МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

Кафедра радіофізики та комп'ютерних технологій

Допустити до захисту

Завідувач кафедри
_____ проф. Карбовник І.Д.
«___» _____2023 р.

Кваліфікаційна робота

Бакалавр
(освітній ступінь)

Розробка пристрою для пошуку ключів на основі контролера ESP32

Виконав:
студент IV курсу групи ФЕП-41
спеціальності 121 – Інженерія
програмного забезпеченя

_____ Трайдакало М.О.

Науковий керівник:
_____ доц. Середницька Х.І.
«____» _____2023 р.

Рецензент:
_____проф. Попович Д.І.

Львів 2023

АНОТАЦІЯ

Дипломний проект на тему «Розробка пристрою для пошуку ключів на основі контролера ESP32» полягає у створенні пристрою для пошуку ключів на основі мікроконтролера ESP32 з використанням BLE технології. Цей пристрій має перебувати в сплячому режимі для того щоб споживати мінімум енергії, що дозволяє забезпечити живлення від джерела невеликих розмірів. При отриманні команди пристрій має активуватися і видавати звуковий сигнал. Звуковий сигнал буде означати, що пошук реалізовано і тоді за сигналом можна віднайти даний пристрій. Запропоновано макет брелку для пошуку ключів та спосіб передавання команди з віддаленого пристрою (смартфону, планшету чи комп’ютера) через Bluetooth.

The diploma project on "Development of a key search device based on the ESP32 controller" is to create a key search device based on the ESP32 microcontroller using BLE technology. This device should be in sleep mode in order to consume a minimum of energy, which allows it to be powered by a small source. When receiving a command, the device should activate and emit a beep. The sound signal will mean that the search is realized and then the device can be found by the signal. A model of a keychain for searching for keys and a method for transmitting a command from a remote device (smartphone, tablet, or computer) via Bluetooth are proposed.

ЗМІСТ

ВСТУП 4

РОЗДІЛ 1 АНАЛІЗ АРХІТЕКТУРИ ПРИСТРОЮ 6

1.1 Мікроконтролер 6

1.2 Середовище програмування 17

1.3 BLE технологія 21

РОЗДІЛ 2 АПАРАТНО-ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ ПРИСТРОЮ 24

2.1 Побудова принципової електричної схеми пристрою 24

2.2 Програмування контролера 25

2.3 Вибір засобів подачі команди для контролера 27

РОЗДІЛ 3 ТЕСТУВАННЯ ПРИСТРОЮ ДЛЯ ПОШУКУ КЛЮЧІВ .. 30

3.1 Виготовлення прототипу за заданою електричною схемою 30

3.2 Прошивка і тестування пристрою 31

3.3 Перевірка роботи пристрою 33

3.4 Аналіз енергоспоживання пристрою 37

ВИСНОВКИ 38

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 39

ДОДАТОК (код програми) 42

ВСТУП

Пристрої Інтернету речей (IoT) стають дедалі популярнішими завдяки своїй здатності підключатися та спілкуватися один з одним, збирати та обмінюватися даними, а також автоматизувати завдання. Вони мають широкий спектр застосування - від "розумних" будинків і будівель до промислової автоматизації та охорони здоров'я. Пристрої Інтернету речей надають цінну інформацію, підвищують ефективність і знижують витрати, дозволяючи здійснювати моніторинг і дистанційне керування в режимі реального часу. Їх важливість полягає в тому, що вони здатні трансформувати різні галузі та покращити наше повсякденне життя, роблячи його зручнішим, безпечнішим та стійкішим.

Технологічний прогрес сучасного світу рухається з величезною швидкістю та заповнює усі сфери людського життя. Технології оточують нас кожен день з усіх боків, починаючи з наших смартфонів і годинників і закінчуючи розумними будинками.

Всі розробки, пов’язані з апаратними системами, базуються на мікроконтролерах.

Мікроконтролер, або однокристальний мікрокомп'ютер — виконаний у вигляді мікросхеми спеціалізований комп'ютер, що включає мікропроцесор, оперативну та постійну пам'ять для збереження виконуваного коду програм і даних, порти вводу-виводу і блоки зі спеціальними функціями (лічильники, компаратори, АЦП та інші).

Використовується для керування електронними пристроями. По суті, це — однокристальний комп'ютер, здатний виконувати прості завдання. Використання однієї мікросхеми значно знижує розміри, енергоспоживання і вартість пристроїв, побудованих на базі мікроконтролерів.

Мікроконтролери можна зустріти в багатьох сучасних приладах, таких як телефони, пральні машини, вони відповідають за роботу двигунів і систем гальмування сучасних автомобілів, з їх допомогою створюються системи контролю і системи збору інформації. Більшість процесорів, що випускаються у світі — мікроконтролери.

ESP32 широко використовується в сучасних пристроях Інтернету речей завдяки своїй універсальності, продуктивності та економічності. Двоядерний процесор, вбудовані модулі Wi-Fi і Bluetooth та широкі можливості вводу/виводу роблять його придатним для різноманітних застосувань. Низьке енергоспоживання ESP32 і підтримка сплячого режиму дозволяють ефективно використовувати пристрої, що живляться від батарейок. Сумісність з Arduino і широка екосистема бібліотек та інструментів спрощує розробку, що робить його популярним вибором для створення прототипів і виробництва IoT. ESP32 є універсальною платформою для створення систем "розумного будинку", натільних пристроїв, промислового моніторингу або сільськогосподарських рішень, а також для розробки численних додатків IoT.

Тема розробки різноманітних систем автоматизації контролю параметрів стає все більш актуальною і знаходить собі використання в усіх сферах життя, починаючи з тематики розумних будинків і закінчуючи великими промисловими комплексами.

Таким чином, виходячи з вищесказаного, актуальність даної роботи неможливо недооцінити.

Метою проекту є створення системи брелоку для пошуку ключів на базі мікроконтролера ESP32.

Для досягнення поставленої мети були виконані наступні завдання:

- Аналіз особливостей мікроконтролерів
- Вибір засобів реалізації
- Вибір програмно-апаратної платформи
- Розробка функціональної схеми приладу
- Розробка принципової схеми приладу

Об’єкт дослідження — процес створення систем автоматизації.

Предмет дослідження — розробка системи брелоку для пошуку ключів.

РОЗДІЛ 1
АНАЛІЗ АРХІТЕКТУРИ ПРИСТРОЮ

1.1 Мікроконтролер

Мікроконтролер (MCU для мікроконтролерного блоку) - це невеликий комп’ютер на єдиному мікросхемі інтегральної мікросхеми (МОП) металоксид-оксид-напівпровідник. Мікроконтролер містить один або кілька центральних процесорів (ядра процесора) разом із пам'яттю та програмованою периферією вводу / виводу. Пам'ять програм у вигляді сегнетоелектричної оперативної пам'яті, NOR-флеш-пам'яті або OTP-ROM також часто включається в мікросхему, а також невелика кількість оперативної пам'яті. Мікроконтролери призначені для вбудованих програм, на відміну від мікропроцесорів, що використовуються в персональних комп'ютерах або інших додатках загального призначення, що складаються з різних дискретних мікросхем.

У сучасній термінології мікроконтролер схожий на, але менш витончений, ніж система на мікросхемі (SoC). SoC може включати мікроконтролер в якості одного зі своїх компонентів, але зазвичай інтегрує його з вдосконаленою периферією, такою як графічний процесор (GPU), модуль Wi-Fi або один або кілька співпроцесорів.

Мікроконтролери використовуються в автоматично керованих виробах та пристроях, таких як автомобільні системи управління двигуном, імплантовані медичні пристрої, пульти дистанційного керування, офісні машини, прилади, електроінструменти, іграшки та інші вбудовані системи. Зменшуючи розмір та вартість порівняно з конструкцією, яка використовує окремий мікропроцесор, пам’ять та пристрої введення / виведення, мікроконтролери роблять економічним цифровий контроль ще більшої кількості пристроїв та процесів. Поширені мікроконтролери змішаного сигналу, що інтегрують аналогові компоненти, необхідні для управління нецифровими електронними системами. У контексті Інтернету речей мікроконтролери є економічним і популярним засобом збору даних, зондування та керування фізичним світом як крайні пристрої.

Деякі мікроконтролери можуть використовувати чотирибітові слова і працювати на частотах до 4 кГц для низького енергоспоживання (однозначні міліват або мікроват). Як правило, вони мають можливість зберегти функціональність під час очікування такої події, як натискання кнопки або інше переривання; споживання енергії під час сну (годинник процесора та більшість периферійних пристроїв вимкнені) може бути просто нановат, що робить багато з них добре придатними для тривалого використання батареї. Інші мікроконтролери можуть виконувати критично важливі функції, де їм, можливо, доведеться діяти більше як цифровий процесор сигналів (DSP), з вищою тактовою частотою та енергоспоживанням.

Походження як мікропроцесора, так і мікроконтролера можна простежити до винаходу MOSFET (польовий транзистор з оксидом метал-напівпровідник), також відомого як MOS-транзистор. [1] Він був винайдений Мохамедом М. Аталою та Давоном Кангом у лабораторіях Белл у 1959 р. Та вперше продемонстрований у 1960 р. [2] Того ж року Аталла запропонувала концепцію інтегральної схеми MOS, яка являла собою мікросхему інтегральної мікросхеми, виготовлену з MOSFET-транзисторів. [3] До 1964 року мікросхеми MOS досягли вищої щільності транзисторів і нижчих виробничих витрат, ніж біполярні мікросхеми. Мікросхеми MOS додатково збільшувались із швидкістю, передбаченою законом Мура, що призвело до широкомасштабної інтеграції (LSI) з сотнями транзисторів на одному чіпі MOS до кінця 1960-х. Застосування мікросхем MOS LSI до обчислень було основою для перших мікропроцесорів, оскільки інженери почали визнавати, що на одному чіпі MOS LSI може міститися повний комп'ютерний процесор. [1]

Перші багатопроцесорні мікропроцесори, чотирифазні системи AL1 у 1969 році та Garrett AiResearch MP944 у 1970 році, були розроблені з декількома чіпами MOS LSI. Першим однопроцесорним мікропроцесором став Intel 4004, випущений на одному чіпі MOS LSI в 1971 році. Він був розроблений Федеріко Фаггіном, використовуючи його кремній-затворні технології MOS, разом з інженерами Intel Марсіаном Хоффом і Станом Мазором та інженером Busicom Масатоши Шима. [4] За ним послідували 4-розрядні Intel 4040, 8-розрядні Intel 8008 та 8-розрядні Intel 8080. Всі ці процесори потребували декількох зовнішніх мікросхем для реалізації робочої системи,

включаючи мікросхеми пам'яті та периферійного інтерфейсу. Як результат, загальна вартість системи становила кілька сотень (1970-ті роки) доларів США, що унеможливлювало економічну комп’ютеризацію дрібних приладів. Технологія MOS представила мікропроцесори 6501 та 6502 до 100 доларів США з основною метою усунення цієї економічної перешкоди, але ці мікропроцесори все ще потребували зовнішньої підтримки, пам'яті та периферійних чіпів, що дозволило зберегти загальну вартість системи в сотні доларів.

Одна книга зараховує інженерів TI Гері Буна і Майкла Кокрана до успішного створення першого мікроконтролера в 1971 р. Результатом їх роботи став TMS 1000, який став комерційно доступним у 1974 р. Він поєднував пам'ять лише для читання, пам'ять читання / запис, процесор і годинник на одному чіпі і був орієнтований на вбудовані системи. [5]

На початку середини 70-х років японські виробники електроніки почали випускати мікроконтролери для автомобілів, включаючи 4-розрядні мікроконтролери для розваг в машині, автоматичні двірники, електронні замки та панель приладів, а також 8-бітні мікроконтролери для управління двигуном. [6]

Частково у відповідь на існування однокристального TMS 1000 [7], Intel розробила комп'ютерну систему на мікросхемі, оптимізованій для програм управління, Intel 8048, комерційні деталі вперше надійшли в 1977 р. [7] Він поєднав оперативну пам’ять і ПЗУ на одному мікросхемі з мікропроцесором. Серед численних додатків цей чіп врешті-решт потрапить у понад мільярд клавіатур ПК. Тоді президент Intel Люк Дж. Валентер заявив, що мікроконтролер був одним з найуспішніших продуктів в історії компанії, і він розширив бюджет підрозділу мікроконтролерів більш ніж на 25%.

У цей час більшість мікроконтролерів мали паралельні варіанти. Один мав програмну пам’ять EPROM з прозорим кварцовим вікном у кришці упаковки, що дозволяє стирати його під впливом ультрафіолету. Ці стираються чіпи часто використовували для створення прототипів. Іншим варіантом була або запрограмована ПЗУ-маска, або варіант PROM, який програмувався лише один раз. Для останнього іноді вживалося позначення OTP, що означає "одноразовий

програмований". У мікроконтролері OTP PROM зазвичай був ідентичного типу EPROM, але в мікросхемі не було кварцового вікна; оскільки не було можливості піддати EPROM ультрафіолетовому світлу, його не можна було стерти. Оскільки для стираних версій потрібні були керамічні пакети з кварцовими вікнами, вони були значно дорожчими, ніж версії OTP, які можна було виготовити в дешевих непрозорих пластикових упаковках. Для стираних варіантів замість менш дорогого скла вимагався кварц для його прозорості для ультрафіолетового світла, для якого скло в основному непрозоре, але головним відмінником вартості була сама керамічна упаковка.

У 1993 році впровадження пам'яті EEPROM дозволило мікроконтролерам (починаючи з Microchip PIC16C84) [8] швидко електрично стирати без дорогого пакета, як це потрібно для EPROM, дозволяючи як швидке прототипування, так і внутрішньосистемне програмування. (Технологія EEPROM була доступна до цього часу [9], але раніше EEPROM була дорожчою та менш довговічною, що робило її непридатною для недорогих мікроконтролерів масового виробництва.) Того ж року Atmel представила перший мікроконтролер із використанням флеш-пам'яті , спеціальний тип EEPROM. [10] Інші компанії швидко наслідували цей приклад, використовуючи обидва типи пам'яті.

На сьогоднішній день мікроконтролери дешеві та легко доступні для любителів, з великими інтернет-спільнотами навколо певних процесорів.

21 червня 2018 року Мічиганський університет оголосив про "найменший комп'ютер у світі". Пристрій являє собою "0,04 мм3 бездротову бездротову датчикову систему із вбудованим процесором Cortex-M0 + та оптичним зв'язком для вимірювання температури в стільниковій мережі". Він "вимірює лише 0,3 мм в сторону - занижує рисове зерно. [...] На додаток до оперативної пам'яті та фотоелектрики, нові обчислювальні пристрої мають процесори та бездротові передавачі та приймачі. Оскільки вони занадто малі, щоб мати звичайні радіоантени, вони приймають і передають дані з видимим світлом. Базова станція забезпечує світло для живлення та програмування, і вона приймає дані ". [24] Пристрій займає 1/10 розміру від заявленого раніше світового рекорду комп'ютера IBM починаючи з

місяців у березні 2018 року [25], який «менший за зерно солі» [26], має мільйон транзисторів, вартість його виготовлення становить менше 0,10 доларів США, і в поєднанні з технологією блокчейну призначена для логістики та «крипто -Якорі "- програми цифрових відбитків пальців. [27]

Мікроконтролер можна вважати автономною системою з процесором, пам'яттю та периферійними пристроями і може використовуватися як вбудована система. [28] Більшість використовуваних сьогодні мікроконтролерів вбудовані в інші машини, такі як автомобілі, телефони, прилади та периферія для комп'ютерних систем.

Хоча деякі вбудовані системи дуже складні, багато хто має мінімальні вимоги до пам'яті та тривалості програми, без операційної системи та низької складності програмного забезпечення. Типові пристрої введення та виведення включають перемикачі, реле, соленоїди, світлодіоди, маленькі або спеціальні рідкокристалічні дисплеї, радіочастотні пристрої та датчики для таких даних, як температура, вологість, рівень освітленості тощо. Вбудовані системи зазвичай не мають клавіатури, екрану, дисків , принтерів або інших впізнаваних пристроїв вводу-виводу персонального комп’ютера, і можуть бути відсутні пристрої людської взаємодії будь-якого виду.

Мікроконтролери повинні забезпечувати реагування у реальному часі (передбачувано, хоча і не обов’язково швидко) на події у вбудованій системі, яку вони контролюють. Коли відбуваються певні події, система переривання може подати сигнал процесору припинити обробку поточної послідовності інструкцій і розпочати процедуру обслуговування переривань (ISR, або "обробник переривань"), яка виконуватиме будь-яку обробку, необхідну на основі джерела переривання, до повернення до початкової послідовності інструкцій. Можливі джерела переривань залежать від пристрою і часто включають такі події, як переповнення внутрішнього таймера, завершення аналого-цифрового перетворення, зміна логічного рівня на вході, наприклад, від натискання кнопки, та дані, отримані на лінії зв'язку. Там, де споживання енергії є важливим, як у акумуляторних пристроях, переривання можуть також вивести мікроконтролер із режиму сну з низьким енергоспоживанням, коли процесор зупиняється, поки не буде потрібно щось робити через периферійну подію.

Зазвичай програми мікроконтролера повинні поміщатися в доступну вбудовану пам'ять, оскільки забезпечити систему зовнішньою розширюваною пам'яттю буде дорого. Компілятори та асемблери використовуються для перетворення як високого рівня, так і асемблерних кодів мови в компактний машинний код для зберігання в пам'яті мікроконтролера. Залежно від пристрою, пам’ять програми може бути постійною, лише для читання, яка може бути запрограмована лише на заводі, а може бути зміненою у полі флеш-пам’яттю або стираною пам’яттю лише для читання.

Виробники часто випускають спеціальні версії своїх мікроконтролерів, щоб допомогти апаратному та програмному забезпеченню цільової системи. Спочатку вони включали версії EPROM, які мають "вікно" у верхній частині пристрою, через яке пам'ять програми може бути стерта ультрафіолетовим світлом, готова до перепрограмування після програмування ("горіння") та тестового циклу. Починаючи з 1998 року, версії EPROM є рідкісними і були замінені EEPROM та флеш-пам'яттю, які простіші у використанні (можуть бути стерті електронним способом) та дешевші у виробництві.

Інші версії можуть бути доступні там, де ПЗП доступний як зовнішній пристрій, а не як внутрішня пам’ять, однак вони стають рідкісними через широку доступність дешевих програмістів мікроконтролерів.

Застосування програмованих на місцях пристроїв на мікроконтролері може дозволити польове оновлення мікропрограми або дозволити пізні заводські перегляди продуктів, які були зібрані, але ще не відвантажені. Програмована пам'ять також зменшує час виконання, необхідний для розгортання нового продукту.

Там, де потрібні сотні тисяч однакових пристроїв, використання деталей, запрограмованих на момент виготовлення, може бути економічним. Ці "запрограмовані" маски частини мають програму, закладену так само, як і логіка мікросхеми, одночасно.

Індивідуальний мікроконтролер містить блок цифрової логіки, який можна персоналізувати для додаткових можливостей обробки, периферійних пристроїв та інтерфейсів, адаптованих до вимог програми. Одним із прикладів є AT91CAP від Atmel.

Мікроконтролери, як правило, містять від кількох до десятків входів / виходів загального призначення (GPIO). Висновки GPIO - це програмне забезпечення, яке можна налаштувати як на вхід, так і на вихід. Коли шпильки GPIO налаштовані на вхідний стан, вони часто використовуються для зчитування датчиків або зовнішніх сигналів. Налаштовані на вихідний стан, штифти GPIO можуть управляти зовнішніми пристроями, такими як світлодіоди або двигуни, часто опосередковано, через зовнішню силову електроніку.

Багато вбудованих систем повинні зчитувати датчики, які виробляють аналогові сигнали. Це призначення аналого-цифрового перетворювача (АЦП). Оскільки процесори побудовані для інтерпретації та обробки цифрових даних, тобто 1s та 0s, вони не можуть нічого робити з аналоговими сигналами, які може надсилати йому пристрій. Отже, аналого-цифровий перетворювач використовується для перетворення вхідних даних у форму, яку процесор може розпізнати. Менш поширеною особливістю деяких мікроконтролерів є цифро-аналоговий перетворювач (ЦАП), який дозволяє процесору виводити аналогові сигнали або рівні напруги.

На додаток до перетворювачів, багато вбудованих мікропроцесорів також включають різноманітні таймери. Одним з найпоширеніших типів таймерів є програмований інтервальний таймер (PIT). ПДФО може або відраховувати від якогось значення до нуля, або до ємності реєстру підрахунку, переповнюючись до нуля. Як тільки він досягає нуля, він надсилає переривання процесору, вказуючи, що він закінчив підрахунок. Це корисно для таких пристроїв, як термостати, які періодично перевіряють температуру навколо них, щоб побачити, чи потрібно їм вмикати кондиціонер, обігрівач тощо.

Спеціальний блок широтно-імпульсної модуляції (ШІМ) дозволяє центральному процесору керувати перетворювачами потужності, резистивними навантаженнями, двигунами тощо, не використовуючи багато ресурсів центрального процесора в обмежених циклах таймера.

Універсальний асинхронний блок приймача / передавача (UART) дозволяє отримувати та передавати дані по послідовній лінії з дуже невеликим навантаженням на центральний процесор. Виділене вбудоване обладнання також часто включає

можливості для зв'язку з іншими пристроями (чіпами) у цифрових форматах, таких як Інтегральна схема (I²C), Послідовний периферійний інтерфейс (SPI), Універсальна послідовна шина (USB) та Ethernet. [29]

Мікроконтролери можуть не реалізовувати зовнішню адресу або шину даних, оскільки вони інтегрують оперативну пам’ять і енергонезалежну пам’ять на одному мікросхемі з процесором. Використовуючи менше штифтів, чіп можна помістити у значно менший та дешевий пакет.

Інтеграція пам'яті та інших периферійних пристроїв на одному мікросхемі та їх тестування як цілісності збільшує вартість цього мікросхеми, але часто призводить до зниження чистої вартості вбудованої системи в цілому. Навіть якщо вартість центрального процесора, який має інтегровану периферію, трохи перевищує вартість центрального процесора та зовнішньої периферії, наявність меншої кількості мікросхем, як правило, дозволяє меншу та дешевшу друковану плату та зменшує робочу силу, необхідну для складання та тестування друкованої плати, на додаток до тенденції до зменшення рівня дефектів готової збірки.

Мікроконтролер - це одна інтегральна схема, яка зазвичай має такі функції:

- центральний процесор - починаючи від невеликих і простих 4-розрядних процесорів і закінчуючи складними 32-розрядними або 64-розрядними процесорами
- енергонезалежна пам’ять (ОЗП) для зберігання даних
- ROM, EPROM, EEPROM або флеш-пам'ять для зберігання програм та робочих параметрів
- дискретні вхідні та вихідні біти, що дозволяють керувати або виявляти логічний стан окремого виводу пакета
- послідовний ввід / вихід, наприклад послідовний порт (UART)
- інші послідовні інтерфейси зв'язку, такі як I²C, послідовний периферійний інтерфейс та контрольна мережа для системного з'єднання
- периферійні пристрої, такі як таймери, лічильники подій, генератори ШІМ та сторожові органи

- тактовий генератор - часто генератор для кварцового синхронізуючого кристала, резонатора або RC-схеми
- багато з них включають аналого-цифрові перетворювачі, деякі - цифро-аналогові перетворювачі
- внутрішньосхемне програмування та підтримка налагодження в ланцюзі

Ця інтеграція різко зменшує кількість мікросхем та обсяг проводки та місця на друкованій платі, що було б потрібно для створення еквівалентних систем з використанням окремих мікросхем. Крім того, на пристроях з низькою кількістю виводів кожен висновок може взаємодіяти з кількома внутрішніми периферійними пристроями, при цьому функція виводу вибирається програмним забезпеченням. Це дозволяє використовувати деталь у більш широкому спектрі застосувань, ніж якби штифти мали спеціальні функції.

Мікроконтролери виявилися надзвичайно популярними у вбудованих системах з моменту їх появи в 1970-х.

Деякі мікроконтролери використовують архітектуру Гарварда: окремі шини пам’яті для інструкцій та даних дозволяють одночасно здійснювати доступ. Там, де використовується архітектура Гарварда, слова інструкцій для процесора можуть мати інший бітовий розмір, ніж довжина внутрішньої пам'яті та регістрів; наприклад: 12-бітові інструкції, що використовуються з 8-бітними регістрами даних.

Рішення, яку периферію інтегрувати, часто є складним. Постачальники мікроконтролерів часто торгують робочими частотами та гнучкістю конструкції системи з урахуванням вимог своїх споживачів щодо часу випуску на ринок та загальної нижчої вартості системи. Виробники повинні збалансувати необхідність мінімізувати розмір чіпа та додаткові функції.

Архітектура мікроконтролера сильно відрізняється. Деякі конструкції включають мікропроцесорні ядра загального призначення з одним або кількома функціями ПЗУ, оперативної пам’яті або введення / виводу, інтегрованих у пакет. Інші конструкції призначені для контрольних програм. Набір інструкцій мікроконтролера зазвичай містить багато інструкцій, призначених для маніпулювання бітами (побітові

операції), щоб зробити програми управління більш компактними. [30] Наприклад, процесор загального призначення може вимагати декількох інструкцій для тестування біта в регістрі та гілці, якщо біт встановлений, де мікроконтролер може мати одну інструкцію для забезпечення тієї загальнообов’язкової функції.

Мікроконтролери традиційно не мають математичного співпроцесора, тому арифметика з плаваючою комою виконується програмним забезпеченням. Однак деякі останні конструкції включають оптимізовані функції FPU та DSP. Прикладом може бути лінія на основі PIC32 MIPS від Microchip.

Мікроконтролери спочатку програмувались лише на мові асемблера, але різні мови програмування високого рівня, такі як C, Python та JavaScript, зараз також широко використовуються для націлювання мікроконтролерів та вбудованих систем. [31] Компілятори для мов загального призначення зазвичай матимуть деякі обмеження, а також вдосконалення для кращої підтримки унікальних характеристик мікроконтролерів. Деякі мікроконтролери мають середовища, які допомагають розробляти певні типи програм. Постачальники мікроконтролерів часто роблять інструменти у вільному доступі, щоб полегшити використання обладнання.

Мікроконтролери зі спеціальним обладнанням можуть вимагати власних нестандартних діалектів мови C, таких як SDCC для 8051, які перешкоджають використанню стандартних інструментів (таких як бібліотеки коду або інструменти статичного аналізу) навіть для коду, не пов'язаного з апаратними характеристиками. Інтерпретатори можуть також містити нестандартні функції, такі як MicroPython, хоча форк, CircuitPython, намагався перенести апаратні залежності до бібліотек і забезпечити відповідність мови більш стандартному стандарту CPython.

Прошивка інтерпретатора також доступна для деяких мікроконтролерів. Наприклад, BASIC на ранніх мікроконтролерах Intel 8052; [32] BASIC і FORTH на Zilog Z8 [33], а також деякі сучасні пристрої. Зазвичай ці інтерпретатори підтримують інтерактивне програмування.

Симулятори доступні для деяких мікроконтролерів. Вони дозволяють розробнику проаналізувати поведінку мікроконтролера та їх програми, якщо вони використовували фактичну деталь. Симулятор покаже внутрішній стан процесора, а

також стан виходів, а також дозволить генерувати вхідні сигнали. Хоча, з одного боку, більшість симуляторів не зможуть моделювати багато іншого обладнання в системі, вони можуть виконувати умови, які в іншому випадку важко відтворити за власним бажанням у фізичній реалізації, і можуть бути найшвидшим способом налагодження та аналізу проблеми.

Останні мікроконтролери часто інтегровані з мікросхемою налагодження на мікросхемі, яка при доступі до внутрішньосхемового емулятора (ICE) через JTAG дозволяє налагоджувати прошивку за допомогою налагоджувача. ICE в режимі реального часу може дозволяти переглядати та / або маніпулювати внутрішніми станами під час роботи. ICE, що відстежує, може записувати виконану програму та стан MCU до / після точки запуску.

На відміну від комп'ютерів загального призначення, мікроконтролери, що використовуються у вбудованих системах, часто прагнуть оптимізувати затримку переривань над пропускною здатністю інструкцій. Проблеми включають як зменшення затримки, так і підвищення її передбачуваності (для підтримки контролю в реальному часі).

Коли електронний пристрій спричиняє переривання, під час перемикання контексту проміжні результати (регістри) потрібно зберігати перед тим, як програмне забезпечення, відповідальне за обробку переривання, може працювати. Вони також повинні бути відновлені після закінчення обробки переривань. Якщо реєстрів процесорів більше, цей процес збереження та відновлення може зайняти більше часу, збільшуючи час очікування. (Якщо ISR не вимагає використання деяких регістрів, він може просто залишити їх у спокої, а не зберігати та відновлювати, тому в цьому випадку ці регістри не пов'язані із затримкою.) Способи зменшення такого контексту / відновлення затримки включають наявність відносно невелика кількість реєстрів у центральних процесорних блоках (небажано, оскільки це суттєво сповільнює більшість обробок без переривань) або, принаймні, апаратне забезпечення не зберігає їх усі (це не вдається, якщо програмне забезпечення потім має компенсувати, зберігаючи решту "вручну") . Інший прийом передбачає витрачання кремнієвих воріт на "тіньові регістри": Один або кілька повторюваних регістрів, що використовуються

лише програмним забезпеченням переривання, можливо, підтримуючи виділений стек.

1.2 Середовище програмування

Arduino - компанія з програмним забезпеченням та програмним забезпеченням з відкритим кодом, проектна та споживча спільнота, яка розробляє та виготовляє однопланові мікроконтролери та набори мікроконтролерів для побудови цифрових пристроїв. Її продукція ліцензується за Ліцензією загальної публічної ліцензії GNU (LGPL) або General Public License (GPL) GNU, що дозволяє виробляти плати Arduino та розповсюджувати програмне забезпечення будь-ким. Плати Arduino випускаються у продажу в заздалегідь зібраному вигляді або як комплекти "зроби сам".

Конструкції плат Arduino використовують різноманітні мікропроцесори та контролери. Плати оснащені наборами цифрових та аналогових штифтів для вводу / виводу (вводу / виводу), які можуть поєднуватися з різними розширювальними платами («шилдами») або макетними платами (для прототипування) та іншими схемами. На платах є послідовний інтерфейс зв'язку, включаючи універсальну послідовну шину (USB) на деяких моделях, які також використовуються для завантаження програм з персональних комп'ютерів. Мікроконтролери можуть бути запрограмовані за допомогою мов програмування C і C ++. Окрім використання традиційних ланцюжків інструментів компілятора, проект Arduino забезпечує інтегроване середовище розробки (IDE) на основі мовного проекту Processing.

Проект Arduino розпочався в 2005 році як програма для студентів Інституту дизайну взаємодії Іврея в Івреї, Італія, з метою забезпечити недорогий та простий спосіб для початківців та професіоналів створити пристрої, які взаємодіють із оточенням за допомогою датчиків та пускачів. Загальні приклади таких пристроїв, призначених для початківців любителів, включають прості роботи, термостати та детектори руху.

Arduino - апаратне забезпечення з відкритим кодом. Конструкції посилань на апаратуру поширюються за ліцензією Creative Commons Attribution Share-Alike 2.5 та

доступні на веб-сайті Arduino. Також доступні файли компонування та виробництва для деяких версій обладнання.

Хоча конструкції апаратних та програмних засобів вільно доступні за ліцензіями на copyleft, розробники вимагають, щоб ім'я Arduino було ексклюзивним для офіційного продукту і не використовувалось для похідних робіт без дозволу. В офіційному політичному документі щодо використання назви Arduino підкреслюється, що проект відкритий для включення роботи в офіційний продукт інших. Кілька продуктів, сумісних з Arduino, комерційно випущених, уникали назви проекту, використовуючи різні назви, що закінчуються на -duino.

Більшість плат Arduino складаються з 8-розрядного мікроконтролера AVR Atmel (ATmega8, ATmega168, ATmega328, ATmega1280 або ATmega2560) з різною кількістю флеш-пам’яті, штифтами та можливостями. 32-розрядний Arduino Due на базі Atmel SAM3X8E був представлений у 2012 році. На платах використовуються одно- або дворядкові штифти або жіночі заголовки, що полегшують з'єднання для програмування та включення в інші схеми. Вони можуть з'єднуватися з додатковими модулями, що називаються шилдами. Декілька і, можливо, складених екранів можуть бути адресовані індивідуально через послідовну шину I²C. Більшість плат включають 5 В лінійний регулятор і 16 МГц кристалічний генератор або керамічний резонатор. Деякі конструкції, такі як LilyPad, працюють на частоті 8 МГц і відмовляються від бортового регулятора напруги через конкретні обмеження форм-фактору.

Мікроконтролери Arduino попередньо запрограмовані з завантажувальним завантажувачем, який спрощує завантаження програм на флеш-пам'ять на мікросхемі. Замовчувачем завантажувача Arduino Uno є завантажувач Optiboot. Плати завантажуються програмним кодом через послідовне підключення до іншого комп'ютера. Деякі послідовні плати Arduino містять схему перемикання рівня для перетворення між логічними рівнями RS-232 та сигналами рівня транзистор-транзистор (TTL). Поточні плати Arduino програмуються за допомогою універсальної послідовної шини (USB), реалізованої за допомогою мікросхем USB-послідовного адаптера, таких як FTDI FT232. Деякі плати, такі як більш пізні моделі Uno, замінюють мікросхему FTDI окремим чіпом AVR, що містить мікропрограму від USB

до послідовного перепрограмування, яку можна перепрограмувати за допомогою власного заголовка ICSP. Інші варіанти, такі як Arduino Mini та неофіційний Boarduino, використовують знімну плату або кабель для адаптер USB-до-серійного адаптера або кабель, Bluetooth або інші способи. При використанні з традиційними інструментами мікроконтролерів замість Arduino IDE використовується стандартне програмування в системному програмуванні (ISP) AVR.

Плата Arduino виставляє більшість контактів вводу / виводу мікроконтролера для використання в інших схемах. Diecimila, Duemilanove, і струм Uno забезпечують 14 цифрових вводів-виводів, шість з яких можуть виробляти модульовані імпульсні сигнали і шість аналогових входів, які також можуть використовуватися як шість цифрових I / O шпильки. Ці шпильки знаходяться у верхній частині плати, через жіночі 0,1-дюймові (2,54 мм) заголовки. Кілька екранів додатків для додатків також доступні у продажу. Плати Arduino Nano і Arduino-сумісні плати голих кісток та Boarduino можуть забезпечити шпильки для чоловіків на нижній стороні плати, які можуть вставлятись у плати без пайки.

Існує багато плат, сумісних з Arduino та похідних від Arduino. Деякі є функціонально еквівалентними Arduino і можуть використовуватися взаємозамінно. Багато хто вдосконалює базовий Arduino, додаючи вихідні драйвери, часто для використання в шкільному рівні, щоб спростити створення баггі та маленьких роботів. Інші електрично еквівалентні, але змінюють форм-фактор, інколи зберігаючи сумісність із шилдами, іноді ні. Деякі варіанти використовують різні процесори різної сумісності.

Вихідний код для IDE випускається згідно з Загальною публічною ліцензією GNU, версія 2. IDE Arduino підтримує мови C та C ++, використовуючи спеціальні правила структуризації коду. Arduino IDE постачає бібліотеку програмного забезпечення від проекту Wiring, яка забезпечує безліч загальних процедур введення та виведення даних. Написаний користувачем код вимагає лише двох основних функцій для запуску ескізу та основного циклу програми, які компілюються та пов'язуються із заглушкою програми main () у виконувану циклічну виконавчу програму з ланцюжком інструментів GNU, що також входить до розподілу IDE. У Arduino IDE

використовується програмна avrdude для перетворення виконуваного коду в текстовий файл у шістнадцятковому кодуванні, який завантажується на плату Arduino програмою-завантажувачем у програмному забезпеченні плати. За замовчуванням avrdude використовується як інструмент для завантаження для відтворення коду користувача на офіційні плати Arduino

З ростом популярності Arduino як програмної платформи, інші постачальники почали впроваджувати власні компілятори та інструменти з відкритим кодом (ядра), які можуть створювати та завантажувати ескізи в інші MCU, які не підтримуються офіційною лінією MCU Arduino.

У жовтні 2019 року організація Arduino почала надавати ранній доступ до нового Arduino Pro IDE з налагодженням та іншими розширеними функціями.

Деякі з ключових особливостей Arduino IDE:

- Простий та інтуїтивно зрозумілий інтерфейс: Arduino IDE має простий та інтуїтивно зрозумілий інтерфейс, що робить його доступним навіть для початківців. Він включає в себе текстовий редактор, де ви пишете свій код, панель інструментів для компіляції та завантаження коду на плату Arduino, а також послідовний монітор для налагодження та взаємодії з платою.
- Крос-платформенна сумісність: Arduino IDE доступна для операційних систем Windows, macOS і Linux, що забезпечує широку сумісність і простоту використання на різних платформах.
- Бібліотеки коду: Arduino IDE постачається з великою колекцією бібліотек, які спрощують процес розробки. Ці бібліотеки містять заздалегідь написаний код для різних функцій і компонентів, що дозволяє швидко інтегрувати датчики, актуатори, дисплеї та інші модулі у проекти.
- Диспетчер плат: Arduino IDE включає в себе функцію Диспетчер плат, яка дозволяє встановлювати та керувати різними платами Arduino. Він підтримує широкий спектр Arduino-сумісних плат, включаючи популярні Arduino Uno, Arduino Nano та ESP32.

- Монітор послідовностей: Інструмент Serial Monitor в Arduino IDE дозволяє спілкуватися з платою Arduino через послідовний інтерфейс. Він відображає дані, надіслані з плати, і дозволяє надсилати команди або дані назад на плату для тестування та налагодження.
- Вбудований компілятор і завантажувач: Arduino IDE містить вбудований компілятор, який перетворює ваш код в машинозчитувані інструкції для плати Arduino. Він також надає інструмент завантаження, який переносить скомпільований код на плату, роблячи процес програмування безперешкодним.
- Спільнота та екосистема: IDE Arduino має велику спільноту користувачів та розробників, що підтримує її. Спільнота Arduino активно ділиться бібліотеками, прикладами та проектами, а для підтримки та співпраці доступні численні онлайн-ресурси, навчальні посібники та форуми.
- Відкритий вихідний код та можливість розширення: Arduino IDE - це проект з відкритим вихідним кодом, що означає, що його вихідний код є вільно доступним для модифікації та вдосконалення. Ця відкритість призвела до розробки альтернативних IDE і плагінів, які покращують функціональність і розширюють можливості розробки Arduino.

1.3 BLE технологія

Bluetooth Low Energy (BLE) - це технологія бездротового зв'язку, розроблена для низького енергоспоживання та зв'язку на малих відстанях. Вона працює в діапазоні 2,4 ГГц ISM і дозволяє пристроям передавати дані на невеликі відстані, споживаючи при цьому мінімальну кількість енергії. BLE широко використовується в різних сферах, включаючи пристрої Інтернету речей, носимі пристрої, охорону здоров'я, домашню автоматизацію тощо.

BLE працює за архітектурою master/slave, де один пристрій є центральним (master), а інші пристрої - периферійними (slave). Головний пристрій ініціює та контролює зв'язок, а підлеглі відповідають на запити головного пристрою. Така архітектура дозволяє ефективно обмінюватися даними та керувати живленням.

Щоб встановити з'єднання, центральний пристрій сканує периферійні пристрої, що знаходяться поблизу, і знаходить їхні сервіси та характеристики. Знайшовши периферійний пристрій, центральний пристрій ініціює запит на з'єднання, а периферійний пристрій може його прийняти або відхилити. Після встановлення з'єднання пристрої можуть обмінюватися даними.

BLE використовує протокол Generic Attribute Profile (GATT) для визначення структури та поведінки даних, якими обмінюються пристрої. GATT організовує дані в ієрархічну структуру з послугами та характеристиками. Сервіси представляють собою набір пов'язаних даних і поведінки, тоді як характеристики представляють конкретні значення даних або дії.

Кожна характеристика має властивості, такі як читання, запис, сповіщення або індикація, які визначають, як можна отримати доступ до даних. Характеристики також можуть мати дескриптори, які надають додаткову інформацію або параметри конфігурації. Пристрої можуть читати, записувати або підписуватися на сповіщення від певних характеристик для обміну даними.

BLE включає механізми енергозбереження для оптимізації споживання енергії. Пристрої можуть переходити в різні режими енергоспоживання, зокрема активний, сплячий та рекламний. Рекламний стан дозволяє пристроям транслювати свою присутність, тоді як сплячий стан зменшує енергоспоживання, коли активний зв'язок не потрібен.

BLE включає в себе функції безпеки для захисту даних і забезпечення безпечного зв'язку. Він підтримує механізми шифрування, автентифікації та авторизації, щоб запобігти несанкціонованому доступу або підробці даних. Заходи безпеки можуть бути реалізовані на канальному рівні або через протоколи вищого рівня.

BLE визначає стандартизовані профілі та сервіси, які визначають, як пристрої можуть взаємодіяти в певних сферах застосування. Профілі визначають набір правил і поведінки для конкретних випадків використання, таких як моніторинг серцевого ритму або визначення наближення. Сервіси представляють конкретну функціональність, яку пропонує пристрій, і можуть бути стандартизованими або призначеними для користувача.

BLE часто називають Bluetooth Smart, оскільки він оптимізований для низького енергоспоживання. Він відрізняється від Bluetooth Classic, який розрахований на вищу швидкість передачі даних і більшу дальність зв'язку. Bluetooth Classic зазвичай використовується для потокового аудіо та інших додатків, що вимагають більшої пропускної здатності.

BLE використовує модель клієнт-сервер для обміну даними. Центральний пристрій (клієнт) ініціює запити на читання або запис даних з периферійного пристрою (сервера). Дані організовані в характеристики, які можуть містити такі значення, як показання датчиків, команди управління або інформацію про стан. Клієнти можуть виявляти та взаємодіяти з конкретними характеристиками на основі своїх унікальних UUID (універсально унікальних ідентифікаторів).

Профілі Bluetooth визначають конкретні випадки використання та фреймворки додатків для різних типів пристроїв. Профілі описують, як пристрої повинні взаємодіяти та обмінюватися даними в певних сценаріях. Прикладами профілів є Health Device Profile (HDP) для медичних пристроїв та Environmental Sensing Profile (ESP) для датчиків навколишнього середовища, заснований на GATT. Сервіси, з іншого боку, надають спосіб організувати та класифікувати певну функціональність пристрою.

На додаток до зв'язку "точка-точка", BLE також підтримує mesh-мережі. Mesh-мережа Bluetooth дозволяє декільком пристроям формувати мережу, в якій повідомлення можуть передаватися через проміжні пристрої для збільшення радіусу дії. Це забезпечує масштабоване розгортання в таких додатках, як розумне освітлення, автоматизація будівель і відстеження активів.

Низьке енергоспоживання, зв'язок на короткі відстані та стандартизовані протоколи зробили BLE популярним вибором для численних додатків Інтернету речей. Гнучкість, простота використання та широка сумісність з різними платформами сприяли його широкому впровадженню в різних галузях і сферах.

РОЗДІЛ 2
АПАРАТНО-ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ ПРИСТРОЮ

2.1 Побудова принципової електричної схеми пристрою

На рисунку 3.2 зображена схема електрична принципова, яка складається з плати ESP32 та динаміку.

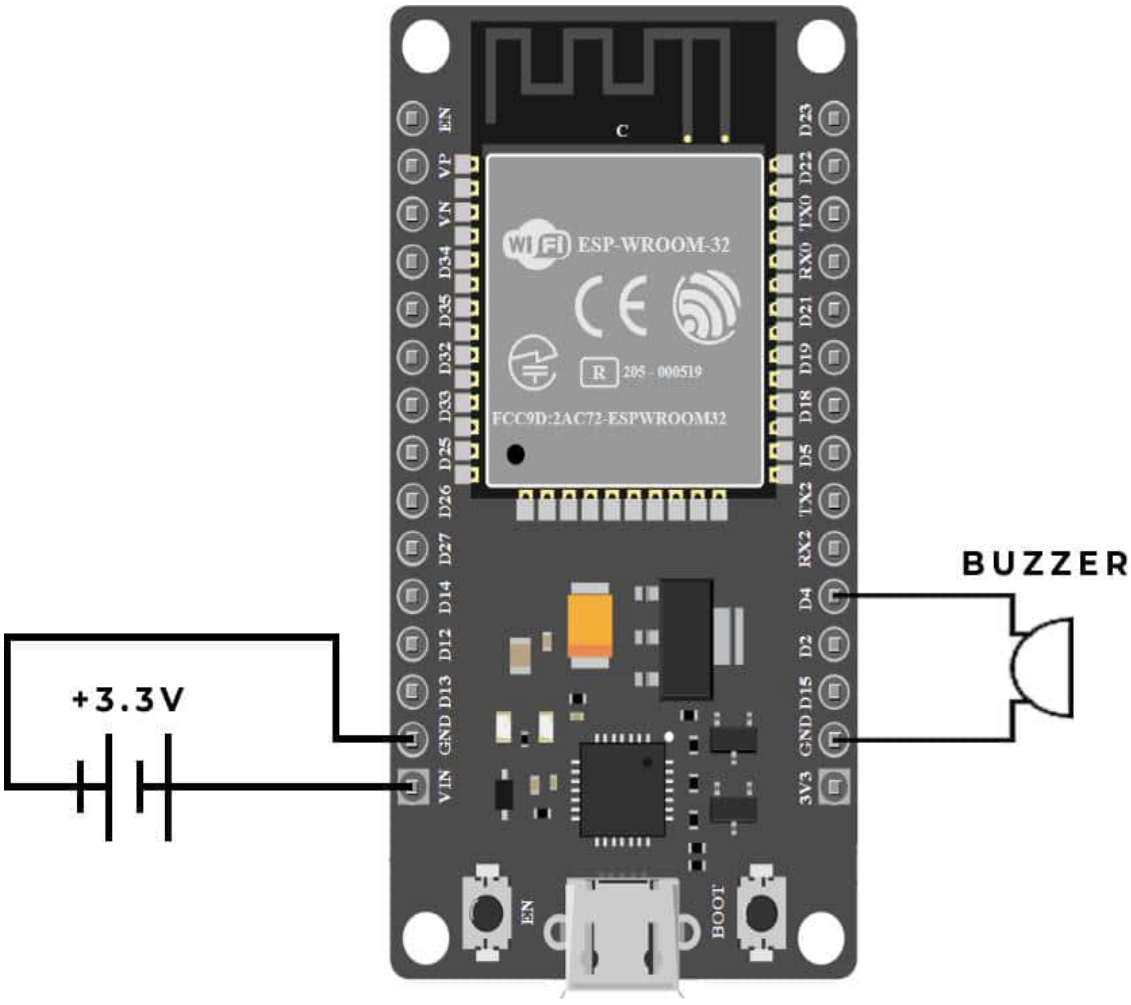

Рисунок 2.1 — Схема електрична принципова

2.2 Програмування контролера

Для програмування контролера було обрано мову С - це процедурна мова комп'ютерного програмування загального призначення, що підтримує структуроване програмування, лексичну область змінних та рекурсію із системою статичного типу. За задумом С пропонує конструкції, які ефективно відображають типові машинні інструкції. Він знайшов тривале використання в додатках, кодованих раніше мовою асемблера. Такі додатки включають операційні системи та різне прикладне програмне забезпечення для комп'ютерних архітектур, які варіюються від суперкомп'ютерів до ПЛК та вбудованих систем.

Для початку роботи було завантажено та інстальовано середовище розробки Arduino IDE.

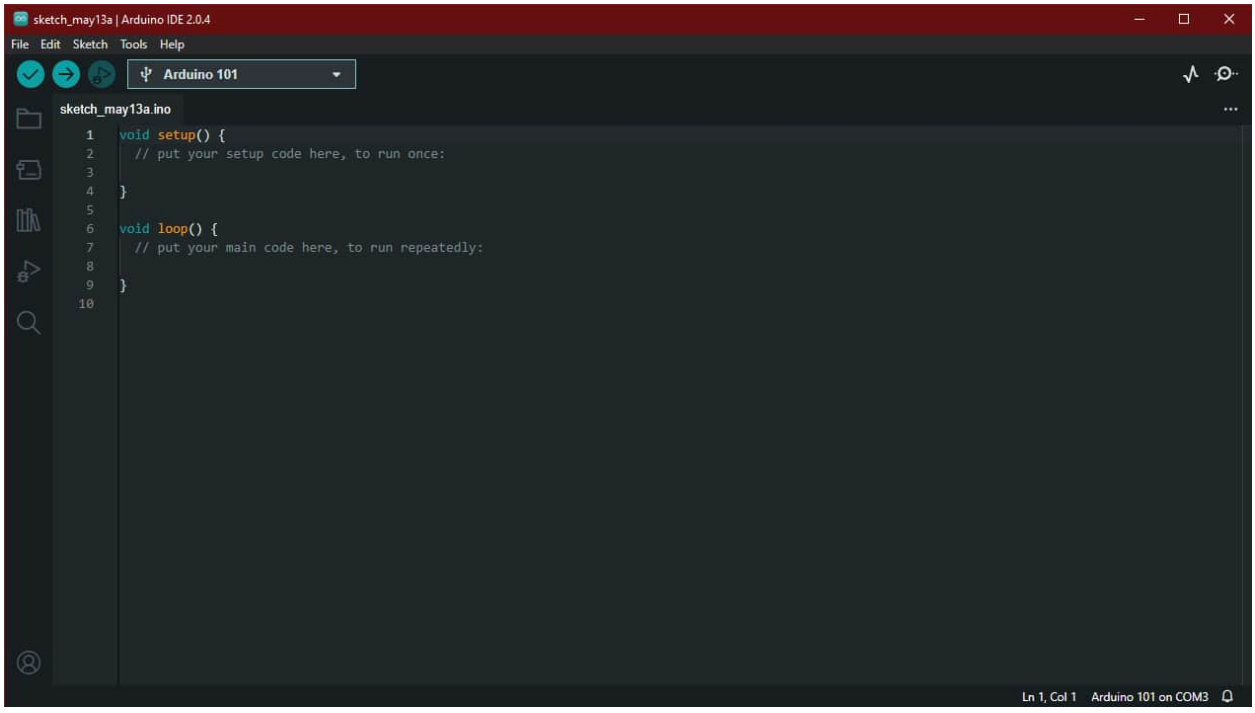

Рисунок 2.2.1 — Початкове вікно середовища розробки

На рисунку 2.2.1 зображено початкове вікно щойно встановленого середовища розробки.

Незважаючи на те, що Arduino IDE підтримує розробку на мікроконтролерах ESP32, спочатку необхідно додати їхню підтримку у вікні ‘Preferences’:

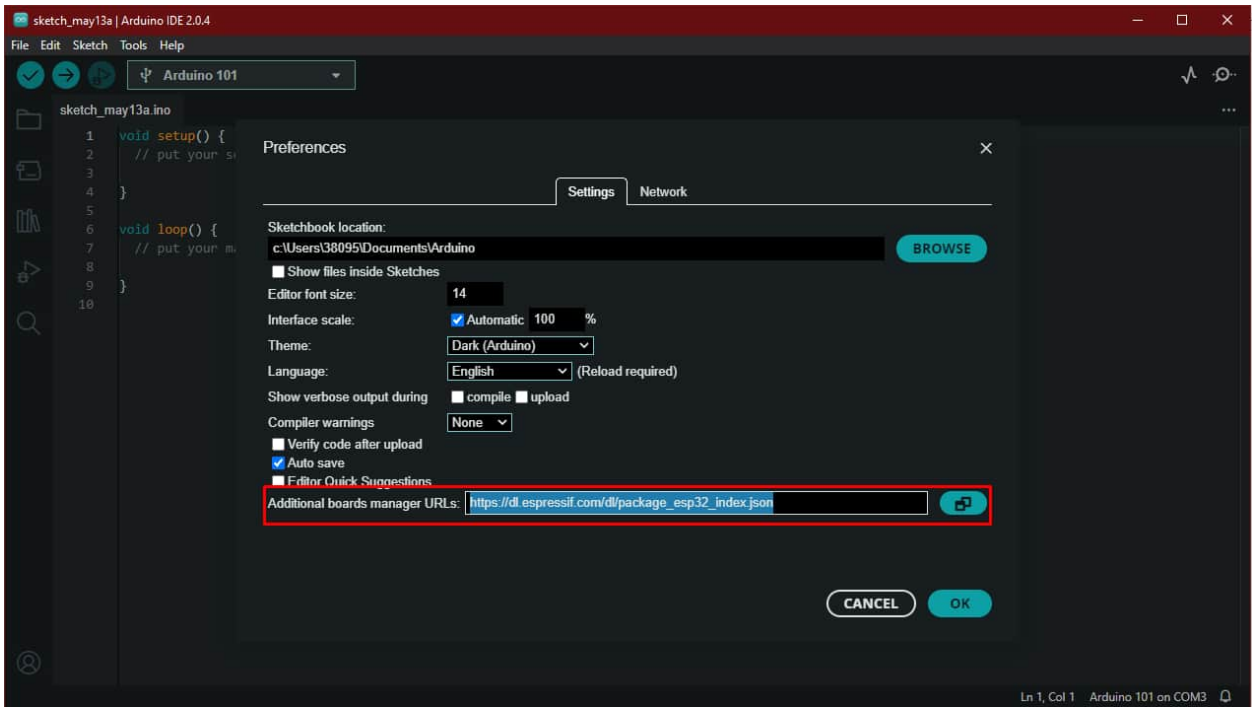

Рисунок 2.2.2 — вікно ‘Preferences’

На рисунку 2.2.2 була додана підтримка мікроконтролерів сімейства ESP32 до диспетчера плат.

Наступним кроком була завантажена підтримка мікроконтролерів сімейства ESP32 і необхідних бібліотек для робот з ними через диспетчер плат:

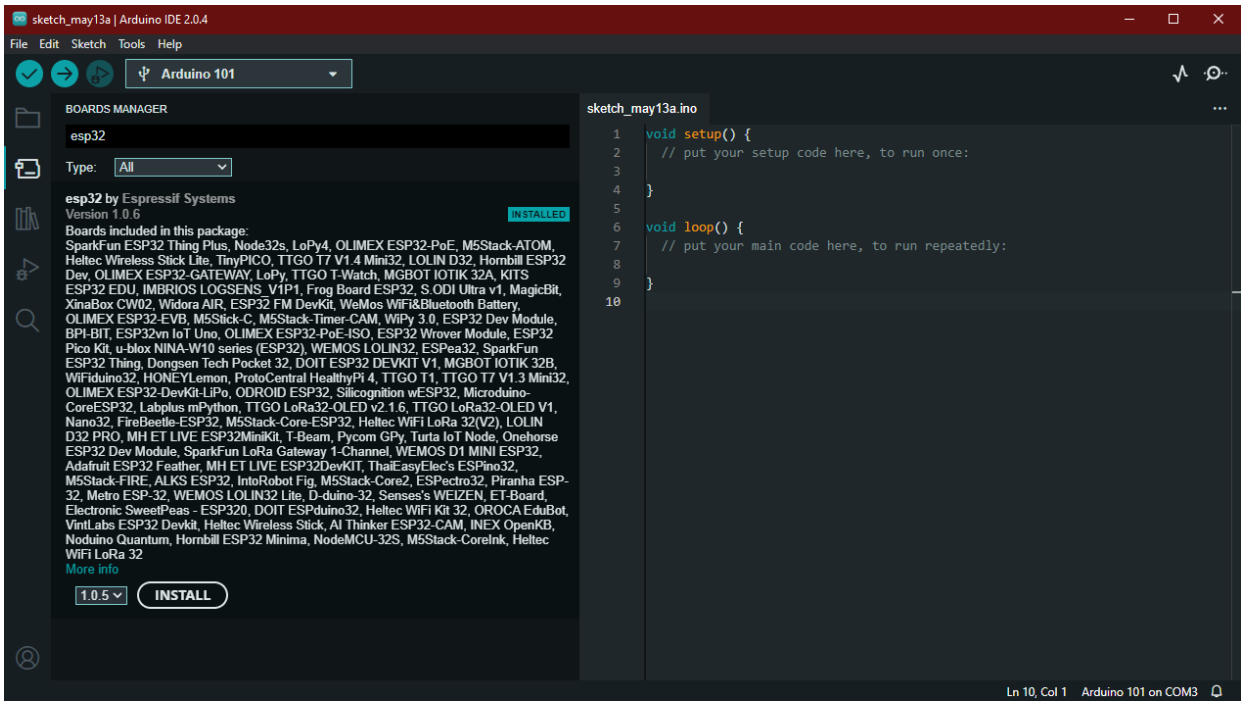

Рисунок 2.2.3 — завантаження необхідних інструментів та бібліотек

Останнім кроком для початку роботи було створити новий скетч (файл з програмою), а також обрати необхідний пристрій і порт, через який програма буде завантажуватись з комп’ютера на мікроконтролер – в нашому випадку пристроєм виступає ‘ESP32 Dev Module’:

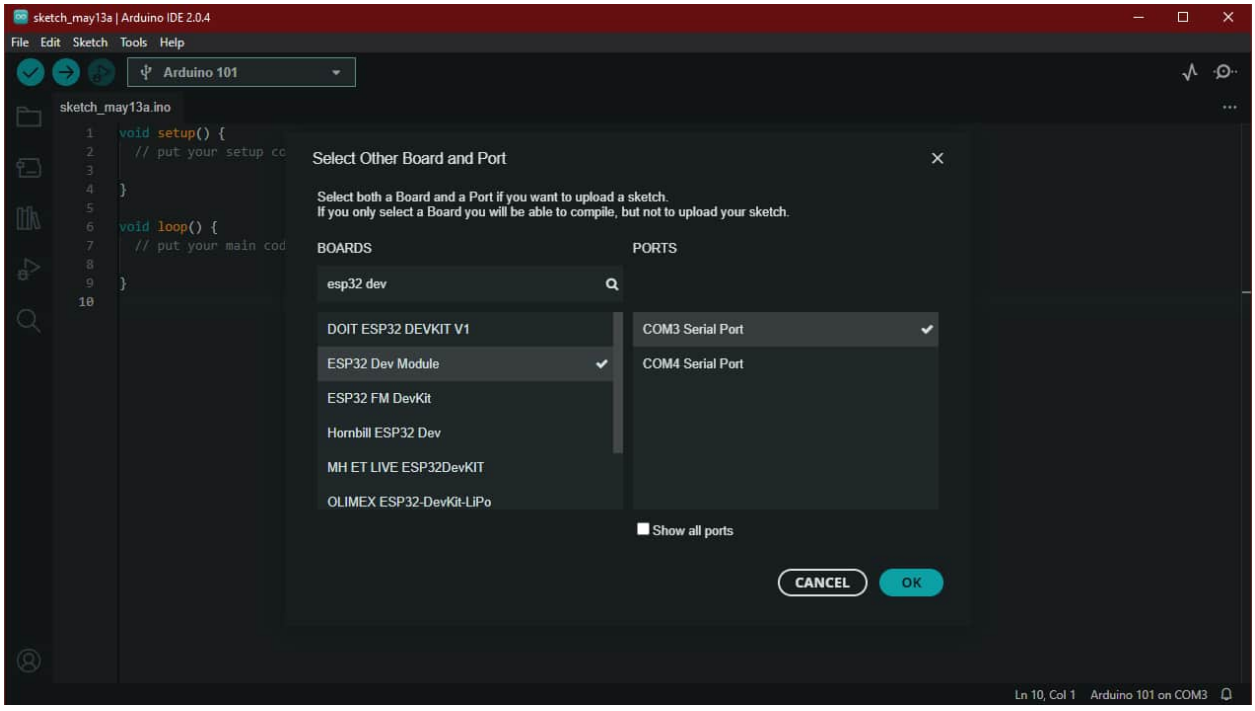

Рисунок 2.2.4 — вікно вибору пристрою і вихідного порту

2.3 Вибір засобів подачі команди для контролера

Оскільки для розробки пристрою була обрана технологія BLE, для подачі сигналу контролеру необхідний пристрій з підтримкою технології Bluetooth.

В якості пристрою для подачі сигналу було обрано смартфон, на який було завантажено додаток nRF Connect, який підтримує технологію BLE.

nRF Connect - це мобільний додаток, розроблений компанією Nordic Semiconductor, який надає комплексний набір інструментів для розробки та тестування Bluetooth Low Energy (BLE). Він доступний як для пристроїв iOS, так і для Android і пропонує цілий ряд функцій і можливостей, які допоможуть вам у вирішенні завдань, пов'язаних з BLE.

Ось деякі ключові можливості програми nRF Connect:

- Сканування пристроїв: nRF Connect дозволяє сканувати пристрої BLE, що знаходяться поблизу вас. Він відображає детальну інформацію про виявлені пристрої, включаючи їхні назви, послуги, характеристики та рівень сигналу. Ця функція допомагає ідентифікувати та підключитися до BLE-пристроїв для подальшого дослідження та взаємодії.
- Керування з'єднанням: Після виявлення BLE-пристрою nRF Connect дозволяє встановити з ним з'єднання. Ви можете встановити з'єднання як з центральними (наприклад, смартфонами), так і з периферійними (наприклад, датчиками) пристроями. Це дозволяє спілкуватися з підключеним пристроєм і керувати ним, зчитувати або записувати характеристики, а також відстежувати сповіщення.
- Дослідження послуг та характеристик: nRF Connect надає зручний інтерфейс для дослідження послуг та характеристик, що пропонуються підключеним BLE-пристроєм. Він дозволяє переглядати ієрархічну структуру послуг і отримувати доступ до даних і властивостей, пов'язаних з кожною характеристикою. Ця функція корисна для розуміння можливостей пристрою та доступних потоків даних.
- Візуалізація та реєстрація даних: Додаток пропонує вбудовані інструменти для візуалізації та реєстрації даних, якими обмінюються між додатком і підключеним BLE-пристроєм. Ви можете переглядати оновлення даних у реальному часі, будувати графіки та відстежувати показники датчиків. Це допомагає відстежувати та аналізувати потік даних, полегшуючи налагодження та усунення несправностей BLE-зв'язку.
- Оновлення прошивки по повітрю (OTA): nRF Connect підтримує оновлення прошивки по повітрю для пристроїв, в яких реалізовано OTA-завантажувач від Nordic Semiconductor. Це дозволяє бездротово оновлювати прошивку на сумісних пристроях, усуваючи необхідність фізичного з'єднання і забезпечуючи зручне та ефективне обслуговування та оновлення пристроїв.

- Конфігурація Bluetooth GATT: nRF Connect пропонує функцію для налаштування послуг і характеристик BLE-пристрою. Вона надає візуальний інтерфейс для визначення структури та властивостей бази даних GATT (Generic Attribute Profile) на пристрої. Ця функція особливо корисна на етапі розробки для налаштування бажаної поведінки та функціональності пристрою.
- Налаштування mesh-мереж Bluetooth: nRF Connect підтримує налаштування та конфігурацію пристроїв у mesh-мережах Bluetooth. Це дозволяє легко додавати пристрої в mesh-мережі та керувати ними, призначати адреси, а також визначати їхні ролі та можливості. Ця функція спрощує налаштування та керування mesh-мережами Bluetooth для таких застосувань, як розумне освітлення.
- Інструменти та бібліотеки для розробників: nRF Connect надає доступ до ряду інструментів та бібліотек для розробників, що пропонуються компанією Nordic Semiconductor. Ці інструменти включають nRF Toolbox, nRF Logger та інші, які допомагають розробникам у різних аспектах розробки, налагодження та тестування BLE-додатків.

РОЗДІЛ 3
ТЕСТУВАННЯ ПРИСТРОЮ ДЛЯ ПОШУКУ КЛЮЧІВ

3.1 Виготовлення прототипу за заданою електричною схемою

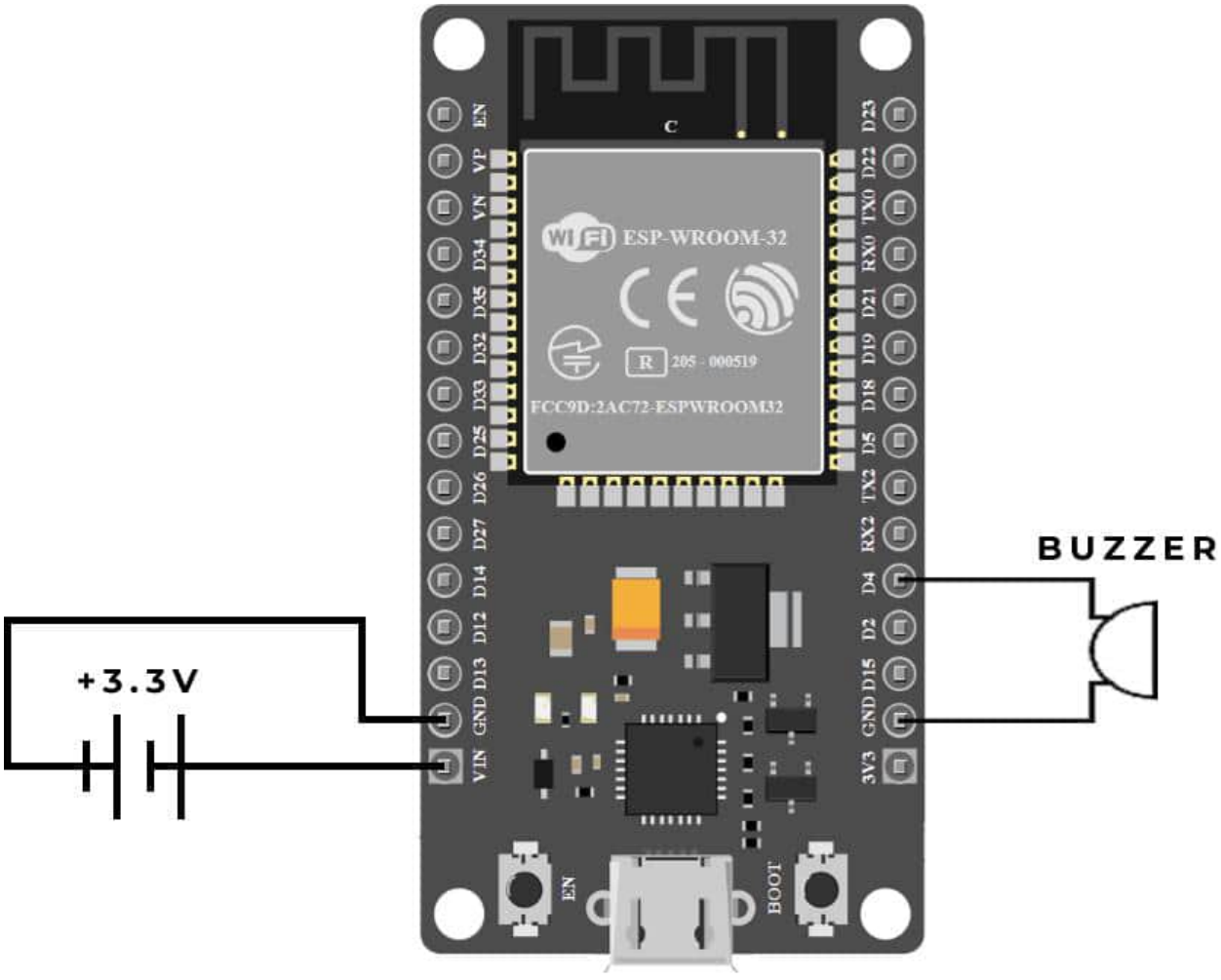

Рисунок 3.1 — Схема електрична принципова

За заданою електричною схемою до мікроконтролера було підключено пасивний динамік до:

Пін D4 – сторона +

Пін GND – сторона – (заземлення)

Таож до мікроконтролера було підключено стандартну батарейку формату AA:

Пін VIN – сторона +

Пін GND – сторона – (заземлення)

3.2 Прошивка і тестування пристрою

Задля прошивки пристрою необхідно підключити його до комп’ютера, на якому встановлено середовище Arduino IDE, написати код програми, обрати необхідний пристрій і вихідний USB порт:

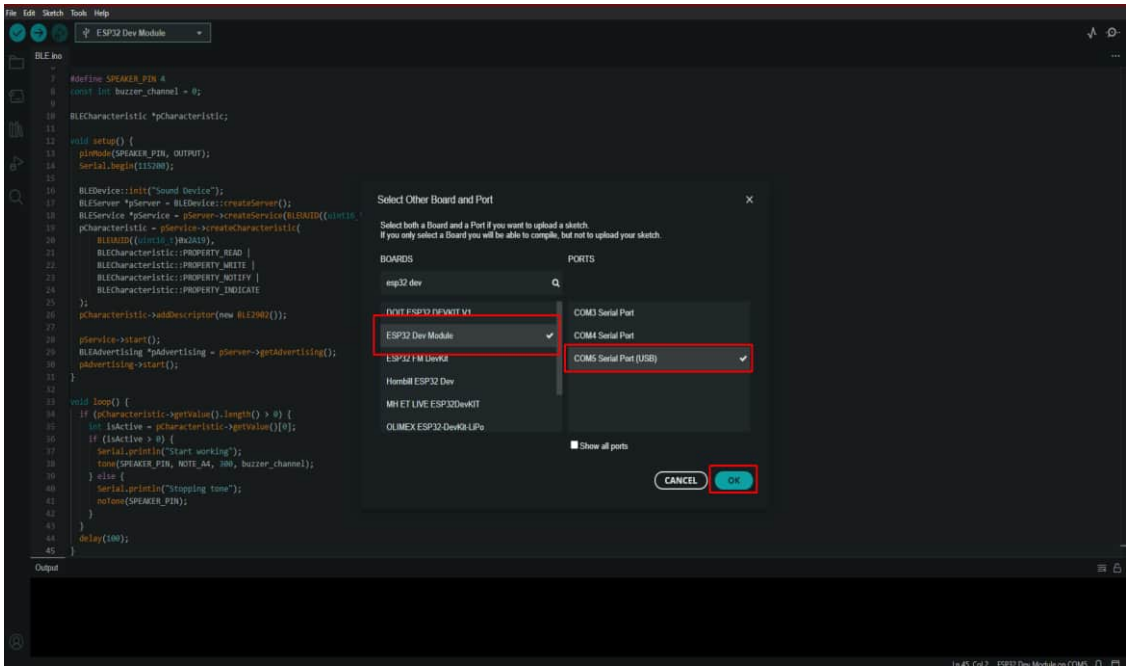

Рисунок 3.2.1

На рисунку 3.2.1 зображено вибір необхідного пристрою і вихідного порту.

Наступним кроком необхідно завантажити код на мікроконтролер за допомогою кнопки ‘Upload’:

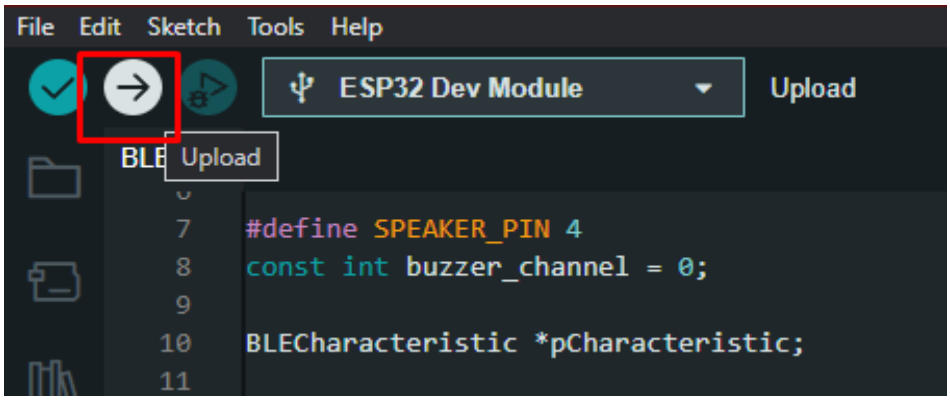

Рисунок 3.2.2 - Кнопка 'Upload'

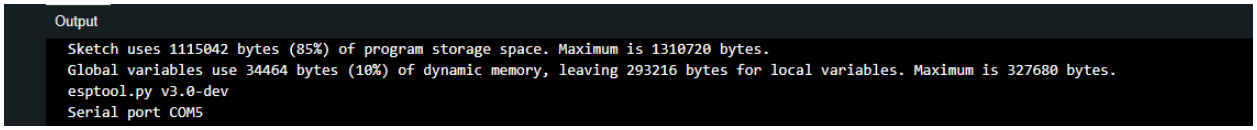

Рисунок 3.2.3

На рисунку 3.2.3 зображено, як програма після перевірки та компіляції коду обраховує необхідну кількість пам’яті для програми, та звіряється чи мікроконтролер має достатньо пам’яті для завантаження програми на нього.

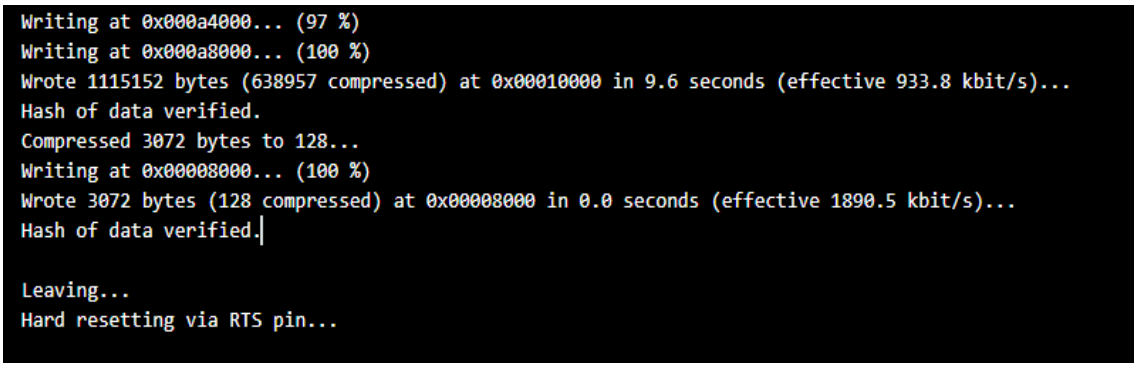

Рисунок 3.2.4

Після усіх перевірок, Arduino IDE завантажує код програми на мікроконтролер (Рисунок 2.3.4)

3.3 Перевірка роботи пристрою

Для перевірки роботи пристрою, як згадувалось раніше, нам знадобиться мобільний пристрій з підтримкою технології Bluetooth і встановленою програмою nRF Connect.

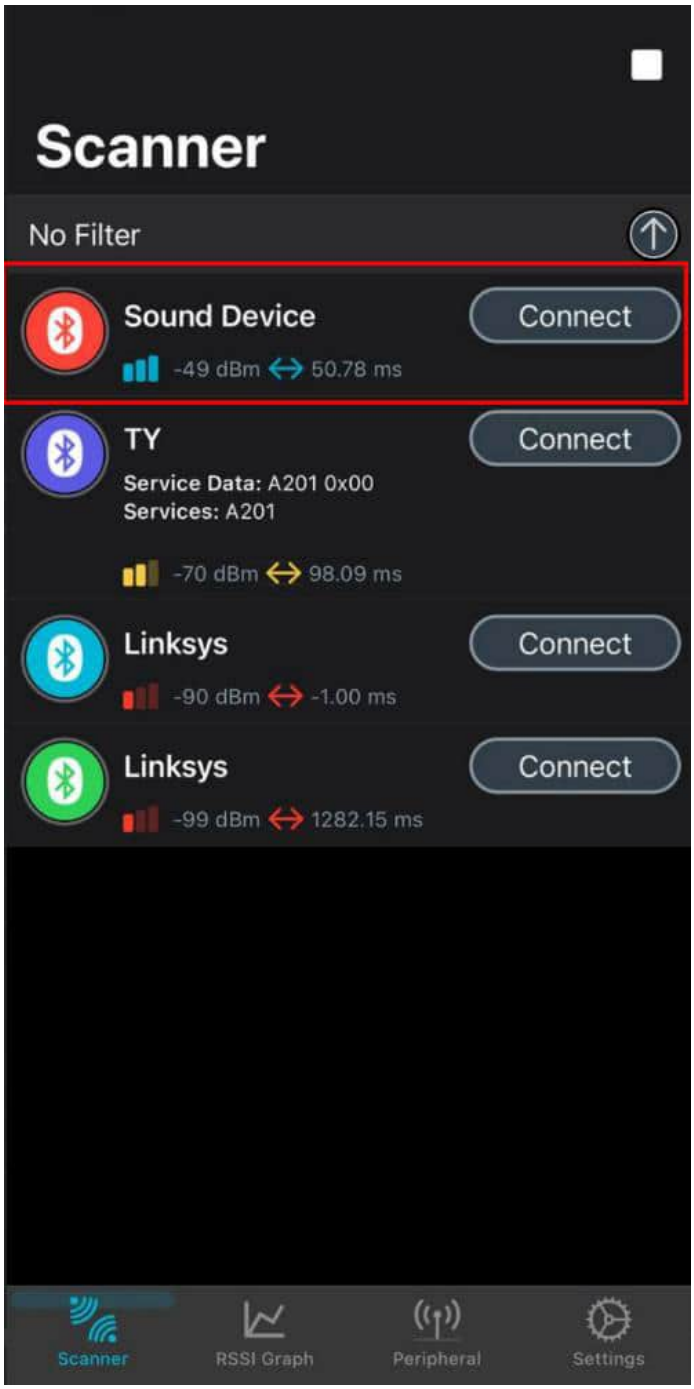

Рисунок 3.3.1

На рисунку 3.3.1 зображене головне меню програми nRF Connect. Необхідний нам пристрій має назву ‘Sound Device’. Підключитись до нього можливо за допомогою кнопки ‘Connect’. Після цього ми потрапляєм у меню керування пристроєм (Рисунок 3.3.2)

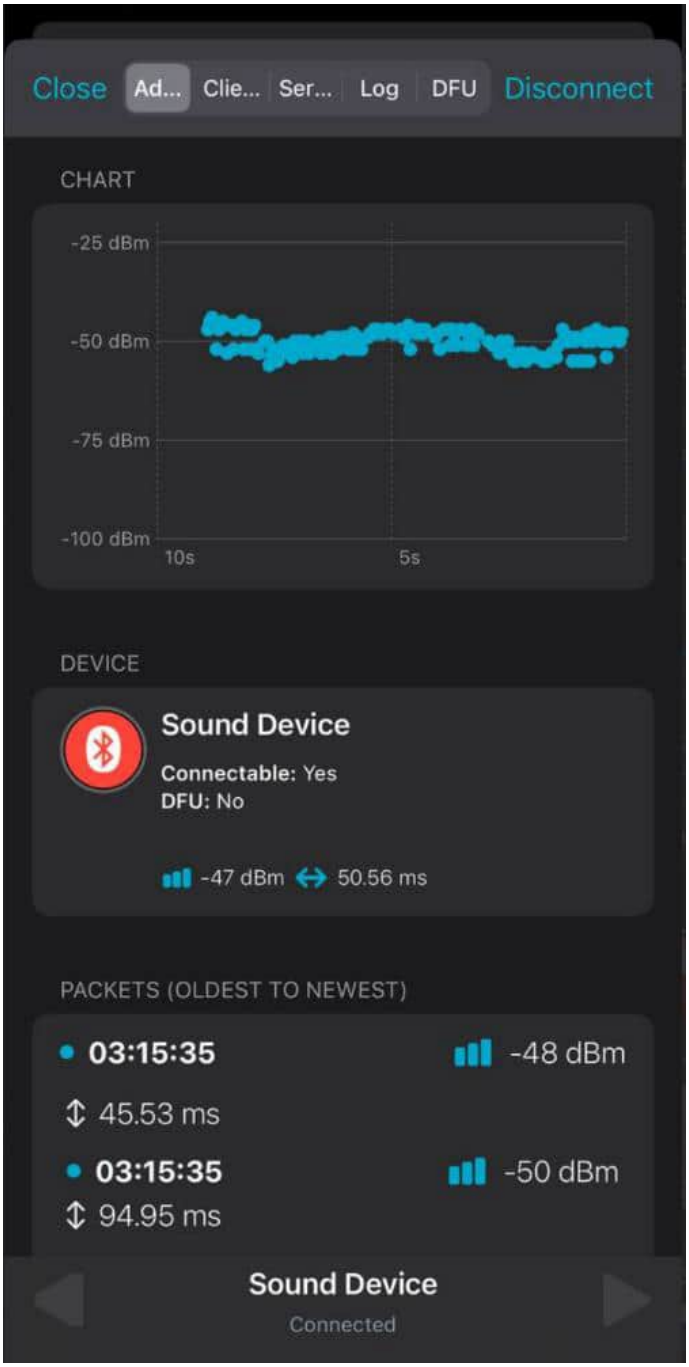

Рисунок 3.3.2

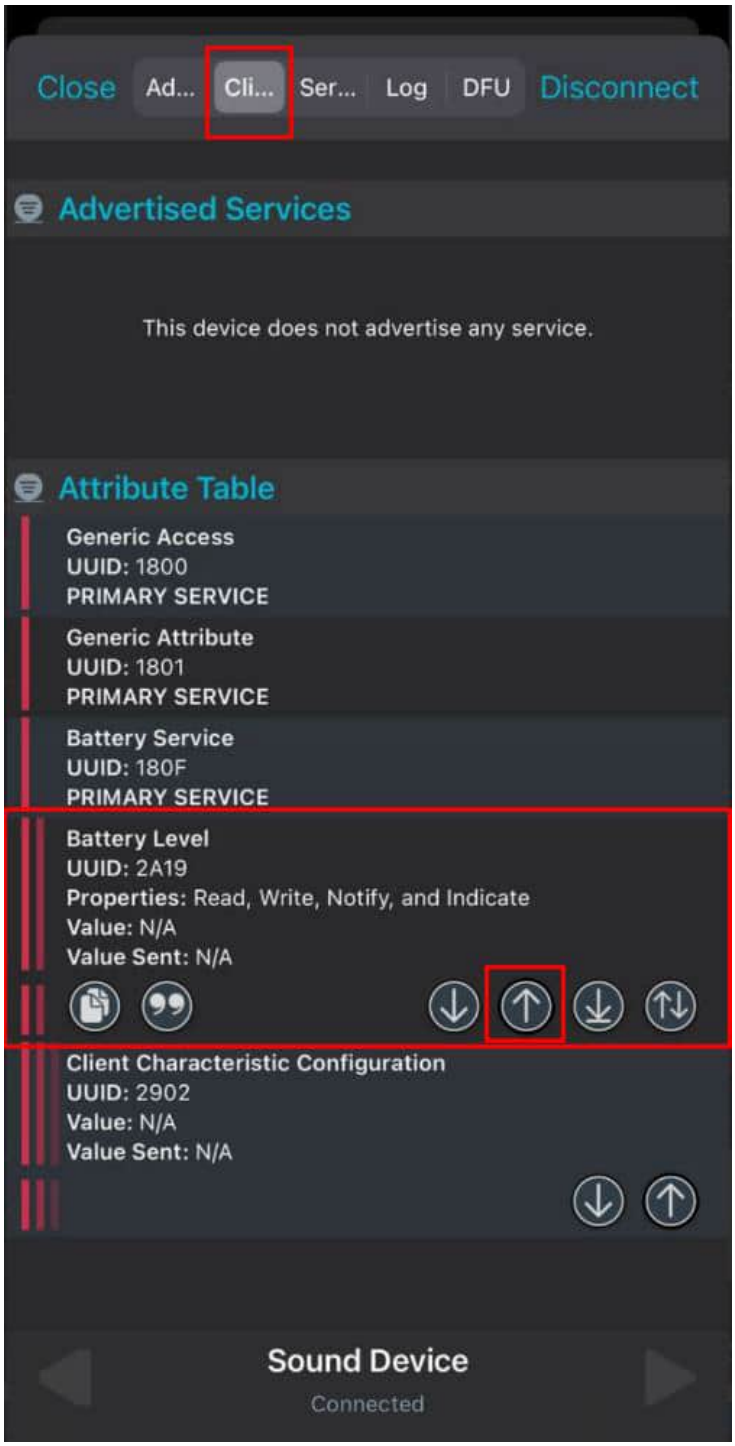

Рисунок 3.3.3

Нас цікавить вкладка ‘Client’ і таблиця атрибутів, за допомогою якої ми можемо подавати команди мікроконтролеру (рисунок 3.3.3)

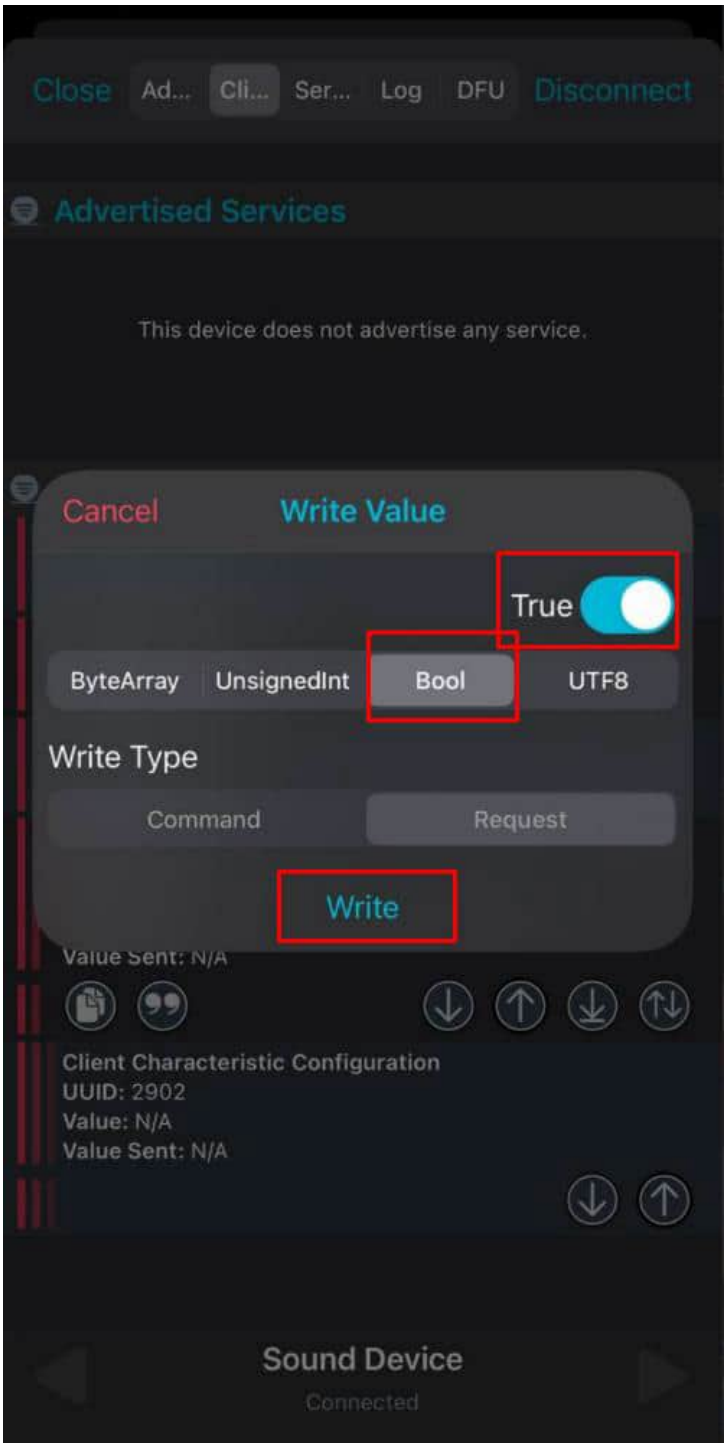

Рисунок 3.3.4

Контролер подає звуковий сигнал, коли отримує команду True (логічну одиницю). Для відправки такої команди нам необхідно обрати тип даних ‘Bool’, обрати опцію ‘True’ або ‘False’, і відправити команду за допомогою кнопки ‘Write’ (рисунок 3.3.4).

3.4 Аналіз енергоспоживання пристрою

Більшість часу своєї роботи пристрій перебуває в режимі ‘глибокого сну’, що помітно зменшує його енергоспоживання і збільшує час його роботи без підзарядки.

Щоб розрахувати приблизну енергоефективність, нам потрібно врахувати енергоспоживання пристрою в різних режимах і тривалість кожного режиму.

Припустимо наступні значення енергоспоживання пристрою в різних станах:

Активний режим: 240 мА

Режим глибокого сну: 10 мкА

І припустимо, що ESP32 проводить 10% часу в активному режимі (виконуючи завдання) і 90% часу в режимі глибокого сну (простоюючи).

Спочатку розрахуємо енергоспоживання в кожному з цих режимів:

Енергоспоживання в активному режимі:

240 мА * 3,7 В * (10% * 30 секунд) = 2,16 мВт-год

Енергоспоживання в режимі глибокого сну:

10 мкА * 3,7 В * (90% * 30 секунд) = 0,009 мВт-год

Загальне споживання енергії за цикл:

2,16 мВт-год + 0,009 мВт-год = 2,169 мВт-год

ВИСНОВКИ

В процесі виконання проекту було створено повноцінну апаратно-програмну систему брелоку для пошуку ключів. В склад системи входять динамік та мікроконтролер ESP32. Система реалізована на базі контролера ESP32 та спроектована таким чином, щоб її можна було в будь-який момент переробити або відремонтувати. На даному етапі система повністю функціональна та готова до використання в реальних умовах.

Окрім цього, в процесі виконання роботи було отримано неоцінимий досвід роботи з мікроконтролерами, датчиками та електронними ланцюгами, було отримано нові і закріплено старі знання в сфері програмування в середовищі Arduino IDE.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Shirriff, Ken (30 August 2016). "The Surprising Story of the First Microprocessors". IEEE Spectrum. Institute of Electrical and Electronics Engineers. 53 (9): 48–54. doi:10.1109/MSPEC.2016.7551353. S2CID 32003640. Retrieved 13 October 2019.

2. "1960: Metal Oxide Semiconductor (MOS) Transistor Demonstrated". The Silicon Engine: A Timeline of Semiconductors in Computers. Computer History Museum. Retrieved August 31, 2019.

3. Moskowitz, Sanford L. (2016). Advanced Materials Innovation: Managing Global Technology in the 21st century. John Wiley & Sons. pp. 165–167. ISBN 9780470508923.

4. "1971: Microprocessor Integrates CPU Function onto a Single Chip". The Silicon Engine. Computer History Museum. Retrieved 22 July 2019.

5. Augarten, Stan (1983). The Most Widely Used Computer on a Chip: The TMS 1000. State of the Art: A Photographic History of the Integrated Circuit. New Haven and New York: Ticknor & Fields. ISBN 978-0-89919-195-9. Retrieved 2009-12-23.

6. "Trends in the Semiconductor Industry". Semiconductor History Museum of Japan. Archived from the original on 2019-06-27. Retrieved 2019-06-27.

7. "Oral History Panel on the Development and Promotion of the Intel 8048 Microcontroller" (PDF). Computer History Museum Oral History, 2008. p. 4. Retrieved 2016-04-04.

8. "Chip Hall of Fame: Microchip Technology PIC 16C84 Microcontroller". IEEE. 2017-06-30. Retrieved September 16, 2018.

9. Motorola. Advance Information, 8-Bit Microcomputers MC68HC05B6, MC68HC05B4, MC68HC805B6, Motorola Document EADI0054RI. Motorola Ltd., 1988.

10. "Atmel's Self-Programming Flash Microcontrollers" (PDF). 2012-01-24. Retrieved 2008-10-25. by Odd Jostein Svendsli 2003

11. Turley, Jim (2002). "The Two Percent Solution". Embedded. Retrieved 2018-07-11.

12. Cantrell, Tom (1998). "Microchip on the March". Circuit Cellar. Archived from the original on 2007-09-27. Retrieved 2018-07-11.

13. "Semico Research".

14. "Momentum Carries MCUs Into 2011 | Semico Research". semico.com. Retrieved 2018-07-11.

15. "MCU Market on Migration Path to 32-bit and ARM-based Devices". April 25, 2013. It typically takes a global economic recession to upset the diverse MCU marketplace, and that’s exactly what occurred in 2009, when the microcontroller business suffered its worst-ever annual sales decline of 22% to $11.1 billion.

16. "The really low cost MCUs". www.additude.se. Retrieved 2019-01-16.

17. Bill Giovino. "Zilog Buys Microcontroller Product Lines from Samsung". 2013.

18. "EFM8BB10F2G-A-QFN20 Silicon Labs | Mouser".

19. "MSP430G2001IPW14R Texas Instruments | Mouser".

20. "CY8C4013SXI-400 Cypress Semiconductor | Mouser". Mouser Electronics. Archived from the original on 2015-02-18.

21. "MSP430FR2000IPW16R Texas Instruments | Mouser".

22. "CY8C4013SXI-400 Cypress Semiconductor | Mouser". Mouser Electronics. Retrieved 2018-07-11.

23. https://eu.mouser.com/ProductDetail/Silicon-Labs/EFM32ZG108F8-QFN24?qs=sGAEpiMZZMuI9neUTtPr75mJ%2fJmU8iJshd%2f59xMDhYo%3d

24. U-M researchers create world's smallest 'computer', University of Michigan, 2018-06-21

25. University of Michigan outdoes IBM with world's smallest 'computer', CNET, 2018-06-22

26. IBM fighting counterfeiters with world's smallest computer, CNET, 2018-03-19

27. IBM Built a Computer the Size of a Grain of Salt. Here's What It's For., Fortune, 2018-03-19

28. Heath, Steve (2003). Embedded systems design. EDN series for design engineers (2 ed.). Newnes. pp. 11–12. ISBN 9780750655460.

29. David Harris & Sarah Harris (2012). Digital Design and Computer Architecture, Second Edition, p. 515. Morgan Kaufmann. ISBN 0123944244.

30. Easy Way to build a microcontroller project

31. Mazzei, Daniele; Montelisciani, Gabriele; Baldi, Giacomo; Fantoni, Gualtiero (2015). Changing the programming paradigm for the embedded in the IoT domain. Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on. Milan: IEEE. pp. 239–244. doi:10.1109/WF-IoT.2015.7389059.

32. "8052-Basic Microcontrollers" by Jan Axelson 1994

33. Edwards, Robert (1987). "Optimizing the Zilog Z8 Forth Microcontroller for Rapid Prototyping" (PDF). Martin Marietta: 3. Retrieved 9 December 2012.

34. www.infineon.com/mcu

ДОДАТОК (Код програми)

```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
#include <Tone32.h>

#define SPEAKER_PIN 4
const int buzzer_channel = 0;

BLECharacteristic *pCharacteristic;

void setup() {
  pinMode(SPEAKER_PIN, OUTPUT);
  Serial.begin(115200);

  BLEDevice::init("Sound Device");
  BLEServer *pServer = BLEDevice::createServer();
                    BLEService *pService = pServer->createService(BLEUUID((uint16_t)0x180F));
  pCharacteristic = pService->createCharacteristic(
    BLEUUID((uint16_t)0x2A19),
    BLECharacteristic::PROPERTY_READ |
    BLECharacteristic::PROPERTY_WRITE |
    BLECharacteristic::PROPERTY_NOTIFY |
    BLECharacteristic::PROPERTY_INDICATE
  );
```

```
  pCharacteristic->addDescriptor(new BLE2902());

  pService->start();
  BLEAdvertising *pAdvertising = pServer->getAdvertising();
  pAdvertising->start();
}

void loop() {
  if (pCharacteristic->getValue().length() > 0) {
    int isActive = pCharacteristic->getValue()[0];
    if (isActive > 0) {
      Serial.println("Start working");
      tone(SPEAKER_PIN, NOTE_A4, 300, buzzer_channel);
    } else {
      Serial.println("Stopping tone");
      noTone(SPEAKER_PIN);
    }
  }
  isConnected = BLEDevice::getAdvertising()->isConnected();
  if (!BLEDevice::getScan()->isScanning() && !isConnected) {
    Serial.println("Entering deep sleep...");
    esp_sleep_enable_timer_wakeup(30000000);
    esp_deep_sleep_start();
  }}
```