Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та кібербезпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

СИСТЕМА СИМУЛЯЦІЇ КЕРУВАННЯ АВТОМОБІЛЕМ НА БАЗІ МІКРОКОНТРОЛЕРА

MICROCONTROLLER-BASED CAR DRIVING SIMULATION SYSTEM

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-41
Михальчук Микола Анатолійович

(підпис)

Керівник:
к.т.н., доцент
Костючко Сергій Миколайович

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 11 » червня 2024 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та кібербезпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Н.Чернящук
« 10 » 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Михальчуку Миколі Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Система симуляції керування автомобілем на базі мікроконтролера*

Керівник роботи *к.т.н., доцент Костючко Сергій Миколайович*

затверджені наказом закладу вищої освіти від «30» грудня 2023 року № 459/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 11.06.2024р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області та різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз проблеми за темою роботи та постановка завдань дослідження (система симуляції автомобілем на базі мікроконтролера)
Обгрунтування актуальності теми кваліфікаційної роботи
Вибір та огляд засобів розробки проекту кваліфікаційної роботи
Опис практичної реалізації проекту
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Існуючі рішення
Використані технології
Архітектура системи
Інтерфейс системи
Схема роботи програмного продукту

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Костючко С.М., доцент*		
Теоретичне дослідження та практична реалізація	*Костючко С.М., доцент*		
Практична реалізація об'єкта проектування	*Костючко С.М., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,асистент*		

7. Дата видачі завдання 10.01.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Розділ 1. Обгрунтування актуальності теми кваліфікаційної роботи*	до 15.02.2024 р.	Виконано
2.	*Розділ 2. Вибір та огляд засобів розробки проекту кваліфікаційної роботи*	до 15.03.2024 р.	Виконано
3.	*Розділ 3. Опис практичної реалізації проекту*	до 04.05.2024 р.	Виконано
4.	*Висновки та пропозиції*	до 07.05.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 10.05.2024 р.	Виконано
6.	*Формування додатків*	до 15.05.2024 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 20.05.2024 р.	Виконано
8.	*Нормоконтроль*	до 01.06.2024 р.	Виконано
9.	*Інструментальна перевірка на академічний плагіат*	до 04.06.2024 р	Виконано
10.	*Представлення кваліфікаційної роботи бакалавра до захисту*	до 11.06.2024 р.	Виконано

Здобувач вищої освіти _____ Михальчук М.А.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Костючко С.М.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Михальчук М.А. Система симуляції автомобілем на базі мікроконтролера. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2024. 58 с.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, додатків.

Перший розділ присвячено огляд та обґрунтування актуальності теми кваліфікаційної роботи.

В другому розділі здійснено вибір та огляд засобів розробки проекту кваліфікаційної роботи.

Третій розділ присвячено опису розробленої програми в середовищі arduino та розробка проекту базі мікроконтролера ESP8266.

Об'єкт дослідження – комп'ютерні системи та їх архітектура.

Предмет дослідження – мікроконтроллери.

Метою роботи є створення контролеру для автомобілів (панель приладів).

Ключові слова: ARDUINO, ESP8266, мікроконтролер, TFT, панель керування, ARDUINO IDE, NODE MCU v3.

ANNOTATION

Mikhalchuk M.A. Microcontroller-based car driving simulation system. Manuscript.

Bachelor's qualifying thesis of the OP «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2024. 58 p.

The qualification work consists of an introduction, three sections, conclusions, a list of used sources, and appendices.

The first section is devoted to the review and justification of the relevance of the topic of the qualification work.

In the second section, the selection and review of the means of developing the project of the qualification work was carried out.

The third chapter is devoted to the description of the developed program in the arduino environment and the development of the instrument panel project based on the ESP8266 microcontroller.

The object of research is computer systems and their architecture.

The subject of research is microcontrollers.

The purpose of the work is to create a controller for cars (instrument panel).

Keywords: ARDUINO, ESP8266, microcontroller, TFT, control panel, ARDUINO IDE, NODE MCU v3.

ЗМІСТ

ВСТУП

Актуальність теми. Безпосереднє керування автомобілем вимагає від водія постійного доступу до ключової інформації, а саме швидкість, кількість оборотів двигуна, залишок пального, пробіг, тощо на панелі приладів. Це своєрідний командний центр. Без цієї системи керувати автомобілем було б не тільки важко, а й небезпечно. Приладова панель автомобіля виконує багато важливих функцій, які часта надають водіям неоціненну допомогу.

Метою роботи є створення контролеру для автомобілів (панель приладів). Цей датчик може показувати швидкість, оберти двигуна, ввімкнуту передачу, індикатори напрямку, гальма, фари.

Об’єкт дослідження – комп’ютерні системи та їх архітектура.

Предмет дослідження – мікроконтроллери.

Завдання, які необхідно виконати:

- Реалізувати систему керування автомобілем на базі мікроконтроллера.
- Розробити панель приладів для автомобіля.
- Дослідити роботу контролера.
- Візуалізувати роботу датчиків швидкості, обертів двигуна, тощо.
- Спроектувати проект на базі плати розробки ESP8266.
- Запропонувати реалізацію схожих проектів.

Практичне значення одержаних результатів. Результати даної роботи дають інженерам можливість розробляти та тестувати різноманітні варіанти дизайну панелі приладів, також дозволяє виявляти помилки до початку масового виробництва та виправити потенційні помилки у роботі панелі приладів. Подібні симулятори навчають майбутніх водіїв для приспособлення до панелі приладів без можливих ризиків аварійних ситуацій.

РОЗДІЛ 1
ОБГРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1 Панель керування автомобілем

Приладова панель автомобіля (рисунок 1.1) – це панель керування, яка розташована на центральній консолі транспортного засобу. Зазвичай вона розташована безпосередньо перед водієм транспортного засобу. Відображає контрольно-вимірувальні прилади та елементи керування для роботи автомобіля.

Рисунок 1.1 – Панель керування автомобілем [1]

Складовими панелі приладів є:

– спідоментр для контролю швидкості пересування;
– контрольні лампи для відстеження роботи вузлів;
– лічильник для відображення контролю пробігу;
– показник температури двигуна;
– рівень палива;
– прилад вимірювач контролю обертів двигуна.

Приладова панель – це найшвидший і ефективний спосіб повідомити водієві про можливу несправність автомобіля. Тому водій завжди в курсі роботи кожного важливого механізму машини.

Перед початком руху водієві можна перевірити безліч індикаторів, просто оцінивши все, що показує приладова панель. Просто протестуйте систему перед початком щоденних операцій. Майте на увазі, що це не займе багато часу.

Згодом після включення запалювання багато сигнальних ламп на консолі загоряться різними кольорами. Як згадувалося вище, їх точний склад відрізняється в залежності від дизайнерських рішень і традицій конкретного виробника або моделі автомобіля. Незалежно від цього загориться наступний сигнал:

– індикація рівня тиску в масляній системі автомобіля;
– індикація ввімкненого ручного гальма;
– показання вольтметру [2].

Значки, перераховані вище, є найбільш поширеними. Але навіть в цьому випадку у Вашого автомобіля можуть бути свої унікальні індикатори. Вони також різняться через різний рік виготовлення машини, звичайно, новий автомобіль оснащений кількома новими системами, яких немає у старому автомобілі.

Кожен водій повинен дотримуватися наступного правила якщо панель нагадує ялинку і ретельно прикрашена різнокольоровими гірляндами, необхідно найближчим часом звернутися до фахівця, який повинен оглянути всі системи і вузли автомобіля. Самостійно визначити несправність може бути неефективно, тому в таких випадках краще негайно звернутися до фахівця.

Життя багатьох людей сьогодні тісно пов’язане з автомобілем, тому безпека автомобіля є дуже важливим фактором. Є люди, які проводять більшу частину дня в машині, і в цьому місці все вирішується. Автомобілі також стали часто використовуватися в тривалих автомобільних поїздках з сім’ями. Діти, батьки та інші родичі стають постійними пасажирами, саме водій відповідає за утримання автомобіля і повинен постійно стежити за автомобілем за всіма

важливими показниками. Таким чином, узгодженість всіх систем є фундаментальним елементом спокою. Зазвичай портрібно слідкувати за індикацією помилок, значків на панелі приладів у авто, так ви будете почувати себе в більшій безпеці [2].

1.2 Архітектура проекту

Еволюційне розвинення електронної апаратури стрімко змінює людство, і ми усвідомлюємо це, перш за все, в соціальній сфері, в сфері комунікацій і комунікацій. Перше, що спадає на думку в цьому відношенні, – це комп'ютери, Інтернет та мобільні телефони. Ми демократично шукаємо необхідну інформацію, у нас є можливість зв'язатися з бажанаю людиною , незважаючи на наше місце розташування. Ми проводимо дистанційне навчання і можемо брати участь у групах, що представляють професійний, соціальний та культурний інтерес. Все це стало можливим в основному завдяки виходу мікропроцесора і створення мікропроцесорної системи.

Мікропроцесори та мікроконтролери широко використовуються в побутовій техніці, автомобільній електроніці, аерокосмічній та військовій промисловості.

Комп'ютер – це електронний пристрій, призначений для обробки та зберігання інформації, виконання різних обчислень та спілкування з користувачами через різні інтерфейси. Він складається з апаратних і програмних компонентів. Апаратне забезпечення включає процесори, оперативну пам'ять, Пристрої зберігання даних, Введення-виведення та різні інші компоненти. Розділ програмного забезпечення містить операційну систему та програми, що використовуються для виконання завдань, від обробки текстів до графічного дизайну та ігор.

Мікропроцесор – це інтегральна схема, яка виконує функцію обробки даних і виконання інструкцій в комп'ютері. Персональні комп'ютери є

важливим обчислювальним компонентом більшості сучасних комп'ютерів, включаючи смартфони, планшети, мікроконтролери та інші пристрої.

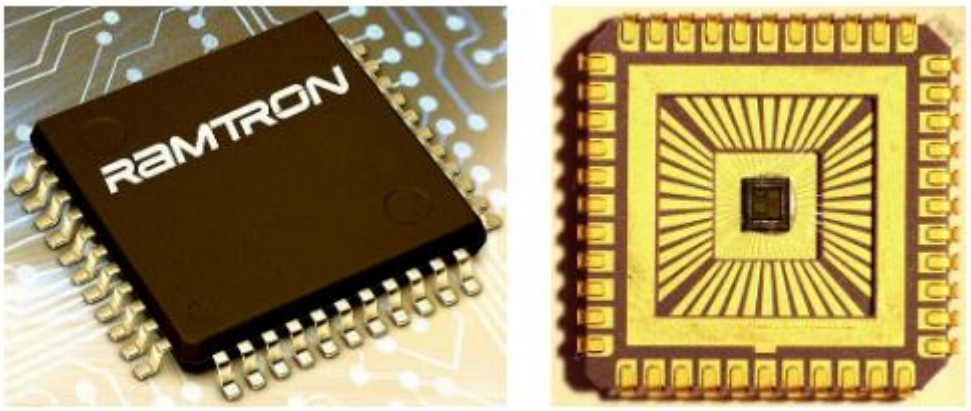

Рисунок 1.2 – Інтегральна мікросхема та її внутрішня будова [3]

Мікропроцесор є важливою частиною комп'ютерної архітектури, і без нього ви не зможете виконувати операції на комп'ютері. Це програмований пристрій, який приймає вхідні дані, виконує з ними деякі арифметичні та логічні операції та дає бажаний результат. Простіше кажучи, мікропроцесор – це цифровий пристрій, який може витягувати інструкції на чіпі з пам'яті, декодувати та виконувати їх, а також видавати результати.

Блок-схема мікропроцесора приймає купу інструкцій машинної мови, виконує їх і повідомляє процесору, що йому робити. При виконанні інструкцій мікропроцесор виконує три основні функції. Він виконує основні операції, такі як додавання, віднімання, множення та ділення, а також різні логічні операції з використанням арифметично-логічної одиниці (ALU). Новий мікропроцесор також виконує операції з використанням чисел з плаваючою комою.

Дані на мікропроцесорі можна переміщати з одного місця в інше. Він має регістр лічильника додатків (ПК), який зберігає адресу наступної інструкції на основі значення ПК. Мікропроцесор переміщується з одного місця в інше і приймає рішення [21].

Мікроконтролери (рисунок 1.3) широко використовуються (рисунок 1.4) в різних сферах завдяки своїй гнучкості, низькій вартості та високій продуктивності.

Рисунок 1.3 – Мікроконтроллер на базі ESP8266 [4]

Рисунок 1.4 – Сфери використання мікроконтроллерів [3]

Мікроконтролери також широко використовуються в автомобільній електроніці. Наприклад, автомобіль Peugeot 206 оснащений 27 мікроконтролерами, в той час як в більш дорогих автомобілях, таких як BMW сьомої серії, використовується більше 60 мікроконтролерів. Він включає впорскування палива, адаптивну жорсткість підвіски, освітлення,

електродвигун склоочисника, Електросклопідйомник, дзеркало заднього виду і т.д. вони перевірять. (рисунок. 1.5).

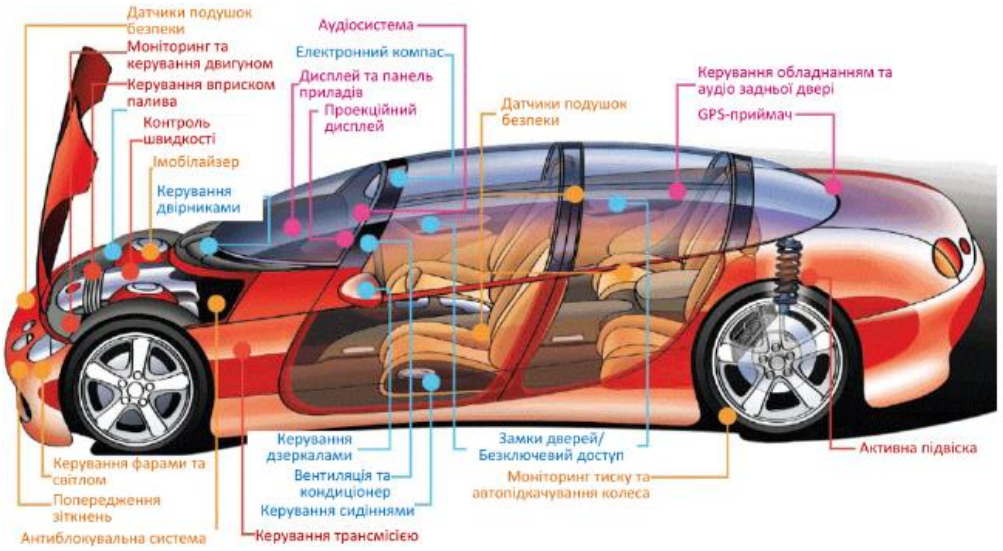

Рисунок 1.5 – Використання мікроконтроллерів в автомобільній техніці [3]

Мікроконтролери, на відміну від мікропроцесорів, зазвичай мають відносно невелику бітову глибину (8-16 біт) і велику кількість командних бітових команд для управління окремими бітами, що дозволяє управляти окремим обладнанням (збільшувати / зменшувати бар'єр, включати / вимикати лампу, включати / вимикати обігрівач, запускати / зупиняти двигун, клапан і т.д. і т. д.). на контролі). Здатність запускати дискретні сигнали окремих бітів, входів і виходів називається «бітовим процесором».

Ще одна ключова відмінність між мікроконтролерами та мікропроцесорами полягає в тому, що мікросхема контролера включає в себе всі елементи для створення простої (а іноді і дуже складної) системи управління. Таким чином, всередині мікроконтролера знаходяться пам'ять даних (ОЗУ), програмна пам'ять (енергонезалежна пам'ять), генератор тактових імпульсів, таймер, лічильник, паралель і siri. Він може складатися з декількох пасивних елементів (резисторів, конденсаторів і кварцових резонаторів). І насправді це одноплатний міні-комп'ютер на базі одного чіпа, що підходить для вбудовування в об'єкти управління. Середня вартість системи мінімальної

конфігурації становить кілька десятків доларів (в порівнянні з середньою вартістю персонального комп’ютера).

Архітектура ModeNCU v3 складається з систем керування та синхронізації (1), логічно-арифметико пристрою (2), регістрів (3), пам’яті (4) та програмна пам’ять (5), портів (6), пристроїв (інтерфейсів, модуляторів, лічильників), регістри (7) (рисунок 1.6) [3].

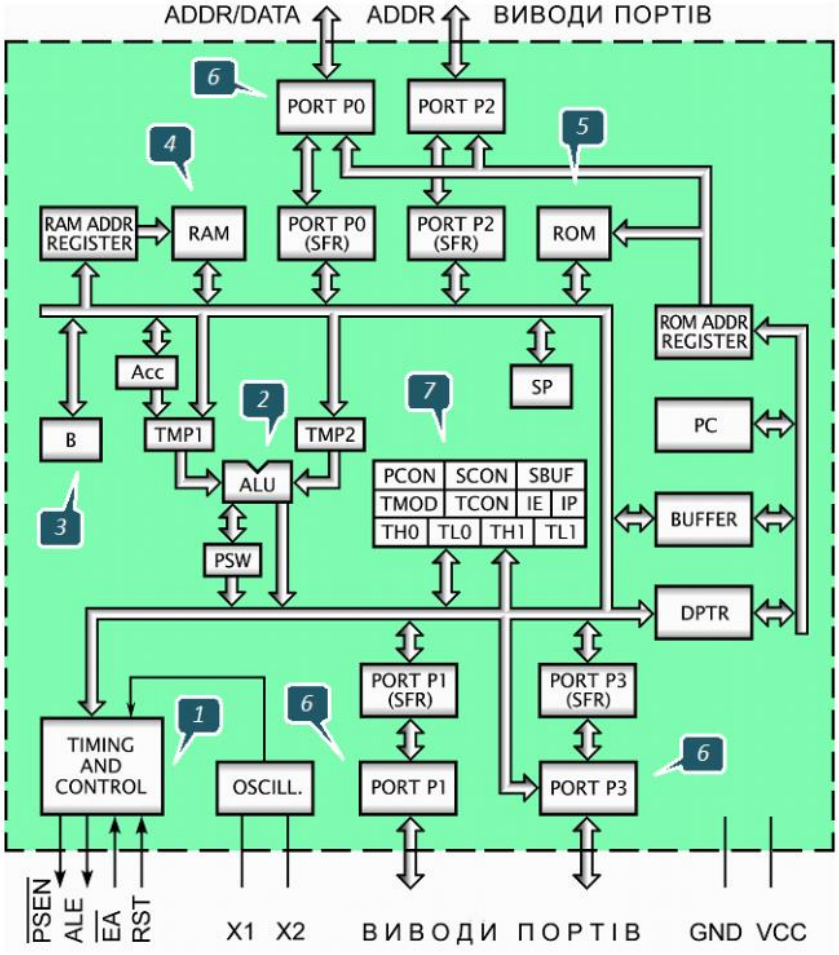

Рисунок 1.6 – Архітектура типового мікроконтролера [3]

1.3 Постановка завдання розробки проекту

Як було написано у п.1.1, панель приладів у автомобілі невід’ємна складова частина. Це найшвидший та найефективніший спосіб повідомити водія про потенційні несправності та слідкувати за станом агрегатів автомобіля.

Головне завдання цієї роботи зробити панель приладів автомобіля для отримання показників температури, датчик рівня палива, покажчик поворотів, спідометр, лічильник відображення контролю пробігу, прилад вимірувач контролю оборотів двигуна.

РОЗДІЛ 2
ВИБІР ТА ОГЛЯД ЗАСОБІВ РОЗРОБКИ ПРОЕКТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ

2.1 Огляд мікроконтролера ESP8266

Мікроконтролер (рисунок 2.1) – це плата, реалізована на чіпі ESP8266, модулі UART WiFi вирізняються з наднизьким енергоспоживанням. Сам чіп призначений для пристроїв зі світу інтернету речей, ця плата вже являє собою USB-з’єднання, регулятор потужності, а весь вихідний сигнал чіпа розділений на гребінку зі стандартним кроком 2,54 мм, що спрощує розробку.Це дозволяє розмістити його на макеті і створити прототип без використання паяльника. Крім того, плата постачається з програмним забезпеченням NodeMCU і може бути запрограмована за допомогою мови Arduino ID [5].

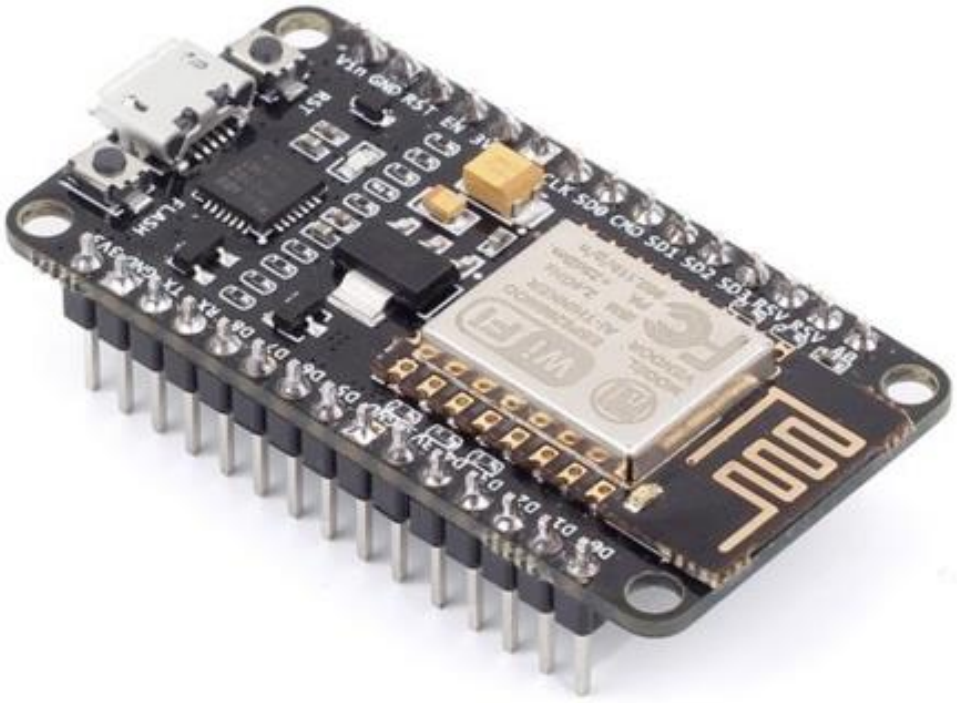

Рисунок 2.1 – Мікроконтролер ESP8266 [5]

2.1.1. Характеристики ESP8266 (рисунок 2.2):

- стандарт WI-FI 802.11;
- AP+STA/AP підтримка;
- вага пристрою становить 18 грам;
- висота контактів 23 мм;
- інтерфейс USB;
- габарити 48*26 мм;
- робочі температури пристрою становлять від -40 до +125 градусів;
- передача даних становить від 110 до 460800 b/сек;
- плата живеться номіналом накпруги 5 вольт;
- сила струму при виведенні становить 15 мА;
- інтерфейси передачі даних UART/GPIO;
- інтерфейси програмування ARDUINO IDE.

Із характеристик видно, що як довго мікроконтролер буде працювати від акумулятора, визначити непросто. Споживання енергії варіюється в дуже широкому діапазоні воно становить 170 мА при передачі на повну потужність і всього 10 мікроампер в сплячому режимі.

ESP8266 – це флеш-пам’ять, призначена для використання підключених до неї модулів пам’яті. Нагадаємо, що кількість циклів перезапису цього типу пам’яті в 10 000 разів більше. Цього достатньо, якщо програма записує налаштування в пам’ять або веде будь-який журнал даних, але якщо програма записує дані занадто швидко, пам’ять швидко перестане працювати [6].

Модуль ESP8266 дозволяє записувати ваші власні програми для їх запуску. Ви можете скопіювати код з мови С і завантажити модуль. Ця процедура називається «перепрошивкою». Для того щоб програма виконувала корисні функції, вона повинна вміти відправляти і отримувати дані по мережі, маніпулювати зовнішніми датчиками, входами і виходами.

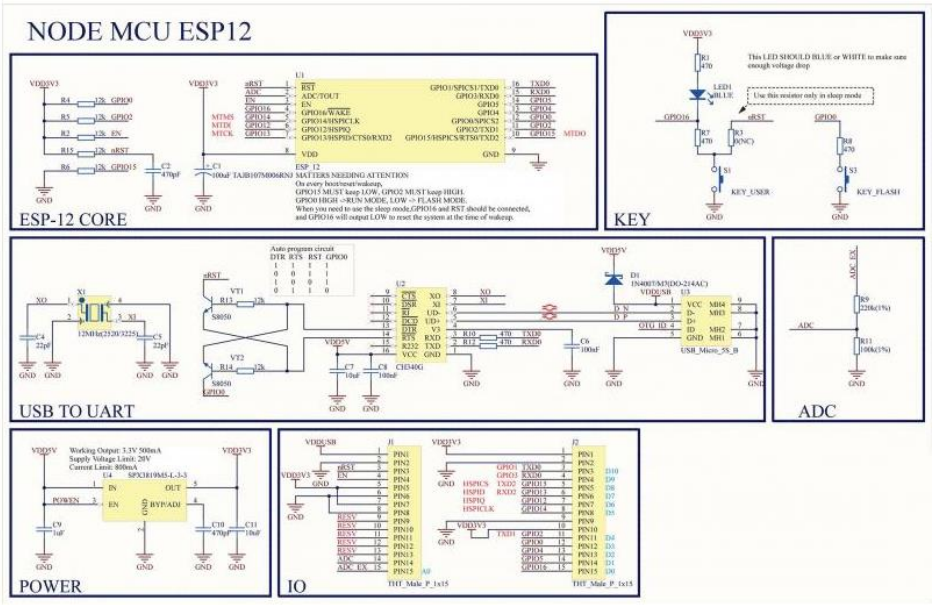

Рисунок 2.2 – Характеристики ESP8266 [6]

2.1.2 Підключення до ESP8266 (рисунок 2.3).

Оскільки ESP8266 є пристроєм Wi-Fi, ви можете підключатися через Wi - Fi, але перед цим потрібно налаштувати його процесор залежить від назви локальної мережі та від неї. Звичайно, це вірно, якщо ви хочете підключити модуль до мережі. Якщо сам модуль працює в режимі точки доступу, то все трохи складніше.

Можна використовувати послідовний порт (UART), щоб полегшити роботу модуля на етапах програмування та налагодження програми. ESP8266 має для цього спеціальний послідовний порт – 2 порти, зазначені Rx і Tx. Tx – допомагає для передачі даних, а Rx-допомагає для прийому. Модуль використовує ці пор ти для підключення до відповідних портів. Найзручніше підключити цей порт до комп’ютера за допомогою адаптера USB-UART. Це з’єднання дозволяє надсилати команди з програми терміналу в модуль безпосередньо з клавіатури, отримувати відповіді від модуля до терміналу або підключати додаток до модуля.

Якщо підключатися через UART, то потрібно встановити однакову швидкість порту. Під час процесу завантаження модуль ESP8266 автоматично

визначить швидкість з’єднання спільного пристрою та спробує встановити той самий пристрій.

У модулі ESP8266 є послідовний порт. Його основна мета – це виведення інформації про діагностику та введення в експлуатацію. Це дуже корисно при перевірці програми.

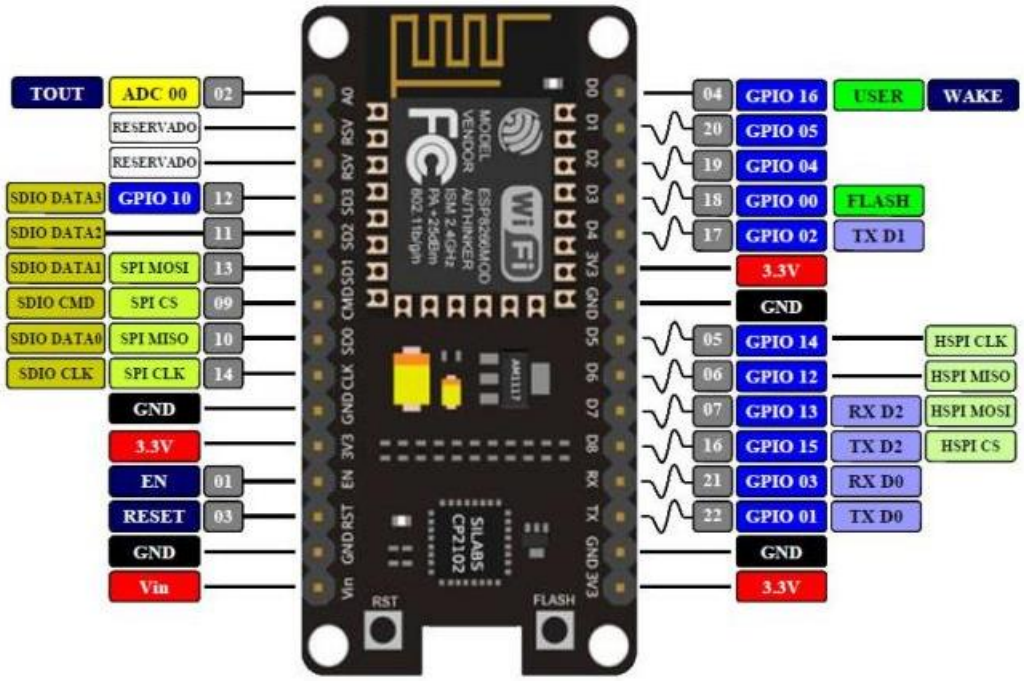

Рисунок 2.3 – Схема підключення до ESP8266 [8]

2.1.3 Теорія WI-FI

Якщо використовувати стандартний пристрій Wi-Fi, рекомендується зрозуміти, як він працює. Високий рівень Wi-Fi – це бездротова мережа для з’єднань TCP / IP. Wi-Fi – це набір протоколів бездротової мережі, наведених за стандартом IEEE802.11.

Пристрій, який називається бездротовою точкою доступу (AP) діє як вузол зв’язку. Зазвичай він підключається або працює в режимі маршрутизатора. Наприклад, домашній Wi-Fi-роутер працює в цьому режимі.

Модуль ESP8266 може працювати як в режимі точки доступу, так і в режимі клієнтської робочої станції, або він може працювати одночасно в обох режимах. У більшості випадків точка доступу має підключення до Інтернету,

яке діє як Міст між декомунізуючим пристроєм та Інтернетом. Кілька робочих станцій у локальній мережі також спілкуються між собою через точки доступу. 1 станція може підключатися лише до 1 точки доступу одночасно. Кожен пристрій у мережі має свою власну MAC-адресу.

Якщо в полі зору є кілька точок доступу, їх потрібно якимось чином розрізнити, тому кожна точка доступу має мережевий ідентифікатор, який називається SSID (також відомий як ідентифікатор набору послуг, BSSID).Це мережеве ім’я довжиною не більше 32 символів.

Цифровий вихід

Цифрові контакти ESP32 можуть бути налаштовані як вихід для управління вихідними пристроями. Ми повинні налаштувати ці контакти для використання в якості виходу.

Щоб налаштувати ці контакти, pinMode() використовується функція, яка встановлює напрямок контакту як входу або виходу.

- pinMode(pin no, Mode) ця функція використовується для налаштування контакту GPIO як входу або виходу.

pin no **(**номер піна, режим якого ми хочемо встановити).

Mode **(**INPUT, OUTPUT або INPUT_PULLUP)

e.g.pinMode (3, OUTPUT); //set pin 3 as output

Ці контакти ESP32 можуть отримувати струм 40 мА та споживати струм 28 мА, чого достатньо для керування світлодіодами, РК-дисплеями тощо, але недостатньо для двигунів, реле тощо.

Застереження: під час підключення пристроїв до вихідних контактів ESP32 використовуйте резистор. Якщо будь-який пристрій, підключений до ESP32, споживає струм понад 40 мА від ESP32, це призведе до пошкодження контакту або модуля ESP32.

Ці контакти дають вихідний сигнал у вигляді HIGH (3,3 В) або LOW (0 В). Ми можемо встановити вихід на ці контакти за допомогою digitalWrite() функції.

- digitalWrite(pin no, Output value)

Ця функція використовується для встановлення вихідного сигналу як HIGH (3,3 В) або LOW (0 В) pin no номер піна, режим якого ми хочемо встановити.

Output value ВИСОКА або НИЗЬКА

e.g. digitalWrite (3, HIGH);

Цифровий вхід

Щоб зчитувати дані з датчика або будь-якого пристрою/схеми, нам потрібно налаштувати цифровий контакт як вхід. Виводи ESP32 встановлені як цифрові входи (за замовчуванням). Отже, немає необхідності налаштовувати пін як вхід.

Щоб налаштувати контакт як цифровий вхід, `pinMode()` використовується функція. Ми можемо читати дані з контакту GPIO за допомогою digitalRead() функції.

- digitalRead(pin)

Він використовується для читання даних із зазначеного контакту GPIO[10].

Цифровий вхід із підтягуючим резистором

Іноді перемикання з одного стану на інший або контакти, налаштовані як вхідні без жодних підключень до них, можуть спричинити стан високого опору, тобто плаваючий стан. Цей стан може повідомляти про випадкові зміни в стані контакту.

Щоб уникнути цього стану, існує варіант n додавання висувного (до +3,3 В) або спадного (до Gnd) резистора, який допомагає встановити вхідний сигнал у відомий стан. Нижче показано стан високого опору (невизначений) і підтягувальний резистор (рисунок 2.4).

ESP8266 оснащений базовими функціями для цього, набором яких є примітивна «операційна система». Сервіс цієї ОС може бути викликаний додатком. Вони повністю задокументовані, і Вам буде дуже зручно ними користуватися.

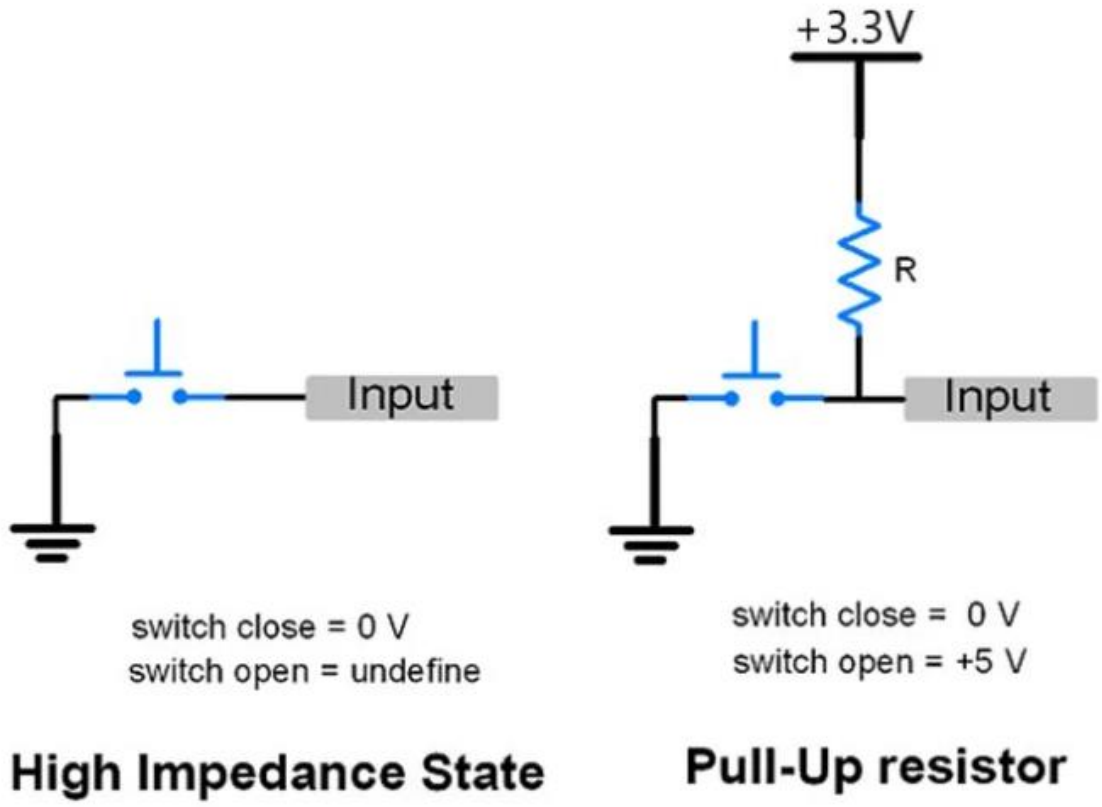

Рисунок 2.4 – Підтягуючий резистор[10]

ESP32 має вбудований настроюваний підтягуючий резистор. Ці резистори дозволяють використовувати pinMode() в режимі INPUT_PULLUP. При підключенні пристрою або датчика до контакту, налаштованого як вхід із підтягуванням, інший кінець має бути підключений до землі.

e.g. pinMode (3, INPUT_PULLUP);

Ми також можемо налаштувати підтягування введення іншим способом. Якщо ми встановимо напрямок штифта як вхід, а потім запишемо на цей штифт значення HIGH, увімкнеться підтягуючий резистор. Іншим чином, якщо ми напишемо HIGH, на контакті, налаштованому як OUTPUT, а потім налаштуємо цей висновок як вхід, також увімкне підтягуючий резистор[10].

e.g.

pinMode (3, INPUT); //set pin as input

digitalWrite (3, HIGH); //setting high on input pin enables pull-up

Розпиновка пінів ESP8266 наведена у таблиці 2.1.

Таблиця 2.1 – Розпіновка пінів GPIO34-39

№ п/п	Pin GPIO	Тип IO
1	2	3
1	GPIO34	Лише введення
2	GPIO35	Лише введення
3	GPIO36	Лише введення
4	GPIO37	Лише введення
5	GPIO38	Лише введення
6	GPIO39	Лише введення

Джерело: [10]

2.1.4 Компіляція

Реалізація програм для ESP8266 написана за допомогою мови С. Перед зашивання коду в модуль необхідно скопіювати її з тексту в машинний код.

Редагувати текстову програму зручно всього в якомусь редакторі, який має підсвітку синтаксису, вбудовану довідку та інші корисні функції. Програмування в такому середовищі дозволяє створити програму, протестувати, та зашити у модуль.

Середовища які найчастіше використовують для розробки програм це ECLIPSE та Arduino IDE.

Платформа Eclipse розроблена для створення інтегрованих середовищ розробки (IDE) і довільних інструментів. Eclipse підтримує розробку різних типів програм, включаючи програми на Java, C/C++, Python, PHP, Java Script та багатьох інших мовах програмування. Він також підтримує різні фреймворки та технології, такі як Android, Spring, MaventaGit.

Arduino IDE – це безшовне, просте та інтуїтивно – зрозуміле середовище програмування. Завдяки системі, керованій спільнотою, і простому інтерфейсу програм аспрощує кодування веб-сайтів ідодатків. Вам не потрібно володіти будь-якими технічними навичками або знаннями, щоб використовувати програмне забезпечення для початківців.

Модулі ESP8266 зазвичай зустрічаються в таких пристроях IoT:

- Розумні пристрої безпеки, включаючи камери спостереження та розумні замки
- Розумні енергетичні пристрої, включно з системами опалення, вентиляції та кондиціонування повітря та термостатами
- Розумні промислові пристрої, включаючи програмовані логічні контролери (ПЛК)
- Розумні медичні пристрої, включно з переносними моніторами.

Як обговорювалося вище, ESP8266 – це лише назва чіпа. По суті, це можна придбати в трьох форматах: мікросхема ESP8266: це базова мікросхема виробництва Espressif, яка постачається неекранованою та її потрібно припаяти до модуля. Це не підходить для більшості користувачів, окрім, можливо, виробників масових пристроїв, які можуть врахувати це у виробничому процесі під вартістю одиниці модуля. Модулі ESP8266: це модулі для поверхневого монтажу, які містять чіп, готовий до монтажу на мікроконтроллер, вироблений Espressif, Ai-Thinker та деякими іншими виробниками. Зазвичай вони захищені та попередньо схвалені FCC для використання. Плати розробки ESP8266: це повні плати розробки IoT MCU , які мають попередньо встановлені модулі. Розробні плати виробляються кількома різними виробниками, а технічні характеристики різних моделей відрізняються. Деякі основні специфікації, про які слід знати під час оцінки варіантів плати розробки ESP8266 IoT, включають:

- контакти GPIO;
- виводи АЦП;
- Wi-Fi антени;
- світлодіоди;
- екранування;
- флеш-пам’ять.

На багатьох міжнародних ринках потрібні екрановані пристрої Wi-Fi, оскільки Wi-Fi створює значні радіочастотні перешкоди (RFI), а екранування

мінімізує ці перешкоди. Тому це має бути ключовим фактором для всіх розробників і виробників вбудованих пристроїв [18].

2.2 Побудова та програмування проекту

Щоб запрограмувати плату потрібно виконати декілька простих кроків:

1. Завантажити програму ARDUINO офіційного сайту (https://www.arduino.cc/en/software)
2. Встановити програму на свій комп’ютер
3. Запусти програму та увійти у вкладку «Настройки» (рисунок 2.4)

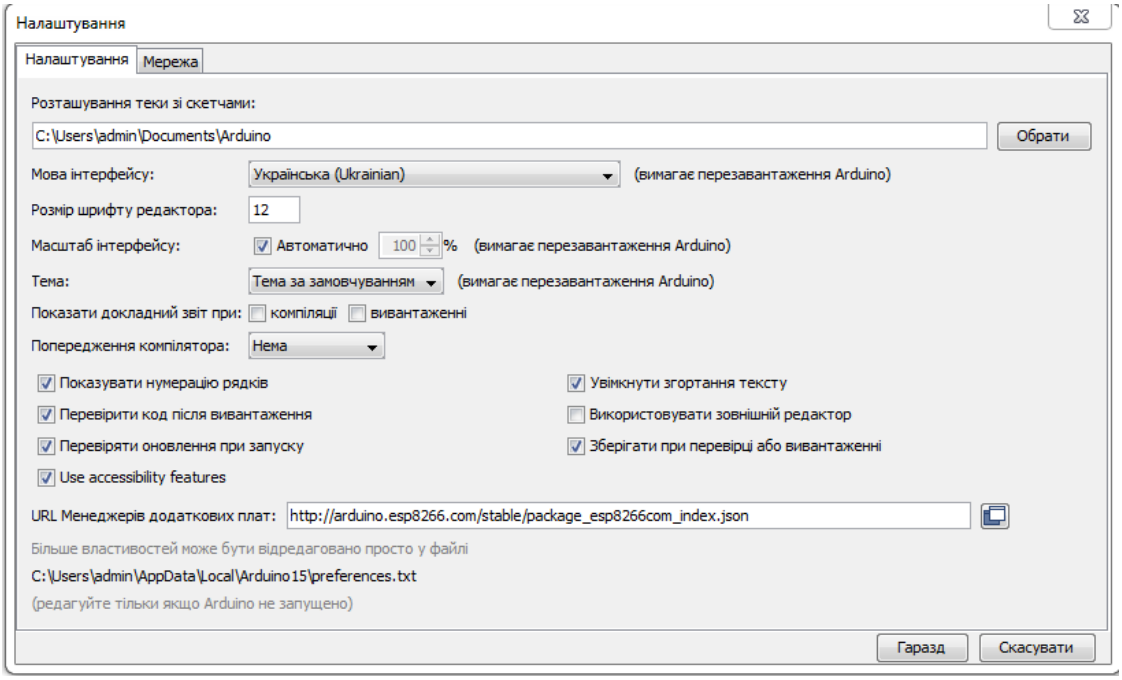

Рисунок 2.4 – Вікно налаштувань програми

4. Вводимо покликання, та натискаємо кнопку «ОК»
5. Далі переходимо до вкладки «Інструменти/Плата/Менеджер плат». Скролимо вікно до самого низу та бачимо (рисунок 2.5)
6. Встановлюємо «esp8266»
7. Переходимо до вкладки «instruments» та у відкритому вікні вибираємо правильну плату (рисунок 2.6).

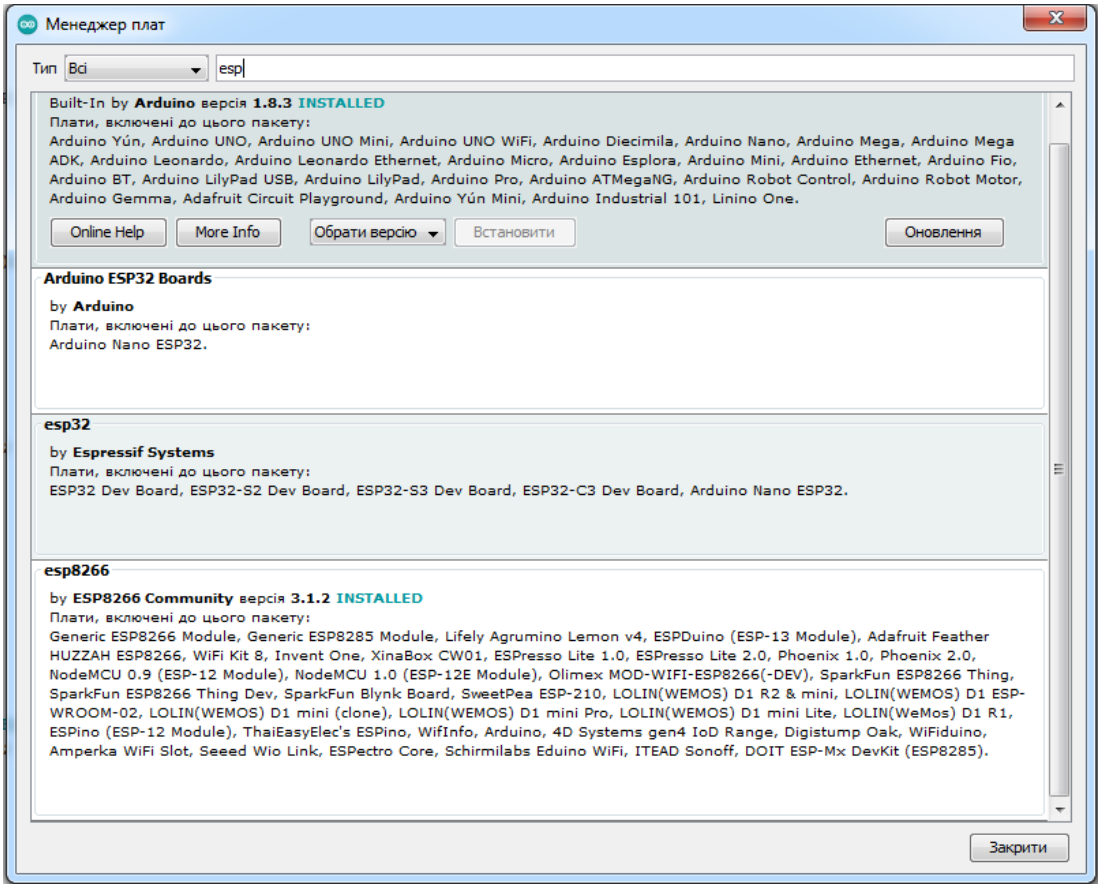

Рисунок 2.5 – Менеджер плат

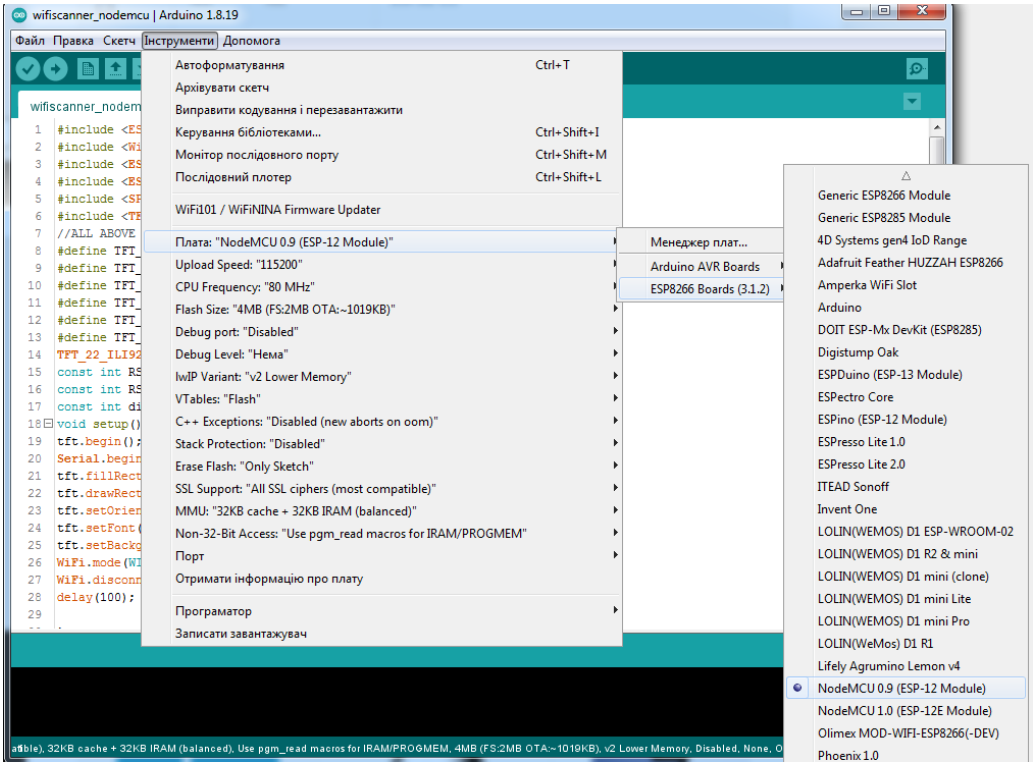

Рисунок 2.6 – Вибір плати

8. Вибираємо правильну швидкість та порт(рисунок 2.7)

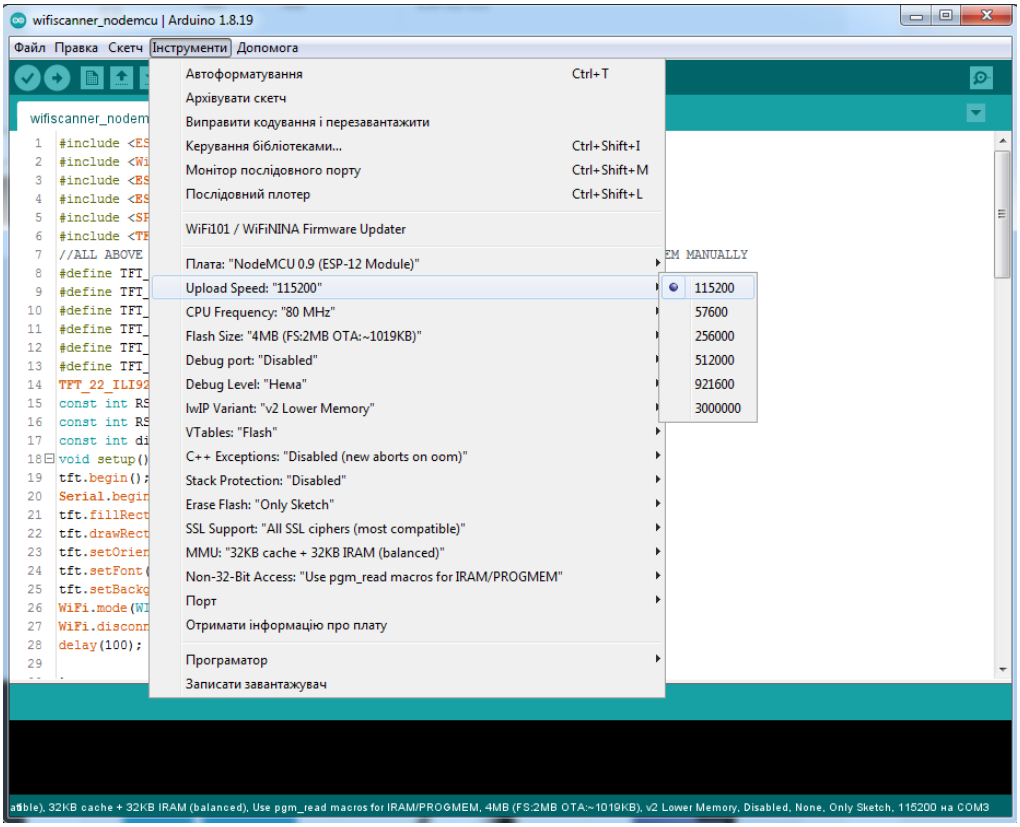

Рисунок 2.7 – Встановлення потрібної швидкості

2.3 Перевірка та тестування мікроконтролера на базі ESP8266

Для перевірки модуля ми візьмемо TFT ili9225 дисплей (рисунок 2.8)

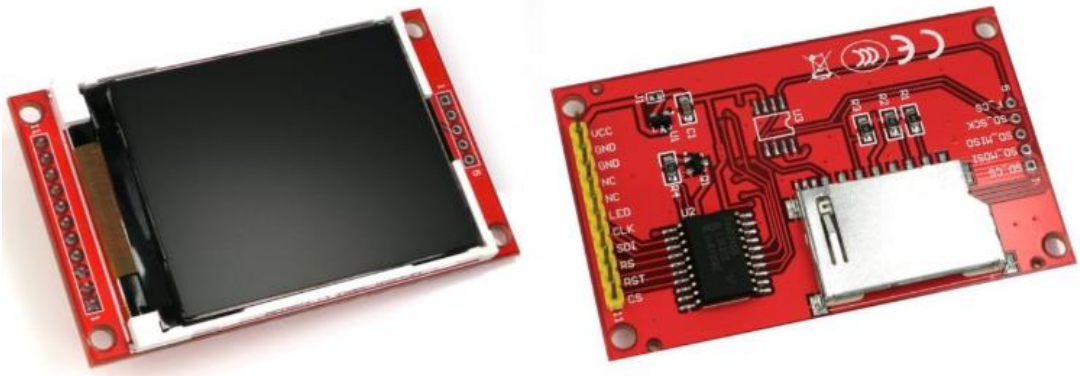

Рисунок 2.8 – TFT ILI9225 DISPLAY [12]

З’єднаємо мікроконтролер та дисплей за допомогою схеми (рисунок 2.9)

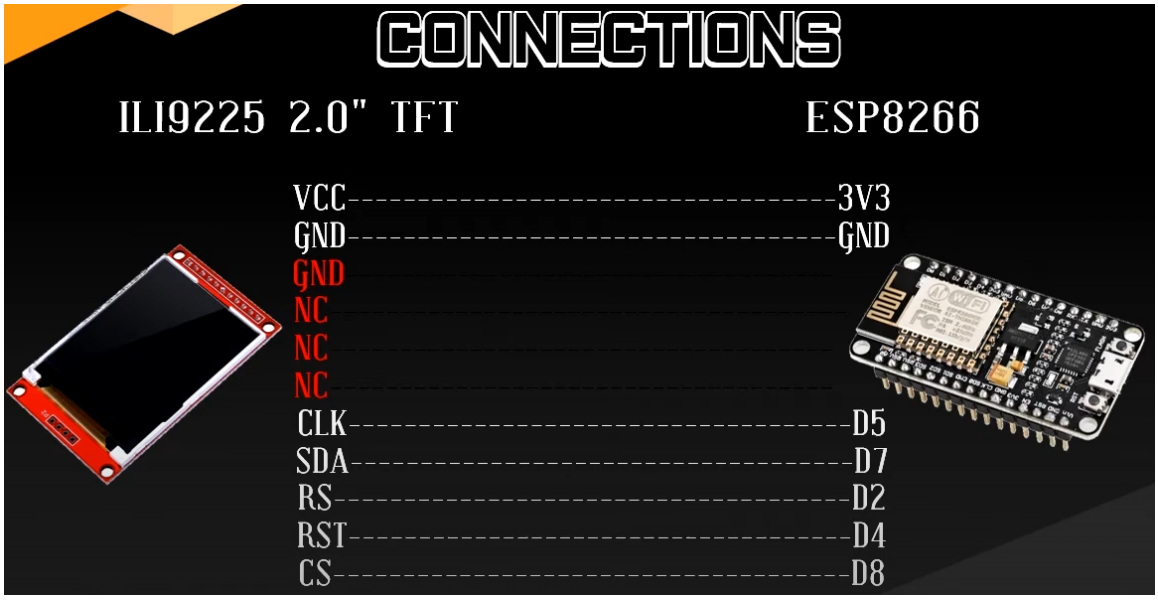

Рисунок 2.9 – З’єднання TFT дисплея до мікроконтролера [7]

Візьмемо для перевірки підключення дисплея готове рішення WI-FI сканера та запрограмуємо мікроконтролер (див. додаток А).

Після перевірки коду в програмі Arduino «зашиваємо» у мікроконтролер та дивимося результат (рисунок 2.10)

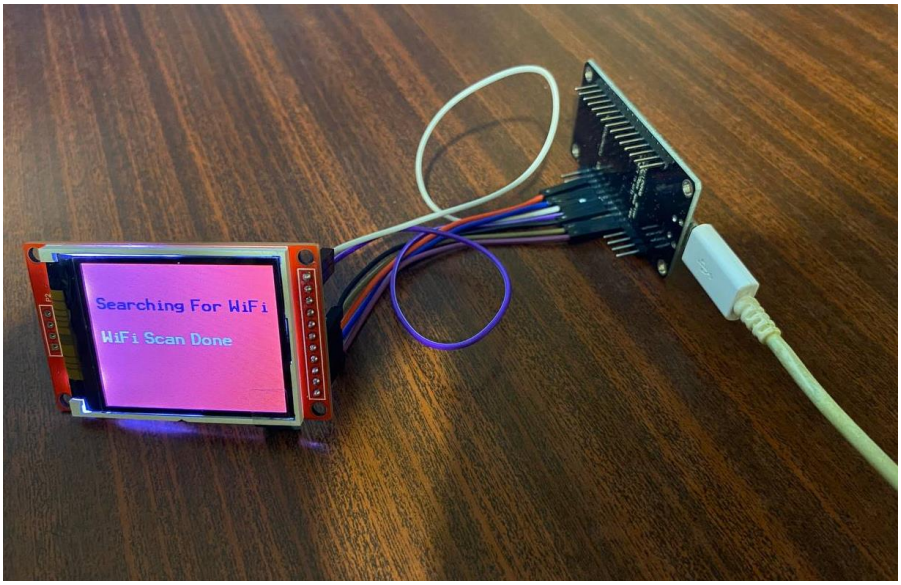

Рисунок 2.10 – Результат перевірки

РОЗДІЛ 3
ОПИС ПРАКТИЧНОЇ РЕАЛІЗАЦІЇ ПРОЕКТУ

3.1 Основні переваги NodeMCUv3 ESP8266

Мікроконтролер ESP8266 NodeMcu V3 із вбудованим підключенням Wi-Fi. Має встановлені інтерфейси GPIO, PWM, IIC, 1-Wire та ADC. Можна налаштувати через інтерфейси керування Arduino IDE та nodemcu. Вирізняється з-поміж інших надзвичайно низькою вартістю та низьким енергоспоживанням, спеціально розробленим для Інтернету речей (IoT), мобільних пристроїв та інших програм [8].

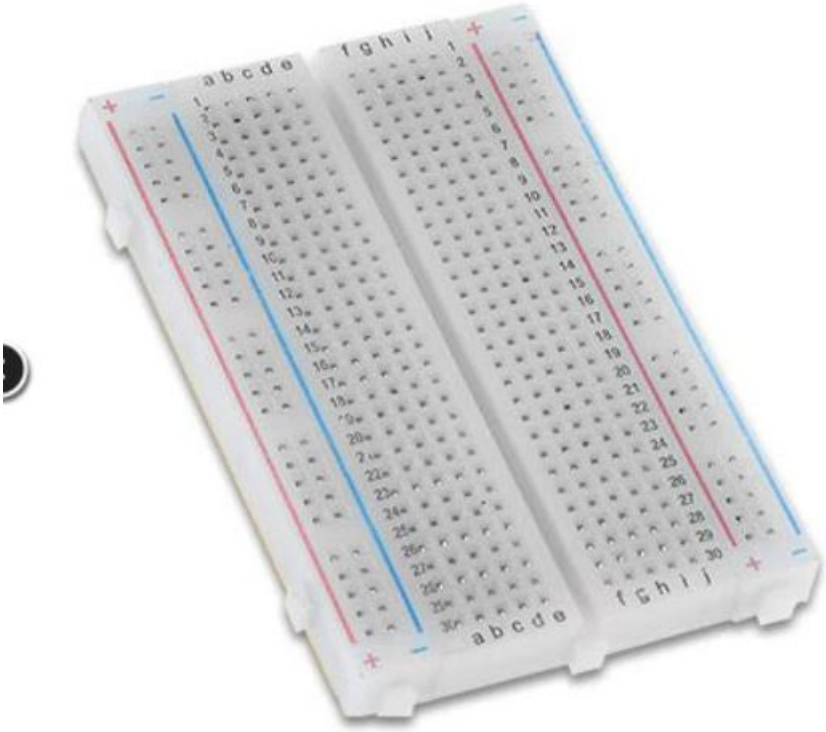

Рисунок 3.1 – Макетна плата [9]

У чому ж переваги плати NodeMCUv3 Wi-Fi на основі модуля ESP8266? По-перше, NodeMCU v3 має вбудований модуль Wi-Fi, що дозволяє підключати пристрої до Інтернету або створювати локальні мережі без необхідності додаткового обладнання. По-друге, плата оснащена USB портом,

що спрощує процес налажування та програмування. Не потрібно використовувати окремі програматори та конвертери. По-третє, NodeMCU має компактні розміри, що зробить її зручною для використання в компактних пристроях та проектах з обмеженим доступом. По-четверте, плата має багато виводів загального призначення (GPIO) (рисунок 3.2) , що дозволяє підключати різноманітні датчики, актуатори та різні периферійні пристрої.

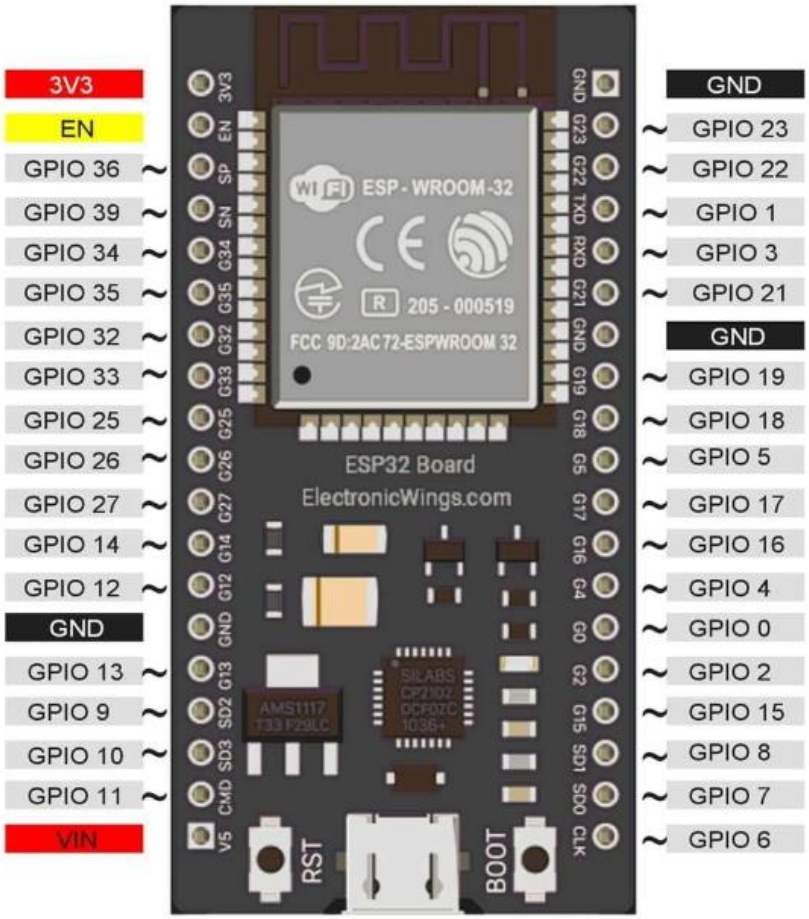

Рисунок 3.2 – ESP32 GPIO [10]

По-п’яте, низька вартість плати. Це дозволяє доступно розробляти аматорські проекти та масового виробництва IoT-пристроїв.

3.1.1.Пояснення розводки та функцій NodeMCU (рисунок 3.3)

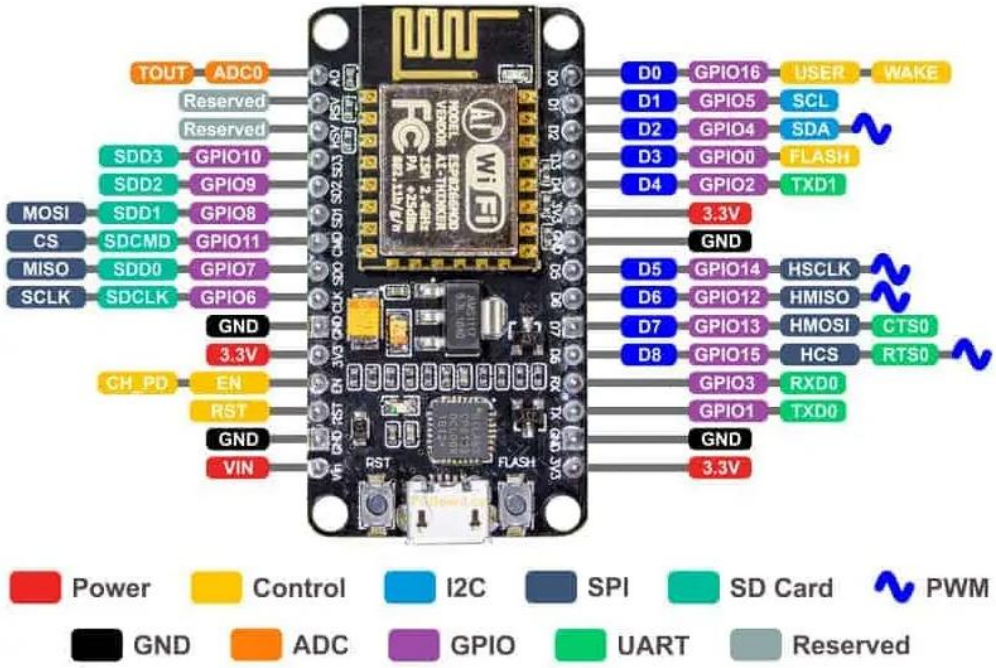

Рисунок 3.3 – Розводка ESP32 [10]

Слід зазначити, що ESP8266 не є стандартною платою розробки, такою як Arduino Uno. Натомість ESP8266 – це підключений мікроконтролер з можливостями Wi-Fi. Це означає, що різні виробники можуть вибирати плати розробки з абсолютно нестандартними конфігураціями виводів вводу-виводу. Але хороша новина полягає в тому, що макет NodeMCU є фактичним стандартом, якого дотримуються більшість виробників плат.

Зверніть увагу, що будова плати може відрізнятися від мого. Тому перед початком роботи рекомендується ознайомитися з етикетками для трафаретного друку на дошці або переглянути керівництво користувача для конкретної розробки. Для цього проекту я вирішив використовувати NodeMCU v3, оскільки на мій погляд він здається найпростіший.

У деяких випадках вхід в систему є одним з 2х чітко визначених станів, таких як високий і низький. Коли вхід ADC0 використовується для зчитування аналогового вхідного сигналу і перетворення його в цифрове значення, яке може відображатися як число, програма, що працює на ESP8266, може його обробити. Слід звернути увагу, що Adc0 – це лише вхід-вихід, ESP8266 не підтримує аналогові виводи.

ESP8266 має кілька версій і моделей, тому розпиновка може трохи відрізнятися залежно від конкретної моделі. Однак основні функціональні піни зазвичай залишаються такими ж.

- VCC – живлення плати, живлення пристрою – 3.3V.
- GPIO (General Purpose Input/Output) – це загального призначення вводу/виводу, який може використовуватися для підключення різноманітних пристроїв або сенсорів.
- GND – земля або нульовий потенціал, який повинен бути з'єднаний з землею джерела живлення.
- TX і RX – використовуються для зв'язку через UART, наприклад, для зв'язку з комп'ютером або іншими мікроконтролерами.

Крім цього, ESP8266 також може мати інші виводи, такі як GPIO4, GPIO5 і т.д., які можуть бути використані для різних цілей, таких як зовнішні переривання, зчитування аналогових сигналів тощо. Рекомендується звертатися до документації конкретної моделі ESP8266 для отримання детальної інформації щодо розпиновки.

ESP8266 має вбудований SPI (Serial Peripheral Interface) і може використовуватися для спілкування з різними пристроями через цей інтерфейс. SPI – це протокол обміну даними, який дозволяє передавати дані між мікроконтролером (ESP8266) та зовнішніми пристроями, такими як датчики, дисплеї, SD-карти, EEPROM і багато інших.

Для використання SPI на ESP8266 потрібно використовувати спеціальні бібліотеки, наприклад, «SPI.h». Зазвичай процес включає наступні кроки:

- ініціалізація SPI;
- налаштування параметрів SPI, таких як режим роботи, швидкість передачі тощо;
- взаємодія з підключеними пристроями через SPI, такими як передача або отримання даних [20].

3.1.2. USB to Serial Converter – CP2102 or CH340G.

Кожен NodeMCU містить конвертер USB-Serial. Офіційна плата базується на чіпсеті CP2102 і забезпечує найкращу сумісність. Заводська ESP8266 застосовує чіпсет CP2102, який включає офіційно оригінальний чіпсет NODEMCU. Другим популярним послідовним USB-конвертером є CH340G, який використовується у буджетних платах, включно lolin-пристрої. Драйвери можуть використовуватися в інших конструкціях, включаючи набори мікросхем FTDI, зазвичай такі конструкції зустрічаються рідко.

Залежно від операційної системи, яку ви використовуєте з NodeMCU, потрібно встановити відповідний драйвер. Як правило, Windows 10 відразу підхоплює контролер CP2102, тоді як CH340G може вимагати додаткової установки [19].

3.2 Прототипування проекту

В даному розділі описуватиметься розробка панелі керування автомобілем допомогою мікроконтроллера ESP8266 та програмування NodeMCU. Цей простий прилад буде відображати зображення панелі приладів автомобіляна TFT дисплеї.

При розробці проекту будемо використовувати наступні деталі:

– TFT 22 ILI9225 display (рисунок 3.4);
– мікроконтролер ESP8266 ;
– макетна дошка;
– світлодіоди;
– кнопки;
– зумер;
– кабелі-перемички;
– кабель Micro USB.

Рисунок 3.4 – TFT 22 ILI9225 display [11]

Характеристики TFT дисплея наступні:

– глибина кольору 16 біт RGB, 65 тис. кольорів;
– розмір екрану 2,0 (дюйми);
– тип TFT;
– контролер ILI9225;
– розширення 176*220 (точок);
– інтерфейс 4-х провідний інтерфейс SPI;
– підсвічування поводиться за допомогою трьох білих світлодіодів;
– активна область 31.68x39.60 (мм);
– розмір друкованої плати модуля 38.30x62.48 (мм);
– робочий діапазон пристрою здійсюється живленням 5 В;
– вага пристрою 20 грам;
– маса 20 (г) [12]

Роспиновка пінів показана на рисунку 3.5

Number	Pin Label	Description
1	VCC	LCD Power positive (3.3V~5V)
2	GND	LCD Power ground
3	GND	LCD Power ground
4	NC	Not defined, reserved
5	NC	Not defined, reserved
6	LED	Backlight control, high level lighting, if not controlled, connect 3.3V always brigh
7	CLK	LCD SPI bus clock signal
8	SDI	LCD SPI bus write data signal
9	RS	LCD register / data selection signal, high level: register, low level: data
10	RST	LCD reset signal, low level reset
11	CS	LCD chip select signal, low level enable

Рисунок 3.5 – Роспиновка пінів TFT 22 ILI9225 [13]

Для того, щоб підключити світлодіод до ESP8266 потрібно:

– плата розробки ESP8266;
– світлодіодні лампи;
– резистори на 330 Ом;
– макетна плата;
– кабелі-перемички.

Підключення виглядає наступним чином (рисунок 3.6): потрібно підключити довгу ніжку (анод) світлодіода через резистор до контакту D1 (GPIO 5) на NodeMCU, а коротку ніжку (катод) безпосередньо до GND. Резистор використовується для обмеження струму, що протікає через світлодіод, щоб запобігти пошкодженню [14].

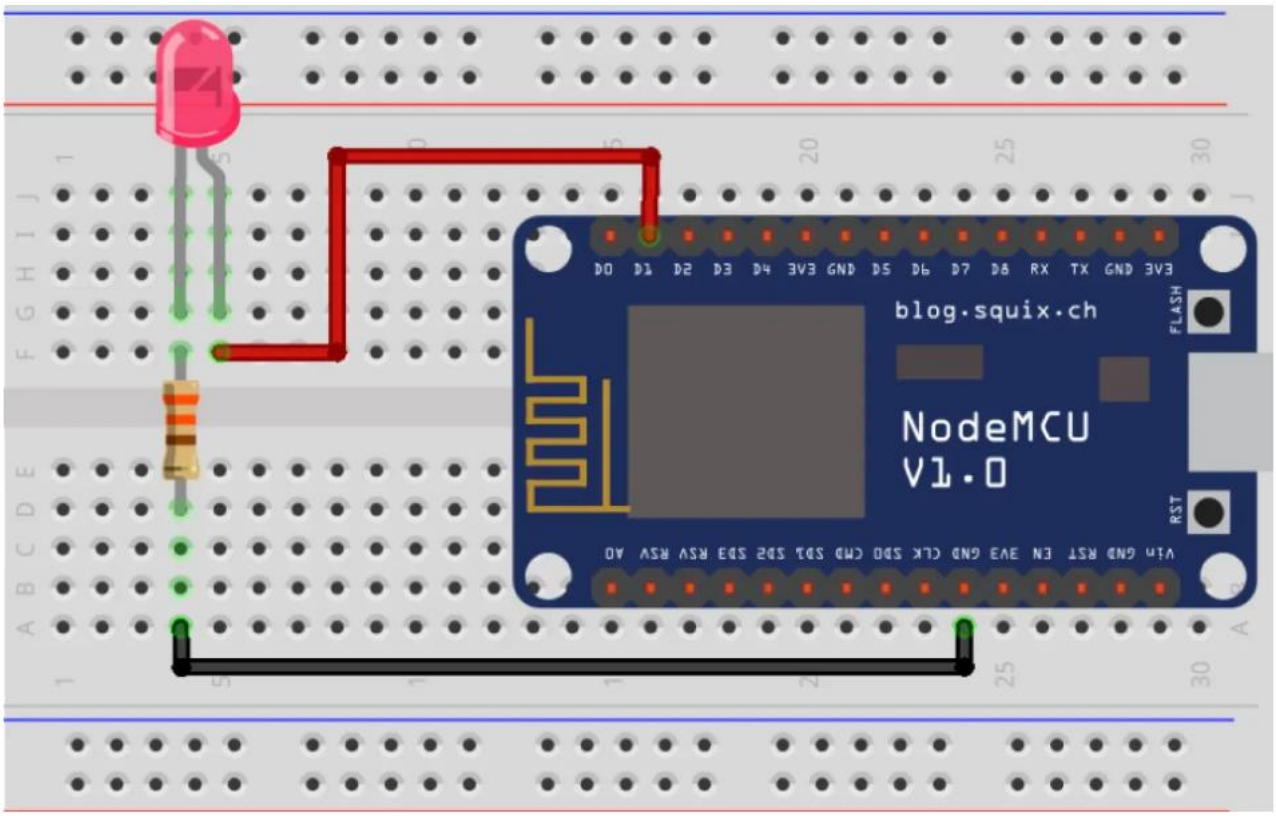

Рисунок .3.6 – Схема підключення світлодіода [14]

Для підключення кнопки до ESP8266 потрібно:

– плата розробки ESP8266;
– кабель microUSB;
– нажимна кнопка;
– макетна плата;
– кабелі перемички;
– резистор на 330 Ом.(рисунок 3.7)

Кнопка є миттєвим перемикачем. Миттєві перемикачі – це перемикачі, які залишаються у включеному стані лише до тих пір, поки їх натискають (натискають, утримують, намагнічують тощо) [14].

Рисунок 3.7 – Компоненти для підключення [15]

Підключення апаратного забезпевення проводиться наступним чином (рисунок 3.8):

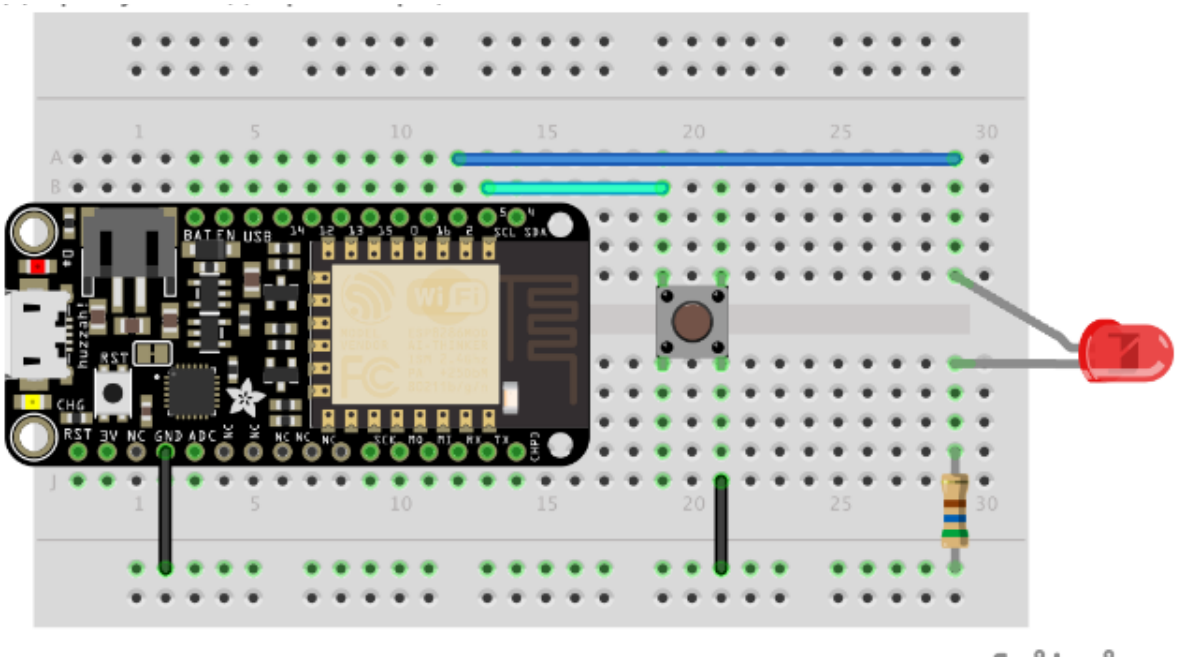

Рисунок 3.8 – Підключення світлодіода через кнопку[15]

Для підключення потрібно звернути на полярність деталей. Поляризовані компоненти можна підключати до ланцюга тільки в одному напрямку. Це стосується особливо світлодіодів.

Для перевірки правильності підключеної схеми в програмі Arduino IDE код(див. додаток Б)

Результат наведений на рисунку 3.9.

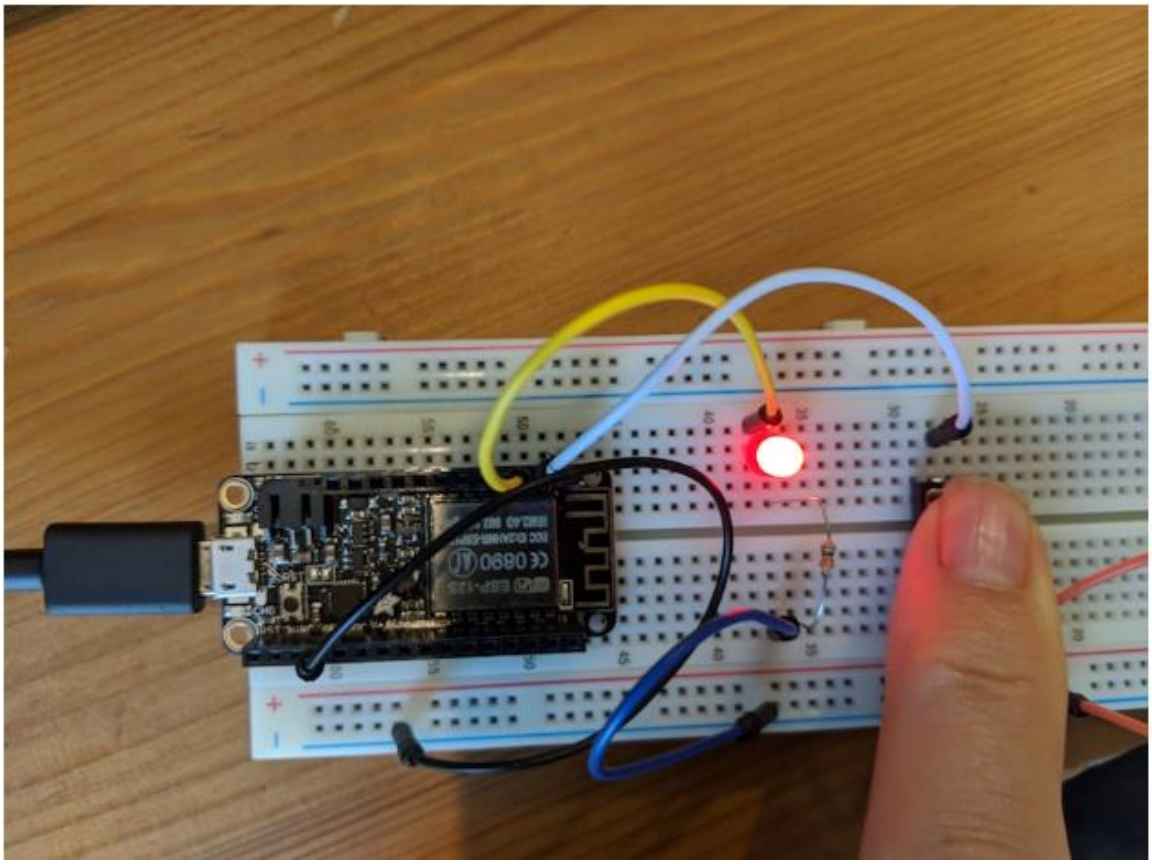

Рисунок 3.9 – Результат [16]

Світлодіод має загорятися коли натискається кнопка [16].

Підключення зумера. Зумер зазвичай має два контакти (рисунок 3.10):

– Негативний (–) контакт потрібно підключити до контакту GND;

– Позитивний (+) контакт отримує керуючий сигнал від ESP8266 (прямо або безпосередньо через реле).

Рисунок 3.10 – Зумер [17]

Схема підключення зумеру показана на рисунку 3.11.

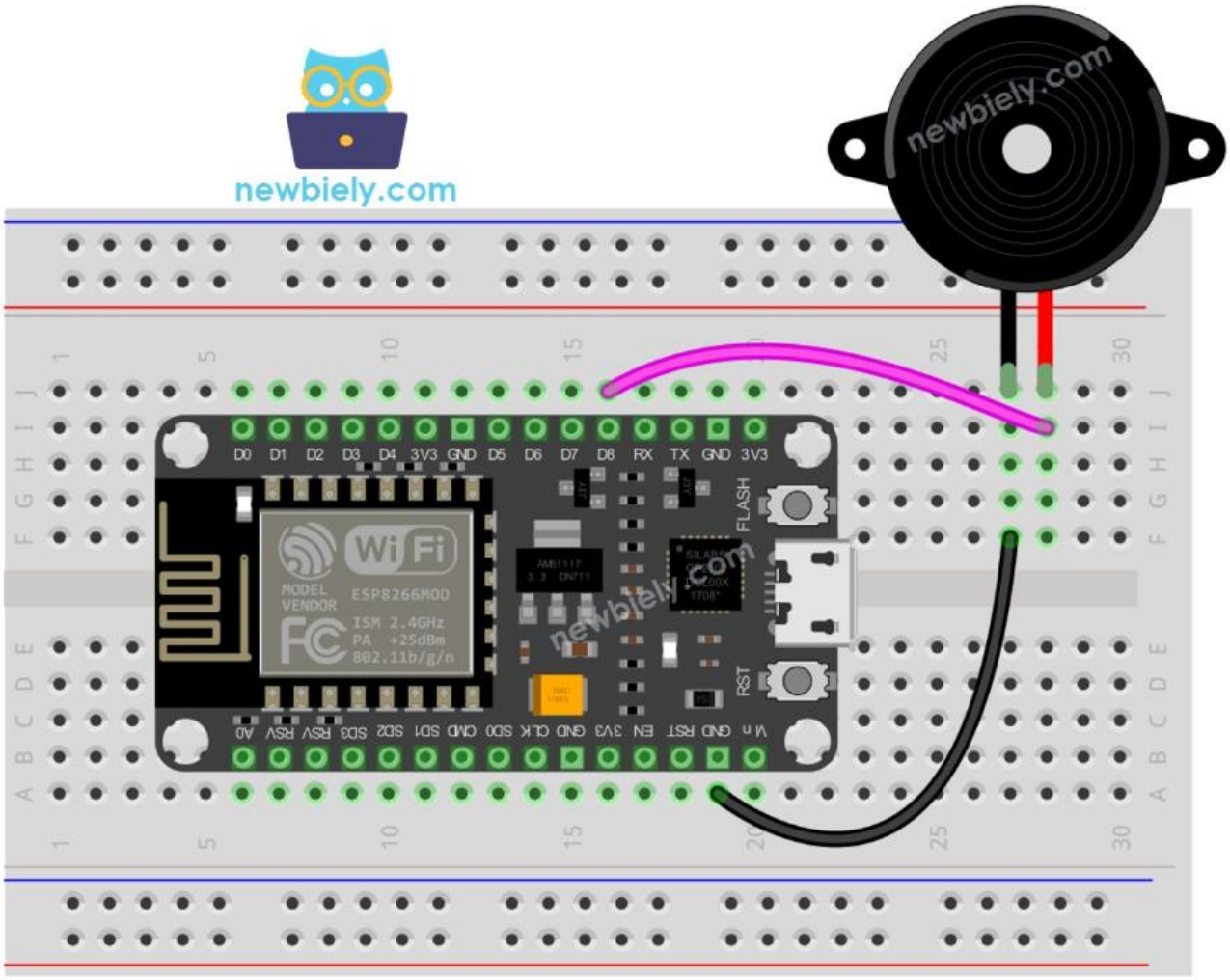

Рисунок 3.11 – Схема підключення [17]

Для перевірки зумеру скористаємося кодом із додатку В.

3.3 Принцип роботи розробленого пристрою

У цьому проекті я використовував плату розробки NodeMCUv3 ESP8266, щоб створити контролер для електромобілів, один датчик може показувати швидкість, інший може показувати оберти та поточну передачу, також є індикатори напрямку, гальма, покажчик світла фар, покажчик кількості

пройденого пробігу. У додатку Г Ви можете знайти код, тому сміливо використовуйте його для свого електромобіля.

ВИСНОВКИ

На основі зроблених досліджень слід провести наступні логічні підсумки: була проведена демонстрація актуальності теми дослідження. У цій роботі показано важливість розробки проекту системи моделювання керування автомобілем на основі мікроконтролера.

Реалізовано систему керування автомобілем на базі мікроконтролера, за допомогою якої можна вирішити велику кількість задач як для виробництва та навчання.

Розроблено симуляцію панелі приладів автомобіля. Розроблено роботу датчиків швидкості, обертів двигуна, рівня палива, покажчика поворотів, датчик кількості пробігу в автомобілі, покажчик ввімкнутого світла.

Досліджено рішення розробки проекту системи симуляції автомобілем на базі мікроконтролера, проведено вивчення мікроконтролерів ESP8266, їх особливості, технічні характеристики, програмування.

Візуалізовано роботу мікроконтролера, представлені схеми підключення різних компонентів.

Спроектований проект на базі мікроконтролера ESP8266 є надзвичайно універсальним пристроєм, який пропонує високу продуктивність і багатий набір функцій за доступною ціною. Він ідеально підходить для реалізації різних IoT проєктів, від простих домашніх автоматизацій до складних промислових систем. Його енергоефективність, підтримка різних інтерфейсів і широка спільнота розробників роблять його одним із найкращих виборів для розробки IoT рішень.

На мою думку, реалізація подібних проектів цілком можлива, адже в цьому проекті використовується мікроконтролер на базі ESP8266. Можна створити подібні навчальні проекти та прототипування. Завдяки своїй простоті та популярності, ESP8266 є відмінним інструментом для навчання і створення прототипів.

Результати розробки даної роботи доцільно використовувати на підприємствах, з виготовлення деталей, приладів до автомобілів, а також у повсякденному житті та навчанні.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Understanding Vehicle Computer System: Key Components and How It Works SINTRONES Technology Corp. SINTRONES. URL: http://www.sintrones.com/application/understanding-vehicle-computer-system-key-components-and-how-it-works/ (date of access: 10.02.2024).

2. Панель приладів машини: що означають поширені символи. Autostate. URL: https://autostate.com.ua/uk/shho-oznachajut-znachki-na-paneli-priladiv-avtomobilja.html (дата звернення: 10.02.2024).

3. Панель приладів машини: що означають поширені символи. Autostate. URL: https://autostate.com.ua/uk/shho-oznachajut-znachki-na-paneli-priladiv-avtomobilja.html (дата звернення: 10.02.2024).

4. Introduction to the Internet of Things (IoT): ESP8266 architecture and Arduino GUI. Anne Fouilloux. URL: https://annefou.github.io/IoT_introduction/02-ESP8266/index.html (date of access: 10.02.2024).

5. WiFi Плата NodeMCU V2 ESP8266 (CP2102) купити в Києві та Україні. Arduino в Україні. URL: https://arduino.ua/prod1495-wifi-plata-nodemcu-v2-esp8266-cp2102 (дата звернення: 10.02.2024).

6. Знайомимся з модулем ESP8266 докладніше | HobbyTech. HobbyTech. URL: https://hobbytech.com.ua/%D0%B7%D0%BD%D0%B0%D0%BA%D0%BE%D0%BC%D0%B8%D0%BC%D1%81%D1%8F-%D1%81-%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D0%B5%D0%BC-esp8266-%D0%BF%D0%BE%D0%B4%D1%80%D0%BE%D0%B1%D0%BD%D0%B5%D0%B5/ (дата звернення: 10.02.2024).

7. Tech Curry. NodeMCU ESP8266 WiFi Scanner with TFT display, 2020. YouTube. URL: https://www.youtube.com/watch?v=yxseln_edUA (date of access: 12.02.2024).

8. Wi-Fi модуль NodeMCU v3 на базі чіпа ESP8266 [#D-3]: продаж, ціна у Запоріжжі. Набори та компоненти для самостійного збирання електроніки від

«Ardu.prom.ua» 862113163. «Ardu.prom.ua» контакти, товари, послуги, ціни. URL: https://ardu.prom.ua/ua/p862113163-modul-nodemcu-baze.html (дата звернення: 12.02.2024).

9. Макетна плата безпаєчна MB-102 400 отворів купити в Києві та Україні. Arduino в Україні. URL: https://arduino.ua/prod218-maketnaya-plata-bespaechnaya-mb-102-400-tochek (дата звернення: 12.02.2024).

10. GPIO of ESP32 | ESP32. ElectronicWings - Hardware Developers Community. URL: https://www.electronicwings.com/esp32/gpio-of-esp32 (date of access: 12.02.2024).

11. Amazon.com. Amazon.com. URL: https://www.amazon.com/HiLetgo-ILI9225-176x220-Support-Arduino/dp/B00LSG5HI0 (date of access: 12.02.2024).

12. TFT LCD SPI 2.0 дюйми кольоровий графічний дисплей для ARDUINO. URL: https://3v3.com.ua/product_9650.html(date of access: 12.02.2024).

13. Getting Started with ILI9255 TFT LCD. Hackster.io. URL: https://www.hackster.io/Arnov_Sharma_makes/getting-started-with-ili9255-tft-lcd-378331 (date of access: 13.02.2024).

14. Remotely Control an LED with ESP8266 and MQTT. www.emqx.com. URL: https://www.emqx.com/en/blog/esp8266_mqtt_led (date of access: 13.02.2024).

15. Experiment 2: With the Touch of a Button1. esp8266book. URL: https://inquiryum.com/esp8266book/experiment2.html (date of access: 13.02.2024).

16. ESP8266 Piezo Buzzer | ESP8266 Tutorial. Tutorials for Newbies. URL: https://newbiely.com/tutorials/esp8266/esp8266-piezo-buzzer (date of access: 13.02.2024).

17. ESP8266 for IoT: A Complete Guide. Nabto. URL: https://www.nabto.com/esp8266-for-iot-complete-guide/ (date of access: 31.05.2024).

18. NodeMCU ESP8266 Specifications, Overview and Setting Up. Make-It.ca. URL: https://www.make-it.ca/nodemcu-details-specifications/ (date of access: 31.05.2024).

19. How to Use Wired Communication with an ESP8266. URL: https://www.digykey.com/en/maker/tutorials/2021/how-to-use-wired/comunication-with-en-esp8266 (date of access: 31.05.2024)

20. Introduction of Microprocessor - GeeksforGeeks. GeeksforGeeks. URL: http://www.geeksforgeeks.org/introduction-of-microprocessor/ (date of access: 05.06.2024)

ДОДАТКИ

Додаток А
Готове рішення для перевірки ESP8266

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <SPI.h>
#include <TFT_22_ILI9225.h>
//ALL ABOVE LIBRARIES ARE REQUIRED. INSTALL THEM FROM LIBRARY MANAGER
OR ADD THEM MANUALLY
#define TFT_RST D4
#define TFT_RS  D2
#define TFT_CS  D8
#define TFT_SDI D7
#define TFT_CLK D5
#define TFT_LED D1
TFT_22_ILI9225 tft = TFT_22_ILI9225(TFT_RST, TFT_RS, TFT_CS, TFT_LED);
const int RSSI_MAX =-50;
const int RSSI_MIN =-100;
const int displayEnc=1;
void setup() {
tft.begin();
Serial.begin(9600);
tft.fillRectangle(0, 0, tft.maxX(), tft.maxY(), COLOR_RED);
tft.drawRectangle(1, 1, tft.maxX() - 2, tft.maxY() - 2, COLOR_WHITE);
tft.setOrientation(3);
tft.setFont(Terminal12x16);
tft.setBackgroundColor(COLOR_RED);
WiFi.mode(WIFI_STA);
WiFi.disconnect();
delay(100);

}

void loop() {

  tft.drawRectangle(0, 0, tft.maxX() - 1, tft.maxY() - 1,
COLOR_WHITE);
  tft.fillRectangle(0, 0, tft.maxX() - 1, tft.maxY() - 1, COLOR_RED);
  tft.setFont(Terminal12x16);
  tft.setBackgroundColor(COLOR_RED);
  int n = WiFi.scanNetworks();
  tft.drawText(10, 40, "Searching For WiFi", COLOR_BLACK);
  delay(1000);
  tft.drawText(10, 80, "WiFi Scan Done", COLOR_WHITE);
  delay(1000);

  if (n == 0)
  {
```

```
    }
  }

}

  int dBmtoPercentage(int dBm)
{
  int quality;
    if(dBm <= RSSI_MIN)
    {
        quality = 0;
    }
    else if(dBm >= RSSI_MAX)
    {
        quality = 100;
    }
    else
    {
        quality = 2 * (dBm + 100);
   }

     return quality;
}
```

Додаток Б
Код для перевірки світлодіода через кнопку

```
int led = 2;
int pushButton = 5;

void setup() {
  pinMode(led, OUTPUT);       // pin 2 is an output pin
  pinMode(pushButton, INPUT_PULLUP); // pin 5 is an input --the button
}

void loop() {
  int pushButtonState;

  pushButtonState = digitalRead(pushButton);
  if (pushButtonState == LOW) {
    //we pushed the button
    digitalWrite(led, HIGH); //so turn on the LED
  }
  else // button not pressed
  {
    digitalWrite(led, LOW);
  }

}
```

ДОДАТОК В
Код для перевірки зумеру

```
# include  "pitches.h"
const    int  BUZZZER_PIN  =  D8;  //  Вивід  ESP8266,  підключений  до
п’єзозумера

// ноти в мелодії:
int melody[] = {
NOTE_C4 , NOTE_G3 , NOTE_G3 , NOTE_A3 , NOTE_G3 , 0, NOTE_B3 , NOTE_C4
};


// тривалість ноти: 4 = чверть, 8 = восьма нота тощо:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void  setup () {
// перейти до нот мелодії: for ( int thisNote = 0; thisNote < 8;
thisNote++) {


    // щоб обчислити тривалість ноти, одну секунду поділити на тип
ноти.
//наприклад, чверть = 1000 / 4, восьма нота = 1000/8 тощо. int
noteDuration = 1000 / noteDurations[thisNote]; тон (BUZZZER_PIN,
melody[thisNote], noteDuration);



    // щоб розрізнити ноти, встановити мінімальний час між ними.
// Тривалість нотатки + 30% виглядає добре: int pauseBetweenNotes =
noteDuration  *  1.30;  затримка  (pauseBetweenNotes);  //  зупинити
відтворення сигналу: noTone (BUZZZER_PIN);   } }




void  loop () {
// не потрібно повторювати мелодію. }
```

ДОДАТОК Г
Код для розробки проекту симуляції автомобілем на базі мікроконтроллера

```
Микола Михальчук, [28.05.2024 23:13]
#include <TFT_eSPI.h>
TFT_eSPI tft = TFT_eSPI();
TFT_eSprite sprite = TFT_eSprite(&tft);

//............INPUT PINS............switches and buttons
#define THROTTLE 43
#define BRAKE 44
#define LEFT  17  //direction pointer
#define RIGHT 18  //direction ponter
#define SHORT 16  //headlights
#define LONG  21 //headlights light
#define GEARUP 12
#define GEARDOWN 13
#define HORN 10
#define BRIGHTNESS 14

//..........OUTPUT PINS............lights, pointes, horn
#define left_pointer 1
#define right_pointer 11
#define head_lights  2
#define buzzer 3

//....................................colors
#define backColor 0x0026
#define gaugeColor 0x055D
#define dataColor 0x0311
#define purple 0xEA16
#define needleColor 0xF811

//..........................dont edit this
int cx=75;
int cy=75;
int r=72;
int ir=70;
int n=0;
int angle=0;

float x[360]; //outer points of Speed gaouges
float y[360];
float px[360]; //ineer point of Speed gaouges
float py[360];
float lx[360]; //text of Speed gaouges
float ly[360];
float nx[360]; //needle low of Speed gaouges
float ny[360];
float x2[360]; //outer points of RPM gaouges
float y2[360];
float px2[360]; //ineer point of RPM gaouges
float py2[360];
float lx2[360]; //text of RPM gaouges
float ly2[360];
float nx2[360]; //needle low of RPM gaouges
float ny2[360];
```

```
double rad=0.01745;
unsigned short color1;
unsigned short color2;
float sA;
float rA;
int blinkPeriod=500;
unsigned long currentTimeL=0;
unsigned long currentTimeR=0;
int brightnesses[5]={40,80,120,150,240};
int selectedBrightness=3;
int deb1=0;
int deb2=0;
int debB=0;

int gearMaxSpeed[8]={12,0,60,90,120,150,190,236};
String gears[8]={"R","N","1","2","3","4","5","6"};
int selectedGear=1;

//.....................................................colors
unsigned short blockColor[4]={0x0312,0x0290,0x01EC,0x016A};
unsigned short dirColor[2]={0x0312,TFT_ORANGE};
unsigned short lightColor[3]={0x01EC,0x0FA8,0xB79F};

// .........important variables
bool leftPointer=0;
bool rightPointer=0;
bool braking;
int lights=0;  //0 is lights off, 1 is short light, 2 is long lights
float speedAngle=0; //...speed variable 0-240
float rpmAngle=5;  //.....RPM variable 0-9

void setup() {
  pinMode(THROTTLE,INPUT_PULLUP);
  pinMode(BRAKE,INPUT_PULLUP);
  pinMode(LEFT,INPUT_PULLUP);
  pinMode(RIGHT,INPUT_PULLUP);
  pinMode(GEARUP,INPUT_PULLUP);
  pinMode(GEARDOWN,INPUT_PULLUP);
  pinMode(SHORT,INPUT_PULLUP);
  pinMode(LONG,INPUT_PULLUP);
  pinMode(HORN,INPUT_PULLUP);
  pinMode(BRIGHTNESS,INPUT_PULLUP);

 pinMode(left_pointer,OUTPUT);
 pinMode(right_pointer,OUTPUT);
 pinMode(head_lights,OUTPUT);
 pinMode( buzzer,OUTPUT);


  tft.init();

    tft.setRotation(1);
    tft.fillScreen(backColor);
    sprite.createSprite(320,150);
    sprite.setSwapBytes(true);
    sprite.setTextDatum(4);
    sprite.setTextColor(TFT_WHITE,backColor);
```

```
    sprite.setTextDatum(4);

     //ledcSetup(0, 10000, 8);
     //ledcAttachPin(38, 0);
     //ledcWrite(0, brightnesses[selectedBrightness]);  //brightnes of
screen


      //ledcSetup(1, 10000, 8);
     //ledcAttachPin(head_lights, 1);
     //ledcWrite(1, 10);



  int a=120;
  for(int i=0;i<360;i++)
    {
       x[i]=((r-10)*cos(rad*a))+cx;
       y[i]=((r-10)*sin(rad*a))+cy;
       px[i]=((r-14)*cos(rad*a))+cx;
       py[i]=((r-14)*sin(rad*a))+cy;
       lx[i]=((r-24)*cos(rad*a))+cx;
       ly[i]=((r-24)*sin(rad*a))+cy;
       nx[i]=((r-36)*cos(rad*a))+cx;
       ny[i]=((r-36)*sin(rad*a))+cy;
       x2[i]=((r-10)*cos(rad*a))+320-cx;
       y2[i]=((r-10)*sin(rad*a))+cy;
       px2[i]=((r-14)*cos(rad*a))+320-cx;
       py2[i]=((r-14)*sin(rad*a))+cy;
       lx2[i]=((r-24)*cos(rad*a))+320-cx;
       ly2[i]=((r-24)*sin(rad*a))+cy;
       nx2[i]=((r-36)*cos(rad*a))+320-cx;
       ny2[i]=((r-36)*sin(rad*a))+cy;

       a++;
       if(a==360)
       a=0;
    }

}



void draw()
{
  sprite.fillSprite(backColor);

   for(int i=0;i<4;i++)
   sprite.fillRect(120,28+i*24,80,22,blockColor[i]);

   for(int i=0;i<selectedBrightness;i++)
   sprite.fillSmoothRoundRect(8+(i*4),6,2,9,1,TFT_ORANGE,backColor);

Микола Михальчук, [28.05.2024 23:13]
sprite.fillSmoothCircle(cx, cy, r+2, backColor);
   sprite.fillSmoothCircle(320-cx, cy, r+2, backColor);
   sprite.fillTriangle(126,14,136,7,136,21,dirColor[leftPointer]);
//dirction pointers
```

```
   sprite.fillRect(136,11,8,7,dirColor[leftPointer]);

sprite.fillTriangle(126+68,14,136+48,7,136+48,21,dirColor[rightPointer
]);
   sprite.fillRect(176,11,8,7,dirColor[rightPointer]);


   for(int i=0;i<5;i++)
   {
   if(i<=2)
   sprite.fillRect(144+(7*i),36,5,5,TFT_WHITE);   /// fuel  rect
   else
   sprite.drawRect(144+(7*i),36,5,5,TFT_WHITE); //empty fuel rect
   }

    sprite.fillSmoothRoundRect(155,54,9,16,2,TFT_WHITE,blockColor[1]);
    sprite.fillSmoothRoundRect(166,56,2,14,2,TFT_WHITE,blockColor[1]);
    sprite.fillSmoothRoundRect(156,56,7,5,1,blockColor[1],TFT_WHITE);
    sprite.drawLine(153,69,166,69,TFT_WHITE);  //pumpimage

  sprite.drawSmoothArc(cx, cy, r, ir, 30, 330, gaugeColor, backColor);
  sprite.drawSmoothArc(320-cx,  cy,  r,  ir,  30,  330,  gaugeColor,
backColor);

  sprite.drawSmoothArc(cx,  cy,  r-5,  r-6,  30,  330,  TFT_WHITE,
backColor);
  sprite.drawSmoothArc(320-cx,  cy,  r-5,  r-6,  30,  330,  TFT_WHITE,
backColor);

  sprite.drawSmoothArc(cx, cy, r-9, r-8, 270, 330, purple, backColor);

  sprite.drawSmoothArc(cx,  cy,  r-38,  ir-37,  10,  350,  gaugeColor,
backColor);
  sprite.drawSmoothArc(320-cx, cy, r-38, ir-37, 10, 350, gaugeColor,
backColor);

   //.....................................................draw GAUGES
     for(int i=0;i<26;i++){
     if(i<20)     {color1=gaugeColor;     color2=TFT_WHITE;}     else
{color1=purple; color2=purple;}

  if(i%2==0) {
  sprite.drawWedgeLine(x[i*12],y[i*12],px[i*12],py[i*12],2,1,color1);
  sprite.setTextColor(color2,backColor);
  sprite.drawString(String(i*10),lx[i*12],ly[i*12]);
  }else
  sprite.drawWedgeLine(x[i*12],y[i*12],px[i*12],py[i*12],1,1,color2);
  }

      for(int i=0;i<19;i++){
      if(i<20)     {color1=gaugeColor;     color2=TFT_WHITE;}     else
{color1=purple; color2=purple;}

  if(i%2==0) {

sprite.drawWedgeLine(x2[i*16],y2[i*16],px2[i*16],py2[i*16],2,1,color1)
;
  sprite.setTextColor(color2,backColor);
```

```
  sprite.drawString(String(i/2),lx2[i*16],ly2[i*16]);
  }else

sprite.drawWedgeLine(x2[i*16],y2[i*16],px2[i*16],py2[i*16],1,1,color2)
;
  }

  // .....................................needles draw
   sA=speedAngle*1.2;
   rA=2*rpmAngle*1.6;

sprite.drawWedgeLine(px[(int)sA],py[(int)sA],nx[(int)sA],ny[(int)sA],2
,2,needleColor);

sprite.drawWedgeLine(px2[(int)rA],py2[(int)rA],nx2[(int)rA],ny2[(int)r
A],2,2,needleColor);


  //.....................................drawing  TEXT
  sprite.setTextColor(TFT_WHITE,backColor);
  sprite.drawString(String((int)speedAngle),cx,cy,4);
  sprite.drawString(String(gears[selectedGear]),320-cx,cy,4);
  sprite.drawString("KM/H",cx,cy+16);
  sprite.drawString("GEAR",320-cx,cy+16);

  sprite.setTextColor(TFT_WHITE,blockColor[3]);
  sprite.drawString("14356",160,110);

  sprite.setTextColor(TFT_ORANGE,backColor);
  sprite.drawString("x1000",320-cx,136);
  sprite.drawString("RPM",320-cx,126);

  //.....................................drawing BRAKE
  if(braking==true){
  sprite.drawSmoothArc(160, 10, 9, 8, 10, 350, TFT_RED, backColor);
  sprite.drawSmoothArc(160, 10, 12, 11, 50, 130, TFT_RED, backColor);
  sprite.drawSmoothArc(160, 10, 12, 11, 230, 310, TFT_RED, backColor);
  //sprite.drawCircle(320-cx,cy-20,6,TFT_RED);
  sprite.setTextColor(TFT_RED,backColor);
  sprite.drawString("!",160,12,2);}

  //.....................................drawing LIGHTs

sprite.fillSmoothRoundRect(152,82,14,10,7,lightColor[lights],lightColo
r[0]);
  sprite.fillRect(161,82,5,10,lightColor[0]);
  sprite.drawLine(163,82,167,84-lights,lightColor[lights]);
  sprite.drawLine(163,85,167,87-lights,lightColor[lights]);
  sprite.drawLine(163,88,167,90-lights,lightColor[lights]);
  sprite.drawLine(163,91,167,93-lights,lightColor[lights]);

   //.........................................draw DOT
     sprite.fillSmoothCircle(300, 10, 4,TFT_RED, backColor);

   //.............................................push    Sprite    to
screen
   sprite.pushSprite(0,10);
```

```
}

Микола Михальчук, [28.05.2024 23:13]
int blinking=1;

void loop() {

  if(digitalRead(SHORT)==0)
  lights=1;
  else if(digitalRead(LONG)==0)
  lights=2;
  else
  lights=0;

  //ledcWrite(1, lights*4);

  if(digitalRead(HORN)==0)
  digitalWrite(buzzer,1);
  else
  digitalWrite(buzzer,0);

  braking=!(digitalRead(BRAKE));



   if(digitalRead(BRIGHTNESS)==0)
  {
  if(debB==0)
    {
      debB=1;
      selectedBrightness++;
      if(selectedBrightness==5) selectedBrightness=0;
      //ledcWrite(0,  brightnesses[selectedBrightness]);    //brightnes
of screen
     }
  }else debB=0;


  if(digitalRead(GEARUP)==0)
  {
  if(deb1==0)
    {
      deb1=1;
      if(selectedGear<7)
      selectedGear++;
      if(speedAngle>10)
      speedAngle=speedAngle-4;
    }
  }else deb1=0;

    if(digitalRead(GEARDOWN)==0)
  {
  if(deb2==0)
    {
      deb2=1;
      if(selectedGear>0)
      selectedGear--;
      if(speedAngle>10)
```

```
      speedAngle=speedAngle-4;
    }
  }else deb2=0;

  if(digitalRead(LEFT)==0){
  if(millis()>currentTimeL+blinkPeriod)
  {
    leftPointer=!leftPointer;
    digitalWrite(left_pointer,leftPointer);
    currentTimeL=millis();
  }}else {leftPointer=0; digitalWrite(left_pointer,leftPointer);}

  if(digitalRead(RIGHT)==0){
  if(millis()>currentTimeR+blinkPeriod)
  {
    rightPointer=!rightPointer;
    digitalWrite(right_pointer,rightPointer);
    currentTimeR=millis();
  }}else {rightPointer=0;   digitalWrite(right_pointer,rightPointer);}

   if(braking==true && speedAngle>4)
   speedAngle=speedAngle-4;

   if(speedAngle<0)
   speedAngle=0;


  draw();

  if(digitalRead(THROTTLE)==0                                          &&
speedAngle<gearMaxSpeed[selectedGear])
  {speedAngle=speedAngle+2-(0.24*selectedGear);}

  if(digitalRead(THROTTLE)==1 && speedAngle>0)
  speedAngle--;

   if(digitalRead(THROTTLE)==0 && rpmAngle<75)
  {rpmAngle=rpmAngle+1-(0.1*selectedGear);}

   if(digitalRead(THROTTLE)==1 && rpmAngle>0)
  { if(rpmAngle>=3)
    rpmAngle=rpmAngle-3;
    else
    rpmAngle=0;
    }



}
```