліцензією GNU. Підтримує мови C та C++ з використанням спеціальних правил структурування коду. Arduino IDE надає бібліотеку програмного

забезпечення з проекту Wiring[en], яка надає безліч загальних процедур введення та виведення. Для написаного користувачем коду потрібні лише дві базові функції для запуску ескізу та основного циклу програми, які скомпільовані та пов'язані з заглушкою програми `main()` у виконувану циклічну програму з ланцюжком інструментів GNU, також включеною до дистрибутиву IDE. Середовище використовує програму `avrdude` для перетворення коду, що виконується, в текстовий файл у шістнадцятковому кодуванні, який завантажується в плату Arduino програмою-завантажувачем у вбудованому програмному забезпеченні плати.

Зі зростанням популярності Arduino інші постачальники в якості програмної платформи почали впроваджувати компілятори та інструменти з відкритим вихідним кодом (ядра), які можуть створювати і завантажувати ескізи в інші мікроконтролери, що не підтримуються офіційною лінійкою мікроконтролерів Arduino [9].

ESP-IDF це портована на ESP32 версія FreeRTOS. Мови програмування C і C++, ніякої принципової різниці в них між середовищами немає. Якщо ви навчилися непогано розумітися на скетчах Arduino, то освоювати принципово нову мову вам не доведеться. Різниця буде помітна лише у підходах до програмування для однозадачного середовища та багатозадачного [11].

FreeRTOS - це хмарна операційна система реального часу з відкритим вихідним кодом, в якій використовується швидке, надійне та чуйне ядро. FreeRTOS можна безкоштовно використовувати за ліцензією Массачусетського технологічного інституту (MIT) на програмне забезпечення з відкритим кодом. Для цієї системи розроблено більше 40 варіантів архітектури, що надає розробникам широкий вибір апаратного забезпечення разом із набором готових програмних бібліотек [10].

У звичній для багатьох Arduino головним завданням кожної з функцій у головному циклі `loop()` було якнайшвидше завершити свою роботу і передати керування наступною. Взаємодія між різними частинами програми

здійснювалася за допомогою callback-ів. Використання `delay()` вважається поганою практикою, оскільки це безпосередньо вплине на інші частини вашої програми. У багатозадачному середовищі кожен із модулів вашої прошивки працює по черзі, коли ОС виділяє задачі квант процесорного часу.

У багатозадачному середовищі кожен із модулів вашої прошивки працює по черзі, коли ОС виділяє задачі квант процесорного часу. І, в принципі, `delay()` не заважає іншим процесам, проте його використання все одно можна вважати небажаним. Коли якась із завдань завершила черговий цикл своєї роботи (наприклад відправила HTTPS-запит), її слід відправити в очікування наступної "вхідної" події, і тоді ОС не буде "порожньо" виділяти їй процесорний час [11].

Atmel Studio 7 - це платформа для розробки та налагодження всіх програм для мікроконтролерів AVR та SAM. Atmel Studio 7 є середовищем для написання, складання та налагодження додатків, написаних на C/C++ або асемблері. Він також має можливість підключається до налагоджувачів, програмістів та наборів для розробки, які підтримують пристрої AVR та SAM. Atmel Studio включає Atmel Gallery, онлайн-магазин додатків, який дозволяє розширити середовище розробки за допомогою плагінів, розроблених Microchip, а також сторонніми розробниками. Платформа підтримує понад 500 пристроїв AVR та SAM [12].

Arduino IDE та ESP IDF представляють два різних підходи до розробки на платформі ESP через їх спосіб роботи з багатозадачністю.

Проте можна виділити декілька ключових переваг Arduino IDE перед ESP IDF чи Atmel Studio:

- Arduino IDE відомий своєю простотою використання. Інтерфейс користувача дружній та інтуїтивний, що робить його неймовірно простим для вивчення для новачків;

- Arduino IDE сумісний з ширшим спектром мікроконтролерів ніж IDF, практично всіма контролерами Arduino та ESP і контролерами інших

виробників. Це означає, що можна використовувати одне і те ж середовище розробки для багатьох різноманітних проєктів;

- Arduino має велику активну спільноту, яка може допомогти при вирішенні майже любых проблем, а також надати багато ресурсів для навчання та розвитку;

- для Atmel Studio та ESP IDF треба додатково встановлювати текстовий редактор або середу типу Visual Studio для написання програмного забезпечення, а вбудований текстовий редактор IDE дуже легкий та зручний в застосуванні;

- у ESP IDF немає драйверів для всіляких датчиків та екранів, через що їх може бути потрібно писати користувачу. Виробники плат самі створюють драйвера для Arduino IDE через її надзвичайну популярність;

- Arduino IDE має велику кількість вбудованих дуже зручних бібліотек, що підтримують широкий спектр сенсорів, модулів та інших компонентів.

Багатозадачність це основна перевага ESP IDF перед Arduino IDE, Структура цього забезпечення спеціально зроблена для досягнення легкої взаємодії мікропроцесору у режимі багатозадачності [11].

Підтримка Atmel Studio більшої кількості мов програмування дає більше можливостей для виконання специфічних більш професійних задач. При компіляції однакові програми за функціоналом можуть займати від 2 до 5 разів більше місця на пам'яті пристрою на Arduino IDE ніж на мовах C та Assembler у середовищі Atmel [13].

У порівнянні з ESP-IDF та Atmel Studio, Arduino IDE може виявитися менш потужним та гнучким для деяких додатків, проте його простота використання, широка сумісність і всесвітня розповсюдженість роблять його ідеальним вибором для багатьох проєктів, особливо для тих, хто тільки починає працювати з ESP32.

У таблиці 2.2 приведено порівняння декількох можливих середовищ розробки.

Таблиця 2.2 – Порівняння середовищ розробки

Категорії	Arduino IDE	ESP IDF	Atmel Studio
Кількість матеріалів	+	-	-
Зручність використання	+	-	-
Багатозадачність	-	+	-/+
Відкритий код	+	+	-
Ефективність використання пам'яті	-	-/+	+
Вбудовані текстовий редактор та бібліотеки	+	-	-
Підтримка різноманітних датчиків	+	-	+

Враховуючи все це, Arduino IDE становить відмінний вибір для розробки проектів на базі мікроконтролерів за рахунок своєї легкої доступності для всіх користувачів та безлічі матеріалів для навчання.

2.3 Вибір та обґрунтування мови програмування

Інтегроване середовище розробки Arduino це додаток на Java для різних платформ, що включає в себе редактор коду, компілятор і модуль передачі прошивки в плату. Середовище розробки засноване на мові програмування Processing та спроектоване для програмування новачками, не знайомими близько з розробкою програмного забезпечення. Мова програмування аналогічна мові Wiring. Загалом, це C++, доповнений деякими бібліотеками, такими як AVR Libc [15] [14].

Програми обробляються за допомогою препроцесора, а потім компілюються за допомогою AVR-GCC.

Програми Arduino пишуться на мові програмування C або C++. Середовище розробки Arduino поставляється разом із бібліотекою програм «Wiring».

C++ – мова програмування яка відноситься до мов вищого рівня, порівняно з Асемблером. Програма цією мовою краще зрозуміла людині. Перевагою C/C++ є величезна кількість програмних засобів і бібліотек, що дозволяють просто створювати необхідний код. Фактично, C/C++ став основною мовою розробки керуючих програм. Компілятори цієї мови реалізовані практично для всіх моделей мікроконтролерів. Мова C/C++ має досить складну для вивчення структуру. Отримуваний машинний код конкретної задачі має більший обсяг, ніж код тієї ж задачі, реалізованої на Асемблері. Проте мову C/C++ слід визнати єдиним правильним вибором професійного програмування мікроконтролерів [16].

Асемблер – мова найнижчого рівня. При цьому він дозволяє найбільш повно розкрити всі можливості мікроконтролерів та отримати максимальну швидкодію та компактний машинний код. У деяких випадках альтернативи асемблеру немає, але він має безліч недоліків. Незважаючи на отримувану компактність машинного коду, програма, написана мовою Асемблер, громіздка і без знання мови абсолютно не зрозуміла. Для її створення

потрібне відмінне знання архітектури та системи команд мікроконтролерів. Асемблер відмінно підходить для програмування мікроконтролерів, що мають обмежені ресурси, наприклад, 8-ми бітних моделей з малим об'ємом пам'яті. Для великих програм, прототипів і 32-розрядних контролерів краще використовувати інші мови, що відрізняються вищим рівнем. Це дозволить створювати складніші і при цьому зрозумілі програми [16].

Pascal – ще більш зручна мова для сприйняття та вивчення. Проте він не має такого поширення як C/C++, особливо при програмуванні мікроконтролерів. Деякі окремі фірми підтримують цю мову з метою спрощення переходу на контролери з великих ПК. Зокрема варіант мови під назвою Micro PASCAL входить до складу постачання засобів налагодження фірми Mikroelektronika [16].

Безперечним плюсом Arduino IDE є легкість опанування. Для того, щоб зробити перший крок і моргнути світлодіодом, не потрібно вивчати мову програмування, не потрібно вивчати архітектуру мікроконтролера та займатися його ініціалізацією. Ще один незаперечний плюс Arduino – безліч прикладів у вільному доступі. Платформа підтримує дуже велику кількість різних датчиків і контролерів, для швидкого запуску нового датчика в більшості випадків достатньо лише скачати готову бібліотеку під цей самий датчик. Це дуже заощаджує час [17].

Головний мінус Arduino IDE - немає розуміння як працює мікроконтролер, для програміста він залишається "чорним ящиком". Зіткнувшись з тим, що в реальному часі потрібно працювати з величезним масивом інформації з зовнішніх датчиків і управляти кількома моторчиками, програмісту буде важко написати код так, щоб все працювало точно і безвідмовно. Ще один мінус Arduino IDE - це те, як це середовище використовує ресурси мікроконтролера. Якщо необхідно багато всього завантажити в мікроконтролер або працювати в режимі реального часу, то можуть виникнути труднощі. Проблема вирішується вибором потужнішого контролера. На C/C++ подібна проблема спочатку відсутня, тому що

середовище розробки не додає в код те, чого туди не пише програміст, що дозволяє більш раціонально використовувати ресурси мікроконтролера [17].

У таблиці 2.3 приведено порівняння мов програмування які можна застосовувати у програмуванні мікроконтролерів.

Таблиця 2.3 – Порівняння мов програмування

Критерій	Мова Arduino	C/C++	Assembler	Pascal
Оптимізація пам'яті	-	-/+	+	-
Підтримка драйверів	+	-/+	-	-
Документованість	+	-/+	-	-
Легкість в опануванні	+	-/+	-	+
Можливості налаштування	-	-/+	+	-
Середовище для використання	+	-	-	-

Загалом мова програмування Arduino— це найшвидший та найпростіший спосіб написання будь-якої програми на мікроконтролері. Він не потребує особливих навичок в опануванні на початку та створення штучних персональних пристроїв або дає можливість зробити прототип пристрою для більш масового продукту.

Якщо ж ціль проекту масовість та максимальна ефективність застосування ресурсів – найкращий вибір C++ з додаванням елементів Assembler, але для цього потрібен програміст-професіонал, який витратив роки на вивчення та отримання досвіду.

2.4 Висновки за розділом 2

Другий розділ було присвячено висвітленню технічних питань проєкту. Був вибраний та аргументований мікроконтролер ESP32 під зазначені задачі, виявленні основні переваги та недоліки. Проведений вибір середовища та мови програмування Arduino IDE що будуть застосовані для розробки програмної частини.

3 ОПИС ПРОГРАМИ

У цьому розділі буде проводитись проектування та реалізація всіх частин проекту для виконання задачі поставленої у розділі 1. Спочатку відбудеться проектування апаратної частини для розуміння з чим треба працювати при проектуванні програмної частини.

3.1 Проектування та реалізація апаратної частини

За основу пристрою було обрано ESP32 у розділі 2. На рисунку 3.1 приведено схему портів вводу/виводу мікроконтролера ESP32 на платі розробника DevKit v1, що буде застосовано у майбутньому проєктуванні системи.

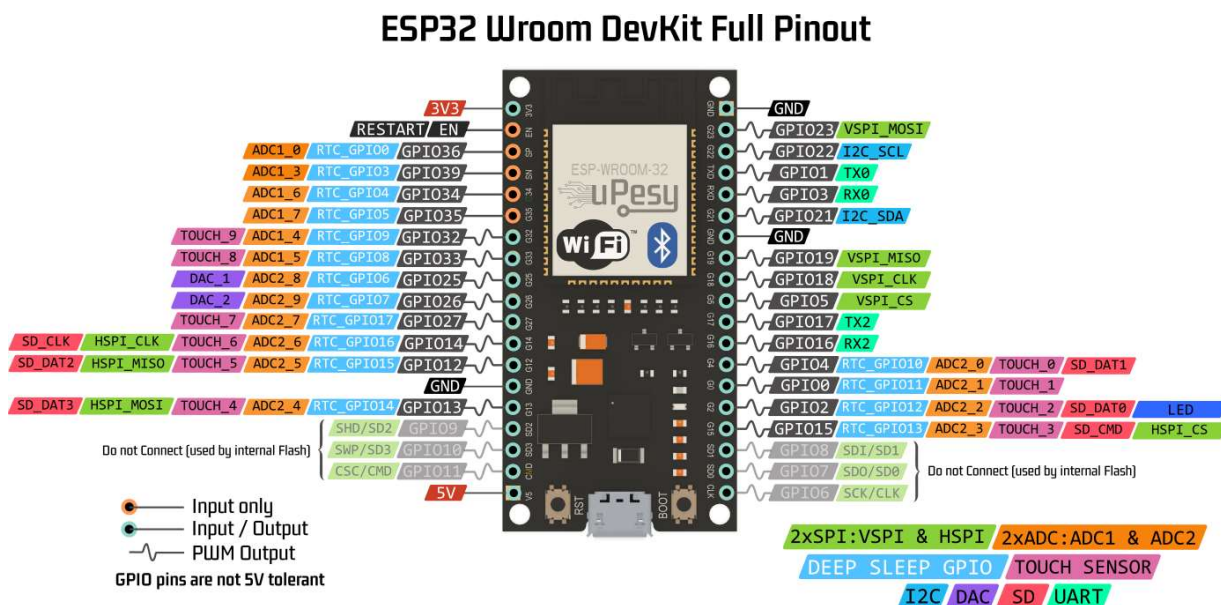


Рисунок 3.1 – Розпіновка (pinout) плати [18]

Порти на платі під назвою EN, SP, SN, G34, G35 та позначені помаранчевим кольором це порти тільки для зчитування.

GPIO - Інтерфейс введення/виведення загального призначення. Контакти GPIO можуть бути як у ролі входу, і у ролі виходу — це, зазвичай, конфігурується.

Контакти під номерами CMD, SD0, SD1, SD2, SD3, CLKрекомендується не підключати через те що вони використовуються внутрішньою пам'яттю контролера.

Також плата має внутрішні резистори, під'єднані до живлення всередині мікросхеми, якими можна керувати. Завдяки цьому підключення контактів можна визначити типом INPUT_PULLUP в програмній частині, тож не буде потрібно використовувати зовнішні резистори.

Після натиснення кнопки плата обробляє сигнал та виконує логічні операції, що при викликанні функції digitalRead(номер контакту), повертають значення 1 або 0 (true/false).

Найпростіша у визначенні константа – false, означає нуль (0).

Часто вважають, що константа true означає 1, що є вірним, проте true має ширше значення. Будь-яке ціле число, що не дорівнює 0, логічно є істиною (true). Тому числа -1, 2 та -200 у двійковій алгебрі будуть також вважатися істиною. Зверніть увагу, що константи true і false пишуться у нижньому регістрі, на відміну констант HIGH, LOW, INPUT і OUTPUT [19].

Для зчитування команд користувача було використано Arduino матричну клавіатуру 1x4 приведену на рисунку 3.2.

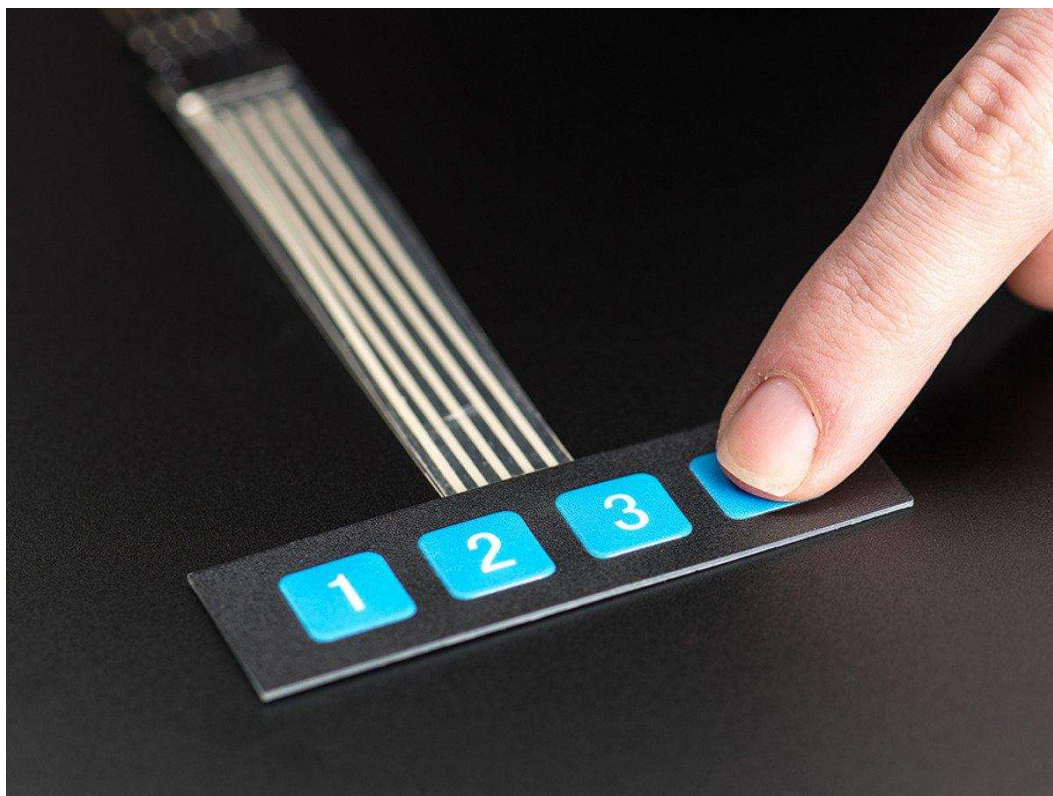


Рисунок 3.2 – Матрична клавіатура 1x4 [20]

Практичним методом були встановлені контакти на екземплярі клавіатури що була використана в проєкті і на рисунку 3.3 зазначена схема підключення контактів плати до клавіатури у форматі G(контакт на платі) – номер кнопки.

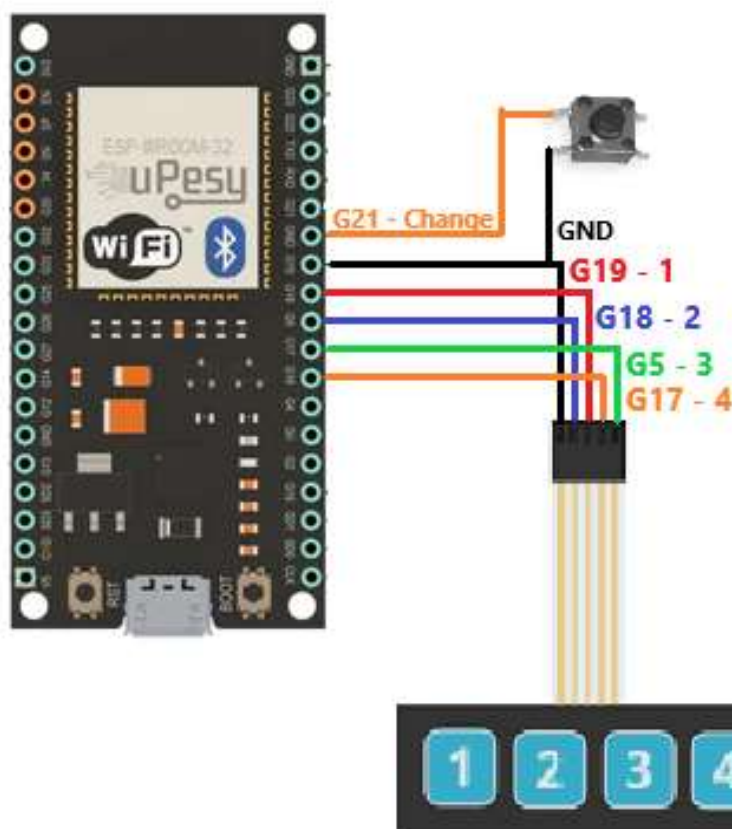


Рисунок 3.3 – Схема під'єднання контактів плати до клавіатури [21]

Для початку роботи з платою все готово, залишилось тільки під'єднати її до комп'ютера за допомогою кабеля USB та почати працювати над програмною частиною у середовищі розробки Arduino, у наступному розділі буде описано середовище та написана інструкція по налаштуванню її для роботи з платою а також початок написання програми.

Як можна побачити, на платі DevKit залишилось ще багато незайнятих контактів що можна буде використати для можливої модернізації проєкту, такої як наприклад підключення більшої клавіатури 4x4, додаткового дисплею для відображення виконання роботи, світлодіоду що сигналізує про режим роботи або датчиків.

3.2 Основні функції та налаштування середовища розробки

Інтерфейс середовища розробки Arduino містить такі основні елементи: текстовий редактор для написання коду, область для виведення повідомлень, текстова консоль, панель інструментів із традиційними кнопками та головне меню. Даний софт дозволяє комп'ютеру взаємодіяти з Arduino як для передачі даних, так і прошивки коду в контролер.

Програми, створювані у середовищі розробки Arduino іноді ще називають скетчами. Скетчі пишуться у текстовому редакторі та зберігаються у файлах з розширенням .ino. Вбудований текстовий редактор має стандартні інструменти копіювання, вставки, пошуку та заміни тексту. Область повідомлень у вікні програми є свого роду зворотним зв'язком для користувача та інформує його про події (у тому числі і про помилки), що виникають у процесі запису або експорту написаного коду. Консоль відображає у вигляді тексту всі повідомлення про помилки та інше. У правому нижньому куту вікна програми показується модель поточної плати і послідовний порт, до якого вона під'єднана. Кнопки на панелі інструментів призначені для створення, відкриття, збереження та прошивки програм у пристрій. Окрема кнопка запускає програму SerialMonitor [22].

На рисунку 3.4 зображений інтерфейс середовища розробки Arduino IDE.

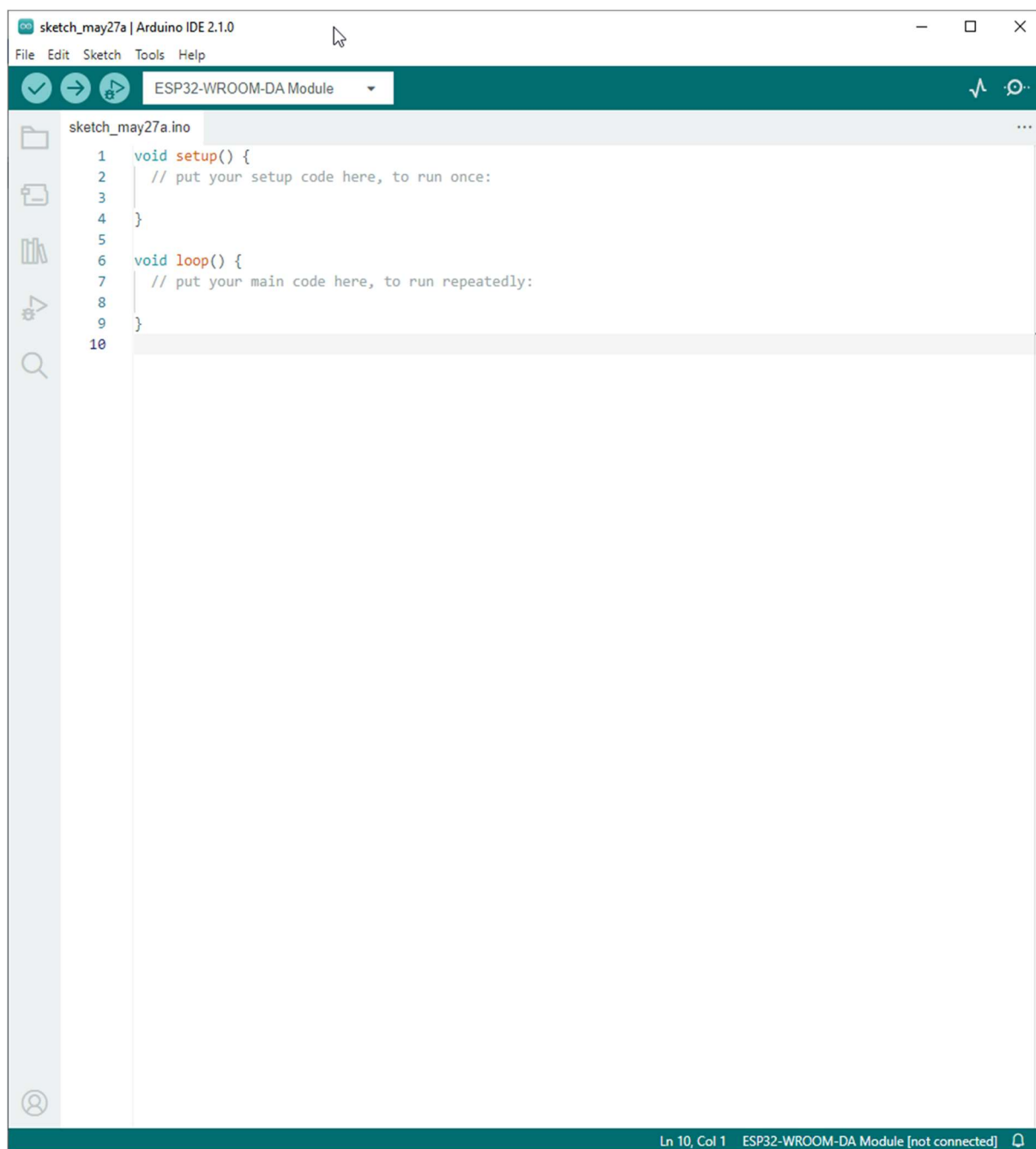


Рисунок 3.4 – Інтерфейс Arduino IDE

На інтерфейсі можна виділити такі основні елементи:

-  (Verify/Перевірити) кнопка для перевірки коду на помилки;
-  (Upload/Прошити) кнопка для компіляції і завантаження програми у мікроконтролер;

ESP32-WROOM-DA Module

Unknown
COM1

Select other board and port...

- поле вибору типу плати та компорту до якого буде підключена програма;
-  (SerialMonitor) програма для роботи з послідовним інтерфейсом;
- поле текстового редактору у якому пишеться код скетчу.

Перед тим, як прошивати програму в контролер, необхідно правильно вибрати плату та послідовний порт у меню Tools>Board та Tools>SerialPort відповідно. На Windows-системах, послідовний порт буде COM1 або COM2 (для пристроїв з послідовним інтерфейсом), або COM4, COM5, COM6 і вище (для Arduino з USB) - визначити необхідний номер порту можна у Диспетчері пристроїв, відшукавши рядок "USBserialdevice" у розділі Порти. Для ESPплатформи може знадобитись встановити драйвера для обробки даних ОС. Вони знаходяться за посиланням (<https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=overview>) [22].

Після вибору порту та плати, необхідно натиснути кнопку Upload на панелі інструментів або вибрати пункт Upload з меню File. Після цього розпочнеться процес завантаження програми на пам'ять контролера. У старих моделях (до Arduino Diecimila) функція авто-скидання відсутня, тому перед прошивкою таких пристроїв необхідно вручну натиснути кнопку скидання на платі. У процесі завантаження на більшості моделей Arduino блиматимуть світлодіоди RX та TX. Після завершення процесу прошивки програма видасть відповідне повідомлення або помилку [22].

Завантаження програми в Arduino здійснюється за допомогою завантажувача - невеликої програми, прошитої в пам'яті мікроконтролера, яка дозволяє завантажувати код без зовнішніх апаратних засобів. Завантажувач активізується на кілька секунд після скидання пристрою, після

чого він запускає на виконання останній завантажений у контролер скетч [22].

Для завантаження скетчу на платформу ESP може знадобитись завантаження скетчу у Arduino IDE шляхом затиснення кнопки BOOT на платі для переключення контролеру у режим перепрошивки.

Для налаштування середовища розробки під роботу з платформою ESP треба виконати кілька обов'язкових кроків:

Крок 1. Відкрити контекстне меню натиснувши на кнопку (File) у лівому верхньому куту і натиснути кнопку (Preferences...), продемонстровано на рисунку 3.5.

Крок 2. Вставити посилання на бібліотеку (https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json) у поле Additional boards manager як зображено на рисунку 3.6.

Крок 3. Відкрити контекстне меню натиснувши на кнопку (Tools), навести курсор на поле (Board) і натиснути на кнопку (Boards Manager...) як показано на рисунку 3.7.

Крок 4. Ввести у поле пошуку “ESP32” і натиснути на кнопку встановлення бібліотеки ESP32 від Espressif Systems як показано на рисунку 3.8.

Крок 5. Після цього слід підключити плату до комп'ютеру та виставити порт до якого треба завантажувати програмне забезпечення, показано на рисунку 3.9.

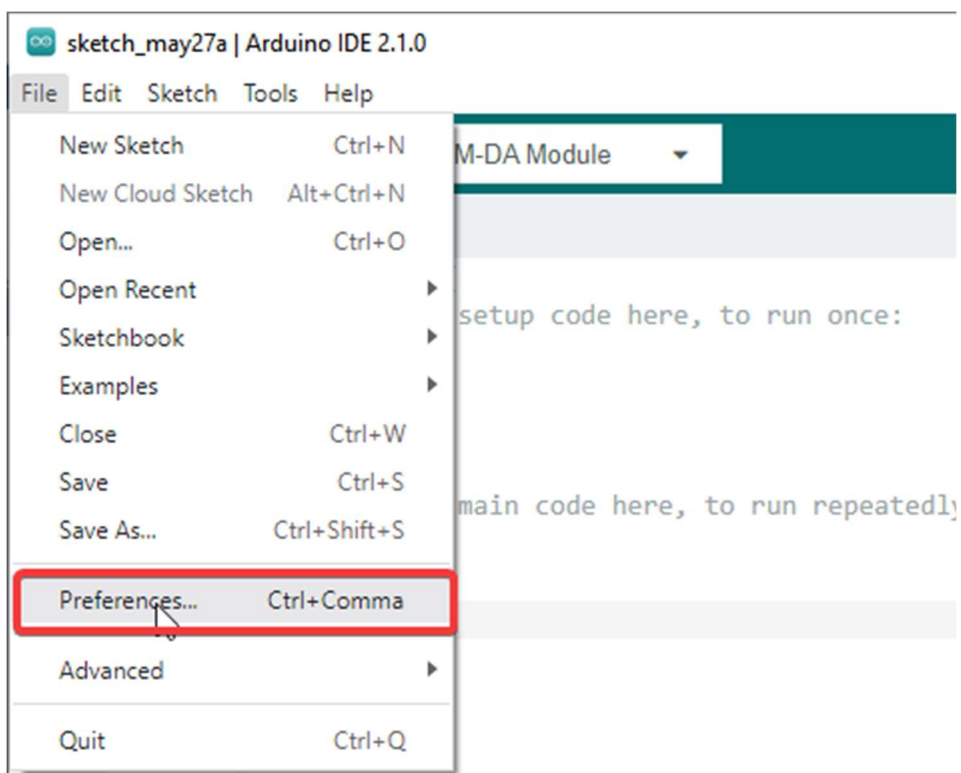


Рисунок 3.5– Контекстне меню файл

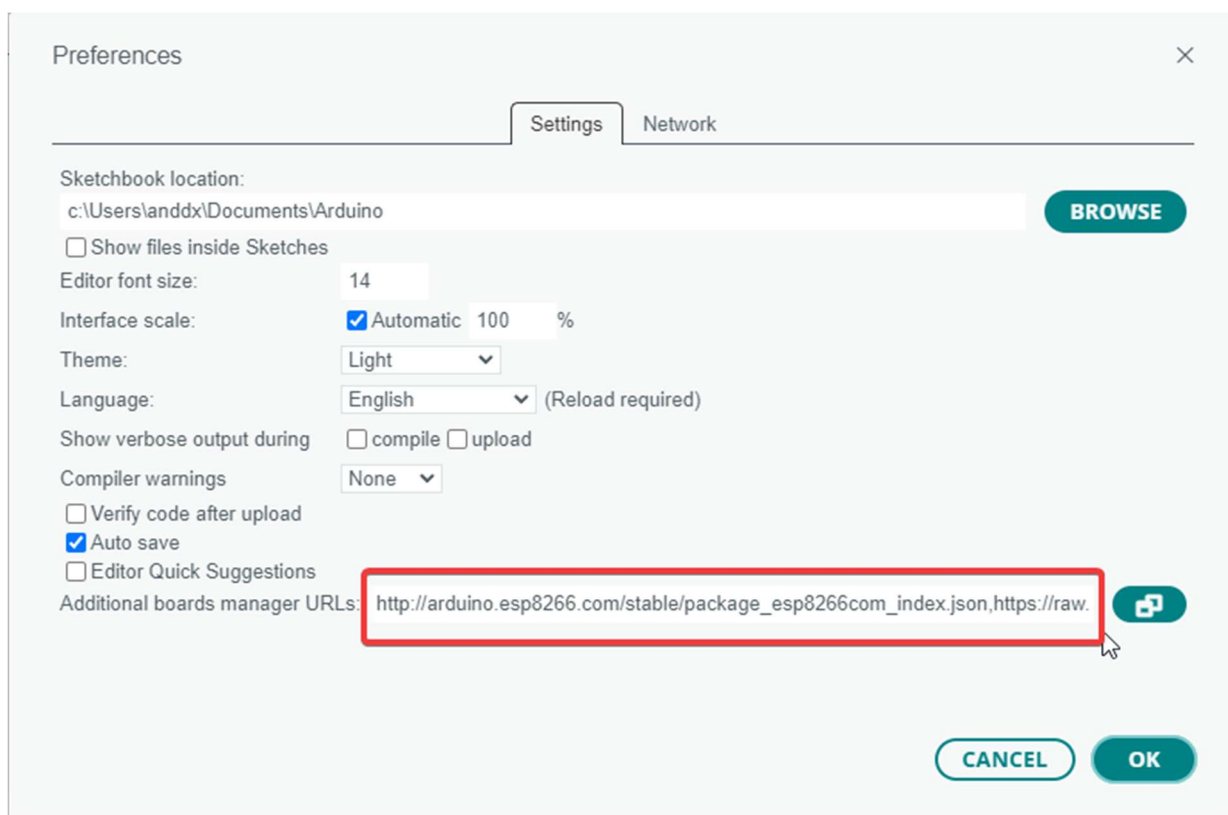
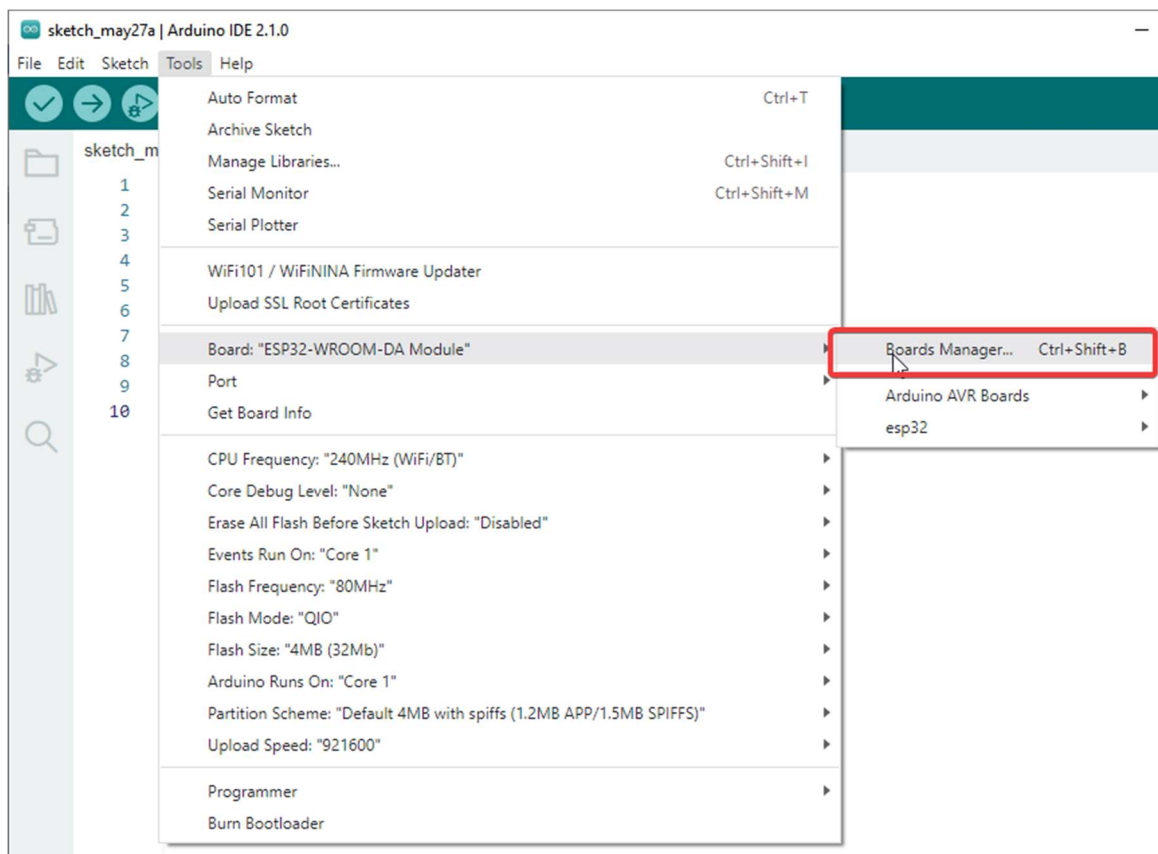


Рисунок 3.6 – Вкладка налаштування



Рисунку 3.7 – Менеджер плат

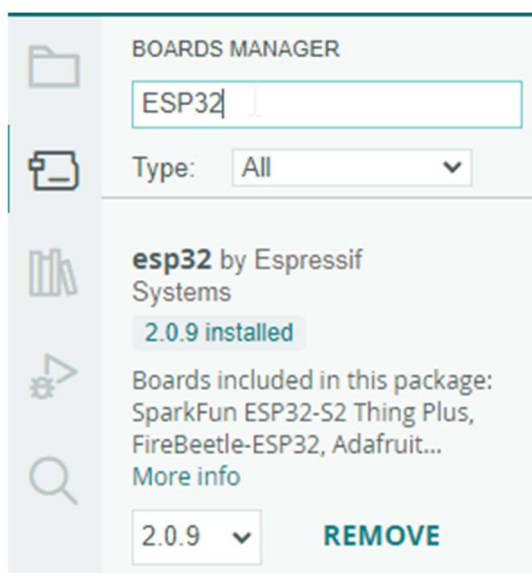


Рисунок 3.8 – Бібліотека плат



Рисунок 3.9 – COM-порт

Налаштування середовища розробки Arduino IDE для роботи з платою ESP32 завершено, можна приступати до розробки та завантаження скетчу.

3.3 Розробка програмної частини проекту (скетч)

Для початку розробки треба визначити структуру скетчу та механіку його роботи. Структура скетча має лише дві головні функції для того, щоб створити програму, яка буде працювати за принципом циклічного виконання:

- `setup()`. Функція виконується лише раз при старті програми і дозволяє задати початкові параметри. Тут зазвичай ініціюють порти вводу/виводу та назначають їм змінні [14];
- `loop()`. Функція виконується періодично, доки плата не буде вимкнена. Тут виконується основна робота програмного застосунку, викликаються функції чи виконуються логічні операції [14].

З самого початку скетча, перед функцією `setup()`, підключаються бібліотеки, які будуть задіяні. У випадку поточного проекту знадобиться

тільки бібліотека BLE Keyboard яка буде підключена командою `#include <BleKeyboard.h>`.

Глобальні змінні визначаються перед початком функції `setup`, у даному проекті їх знадобиться декілька у особливому форматі `unsigned long`, що представляє з себе формат що може зберігати тільки додатні числа від 0 до 4 294 967 295 та декілька констант що потрібні для ініціалізації портів плати. Константи буде для зручності названо нумерованими кнопками і присвоєно значення портів що були зазначені на Рисунку 3.3, тобто відповідно 19, 18, 5, 17 що дозволить при зміні контактів просто змінити значення констант, не шукаючи у головній функції які порти використані.

У функції `setup` буде визначено константи портів як `INPUT_PULLUP`. Мікроконтроллер ESP32 має внутрішні підтягуючі резистори (резистори, підключені до живлення всередині мікросхеми), якими можна керувати. Їх можна використовувати замість зовнішніх резисторів, підключених до землі за допомогою параметру `INPUT_PULLUP` у функції `pinMode()`. Це дозволить інвертувати поведінку підключеного до виведення зовнішнього датчика: `HIGH` значить його відключення, а `LOW` – включення [19].

Керувати підключеною бібліотекою BLE Keyboard дуже просто, керівництво по застосуванню можна взяти на сайті бібліотеки Arduino Keyboard яка являється прототипом бібліотеки для Bluetooth клавіатури за посиланням <https://www.arduino.cc/reference/en/language/functions/usb/keyboard/> за посиланням

Основні команди бібліотеки являють собою:

- `press()` яка емулює затиснення кнопки у продовж часу до виклику команди `release()`;
- `write()` що представляє собою аналог команд `press()` але емулює одноразове натиснення зазначеної клавіши;
- `print()` команда вводить текст, який може бути прописаний у параметрах функції у вигляді рядку у лапках, наприклад `print("helloworld!")`,

існує також підтип цієї команди, команда `println()` що одразу після введення тексту натискає клавішу ENTER, тобто переносить рядок.

Слід зазначити що при використанні функцій натискання клавіш слід використовувати код клавіш, що також приведений у керівництві бібліотеки за посиланням [23] аніж назви клавіш, так як клавіатури різних розкладок, таких як ноутбуків різних виробників, можуть бути не так зчитані, що може доставити проблеми у проєктуванні програм.

У проєкті буде використано клавіші, приведені у таблиці 3.1, а також їх код та опис, який зазначений розробниками бібліотеки за посиланням вище.

Таблиця 3.1 – Приклад використаних клавіш

Клавіша	Назва	Десяткове значення
Лівий альт	KEY_LEFT_ALT	130
Знак операційної системи на клавіатурі віндовс та ⌘ на Мак	KEY_LEFT_GUI	131
Лівий шифт	KEY_LEFT_SHIFT	129
Ескейп	KEY_ESC	177
Лівий контрол	KEY_LEFT_CTRL	128
Ентер	KEY_RETURN	176
Кнопка медіа	KEY_MEDIA_PLAY_PAUSE	208
Кнопка скріншоту	KEY_PRINT_SCREEN	206

Слід зауважити що мікропроцесор працює дуже швидко, швидкість виконання головної функції `loop()` може доходити до тисячі разів за секунду, тому дуже важливо використовувати ресурси пристрою більш обережно,

адже при помилці у проектуванні алгоритму програми можна змусити відправляти тисячу команд в секунду до під'єданого пристрою і не мати змоги вибратись із цього циклу, якщо переміщати за допомогою програми курсором мишки або, наприклад викликати диспетчер задач.

Окрім обережного проектування алгоритму слід розуміти як працюють функції часу у Arduino, таких функцій існує дві: `delay()` та `millis()`.

Функція `delay()` просто зупиняє виконання програми на проміжок, указаний у аргументах функції при виклику у мілісекундах.

Крім цього, у багатьох програмах функція затримки використовується для таких завдань, як обробка брязкання контактів та інше. Незважаючи на це, використання функції `delay()` у коді програми має істотні недоліки. У процесі дії `delay()` такі операції, як зчитування даних датчиків, математичні обчислення чи операції з виводами неможливо виконувати. Фактично, функція `delay()` призводить до зупинки практично всіх операцій [24].

Досвідчені програмісти зазвичай не використовують функцію `delay()` для виставлення інтервалів (за винятком найпростіших скетчів Arduino).

Функція `millis()` при виклику повертає кількість мілісекунд, які пройшли з початку роботи програми у форматі `unsigned long`, через обмеження платформи у виділені пам'яті у типах даних фактично програма не може працювати більше 49.7 діб. Ця функція позбавлена недоліків функції `delay()` та при правильному використанні може дуже допомогти у створенні затримок та пауз між виконанням ділянок коду. Так, для цього у самому початку програми було створено декілька змінних формату `unsigned long` для використання їх у логічних операціях з функцією `millis()`.

Для кожної кнопки було розроблено частини програми з логікою функцій часу для забезпечення адекватної швидкості зчитування сигналу, це в основному досягається шляхом використання простої логічної конструкції:

```
if(millis() – минулий час > таймер){
    минулий час = millis();
```

Це запобігає як брязкотанню кнопки так і дає час комп'ютеру

виконати задачу, а користувачу здійснити маніпуляції обробки інформації з швидкістю людини. Адже для людини 100 мілісекунд це мить а для працюючого мікропроцесора на максимальній швидкості ціла вічність. А також дає можливість обробки всіх частин програми паралельно, а не зупиняється, як у випадку використання функції `delay()`.

На рисунку 3.10 показана запрограмована схема переключення режимів макросів.

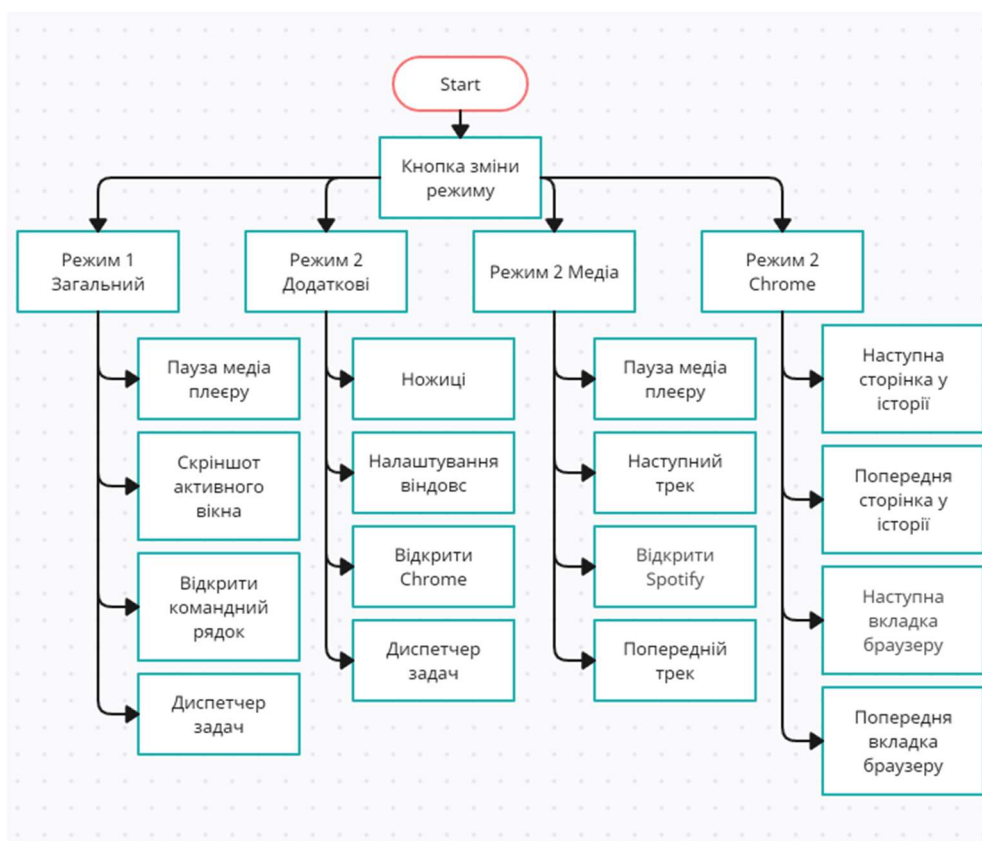


Рисунок 3.10 – Блок-схема режимів макросів

На рисунку 3.11 представлений приклад алгоритму кнопки 3 у першому режимі у вигляді блок-схеми.

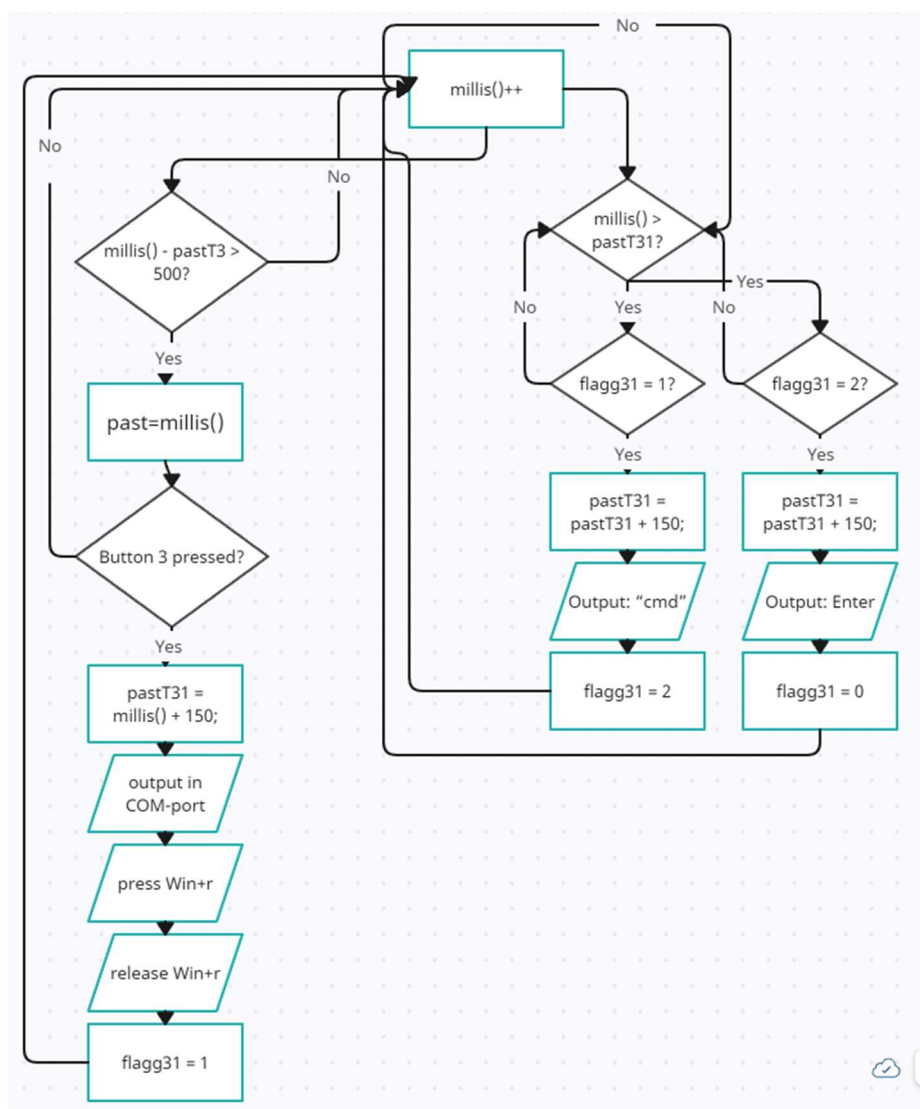


Рисунок 3.11 – Блок-схема алгоритму роботи кнопки 3 у першому режимі

3.4 Висновки за розділом 3

У цьому розділі було освітлено проектування та реалізація як апаратної так і програмної частин проекту, досліджені проблеми з якими можна зіштовхнутися при проектуванні та потенціальні рішення що були та можуть бути використані для проектування інших проектів на цій платформі.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОЕКТУ

4.1 Призначення та умови застосування проекту

Пристрій призначений для застосування програмованих макросів на пристроях платформи Windows з функцією Bluetooth. Пристрій заснований на мікроконтролері ESP32 на платі розробника. Програма написана на мові програмування Arduinoяка заснована на C++ з додаванням деяких бібліотек, таких як AVR Libc [15].

Пристрій виконує функції підключення до пристроїв по технології бездротового з'єднання Bluetooth, виконання програмного забезпечення, завантаженого з комп'ютера.

Програмний застосунок забезпечує виконання 16тьох програмованих макросів згідно алгоритму та переключення розкладок макросів.

Опис програми та текст програми скетчу наведені у додатках Б та В.

4.2 Інструкція користувача по експлуатації

Для використання пристрою достатньо підключити його до живлення через дріт microUSB від роз'єму комп'ютера, павербанку, акумулятора або інших джерел живлення від 3.3 до 12 Вольт максимум на протязі 50 діб та під'єднати через бездротове з'єднання по технології Bluetooth до пристрою, на котрому потрібне виконання макросів.

Встановлено 5 кнопок керування пристроєм. Одна, від'єднана від інших це перемикач розкладки макросів і чотири кнопки на дротовій клавіатурі виконання макросів.

Користувачу дозволено натискати кнопки не частіше одного разу в 200 мілісекунд.

Макроси запрограмовані враховуючі розкладку пристроїв на платформі Windows 10 та 11.

Список запрограмованих макросів:

а) режим1:

- 1) Media Play/Pause;
- 2) Alt+PrintScreen (ScreenShot);
- 3) Windows+R -> cmd (Open Comand console);
- 4) Control+Shift+Escape(Open Task menager);

б) режим 2:

- 1) Windows+Shift+S(Open Scissors);
- 2) Open the Windows Settings;
- 3) Open Google Chrome;
- 4) Write time from start;

в) режим 3. Медіа:

- 1) Media Play/Pause;
- 2) Next track;
- 3) Open music;
- 4) Previous track;

г) режим 4 Google Chrome:

- 1) Next page in history;
- 2) Previous page in history;
- 3) Next open page in browser;
- 4) Previous open page in browser.

4.3 Тестування продукту та результати роботи пристрою

Тестування та робота пристрою перевірялась на планшеті з ОС Windows 11, підключення до пристрою знаходиться у налаштуваннях та виглядає як показано на рисунку 4.1.

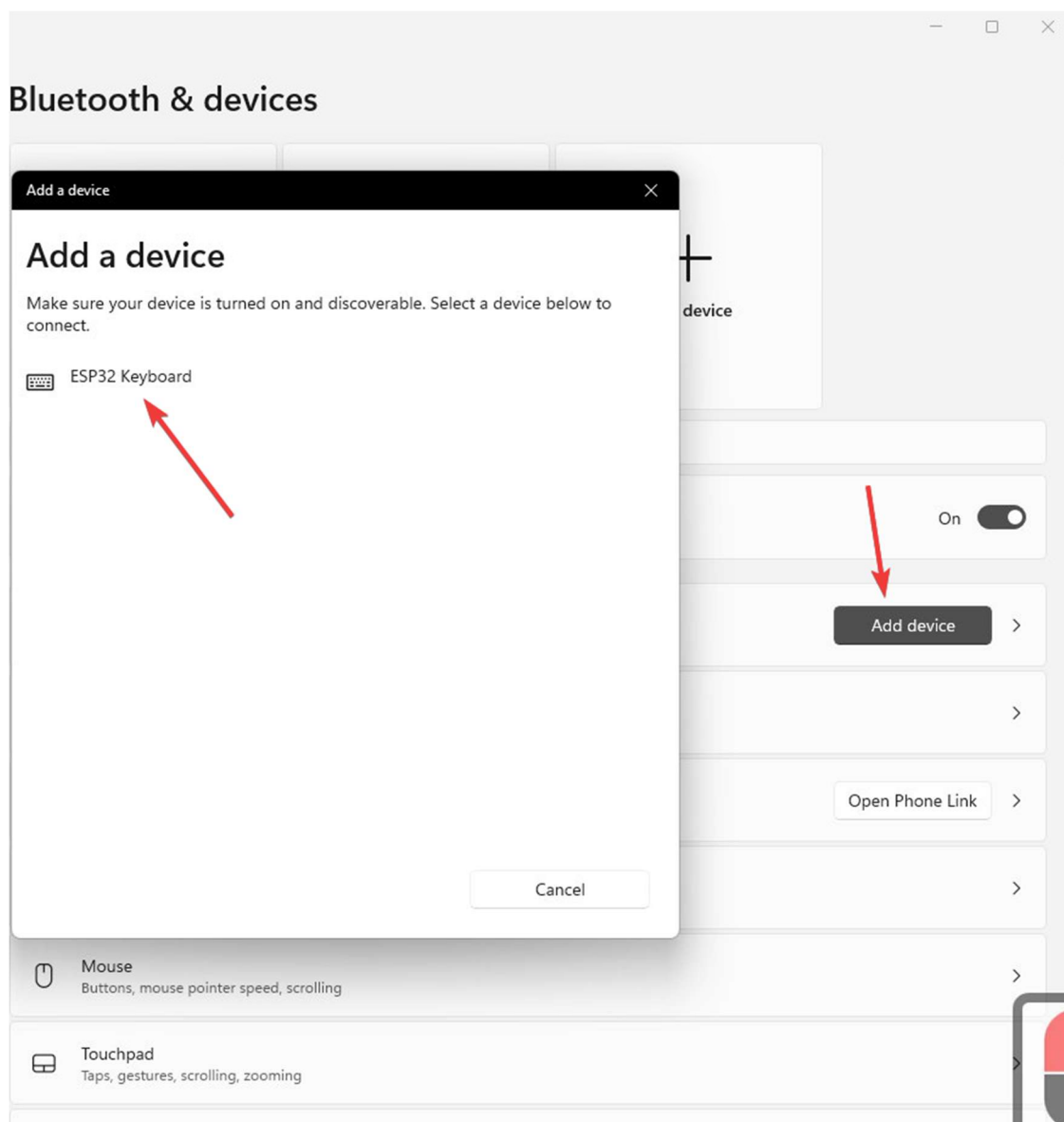


Рисунок 4.1 – Підключення пристрою до комп'ютеру

Тестування та відлагодження натиснених клавіш на пристрої були здійснені за допомогою програми під назвою Carnas, яка є у вільному доступі за посиланням [25].

На рисунку 4.2 можна побачити відображення програмою Carnas натиснутих клавіш у правому нижньому куту екрану за допомогою пристрою ESP четвертої кнопки у першому режимі.

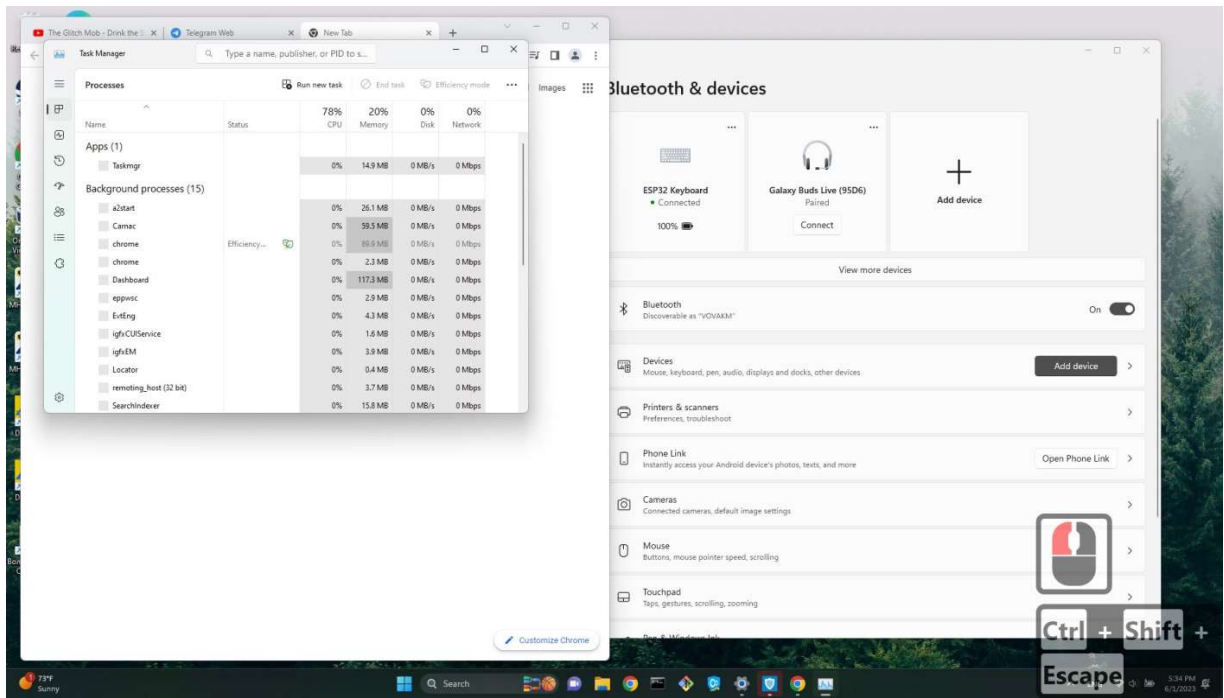


Рисунок 4.2 – Відображення вводу за допомогою програми Carnas

На рисунку 4.3 продемонстрована натиснення третьої кнопки у першій розкладці, це активує макрос, який складається з комбінації клавіш Windows+r, затримки у 150 мілісекунд, вводу тексту «cmd» і натисненню клавіші Enter після ще одної 150 мілісекундної затримки.

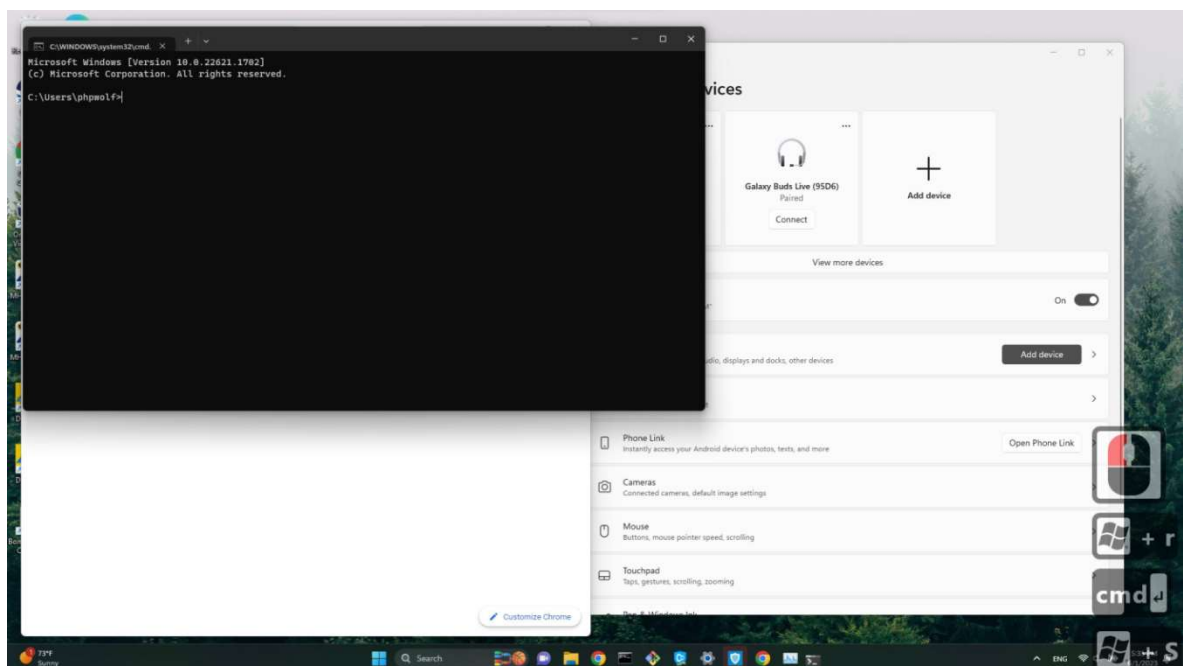


Рисунок 4.3 – Відображення роботи пристрою

На рисунку 4.4 відображено роботу четвертої кнопки другої розкладки пристрою, що виводить текстом час, з початку роботи пристрою. Цю функцію було виконано використовуючи функцію `millis()`, описану у розділі 3.3 і математичні операції остатку ділення отриманих мілісекунд.

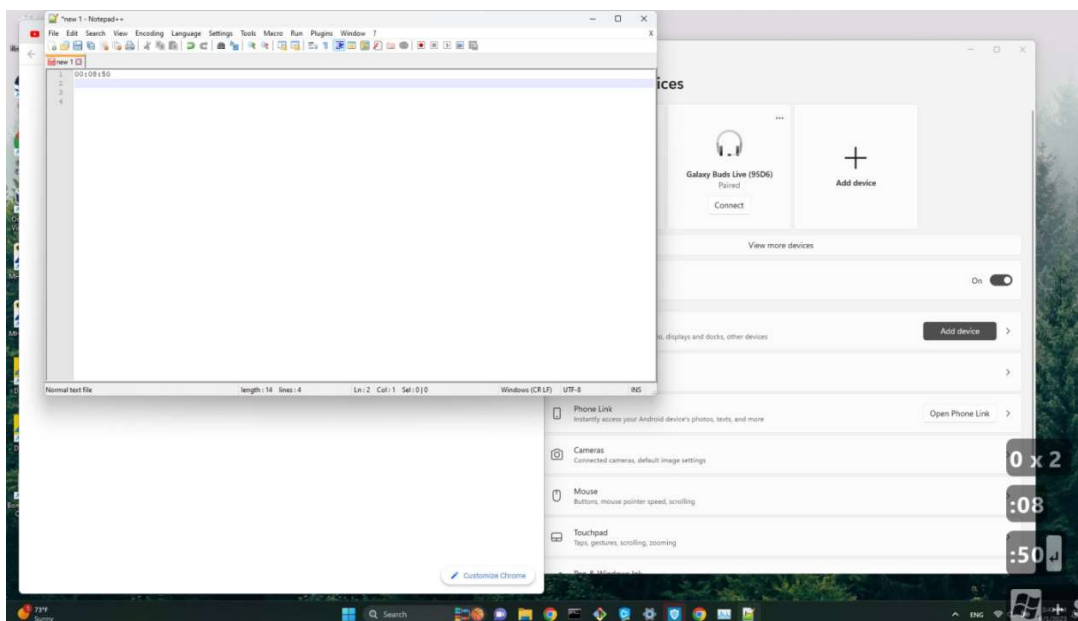
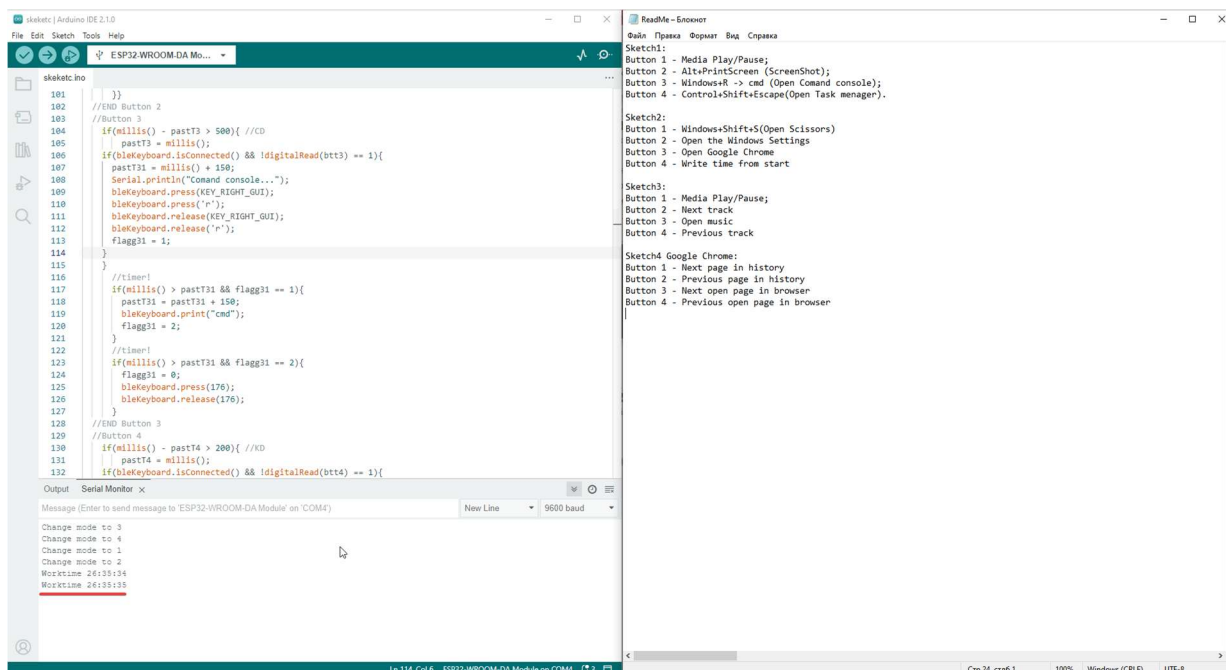


Рисунок 4.4 – Робота виведення часу роботи пристрою

На рисунку 4.5 продемонстрована відображення вивід виконання задач пристроєм у послідовний СОМпорт середовища розробки та документу у якому записані всі розкладки запрограмованих макросів. Як можна побачити у СОМпорті відображено роботу кнопки перемикавання розкладок та протестована стабільність довгострокової роботи пристрою протягом двадцяти шести годин.



Рисунку 4.5 – Демонстрація роботи COM порта середовища розробки

На рисунку 4.6 зображене завантаження скомпільованої програми у пам'ять мікропроцесору ESP за допомогою CP Arduino IDE у вкладці «Output».

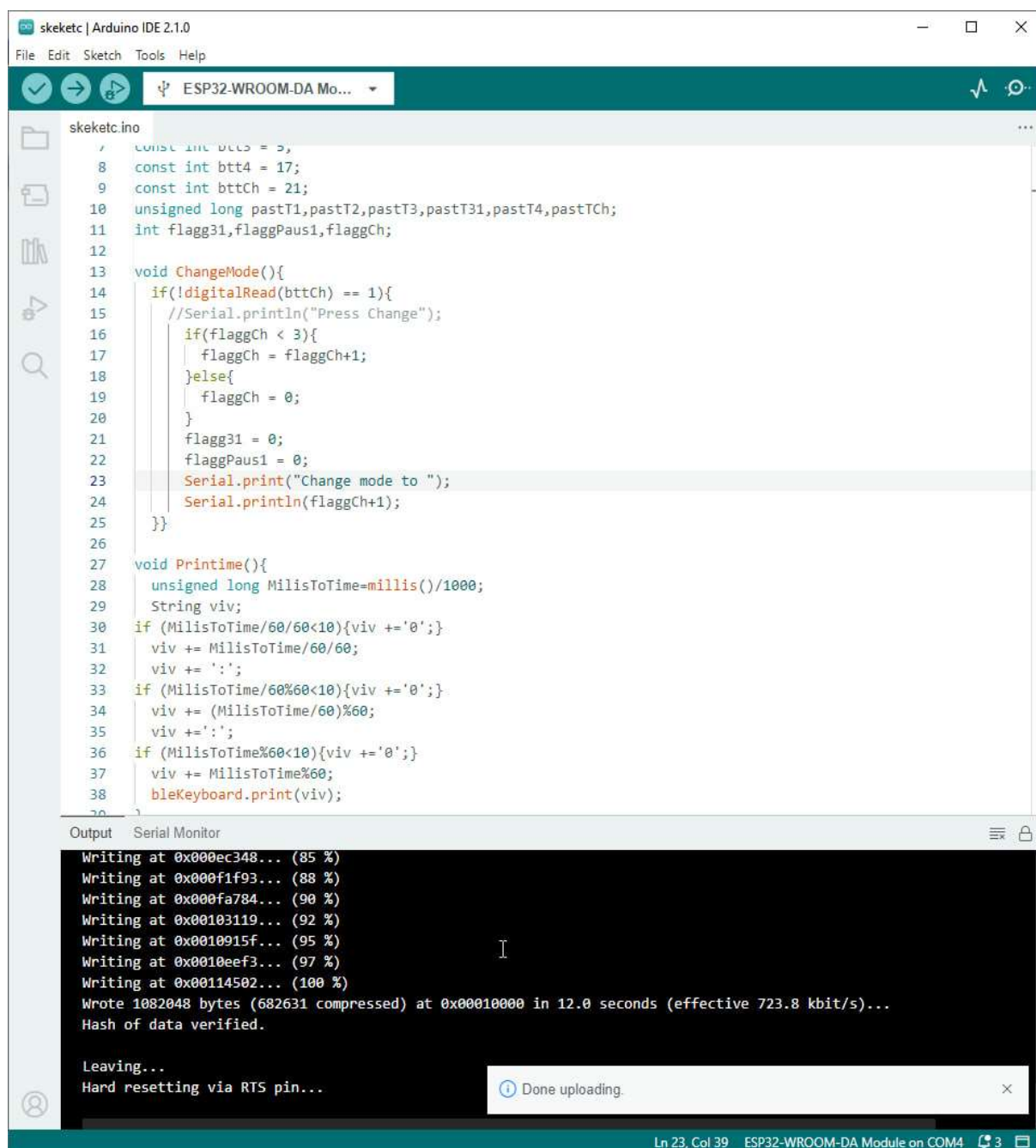


Рисунок 4.6 – Завантаження прошивки на пристрій

На рисунку 4.7 показаний пристрій під час роботи, підключений до комп'ютеру.

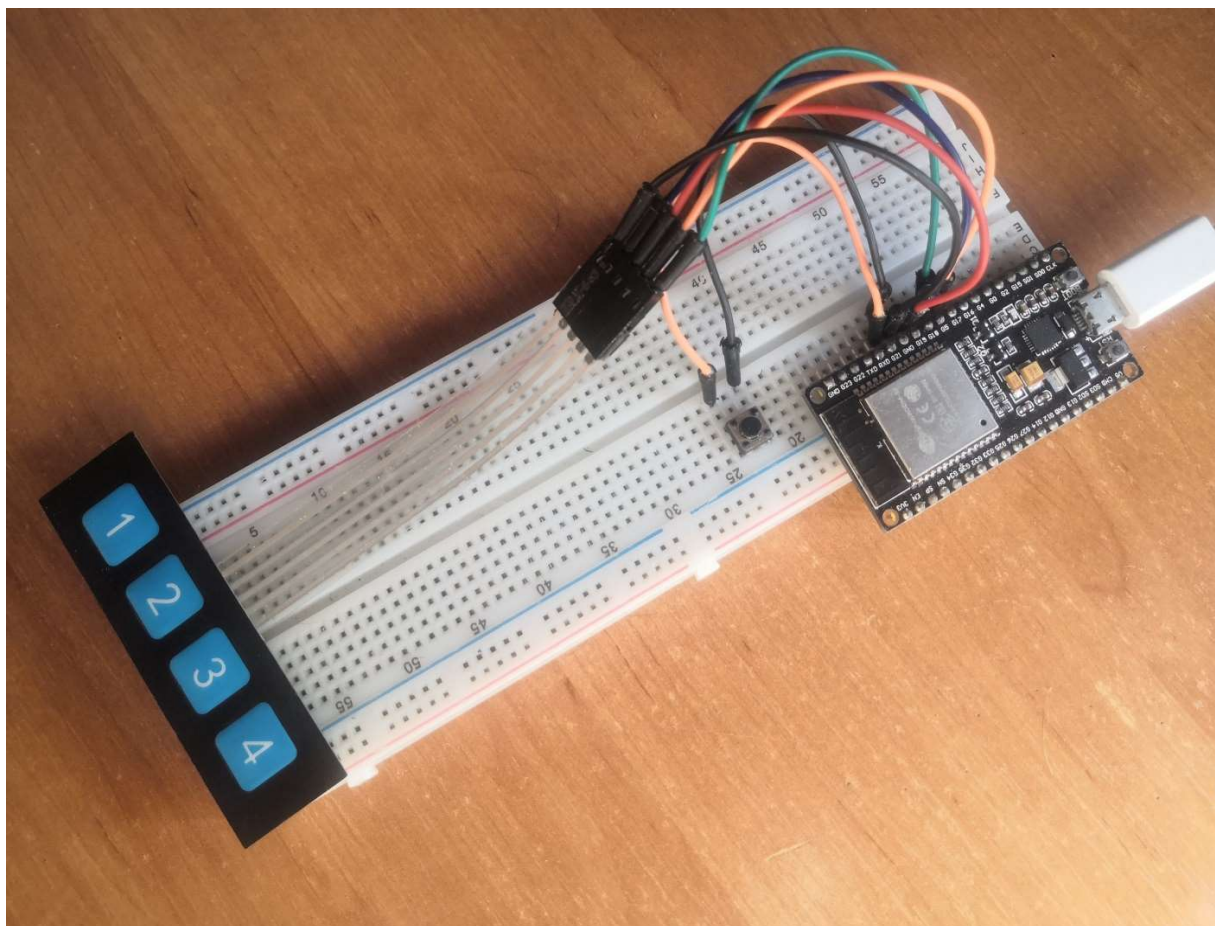


Рисунок 4.7 – Працюючий пристрій

4.4 Висновки за розділом 4

У цьому розділі описана робота пристрою у практичних умовах. Проведено тестування за різними сценаріями, які підтвердили працездатність, стабільність роботи пристрою та виконання умов технічного завдання.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було створено пристрій і програмне забезпечення для нього. Для досягнення поставленої мети було проведено кілька основних кроків реалізації. На початку було проведено аналіз та обґрунтування можливих рішень апаратної частини, дослідження середовища розробки, мови програмування та основних її функцій.

Було проведено дослідження актуальності геймінгу і схожих пристроїв у сучасних тенденціях.

Визначено що доцільно вивчити технології та створити пристрій для виконання поставлених задач самостійно, через невелику складність та дешевизну реалізації.

Основою для створення пристрою було вирішено обрати мікропроцесор ESP32 на платі розробника, яка відрізняється від інших рішень вбудованими технологіями Wi-Fi та Bluetooth а також порівняно низькою ціною за відповідні можливості.

Для створення програмного забезпечення було обрано середовища розробки Arduino IDE, досліджені її функціонал та мова програмування.

Розроблено програмне забезпечення що дозволяє виконувати певні запрограмовані макроси, забезпечення ефективно використовує ресурси плати.

Описано програмну частину пристрою і проведено тестування створеної програми, що показало здатність нормально виконувати макроси і не перевантажувати систему непотрібними сигналами, стабільно працювати впродовж п'ятдесяти діб роботи.

Реалізовано пристрій, що живиться від різних джерел живлення, підключається до комп'ютерів через технологію Bluetooth та адекватно реагує на введений сигнал від користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Інновації в онлайн-іграх: еволюція геймінгу [Electronic resource]. – Access <https://senior.ua/articles/nnovac-v-onlaynngrah-evolyucya-geymngu>.
2. Кіберспорт стає професією [Electronic resource]. – Access <https://www.radiosvoboda.org/a/25089223.html>.
3. Payoneer about live-striming [Electronic resource]. – Access <https://blog.payoneer.com/ua/freelancers-ua/industry-tips-fl-ua/global-live-streaming-in-2020>.
4. Programming Macro Custom Keyboard [Electronic resource]. – Access <https://www.aliexpress.com/item/1005004127594069.html>.
5. Oscar a4tech [Electronic resource]. – Access <http://www.x7.cn/oscaren/>.
6. Bloody a4tech [Electronic resource]. – Access <https://drivermaster.ru/?p=27031>.
7. Arduino Micro [Electronic resource]. – Access <https://doc.arduino.ua/ru/hardware/>.
8. ESP32 vs ESP8266 [Electronic resource]. – Access <https://makeradvisor.com/esp32-vs-esp8266/>.
9. Arduino IDE [Electronic resource]. – Access https://ru.wikipedia.org/wiki/Arduino_IDE.
10. Freertos [Electronic resource]. – Access <https://aws.amazon.com/ru/freertos/>.
11. ESP32 IDF [Electronic resource]. – Access <https://kotyara12.ru/iot/esp32idf>.
12. Atmel Studio [Electronic resource]. – Access <https://radio-blogs.ru/blog/programmy/atmel-studio-7>.
13. На роздоріжжі Arduino, C або Ассемблер [Electronic resource]. – Access <https://habr.com/ru/articles/547752/>.

14. Arduino Platform [Electronic resource]. – Access <https://uk.wikipedia.org/wiki/Arduino>.
15. AVR Libc [Electronic resource]. – Access <https://www.nongnu.org/avr-libc/user-manual/index.html>.
16. Порівняння мов [Electronic resource]. – Access <https://mcucpu.ru/index.php/soft/42-lmcu/67-programmlang>.
17. Arduino IDE Або C/C++ [Electronic resource]. – Access <https://avr-cpp.blogspot.com/2019/02/arduino-ide-cc.html>.
18. ESP32 devkit v1 pinout scheme [Electronic resource]. – Access <https://www.upesy.com/blogs/tutorials/esp32-pinout-reference-gpio-pins-ultimate-guide/>.
19. INPUT_PULLUP [Electronic resource]. – Access <https://doc.arduino.ua/ru/prog/Constants>.
20. Matrickeyboard 1x4 picture [Електрон. ресурс]. – Режим доступу: <https://prom.ua/p534366052-matrichnaya-klaviatura-1x4.html>.
21. Keypad 1x4 [Electronic resource]. – Access <https://onlinesrs.co/product/1x4-4-key-matrix-membrane-switch-keypad-control-panel-scm-extended-keyboard-super-slim-controller/>.
22. Arduino IDE [Electronic resource]. – Access <https://doc.arduino.ua/ru/guide/Environment>.
23. Keyboard library documentation [Electronic resource]. – Access <https://www.arduino.cc/reference/en/language/functions/usb/keyboard/>.
24. Delay [Electronic resource]. – Access <https://doc.arduino.ua/ru/prog/Delay>.
25. Carnac utility [Electronic resource]. – Access <https://github.com/Code52/carnac>.
26. Keypad search [Electronic resource]. – Access https://www.ebay.com/sch/i.html?_from=R40&_nkw=keypad&_sacat=0&_dmd=2&rt=nc.

27. Arduino IDE ESP32 instructions [Electronic resource]. – Access <https://randomnerdtutorials.com/installing-esp32-arduino-ide-2-0/>.

ДОДАТОК А
Технічне завдання

Вступ

Проект – створення пристрою на основі платформи мікропроцесорів Arduino і схожих що дозволить виконувати програмовані макроси і передавати сигнали до комп'ютеру як пристрій вводу. За допомогою цього проекту можна покращити життя просунутих користувачів ПК та стримерів.

A.1 Підстави для створення

Підставою для створення пристрою є завдання на дипломну кваліфікаційну роботу на тему «Розробка застосунку для використання програмованих макросів на платформі Arduino», затверджене наказом Національного університету «Запорізька політехніка» № 87 від 4 квітня 2023 р.

A.2 Призначення створення

Пристрій повинен служити макетом для просунутих користувачів ПК, котрим може знадобитись поліпшення виконання монотонних задач введення даних з клавіатури, на основі зробленого застосунку можна створити майже будь-який сценарій використання клавіатури.

Розробка повинна бути максимально економічною та зрозумілою для користувачів що мають ціль створити проект на основі неї.

У папці проекту знаходиться файл з кодом та текстовий документ з списком реалізованих макросів.

A.3 Основні вимоги до пристрою

A.3.1 Вимоги до функціональних характеристик

Пристрій повинен під'єднуватись до необхідного комп'ютеру через Bluetooth та виконувати задані програмою макроси, перемикає розкладки макросів. Прописані макроси повинні:

- емулювати натиснення комбінацій декількох клавіш;
- викликати програми;
- керувати медіа;
- мати затримки для адекватного виконання дій пристроєм;
- виводити час роботи пристрою по команді.

A.3.2 Вимоги до програмної частини

Код програми повинен бути структурованим та мати помітки для облегшення читання. Програма повинна обчислювати та мати затримки для керування кількістю команд вводу, повинна мати перевірки на дійсність команд, які надходять від кнопок.

A.3.3 Вимоги до надійності

Пристрій повинен коректно реагувати на натиснення кнопок керування, працювати при подачі живлення та підключатись до пристроїв за допомогою бездротового з'єднання.

A.3.4 Умови експлуатації

Пристрій повинен працювати при підключенні його до живлення через дріт microUSB від роз'єму комп'ютера, павербанку, акумулятора або

інших джерел живлення від 3.3 до 12 Вольт максимум на протязі 50 діб. Під'єднуватись до пристрою користувача через бездротове з'єднання по технології Bluetooth.

Пристрій повинен мати 5 кнопок керування. Одна, від'єднана від інших як перемикач розкладки макросів і чотири кнопки на дротовій клавіатурі для здійснення макросів.

Пристрій повинен давати користувачу натискати кнопки не частіше 200 мілісекунд, бути здатним працювати враховуючі розкладку тільки на платформах Windows 10 та 11.

A.3.5 Вимоги до складу та параметрів технічних засобів

Для користуванням пристроєм користувач повинен мати ПК або ноутбук з операційною системою Windows 10 чи 11, з такими мінімальними характеристиками та зовнішніми пристроями:

- процесор 1 ГГц та вище;
- вільна оперативна пам'ять 500 Мб та більше;
- підтримкою технології Bluetooth;
- монітор, маніпулятор «миша».

A.3.6 Стадії та етапи розробки

На першому етапі необхідно дослідити предметну область та представити основну ідею майбутнього пристрою.

Визначити мету та цільову аудиторію користувачів.

З'ясувати можливості цільової аудиторії у відтворенні проекту та розповсюдженість подібних пристроїв.

Визначитись з апаратною частиною пристрою і обґрунтувати вибір середовища можливих конкурентів. Скласти технічне завдання до проекту та почати проектування.

Створити пристрій з відповідними характеристиками до ТЗ.

Провести вибір та дослідження середовища розробки та мови програмування.

Створити програмне забезпечення для виконання технічного завдання та протестувати його працездатність, усовуючи проблеми.

Оформити документацію з реалізованими функціями.

Після цього проєкт повинен бути готовий до роботи.

ДОДАТОК Б**Опис проєкту**

Б.1 Загальні відомості

Даний проект це пристрій що полегшить користування комп'ютером методом відтворення програмованих макросів на платформі мікропроцесора ESP32. Створений пристрій дозволить швидше виконувати частіше за все використовувані команди з пристроєм введення типу «клавіатура».

Для просунутих користувачів, які можуть собі дозволити розібратись в доволі простому коді і які часто працюють на ПК, може значно допомогти такий тип пристрою, що дозволить виконувати від простих команд таких як змінення мови через натиснення клавіш Alt+Shift до складних макросів чіткого натиснення певних клавіш у певний момент часу.

Б.2 Функціональне призначення

Пристрій повинен підключатись до необхідного пристрою через блютуз та виконувати задані програмою макроси, перемикає розкладки макросів. Прописані макроси повинні:

- емулювати натиснення комбінацій декількох клавіш;
- викликати програми;
- керувати медіа;
- мати затримки для адекватного виконання дій пристроєм;
- виводити час роботи пристрою по команді.

Б.3 Опис логічної структури

На рисунку Б.1 зазначена схема підключення контактів плати до клавіатури у форматі G(контакт на платі) – номер кнопки.

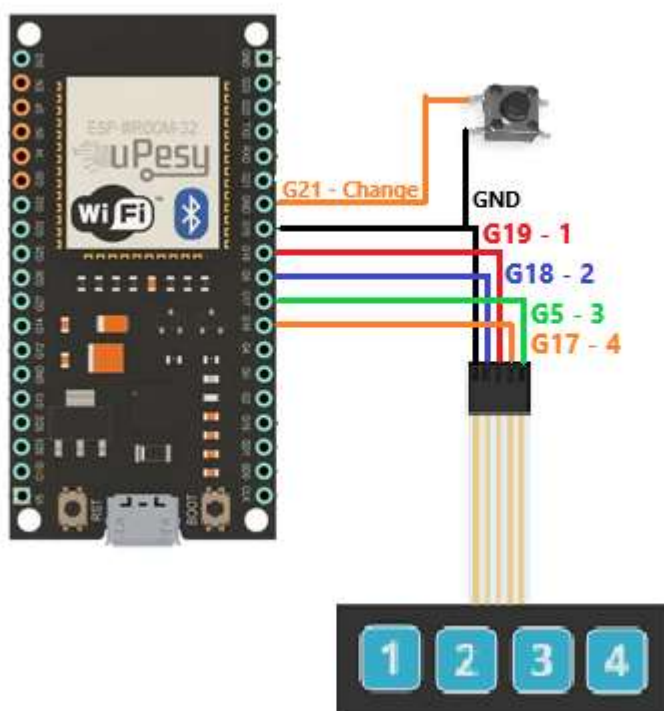


Рисунок Б.1 – Схема під'єднання контактів плати до клавіатури

Через те, що плата має внутрішні резистори, підключені до живлення всередині мікросхеми, якими можна керувати, підключення контактів визначається типом `INPUT_PULLUP` в програмній частині, тож до кнопок постійно йде струм з плати. Отже сигнал відсутності струму на контакті кнопки буде означати для плати що її натиснено.

Після натиснення кнопки плата обробляє сигнал та виконує логічні операції що при викликанні функції повертають значення 1 або 0 (TRUE/FALSE).

Виконувати функції кнопок плата буде лише в тому випадку, якщо вона підключена до пристрою через Bluetooth.

Б.4 Використані технічні засоби

Для користування пристроєм користувач повинен мати ПК або ноутбук з операційною системою Windows 10 чи 11, з такими мінімальними характеристиками та зовнішніми пристроями:

- процесор 1 ГГц та вище;
- вільна оперативна пам'ять 200 Мб та більше;
- підтримкою технології Bluetooth;
- монітор, маніпулятор «миша».

Б.5 Виклик та завантаження

Для початку роботи, пристрій слід підключити до живлення через дрот micro USB від розйому комп'ютера, павербанку, акумулятора або інших джерел живлення від 3.3 до 12 Вольт. Під'єднатись до потрібного пристрою через бездротове з'єднання по технології Bluetooth. Пристрій буде відображений під назвою ESP Keyboard.

Б.6 Вхідні та вихідні дані

Пристрій при роботі не має вхідних даних окрім сигналів від кнопок на самому пристрої.

ДОДАТОК В
Текст програми

```
#include <BleKeyboard.h>
```

```
BleKeyboard bleKeyboard;
```

```
const int btt1 = 19;
```

```
const int btt2 = 18;
```

```
const int btt3 = 5;
```

```
const int btt4 = 17;
```

```
const int bttCh = 21;
```

```
unsigned long pastT1,pastT2,pastT3,pastT31,pastT4,pastTCh;
```

```
int flagg31,flaggPaus1,flaggCh;
```

```
void ChangeMode(){
```

```
  if(!digitalRead(bttCh) == 1){
```

```
    //Serial.println("Press Change");
```

```
    if(flaggCh < 3){
```

```
      flaggCh = flaggCh+1;
```

```
    }else{
```

```
      flaggCh = 0;
```

```
    }
```

```
    flagg31 = 0;
```

```
    flaggPaus1 = 0;
```

```
    Serial.print("Change mode to ");
```

```
    Serial.println(flaggCh+1);
```

```
  }}
```

```
void Printime(){
```

```
  unsigned long MilisToTime=millis()/1000;
```

```
  String viv;
```

```
  if (MilisToTime/60/60<10){viv += '0';}
```

```

    viv += MilisToTime/60/60;
    viv += ':';
    if (MilisToTime/60%60<10){viv += '0';}
    viv += (MilisToTime/60)%60;
    viv += ':';
    if (MilisToTime%60<10){viv += '0';}
    viv += MilisToTime%60;
    bleKeyboard.print(viv);
}

void PrintimetoCOM() {
    unsigned long MilisToTime=millis()/1000;
    if (MilisToTime/60/60<10){ Serial.print("0");}
    Serial.print(MilisToTime/60/60);
    Serial.print(":");
    if (MilisToTime/60%60<10){ Serial.print("0"); }
    Serial.print((MilisToTime/60)%60);
    Serial.print(":");
    if (MilisToTime%60<10){ Serial.print("0"); }
    Serial.println(MilisToTime%60);
}

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

    pinMode(btt1, INPUT_PULLUP);
    pinMode(btt2, INPUT_PULLUP);
    pinMode(btt3, INPUT_PULLUP);
    pinMode(btt4, INPUT_PULLUP);
    pinMode(bttCh, INPUT_PULLUP);

```

```

Serial.println("Starting BLE work!");

bleKeyboard.begin();
}

void loop() {
//Button Change
if(millis() - pastTCh > 300){ //CD
    pastTCh = millis();
    ChangeMode();
}
//END Change
//////////Mode 1//////////
if(flaggCh == 0){
//Button 1
if(millis() - pastT1 > 200){ //CD
    pastT1 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt1) == 1){
        Serial.println("Press Play");
        bleKeyboard.press(KEY_MEDIA_PLAY_PAUSE);
        flaggPaus1 = 1;
    }
}
//Release
if(digitalRead(btt1) == 1 && flaggPaus1 == 1){
    flaggPaus1 = 0;
    bleKeyboard.release(KEY_MEDIA_PLAY_PAUSE);
    Serial.println("Release Play");
}
//END Button 1

```

```

//Button 2
if(millis() - pastT2 > 500){ //CD
    pastT2 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt2) == 1){
        Serial.println("Press Alt+PrtSc");
        bleKeyboard.press(KEY_LEFT_ALT);
        bleKeyboard.press(206);
        //Release
        bleKeyboard.release(KEY_LEFT_ALT);
        bleKeyboard.release(206);
    }
}
//END Button 2

//Button 3
if(millis() - pastT3 > 500){ //CD
    pastT3 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt3) == 1){
        pastT31 = millis() + 150;
        Serial.println("Comand console...");
        bleKeyboard.press(KEY_RIGHT_GUI);
        bleKeyboard.press('r');
        bleKeyboard.release(KEY_RIGHT_GUI);
        bleKeyboard.release('r');
        flagg31 = 1;
    }
}

//timer!
if(millis() > pastT31 && flagg31 == 1){
    pastT31 = pastT31 + 150;
    bleKeyboard.print("cmd");
    flagg31 = 2;
}

```

```

    }
    //timer!
    if(millis() > pastT31 && flagg31 == 2){
        flagg31 = 0;
        bleKeyboard.press(176);
        bleKeyboard.release(176);
    }
//END Button 3
//Button 4
if(millis() - pastT4 > 200){ //KD
    pastT4 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt4) == 1){
        Serial.println("Open Task manager");
        bleKeyboard.press(KEY_LEFT_CTRL);
        bleKeyboard.press(KEY_LEFT_SHIFT);
        bleKeyboard.write(KEY_ESC);

        bleKeyboard.release(KEY_LEFT_CTRL);
        bleKeyboard.release(KEY_LEFT_SHIFT);
        bleKeyboard.release(KEY_ESC);
    }
}
//END Button 4
} //END Mode 1

//////////Mode 2//////////
if(flaggCh == 1){

    //Button 1
    if(millis() - pastT1 > 200){ //CD
        pastT1 = millis();

```

```

    if(bleKeyboard.isConnected() && !digitalRead(btt1) == 1) {
        Serial.println("Press Win+Shift+S");
        bleKeyboard.press(KEY_RIGHT_GUI);
        bleKeyboard.press(KEY_LEFT_SHIFT);
        bleKeyboard.press('s');
        bleKeyboard.release(KEY_RIGHT_GUI);
        bleKeyboard.release(KEY_LEFT_SHIFT);
        bleKeyboard.release('s');
    }
} //END Button 1

//Button 2

if(millis() - pastT2 > 300){ //CD
    pastT2 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt2) == 1){
        Serial.println("Open the Windows Settings");
        bleKeyboard.press(KEY_RIGHT_GUI);
        bleKeyboard.press('i');
        //Release
        bleKeyboard.release(KEY_RIGHT_GUI);
        bleKeyboard.release('i');
    }
}

//END Button 2

//Button 3

if(millis() - pastT3 > 500){ //CD
    pastT3 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt3) == 1){
        pastT31 = millis() + 150;
        Serial.println("Start Google Chrome");
        bleKeyboard.press(KEY_RIGHT_GUI);
        bleKeyboard.release(KEY_RIGHT_GUI);
    }
}

```

```

    flagg31 = 1;
}
}
//timer!
if(millis() > pastT31 && flagg31 == 1){
    pastT31 = pastT31 + 150;
    bleKeyboard.print("Google Chrome");
    flagg31 = 2;
}
//timer!
if(millis() > pastT31 && flagg31 == 2){
    flagg31 = 0;
    bleKeyboard.press(176);
    bleKeyboard.release(176);
}
//END Button 3
//Button 4
if(millis() - pastT4 > 200){ //KD
    pastT4 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt4) == 1){
        Serial.print("Worktime ");
        Printime();
        PrintimetoCOM();
    }
}
//END Button 4
} //END Mode 2
//////////Mode 3//////////
if(flaggCh == 2){
    //Button 1
    if(millis() - pastT1 > 200){ //CD

```

```

    pastT1 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt1) == 1){
        Serial.println("Press Play");
        bleKeyboard.press(KEY_MEDIA_PLAY_PAUSE);
        flaggPaus1 = 1;
    }
}

//Release
if(digitalRead(btt1) == 1 && flaggPaus1 == 1){
    flaggPaus1 = 0;
    bleKeyboard.release(KEY_MEDIA_PLAY_PAUSE);
    Serial.println("Release Play");
}

//END Button 1

//Button 2
if(millis() - pastT2 > 300){ //CD
    pastT2 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt2) == 1){
        Serial.println("Press Next track");
        bleKeyboard.press(KEY_MEDIA_NEXT_TRACK);

        bleKeyboard.release(KEY_MEDIA_NEXT_TRACK);
    }
}

//END Button 2

//Button 3
if(millis() - pastT3 > 500){ //CD
    pastT3 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt3) == 1){
        pastT31 = millis() + 150;
        Serial.println("Open Music");
    }
}

```

```

    bleKeyboard.press(KEY_RIGHT_GUI);
    bleKeyboard.release(KEY_RIGHT_GUI);
    flagg31 = 1;
}
}

//timer!
if(millis() > pastT31 && flagg31 == 1){
    pastT31 = pastT31 + 150;
    bleKeyboard.print("Spotify");
    flagg31 = 2;
}

//timer!
if(millis() > pastT31 && flagg31 == 2){
    flagg31 = 0;
    bleKeyboard.press(176);
    bleKeyboard.release(176);
}

//END Button 3

//Button 4
if(millis() - pastT4 > 200){ //KD
    pastT4 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt4) == 1){
        Serial.println("Press Previos track");
        bleKeyboard.press(KEY_MEDIA_PREVIOUS_TRACK);

        bleKeyboard.release(KEY_MEDIA_PREVIOUS_TRACK);
    }
}

//END Button 4

}///END Mode 3///

```

```

//////////Mode 4//////////
if(flaggCh == 3){
  //Button 1
  if(millis() - pastT1 > 200){ //CD
    pastT1 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt1) == 1){
      Serial.println("Next page in history");
      bleKeyboard.press(KEY_LEFT_ALT);
      bleKeyboard.press(KEY_RIGHT_ARROW);

      bleKeyboard.release(KEY_LEFT_ALT);
      bleKeyboard.release(KEY_RIGHT_ARROW);
    }
  }
  //END Button 1

  //Button 2
  if(millis() - pastT2 > 300){ //CD
    pastT2 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt2) == 1){
      Serial.println("Previous page in history");
      bleKeyboard.press(KEY_LEFT_ALT);
      bleKeyboard.press(KEY_LEFT_ARROW);

      bleKeyboard.release(KEY_LEFT_ALT);
      bleKeyboard.release(KEY_LEFT_ARROW);
    }
  }
  //END Button 2

  //Button 3
  if(millis() - pastT3 > 500){ //CD
    pastT3 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt3) == 1){

```

```

    Serial.println("Next open page");
    bleKeyboard.press(KEY_LEFT_CTRL);
    bleKeyboard.press(KEY_PAGE_DOWN);

    bleKeyboard.release(KEY_LEFT_CTRL);
    bleKeyboard.release(KEY_PAGE_DOWN);
}
//END Button 3
//Button 4
if(millis() - pastT4 > 200){ //KD
    pastT4 = millis();
    if(bleKeyboard.isConnected() && !digitalRead(btt4) == 1){
        Serial.println("Previous open page");
        bleKeyboard.press(KEY_LEFT_CTRL);
        bleKeyboard.press(KEY_PAGE_UP);

        bleKeyboard.release(KEY_LEFT_CTRL);
        bleKeyboard.release(KEY_PAGE_UP);
    }
}
//END Button 4
}///END Mode 4///
}

```

ДОДАТОК Г
Слайди презентації

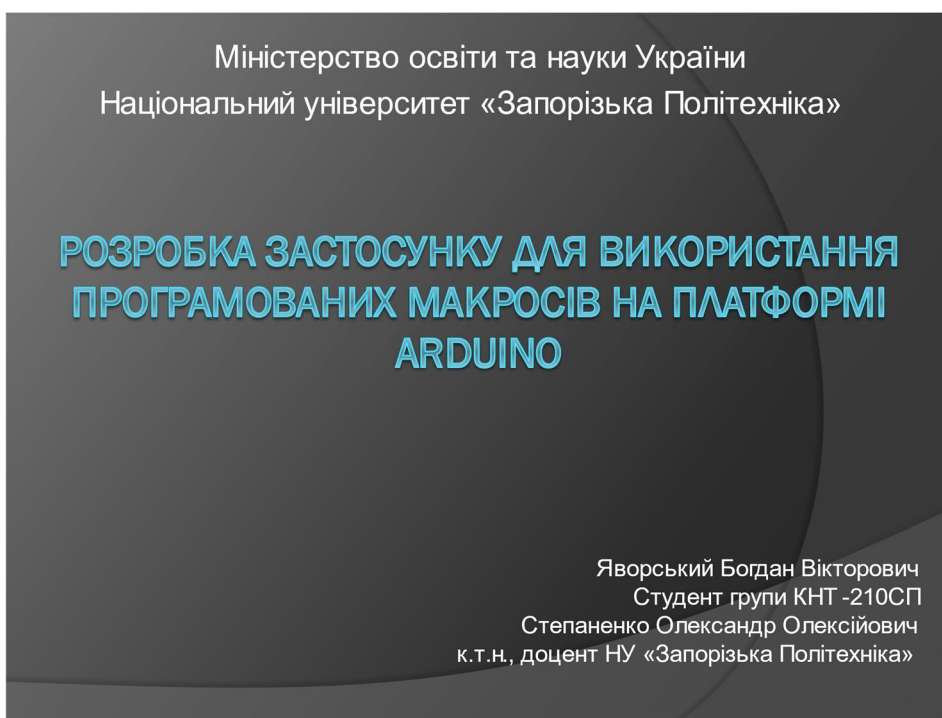


Рисунок Г.1 – Слайд 1

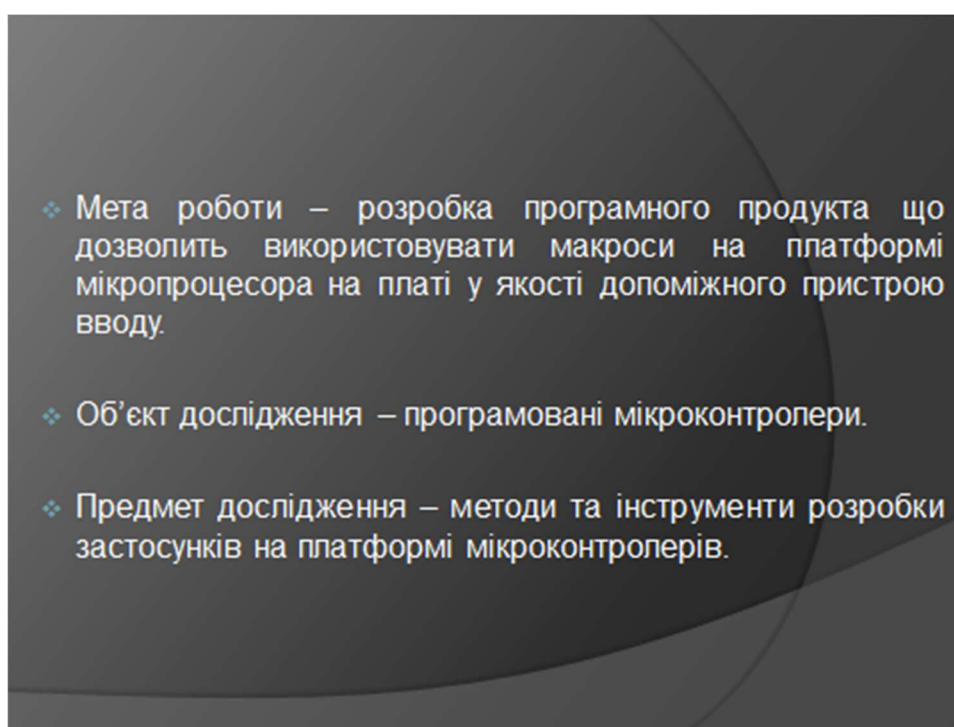


Рисунок Г.2 – Слайд 2

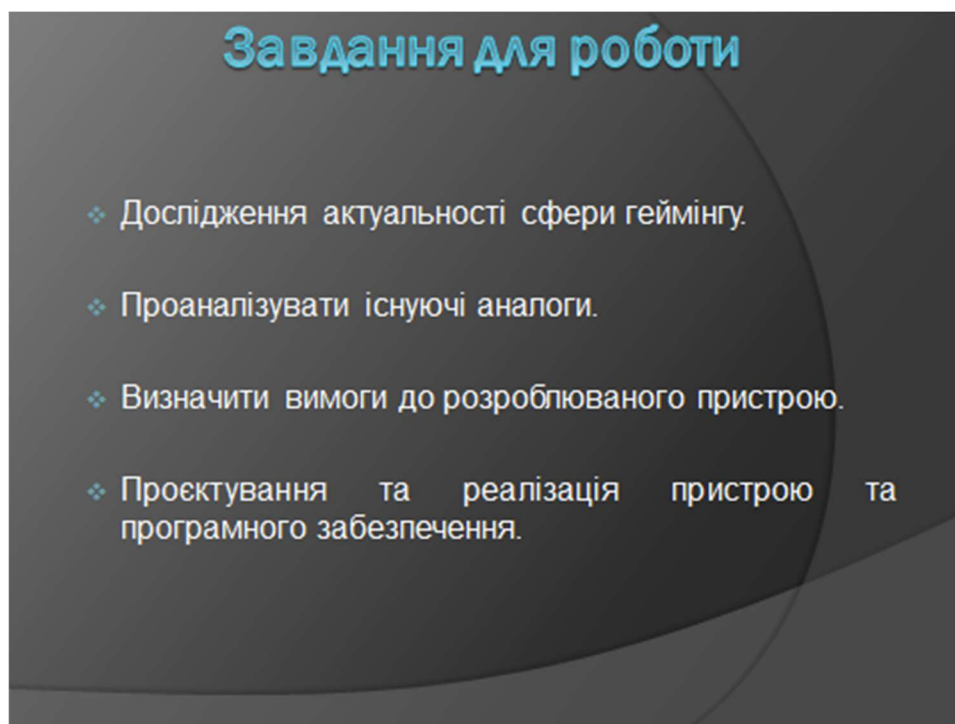


Рисунок Г.3 – Слайд 3

Аналіз аналогів

Пристрій	Ціна	Гнучкість	Зручність	Потенціал	Надійність
Саморобний пристрій	+	+	-	+	-/+
Покупна клавіатура	-/+	+	-/+	-	+
Oscar a4tech	-	-	+	-	+
Bloody a4tech	-	-	-	-	+

Рисунок Г.4 – Слайд 4

Порівняння апаратних аналогів

Плата	Чіп	Т. частота	Пам'ять	Порти	Bluetooth	Wi-Fi
Uno	ATmega328	16 МГц	32 Кб	14	-	-
Leonardo	ATmega32u4	16 МГц	32 Кб	20	-	-
Micro	ATmega32u4	16 МГц	32 Кб	20	-	-
ESP8266	1Core 32-bit L108	80 МГц	16 Мб	17	-	+
ESP32	2Cores 32-bit LX8 600	160 МГц	16 Мб	36	+	+

Рисунок Г.5 – Слайд 5

Порівняння середовищ розробки

Категорії	Arduino IDE	ESP IDF	Atmel Studio
Кількість матеріалів	+	-	-
Зручність використання	+	-	-
Багатозадачність	-	+	-/+
Відкритий код	+	+	-
Ефективність використання пам'яті	-	-	+
Вбудовані редактор та бібліотеки	+	-	-
Підтримка датчиків	+	-	+

Рисунок Г.6 – Слайд 6

Порівняння мов програмування

Критерії	Мова Arduino	C/C++	Assembler	Pascal
Оптимізація пам'яті	-	-/+	+	-
Підтримка драйверів	+	-/+	-	-
Документованість	+	-/+	-	-
Легкість в опануванні	+	-/+	-	+
Можливість налаштування	-	-/+	+	-
Середовище для використання	+	-	-	-

Рисунок Г.7 – Слайд 7

Схема підключення контактів

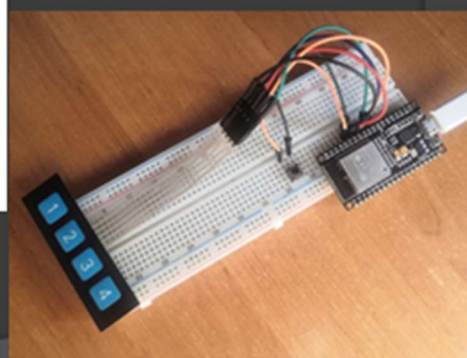
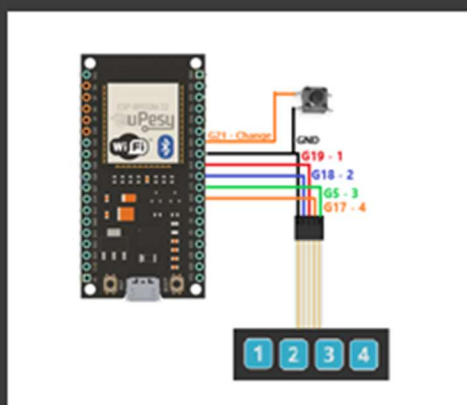


Рисунок Г.8 – Слайд 8



Рисунок Г.9 – Слайд 9

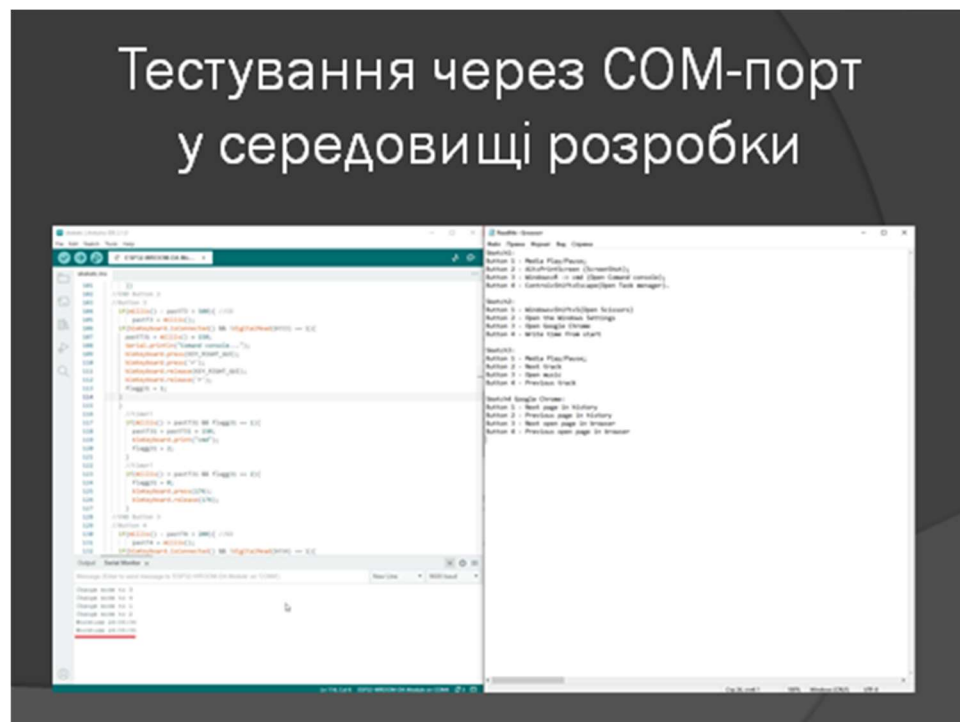


Рисунок Г.10 – Слайд 10

Демонстрація роботи пристрою

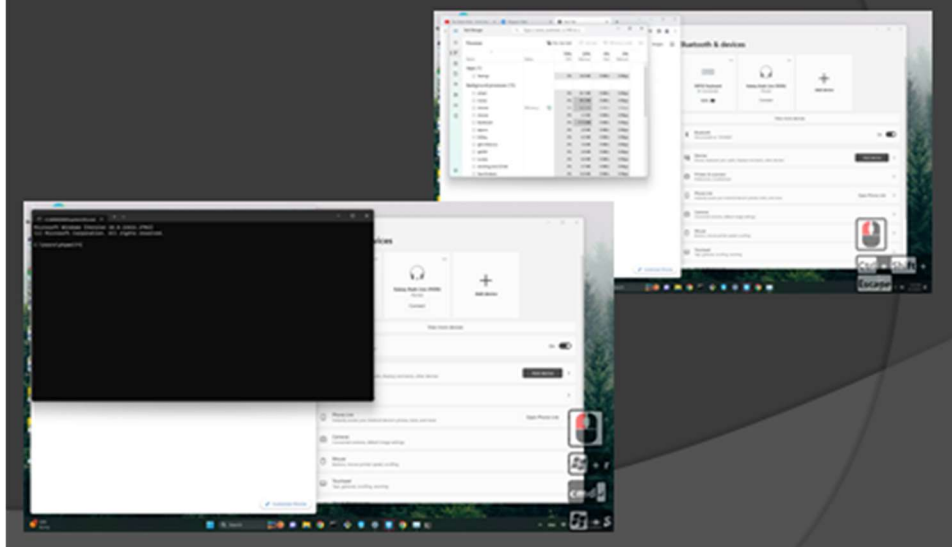


Рисунок Г.11 – Слайд 11

Висновки

- ❖ В ході виконання дипломної кваліфікаційної роботи бакалавра було проведено дослідження актуальності геймінгу.
- ❖ Створено пристрій, що живиться від різних джерел живлення, підключається до комп'ютерів через технологію Bluetooth та адекватно реагує на введений сигнал від користувача.
- ❖ Розроблено програмне забезпечення що дозволяє виконувати певні запрограмовані макроси, забезпечення ефективно використовує ресурси плати.
- ❖ Проведено тестування створеної програми, що показало здатність нормально виконувати макроси і стабільно працювати впродовж п'ятдесяти діб роботи.

Рисунок Г.12 – Слайд 12

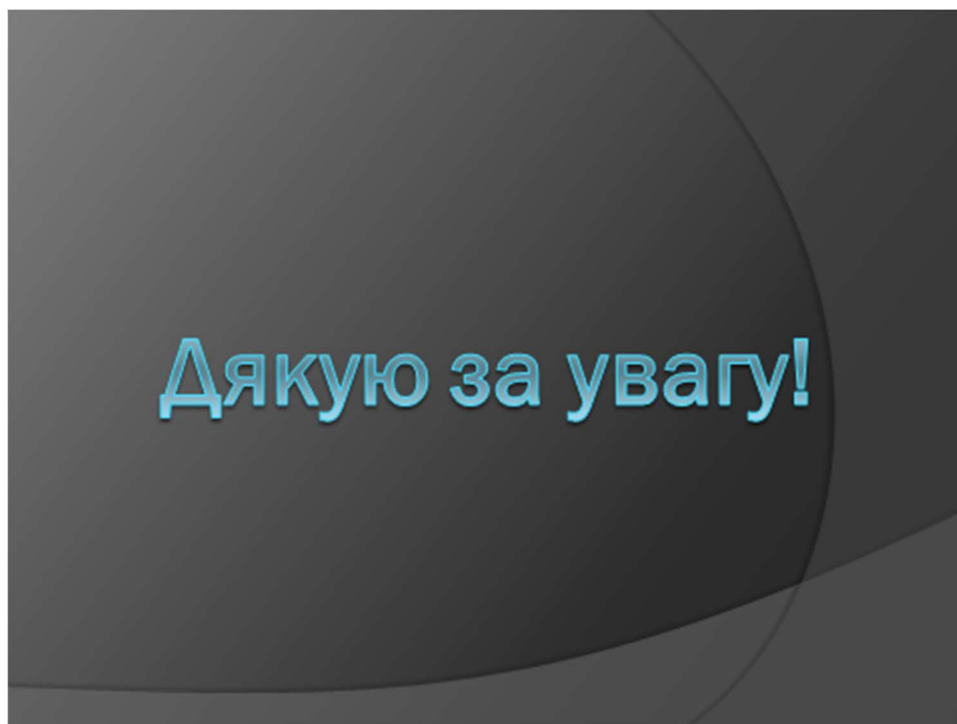


Рисунок Г.13 – Слайд 13