МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет

імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри,
д-р техн. наук, проф.

_____ І. М. Журавська

« __ » _____ 2024 р.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Комплекс моніторингу стану пацієнта на базі ESP32

Спеціальність 123 Комп'ютерна інженерія

123 – КБР.01 – 405.22010608

Студент _____ М. В. Кайданович
підпис
«__» _____ 202__ р.

Керівник ст. викладач _____ Є. С. Дарнапук
підпис
«__» _____ 202__ р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Зав. кафедри _____ І. М. Журавська

«_____» _____ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної бакалаврської роботи

Видано студенту групи 405 факультету комп'ютерних наук

_____Кайдановичу Микиті Вячеславовичу _____
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

_____Комплекс моніторингу стану пацієнта на базі ESP32_____

Затверджена наказом по ЧНУ ім. Петра Могили від 30.01.2024 № 17.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Предметна сфера моніторингу стану здоров'я пацієнтів та технологій бездротових сенсорів, набір даних з параметрами життєдіяльності пацієнтів, технічні характеристики ESP32 та вимоги до програмного забезпечення для збору і обробки медичних даних

4. Перелік питань, що підлягають розробці
- аналіз предметної сфери моніторингу стану здоров'я пацієнтів та дослідження існуючих технологій бездротових сенсорів для медичних цілей;
- обґрунтування вибору технологій та засобів розробки комплексу моніторингу на базі мікроконтролера ESP32;
- розробка та здійснення програмної реалізації комплексу моніторингу стану пацієнта, включаючи підключення біомедичних сенсорів і обробку отриманих даних.

5. Перелік графічних матеріалів
презентація, рисунки, таблиці

6. Завдання до спеціальної частини
Проаналізувати вплив ергономічних та безпекових аспектів на роботу медичного персоналу при використанні даного комплексу, а також запропонувати рекомендації щодо зменшення професійних ризиків та підвищення ефективності роботи.

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
ст. викладач О. В. Макарова	Кафедра екології Медичного інституту ЧНУ ім. Петра Могили	Спеціальна частина з охорони праці

Керівник роботи
ст. викладач Дарнапук Євген Сергійович
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

Кайданович Микита Вячеславович
(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання «____» _____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: _____Комплекс моніторингу стану пацієнта на базі ESP32_____

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КР	11.012.2023	12.12.2023	Виконав
2	Огляд літератури за темою роботи	15.12.2023	15.01.2024	Виконав
3	Складання календарного плану БКР	16.01.2024	01.02.2024	Виконав
4	Аналіз предметної області	01.02.2024	01.03.2024	Виконав
5	Розробка проєктних рішень	01.03.2024	20.03.2024	Виконав
6	Моделювання та конструювання АПЗ	20.03.2024	20.04.2024	Виконав
7	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування	20.04.2024	01.05.2024	Виконав
8	Відгук керівника КР	06.06.2024	15.06.2024	Виконав
9	Оформлення БКР та презентації	01.05.2024	28.05.2024	Виконав
10	Перший попередній захист	28.05.2024	28.05.2024	Виконав
11	Другий попередній захист	05.06.2024	05.06.2024	Виконав
12	Рецензування	06.06.2024	15.06.2024	Виконав
13	Завершення оформлення КР та презентації	06.06.2024	15.06.2024	Виконав
14	Захист бакалаврської кваліфікаційної роботи	24.06.2024	26.06.2024	Виконав

Розробив здобувач ВО __Кайданович М. В.___ _____

(прізвище, ім'я, по батькові) *(підпис)*

«____» _____ 20__ р.

Керівник роботи _ст. викладач_Дарнапук Є. С.___ _____

(посада, прізвище, ім'я, по батькові) *(підпис)*

«____» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи
«Комплекс моніторингу стану пацієнта на базі ESP32»
Студент 405 гр.: Кайданович Микита Вячеславович
Керівник: ст. викладач Дарнапук Євген Сергійович

Кваліфікаційна бакалаврська робота присвячена розробці та здійсненню програмної реалізації комплексу моніторингу стану пацієнта на базі мікроконтролера ESP32. Це є актуальним, оскільки дослідження ефективних підходів до віддаленого моніторингу стану здоров'я пацієнтів сприяє підвищенню якості медичної допомоги в умовах зростання попиту на телемедичні послуги.

Об'єкт роботи – процес моніторингу стану здоров'я пацієнтів.

Предмет роботи – програмні засоби, методи та алгоритми моніторингу стану пацієнтів з використанням мікроконтролера ESP32.

Мета роботи – підвищення ефективності медичного нагляду за станом пацієнтів шляхом розробки комплексу моніторингу, який забезпечує збір, передачу та обробку біомедичних даних у реальному часі.

Кваліфікаційна бакалаврська кваліфікаційна робота складається з фахової та спеціальної частини з охорони праці. Пояснювальна записка фахової частини складається зі вступу, трьох розділів, висновків та додатків. У першому розділі розкрито можливості мікроконтролера Arduino ESP32-WROOM-32. У другому розділі подано специфікацію матеріалів кваліфікаційного проєкту. У третьому розділі описано налаштування Arduino ESP32-WROOM-32 як вебсервера.

Кваліфікаційна бакалаврська робота містить ___ сторінок (без додатків), ___ рисунків, ___ таблиць, ___ джерел посилання та ___ додатків.

***Ключові слова**: телемедицина, моніторинг здоров'я, біомедичні сенсори, ESP32, бездротові технології, обробка даних.*

ABSTRACT
of the Bachelor’s Thesis
"Patient Condition Monitoring System Based on ESP32"
Student: Kaidanovych Mykyta
Supervisor: Senior teacher Darnapuk Yevhen

This bachelor's thesis is dedicated to the development and implementation of a patient condition monitoring system based on the ESP32 microcontroller. This is relevant because researching effective approaches to remote health monitoring of patients contributes to improving the quality of medical care amidst the increasing demand for telemedicine services.

The object of the work is the process of monitoring patients' health conditions.

The subject of the work is the software tools, methods, and algorithms for monitoring patient conditions using the ESP32 microcontroller.

The goal of the work is to enhance the efficiency of medical supervision over patient conditions by developing a monitoring system that ensures the collection, transmission, and processing of biomedical data in real-time.

The bachelor's thesis consists of a professional part and a special part on occupational safety. The explanatory note of the professional part consists of an introduction, three chapters, conclusions, and appendices. The first chapter explores the capabilities of the Arduino ESP32-WROOM-32 microcontroller. The second chapter provides the specification of the qualification project's materials. The third chapter describes the configuration of the Arduino ESP32-WROOM-32 as a web server.

The bachelor's thesis contains ___ pages (excluding appendices), ___ figures, ___ tables, ___ references, and ___ appendices.

Keywords*: telemedicine, health monitoring, biomedical sensors, ESP32, wireless technologies, data processing.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 4
ВСТУП 5
1 АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА СКЛАДОВИХ СИСТЕМИ 7
1.1 Комплекси моніторингу стану пацієнта 7
1.2 Наявні альтернативні рішення 8
1.3 Огляд платформи Arduino ESP32-WROOM-32 та її характеристики 11
1.4 Відмінність ESP32-WROOM-32 від ESP32-DEVKITC 13
1.5 Процес створення вебсервера на базі Arduino 13
1.6 Типові застосування та проєкти з використанням ESP32-WROOM-32 15
1.7 Інтеграція ESP32-WROOM-32 з хмарними сервісами 17
1.8 Приклади успішних проєктів на базі ESP32-WROOM-32 19
1.9 Переваги та недоліки використання ESP32-WROOM-32 у проєктах IoT 22
Висновок до розділу 1 23
2 ВИМОГИ ТА ВИБІР ОБЛАДНАННЯ ДЛЯ КОМПЛЕКСУ МОНІТОРИНГУ СТАНУ ПАЦІЄНТА НА БАЗІ ESP32 25
2.1 Основні складові системи моніторингу стану пацієнта 25
2.2 Специфікація матеріалів кваліфікаційного проєкту 26
2.3 Датчик пульсоксиметра MAX30100 28
2.4 Датчик температури DS18B20 30
2.5 DHT11 Датчик вологості та температури 32
2.6 Вибір плати ESP32 34
2.7 Програмування та налагодження на платформі ESP32-WROOM-32 37
2.8 Архітектура системи 38

2.9 Алгоритм роботи системи 40
2.10 Тестування та валідація 41
2.11 Переваги та недоліки системи моніторингу стану пацієнта на базі ESP32 42
Висновок до розділу 2 43
3 РОЗРОБКА АПАРАТНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ... 44
3.1 Розробка апаратного забезпечення 44
3.2 Мова програмування Arduino 48
3.3 Arduino IDE 49
3.4 Програмування системи моніторингу стану пацієнта 51
3.5 Налаштування arduino esp32-wroom-32 як вебсервера 56
3.6 Приклади використання Arduino ESP32-WROOM-32 як вебсервера 57
3.7 Код реалізації та результат роботи проєкту 58
3.8 Інтеграція комплексу моніторингу стану пацієнта в інших галузях 65
Висновок до розділу 3 66
ВИСНОВКИ 68
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 70
ДОДАТОК А Довідка про перевірку на унікальність пояснювальної записки 73
ДОДАТОК Б Код програми 74
ДОДАТОК В Графічні матеріали КБР 82

ПЕРЕЛІК СКОРОЧЕНЬ

BPM – Heart Rate/Pulse

BT – Body Temperature

RH – Room Humidity

RT – Room Temperature

SpO2 – Blood Oxygen Level

ВСТУП

У сучасному світі Інтернет речей (IoT) швидко змінює різні галузі, включаючи охорону здоров'я. Новітні стартапи в медичних технологіях пропонують рішення, які полегшують моніторинг здоров'я пацієнтів, що є особливо важливим у контексті надання медичної допомоги вдома. Відстежування стану здоров'я пацієнта вдома стає дедалі складнішим завданням через щільний графік і повсякденну роботу як медичних працівників, так і пацієнтів. Особливо це актуально для пацієнтів похилого віку, які потребують постійного нагляду.

Багато пацієнтів потребують регулярного моніторингу життєво важливих показників, таких як частота серцевих скорочень, рівень кисню в крові та температура тіла. Традиційні методи моніторингу передбачають часті візити до лікаря або використання громіздких стаціонарних пристроїв, що може бути незручним і стресовим для пацієнтів. У таких умовах виникає необхідність у розробці портативних, ефективних та автоматизованих систем моніторингу, які могли б забезпечити віддалене спостереження за пацієнтами.

Мета роботи: вивчення технічних можливостей та практична реалізація вебсервера на мікроконтролері Arduino ESP32-WROOM-32 для підвищення ефективності та функціональності відстежування стану здоров'я пацієнта вдома. Основні завдання включають розробку програмного забезпечення для забезпечення взаємодії з мережею Інтернет, створення інтерфейсу користувача та обробку HTTP-запитів.

Об'єкт дослідження: система моніторингу стану здоров'я пацієнтів на основі мікроконтролера ESP32, яка включає різноманітні сенсори для збору життєво важливих даних, таких як температура, вологість, частота серцевих скорочень і рівень кисню в крові.

Предмет дослідження: процеси та методи інтеграції сенсорів з мікроконтролером ESP32 для створення портативного пристрою, який

забезпечує автоматизоване відстеження стану здоров'я пацієнтів та передачу зібраних даних на вебсервер для віддаленого моніторингу.

Завдання роботи:

- розробка програмного забезпечення для мікроконтролера ESP32, яке забезпечує підключення до мережі Інтернет та обробку даних від сенсорів;
- інтеграція сенсорів для вимірювання температури, вологості, частоти серцевих скорочень та рівня кисню в крові з мікроконтролером ESP32;
- створення вебсервера для зберігання та відображення зібраних даних у реальному часі;
- розробка інтерфейсу користувача для доступу до зібраних даних через веббраузер;
- тестування та налагодження системи для забезпечення стабільної роботи та точності вимірювань.

Отже, розробка на базі Arduino ESP32-WROOM-32 пропонує інноваційну систему, яка автоматизує завдання моніторингу стану здоров'я пацієнта. Цей пристрій забезпечує інтелектуальну систему відстеження життєво важливих параметрів пацієнта за допомогою вебсервера. Вона дозволяє медичним працівникам та родичам пацієнта отримувати доступ до даних про частоту серцевих скорочень, рівень кисню в крові та температуру тіла у реальному часі через зручний вебінтерфейс. Це значно покращує якість догляду за пацієнтами, знижує навантаження на медичний персонал та забезпечує пацієнтам більший комфорт і безпеку.

1 АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА СКЛАДОВИХ СИСТЕМИ

1.1 Комплекси моніторингу стану пацієнта

Комплекси моніторингу стану пацієнта стали невід'ємною частиною сучасної медицини, забезпечуючи постійний контроль за важливими показниками здоров'я. Ці системи використовуються в лікарнях, клініках і навіть у домашніх умовах для постійного спостереження за станом пацієнтів з хронічними захворюваннями, післяопераційними станами та іншими медичними проблемами. Комплекс моніторингу стану пацієнта на базі ESP32 є одним із таких рішень, що забезпечує гнучкість і доступність моніторингу завдяки використанню сучасних технологій.

Використання традиційних методів моніторингу, таких як періодичні візити до лікаря або використання автономних медичних пристроїв, має певні недоліки. По-перше, ці методи часто вимагають присутності медичного персоналу для збору і аналізу даних, що може бути незручним і дорогим. По-друге, такі методи не забезпечують постійний контроль, що може бути критично важливим для пацієнтів з нестабільним станом здоров'я. Використання комплексів моніторингу на базі мікроконтролерів, таких як ESP32, дозволяє вирішити ці проблеми, забезпечуючи постійний збір, передачу і аналіз даних у режимі реального часу.

Однією з основних переваг використання ESP32 є його можливості бездротового зв'язку. Цей мікроконтролер підтримує Wi-Fi і Bluetooth, що дозволяє легко інтегрувати його в існуючі медичні системи і забезпечити передачу даних до хмарних сервісів для подальшого аналізу. Крім того, ESP32 має високу продуктивність і низьке енергоспоживання, що робить його ідеальним вибором для мобільних і автономних пристроїв моніторингу.

1.2 Наявні альтернативні рішення

На сьогоднішній день на ринку існує багато різних систем моніторингу стану пацієнта, кожна з яких має свої особливості і переваги. Одним із популярних рішень є використання портативних моніторів життєвих показників, які збирають дані про пульс, артеріальний тиск, рівень кисню в крові та інші параметри. Ці пристрої зазвичай мають вбудовані дисплеї для відображення інформації і можуть зберігати дані для подальшого аналізу.

Holter Monitor – це портативний пристрій для безперервного моніторингу серцевої діяльності протягом 24–48 годин. Він використовується для виявлення аритмій, які можуть бути пропущені під час короткого ЕКГ-дослідження в клініці. Holter Monitor дозволяє лікарям аналізувати дані і виявляти аномалії в ритмі серця.

Рисунок 1.1 – Holter Monitor

Таблиця 1.1 – Переваги та недоліки Holter Monitor

Переваги	Недоліки
Безперервний моніторинг серцевої діяльності	Обмежений час моніторингу (24-48 годин)
Висока точність даних.	Потреба у спеціалізованому аналізі даних лікарем
Можливість виявлення прихованих аритмій	

AliveCor KardiaMobile – це портативний ЕКГ-монітор, який дозволяє знімати ЕКГ в домашніх умовах за допомогою смартфона. Пристрій підключається до мобільного додатку, який зберігає і аналізує дані, надаючи можливість поділитися ними з лікарем.

Рисунок 1.2 – ЕКГ-монітор KardiaMobile

Таблиця 1.2 – Переваги та недоліки KardiaMobile

Переваги	**Недоліки**
Зручність використання вдома.	Обмежена функціональність у порівнянні зі стаціонарними системами.
Швидкий і легкий доступ до даних ЕКГ.	Залежність від смартфона і мобільного додатку.
Можливість обміну даними з лікарем.	

Однак, такі пристрої мають обмеження щодо обсягу зібраних даних і можливостей їхнього аналізу. Більшість з них не забезпечують автоматичну передачу даних до хмарних сервісів або медичних інформаційних систем, що

ускладнює їхнє використання для постійного моніторингу. Крім того, портативні монітори часто мають обмежений час автономної роботи, що може бути критичним у випадках, коли постійний моніторинг є необхідним.

Іншою альтернативою є використання стаціонарних систем моніторингу, які зазвичай встановлюються в лікарнях і клініках. Ці системи забезпечують високу точність вимірювань і можуть бути інтегровані з іншими медичними системами для автоматичного збору і аналізу даних.

Philips IntelliVue – це стаціонарна система моніторингу, яка використовується в лікарнях для безперервного спостереження за життєво важливими показниками пацієнтів. Система може моніторити ЕКГ, артеріальний тиск, рівень кисню в крові, частоту дихання та інші параметри.

Рисунок 1.3 – Philips IntelliVue

Таблиця 1.3 – Переваги та недоліки Philips IntelliVue

Переваги	**Недоліки**
Висока точність і надійність.	Висока вартість обладнання.
Інтеграція з лікарняними інформаційними системами.	Обмежене використання поза межами лікарень.
Широкий спектр вимірюваних параметрів.	

Однак, такі системи є дорогими і не завжди доступними для використання в домашніх умовах або в невеликих медичних закладах. Вартість таких систем часто є бар'єром для їх впровадження в невеликих клініках або для використання в домашніх умовах. Крім того, їх використання вимагає спеціалізованого навчання медичного персоналу і постійного технічного обслуговування. Це створює додаткові витрати і складності для медичних закладів, особливо у регіонах з обмеженими фінансовими ресурсами.

З огляду на ці обмеження, розробка портативних та більш доступних систем моніторингу стає все більш актуальною. Використання мікроконтролерів, таких як ESP32, дозволяє створювати компактні і дешеві рішення, які можуть забезпечити достатню точність вимірювань для використання як в медичних закладах, так і в домашніх умовах. Такі системи можуть бути легко інтегровані з мобільними додатками і хмарними сервісами для зберігання і аналізу даних, що відкриває нові можливості для дистанційного моніторингу пацієнтів і телемедицини

1.3 Огляд платформи Arduino ESP32-WROOM-32 та її характеристики

ESP32-WROOM-32 – це потужна мікроконтролерна платформа, яка використовує чіп ESP32 від Espressif Systems. Ця плата розробки забезпечує широкі можливості для створення інтернет-підключених пристроїв та систем. Ось основні характеристики цієї платформи:

− мікроконтролер ESP32-WROOM-32 має вбудований мікроконтролер ESP32, який забезпечує високу продуктивність та широкі можливості програмування;

– вбудований Wi-Fi та Bluetooth модуль: однією з ключових переваг цієї платформи є вбудований модуль Wi-Fi та Bluetooth, що дозволяє

підключатися до бездротових мереж та взаємодіяти з іншими пристроями через Bluetooth;

– компактний розмір: ESP32-WROOM-32 має компактні розміри, що робить її ідеальним вибором для проєктів з обмеженим простором;

– різноманітність портів та інтерфейсів: незважаючи на свій компактний розмір, ця плата має багато вбудованих портів та інтерфейсів, включаючи цифрові та аналогові входи/виходи, інтерфейси I2C та UART;

– живлення: ESP32-WROOM-32 може живитися від USB або зовнішнього джерела живлення з напругою від 5 В до 9 В, що робить її досить гнучкою у використанні;

– простота використання: плата сумісна з Arduino IDE та має простий інтерфейс програмування, що робить її доступною для широкого кола розробників, включаючи початківців.

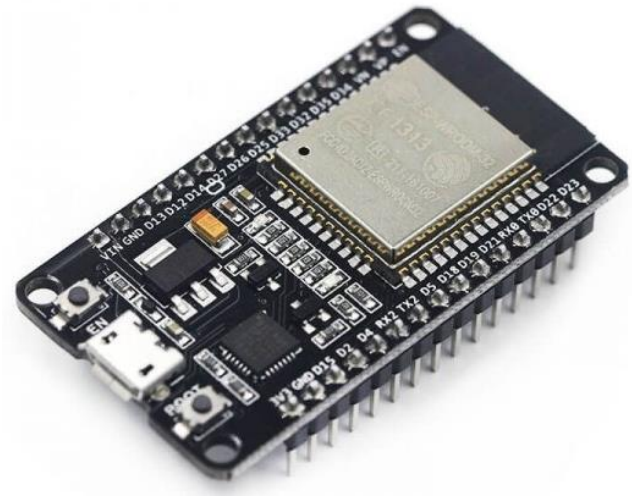


Рисунок 1.4 – ESP32-WROOM-32

ESP32-WROOM-32 – це не лише простий мікроконтролер, але і потужна платформа для творчих інженерних рішень. З її вбудованим Wi-Fi модулем, компактним розміром і різноманіттям функціональностей вона є ідеальним вибором для розробки інтернет-підключених пристроїв. Простота в використанні, широкий спектр доступних бібліотек та активна спільнота розробників роблять ESP32-WROOM-32 першим вибором для інженерів у всьому світі. Від простих проєктів домашньої автоматизації до складних

систем Інтернету речей – ESP32-WROOM-32 відкриває безмежні можливості для творчості та інновацій.

1.4 Відмінність ESP32-WROOM-32 від ESP32-DEVKITC

ESP32-WROOM-32 та ESP32-DEVKITC – це два популярні варіанти для роботи з мікроконтролером ESP32, що різняться за функціоналом та призначенням.

ESP32-WROOM-32 – це компактний модуль Wi-Fi + Bluetooth, який містить сам мікроконтролер ESP32 та радіочастотні компоненти. Це ідеальний вибір для досвідчених розробників, які самостійно проєктують схеми та пишуть код, а також для кінцевих продуктів завдяки компактності та низькій вартості. Він працює на напрузі 3.3 В, має вбудовану пам'ять Flash ємністю до 4 Мбайт.

ESP32-DEVKITC – це плата розробки на базі ESP32, яка доповнює ESP32-WROOM-32 корисними компонентами для простоти використання. Вона має USB-UART перетворювач для зручного програмування та відладки, кнопку скидання для перезавантаження, світлодіоди живлення та статусу, а також роз'єми для підключення сенсорів, дисплеїв та іншої периферії. Це робить її ідеальним вибором як для початківців, так і для досвідчених розробників, які шукають простоту в прототипуванні та розробці. ESP32-DEVKITC також працює на напрузі 3.3 В та має вбудовану пам'ять Flash ємністю до 16 Мбайт.

1.5 Процес створення вебсервера на базі Arduino

Створення вебсервера на базі Arduino відкриває необмежені можливості для розробки різноманітних проєктів, які потребують інтерактивного взаємодії з користувачем через Інтернет. Завдяки широкому спектру функцій та різноманіттю доступних модулів, платформа Arduino дозволяє створювати вебсервери для вирішення різноманітних завдань – від керування вбудованими пристроями до збору та обробки даних.

Розглянемо процес створення вебсервера на базі Arduino та основні кроки для його реалізації:

– налаштування мережевого з'єднання: налаштування підключення до мережі, такої як Wi-Fi або Ethernet, відповідно до ваших потреб та вимог проєкту;

– конфігурація HTTP–сервера: потім необхідно налаштувати http–сервер на пристрої Arduino. Це включає визначення URL–адрес для обслуговування, а також визначення дій, які потрібно виконати у відповідь на отримані запити;

– обробка запитів та формування відповідей: розробка коду для обробки вхідних запитів та формування відповідей. Це може включати зчитування даних з датчиків або управління пристроями згідно з отриманими командами;

– розробка вебінтерфейсу: нарешті, розробка вебінтерфейсу, який буде відображатися користувачеві через веббраузер. Це може бути проста HTML–сторінка або більш складний інтерфейс з використанням JavaScript та CSS;

– забезпечення безпеки: додавання заходів безпеки для захисту від несанкціонованого доступу до вашого вебсервера, такі як аутентифікація користувачів або шифрування даних.

Створення вебсервера на базі Arduino відкриває великі можливості для інтерактивних інтернет-проєктів. Процес включає налаштування мережевого з'єднання, конфігурацію HTTP-сервера, обробку запитів, розробку вебінтерфейсу та забезпечення безпеки. Завдяки гнучкості та функціоналу платформи Arduino, можна легко створювати проєкти для автоматизації та дистанційного керування.

1.6 Типові застосування та проєкти з використанням ESP32-WROOM-32

ESP32-WROOM-32 широко використовується у різних проєктах, від домашніх автоматизацій до промислових систем моніторингу. Ось кілька типових прикладів:

Домашня автоматизація:

ESP32-WROOM-32 дозволяє створювати розумні пристрої для дому, такі як розумні лампи, термостати, системи безпеки та контролю доступу. Завдяки вбудованим модулям Wi-Fi та Bluetooth, ці пристрої можуть легко інтегруватися в існуючі мережі та керуватися віддалено через інтернет.

ESP32-WROOM-32 може використовуватись для створення розумних освітлювальних приладів, які можна керувати через мобільний додаток або голосові команди. Наприклад, система може автоматично вмикати та вимикати світло залежно від часу доби або наявності людей в кімнаті.

Розумний термостат на базі ESP32-WROOM-32 може регулювати температуру в домі залежно від заданих параметрів, а також віддалено керуватися через інтернет. Такий пристрій може використовувати дані про погоду з інтернету для оптимального управління опаленням або охолодженням.

Моніторинг довкілля:

З використанням датчиків температури, вологості, якості повітря та інших параметрів ESP32-WROOM-32 може бути використаний для моніторингу навколишнього середовища в реальному часі. Це корисно для розумних теплиць, метеостанцій та систем контролю якості повітря.

ESP32-WROOM-32 може використовуватись у системах, які відстежують рівень забруднення повітря в приміщеннях або на вулиці. Такі системи можуть повідомляти користувача про небезпечні рівні забруднення та рекомендувати дії для покращення якості повітря.

У розумній теплиці на базі ESP32-WROOM-32 можна використовувати датчики для моніторингу температури, вологості, рівня освітлення та інших параметрів. Система може автоматично керувати поливом, освітленням та вентиляцією для створення оптимальних умов для вирощування рослин.

Індустріальні IoT-додатки:

У промислових умовах ESP32-WROOM-32 може бути застосований для моніторингу обладнання, збору даних із датчиків, управління виробничими процесами та передиктивного обслуговування. Це сприяє підвищенню ефективності та зниженню витрат на обслуговування.

ESP32-WROOM-32 можна використовувати для відстеження стану виробничого обладнання в реальному часі. Система може збирати дані про температуру, вібрації, рівень зношування та інші параметри, що дозволяє своєчасно виявляти потенційні несправності та проводити профілактичне обслуговування.

Управління виробничими процесами на базі ESP32-WROOM-32 включає автоматизацію різних етапів виробництва, збирання даних про виконання процесів та їх оптимізацію. Це дозволяє підвищити продуктивність, знизити витрати та забезпечити стабільну якість продукції.

Здоров'я та фітнес:

Пристрої на базі ESP32-WROOM-32 можуть використовуватися в гаджетах, що носяться, таких як фітнес-трекери і розумний годинник, для моніторингу активності, серцевого ритму та інших показників здоров'я.

Фітнес-трекери на базі ESP32-WROOM-32 можуть відстежувати кількість пройдених кроків, витрачені калорії, частоту серцевих скорочень та інші показники. Дані можуть передаватися на смартфон користувача для подальшого аналізу та створення персональних тренувальних програм.

Розумні годинники, оснащені ESP32-WROOM-32, можуть відстежувати різні параметри здоров'я, такі як сон, фізична активність, стрес та серцевий ритм. Годинник може також використовуватись для віддаленого керування іншими пристроями та отримання сповіщень з мобільного телефону.

1.7 Інтеграція ESP32-WROOM-32 з хмарними сервісами

Однією з ключових переваг використання ESP32-WROOM-32 є можливість інтеграції з різними хмарними сервісами. Це дозволяє розширити функціональність пристроїв та надати додаткові можливості для аналізу та обробки даних.

Google Cloud IoT Core:

ESP32-WROOM-32 може бути підключений до Google Cloud IoT Core для збору даних, їх аналізу та візуалізації в реальному часі. Це дозволяє розробникам створювати масштабовані IoT-рішення з використанням потужних інструментів Google Cloud.

Процес інтеграції:

– налаштування облікового запису: створення проєкту у Google Cloud Platform (GCP) та активація Google Cloud IoT Core;

– реєстрація пристрою: реєстрація ESP32-WROOM-32 в Google Cloud IoT Core, отримання сертифікатів безпеки та налаштування аутентифікації;

– збір даних: використання бібліотек Google Cloud IoT Core для ESP32, які дозволяють збирати дані з датчиків і відправляти їх до хмарного сервісу;

– аналіз даних: використання Google BigQuery для аналізу великих обсягів даних або Google Data Studio для створення візуалізацій.

Приклад використання:

Моніторинг здоров'я пацієнтів: Дані з сенсорів, що вимірюють життєві показники пацієнтів, можуть бути зібрані ESP32-WROOM-32 та передані до Google Cloud IoT Core. Це дозволяє лікарям відстежувати стан пацієнтів у реальному часі та приймати обґрунтовані рішення на основі аналітики даних.

Amazon Web Services (AWS):

Інтеграція з AWS дозволяє використовувати такі сервіси, як AWS IoT, для керування пристроями, збирання та аналізу даних, а також автоматизації різних процесів. Це забезпечує високу надійність та масштабованість рішень на базі ESP32-WROOM-32.

Індустріальний моніторинг: Збір даних з виробничих машин за допомогою ESP32-WROOM-32 та передача їх до AWS IoT Core для аналізу та прогнозування відмов обладнання. Це допомагає мінімізувати простої та оптимізувати обслуговування.

Microsoft Azure IoT Hub:

ESP32-WROOM-32 також може бути інтегрований з Microsoft Azure IoT Hub, що дозволяє створювати комплексні рішення для управління IoT-пристроями, аналізу даних та побудови звітів. Azure надає широкий спектр інструментів для роботи з великими даними та штучним інтелектом.

Розумна лікарня: Дані з медичних приладів у лікарні можуть бути зібрані за допомогою ESP32-WROOM-32 та передані до Azure IoT Hub. Це дозволяє відстежувати стан пацієнтів, оптимізувати розклад прийомів та прогнозувати навантаження на медичний персонал.

IBM Watson IoT:

Використання IBM Watson IoT дозволяє підключати пристрої на базі ESP32-WROOM-32 до платформи IBM для аналізу даних із використанням штучного інтелекту та машинного навчання. Це відкриває нові можливості для створення розумних та адаптивних систем.

Моніторинг пацієнтів вдома: Система може збирати дані про стан пацієнтів за допомогою носимих пристроїв на базі ESP32-WROOM-32 та передавати їх до IBM Watson IoT для аналізу. Це дозволяє лікарям відстежувати стан пацієнтів у реальному часі та надавати дистанційну підтримку.

1.7.1 Переваги та виклики інтеграції ESP32-WROOM-32 з хмарними сервісами

Переваги:

– хмарні сервіси дозволяють легко масштабувати рішення, додаючи нові пристрої та обробляючи більші обсяги даних;

– використання хмарних сервісів забезпечує високий рівень безпеки даних, включаючи шифрування та контроль доступу;

– можливість використовувати потужні інструменти для аналізу даних та побудови моделей машинного навчання;

– хмарні платформи надають зручні інтерфейси для управління пристроями, моніторингу стану системи та налаштування параметрів.

Виклики:

– використання хмарних сервісів може спричиняти затримки в передачі даних, що може бути критичним для реального часу застосувань;

– система залежить від стабільного інтернет-з'єднання для передачі даних до хмари;

– використання хмарних сервісів може бути дорогим, особливо при обробці великих обсягів даних або високих вимогах до обчислювальних ресурсів.

Інтеграція ESP32-WROOM-32 з хмарними сервісами має свої переваги та виклики. Переваги включають легке масштабування, високий рівень безпеки, потужні інструменти для аналізу даних та зручні інтерфейси управління. Водночас, виклики полягають у можливих затримках передачі даних, залежності від стабільного інтернет-з'єднання та високих витратах на використання хмарних сервісів.

1.8 Приклади успішних проєктів на базі ESP32-WROOM-32

ESP32-WROOM-32 знайшов застосування у різних успішних проєктах, що демонструють його потенціал та гнучкість. Ось кілька прикладів:

Розумний будинок:

В одному з проєктів ESP32-WROOM-32 був використаний для створення системи розумного будинку, що включає управління освітленням, опаленням та системою безпеки. Користувачі могли керувати всіма пристроями через мобільний додаток або голосові команди.

Деталі проєкту:

– освітлення: користувачі можуть керувати інтенсивністю та кольором світла в приміщенні за допомогою мобільного додатка або голосових команд. Система автоматично регулює освітлення в залежності від часу доби та присутності людей у кімнаті;

– опалення: за допомогою сенсорів температури ESP32-WROOM-32 регулює роботу обігрівачів для підтримання комфортної температури в будинку, зменшуючи споживання енергії;

– безпека: система безпеки включає датчики руху, камери та дверні замки, які контролюються через мобільний додаток. У разі виявлення підозрілої активності користувачі отримують сповіщення на свої пристрої.

Результати:

– зменшення споживання електроенергії на 30 % завдяки автоматизації;

– підвищення рівня безпеки та комфорту мешканців;

– зручність управління системою через мобільний додаток та голосові команди.

Система моніторингу здоров'я:

Інший проєкт включав створення пристрою для моніторингу здоров'я пацієнтів. За допомогою ESP32-WROOM-32 пристрій збирав дані про серцевий ритм, рівень кисню в крові та інші показники, передаючи їх у реальному часі лікарям для аналізу та прийняття рішень.

Деталі проєкту:

– датчики здоров'я: використовуються датчики для вимірювання серцевого ритму, рівня кисню в крові та температури тіла;

– збір та передача даних: ESP32-WROOM-32 збирає дані з датчиків і передає їх через Wi-Fi або Bluetooth до хмарного сервісу для зберігання та аналізу;

– аналіз даних: лікарі мають доступ до зібраних даних у реальному часі через спеціальний вебінтерфейс або мобільний додаток, що дозволяє швидко реагувати на зміни стану пацієнтів.

Результати:

– підвищення точності моніторингу здоров'я пацієнтів;

– зменшення кількості необхідних візитів до лікаря;

– можливість надання негайної медичної допомоги у випадку екстрених ситуацій.

Автоматизація сільського господарства:

У проєкті з автоматизації сільського господарства ESP32-WROOM-32 використовувався для моніторингу умов у теплицях та управління системами поливу. Це дозволило значно підвищити врожайність та знизити витрати на водні ресурси.

Деталі проєкту:

– моніторинг умов: використання датчиків для вимірювання вологості ґрунту, температури та рівня освітлення в теплицях;

– управління поливом: ESP32-WROOM-32 контролює роботу систем поливу, автоматично регулюючи подачу води в залежності від умов ґрунту та погоди;

– дистанційне управління: фермери можуть контролювати та налаштовувати системи через мобільний додаток, отримуючи сповіщення про стан рослин та необхідність вжити заходів.

Результати:

– підвищення врожайності на 25% завдяки оптимізованому поливу;

– зменшення витрат на водні ресурси на 40%;

– зручність дистанційного управління та моніторингу.

ESP32-WROOM-32 успішно використовується в різних проєктах, таких як розумний будинок, система моніторингу здоров'я та автоматизація сільського господарства. Ці проєкти демонструють його гнучкість,

ефективність в енергоспоживанні, точність діагностики та підвищення врожайності.

1.9 Переваги та недоліки використання ESP32-WROOM-32 у проєктах IoT

Використання ESP32-WROOM-32 у проєктах IoT має свої переваги та недоліки.

Переваги:

– висока продуктивність: два ядра, що працюють на частоті до 240 МГц, дозволяють обробляти складні завдання та виконувати їх паралельно. Це особливо корисно для IoT-застосувань, де обробка даних у реальному часі є критично важливою;

– інтегровані модулі Wi-Fi та Bluetooth: наявність вбудованих модулів Wi-Fi та Bluetooth значно спрощує процес розробки та знижує вартість проєкту. Це дозволяє легко забезпечити бездротове підключення пристроїв до мережі або взаємодію з іншими Bluetooth-пристроями;

– компактний розмір: малий розмір ESP32-WROOM-32 робить його ідеальним вибором для проєктів з обмеженим простором, таких як носимі пристрої, сенсорні мережі та інші компактні IoT-рішення;

– низьке енергоспоживання: підтримка режимів енергозбереження дозволяє використовувати ESP32-WROOM-32 в автономних проєктах, таких як системи моніторингу, що працюють на батареях. Це особливо важливо для довготривалих IoT-додатків, де збереження енергії є пріоритетом;

– гнучкість і розширюваність: ESP32-WROOM-32 підтримує різні інтерфейси, такі як GPIO, SPI, I2C, UART та інші, що дозволяє легко підключати додаткові периферійні пристрої та модулі.

Недоліки:

– складність налаштування: для ефективної роботи з ESP32-WROOM-32 потрібно витратити деякий час на вивчення та налаштування середовища

розробки та інструментів. Початківці можуть зіткнутися з труднощами під час першого налаштування та запуску;

– проблеми з сумісністю: деякі бібліотеки або програми можуть не бути сумісними з ESP32-WROOM-32 або вимагати додаткових налаштувань для коректної роботи. Це може ускладнити розробку та інтеграцію проєктів;

– відсутність апаратного захисту: ESP32-WROOM-32 не має вбудованих апаратних засобів захисту, таких як безпечне зберігання ключів або шифрування даних. Це може створити ризики безпеки в критичних додатках, де конфіденційність і цілісність даних є важливими.

Використання ESP32-WROOM-32 в проєктах IoT має значні переваги, такі як висока продуктивність, інтегровані модулі Wi-Fi та Bluetooth, компактний розмір, низьке енергоспоживання та гнучкість у підключенні додаткових пристроїв. Однак, є і недоліки, включаючи складність налаштування, можливі проблеми з сумісністю деяких бібліотек і відсутність вбудованих апаратних засобів захисту.

Висновок до розділу 1

Комплекси моніторингу стану пацієнта стали критично важливими в сучасній медицині, забезпечуючи постійний контроль життєво важливих показників. Зокрема, системи на базі мікроконтролерів, таких як ESP32, пропонують ефективне вирішення для безперервного спостереження за пацієнтами, завдяки їх високій продуктивності, низькому енергоспоживанню, і можливостям бездротового зв’язку.

Традиційні методи моніторингу, хоча і надійні, часто потребують регулярного втручання медичного персоналу і не забезпечують постійний контроль, що може бути небезпечним для пацієнтів з нестабільним станом здоров’я. У порівнянні з цим, системи на базі ESP32 дозволяють постійно збирати і передавати дані в режимі реального часу, що покращує якість медичного обслуговування.

Таким чином, використання ESP32-WROOM-32 у медичних системах моніторингу відкриває нові можливості для дистанційного контролю пацієнтів, телемедицини, та поліпшення загальної якості медичного обслуговування. Ця платформа забезпечує не лише технічні переваги, але й робить медичні технології більш доступними та ефективними, що є важливим фактором для сучасної медицини.

2 ВИМОГИ ТА ВИБІР ОБЛАДНАННЯ ДЛЯ КОМПЛЕКСУ МОНІТОРИНГУ СТАНУ ПАЦІЄНТА НА БАЗІ ESP32

2.1 Основні складові системи моніторингу стану пацієнта

Комплекс моніторингу стану пацієнта є інтегрованою системою, призначеною для безперервного спостереження за різними фізіологічними параметрами пацієнта. Основна перевага такого комплексу полягає в його здатності збирати, аналізувати та передавати дані в режимі реального часу, що дозволяє медичному персоналу своєчасно реагувати на зміни стану пацієнта.

До складу системи входять сенсори життєво важливих параметрів, які вимірюють артеріальний тиск, частоту серцевих скорочень, рівень кисню в крові (пульсоксиметри), температуру тіла та рівень глюкози в крові. Ці сенсори можуть бути як носимими, так і стаціонарними. Важливою складовою комплексу є електрокардіографи (ЕКГ), що забезпечують моніторинг електричної активності серця, допомагаючи виявляти аритмії, ішемію та інші серцеві порушення. ЕКГ можуть бути портативними або стаціонарними, що дозволяє використовувати їх у різних умовах.

Капнографи, які входять до комплексу, вимірюють концентрацію вуглекислого газу в диханні пацієнта, що дозволяє моніторити вентиляційну функцію легень та виявляти дихальні порушення. Системи моніторингу центральної нервової системи, зокрема електроенцефалографи (ЕЕГ), дозволяють оцінювати електричну активність мозку та виявляти епілепсію або інші неврологічні стани.

Також комплекс включає системи для моніторингу дихання, які спостерігають за дихальними параметрами, такими як частота дихання та об’єм легень. Це можуть бути респіратори або простіше обладнання, що аналізує дихальні цикли. Інфузійні насоси забезпечують точне введення медикаментів, рідин та поживних речовин в організм пацієнта, що дозволяє точно контролювати швидкість і об'єм введених речовин.

Системи передачі даних використовують мережеві модулі та бездротові технології для передачі даних від сенсорів до центральної станції моніторингу або мобільних пристроїв медичного персоналу, що забезпечує віддалений контроль за станом пацієнта. Аналізуюче програмне забезпечення обробляє та інтерпретує зібрані дані, включаючи алгоритми для виявлення аномалій, прогнозування ризиків та підтримки прийняття рішень медичним персоналом.

Інтерфейси користувача, що входять до складу комплексу, представляють собою екрани та панелі керування, які дозволяють медичному персоналу легко взаємодіяти з системою, переглядати дані, налаштовувати параметри моніторингу та отримувати сповіщення про критичні зміни стану пацієнта.

Комплекс моніторингу стану пацієнта дозволяє здійснювати постійний контроль за здоров’ям пацієнтів, особливо тих, хто знаходиться у відділеннях інтенсивної терапії або має хронічні захворювання. Ця система значно підвищує точність діагностики, дозволяє швидше реагувати на зміни стану пацієнта та запобігає ускладненням. Завдяки своїй гнучкості та адаптивності, комплекс моніторингу може бути налаштований під індивідуальні потреби кожного пацієнта та вдосконалюватися з часом, інтегруючи нові технології та функціональні можливості.

2.2 Специфікація матеріалів кваліфікаційного проєкту

З цими компонентами (табл.2.1) ви зможете побудувати систему моніторингу стану пацієнта, яка забезпечить збір та аналіз даних про пульс, кисневий рівень в крові, температуру тіла та вологість оточуючого середовища.

Таблиця 2.1 – Матеріали

№	Назва компоненту	Кількість, шт.
1	Плата ESP32	1
2	Датчик пульсоксиметра MAX30100	1
3	Датчик DS18B20	1
4	Датчик DHT11	1
5	Резистори 4,7К	1
6	З'єднувальні дроти	10
7	Макетна дошка	1

Використання плати ESP32 разом з датчиками пульсоксиметра MAX30100, DS18B20 та DHT11, а також іншими необхідними компонентами, надає зручну платформу для реалізації комплексної системи моніторингу стану пацієнта. Ця система може забезпечувати неперервний збір та аналіз важливих показників, таких як пульс, кисневий рівень в крові, температура тіла та вологість оточуючого середовища.

Плата ESP32 виступає центральним елементом системи, обробляючи дані від датчиків та забезпечуючи бездротову передачу інформації до централізованої бази даних або мобільного пристрою для подальшого аналізу та моніторингу. Завдяки вбудованим модулям Wi-Fi та Bluetooth, ESP32 дозволяє легко інтегрувати систему у вже існуючу інфраструктуру, забезпечуючи безперебійний зв'язок і доступ до даних у режимі реального часу.

Датчик пульсоксиметра MAX30100 дозволяє вимірювати пульс та рівень кисню в крові пацієнта, що є критично важливими показниками для оцінки його стану. Датчик температури DS18B20 забезпечує точні вимірювання температури тіла, що є необхідним для контролю за можливими ознаками інфекцій або інших медичних станів. Датчик DHT11 вимірює

температуру та вологість оточуючого середовища, що дозволяє враховувати вплив зовнішніх умов на стан пацієнта.

Використання резисторів 4,7 К і з'єднувальних дротів дозволяє забезпечити надійне з'єднання між компонентами системи, а макетна дошка служить для зручної організації та тестування компонентів перед їх остаточною інтеграцією у пристрій.

Завдяки цим компонентам, створена система моніторингу стану пацієнта буде здатна виконувати складні завдання з високою точністю та надійністю, забезпечуючи важливі дані для медичного персоналу та допомагаючи у прийнятті обґрунтованих рішень щодо лікування та догляду за пацієнтами.

2.3 Датчик пульсоксиметра MAX30100

Датчик інтегрований датчик пульсоксиметрії та датчика серцевого ритму. Він поєднує в собі два світлодіоди, фотодетектор, оптимізовану оптику та обробку аналогового сигналу з низьким рівнем шуму для виявлення сигналів пульсу та частоти серцевих скорочень. Він працює від джерел живлення 1,8 В і 3,3 В і може бути відключений за допомогою програмного забезпечення з незначним струмом очікування, що дозволяє блоку живлення залишатися підключеним у будь-який час.

Рисунок 2.1 – MAX30100

Пристрій має два світлодіоди, один випромінює червоне світло, інший випромінює інфрачервоне світло. Для частоти пульсу потрібне лише інфрачервоне світло. І червоне, і інфрачервоне світло використовуються для вимірювання рівня кисню в крові. Коли серце перекачує кров, відбувається збільшення насиченої киснем крові внаслідок збільшення кількості крові. Коли серце розслабляється, об’єм насиченої киснем крові також зменшується. Знаючи час між збільшенням і зменшенням насиченої киснем крові, визначають частоту пульсу.

Таблиця 2.2 – Розпинування MAX30100

Пін	**Значення**
GND:	«земля»
RD:	драйвер червоного світлодіода
IRD:	драйвер ІК світлодіоди
INT:	переривання
SDA:	лінія даних
SCL:	лінія тактування
UIN:	напруга живлення

Таблиця 2.3 – Технічні характеристики модуля MAX30100

Характеристика	**Значення**
Мікросхема	MAX30100
Напруга живлення, В	5 (внутрішній стабілізатор)
Споживаний струм у режимі вимірювання, мА	1,2
Споживаний струм у режимі sleep, мкА	до 10
Інтерфейс	I2C
Максимальна частота інтерфейсу, кГц	400
Розміри модуля, мм	18,5 x 14,4 x 3

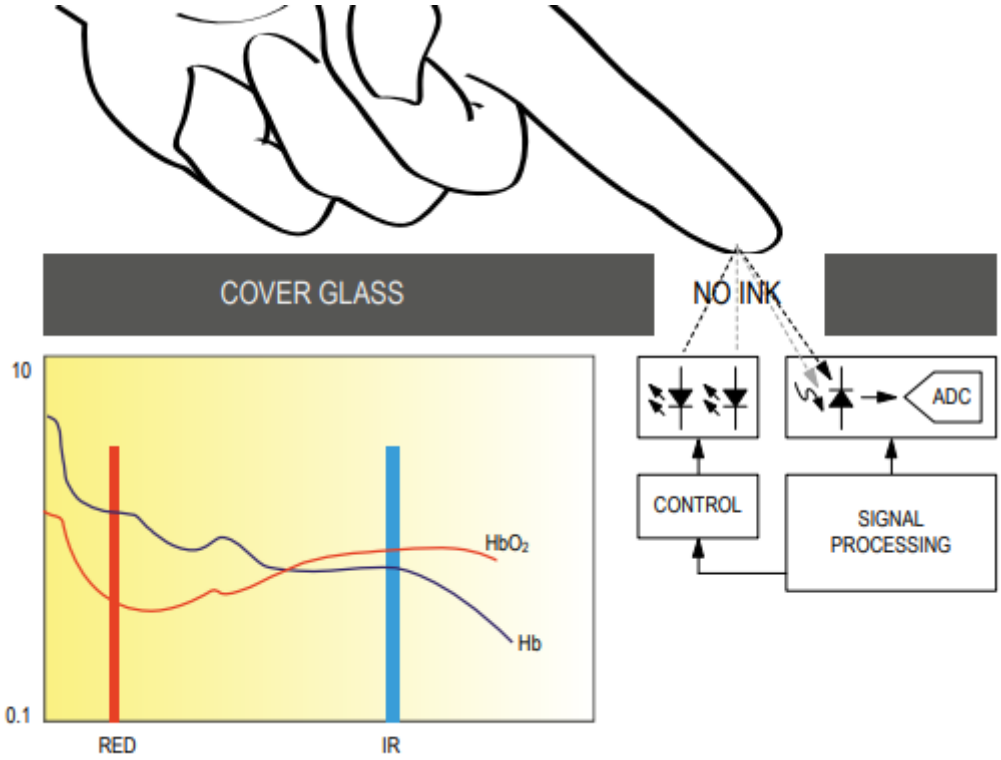


Рисунок 2.2 – Принцип роботи MAX30100

Датчик пульсоксиметра MAX30100 є ідеальним рішенням для інтеграції в системи моніторингу стану пацієнта завдяки своїм високим рівнем точності та ефективності, забезпечуючи надійний моніторинг пульсу та рівня кисню в крові в реальному часі.

2.4 Датчик температури DS18B20

Це попередньо підключена та водонепроникна версія датчика DS18B20. Датчик може вимірювати температуру від – 55 до 125 °C. Кабель має оболонку з ПВХ, що забезпечує його стійкість до вологи та інших впливів навколишнього середовища.

Однією з головних переваг DS18B20 є те, що він є цифровим датчиком, що забезпечує високу точність вимірювань навіть на великій відстані від мікроконтролера. Ці 1-провідні цифрові датчики температури мають точність до ± 0,5 °C у більшій частині робочого діапазону та можуть забезпечувати до 12 біт точності завдяки вбудованому цифро-аналоговому перетворювачу.

Вони чудово працюють з будь-яким мікроконтролером, використовуючи лише один цифровий контакт для зв'язку.

Рисунок 2.3 – DS18B20

Єдиним недоліком є використання протоколу Dallas 1-Wire, який є досить складним і вимагає значної кількості коду для аналізу зв'язку. Проте, цей недолік компенсується високою надійністю та точністю даних, що забезпечуються датчиком. Для коректної роботи датчика необхідний резистор на 4,7 кОм, який використовується для підтягування DATA лінії до лінії VCC.

Таблиця 2.4 – Основні характеристики DS18B20

Характеристика	**Значення**
Напруга живлення	3.0..5.5 В
Діапазон температур	-55°C..+125°C
Точність показань температури	0.5 °C
Крок показань	0.0625 °C
Інтерфейс	1-Wire
Споживаний струм	1 мА

Датчик DS18B20 ідеально підходить для використання в системі моніторингу стану пацієнта завдяки своїм характеристикам та надійності. Він забезпечує точне вимірювання температури тіла пацієнта, що є критично важливим для моніторингу стану здоров'я.

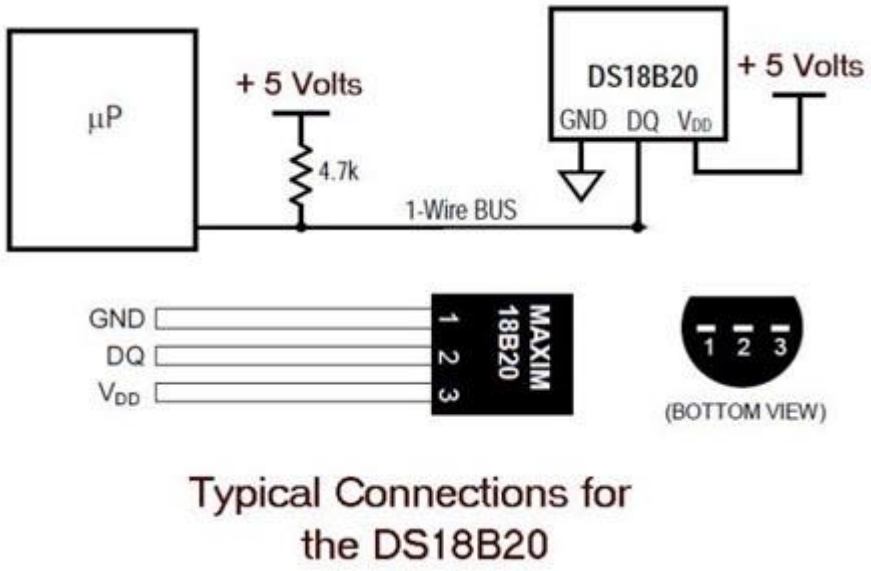


Рисунок 2.4 – Схема підключення датчика DS18B20

Датчик температури DS18B20 є водонепроникним, точним і надійним, здатним вимірювати температуру від – 55 до 125 °C з точністю до ± 0,5°C. Хоча протокол Dallas 1-Wire складний, цей датчик ідеально підходить для систем моніторингу, таких як вимірювання температури тіла пацієнтів, забезпечуючи точні дані навіть на великій відстані від мікроконтролера.

2.5 DHT11 Датчик вологості та температури

DHT11 – це базовий, недорогий цифровий датчик температури та вологості. Він використовує ємнісний датчик вологості та термістор для вимірювання навколишнього повітря та виводить цифровий сигнал на контакт даних (аналогові входи не потрібні).

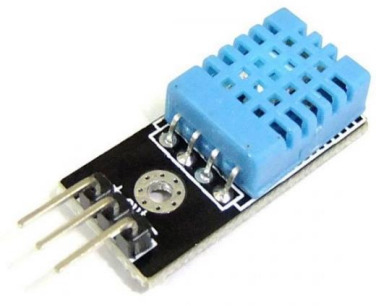


Рисунок 2.5 – DHT11

Він досить простий у використанні, але вимагає ретельного визначення часу для отримання даних. Єдиним справжнім недоліком цього датчика є те,

що ви можете отримувати нові дані з нього лише кожні 2 секунди, тому під час використання бібліотеки показання датчика можуть бути давнішими до 2 секунд.

DHT11 є ефективним рішенням для простих систем моніторингу, де потрібні базові вимірювання температури та вологості з достатньою точністю. Він знаходить застосування в медичних пристроях для моніторингу середовища, в умовах контролю якості повітря, а також в системах з контролем клімату в приміщеннях.

Таблиця 2.5 – Технічні характеристики DHT11

Температура	
Дискретність	1°C
Вимірюваний діапазон	0°C ~ 50°C
Точність	±2°C
Вологість	
Дискретність	1%RH
Вимірюваний діапазон	20%RH ~ 90%RH
Точність	±5%RH (0~50°C)
Робоча напруга	3.3 В ~ 5.5 В

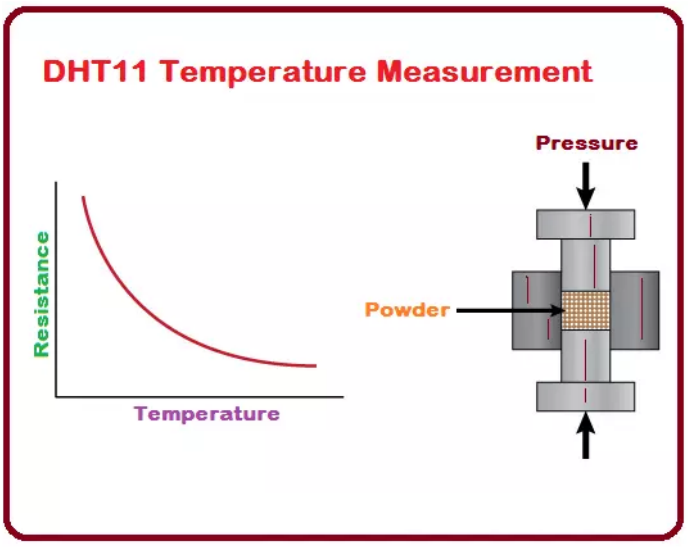


Рисунок 2.6 – Відображення залежності між температурою та опором для датчика DHT11

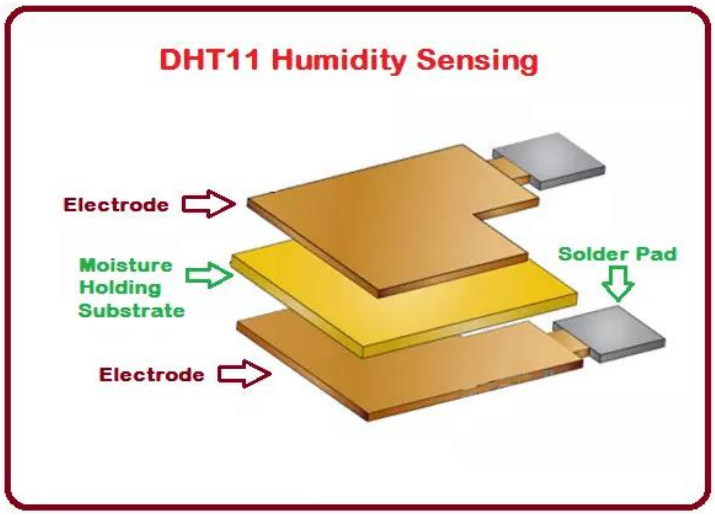


Рисунок 2.7 – Візуальний опис вимірювання вологості за допомогою датчика DHT11

DHT11 є ефективним рішенням для вимірювання температури та вологості в простих системах моніторингу, де потрібні базові показники з точністю, достатньою для багатьох медичних застосувань та інших областей.

2.6 Вибір плати ESP32

ESP32 – це багатофункціональний мікроконтролер, який має вбудовані модулі Wi-Fi та Bluetooth, що дозволяє зручно інтегрувати систему моніторингу стану пацієнта в існуючу мережу. Плата ESP32 відзначається високою продуктивністю, низьким енергоспоживанням та можливістю одночасної обробки кількох завдань, що робить її ідеальним вибором для медичних пристроїв.

Однією з ключових характеристик ESP32 є двоядерний процесор Xtensa LX106 з тактовою частотою до 240 МГц, який забезпечує достатню обчислювальну потужність для складних обчислень і обробки даних у режимі реального часу. Цей мікроконтролер має 320 КБ SRAM та 4 МБ флеш-пам'яті, що дозволяє зберігати та обробляти великий обсяг даних, необхідних для моніторингу стану пацієнта.

ESP32 підтримує бездротове підключення через Wi-Fi 802.11 b/g/n та Bluetooth v4.2 BR/EDR та BLE, що забезпечує надійний зв'язок між

пристроями та зручність у передачі даних до централізованої системи моніторингу. Це дозволяє здійснювати безперервний моніторинг і передавання даних у режимі реального часу, що є критичним для медичних застосувань.

Завдяки 17 GPIO-пінам та наявності 3 АЦП (12-біт) і 2 ЦАП, плата ESP32 може підключати та обробляти сигнали від різноманітних датчиків і виконавчих механізмів, що є важливим для комплексного моніторингу різних фізіологічних параметрів пацієнта. Підтримка інтерфейсів зв'язку, таких як I2C, SPI, UART та SDIO, забезпечує гнучкість у підключенні додаткових пристроїв та модулів, необхідних для повноцінного функціонування системи.

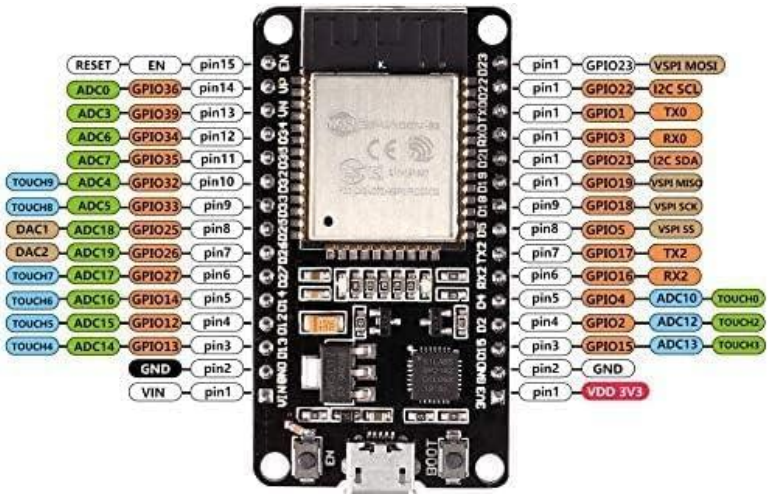


Рисунок 2.8 – Піни ESP32-WROOM-32

Додаткові функції, такі як сенсор дотику та датчик температури, розширюють можливості плати та дозволяють використовувати її в різних медичних та IoT-застосуваннях. Живлення від 3,3 В з низьким споживанням енергії, що досягає до 5 мкА в режимі глибокого сну, робить ESP32 енергоефективним рішенням для тривалого використання без необхідності частого підзаряджання.

Розміри плати (25,4 мм х 15,2 мм) дозволяють інтегрувати її у компактні пристрої, що є важливим для носимих медичних сенсорів та інших компактних медичних пристроїв. Діапазон робочих температур від – 40 °С до + 80°С забезпечує стабільну роботу плати в різних умовах експлуатації.

Таблиця 2.5 – Основні характеристики ESP32-WROOM-32

Характеристика	Специфікація
Мікроконтролер	Двоядерний процесор Xtensa LX106, з тактовою частотою до 240 МГц
Пам'ять	320 КБ SRAM, 4 МБ флеш-пам'яті
Бездротове підключення	Wi-Fi 802.11 b/g/n, Bluetooth v4.2 BR/EDR та BLE
GPIO	17 GPIO-пінів
Аналогові периферійні пристрої	3 АЦП (12-біт), 2 ЦАП
Інтерфейси зв'язку	2 шини I2C, 1 шина SPI, 1 UART, 1 шина SDIO
Спеціальні функції	1 сенсор дотику, 1 датчик температури
Живлення	3,3 В, Низьке споживання енергії (до 5 мкА в режимі глибокого сну)
Розміри	25,4 мм x 15,2 мм
Застосування	IoT-пристрої, Носимі пристрої, Домашня автоматизація, Промислова автоматизація, Датчики та виконавчі механізми
Додаткові функції	Безпечний запуск, Оновлення по повітрю (OTA), Ультранизьке споживання енергії, Широкий діапазон робочих температур (-40°C до +80°C)

ESP32 є оптимальним вибором для розробки системи моніторингу стану пацієнта завдяки своїм багатофункціональним можливостям і високій продуктивності. Завдяки наявності вбудованих модулів Wi-Fi та Bluetooth, ESP32 забезпечує зручну інтеграцію з існуючими мережами та пристроями, дозволяючи безперервний моніторинг і передавання даних у режимі реального часу. Висока швидкість обробки даних, низьке енергоспоживання та здатність одночасно виконувати кілька завдань роблять цю плату ідеальним вибором для медичних застосувань, де надійність та ефективність є критичними.

2.7 Програмування та налагодження на платформі ESP32-WROOM-32

Програмування ESP32-WROOM-32 здійснюється з використанням різних інструментів та середовищ розробки, що робить його доступним для розробників із різним рівнем досвіду.

Arduino IDE:

Найпопулярніший інструмент для програмування ESP32-WROOM-32 – це Arduino IDE. Він надає простий та інтуїтивно зрозумілий інтерфейс, широкий набір бібліотек та прикладів, що полегшує процес розробки.

Особливості:

– інтуїтивний інтерфейс робить Arduino IDE ідеальним вибором для початківців та досвідчених розробників;

– широкий набір бібліотек дозволяє швидко створювати складні програми для ESP32-WROOM-3.

PlatformIO:

PlatformIO – це потужне середовище розробки для вбудованих систем, яке підтримує ESP32-WROOM-32. Вона надає інструменти для управління проєктами, бібліотеки та розширені функції налагодження.

Особливості:

– PlatformIO надає багатофункціональне середовище розробки з інтеграцією з різними IDE та підтримкою багатьох мікроконтролерів, включаючи ESP32-WROOM-32;

– зручне управління проєктами та бібліотеками дозволяє зосередитися на самому процесі розробки.

Espressif IDF:

Офіційний фреймворк для розробки на ESP32 від Espressif Systems – це Espressif IoT Development Framework (IDF). Він надає низькорівневий доступ до функцій чіпа і дозволяє створювати високоефективні програми.

Особливості:

– низькорівневий доступ: IDF надає прямий доступ до функцій мікроконтролера ESP32, що дозволяє розробникам створювати оптимізовані програми;

– широкі можливості: IDF має багато функцій для роботи з різними інтерфейсами та протоколами, такими як Wi-Fi, Bluetooth, GPIO та інші.

Налагодження:

Налагодження коду на ESP32-WROOM-32 може здійснюватися за допомогою JTAG, UART та інших інтерфейсів. Існують різні налагоджувальні плати та адаптери, які спрощують процес тестування та усунення несправностей.

Особливості:

– JTAG і UART: інтерфейси JTAG і UART дозволяють підключати пристрої для налагодження та отримання відлагоджувальних повідомлень;

– налагоджувальні плати: існують спеціальні налагоджувальні плати та адаптери, які дозволяють підключати ESP32-WROOM-32 до комп'ютера для налагодження через USB або інші інтерфейси.

Ці інструменти та методи налагодження роблять ESP32-WROOM-32 доступним та зручним для програмістів із різним рівнем досвіду, що дозволяє швидко та ефективно розробляти програми для IoT-пристроїв.

2.8 Архітектура системи

Архітектура системи моніторингу стану пацієнта на базі ESP32 побудована таким чином, щоб забезпечити ефективний і надійний збір, обробку та передачу даних про стан пацієнта. Ця архітектура включає декілька ключових компонентів, кожен з яких виконує важливу роль у забезпеченні загальної функціональності системи. Розглянемо докладніше кожен з елементів системи.

Основні елементи системи

1) Плата ESP32:

– плата ESP32 виконує функцію центрального контролера. Вона відповідає за збір даних з підключених датчиків, їх обробку, а також передачу оброблених даних на сервер для подальшого аналізу та зберігання;

– завдяки вбудованим модулям Wi-Fi та Bluetooth, ESP32 забезпечує бездротову комунікацію, що дозволяє легко інтегрувати систему в існуючу мережеву інфраструктуру. Крім того, плата має високу обчислювальну потужність, що дозволяє обробляти дані в реальному часі;

– плата ESP32 підключається до датчиків через відповідні інтерфейси (GPIO, I2C, SPI, UART), що дозволяє ефективно зчитувати дані з різних сенсорів;

2) Датчики:

– пульсоксиметр MAX30100: цей датчик використовується для вимірювання пульсу та рівня кисню в крові пацієнта. Він поєднує два світлодіоди (червоний і інфрачервоний) та фотодетектор, що дозволяє точно визначати фізіологічні параметри;

– датчик температури DS18B20: використовується для вимірювання температури тіла пацієнта. Цей датчик забезпечує високу точність і стабільність вимірювань, що є важливим для медичних застосувань;

– датчик вологості DHT11: цей датчик вимірює вологість навколишнього середовища та температуру, що дозволяє оцінювати умови, в яких знаходиться пацієнт, та здійснювати відповідні коригування в обробці даних;

3) Комунікаційні модулі:

– вбудовані в ESP32 модулі Wi-Fi та Bluetooth: забезпечують передачу зібраних даних на віддалений сервер для подальшого аналізу та

зберігання. Завдяки цим модулям, система може передавати дані в режимі реального часу, що є критично важливим для моніторингу стану пацієнта;

– Wi-Fi модуль: використовується для підключення до локальної мережі та передачі даних на сервер через Інтернет;

– Bluetooth модуль: може використовуватись для локальної передачі даних на мобільні пристрої або інші сумісні пристрої, що знаходяться поблизу.

Додаткові компоненти

1) Макетна дошка: використовується для зручного підключення та розміщення компонентів системи під час розробки та тестування;

2) З'єднувальні дроти: забезпечують електричне з'єднання між платою ESP32 та датчиками, а також іншими компонентами системи;

3) Резистор: використовуються для налаштування робочих параметрів датчиків і забезпечення стабільної роботи всієї системи.

2.9 Алгоритм роботи системи

Алгоритм роботи системи моніторингу стану пацієнта включає кілька основних етапів, які забезпечують ефективний збір, обробку, передачу та аналіз даних. Нижче наведено детальний опис кожного з цих етапів:

− **збір даних**: датчики, підключені до плати ESP32, збирають дані про пульс, рівень кисню в крові, температуру тіла та вологість навколишнього середовища. Датчик пульсоксиметра MAX30100 вимірює пульс і рівень кисню в крові, датчик DS18B20 визначає температуру тіла, а датчик DHT11 реєструє температуру та вологість оточуючого середовища;

− **обробка даних**: зібрані дані обробляються мікроконтролером ESP32. Це включає первинну фільтрацію та обробку сигналів, конвертацію аналогових даних у цифрові та підготовку даних для передачі. Наприклад, сигнал від пульсоксиметра проходить через етапи фільтрації для видалення шумів, а потім цифрові дані піддаються алгоритмічному аналізу для виявлення важливих медичних показників;

− **передача даних**: оброблені дані передаються на віддалений сервер за допомогою вбудованих модулів Wi-Fi або Bluetooth. На сервері дані зберігаються та аналізуються для подальшого використання медичним персоналом. Це забезпечує можливість віддаленого моніторингу стану пацієнта в режимі реального часу, дозволяючи лікарям отримувати доступ до даних незалежно від їхнього місцезнаходження.

Цей алгоритм забезпечує надійну роботу системи моніторингу стану пацієнта, дозволяючи оперативно збирати, обробляти та передавати важливу інформацію про стан здоров’я пацієнта, що є критичним для ефективного медичного догляду.

2.10 Тестування та валідація

Тестування системи моніторингу стану пацієнта на базі ESP32 включає кілька ключових етапів, які необхідно пройти для забезпечення її надійності та ефективності.

Функціональне тестування

Цей етап орієнтований на перевірку роботи окремих компонентів системи, таких як ESP32, MAX30100, DS18B20, та DHT11. Він включає перевірку коректності зчитування даних з датчиків, обробку сигналів і відправлення інформації на сервер.

Інтеграційне тестування

На цьому етапі оцінюється взаємодія між різними компонентами системи, зокрема ESP32 і вебсервером. Проводиться перевірка синхронізації зчитування та відправлення даних, а також взаємодії між різними типами датчиків та мікроконтролером.

Тестування на точність

Цей етап спрямований на перевірку точності вимірювань датчиків (DS18B20 і DHT11) у різних умовах експлуатації. Включає проведення вимірювань температури та вологості при різних температурах і рівнях вологості для оцінки відповідності вимірювань технічним характеристикам.

Кожен з цих етапів є критично важливим для забезпечення якості системи моніторингу. Після успішного проходження всіх етапів тестування можна переходити до фази валідації, що передбачає підтвердження того, що система відповідає вимогам та специфікаціям проєкту.

2.11 Переваги та недоліки системи моніторингу стану пацієнта на базі ESP32

Переваги системи моніторингу стану пацієнта на базі ESP32 включають:

– висока точність: використання точних датчиків для збору даних;

– безперервний моніторинг: можливість постійного збору даних у реальному часі;

– безпровідна передача даних: зручне підключення до мережі та передача даних на сервер;

– гнучкість: можливість підключення додаткових датчиків та розширення функціональності системи;

Недоліками можуть бути:

– складність налаштування: необхідність програмування та налаштування компонентів;

– обмеження енергоспоживання: необхідність забезпечення стабільного живлення для всіх компонентів.

Система моніторингу стану пацієнта на базі ESP32 має такі переваги: висока точність завдяки точним датчикам, безперервний моніторинг у реальному часі, безпровідна передача даних на сервер та гнучкість у підключенні додаткових датчиків. Недоліки включають складність налаштування та необхідність стабільного живлення для всіх компонентів.

Висновок до розділу 2

У даному розділі було розглянуто основні складові системи моніторингу, які включають в себе широкий спектр сенсорів, таких як пульсоксиметри, датчики температури та вологості, а також модулі для вимірювання інших фізіологічних параметрів пацієнтів. Важливими компонентами є також ЕКГ, капнографи, системи моніторингу дихання та інші, які спільно забезпечують повний спектр необхідних даних для медичного персоналу.

Основними перевагами обраної системи є її здатність збирати, аналізувати та передавати дані в режимі реального часу, що дозволяє своєчасно реагувати на зміни стану пацієнта та забезпечувати необхідний рівень моніторингу. Використання мікроконтролера ESP32 забезпечує високу інтегрованість з вбудованими модулями Wi-Fi та Bluetooth, що робить його ідеальним рішенням для забезпечення бездротового зв'язку та доступу до даних у реальному часі.

Вибір конкретних компонентів, таких як датчики пульсоксиметра MAX30100, DS18B20 та DHT11, показує комплексний підхід до забезпечення точності та надійності вимірювань, необхідних для моніторингу стану пацієнта. Використання макетної дошки та необхідних з'єднувальних елементів демонструє гнучкість системи та можливість тестування компонентів перед їх інтеграцією.

Комплекс моніторингу стану пацієнта на базі ESP32 є високоефективною та прогресивною системою, яка відповідає сучасним вимогам у медичній діагностиці та нагляді за пацієнтами, забезпечуючи надійність, точність та зручність в експлуатації.

3 РОЗРОБКА АПАРАТНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка апаратного забезпечення

Головною метою дипломної роботи є створення комплексної системи моніторингу стану пацієнта на базі мікроконтролера ESP32. Враховуючи висновки попередніх розділів, було встановлено, що для моніторингу основних фізіологічних показників пацієнта використовуються три датчики: MAX30100, DS18B20 та DHT11. Ці датчики дозволяють отримувати інформацію про частоту серцевих скорочень, температуру тіла та вологість навколишнього середовища, що є важливими параметрами для оцінки загального стану здоров'я пацієнта.

Аналізуючи наявні аналогічні системи від провідних компаній, таких як Holter Monitor, Philips та AliveCor, було виявлено, що такі системи часто є дорогими та складними у використанні. Тому було прийнято рішення розробити доступну і просту у використанні систему, яка могла б бути впроваджена в будь-яких умовах – від домашнього використання до використання у медичних закладах.

Згідно з висновками другого розділу, основними компонентами системи є:

- мікроконтролер ESP32;
- датчик пульсу та рівня кисню в крові MAX30100;
- датчик температури DS18B20;
- датчик температури та вологості DHT11.

Рисунок 3.1 – Функціональна схема роботи пристрою

Всі компоненти підключаються до мікроконтролера ESP32, який виконує роль «мозку» в цій системі. ESP32 отримує сигнали від датчиків, обробляє їх і передає на дисплей або інший пристрій для подальшого аналізу. Принцип роботи можна описати наступним чином:

1) живлення системи: після вмикання живлення система запускається. Увімкнення та вимкнення здійснюється за допомогою перемикача або кнопки. Живлення може здійснюватися від акумулятора або блоку живлення;

2) збір даних від датчиків:

− датчик MAX30100: вимірює частоту серцевих скорочень та рівень кисню в крові. Інфрачервоний і червоний світлодіоди постійно випромінюють світло, яке проходить через тканини, і фоторецептори вимірюють кількість відбитого світла, що дозволяє визначити частоту пульсу та рівень SpO2;

− датчик DS18B20: визначає температуру тіла пацієнта з високою точністю. Цей датчик використовує цифровий сигнал для передачі даних, що забезпечує високу точність вимірювань;

− датчик DHT11: вимірює температуру і вологість навколишнього середовища, що є важливими показниками для загального стану здоров'я пацієнта;

3) обробка даних: зібрані дані передаються на мікроконтролер ESP32, який їх обробляє та виконує необхідні розрахунки. ESP32 має вбудований модуль Wi-Fi, що дозволяє передавати дані на віддалений сервер для зберігання та подальшого аналізу;

4) відображення та зберігання даних: оброблені дані можуть відображатися на локальному дисплеї або передаватися на мобільний пристрій за допомогою спеціального додатку. Також дані можуть зберігатися на сервері для довгострокового моніторингу стану пацієнта.

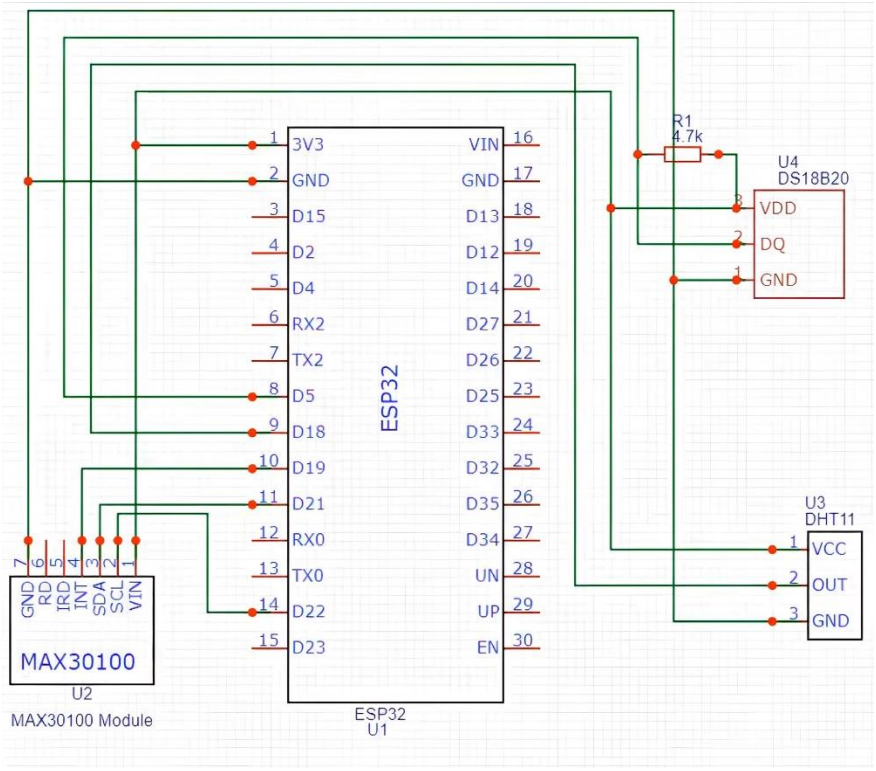


Рисунок 3.2 – Детальна схема підключення компонентів

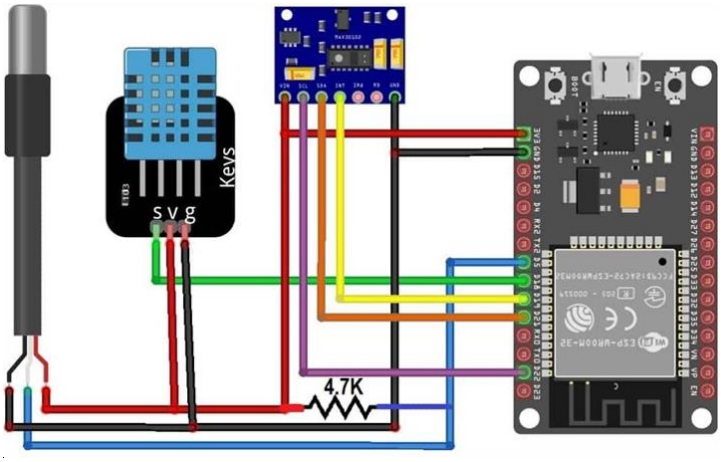


Рисунок 3.3 – Схема фізичного підключення компонентів

Для створення даної системи були використані компоненти, які забезпечують високу надійність і точність вимірювань при мінімальних витратах. Зібрані компоненти були змонтовані на спеціальній платі, що забезпечує компактність і зручність використання системи.

Після завершення зборки було проведено тестування системи для перевірки її працездатності та точності вимірювань. Всі компоненти показали високу точність і стабільність в роботі, що підтвердило ефективність розробленої системи.

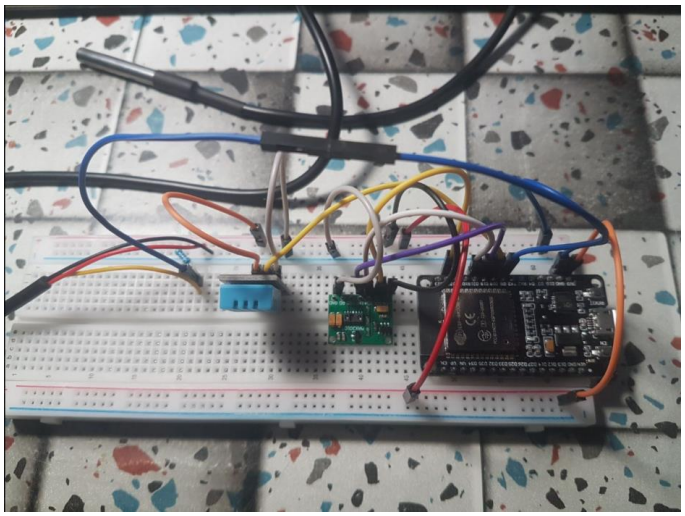


Рисунок 3.4 – Кінцевий вигляд готового пристрою

Для підтвердження доступності даної системи було складено кошторис собівартості комплексу моніторингу стану пацієнта.

Таблиця 3.1 – Кошторис обладнання для системи моніторингу стану пацієнта

Найменування	Кількість	Од. вим.	Ціна, грн	Вартість, грн
ESP32	1	шт.	229,00	229,00
MAX30100	1	шт.	79,00	79,00
DS18B20	1	шт.	54,00	54,00
DHT11	1	шт.	64,00	64,00
Резистор 4,7К	1	набір	9,00	9,00
З’єднувальні проводи	1	набір	50,00	50,00
Макетна плата	1	шт.	68,00	68,00
			РАЗОМ	**553,00 грн**

Таким чином, розроблена система є доступною і зручною у використанні, що дозволяє її впровадження у різних умовах. Використання ESP32 та стандартних датчиків забезпечує високу точність вимірювань та можливість дистанційного моніторингу стану пацієнта

3.2 Мова програмування Arduino

Мова програмування Arduino – ключовий інструмент для розробки електронних пристроїв на платформі Arduino, зокрема для ESP32. Вона базується на мові програмування C/C++, але має спеціалізовані функції та бібліотеки, що спрощують роботу з мікроконтролерами Arduino.

Основна структура програми в Arduino включає дві основні функції: *setup()* та *loop()*. Функція *setup()* викликається один раз при запуску програми та використовується для налаштування початкових параметрів, ініціалізації змінних та об'єктів. Натомість, функція loop() викликається постійно у безкінечному циклі після функції *setup()*. Ця функція відповідає за виконання основної логіки програми, зчитування даних з сенсорів, керування виводами та іншими завданнями.

У мові програмування Arduino є вбудовані функції та бібліотеки, які значно спрощують роботу з апаратними можливостями платформи Arduino. Ці функції та бібліотеки дозволяють легко взаємодіяти з різноманітними пристроями та модулями, такими як сенсори, дисплеї, активні модулі та мережеві модулі. Вони забезпечують широкий функціонал для розробки різноманітних проєктів та дозволяють програмістам швидко реалізувати свої ідеї без необхідності писати надскладні коди.

Переваги:

− простота в освоєнні: мова має простий синтаксис, що дозволяє швидко вивчити її новачкам;

− багато вбудованих функцій та бібліотек: це спрощує роботу з апаратними можливостями Arduino та дозволяє швидко реалізувати різноманітні проєкти;

− підтримка стандартних конструкцій: вона дозволяє використовувати звичайні підходи до програмування для створення складних програмних рішень.

Узагальнення: Arduino Language є простим та потужним інструментом для розробки електронних пристроїв на платформі Arduino. Вона дозволяє розробникам швидко та ефективно створювати різноманітні проєкти, незалежно від їхнього рівня досвіду в програмуванні.

3.3 Arduino IDE

Arduino IDE є ключовим інструментом для розробки проєктів на платформі Arduino. Його головною перевагою є доступність на різних операційних системах, таких як Windows, macOS та Linux, що робить його універсальним для користувачів з різними комп’ютерами та умовами. Arduino IDE пропонує широкий спектр функцій та інструментів, які полегшують написання програм для мікроконтролерів Arduino. Зокрема, серед цих функцій є підсвічування синтаксису, автодоповнення коду, перевірка наявності помилок та інші зручності, що сприяють розробці

Рисунок 3.5 – Інтерфейс програми

Arduino IDE є незамінним інструментом для створення та розробки різноманітних електронних проєктів завдяки наступним ключовим особливостям:

– підтримка різних операційних систем: Arduino IDE підтримується на Windows, macOS та Linux, що дозволяє користувачам з будь-якими пристроями легко розробляти програми для мікроконтролерів Arduino;

− зручний інтерфейс: інтуїтивно зрозумілий і простий інтерфейс робить процес розробки приємним та ефективним;

− підсвічування синтаксису та автодоповнення: ці функції полегшують написання коду та допомагають уникнути помилок;

− вбудовані бібліотеки: інтегрована система керування бібліотеками дозволяє швидко додавати та використовувати різноманітні бібліотеки для розширення функціональності Arduino;

− приклади та документація: велика кількість прикладів програм та документації допомагає користувачам швидко освоїти різні функції та можливості платформи Arduino;

− можливість вивчення та навчання: простота та доступність Arduino IDE дозволяють користувачам вчитися програмуванню та розробці електронних пристроїв на базі Arduino;

− широкий спектр функцій: Arduino IDE надає багато інструментів для розробки програм, включаючи умовні оператори, цикли, математичні операції та інші стандартні конструкції;

− середовище налагодження: вбудовані інструменти для виводу даних у серійний монітор та підтримка точок зупинки дозволяють ефективно налагоджувати програми на мікроконтролері Arduino;

− простота використання: навіть початківці можуть швидко освоїти Arduino IDE завдяки доступності документації та великої спільноти, яка завжди готова допомогти;

− можливість розширення: Arduino IDE можна розширювати за допомогою сторонніх плагінів та розширень, що відкриває нові можливості для розробників та дозволяє адаптувати середовище розробки під власні потреби;

− підтримка спільноти: активна спільнота користувачів Arduino пропонує безліч ресурсів, допомоги та прикладів, що полегшує навчання та вирішення практичних завдань.

Arduino IDE - це ключовий інструмент для розробки проєктів на платформі Arduino. Його головною перевагою є універсальність (підтримка різних операційних систем), зручний інтерфейс та широкий спектр функцій, таких як підсвічування синтаксису, автодоповнення коду та інші зручності. Arduino IDE наділяє розробників вбудованими бібліотеками, прикладами та підтримкою спільноти, що робить процес написання програм доступним та ефективним для користувачів на будь-якому рівні вмінь.

3.4 Програмування системи моніторингу стану пацієнта

Перш за все, для роботи нашої системи моніторингу стану пацієнта необхідно програмувати мікроконтролер ESP32. Це є критичним етапом, оскільки саме програмне забезпечення визначає функціональність та ефективність системи. Для цього треба виконати наступні кроки:

− завантаження Arduino IDE – це перший крок у програмуванні платформи ESP32. Це інтегроване середовище розробки (IDE), яке надає інструменти для створення програмного коду. Воно доступне для різних операційних систем, включаючи Windows, macOS та Linux, що робить його доступним для широкого кола користувачів. Arduino IDE має простий та зрозумілий інтерфейс, що дозволяє навіть початківцям швидко освоїти процес програмування на ESP32;

− підключення ESP32 до комп'ютера через USB-кабель – це крок, що передує програмуванню. Це дозволяє передавати програми з комп'ютера на плату ESP32 для виконання. Під час підключення ESP32 до комп'ютера, важливо встановити драйвери, які забезпечують правильне визнання плати комп'ютером та співпрацю з IDE. Інструкції щодо встановлення драйверів можна знайти на офіційному вебсайті Arduino або в документації до плати ESP32;

− вибір платформи ESP32 у Arduino IDE – це дозволяє налаштувати IDE для роботи з конкретною платою. Користувач може обирати з різних варіантів плат, що дозволяє розробляти проєкти для різних моделей ESP32.

Після вибору платформи, необхідно також вибрати відповідний послідовний порт, до якого підключено ESP32. Це забезпечить правильне зв'язок між комп'ютером та платою під час завантаження програми;

− написання програмного коду – це основний етап в розробці будь-якого проєкту на ESP32. Код визначає логіку та функціональність пристрою. Arduino IDE надає зручний текстовий редактор з підсвічуванням синтаксису, що допомагає зрозуміти структуру коду та виявити помилки. Користувачі можуть використовувати вбудовані шаблони та приклади коду для швидкого початку роботи над проєктом;

− компіляція програмного коду – це процес перетворення вихідного коду на машинний код, який може виконуватися мікроконтролером ESP32. Arduino IDE надає вбудований компілятор, який перевіряє код на наявність синтаксичних помилок та генерує виконавчий файл. Після успішної компіляції, користувач може завантажити програму на ESP32 за допомогою USB-кабеля. Процес завантаження включає передачу виконавчого файлу на плату та його виконання мікроконтролером;

− тестування роботи програми – це процес завантаження програми на ESP32, важливо перевірити, чи працює вона належним чином. Це включає підключення необхідних сенсорів та актуаторів і тестування їх функціональності. Arduino IDE надає можливість взаємодії з платою через вбудований серійний монітор, який відображає виведення програми та дозволяє відстежувати значення змінних. Під час тестування, користувач може виявити можливі помилки та недоліки програми та виправити їх;

− налагодження програми – це процес виявлення та виправлення помилок у програмному коді для досягнення оптимальної роботи пристрою. Використання різних інструментів для налагодження, таких як серійний монітор, дозволяє відстежувати значення змінних та виводити повідомлення для діагностики проблем. Вдосконалення програми може включати оптимізацію коду, додавання нових функцій та вирішення проблем, виявлених під час тестування.

Загалом, щоб система запрацювала, потрібно встановити програму Arduino IDE, підключити мікроконтролер, написати код, скомпілювати його та виправити будь-які помилки, якщо вони виявляться. У випадку виникнення проблем можна скористатися документацією по роботі з бібліотеками на вебсайті Arduino IDE.

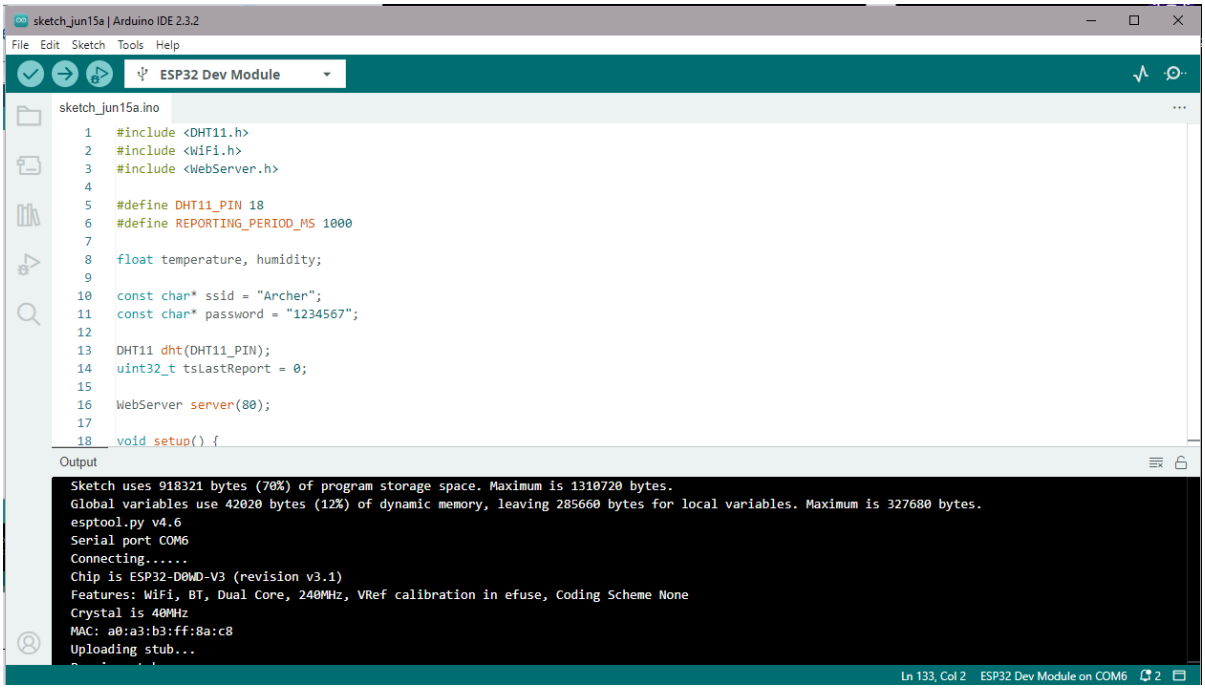


Рисунок 3.6 – Процес створення програмного коду
за допомогою застосунку Arduino IDE

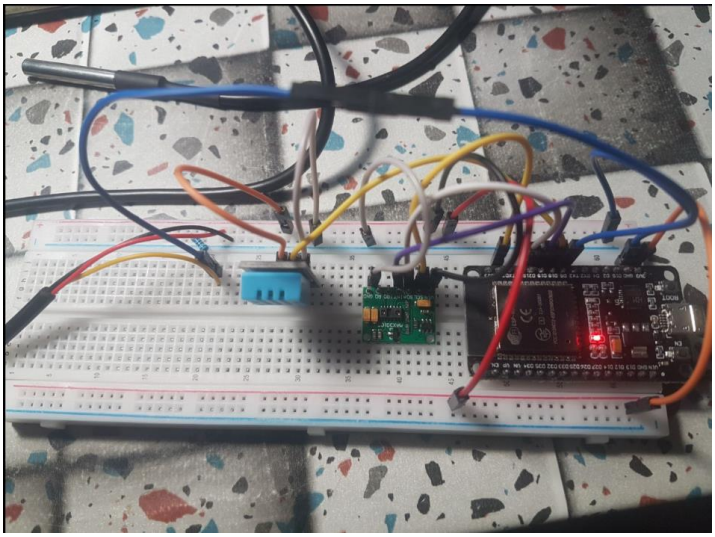


Рисунок 3.7 – Результат після встановлення прошивки

Принцип роботи:

- використовуються SSID та пароль для підключення до бездротової мережі;
- після успішного підключення до Wi-Fi, на серійному порту виводиться IP-адреса пристрою;
- сенсор DHT11 підключений до піна 18 та використовується для вимірювання температури повітря і вологості;
- пульсоксиметр MAX30100 підключений до пінів 19, 21, 22 та використовується для вимірювання пульсу та рівня кисню в крові;
- сенсор температури DS18B20 підключений до піна 5 і використовується для вимірювання температури тіла;
- вебсервер налаштований на обробку запитів на порту 80;
- доступні два основні обробники запитів: для головної сторінки ("/") та для обробки невідомих запитів (404 сторінка);
- обробляються запити вебсервера;
- оновлюються дані пульсоксиметра та сенсорів температури;
- якщо пройшов певний час (1 секунда), зчитуються нові значення з усіх сенсорів і виводяться на серійний порт;
- дані з сенсорів передаються до вебінтерфейсу для відображення в браузері;
- вебсторінка містить інформацію про температуру повітря, вологість, пульс, рівень кисню в крові та температуру тіла, представлені у вигляді текстових блоків та SVG графіки.

Рисунок 3.8 – Блок-схема роботи пристрою

Програмування мікроконтролера ESP32 для системи моніторингу стану пацієнта є критичним етапом, що вимагає кропіткої уваги до деталей. Використання Arduino IDE спрощує процес завдяки зручному інтерфейсу та підтримці різних операційних систем. Налаштування, програмування та тестування коду дозволяють забезпечити надійну та ефективну роботу пристрою, забезпечуючи зчитування температури тіла, пульсу та рівня кисню в крові у реальному часі через бездротове з'єднання.

3.5 Налаштування arduino esp32-wroom-32 як вебсервера

ESP32-WROOM-32 – це мікроконтролер, який базується на мікросхемі ESP32. Він має вбудований модуль Wi-Fi, що дозволяє йому підключатися до бездротових мереж і взаємодіяти з іншими пристроями через Інтернет.

Кроки налаштування плати ESP32-WROOM-32 як вебсервера:

– підключення ESP32 до комп'ютера: використовуйте USB-кабель для підключення ESP32-WROOM-32 до комп'ютера;

– встановлення Arduino IDE: завантажте та встановіть Arduino IDE з офіційного вебсайту Arduino (https://www.arduino.cc/en/Main/Software), якщо ви ще цього не зробили;

– встановлення драйверів (за необхідності): переконайтеся, що комп'ютер розпізнає плату ESP32-WROOM-32. Якщо потрібно, встановіть драйвери для плати;

– встановлення плати ESP32 в Arduino IDE: увімкніть Arduino IDE і перейдіть у меню "File" -> "Preferences". У полі "Additional Board Manager URLs" додайте наступне посилання: https://dl.espressif.com/dl/package_esp32_index.json. Натисніть "OK", а потім перейдіть у меню "Tools" -> "Board" -> "Boards Manager". Знайдіть та встановіть "ESP32" від Espressif Systems;

– вибір плати ESP32 в Arduino IDE: після встановлення платформи ESP32 перейдіть у меню "Tools" -> "Board" та виберіть "ESP32 Dev Module";

– вибір порту: оберіть порт, до якого підключена ваша плата ESP32-WROOM-32. Це можна зробити у меню "Tools" -> "Port";

– написання програми для вебсервера: напишіть програму для вебсервера в Arduino IDE, використовуючи бібліотеки WiFi.h та WebServer.h. Створіть маршрути для обробки HTTP-запитів та відправлення відповідей клієнтам;

– завантаження програми на плату: після написання програми завантажте її на плату ESP32-WROOM-32, натиснувши кнопку "Upload" в Arduino IDE;

– перевірка роботи вебсервера: після успішного завантаження програми відкрийте вебпереглядач і введіть IP-адресу ESP32-WROOM-32 (яку можна знайти у програмі). Ви повинні побачити вебсторінку, яка була створена в програмі для Arduino.

Ці кроки допоможуть налаштувати ESP32 як вебсервер і взаємодіяти з ним через Інтернет. Це відкриває безліч можливостей для взаємодії з іншими пристроями та створення різноманітних вебзастосунків.

3.6 Приклади використання Arduino ESP32-WROOM-32 як вебсервера

Налаштування ESP32-WROOM-32 як вебсервера дозволяє керувати платою та підключеними до неї пристроями через Інтернет. Для цього використовуються бібліотеки WiFi.h та WebServer.h, які забезпечують зручний інтерфейс для роботи з Wi-Fi та створення вебсторінок та обробки HTTP-запитів відповідно. Перед початком роботи переконайтеся, що ці бібліотеки встановлені в вашому Arduino IDE.

Приклад налаштування вебсервера:

```
#include <WiFi.h>
#include <WebServer.h>

const char* ssid = "Archer";  // Введіть своє ім'я мережі Wi-Fi
const char* password = "12345678";  // Введіть свій пароль Wi-Fi
```

```
WebServer server(80);  // Створення об'єкту вебсервера на порті 80
void setup() {
  Serial.begin(115200);

  WiFi.begin(ssid, password);  // Підключення до Wi-Fi мережі
  while (WiFi.status() != WL_CONNECTED) {  // Очікування підключення до Wi-Fi
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");
  Serial.println(WiFi.localIP());

  server.on("/", handle_OnConnect);   // Встановлення обробника для головної сторінки
  server.onNotFound(handle_NotFound);  // Встановлення обробника для відсутніх сторінок

  server.begin();  // Запуск вебсервера
}
void loop() {
  server.handleClient();  // Обробка HTTP-запитів
}
```

Цей код демонструє створення простого вебсервера на платформі ESP32 за допомогою мови програмування Arduino. Основна мета цього вебсервера полягає в обслуговуванні HTTP-запитів, які можуть бути надіслані до плати через Wi-Fi мережу.

Такий підхід може бути використаний для реалізації різноманітних вебсервісів на ESP32, від простого вебінтерфейсу для керування пристроями до складніших систем моніторингу та керування.

3.7 Код реалізації та результат роботи проєкту

Код для моніторингу здоров'я пацієнтів на основі Інтернет-речей на вебсервері ESP32 наведено нижче. Необхідно встановити кілька бібліотек для

компіляції вихідного коду. Завантажуємо ці бібліотеки з вбудованого менеджеру та додаємо їх до нашого проєкту:

1) Arduino MAX30100 Library;
2) OneWire Library;
3) Dallas Temperature Library;
4) DHT11 Library.

graduate_work.ino

```
#include <DHT11.h>
#include <WiFi.h>
#include <WebServer.h>
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"
#include <OneWire.h>
#include <DallasTemperature.h>

float temperature, humidity, BPM, SpO2, bodytemperature;
const char* ssid = "Archer";
const char* password = "12345678";
WebServer server(80);
```

Рисунок 3.9 – Підключення бібліотек і оголошення змінних

Цей фрагмент коду налаштовує всі необхідні компоненти для створення системи моніторингу стану пацієнта, яка буде підключатися до Wi-Fi мережі та надавати доступ до вимірювань через вебсервер.

```
#define DHT11_PIN 18
#define DS18B20 5
#define REPORTING_PERIOD_MS     1000
```

Рисунок 3.10 – Оголошення пінів та констант

Ці рядки коду оголошують константи для пінів підключення датчиків і період звітності.

```
DHT11 dht(DHT11_PIN);
PulseOximeter pox;
uint32_t tsLastReport = 0;
OneWire oneWire(DS18B20);
DallasTemperature sensors(&oneWire);
```

Рисунок 3.11 – Ініціалізація об'єктів і налаштування

Ці налаштування ініціалізують об'єкти для взаємодії з датчиками DHT11, MAX30100 та DS18B20, забезпечуючи можливість зчитування відповідних даних про стан пацієнта.

```
void onBeatDetected()
{
  Serial.println("Beat!");
}
```

Рисунок 3.12 – Callback-функція для виявлення серцебиття

Ця функція використовується для обробки події виявлення удару серця і виведення відповідного повідомлення в серійний монітор для спостереження за роботою пульсоксиметра.

```
32  void setup() {
33    Serial.begin(115200);
34    pinMode(19, OUTPUT);
35    delay(100);
36
37    Serial.println("Connecting to ");
38    Serial.println(ssid);
39
40    WiFi.begin(ssid, password);
41
42    while (WiFi.status() != WL_CONNECTED) {
43    delay(1000);
44    Serial.print(".");
45    }
46    Serial.println("");
47    Serial.println("WiFi connected..!");
48    Serial.print("Got IP: ");  Serial.println(WiFi.localIP());
49
50    server.on("/", handle_OnConnect);
51    server.onNotFound(handle_NotFound);
52
53    server.begin();
54    Serial.println("HTTP server started");
55
56    Serial.print("Initializing pulse oximeter..");
57
```

а)

```
58    if (!pox.begin()) {
59      Serial.println("FAILED");
60      for (;;);
61    } else {
62      Serial.println("SUCCESS");
63      pox.setOnBeatDetectedCallback(onBeatDetected);
64    }
65
66    pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
67  }
```

б)

Рисунок 3.13 – Налаштування Wi-Fi та вебсервера у функції setup() (а, б)

Цей фрагмент коду представляє собою функцію setup(), яка є частиною скетчу для мікроконтролера Arduino (або сумісного мікроконтролера, наприклад, ESP32). Основні дії, які виконує ця функція:

– ініціалізація серійного порту: установлюється зв'язок з Серійним монітором зі швидкістю 115200 бод;

– налаштування піна як вихідний: пін 19 налаштовується як вихідний для керування певними пристроями чи індикаційними світлодіодами;

– підключення до WiFi мережі: виконується спроба підключення до WiFi мережі з використанням вказаного імені мережі (SSID) та пароля;

– очікування підключення до WiFi: виконується очікування успішного підключення до WiFi мережі, індикуючи це через Серійний монітор символами ".";

– ініціалізація HTTP сервера: запускається HTTP сервер на порту 80, обробники якого встановлені для кореневого шляху та не знайдених шляхів;

– ініціалізація пульсоксиметра: починається ініціалізація пульсоксиметра MAX30100. Якщо ініціалізація успішна, встановлюється зворотний виклик onBeatDetected для обробки виявлення удару серця;

– налаштування струму ІЧ світлодіоду: встановлюється поточний струм ІЧ світлодіоду пульсоксиметра.

Цей код ініціалізує необхідні ресурси та налаштовує пристрій для подальшої роботи з мережею WiFi, HTTP сервером та пульсоксиметром.

```
76  void loop() {
77    server.handleClient();
78    pox.update();
79    sensors.requestTemperatures();
80    int chk = dht.readTemperature();

89    if (millis() - tsLastReport > REPORTING_PERIOD_MS)
90    {
91      Serial.print("Room Temperature: ");
92      Serial.print(dht.readTemperature());
93      Serial.println("°C");
94
95      Serial.print("Room Humidity: ");
96      Serial.print(dht.readHumidity());
97      Serial.println("%");
98
99      Serial.print("BPM: ");
100     Serial.println(BPM);
101
102     Serial.print("SpO2: ");
103     Serial.print(SpO2);
104     Serial.println("%");
105
106     Serial.print("Body Temperature: ");
107     Serial.print(bodytemperature);
108     Serial.println("°C");
109
110     Serial.println("*********************************");
111     Serial.println();
112
113     tsLastReport = millis();
114   }
115
116 }
```

Рисунок 3.14 – Основний цикл у функції loop()

Цей код постійно виконується в циклі, обробляючи клієнтські запити, оновлюючи значення пульсоксиметра і датчиків температури, а також періодично відправляючи дані на серійний монітор для моніторингу.

```
110 void handle_OnConnect() {
111
112   server.send(200, "text/html", SendHTML(temperature, humidity, BPM, SpO2, bodytemperature));
113 }
114
115 void handle_NotFound(){
116   server.send(404, "text/plain", "Not found");
117 }
```

Рисунок 3.15 – Обробка HTTP-запитів та відправка відповіді у функціях handle_OnConnect() та handle_NotFound()

Ці функції відповідають на вхідні запити вебсервера відповідно до їхнього призначення: генерувати HTML-сторінку з даними стану системи або відправляти відповідь про помилку, якщо запитаний ресурс не існує.

```
127   String SendHTML(float temperature,float humidity,float BPM,float SpO2, float bodytemperature){
```

Рисунок 3.16 – Функція для формування HTML-сторінки

Ця функція дозволяє динамічно створювати HTML-сторінки на основі актуальних даних системи і передавати їх клієнтам через вебсервер.

```
WiFi connected..!
Got IP: 192.168.0.112
HTTP server started
Initializing pulse oximeter..SUCCESS
Room Temperature: 28.00°C
Room Humidity: 60.00%
BPM: 0.00
SpO2: 0.00%
Body Temperature: 29.62°C
**********

Room Temperature: 28.00°C
Room Humidity: 60.00%
BPM: 0.00
SpO2: 0.00%
Body Temperature: 29.50°C
**********

Room Temperature: 28.00°C
Room Humidity: 60.00%
BPM: 43.94
SpO2: 0.00%
Body Temperature: 30.50°C
**********

Room Temperature: 28.00°C
Room Humidity: 60.00%
BPM: 72.00
SpO2: 96.00%
Body Temperature: 31.56°C
**********
```

Рисунок 3.17 – Результат роботи комплексу в Serial Monitor

Копіюємо створену IP-адресу та водимо її в браузер вашего пристрою, та бачимо наступне:

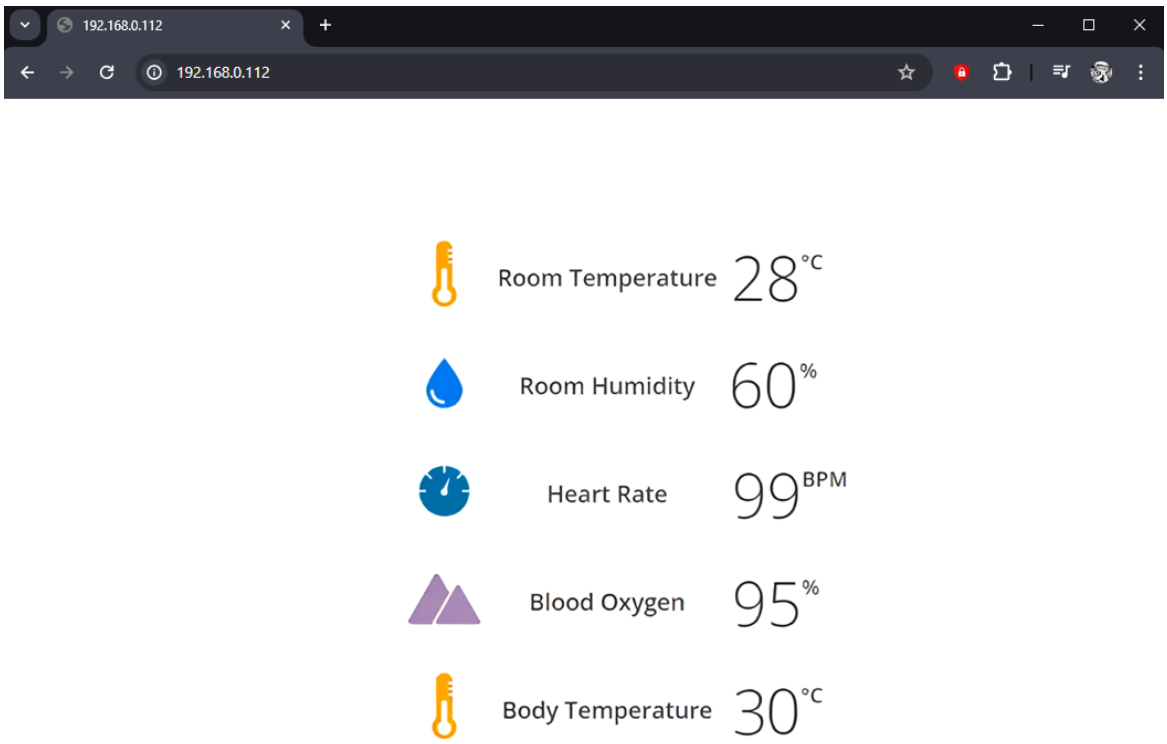


Рисунок 3.18 – Десктопний варіант відображення даних

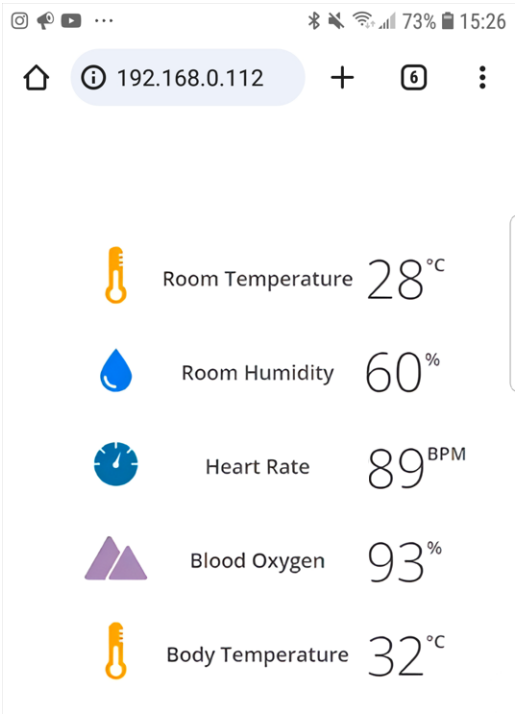


Рисунок 3.19 – Відображення даних з мобільного пристрою

Таким чином, ми реалізували проєкт для моніторингу здоров'я пацієнтів на основі Інтернет-речей (IoT) з використанням вебсервера ESP32. Було встановлено кілька необхідних бібліотек: Arduino MAX30100 Library, OneWire Library, Dallas Temperature Library та DHT11 Library, для успішної компіляції та виконання вихідного коду. Результати роботи проєкту можна побачити у серійному моніторі та через вебінтерфейс. Ввівши отриману IP-адресу в браузер, користувач може переглядати дані як на ПК, так і на мобільному пристрої.

3.8 Інтеграція комплексу моніторингу стану пацієнта в інших галузях

Комплекс моніторингу стану пацієнта на основі ESP32, розроблений для медичних застосувань, має потенціал для широкого використання в різних галузях. Завдяки своїм характеристикам, таким як бездротова функціональність, реальний час моніторингу та висока точність, цей комплекс може бути адаптований для використання в інших сферах, де важливі постійний моніторинг та збір даних про фізичний стан.

Даний комплекс може використовуватися в різних галузях. Наприклад, у спортивній індустрії, розумному будинку, дистанційному моніторингу здоров'я літніх людей чи для професійного використання в галузі охорони здоров'я.

Розглянемо одну із галузей більш детально. Галузь спортивної індустрії вимагає постійного моніторингу фізичного стану спортсменів для оптимізації тренувального процесу та попередження травм. Комплекс моніторингу стану пацієнта на основі ESP32 може стати незамінним інструментом для тренерів та спортивних лікарів завдяки кільком ключовим перевагам.

По-перше, бездротова функціональність дозволяє спортсменам вільно рухатись під час тренувань, при цьому дані про їх фізичний стан збираються в реальному часі. Це включає частоту серцевих скорочень, рівень кисню в крові, температуру тіла та інші важливі показники. Завдяки цьому тренери можуть

оперативно отримувати інформацію та вносити корективи в тренувальний процес, підвищуючи його ефективність.

По-друге, можливість інтеграції з мобільними додатками та хмарними сервісами дозволяє зберігати дані, аналізувати їх та надавати рекомендації спортсменам і тренерам. Це допомагає створювати персоналізовані тренувальні програми, що враховують індивідуальні особливості та фізичний стан кожного спортсмена.

Нарешті, зручність використання комплексу дозволяє легко інтегрувати його в існуючі спортивні програми та інфраструктуру. Інтуїтивно зрозумілий інтерфейс та можливість налаштування різних функцій роблять його ідеальним інструментом для моніторингу стану спортсменів як під час тренувань, так і під час змагань.

Таким чином, комплекс моніторингу стану пацієнта на основі ESP32 має великий потенціал для застосування в різних галузях, де важливий постійний моніторинг фізичного стану та оперативний аналіз даних.

Висновок до розділу 3

Загалом, розділ був присвячений створенню системи моніторингу стану пацієнта на базі мікроконтролера ESP32. Основна мета роботи полягала у створенні доступної, простої у використанні та ефективної системи для застосування як вдома, так і в медичних закладах. Для моніторингу фізіологічних показників обрано три датчики: MAX30100 для вимірювання частоти серцевих скорочень та рівня кисню в крові, DS18B20 для вимірювання температури тіла, та DHT11 для визначення температури і вологості навколишнього середовища.

Аналіз існуючих аналогів показав їх високу вартість та складність, що обґрунтовує розробку більш доступної системи. Компоненти підключаються до мікроконтролера ESP32, який отримує, обробляє та передає дані на дисплей або сервер для подальшого аналізу.

Компоненти системи змонтовані на платі для забезпечення компактності. Після зборки система пройшла тестування, яке підтвердило її точність та стабільність роботи. Кошторис підтверджує доступність системи.

Програмне забезпечення розроблено на мові програмування Arduino, що забезпечує обробку даних з сенсорів та їх передачу на вебінтерфейс. Після написання коду та його завантаження на ESP32, проведено тестування та налагодження програми для досягнення оптимальної роботи пристрою. Вебсервер на ESP32 дозволяє дистанційний моніторинг стану пацієнта.

Загалом, розроблена система є доступною, надійною та точною, що дозволяє її використання в різних умовах та забезпечує можливість дистанційного моніторингу стану пацієнта. Використання ESP32 та стандартних датчиків забезпечує високу функціональність при мінімальних витратах, роблячи систему ефективним інструментом для оцінки здоров'я пацієнтів.

ВИСНОВКИ

Під час вивчення та практичної реалізації комплексу моніторингу стану пацієнта на базі ESP32 було розглянуто різноманітні технічні аспекти та можливості управління проєктами за допомогою мікроконтролерів та мережевого зв'язку. Мета дослідження полягала у вивченні технічних можливостей та реалізації вебсервера на мікроконтролері Arduino ESP32-WROOM-32 з метою підвищення ефективності та функціональності відстежування стану здоров'я пацієнта вдома.

Виконано поставлені завдання: розроблено програмне забезпечення для мікроконтролера ESP32, що забезпечує підключення до мережі Інтернет та обробку даних від сенсорів. Інтегровано сенсори для вимірювання температури, вологості, частоти серцевих скорочень та рівня кисню в крові з мікроконтролером ESP32. Реалізовано вебсервер для зберігання та відображення зібраних даних у реальному часі, а також створено інтерфейс користувача для доступу до цих даних через веббраузер. Проведено тестування та налагодження системи для забезпечення стабільної роботи та точності вимірювань.

Актуальність проблематики полягає в необхідності відстежування стану здоров'я пацієнтів вдома, особливо для пацієнтів похилого віку або тих, хто потребує постійного моніторингу. Реалізація комплексу моніторингу на базі ESP32 використовує вебсервер для відстеження параметрів здоров'я пацієнта, таких як частота серцевих скорочень, рівень кисню в крові та температура тіла. Цей підхід дозволяє ефективно взаємодіяти з медичними пристроями та забезпечує зручний інтерфейс користувача для відстеження показників здоров'я.

Інтеграція ідеї покращення полягає в тому, що розроблений комплекс моніторингу здоров'я пацієнта на базі ESP32 пропонує інноваційну систему, яка автоматизує відстеження стану здоров'я пацієнта з використанням

вебсервера. Ця система може бути використана для покращення догляду за пацієнтами та підвищення якості медичної допомоги.

Отже, розглянуті технічні аспекти та можливості управління проєктом підкреслюють потенціал використання мікроконтролерів та мережевого зв'язку для вирішення актуальних проблем у галузі медичних технологій. Вивчення та впровадження інноваційних технологій, які базуються на ESP32, відкриває нові можливості для покращення охорони здоров'я та підвищення якості життя пацієнтів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Patient monitoring device and patient monitoring alarm method : patent WO2023126012 China : A61B 5/00 2006.1 G08B 23/00 2006.1 A61B 5/145 2006.1 G08B 21/02 2006.1. No. WO2023126012 ; international filing date on 03.01.2023 ; publication on 06.07.2023. URL: https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2023126012&_cid=P21-LW596D-64272-1 (Last accessed: 12.05.2024).

2. Remote controlled complex patient monitoring system by using web camera : 1020040080801 Republic of Korea : G06Q 10/00. No 1020040080801 ; applied on 11.10.2004 ; published on 17.04.2006, Bulletin no. A. URL: https://patentscope.wipo.int/search/en/detail.jsf?docId=KR823836&_cid=P21-LW596D-64272-1 (Last accessed: 12.05.2024).

3. Smart healthcare solutions for patient monitoring and predicting pathological status using ai and iot-based technology : 2021101734 Australia : G16H 50/20 G16H 30/40 G16H 50/70 G16H 80/00 G16Y 10/60. No 2021101734 : applied on 06.04.2021 ; published on 29.04.2021, Bulletin no. A4. URL: https://patentscope.wipo.int/search/en/detail.jsf?docId=AU322897549&_cid=P21-LW596D-64272-1 (Last accessed: 12.05.2024).

4. Classification Techniques for Arrhythmia Patterns Using Convolutional Neural Networks and Internet of Things (IoT) Devices. URL: https://ieeexplore.ieee.org/document/9832886 (Last accessed: 12.05.2024).

5. Evaluation of a Telemedicine-Based Service for the Follow-Up and Monitoring of Patients Treated With Oral Anticoagulant Therapy. URL: https://ieeexplore.ieee.org/document/4668462 (Last accessed: 12.05.2024).

6. Systematic Review of Platforms Used for Remote Monitoring of Vital Signs in Patients With Hypertension, Asthma and/or Chronic Obstructive Pulmonary Disease. URL: https://ieeexplore.ieee.org/document/8886476 (Last accessed: 12.05.2024).

7. MAX30100 – Heart Rate Oxygen Pulse Sensor. URL: https://components101.com/sensors/max30100-heart-rate-oxygen-pulse-sensor-pinout-features-datasheet (Last accessed: 13.05.2024).

8. DS18B20 Temperature Sensor. URL: https://components101.com/sensors/ds18b20-temperature-sensor (Last accessed: 13.05.2024).

9. DHT11–Temperature and Humidity Sensor. URL: https://components101.com/sensors/dht11-temperature-sensor (Last accessed: 13.05.2024).

10. ESP32 HTTP GET and HTTP POST with Arduino IDE (JSON, URL Encoded, Text). URL: https://randomnerdtutorials.com/esp32-http-get-post-arduino/ (Last accessed: 13.05.2024).

11. The Components of Patient Monitoring Systems. URL: https://vantagemedtech.com/the-components-of-patient-monitoring-systems/ (Last accessed: 13.05.2024).

12. Bush S. Updated: Arduino announces FPGA board, ATmega4809 in Uno Wi-Fi mk2, cloud-based IDE and IoT hardware. URL: https://www.electronicsweekly.com/news/products/bus-systems-sbcs/arduino-announced-fpga-board-new-atmega-uno-wi-fi-2018-05/ (Last accessed: 13.05.2024).

13. Monk, Simon (2022). Programming Arduino: Getting Started with Sketches (3rd ed.). McGraw-Hill Education. ISBN 978-1264676989. URL: https://www.accessengineeringlibrary.com/content/book/9781264676989/chapter/chapter4#/c9781264676989ch04lev1sec01 (Last accessed: 13.05.2024).

14. ESP32-DevKitC V4 Getting Started Guide URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/hw-reference/esp32/get-started-devkitc.html (Last accessed: 13.05.2024).

15. ESP32-S3-DEV-KIT-N8R8. URL: https://www.waveshare.com/wiki/ESP32-S3-DEV-KIT-N8R8 (Last accessed: 13.05.2024).

16. Monitoring System Based on IoT Sensor Data with Complex Event Provessing and Artificial Neural Networks for Patient Stress Detection. URL: https://www.researchgate.net/publication/331951035_Monitoring_System_Based_on_IoT_Sensor_Data_with_Complex_Event_Provessing_and_Artificial_Neural_Networks_for_Patient_Stress_Detection (Last accessed: 13.05.2024).

17. The Role of IoT in Enhancing Healthcare: Remote Patient Monitoring and Beyond. URL: https://www.linkedin.com/pulse/role-iot-enhancing-healthcare-remote-patient-monitoring-beyond-1e/ (Last accessed: 13.05.2024).

18. ScalableDigitalHealth (SDH): An IoT-Based Scalable Framework for Remote Patient Monitoring. URL: https://www.mdpi.com/1424-8220/24/4/1346 (Last accessed: 13.05.2024).

19. Internet of Things system for monitoring patient’s state. URL: https://www.researchgate.net/publication/338134635_Internet_of_Things_system_for_monitoring_patient's_state (Last accessed: 13.05.2024).

20. IoT Based Patient Monitoring System using Sensors to Detect, Analyse and Monitor Two Primary Vital Signs. URL: https://iopscience.iop.org/article/10.1088/1742-6596/1535/1/012004 (Last accessed: 13.05.2024).

ДОДАТОК А

Довідка

про перевірку на унікальність пояснювальної записки

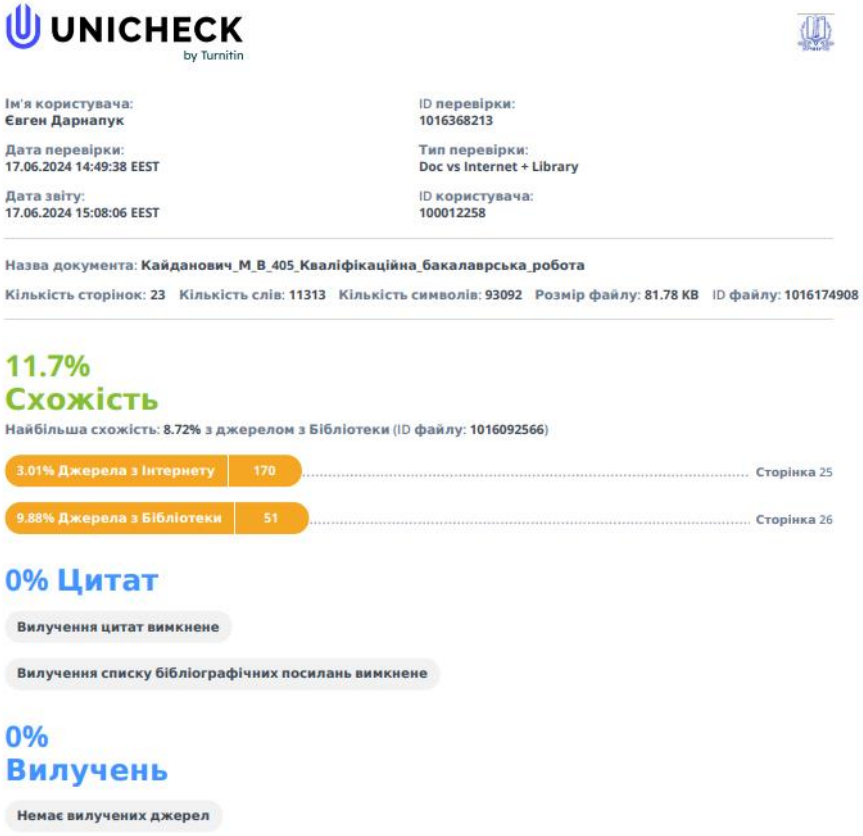
бакалаврської кваліфікаційної роботи на тему:

«Комплекс моніторингу стану пацієнта на базі ESP32»

студента спеціальності 123 «Комп'ютерна інженерія», 405 групи
Кайданович Микита Вячеславович
прізвище, ім'я, по-батькові

Перевірку тексту здійснено сервісом: онлайн-сервіс Unicheck

Результат перевірки тексту бакалаврської кваліфікаційної роботи: схожість складає 11.7% .



Здобувач: Керівник:

студент 405 групи ст. викладач

_____ М. В. Кайданович _____ Є. С. Дарнапук
підпис ініціали, прізвище підпис ініціали, прізвище

Дата: «___» _____ 2024 р.

ДОДАТОК Б

Код програми

```
#include <DHT11.h>
#include <WiFi.h>
#include <WebServer.h>
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"
#include <OneWire.h>
#include <DallasTemperature.h>



#define DHT11_PIN 18
#define DS18B20 5
#define REPORTING_PERIOD_MS     1000

float temperature, humidity, BPM, SpO2, bodytemperature;



const char* ssid = "Archer";
const char* password = "123456789";

DHT11 dht(DHT11_PIN);
PulseOximeter pox;
uint32_t tsLastReport = 0;
OneWire oneWire(DS18B20);
DallasTemperature sensors(&oneWire);



WebServer server(80);

void onBeatDetected()
{
  Serial.println("Beat!");
}

void setup() {
  Serial.begin(115200);
  pinMode(19, OUTPUT);
  delay(100);
```

```
  Serial.println("Connecting to ");
  Serial.println(ssid);


  WiFi.begin(ssid, password);


  while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");  Serial.println(WiFi.localIP());

  server.on("/", handle_OnConnect);
  server.onNotFound(handle_NotFound);

  server.begin();
  Serial.println("HTTP server started");

  Serial.print("Initializing pulse oximeter..");

  if (!pox.begin()) {
    Serial.println("FAILED");
    for (;;);
  } else {
    Serial.println("SUCCESS");
    pox.setOnBeatDetectedCallback(onBeatDetected);
  }

   pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);



}
void loop() {
  server.handleClient();
  pox.update();
  sensors.requestTemperatures();
```

```
  int chk = dht.readTemperature();

  temperature = dht.readTemperature();
  humidity = dht.readHumidity();
  BPM = pox.getHeartRate();
  SpO2 = pox.getSpO2();
  bodytemperature = sensors.getTempCByIndex(0);


  if (millis() - tsLastReport > REPORTING_PERIOD_MS)
  {
    Serial.print("Room Temperature: ");
    Serial.print(dht.readTemperature());
    Serial.println("°C");

    Serial.print("Room Humidity: ");
    Serial.print(dht.readHumidity());
    Serial.println("%");

    Serial.print("BPM: ");
    Serial.println(BPM);

    Serial.print("SpO2: ");
    Serial.print(SpO2);
    Serial.println("%");

    Serial.print("Body Temperature: ");
    Serial.print(bodytemperature);
    Serial.println("°C");

    Serial.println("*********************************");
    Serial.println();

    tsLastReport = millis();
  }

}

void handle_OnConnect() {
```

```
   server.send(200, "text/html", SendHTML(temperature, humidity, BPM, SpO2,
bodytemperature));
}

void handle_NotFound(){
  server.send(404, "text/plain", "Not found");
}

  String SendHTML(float temperature,float humidity,float BPM,float SpO2, float
bodytemperature){
  String ptr = "<!DOCTYPE html>";
  ptr +="<html>";
  ptr +="<head>";
    ptr   +="<meta   name='viewport'   content='width=device-width,   initial-
scale=1.0'>";
                                    ptr                                    +="<link
href='https://fonts.googleapis.com/css?family=Open+Sans:300,400,600'
rel='stylesheet'>";
  ptr +="<style>";
  ptr +="html { font-family: 'Open Sans', sans-serif; display: block; margin:
0px auto; text-align: center;color: #444444;}";
  ptr +="body{margin: 0px;} ";
  ptr +="h1 {margin: 50px auto 30px;} ";
   ptr  +=".side-by-side{display:  table-cell;vertical-align:  middle;position:
relative;}";
  ptr +=".text{font-weight: 600;font-size: 19px;width: 200px;}";
  ptr +=".reading{font-weight: 300;font-size: 50px;padding-right: 25px;}";
  ptr +=".temperature .reading{color: #F29C1F;}";
  ptr +=".humidity .reading{color: #3B97D3;}";
  ptr +=".BPM .reading{color: #006EAA;}";
  ptr +=".SpO2 .reading{color: #955BA5;}";
  ptr +=".bodytemperature .reading{color: #F29C1F;}";
  ptr +=".superscript{font-size: 17px;font-weight: 600;position: absolute;top:
10px;}";
  ptr +=".data{padding: 10px;}";
  ptr +=".container{display: table;margin: 0 auto;}";
  ptr +=".icon{width:65px}";
  ptr +="</style>";
  ptr +="</head>";
  ptr +="<body>";
  ptr +="<div class='container'>";
```

```
  ptr +="<div class='data temperature'>";
  ptr +="<div class='side-by-side icon'>";
   ptr  +="<svg  enable-background='new  0  0  19.438  54.003'height=54.003px
id=Layer_1  version=1.1  viewBox='0  0  19.438  54.003'width=19.438px  x=0px
xml:space=preserve                              xmlns=http://www.w3.org/2000/svg
xmlns:xlink=http://www.w3.org/1999/xlink   y=0px><g><path   d='M11.976,8.82v-
2h4.084V6.063C16.06,2.715,13.345,0,9.996,0H9.313C5.965,0,3.252,2.715,3.252,6.0
63v30.982";
                                                                          ptr
+="C1.261,38.825,0,41.403,0,44.286c0,5.367,4.351,9.718,9.719,9.718c5.368,0,9.7
19-4.351,9.719-9.718";
    ptr   +="c0-2.943-1.312-5.574-3.378-7.355V18.436h-3.914v-2h3.914v-2.808h-
4.084v-2h4.084V8.82H11.976z M15.302,44.833";
   ptr  +="c0,3.083-2.5,5.583-5.583,5.583s-5.583-2.5-5.583-5.583c0-2.279,1.368-
4.236,3.326-5.104V24.257C7.462,23.01,8.472,22,9.719,22";
                                                                          ptr
+="s2.257,1.01,2.257,2.257V39.73C13.934,40.597,15.302,42.554,15.302,44.833z'fi
ll=#F29C21 /></g></svg>";
  ptr +="</div>";
  ptr +="<div class='side-by-side text'>Room Temperature</div>";
  ptr +="<div class='side-by-side reading'>";
  ptr +=(int)temperature;
  ptr +="<span class='superscript'>&deg;C</span></div>";
  ptr +="</div>";

  ptr +="<div class='data humidity'>";
  ptr +="<div class='side-by-side icon'>";
  ptr +="<svg enable-background='new 0 0 29.235 40.64'height=40.64px id=Layer_1
version=1.1 viewBox='0 0 29.235 40.64'width=29.235px x=0px xml:space=preserve
xmlns=http://www.w3.org/2000/svg      xmlns:xlink=http://www.w3.org/1999/xlink
y=0px><path
d='M14.618,0C14.618,0,0,17.95,0,26.022C0,34.096,6.544,40.64,14.618,40.64s14.61
7-6.544,14.617-14.617";
  ptr +="C29.235,17.95,14.618,0,14.618,0z M13.667,37.135c-5.604,0-10.162-4.56-
10.162-10.162c0-0.787,0.638-1.426,1.426-1.426";
                                                                          ptr
+="c0.787,0,1.425,0.639,1.425,1.426c0,4.031,3.28,7.312,7.311,7.312c0.787,0,1.4
25,0.638,1.425,1.425";
  ptr +="C15.093,36.497,14.455,37.135,13.667,37.135z'fill=#3C97D3 /></svg>";
  ptr +="</div>";
```

```
  ptr +="<div class='side-by-side text'>Room Humidity</div>";
  ptr +="<div class='side-by-side reading'>";
  ptr +=(int)humidity;
  ptr +="<span class='superscript'>%</span></div>";
  ptr +="</div>";

  ptr +="<div class='data Heart Rate'>";
  ptr +="<div class='side-by-side icon'>";
   ptr  +="<svg  enable-background='new  0  0  40.542  40.541'height=40.541px
id=Layer_1  version=1.1  viewBox='0  0  40.542  40.541'width=40.542px  x=0px
xml:space=preserve                              xmlns=http://www.w3.org/2000/svg
xmlns:xlink=http://www.w3.org/1999/xlink  y=0px><g><path  d='M34.313,20.271c0-
0.552,0.447-1,1-1h5.178c-0.236-4.841-2.163-9.228-5.214-12.593l-3.425,3.424";
  ptr +="c-0.195,0.195-0.451,0.293-0.707,0.293s-0.512-0.098-0.707-0.293c-0.391-
0.391-0.391-1.023,0-1.414l3.425-3.424";
                    ptr                    +="c-3.375-3.059-7.776-4.987-12.634-
5.215c0.015,0.067,0.041,0.13,0.041,0.202v4.687c0,0.552-0.447,1-1,1s-1-0.448-1-
1V0.25";
                       ptr                        +="c0-0.071,0.026-0.134,0.041-
0.202C14.39,0.279,9.936,2.256,6.544,5.385l3.576,3.577c0.391,0.391,0.391,1.024,
0,1.414";
        ptr       +="c-0.195,0.195-0.451,0.293-0.707,0.293s-0.512-0.098-0.707-
0.293L5.142,6.812c-2.98,3.348-4.858,7.682-5.092,12.459h4.804";
                      ptr                      +="c0.552,0,1,0.448,1,1s-0.448,1-
1,1H0.05c0.525,10.728,9.362,19.271,20.22,19.271c10.857,0,19.696-8.543,20.22-
19.271h-5.178";
     ptr    +="C34.76,21.271,34.313,20.823,34.313,20.271z    M23.084,22.037c-
0.559,1.561-2.274,2.372-3.833,1.814";
   ptr  +="c-1.561-0.557-2.373-2.272-1.815-3.833c0.372-1.041,1.263-1.737,2.277-
1.928L25.2,7.202L22.497,19.05";
        ptr       +="C23.196,19.843,23.464,20.973,23.084,22.037z'fill=#26B999
/></g></svg>";
  ptr +="</div>";
  ptr +="<div class='side-by-side text'>Heart Rate</div>";
  ptr +="<div class='side-by-side reading'>";
  ptr +=(int)BPM;
  ptr +="<span class='superscript'>BPM</span></div>";
  ptr +="</div>";

  ptr +="<div class='data Blood Oxygen'>";
  ptr +="<div class='side-by-side icon'>";
```

```
   ptr  +="<svg  enable-background='new  0  0  58.422  40.639'height=40.639px
id=Layer_1  version=1.1  viewBox='0  0  58.422  40.639'width=58.422px  x=0px
xml:space=preserve                          xmlns=http://www.w3.org/2000/svg
xmlns:xlink=http://www.w3.org/1999/xlink                      y=0px><g><path
d='M58.203,37.754l0.007-0.004L42.09,9.935l-0.001,0.001c-0.356-0.543-0.969-
0.902-1.667-0.902";
      ptr     +="c-0.655,0-1.231,0.32-1.595,0.808l-0.011-0.007l-0.039,0.067c-
0.021,0.03-0.035,0.063-0.054,0.094L22.78,37.692l0.008,0.004";
                        ptr                      +="c-0.149,0.28-0.242,0.594-
0.242,0.934c0,1.102,0.894,1.995,1.994,1.995v0.015h31.888c1.101,0,1.994-
0.893,1.994-1.994";
    ptr   +="C58.422,38.323,58.339,38.024,58.203,37.754z'fill=#955BA5   /><path
d='M19.704,38.674l-0.013-0.004l13.544-23.522L25.13,1.156l-
0.002,0.001C24.671,0.459,23.885,0,22.985,0";
  ptr +="c-0.84,0-1.582,0.41-2.051,1.038l-0.016-0.01L20.87,1.114c-0.025,0.039-
0.046,0.082-0.068,0.124L0.299,36.851l0.013,0.004";
                                                                      ptr
+="C0.117,37.215,0,37.62,0,38.059c0,1.412,1.147,2.565,2.565,2.565v0.015h16.989
c-0.091-0.256-0.149-0.526-0.149-0.813";
         ptr       +="C19.405,39.407,19.518,39.019,19.704,38.674z'fill=#955BA5
/></g></svg>";
  ptr +="</div>";
  ptr +="<div class='side-by-side text'>Blood Oxygen</div>";
  ptr +="<div class='side-by-side reading'>";
  ptr +=(int)SpO2;
  ptr +="<span class='superscript'>%</span></div>";
  ptr +="</div>";

  ptr +="<div class='data Body Temperature'>";
  ptr +="<div class='side-by-side icon'>";
   ptr  +="<svg  enable-background='new  0  0  19.438  54.003'height=54.003px
id=Layer_1  version=1.1  viewBox='0  0  19.438  54.003'width=19.438px  x=0px
xml:space=preserve                          xmlns=http://www.w3.org/2000/svg
xmlns:xlink=http://www.w3.org/1999/xlink   y=0px><g><path   d='M11.976,8.82v-
2h4.084V6.063C16.06,2.715,13.345,0,9.996,0H9.313C5.965,0,3.252,2.715,3.252,6.0
63v30.982";
                                                                      ptr
+="C1.261,38.825,0,41.403,0,44.286c0,5.367,4.351,9.718,9.719,9.718c5.368,0,9.7
19-4.351,9.719-9.718";
    ptr   +="c0-2.943-1.312-5.574-3.378-7.355V18.436h-3.914v-2h3.914v-2.808h-
4.084v-2h4.084V8.82H11.976z M15.302,44.833";
```

```
   ptr  +="c0,3.083-2.5,5.583-5.583,5.583s-5.583-2.5-5.583-5.583c0-2.279,1.368-
4.236,3.326-5.104V24.257C7.462,23.01,8.472,22,9.719,22";
                                                                    ptr
+="s2.257,1.01,2.257,2.257V39.73C13.934,40.597,15.302,42.554,15.302,44.833z'fi
ll=#F29C21 /></g></svg>";
  ptr +="</div>";
  ptr +="<div class='side-by-side text'>Body Temperature</div>";
  ptr +="<div class='side-by-side reading'>";
  ptr +=(int)bodytemperature;
  ptr +="<span class='superscript'>&deg;C</span></div>";
  ptr +="</div>";

  ptr +="</div>";
  ptr +="</body>";
  ptr +="</html>";
  return ptr;
}
```

ДОДАТОК В

Графічні матеріали КБР

Рисунок В.1 – Блок-схема роботи пристрою