

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

«До захисту допущено»
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис)
“ ____ ” червня 2025 р.

Дипломний проєкт
на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Системне програмування та спеціалізовані комп’ютерні системи»
123 «Комп’ютерна інженерія»

**на тему: «Багатофункціональний пристрій на базі ESP32 для моніторингу
параметрів середовища»**

Виконав: студент IV курсу, групи KB-12
Левчук Іван Володимирович

(підпис)

Керівник ст. викладач каф. СПіСКС, к.т.н.
Коляда Костянтин Вячеславович

(підпис)

Консультант з нормоконтролю, доц. каф. СПіСКС, к.т.н., доц.,
Клятченко Ярослав Михайлович

(підпис)

Рецензент доц. каф. ОТ, к.т.н., доц.,
Марковський Олександр Петрович

(підпис)

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент

(підпис)

Київ – 2025 року

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Освітньо-професійною програма

**«Системне програмування та спеціалізовані
комп'ютерні системи»**

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ
(підпис)

“ ____ ” червня 2024 р.

ЗАВДАННЯ

на дипломний проєкт студента

Левчука Івана Володимировича

1. Тема проєкту «Багатофункціональний пристрій на базі ESP32 для моніторингу параметрів середовища»,

керівник проєкту ст. викладач каф. СПіСКС, к.т.н.

Коляда Костянтин Вячеславович

затверджені наказом по університету від _____

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту див. Технічне завдання

4. Зміст пояснювальної записки:

1. Аналіз існуючих рішень

2. Аналіз обраних технологій

3. Програмне Забезпечення

5. Перелік графічного матеріалу:

1) Структурна схема

- 2) Функціональна схема
- 3) Схема обміну даними
- 4) Інтерфейс користувача

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я. М., доцент кафедри СПіСКС, к.т.н., доцент		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Розроблення та узгодження технічного завдання	19.01.2025	
2	Вивчення літератури за тематикою роботи	02.02.2025	
3	Розроблення структури додатку	16.02.2025	
4	Аналіз існуючих рішень	02.03.2025	
5	Програмна реалізація додатку	16.03.2025	
6	Розроблення дизайну додатку	13.04.2025	
7	Тестування додатку	20.04.2025	
8	Підготовка матеріалів першого розділу дипломного проекту	27.05.2025	
9	Підготовка матеріалів другого розділу дипломного проекту	04.05.2025	
10	Підготовка матеріалів графічної частини проекту	11.05.2025	
11	Підготовка матеріалів третього розділу дипломного проекту	18.05.2025	
12	Оформлення технічної документації проекту	25.05.2025	

Студент

Іван ЛЕВЧУК

Керівник проекту

Костянтин КОЛЯДА

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (56 с., 18 рис., список використаної літератури з 15 найменувань, 3 додатки, 20 слайдів).

Об'єкт розробки – створення багатофункціонального пристрою на базі мікроконтролера ESP32 для моніторингу параметрів навколишнього середовища, таких як температура, вологість, наявність руху та відстань до об'єктів. Пристрій забезпечує зчитування даних із сенсорів, обробку інформації та її передачу через вбудований вебсервер у форматі JSON для подальшої візуалізації.

Система дозволяє: здійснювати бездротовий обмін даними між пристроєм та клієнтом без використання зовнішніх серверів; відображати дані у вебінтерфейсі з будь-якого пристрою, підключеного до однієї WiFi мережі; контролювати стан сенсорів у реальному часі; формувати відповідь у форматі JSON для інтеграції з іншими застосунками.

В процесі розробки:

- проведено аналіз аналогічних вбудованих систем та пристроїв на базі ESP32;
- сформульовано технічні вимоги до апаратного та програмного забезпечення;
- розроблено схему підключення та програмне забезпечення для ESP32 з обробкою даних сенсорів DHT22, HC-SR04 та PIR;
- реалізовано локальний вебсервер для передачі показників у форматі JSON;
- створено вебклієнт на Flask для виведення отриманих даних у зручному вигляді.

Розроблений пристрій може бути використаний у системах моніторингу мікроклімату, охоронних рішеннях, побутових автоматизованих системах та освітніх цілях. Завдяки відкритому протоколу передачі даних, розробка легко масштабована та інтегрується з іншими платформами.

Ключові слова: ESP32, вебсервер, Flask, DHT22, PIR, HC-SR04, сенсорна система, моніторинг, JSON, IoT.

ANNOTATION

The qualification project includes an explanatory report (56 pages, 18 figures, a list of used literature from 15 names, 3 appendices, 20 slides).

The object of the development is to create a multifunctional device based on the ESP32 microcontroller for monitoring environmental parameters such as temperature, humidity, motion detection, and distance to objects. The device ensures reading data from sensors, processing the information, and transmitting it via an embedded web server in JSON format for further visualization.

The system allows: wireless data exchange between the device and the client without using external servers; displaying data in a web interface from any device connected to the same WiFi network; real-time monitoring of sensor states; generating a response in JSON format for integration with other applications.

During development:

- an analysis of similar embedded systems and devices based on ESP32 was conducted;
- technical requirements for hardware and software were formulated;
- a wiring diagram and software for the ESP32 were developed to process data from the DHT22, HC-SR04, and PIR sensors;
- a local web server was implemented to transmit measurements in JSON format;
- a web client on Flask was created to display the obtained data in a user-friendly format.

The developed device can be used in microclimate monitoring systems, security solutions, home automation systems, and educational applications. Thanks to the open data transmission protocol, the development is easily scalable and integrates with other platforms.

Keywords:

ESP32, web server, Flask, DHT22, PIR, HC-SR04, sensor system, monitoring, JSON, IoT.

Поз.		Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	A4	ІАЛЦ.467200.002 ТЗ	Багатофункціональний пристрій на базі ESP32 для моніторингу параметрів середовища	Технічне завдання			
	A4	ІАЛЦ.467200.003 ТП	Багатофункціональний пристрій на базі ESP32 для моніторингу параметрів середовища	Відомість технічного проекту			
	A4	ІАЛЦ.467200.004 ПЗ	Багатофункціональний пристрій на базі ESP32 для моніторингу параметрів середовища	Пояснювальна записка			
	A3	ІАЛЦ.467200.005 Д1	Схема структурна				
	A3	ІАЛЦ.467200.006 Е2	Схема функціональна				

[illegible]

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ.....	2
2. ПІДСТАВА ДЛЯ РОЗРОБКИ	2
3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ	2
4. ДЖЕРЕЛА РОЗРОБКИ	2
5. ТЕХНІЧНІ ВИМОГИ	2
5.1. Вимоги до програмного продукту, що розробляється.....	2
5.2. Вимоги до апаратного забезпечення	3
5.3. Вимоги до програмного та апаратного забезпечення користувача.....	3
6. ЕТАПИ РОЗРОБКИ	4

					<i>ІАЛЦ.467200.002 ТЗ</i>			
Змін.	Арк.	№ докум.	Підпис	Дата				
Розроб.	Левчук І. В.				Багатофункціональний пристрій на базі ESP32 для моніторингу параметрів середовища <i>Технічне завдання</i>	Літ.	Арк.	Акрушів
Перевір.	Коляда К. В.						1	4
Консульт						КПІ ім. Ігоря Сікорського, ФПМ КВ-12		
Н. Контр.	Клятченко Я.М.							
Затверд.	Романкевич В.О							

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Багатофункціональний пристрій на базі ESP32 для моніторингу параметрів середовища».

Галузь застосування: моніторинг мікрокліматичних умов, присутності та просторових змін у приміщеннях для систем безпеки, енергозбереження та автоматизованого управління в адміністративних, виробничих і побутових об'єктах.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даної розробки є створення багатофункціонального пристрою на базі ESP32 для збору, моніторингу та передачі даних від сенсорів температури, вологості, руху та відстані, з можливістю віддаленого доступу через вебінтерфейс.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- підтримка передачі даних через вебінтерфейс;
- зчитування даних з датчиків температури, вологості, руху та відстані;
- обробка та збереження зібраних даних;

					ІАЛЦ.467200.002 ТЗ	Арк.
						2
Змн.	Арк.	№ докум.	Підпис	Дата		

- відображення показників у режимі реального часу;
- можливість віддаленого доступу до даних через локальну мережу;
- наявність простого і зручного вебінтерфейсу;
- автоматичне оновлення даних з визначеним інтервалом часу;

5.2.Вимоги до апаратного забезпечення

- Мікроконтролер: ESP32 з підтримкою WiFi та Bluetooth;
- Датчики: температури та вологості (DHT11/22 або аналог), руху (HC-SR501), відстані (HC-SR04 або аналог);
- Живлення: 5 В через USB або акумуляторна батарея;
- Підключення: можливість підключення до локальної WiFi мережі;
- Обсяг пам'яті ESP32: не менше 520 КБ SRAM та 4 МБ Flash;

5.3.Вимоги до програмного та апаратного забезпечення користувача

- Веббраузер з підтримкою HTML5 (Google Chrome, Firefox, Safari, Edge);
- Доступ до локальної мережі, в якій працює ESP32;
- Пристрій користувача: смартфон, планшет або комп'ютер з підключенням до мережі WiFi.

					ІАЛЦ.467200.002 ТЗ	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2025
2.	Розроблення та узгодження технічного завдання	30.04.2025
3.	Аналіз існуючих рішень	05.05.2025
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2025
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2025
6.	Підготовка графічної частини дипломного проекту	20.05.2025
7.	Оформлення документації дипломного проекту	25.05.2025
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2025

					ІАЛЦ.467200.002 ТЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

Пояснювальна записка
до дипломного проєкту
на тему: «Багатофункціональний пристрій на базі ESP32
для моніторингу параметрів середовища»

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ	3
ВСТУП	4
1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ	5
1.1 Загальна інформація про ESP32	5
1.2 Системи моніторингу на базі ESP32	6
1.2.1 Хмарна платформа Blynk	7
1.2.2 Платформа для автоматизації розумного будинку Home Assistant .	11
1.2.3 Серія розумних пристроїв Shelly	14
1.3 Обґрунтування теми	17
2. АНАЛІЗ ОБРАНИХ ТЕХНОЛОГІЙ	19
2.1 Мікроконтроллер ESP32.....	19
2.3 Онлайн-платформа для симуляції електронних проєктів Wokwi.....	24
2.4 Інтегроване середовище для розробки PlatformIO	27
2.5 Мова розмітки HTML	29
3. СТРУКТУРА І ВЕБЗАСТОСУНОК ДЛЯ ПРИСТРОЇВ З СИСТЕМОЮ МОНІТОРІНГУ	35
3.1 Структура системи моніторингу	35
3.1.1.Програмна частина ESP (скетч Arduino).....	36
3.1.2.Серверна частина (Flask)	40
3.1.3.Клієнтська частина (HTML/CSS).....	41
3.2. Функціонал пристрою системи керування.....	41
3.3 Інтерфейс керування пристроєм	43
3.4 Тестування програми.....	45
ВИСНОВКИ.....	54
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	55

ДОДАТКИ

Додаток А. Копії графічного матеріалу.

					ІАЛЦ.467200.004 ПЗ		
Змін.	Арк.	№ докум.	Підпис	Дата	Багатофункціональний пристрій на базі ESP32 моніторингу параметрів середовища Пояснювальна записка		
Розроб.		Левчук І. В.					
Перевір.		Коляда К. В.					
Консульт							
Н. Контр.		Клятченко Я.М.					
Затверд.		Романкевич В.О					
					Літ.	Арк.	Акрушів
						1	56
					КП ім. Ігоря Сікорського, ФПМ КВ-12		

- 1) ІАЛЦ.467200.005 Д1. Схема структурна
- 2) ІАЛЦ.467200.006 Е2. Схема функціональна
- 3) ІАЛЦ.467200.007 Д3. Обробка сенсорів. Схема алгоритму
- 4) ІАЛЦ.467200.008 Д4. Вебзастосунок. Схема алгоритму

Додаток Б. Фрагменти програмного коду

Додаток В. Презентація

					ІАЛЦ.467200.004 ПЗ	Арк.
						2
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

API — Application Programming Interface – Інтерфейс прикладного програмування, дозволяє взаємодію між різними програмними компонентами.

CoAP — Constrained Application Protocol — легкий протокол для IoT, оптимізований для пристроїв із обмеженими ресурсами, передає дані між клієнтом та сервером у форматі запитів і відповідей.

ESP32 — Енергоефективний мікроконтролер із вбудованими модулями Wi-Fi та Bluetooth, розроблений компанією Espressif Systems

HTTP — HyperText Transfer Protocol – Протокол передачі гіпертексту, використовується для комунікації між веббраузером та сервером.

IoT — Internet of Things – Інтернет речей

JSON — JavaScript Object Notation – Формат обміну даними між клієнтом і сервером, простий та легкий для машинної обробки.

MQTT — Message Queuing Telemetry Transport – Протокол обміну повідомленнями, легкий, для IoT-додатків, працює за принципом «публікація–підписка».

WebSocket — Двосторонній протокол обміну даними поверх TCP, забезпечує постійне з'єднання для інтерактивного обміну даними між клієнтом і сервером у реальному часі.

					ІАЛЦ.467200.004 ПЗ	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Сучасні системи автоматизації та моніторингу навколишнього середовища потребують гнучких, доступних і масштабованих рішень. Мікроконтролер ESP32 завдяки своїй високій продуктивності, підтримці бездротових протоколів зв'язку та численним можливостям інтеграції з різноманітними сенсорами і платформами, став основою для багатьох проєктів у сфері Інтернету речей .

У рамках цієї дипломної роботи було розроблено багатофункціональний пристрій на базі ESP32 для моніторингу параметрів середовища, зокрема температури, вологості, наявності руху та відстані до об'єктів.

Для візуалізації даних та інтерактивної взаємодії з користувачем реалізовано вебсервер та вебінтерфейс, що забезпечують локальний доступ до інформації без використання сторонніх хмарних сервісів. Такий підхід підвищує автономність системи, її безпеку та швидкість реагування.

Робота містить аналіз існуючих рішень, обґрунтування вибору апаратних і програмних компонентів, опис реалізації та тестування системи.

					ІАЛЦ.467200.004 ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Загальна інформація про ESP32

ESP32 — це потужна платформа для створення пристроїв Інтернету речей (IoT), що поєднує в собі двоядерний 32-бітний процесор, модулі WiFi та Bluetooth, розширені можливості енергозбереження та підтримку численних інтерфейсів. Цей мікроконтролер розроблений компанією Espressif Systems — китайським виробником з офісом у Шанхаї, що спеціалізується на розробці інтегральних схем (SoC) для бездротових комунікацій.

ESP32 був офіційно представлений 6 вересня 2016 року на пресконференції в готелі Shanghai Parkyard Hotel як "Cloud on Chip" — чип, що забезпечує пряме підключення пристроїв до хмари, об'єднуючи обчислювальні потужності та мережеву комунікацію в одному рішенні.

Його створення стало наступним етапом розвитку після великого успіху ESP8266, який популяризував WiFi серед DIY-розробників. Новий чип ESP32 отримав:

- двоядерний процесор Xtensa LX6 з тактовою частотою до 240 МГц;
- обчислювальну потужність до 600 DMIPS;
- підтримку WiFi (802.11 b/g/n) та Bluetooth (Classic + BLE);
- велику кількість GPIO, аналогових входів/виходів, та інших інтерфейсів (SPI, I2C, UART, I2S, PWM тощо);
- апаратну підтримку криптографії та режимів енергозбереження;
- гнучку архітектуру, що дозволяє працювати з аудіо, зображеннями, сенсорами та навіть іграми.

Презентація ESP32 також стала нагодою для оголошення раунду інвестицій серії В, очолюваного Fosun Group за участі Haier, Midea Group та інших стратегічних партнерів[1]. Це підкреслює важливу роль ESP32 у майбутньому розвитку індустрії IoT.

Під час заходу було оголошено про партнерства з провідними гравцями ринку:

					ІАЛЦ.467200.004 ПЗ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

- Haier — співпраця для розвитку платформи U+ Smart Life;
- Xiaomi — впровадження ESP32 в екосистему розумного дому;
- Alibaba (YunOS) — спільна розробка YoC (Yun on Chip), з інтегрованою підтримкою хмарних сервісів та захисту даних (ID²);
- Tuuya, Gizwits — платформи для хмарних IoT-сервісів та стандартизації застосування мікроконтролерів.

Ці співпраці демонструють важливість ESP32 не лише як апаратного рішення, а як ключової частини широкої екосистеми смарт-пристроїв.

ESP32 завдяки своїй універсальності знайшов застосування в:

- смарт-пристроях для дому та бізнесу;
- системах моніторингу навколишнього середовища;
- інтерактивних аудіосистемах;
- мобільних додатках з голосовим або сенсорним керуванням;
- промислових рішеннях;
- інноваційних навчальних платформах.

Ключовою особливістю є можливість запуску вбудованого вебсерверу та безпосередньої взаємодії з користувачем через браузер, що дозволяє створювати автономні системи збору та виведення даних.

1.2 Системи моніторингу на базі ESP32

На початковому етапі розробки системи моніторингу з використанням мікроконтролера ESP32 було проведено аналіз існуючих апаратно-програмних рішень, які дозволяють реалізувати збір і передачу даних з різних сенсорів (температури, вологості, руху, відстані) та їх візуалізацію через вебінтерфейс.

Один із найпоширеніших підходів — використання мікроконтролера (Arduino, ESP8266 або ESP32) у поєднанні з сенсорами та передачею даних через протоколи MQTT або HTTP. Часто такі рішення інтегруються з

					ІАЛЦ.467200.004 ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

платформами, як-от Blynk, ThingsBoard, Node-RED, Home Assistant або власними вебсерверами. MQTT є дуже зручним для великих систем, але для простих автономних рішень достатньо реалізувати HTTP-сервер безпосередньо на мікроконтролері.

Порівнявши різні підходи, було прийнято рішення створити власну систему, в якій ESP32 збирає дані з сенсорів та передає їх у JSON-форматі до Python-сервера на Flask. Це дозволяє досягти балансу між простотою реалізації, гнучкістю налаштування та повною автономністю системи.

1.2.1 Хмарна платформа Blynk

Blynk — це хмарна платформа для розробки IoT-систем, створена компанією Blynk Inc.. Основною метою Blynk є забезпечення простого та гнучкого інструменту для керування мікроконтролерами та пристроями Інтернету речей через мобільний або вебінтерфейс без необхідності розробки власного застосунку. Платформа призначена як для розробників-початківців, так і для професійного використання в комерційних та промислових рішеннях[2].

Blynk дозволяє реалізувати повноцінну систему моніторингу та керування пристроями за допомогою інтуїтивного інтерфейсу. Основний акцент зроблено на швидкому використанні прототипів: користувач може створити мобільний застосунок для ESP32 або іншого контролера за лічені хвилини.

Платформа підтримує широкий перелік пристроїв, зокрема ESP32, ESP8266, Arduino, Raspberry Pi, STM32, Particle, NodeMCU та інші. Зв'язок з пристроями реалізується за допомогою таких протоколів, як WiFi, Ethernet, GSM/LTE, Bluetooth та BLE. Крім цього, Blynk підтримує MQTT, HTTP-запити, IFTTT, що дозволяє інтегрувати систему з зовнішніми сервісами, голосовими помічниками та автоматизаційними платформами. Завдяки

					ІАЛЦ.467200.004 ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

гнучкій архітектурі, Blynk можна використовувати як з офіційною хмарою, так і з власним локальним сервером.

Платформа Blynk побудована на клієнт-серверній архітектурі та включає кілька основних компонентів: мобільний застосунок, Blynk.Cloud, пристрій-контролер та вебінтерфейс Blynk Console.

Мобільний застосунок доступний для Android та iOS і виконує функцію візуального інтерфейсу користувача. У ньому можна створювати панелі керування, використовуючи віртуальні віджети (кнопки, слайдери, графіки, індикатори тощо), які зв'язуються з піннами мікроконтролера через інтернет.

Хмарна інфраструктура Blynk.Cloud є центральним елементом, який обробляє комунікацію між застосунком і пристроями. Пристрій підключається до сервера через бібліотеку Blynk та постійно підтримує з'єднання. Сервер отримує дані від пристрою й передає їх на мобільний додаток або навпаки — надсилає команди керування від користувача на пристрій.

Локальний сервер може бути встановлений замість хмари для офлайн-роботи або підвищеної безпеки.

Blynk Console (Рисунок 1.1) — це вебінтерфейс для розробників, який дозволяє керувати пристроями, шаблонами, користувачами, подіями та аналітикою. Усі компоненти працюють у зв'язці через захищені канали передачі даних, що забезпечує надійність та масштабованість системи.

Платформа Blynk надає розробникам широкий набір функціональних можливостей для створення гнучких і зручних IoT-інтерфейсів. Основу користувацького досвіду складають візуальні віджети, які додаються до мобільного застосунку у форматі drag-and-drop. Серед найпопулярніших віджетів — кнопки (для увімкнення/вимкнення реле або інших пристроїв), слайдери (для зміни значень напруги, яскравості, температури), графіки (для візуалізації зібраних даних у реальному часі), індикатори, датчики геолокації, термометри, LED, LCD-дисплеї тощо.

					ІАЛЦ.467200.004 ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

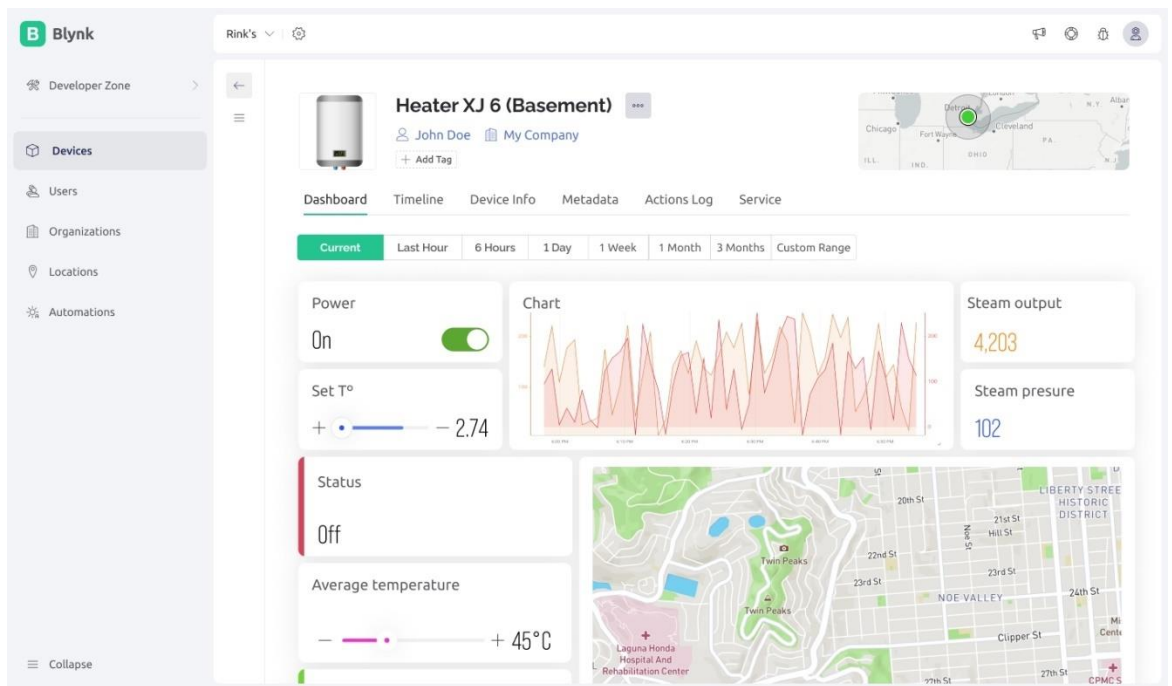


Рисунок 1.1 – Зовнішній вигляд Blynk.Console

Blynk підтримує створення автоматизованих сценаріїв, у яких можна задати дії при досягненні певного значення змінної, часу або події. Наприклад, система може надіслати сповіщення на смартфон при перевищенні температури або при спрацюванні датчика руху. Також підтримуються таймери, які дозволяють запускати події за розкладом (увімкнення світла, поливу тощо).

Одна з ключових переваг платформи — модульність інтерфейсу. Користувач самостійно формує вигляд панелі керування, додає або видаляє віджети без потреби у перекомпіляції коду. Це забезпечує високу гнучкість і зручність під час налаштування та подальшого використання системи.

Платформа Blynk надає зручні та інтуїтивно зрозумілі інтерфейси для взаємодії з IoT-пристроями як у мобільному застосунку, так і через вебсайт-консоль Blynk Console.

Мобільний застосунок Blynk(Рисунок 1.2) призначений для керування пристроями зі смартфонів і планшетів на базі Android та iOS. У ньому користувач може створювати віртуальні панелі керування, додаючи до них віджети: кнопки, слайдери, графіки, індикатори, дисплеї та багато інших. Всі елементи можна налаштовувати без знань програмування — достатньо

перетягнути віджет, вибрати пін мікроконтролера (наприклад, ESP32) та вказати основні параметри. Застосунок підтримує push-сповіщення, таймери, сценарії автоматизації та події за умовами.



Рисунок 1.2 – Зовнішній вигляд Blynk.Apps

Blynk Console — це вебінтерфейс для розробників, доступний через браузер. Вона дозволяє керувати пристроями, створювати шаблони, переглядати телеметричні дані в реальному часі, налаштовувати доступи для користувачів і створювати інформаційні панелі у вигляді вебсайту. Всі дії виконуються через сучасний графічний інтерфейс без необхідності роботи з кодом.

Загалом, інтерфейси Blynk забезпечують високу зручність використання, швидкість розробки та налаштування, що робить платформу доступною як для початківців, так і для досвідчених розробників.

Платформа Blynk має низку переваг, які роблять її дуже зручною для створення IoT-проектів. Насамперед це простота у використанні — розробник може створити мобільний інтерфейс для керування пристроєм без знань мобільного програмування. Завдяки візуальному редактору користувачі просто додають віджети, пов'язують їх з пінами мікроконтролера (наприклад, ESP32) і отримують готову систему керування. Blynk також підтримує

автоматизацію, таймери, сценарії, сповіщення та аналітику, що значно розширює функціональність проєктів.

Ще однією перевагою є гнучкість — платформа підтримує як хмарну інфраструктуру Blynk.Cloud, так і локальний сервер, що дозволяє обирати між централізованою та автономною моделлю. Швидкий запуск, підтримка різних контролерів (ESP32, ESP8266, Arduino тощо) та готові шаблони — саме це робить Blynk зручною для прототипування та реального використання.

До недоліків відноситься те, що безкоштовна версія має обмеження на кількість пристроїв, користувачів і обсяг збережених даних. Також при використанні хмарного сервера Blynk залежить від стабільного інтернет-з'єднання, що може бути великим мінусом в офлайн-сценаріях або для систем безпеки.

1.2.2 Платформа для автоматизації розумного будинку Home Assistant

Home Assistant — це потужна, відкрита платформа для автоматизації розумного будинку, яка дозволяє об'єднати управління всіма пристроями в єдиному інтерфейсі. Вона розроблена для локального управління розумними пристроями без залежності від хмарних сервісів. Home Assistant підтримує понад 2500 інтеграцій, включаючи популярні бренди, як-от Philips Hue, Sonos, MQTT, Zigbee та інші. Платформа використовується як для приватного використання в будинках, так і в малих автоматизованих системах[3].

Проєкт Home Assistant був започаткований у 2013 році розробником Паулем Штаубером (Paulus Schoutsen). З того часу він перетворився на одну з найпопулярніших систем розумного будинку з відкритим кодом. Завдяки активній спільноті та регулярним оновленням (нові версії виходять щомісяця), платформа постійно удосконалюється та розширює свої можливості.

Home Assistant є повністю відкритою платформою з вихідним кодом, доступним на GitHub. Спільнота розробників і користувачів бере активну участь у розвитку проєкту, створюючи нові інтеграції, шаблони автоматизації

					ІАЛЦ.467200.004 ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

та графічні компоненти. Це робить Home Assistant гнучким, розширюваним і стабільним рішенням для будь-якого сценарію автоматизації.

Home Assistant є потужною платформою для створення інтелектуальних систем автоматизації у будинку. Вона підтримує керування широким спектром пристроїв через популярні протоколи, такі як Zigbee, Z-Wave і MQTT. Zigbee інтегрується за допомогою модуля ZHA, що дозволяє підключати лампи, сенсори, розетки та інші пристрої без хмарних сервісів. Для Z-Wave використовується сучасна інтеграція Z-Wave JS, яка забезпечує гнучкість і стабільність. Протокол MQTT дозволяє інтегрувати пристрої з інших систем і навіть створювати власні IoT-рішення.

Щодо автоматизації, Home Assistant пропонує розвинену систему сценаріїв. Користувач може створювати автоматизації на основі тригерів (подій), умов і дій. Наприклад, можна налаштувати ввімкнення світла при русі в кімнаті після заходу сонця або вимикання розеток у нічний час. Автоматизації задаються через YAML-файли або вбудований графічний інтерфейс, що робить систему доступною для користувачів з різним рівнем технічних знань.

Крім того, Home Assistant підтримує голосове керування через інтеграції з Google Assistant та Amazon Alexa, а також має власного голосового помічника Assist, який працює локально, без хмари.

Однією з ключових переваг Home Assistant є його здатність до інтеграції з великою кількістю сторонніх сервісів і платформ, що забезпечує гнучке налаштування системи автоматизації під будь-які потреби користувача. Платформа підтримує інтеграцію з такими популярними сервісами, як IFTTT, MySensors і Node-RED.

IFTTT (If This Then That) дозволяє зв'язати Home Assistant з інтернет-сервісами, які не мають прямої інтеграції, наприклад, отримувати сповіщення або реагувати на події у соціальних мережах. MySensors — це платформа з відкритим кодом для створення власних бездротових сенсорів і пристроїв, яку можна інтегрувати в Home Assistant для збору даних або управління

					ІАЛЦ.467200.004 ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

пристроями. Node-RED забезпечує графічне середовище для побудови логіки автоматизацій через "потoki", що дозволяє легко створювати складні сценарії без програмування.

Home Assistant також підтримує підключення як до локальних, так і хмарних пристроїв. Це дозволяє уникнути залежності від зовнішніх серверів при використанні локального контролю (наприклад, через MQTT або Zigbee), але за потреби дає змогу інтегрувати сервіси як Alexa чи Google Assistant через хмарні API.

Home Assistant приділяє велику увагу безпеці та захисту конфіденційності користувачів. Однією з головних особливостей системи є локальне зберігання даних, що дозволяє уникнути надмірної залежності від хмарних сервісів. Усі автоматизації, журнали подій, налаштування та інформація про пристрої зберігаються безпосередньо на пристрої користувача (наприклад, Raspberry Pi або сервері), що зменшує ризики витоку даних.

Для забезпечення захищеного доступу до інтерфейсу Home Assistant використовується протокол HTTPS, який дозволяє шифрувати передані дані. Сертифікати SSL можна налаштувати вручну або за допомогою вбудованої інтеграції Let's Encrypt. Крім того, система підтримує авторизацію користувачів, даючи змогу створювати кілька облікових записів з різними рівнями доступу. Адміністратор може обмежити доступ до налаштувань або функцій для окремих користувачів, що особливо важливо для спільного використання в сім'ї або офісі.

Home Assistant також надає засоби для моніторингу безпеки, зокрема журнал входів та активності користувачів. Це дозволяє швидко виявляти підозрілу активність та вживати заходів.

Home Assistant — одна з найпотужніших платформ для створення розумного дому з відкритим кодом. Вона має низку важливих переваг. По-перше, це локальне управління, що забезпечує високу швидкість реакції та захист конфіденційності. Дані зберігаються безпосередньо на пристрої користувача, без участі сторонніх серверів. По-друге, Home Assistant

					ІАЛЦ.467200.004 ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

підтримує широкий спектр інтеграцій, включаючи як популярні сервіси (Google Assistant, Alexa, IFTTT), так і протоколи зв'язку (Zigbee, Z-Wave, MQTT). Це дозволяє створити комплексну автоматизовану систему для будь-яких потреб.

Ще однією сильною стороною є гнучкість і розширюваність. Користувачі можуть створювати складні автоматизації за допомогою YAML або використовувати Node-RED для візуального програмування. Активна спільнота постійно додає нові інтеграції та шаблони автоматизацій.

Однак платформа має й недоліки, особливо для новачків. Початкове налаштування може бути складним, особливо при роботі з YAML або нестандартними пристроями. Деякі інтеграції потребують технічних знань або мережевої конфігурації.

1.2.3 Серія розумних пристроїв Shelly

Shelly — це серія розумних пристроїв для автоматизації будинку, розроблена болгарською компанією Allterco Robotics. Ці пристрої призначені для дистанційного керування електроприладами, моніторингу енергоспоживання, а також для створення систем автоматизації без необхідності у складній інфраструктурі або дорогих хабах. Пристрої Shelly підходять як для домашнього, так і для комерційного використання, дозволяючи керувати світлом, опаленням, розетками, воротами тощо[4].

Однією з головних переваг Shelly є компактність — більшість пристроїв мають мініатюрний форм-фактор, що дозволяє встановлювати їх безпосередньо за настінними вимикачами або в розподільчих коробках. Це робить систему зручною для модернізації вже існуючої електромережі без потреби у капітальному ремонті.

Shelly працює через WiFi, що дає змогу підключати пристрої безпосередньо до домашньої мережі без додаткових шлюзів або хабів. Вони підтримують як локальне керування, так і хмарний доступ через офіційний

					ІАЛЦ.467200.004 ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

додаток або вебінтерфейс. Завдяки відкритим API (HTTP, MQTT, CoAP), Shelly легко інтегрується з популярними платформами, такими як Home Assistant, Node-RED, OpenHAB, Domoticz тощо.

Ще однією важливою особливістю є підтримка локального керування без хмари. Користувачі можуть взаємодіяти з пристроями напряму, використовуючи локальну IP-мережу, що підвищує рівень приватності та швидкість реакції. Це особливо цінується у професійному середовищі та серед користувачів, які надають перевагу автономним рішенням.

Система Shelly побудована за принципом модульної архітектури, що дозволяє комбінувати різні типи пристроїв відповідно до потреб користувача. Пристрої Shelly представлені в широкому асортименті:

- Реле (наприклад, Shelly 1, 1PM, 2.5, Plus 1, Plus 2PM) дозволяють дистанційно вмикати та вимикати живлення на електроприладах.
- Димери (Shelly Dimmer 2) — для плавного регулювання яскравості освітлення.
- Сенсори — руху, температури, вологості, відкривання дверей тощо.
- Енергометри (Shelly EM, 3EM) — призначені для моніторингу електроспоживання в реальному часі.
- Розумні розетки та модулі керування роletами (Shelly Plug, Shelly Plus Shutter) також входять до екосистеми.

З технічної точки зору, пристрої працюють через WiFi — це ключова особливість платформи, адже не потребується окремий хаб або шлюз. Комунікація з пристроями можлива через HTTP REST API, MQTT, CoAP або WebSocket, що відкриває широкі можливості інтеграції та автоматизації. Для просунутих користувачів є доступ до повної технічної документації API включаючи приклади команд, параметри конфігурації, структури відповіді тощо.

					ІАЛЦ.467200.004 ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

Shelly добре інтегрується з такими популярними системами автоматизації:

- Home Assistant має офіційну інтеграцію для Shelly з підтримкою автоматичного виявлення.
- OpenHAB та Domoticz підтримують Shelly завдяки відкритим протоколам.
- SmartThings також підтримує інтеграцію через сторонні компоненти або MQTT-шлюз.

Система Shelly пропонує широкий набір функціональних можливостей для управління пристроями розумного будинку, як через хмару, так і локально. Основна перевага — це гнучкість у використанні та високий рівень контролю над пристроями.

Керування здійснюється через офіційний мобільний додаток Shelly Cloud, доступний для iOS та Android. У додатку можна додавати пристрої, створювати групи, контролювати стан обладнання в реальному часі, а також задавати розклади та сценарії автоматизації. Інтерфейс інтуїтивно зрозумілий і дозволяє налаштувати більшість функцій без потреби в технічних знаннях.

Shelly підтримує автоматизацію та таймери. Користувачі можуть створювати розклади ввімкнення/вимкнення реле або димерів, задати умови спрацьовування на основі часу, температури, руху або подій інших пристроїв. Це дає змогу, наприклад, автоматично вмикати освітлення ввечері або керувати опаленням за розкладом.

Окремі моделі (Shelly 1PM, 2PM, EM, 3EM тощо) оснащені функцією моніторингу енергоспоживання. Завдяки цьому користувачі можуть бачити точні дані про споживання електроенергії для кожного пристрою або лінії, що дозволяє виявляти неефективне використання ресурсів.

Особливо важливою перевагою є можливість роботи без хмари, що реалізується через локальний API. Усі пристрої Shelly мають вбудований вебсервер і відкритий REST API, що дозволяє керувати ними напряму через локальну мережу. Також доступна підтримка MQTT, CoAP і WebSocket, що

					ІАЛЦ.467200.004 ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

полегшує інтеграцію з іншими системами, зберігаючи при цьому конфіденційність даних і високу швидкість відгуку.\

Система Shelly має ряд помітних переваг, що зробили її популярною серед користувачів, які хочуть створити розумний дім без надмірної складності. Перш за все, Shelly вирізняється доступністю — пристрої мають конкурентну ціну, а завдяки підтримці WiFi не вимагають придбання додаткових хабів або шлюзів. Це робить старт з Shelly особливо привабливим для новачків або для поетапної модернізації традиційного житла.

Ще однією сильною стороною є простота встановлення. Більшість пристроїв мають компактні розміри та легко монтуються в стандартні розподільчі коробки, за вимикачами або розетками. При цьому система залишається майже непомітною зовні. Крім того, мобільний застосунок і вебінтерфейс забезпечують інтуїтивно зрозуміле налаштування без складного програмування.

З технічної точки зору, Shelly використовує відкриті протоколи та API — REST, MQTT, CoAP, WebSocket — що дозволяє гнучко інтегрувати пристрої в інші системи, зокрема Home Assistant, OpenHAB, Node-RED. Це дає змогу побудувати як просту домашню автоматизацію, так і масштабовані комерційні рішення.

Водночас, у Shelly є деякі обмеження. Найважливіше з них — потреба в базових знаннях електромонтажу. Монтаж реле чи димера потребує втручання в електропроводку, що вимагає обережності або допомоги кваліфікованого фахівця. Також для налаштування локального доступу, інтеграцій через MQTT чи API, корисно мати мінімальні знання мережевих технологій.

У підсумку, Shelly — це потужне рішення для створення гнучкої та безпечної системи автоматизації, проте воно вимагає відповідального підходу до монтажу та налаштування.

1.3 Обґрунтування теми

Сучасні системи автоматизації та контролю часто є або надто дорогими, або залежними від хмарних сервісів, або вимагають глибоких знань у

					ІАЛЦ.467200.004 ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

конфігуруванні спеціалізованих платформ. Саме тому головною ідеєю створення даного пристрою було поєднання автономності, гнучкості, доступності та простоти розгортання. Також було прийнято рішення виводити дані через вебсторінку, оскільки це значно полегшує встановлення системи і дає можливість мати доступ з будь якої платформи.

					ІАЛЦ.467200.004 ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

2. АНАЛІЗ ОБРАНИХ ТЕХНОЛОГІЙ

2.1 Мікроконтроллер ESP32

ESP32 — це серія високопродуктивних, енергоефективних мікроконтролерів, розроблених компанією Espressif Systems, яка базується в Шанхаї (Китай). Компанія спеціалізується на створенні рішень для бездротової передачі даних та Інтернету речей. Платформа ESP32 є наступником популярного ESP8266 і отримала значні покращення в потужності, функціональності та гнучкості.

ESP32 належить до класу SoC (System-on-Chip), тобто об'єднує в одному корпусі центральний процесор, пам'ять, модулі бездротового зв'язку та периферійні інтерфейси. Чіп містить двоядерний 32-бітний процесор Tensilica LX6 або LX7, що працює на частоті до 240 МГц, а також має вбудовані модулі WiFi та Bluetooth (у тому числі BLE). Завдяки великій кількості GPIO, а також підтримці SPI, I2C, UART, ADC, DAC, PWM, пристрій ідеально підходить для взаємодії з широким спектром сенсорів та виконавчих пристроїв[5].

Основні сфери застосування ESP32 включають: розумний дім, моніторинг середовища, носимі пристрої, робототехніку, автоматизацію промислових процесів, бездротову передачу аудіо, а також освітні та дослідницькі проекти. Завдяки відкритим бібліотекам і підтримці Arduino, ESP-IDF та PlatformIO, ESP32 став стандартом для розробки IoT-пристроїв.

ESP32 оснащений двоядерним 32-бітним процесором Tensilica Xtensa LX6 (в окремих моделях — LX7 або RISC-V), що працює на частоті до 240 МГц. Завдяки цьому мікроконтролер може обробляти паралельні задачі — наприклад, керування периферією та обробку даних одночасно. Також існує можливість використання лише одного ядра для зниження енергоспоживання. Обчислювальна потужність сягає понад 600 DMIPS, що робить ESP32 придатним навіть для обробки аудіо або графіки.

Стандартні модифікації ESP32 мають 520 КБ SRAM (оперативна пам'ять) і 4 МБ Flash-пам'яті (у деяких модулях до 16 МБ). SRAM

					ІАЛЦ.467200.004 ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

використовується для змінних і обробки даних у реальному часі, тоді як Flash зберігає прошивку, вебсторінки та конфігурації. Також можливе підключення зовнішньої пам'яті PSRAM (до 8 МБ), що розширює функціональність у складніших проєктах.

ESP32 підтримує WiFi 802.11 b/g/n (2.4 ГГц) з режимами STA/AP, а також Bluetooth 4.2 + BLE — це дає змогу використовувати пристрій як точку доступу або підключати його до мережі.

Також доступні численні апаратні інтерфейси:

- до 34 GPIO-пінів (залежно від модуля),
- SPI, I2C, I2S — для роботи з дисплеями, аудіо, сенсорами,
- UART — для послідовного зв'язку (у т.ч. з GPS, GSM),
- ADC (до 18 каналів), DAC (2 канали) — для аналогових вимірювань,
- PWM, Touch-сенсори, Hall Sensor, LED Control.

Мікроконтролер ESP32 вирізняється серед аналогів великою кількістю переваг, що робить його надзвичайно популярним для розробки вбудованих систем та пристроїв Інтернету речей (IoT).

Одна з головних переваг ESP32 — вбудовані модулі WiFi 802.11 b/g/n та Bluetooth 4.2/BLE. Це дозволяє будувати пристрої без додаткових модулів зв'язку, зменшуючи загальні габарити, споживання енергії та вартість. Підтримка режимів точки доступу (AP) і станції (STA) дає змогу підключатися до мереж або створювати власні.

Завдяки двоядерному процесору Tensilica LX6 на частоті до 240 МГц та продуктивності до 600 DMIPS, ESP32 здатний виконувати одночасно обробку даних із сенсорів, обчислення, керування пристроями та мережеву взаємодію. Це робить його придатним навіть для обробки сигналів, керування дисплеями або роботи з аудіо.

Чип підтримує декілька енергозберігаючих режимів, включаючи глибокий сон (deep sleep) з живленням у кілька мікроамперів. Це дозволяє

					ІАЛЦ.467200.004 ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

використовувати ESP32 у пристроях, які працюють від батареї, наприклад — трекери, метеостанції, бездротові датчики.

ESP32 підтримується великою глобальною спільнотою розробників, що створює численні приклади, бібліотеки, форуми, гайди. Офіційна документація від Espressif є вичерпною, регулярно оновлюється та охоплює всі функції чипа.

ESP32 можна програмувати за допомогою Arduino IDE, PlatformIO, або ESP-IDF — офіційного фреймворку від Espressif. Це дозволяє як початківцям, так і досвідченим розробникам працювати з мікроконтролером на зручному для них рівні.

Таким чином, ESP32 поєднує у собі універсальність, продуктивність і доступність — що робить його ідеальним вибором для сучасних IoT-проектів.

2.2 Мікрофреймворк Flask

Flask — це легкий мікрофреймворк для створення вебзастосунків на мові програмування Python. Він був розроблений Арміном Ронахером і вперше випущений у 2010 році як частина проекту Pallets. Основною метою Flask є надання гнучкої та мінімалістичної основи для розробки вебпрограм, без надмірного нав'язування структури чи інструментів. Це дозволяє розробникам самостійно обирати компоненти для роботи з базами даних, автентифікацією, формами тощо[6].

Фреймворк базується на двох ключових бібліотеках: Werkzeug (для WSGI-сумісного вебсервера) та Jinja2 (для шаблонізації HTML). Його основні особливості включають підтримку маршрутизації URL, роботу з HTTP-запитами та відповідями, шаблони Jinja2, а також можливість розширення за допомогою сторонніх бібліотек, таких як Flask-Login, Flask-WTF, Flask-SQLAlchemy[7].

Flask — це “мікрофреймворк”, тобто він не включає за замовчуванням функцій для роботи з базами даних, формами або автентифікацією, але дозволяє легко додати потрібні модулі. Саме тому він ідеально підходить для

					ІАЛЦ.467200.004 ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

невеликих проєктів або швидкого прототипування, а також для досвідчених розробників, які хочуть повністю контролювати архітектуру застосунку.

У порівнянні з іншими фреймворками, як-от Django, Flask надає більшу гнучкість, але вимагає більше рішень від самого розробника. Django — це “батареями укомплектований” фреймворк, що включає ORM, адмін-панель, аутентифікацію, тоді як Flask пропонує базовий каркас із можливістю вільного вибору технологій.

Завдяки своїй простоті, гнучкості та активній спільноті, Flask є одним з найпопулярніших вебфреймворків у Python-екосистемі.

Flask має мінімалістичну архітектуру, яка робить його надзвичайно легким у використанні, але водночас гнучким та розширюваним. Основна ідея Flask — надати простий і чистий каркас для створення вебзастосунків, без надмірної кількості вбудованих компонентів. Завдяки цьому розробник може сам обирати інструменти і бібліотеки, які найкраще відповідають вимогам проєкту.

Однією з головних складових є маршрутизація (routing) — механізм, який дозволяє визначати URL-адреси та відповідні до них функції. Наприклад, можна легко задати маршрут /home, який обробляється функцією Python, що повертає HTML-сторінку або JSON-відповідь.

Для створення HTML-інтерфейсів Flask використовує потужну систему шаблонів Jinja2, яка дозволяє створювати динамічні вебсторінки, розділяючи логіку та подання. В шаблонах можна використовувати умовні вирази, цикли, наслідування шаблонів тощо, що робить код чистим та зручним для підтримки.

Flask не має вбудованої підтримки баз даних, але його легко інтегрувати з SQLAlchemy — потужним ORM для Python, який дозволяє працювати з базами даних як з об'єктами. Також через розширення Flask-WTF можна реалізувати роботу з вебформами.

Ще одна сильна сторона Flask — простота створення REST API. Завдяки підтримці JSON, методів GET/POST/PUT/DELETE та гнучкій обробці запитів,

					ІАЛЦ.467200.004 ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Flask часто використовується для створення бекендів мобільних застосунків або SPA (Single Page Application). Простий декоратор `@app.route()` дозволяє визначити API-ендпоінти з мінімальним кодом.

Завдяки цій гнучкості, Flask підходить як для маленьких скриптів, так і для масштабованих сервісів, особливо коли потрібна повна свобода в архітектурному рішенні.

Вибір Flask як фреймворку для розробки проєкту базується на низці переваг, які роблять його ідеальним рішенням для швидкої, гнучкої та ефективної реалізації вебзастосунків. Однією з ключових причин є гнучкість і контроль над структурою програми. Flask не нав'язує жорстких правил або шаблонів: розробник сам обирає архітектурні підходи, бібліотеки для баз даних, аутентифікацію та інші компоненти. Це особливо цінно для тих, хто прагне максимальної кастомізації або хоче створити унікальний функціонал.

Іншою важливою перевагою є простота розгортання. Flask легко інтегрується з сервісами, такими як Heroku, Docker, або локальні сервери. Завдяки своїй легкій вазі, застосунки на Flask швидко завантажуються і не потребують складної конфігурації. Крім того, Flask має хорошу сумісність з популярними сервісами і API, що дозволяє без зусиль підключати сторонні системи — бази даних, хмарні сервіси, сторонні API або інтерфейси IoT.

Flask також підходить для початківців, оскільки має інтуїтивно зрозумілий синтаксис, логічну структуру, а його документація містить багато прикладів. Це значно спрощує навчання та дозволяє швидко зануритися в практичну розробку.

Ще один вагомий аргумент — швидке прототипування. Завдяки мінімалістичному підходу, Flask дозволяє розробляти MVP (Minimum Viable Product) в стислі терміни, що ідеально підходить для стартапів або тестування нових ідей.

Загалом, Flask — це легкий, але потужний інструмент, який поєднує в собі простоту, масштабованість і контроль, що й визначило його вибір для реалізації проєкту.

					ІАЛЦ.467200.004 ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

Flask є мікрофреймворком, але його справжня сила розкривається завдяки розширенням, які дозволяють швидко додати до застосунку необхідний функціонал без складних налаштувань. Одним із найпопулярніших розширень є Flask-Login, яке забезпечує систему аутентифікації користувачів, дозволяє керувати сесіями, доступом до сторінок і реалізовувати захист маршруту.

Для роботи з вебформами часто використовується Flask-WTF, який базується на бібліотеці WTForms. Це розширення дозволяє легко створювати HTML-форми, здійснювати валідацію полів та обробку введених даних. Ще одним важливим інструментом є Flask-SQLAlchemy, що забезпечує інтеграцію з реляційними базами даних через ORM. Це дозволяє працювати з таблицями як з Python-класами, зберігаючи при цьому читабельність та зручність коду.

Flask також чудово підходить для створення REST API. Він дозволяє легко налаштовувати маршрути, методи запитів (GET, POST, PUT, DELETE) і повертати JSON-відповіді. Таким чином, Flask може виступати як backend для фронтенд-застосунків, створених на React, Vue або Angular, або бути частиною більшого мікросервісного середовища.

Крім того, Flask легко інтегрується з іншими сторонніми сервісами та бібліотеками, такими як Celery (для фонових задач), Flask-Mail (для відправки email), Flask-Migrate (для міграцій бази даних) тощо. Завдяки простій структурі проєкту, додавання нових розширень та інтеграцій не викликає труднощів.

У результаті, Flask поєднує легкість старту з потужністю функціоналу — що робить його універсальним інструментом як для початківців, так і для професійних розробників.

2.3 Онлайн-платформа для симуляції електронних проєктів Wokwi

Wokwi — це сучасна онлайн-платформа для симуляції електронних проєктів, орієнтована на мікроконтролери, такі як Arduino UNO, ESP32 та

					ІАЛЦ.467200.004 ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

Raspberry Pi Pico. Сервіс надає інтерактивне середовище, у якому користувачі можуть створювати схеми, писати програмний код та запускати симуляцію у браузері. Основною метою Wokwi є забезпечення можливості тестування та навчання без потреби у фізичному обладнанні, що робить платформу надзвичайно зручною як для початківців, так і для викладачів та досвідчених інженерів[8].

Серед особливостей — графічний редактор схем, вбудований симулятор та текстовий редактор коду. Підтримується мова програмування C/C++ із синтаксисом, сумісним з Arduino IDE. Користувачі можуть швидко переключатися між схемою і кодом, запускати проекти в реальному часі, а також ділитися ними за допомогою URL.

Wokwi постійно розвивається та підтримує все більше компонентів: датчики температури, LCD-дисплеї, сервоприводи, модулі WiFi тощо. Також підтримується підключення зовнішніх бібліотек Arduino через симульовану систему завантаження. Завдяки зручному вебінтерфейсу, Wokwi не потребує встановлення додаткового ПЗ, що значно спрощує доступ.

Таким чином, Wokwi — це не лише симулятор, а повноцінне середовище для створення, тестування та презентації вбудованих систем, яке поєднує функціональність, зручність та швидкість розробки.

Однією з ключових переваг Wokwi є те, що він повністю онлайн, що дозволяє працювати з будь-якого пристрою, не встановлюючи додаткового програмного забезпечення. Це ідеальне рішення для освітніх закладів, студентів, розробників та ентузіастів, які хочуть швидко перевірити ідею або навчитись основам мікроконтролерного програмування.

Wokwi підтримує широкий спектр пристроїв: від класичних Arduino UNO до ESP32 і Raspberry Pi Pico. Усі ці мікроконтролери можна інтерактивно підключати до компонентів, як-от світлодіоди, кнопки, дисплеї, сервомотори, сенсори температури тощо. Також можна візуалізувати з'єднання між пін-контактами, що значно знижує вірогідність помилок.

					ІАЛЦ.467200.004 ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдяки підтримці реального коду Arduino, Wokwi дозволяє написати програму на звичному синтаксисі, імпортувати бібліотеки та перевірити її роботу без фізичного контролера. Крім того, є можливість підключення до Visual Studio Code для професійнішої розробки, а також спільного редагування проєктів через URL, що корисно для командної роботи чи демонстрацій.

Wokwi також має багатий набір прикладів, які можна легко модифікувати. Це прискорює навчання та дозволяє будувати проєкти на основі вже готових шаблонів.

Завдяки вищезгаданим перевагам, платформа стає потужним інструментом для прототипування, розробки та навчання в галузі вбудованих систем.

Wokwi надає багатий функціонал, який дозволяє реалізовувати складні проєкти в інтерактивному середовищі. Основна можливість — створення схем шляхом перетягування компонентів на поле, їх логічне з'єднання та миттєва симуляція. Це дозволяє бачити результат без фізичного обладнання.

Користувачі можуть писати програмний код на C/C++, сумісний із середовищем Arduino IDE. Після написання коду можна натиснути «Play» — і схема оживе. Це чудово імітує процес розробки у фізичному середовищі, особливо для тестування логіки, циклів, обробки сигналів, роботи з сенсорами та екраном.

Платформа підтримує популярні бібліотеки Arduino, які завантажуються автоматично у віртуальне середовище. Також реалізовано імітацію затримок, PWM, серійного зв'язку, що дозволяє імітувати поведінку більшості прикладних задач.

Ще одна сильна сторона Wokwi — можливість інтеграції з фронтендом. Наприклад, користувач може використовувати Wokwi для тестування прошивки ESP32, яка потім взаємодіє з вебінтерфейсом або MQTT-брокером.

Крім того, є Wokwi для Teams, що підтримує спільну роботу над проєктами, а також функції коментування й відстеження змін, що корисно у навчанні чи професійному середовищі.

					ІАЛЦ.467200.004 ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

Загалом, можливості Wokwi роблять його потужним інструментом не лише для демонстрацій, а й для реального проектування систем.

2.4 Інтегроване середовище для розробки PlatformIO

PlatformIO — це сучасне, відкритого коду інтегроване середовище для розробки (IDE) та система збірки, призначена спеціально для створення програмного забезпечення для вбудованих систем. Вона підтримує кросплатформенну розробку, що дає змогу програмістам працювати з різними мікроконтролерами, платформами та середовищами без необхідності змінювати інструментарій або конфігурацію проєкту.

Основна мета PlatformIO — забезпечити зручний, ефективний та універсальний інструмент для розробки вбудованого ПЗ. На відміну від традиційних середовищ, які орієнтовані лише на одну платформу (наприклад, Arduino IDE), PlatformIO підтримує велику кількість платформ, включаючи популярні мікроконтролери: Arduino, ESP32, ESP8266, STM32, Atmel AVR, Nordic NRF52, TI MSP430 та десятки інших. Це дозволяє інженерам легко перемикатися між різними апаратними платформами без суттєвих змін у структурі проєкту[9].

PlatformIO працює на базі Python і включає в себе потужний CLI (command-line interface), а також має інтеграцію з популярними середовищами розробки, зокрема Visual Studio Code. Завдяки цьому розробники можуть використовувати такі функції, як автодоповнення, налагодження, інтеграція з системами контролю версій (наприклад, Git), а також керування бібліотеками.

Крім того, середовище підтримує unit-тестування, емуляцію, хмарну компіляцію, автоматизацію CI/CD та моніторинг портів. Це робить PlatformIO не просто альтернативою Arduino IDE, а повноцінним професійним інструментом, придатним як для хобістів, так і для промислових проєктів.

PlatformIO побудовано як модульну систему, яка складається з кількох ключових компонентів. Основу архітектури становить ядро PlatformIO CLI

					ІАЛЦ.467200.004 ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

(Command Line Interface) — потужний інструмент, який забезпечує основний функціонал: компіляцію проєктів, завантаження прошивки, моніторинг послідовного порту, тестування, керування платформами та бібліотеками. CLI дозволяє розробникам автоматизувати робочі процеси та інтегрувати PlatformIO у CI/CD середовища.

Другим важливим елементом є інтеграція з IDE, особливо з Visual Studio Code — найпопулярнішою інтегрованою середою, рекомендованою командою PlatformIO. Завдяки розширенню PlatformIO IDE для VS Code розробники отримують повноцінне середовище з підтримкою автодоповнення коду, інтерактивного відлагодження, перегляду структур проєкту, швидкого запуску завдань і зручної роботи з портами. Також існує підтримка для інших середовищ, зокрема Atom, Sublime Text і CLion.

Третім ключовим компонентом є конфігураційний файл `platformio.ini`, який використовується для опису параметрів проєкту. У ньому задаються платформи (наприклад, `platform = espressif32`), плати (`board = esp32dev`), середовище (`framework = arduino, mbed, zephyr` тощо), параметри компілятора, бібліотеки, завантаження прошивки та інші налаштування. Цей підхід забезпечує високу гнучкість та контроль над проєктом.

Крім цього, PlatformIO має власний менеджер бібліотек, який дозволяє легко підключати, оновлювати та видаляти сторонні або власні бібліотеки з репозиторіїв. Також платформа підтримує систему платформ — завантажуваних пакетів з описами для конкретних мікроконтролерів, компіляторів та налагоджувачів.

Завдяки такій архітектурі PlatformIO забезпечує масштабованість, гнучкість і професійний підхід до розробки прошивок для вбудованих систем.

PlatformIO — це не просто середовище розробки, а повноцінна екосистема для створення, тестування та розгортання програмного забезпечення для вбудованих систем. Однією з головних можливостей є компіляція та завантаження прошивки на широкий спектр мікроконтролерів. Платформа автоматично підбирає компілятор та інструменти відповідно до

					ІАЛЦ.467200.004 ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

вказаної плати, що значно спрощує конфігурацію. Завантаження прошивки можливе через USB, OTA (over-the-air) або інші інтерфейси, що залежить від обраної платформи.

Ще одна важлива функція — моніторинг послідовного порту. Завдяки вбудованому серійному монітору можна отримувати дані від мікроконтролера в реальному часі без необхідності сторонніх програм. Це зручно для налагодження й перевірки роботи прошивки.

PlatformIO також включає вбудований менеджер бібліотек, який дозволяє швидко знаходити, встановлювати, оновлювати або видаляти бібліотеки з центрального репозиторію. Бібліотеки можна встановлювати як глобально, так і для конкретного проєкту. Це значно спрощує керування залежностями.

Особливої уваги заслуговує підтримка unit-тестування, що є рідкістю серед середовищ для вбудованих систем. Розробники можуть створювати модульні тести для перевірки окремих функцій або модулів і запускати їх як на апаратному рівні, так і в емуляторі.

PlatformIO також підтримує CI/CD-пайплайни, зокрема інтеграцію з GitHub Actions, Travis CI, GitLab CI та іншими сервісами. Це дозволяє автоматизувати процеси тестування та збирання прошивки під час командної розробки.

На додачу, платформа має підтримку емуляції (через зовнішні інструменти або середовища, такі як QEMU), що дозволяє проводити тестування без фізичного обладнання. Усе це робить PlatformIO потужним інструментом як для хобістів, так і для інженерів, що працюють над комерційними проєктами.

2.5 Мова розмітки HTML

HTML (HyperText Markup Language) — це стандартна мова розмітки, яка використовується для створення та структурування вебсторінок. HTML не є

					ІАЛЦ.467200.004 ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

мовою програмування — це мова розмітки, яка описує структуру вмісту документа за допомогою елементів, що позначають заголовки, абзаци, списки, посилання, зображення, відео тощо. Основна функція HTML — визначити логічну структуру вебсторінки, яку браузер зможе інтерпретувати й відобразити користувачеві[10].

HTML виконує ключову роль у побудові Інтернету. У поєднанні з CSS (каскадними таблицями стилів) для оформлення і JavaScript для інтерактивності, HTML формує основу сучасної веброзробки. HTML створює «каркас» сторінки, CSS відповідає за її вигляд, а JavaScript — за поведінку.

Стандарти HTML постійно розвиваються. Спочатку була версія HTML 4, яка широко використовувалася в 1990-х і на початку 2000-х. Вона мала обмежені можливості для роботи з мультимедіа й потребувала сторонніх технологій, таких як Flash. У відповідь на потребу в сучасному, гнучкому вебі було створено HTML5 — актуальну версію, яка додала підтримку відео, аудіо, графіки (через елемент `<canvas>`), семантичної розмітки (`<header>`, `<footer>`, `<article>`) та покращила доступність.

HTML5 також запровадив API для геолокації, локального сховища даних (localStorage), офлайн-режимів тощо. Ці зміни дозволили створювати складні вебзастосунки без використання сторонніх плагінів.

Завдяки відкритості, простоті та потужності HTML залишається основним стандартом для створення контенту в Інтернеті, від простих статичних сторінок до складних вебзастосунків.

HTML-документ побудований за чітко визначеною структурою, яка починається з декларації типу `<!DOCTYPE html>`, що вказує браузеру, що документ використовує HTML5. Уся сторінка розміщується всередині кореневого тегу `<html>`, який ділиться на дві основні частини: `<head>` і `<body>`.

У блоці `<head>` міститься метайнформація про сторінку — назва (`<title>`), кодування (`<meta charset="UTF-8">`), підключення стилів (`<link rel="stylesheet">`), скриптів (`<script>`), а також інші технічні дані. Основний вміст вебсторінки розміщується в тілі документа — у блоці `<body>`.

					ІАЛЦ.467200.004 ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

HTML використовує теги для опису структури контенту: заголовки (<h1>–<h6>), абзаци (<p>), списки (,), зображення (), посилання (<a>) тощо. Більшість тегів можуть містити атрибути, які задають додаткові властивості — наприклад, href у <a> або src у .

Важливою особливістю сучасного HTML є підтримка семантичної розмітки. Вона включає теги на кшталт <header>, <nav>, <main>, <section>, <article>, <footer>, які описують логічну структуру сторінки та покращують доступність, SEO і машинне розуміння контенту.

HTML тісно взаємодіє з CSS і JavaScript. За допомогою CSS задається вигляд елементів (кольори, шрифти, розміщення), а JavaScript додає динамічні дії та взаємодію (обробка подій, анімація, AJAX-запити).

Загалом, HTML — це основа, на якій будується вебсторінка. Структура документа забезпечує логічну організацію вмісту, а за рахунок CSS і JS сторінка стає візуально привабливою та функціональною.

HTML забезпечує широкі можливості для створення повноцінних вебінтерфейсів. Однією з ключових функцій є створення форм, які дозволяють користувачам вводити дані. Використовуючи теги <form>, <input>, <textarea>, <select> та інші, можна будувати як прості контактні форми, так і складні інтерфейси з перевіркою даних, інтеграцією з базами даних або бекендом через JavaScript.

HTML також дозволяє створювати таблиці для представлення структурованої інформації. Елементи <table>, <tr>, <td>, <th> забезпечують логічне розміщення даних у рядках і стовпцях. Хоча сучасні інтерфейси часто відмовляються від таблиць для верстки, вони залишаються актуальними для відображення статистики, звітів тощо.

Для розміщення зображень використовується тег , який дозволяє вставляти графіку різних форматів (JPEG, PNG, SVG) з підтримкою альтернативного тексту (alt). HTML також підтримує навігаційні елементи — наприклад, <nav> та <a> — які створюють меню, посилання на інші сторінки чи ресурси.

HTML5 додав підтримку мультимедійного контенту без використання сторонніх плагінів. Теги <audio> та <video> дозволяють вбудовувати аудіо- й відеофайли з контролерами відтворення, субтитрами та альтернативними форматами.

Завдяки сучасним тегам (наприклад, <section>, <article>, <aside>, <figure>), HTML дозволяє створювати адаптивну верстку, яка змінює вигляд сторінки відповідно до розміру екрана. Це особливо важливо для мобільних пристроїв і забезпечується у поєднанні з CSS-медіазапитами.

Загалом, HTML — це потужна основа для побудови доступного, гнучкого і функціонального вебконтенту.

2.6 Мова стилів CSS

CSS (Cascading Style Sheets, каскадні таблиці стилів) — це мова стилів, яка використовується для опису зовнішнього вигляду вебсторінок. Вона визначає, як HTML-елементи мають відображатися у браузері. CSS дозволяє задавати кольори, шрифти, відступи, розміри, позиціонування елементів, а також створювати анімації та адаптивні макети.

Основна мета CSS — розділити структуру (HTML) та оформлення вебдокументу. Такий підхід покращує підтримку й масштабованість вебпроектів. Завдяки CSS, один і той самий HTML-код може мати різне візуальне представлення в залежності від того, який файл стилів до нього підключений. Це спрощує оновлення дизайну — достатньо змінити стилі, не торкаючись HTML-коду[11].

CSS базується на принципі каскадності — це означає, що стилі можуть успадковуватись, перевизначатись та комбінуватись залежно від специфічності селектора та порядку оголошення. Селектори CSS визначають, до яких елементів HTML застосовуються певні стилі. Наприклад, можна застосувати стилі до всіх заголовків <h1>, або тільки до конкретного елемента за ідентифікатором чи класом.

					ІАЛЦ.467200.004 ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

CSS підтримується усіма сучасними браузером та є стандартом W3C (World Wide Web Consortium). Існує декілька версій CSS — від CSS1 до CSS3 (останній активно використовується та містить модулі: Flexbox, Grid, Transitions, Media Queries тощо).

Завдяки CSS можна створювати сучасні, привабливі та адаптивні інтерфейси без необхідності використання графічних редакторів. Веброзробка неможлива без CSS, оскільки саме він відповідає за візуальний досвід користувача.

Однією з головних переваг CSS є простота у використанні. Синтаксис цієї мови інтуїтивно зрозумілий і легко запам'ятовується. Для застосування стилю достатньо вказати HTML-елемент (через селектор) і описати для нього набір властивостей, наприклад: `color`, `font-size`, `margin`, `padding` тощо. Це дозволяє розробникам швидко вивчити основи CSS і застосовувати їх навіть у невеликих проєктах без глибокого занурення у складні концепції.

Другою важливою особливістю є повна підтримка CSS усіма сучасними браузерами. Незалежно від того, який браузер використовує користувач (Google Chrome, Firefox, Safari, Edge тощо), CSS-код інтерпретується майже однаково. Це дозволяє розробляти інтерфейси, які працюють стабільно на різних пристроях і операційних системах. У разі потреби можна використовувати спеціальні префікси для сумісності зі старими версіями браузерів.

CSS також забезпечує розділення логіки і представлення. Весь вміст сторінки описується за допомогою HTML, тоді як вигляд, стилізація та розташування елементів задається окремо — у CSS. Такий підхід значно покращує структурування коду, дозволяє різним командам (наприклад, програмістам і дизайнерам) працювати паралельно, а також полегшує обслуговування й оновлення сайту.

Завдяки своїй гнучкості CSS дозволяє швидко змінювати вигляд сторінки без зміни HTML-коду. Наприклад, змінивши лише одну властивість у CSS-файлі, можна змінити колірну гаму всього сайту. Це особливо важливо

					ІАЛЦ.467200.004 ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

для адаптації сайту до різних екранів, створення нічного режиму або зміни дизайну згідно з оновленим брендом.

Однією з важливих переваг використання CSS є підвищення читабельності даних. Завдяки стилям можна чітко структурувати вміст сторінки, виділити ключові елементи за допомогою кольорів, шрифтів, відступів і рамок. Наприклад, використання більших шрифтів для заголовків, помітних кольорів для важливих повідомлень (наприклад, червоного для помилок) або контрастного фону для блоків із даними значно полегшує сприйняття інформації. Це особливо важливо для інтерфейсів, які показують динамічні або числові дані, як-от сенсори в системі моніторингу.

Друга важлива перевага — покращення користувацького досвіду (UX). CSS дозволяє створити зрозумілу, зручну і візуально приємну навігацію. Наприклад, стильоване меню вибору сенсорів, плавні переходи між станами або анімації наведення значно підвищують інтуїтивність користування. Гарний UX зменшує когнітивне навантаження на користувача, полегшує взаємодію з системою і стимулює повторне використання сервісу.

Третім важливим аспектом є естетичність інтерфейсу без використання складних бібліотек. CSS дозволяє досягати сучасного, акуратного вигляду вебсторінки лише завдяки базовим можливостям: тіням, заокругленням, градієнтам, шрифтам, макетам Flexbox чи Grid. Це дозволяє уникати надмірного навантаження на сторінку та залежності від сторонніх фреймворків, що робить інтерфейс легким, швидким і зручним для адаптації.

Таким чином, CSS — це не лише засіб стилізації, а потужний інструмент покращення функціональності, зручності та естетики вебінтерфейсу без потреби у складному технічному стеку.

					ІАЛЦ.467200.004 ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

3. СТРУКТУРА І ВЕБЗАСТОСУНОК ДЛЯ ПРИСТРОЇВ З СИСТЕМОЮ МОНІТОРІНГУ СЕРЕДОВИЩА

3.1 Структура системи моніторингу

У склад системи моніторингу параметрів входять як апаратні, так і програмні компоненти, які взаємодіють між собою через локальну мережу. Основним елементом є мікроконтролер ESP32, який виступає як вузол збору даних із підключених сенсорів. До ESP32 підключено декілька типів сенсорів: датчик температури та вологості DHT22, ультразвуковий датчик відстані HC-SR04, а також інфрачервоний датчик руху PIR.

ESP32 працює як вебсервер і через протокол HTTP надає доступ до актуальних показників у форматі JSON. На стороні комп'ютера або смартфона функціонує вебсервер Flask, реалізований на мові Python. Його завдання — надсилати запити до ESP32, отримувати відповіді у вигляді JSON-даних, обробляти їх та відображати у зручному для користувача вигляді.

Клієнтський інтерфейс — це вебсторінка, створена з використанням HTML та CSS, яка дозволяє користувачу вибрати один із доступних сенсорів і переглянути його показники. Таким чином, програмне забезпечення складається з двох основних модулів: прошивка для ESP32 (Arduino/C++) та вебдодаток на Flask (Python), які спільно забезпечують повноцінну роботу системи моніторингу.

У системі керування пристроями використовується протокол HTTP (HyperText Transfer Protocol) як основний засіб комунікації між клієнтською частиною (вебдодатком Flask) та апаратною частиною (контролером ESP32). HTTP — це текстовий протокол, який широко використовується для обміну інформацією в мережі Інтернет. Його перевагою є простота, зрозуміла структура та сумісність з великою кількістю інструментів і бібліотек.

ESP32 виступає у ролі вебсервера, який обробляє вхідні HTTP-запити від клієнта. Після отримання запиту ESP32 зчитує значення з підключених

сенсорів, формує відповідь у вигляді JSON-об'єкта (JavaScript Object Notation) і відправляє її у відповідь.

Формат JSON обраний через його компактність, легкість обробки як на стороні сервера, так і на стороні клієнта, а також підтримку в більшості мов програмування. У вебдодатку Flask цей JSON розбирається і на його основі формуються HTML-сторінки для користувача.

Такий підхід забезпечує ефективну, стандартизовану та масштабовану передачу даних. Зокрема, клієнт може запитати лише ті дані, які йому необхідні, а сервер відправляє лише актуальні показники. Крім того, формат JSON дозволяє легко виявляти помилки — наприклад, за допомогою поля "status" система визначає, чи працює відповідний сенсор.

Завдяки використанню HTTP та JSON ця система може бути інтегрована з іншими програмами, мобільними додатками чи хмарними платформами з мінімальними зусиллями.

У системі керування пристроями виділяються три основні модулі програмного забезпечення: програмна частина ESP32 (скетч Arduino), серверна частина (Flask) та клієнтська частина (HTML/CSS). Кожен з модулів виконує окрему роль, забезпечуючи взаємодію користувача з пристроями через інтернет.

3.1.1. Програмна частина ESP (скетч Arduino)

Цей модуль відповідає за роботу з сенсорами та обробку запитів HTTP. Код написаний на базі середовища Arduino з використанням бібліотек WiFi.h, WebServer.h та DHT.h. ESP32 зчитує показники з сенсорів. Дані перевіряються на наявність помилок і формуються у форматі JSON. У відповідь на HTTP-запит ESP32 надсилає ці дані клієнту.

Було обрано шаблон ESP32 HTTP IoT Server на платформі Wokwi для реалізації даної частини, оскільки він надає ідеальну основу для створення компактної системи моніторингу параметрів середовища з передачею даних

					ІАЛЦ.467200.004 ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

через локальну мережу. Цей шаблон вже містить попередньо налаштований вебсервер на ESP32, який відповідає на HTTP-запити та формує відповіді у форматі JSON, що повністю відповідає вимогам мого проєкту.

Основною перевагою є простота налаштування та швидкий старт, оскільки не потрібно додатково конфігурувати серверну частину або створювати базову структуру коду. Крім того, Wokwi забезпечує візуальне моделювання схеми з ESP32 та сенсорами, що дозволяє перевірити логіку зчитування даних з таких датчиків, ще до завантаження прошивки на реальний пристрій[12].

Шаблон також спрощує налагодження обробки помилок, оскільки вже містить типову логіку для перевірки сенсорів та підготовки відповідей. Це дозволило мені зосередитись на розширенні функціоналу, додаванні додаткових датчиків, реалізації умовної логіки та тестуванні взаємодії з Flask-сервером без зайвих витрат часу на базове програмування.

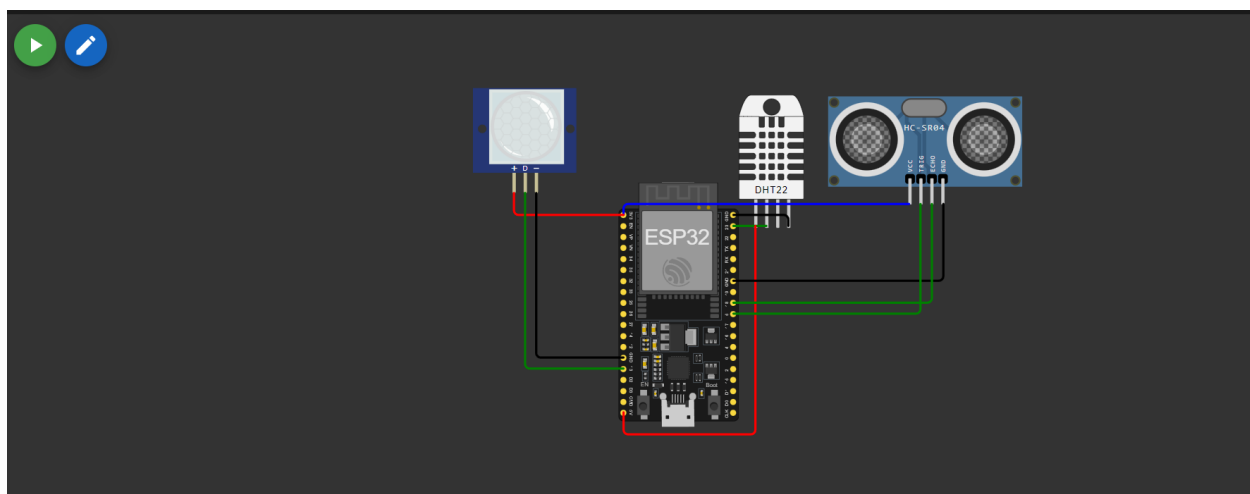


Рисунок 3.1 – Зовнішній вигляд симульованої плати

Таким чином, вибір шаблону ESP32 HTTP IoT Server забезпечив ефективну основу для розробки багатофункціонального пристрою моніторингу з підтримкою локального API.

Для макету було обрано 3 сенсора з якими можна буде показати мінімальні можливості даного проєкту. Обраними сенсорами є DHT22 — датчик температури та вологості, HC-SR04 — ультразвуковий датчик відстані, і PIR — інфрачервоний датчик руху.

					ІАЛЦ.467200.004 ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

DHT22 є одним з найпопулярніших цифрових сенсорів для вимірювання температури повітря та відносної вологості. Його робота базується на використанні цифрового сигналу, що дозволяє підключати його безпосередньо до мікроконтролера, зокрема ESP32. У даному проєкті сенсор з'єднано з ESP32 через пін GPIO 23. DHT22 має хорошу точність вимірювання: $\pm 0.5^{\circ}\text{C}$ для температури та $\pm 2\text{--}5\%$ RH для вологості, що робить його придатним для застосування в побутових, лабораторних та навчальних системах[13].

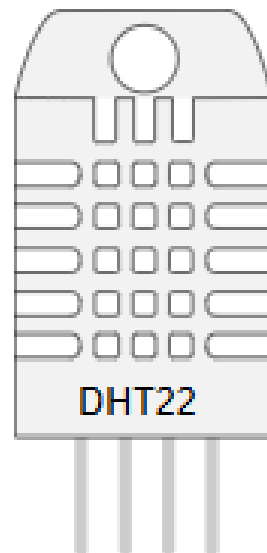


Рисунок 3.2 – Сенсор DHT22

Основною перевагою DHT22 є баланс між ціною, надійністю та простотою використання. Він підтримує широкий діапазон температур — від -40 до $+80^{\circ}\text{C}$, а також вологість у межах $0\text{--}100\%$, що дозволяє використовувати його в різноманітних умовах. Завдяки цифровому виходу не потребується складних аналогових підключень або калібрування — достатньо скористатись готовою бібліотекою.

DHT22 був обраний для цього проєкту через його просту інтеграцію з ESP32, наявність бібліотек для Arduino/PlatformIO, а також завдяки можливості отримувати два показники одночасно, що дозволяє скоротити кількість сенсорів у системі. Крім того, цей сенсор стабільно працює навіть при невеликих змінах температури, що важливо для постійного моніторингу навколишнього середовища.

HC-SR04 — це недорогий і популярний ультразвуковий сенсор, призначений для вимірювання відстані до об'єктів безконтактним способом. Він працює на основі ультразвукових хвиль, що дозволяє точно визначати відстань у межах від 2 до 400 см з точністю до кількох міліметрів. У цьому

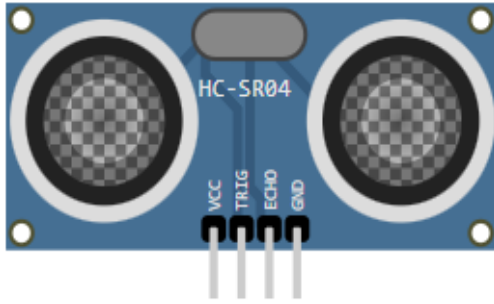


Рисунок 3.3 – Сенсор HC-SR04 Ultrasonic Distance Sensor

проекті він підключений до ESP32 через пін TRIG (GPIO 5) для ініціації сигналу і ECHO (GPIO 18) для прийому відбитого сигналу[14].

Принцип роботи полягає у вимірюванні часу затримки між передачею та отриманням ультразвукового імпульсу. Отримане

значення обробляється ESP32 для розрахунку фактичної відстані. Такий метод надає швидкі та досить точні результати, придатні для задач контролю об'єктів у

Цей сенсор було обрано через його доступність, простоту інтеграції та стабільну роботу у багатьох практичних сценаріях. Він ідеально підходить для систем моніторингу рівня заповнення, автоматичних воріт, робототехніки, а також як елемент безконтактного управління. У нашій системі він дозволяє отримувати додаткову інформацію про оточення та може бути використаний для активації сценаріїв автоматизації на основі наближення або присутності об'єктів. Простота підключення до ESP32 та наявність готових бібліотек робить HC-SR04 ідеальним вибором для проєктів з Arduino або PlatformIO.

ї PIR (Passive Infrared Sensor) — це недорогий і енергоефективний сенсор, який використовується для виявлення руху людей або тварин. У даному проєкті він підключений до GPIO 13 мікроконтролера ESP32. Робота сенсора базується на виявленні змін інфрачервоного випромінювання в зоні дії. Коли в полі зору з'являється теплий об'єкт (наприклад, людина), сенсор фіксує зміну температури фону і видає логічний сигнал "1", що інтерпретується як рух[15].

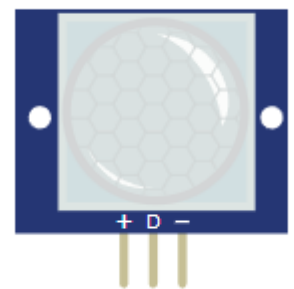


Рисунок 3.4 – Сенсор Passive Infrared (PIR) motion sensor.

Однією з основних переваг PIR-сенсора є низьке енергоспоживання, а також простота підключення — він має лише вихідний пін і не потребує

складної конфігурації. Датчик не фіксує присутність постійно — лише появу/зникнення теплового об'єкта, що робить його придатним для тригерних сценаріїв: вмикання світла, запуск інших пристроїв, надсилання сповіщення тощо.

Цей сенсор було обрано для проєкту через його ефективність у виявленні присутності, особливо в умовах, де потрібно уникати постійного моніторингу. Він ідеально доповнює систему моніторингу середовища, дозволяючи не лише отримувати фізичні показники, а й реагувати на поведінкові фактори. Завдяки стандартному цифровому виходу PIR легко інтегрується з ESP32 та працює стабільно без складного налаштування. Це робить його оптимальним вибором для розумного дому або охоронних систем.

3.1.2. Серверна частина (Flask)

Серверна частина системи реалізована за допомогою мікрофреймворку Flask, написаного на мові програмування Python. Flask виконує роль посередника між мікроконтролером ESP32 та кінцевим користувачем, забезпечуючи запити, обробку відповідей і формування інтерфейсу.

Після запуску Flask-додаток створює локальний вебсервер, який доступний через браузер. При відкритті сторінки користувачем, Flask автоматично надсилає HTTP-запит до ESP32 на визначену IP-адресу. У відповідь ESP32 повертає дані з сенсорів у форматі JSON, що включає дані сенсора.

Отримана інформація обробляється сервером: Flask перевіряє статуси сенсорів, визначає, які з них активні, і відповідно формує динамічний HTML-код, який виводить тільки доступні значення. У разі помилки з'єднання, система відображає повідомлення про неможливість отримання даних, завдяки реалізованій обробці виключень.

Таким чином, Flask забезпечує гнучкість, масштабованість і простоту інтеграції з ESP32, дає змогу розробити зручний вебінтерфейс та

					ІАЛЦ.467200.004 ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

контролювати роботу системи в реальному часі через будь-який браузер без складного налаштування або сторонніх сервісів.

3.1.3. Клієнтська частина (HTML/CSS)

Це інтерфейс, з яким взаємодіє користувач. Вона реалізована за допомогою HTML для структури сторінки та CSS для візуального оформлення. Сторінка містить меню вибору сенсора, а також блок для виводу даних. Оформлення включає естетичний дизайн з адаптивним контейнером, кольорами та шрифтами, що підвищують зручність користування.

Разом ці три модулі формують повноцінну систему моніторингу, що дозволяє користувачеві віддалено переглядати стан пристроїв у реальному часі через веббраузер.

3.2. Функціонал пристрою системи керування

Flask було обрано як серверну технологію для реалізації проєкту з моніторингу параметрів середовища на базі ESP32 завдяки його простоті, гнучкості та ефективності при розробці невеликих вебдодатків. Це легкий фреймворк на мові програмування Python, який ідеально підходить для таких проєктів, де потрібне динамічне формування HTML-контенту, обробка HTTP-запитів і взаємодія з іншими пристроями, як-от ESP32.

Однією з ключових причин вибору Flask є його мінімалістичний підхід. Він не нав'язує структуру проєкту, як це робить, наприклад, Django, і дозволяє зосередитися саме на реалізації необхідного функціоналу. Для цього проєкту потрібно було просто: отримати JSON з ESP32, обробити дані, вивести їх у зручному вигляді та обробити можливі помилки — з чим Flask справляється ідеально.

Також важливо, що Flask підтримує шаблонізатор Jinja2, що дозволяє створювати зручні динамічні HTML-сторінки з можливістю вставки змінних

					ІАЛЦ.467200.004 ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

Python без зайвих складнощів. Це значно спрощує формування адаптивного вебінтерфейсу, який залежить від стану сенсорів і зчитаних значень.

Крім того, Flask легко розгортається на будь-якому ПК або сервері, має хорошу документацію та спільноту. У межах проєкту не потрібно складної авторизації, баз даних або великої кількості маршрутів, тому використання важких фреймворків було б недоцільним. Flask дозволив реалізувати все необхідне за короткий час з високою стабільністю та хорошою продуктивністю.

Завдяки цьому Flask став ідеальним вибором для створення локального вебінтерфейсу, що працює спільно з ESP32 і дозволяє швидко протестувати, розширити й адаптувати систему моніторингу.

Після запуску, мікроконтролер ESP32 виконує ініціалізацію через послідовний порт (`Serial.begin()`) і підключається до WiFi-мережі, використовуючи бібліотеку `WiFi.h`. У скетчі вказані назва мережі (`ssid`) і пароль (у прикладі відкритий доступ). За допомогою циклу `while(WiFi.status() != WL_CONNECTED)` пристрій очікує стабільного з'єднання. Після підключення виводиться локальна IP-адреса, яка потрібна для зв'язку з сервером Flask.

На ESP32 запускається вбудований вебсервер на базі бібліотеки `WebServer.h`, який слухає порт 80. Основний маршрут / обробляється функцією `handleRoot()`, що зчитує показники з підключених сенсорів: температури, вологості, відстані та руху. Дані перевіряються на коректність, формуються у форматі JSON, і надсилаються у відповідь на HTTP-запит.

Серверна частина на Flask запускає локальний вебдодаток. При відкритті сторінки користувачем, Flask надсилає HTTP-запит до ESP32 (на його IP-адресу), використовуючи бібліотеку `requests`. У відповідь приходить JSON з даними від ESP32. Якщо пристрій недоступний або виникає інша помилка, Flask обробляє виняток і відображає відповідне повідомлення на сторінці.

Отримані JSON-дані аналізуються у Flask і передаються до HTML-шаблону через `render_template_string`. Там дані вставляються у відповідні місця в шаблоні залежно від вибраного сенсора. Наприклад, якщо обрано "температуру", відображається тільки відповідне значення. Так забезпечується інтерактивність і адаптивність відображення даних.

3.3 Інтерфейс керування пристроєм

Інтерфейс програмного забезпечення керування реалізовано як вебсторінку за допомогою технологій HTML та CSS. Основна мета інтерфейсу — забезпечити користувачу зручний доступ до інформації, отриманої від сенсорів, підключених до ESP32. Вебінтерфейс є простим, інтуїтивно зрозумілим та не вимагає спеціальних технічних навичок для користування.

На головній сторінці вебдодатку розміщено заголовок, що вказує на призначення системи — «Моніторинг ESP32». Нижче розташоване випадаюче меню (селектор), за допомогою якого користувач може обрати один з доступних датчиків: температуру, вологість, відстань або рух. Після вибору сенсора та підтвердження запиту, система звертається до мікроконтролера ESP32, отримує актуальні дані та відображає їх на сторінці.

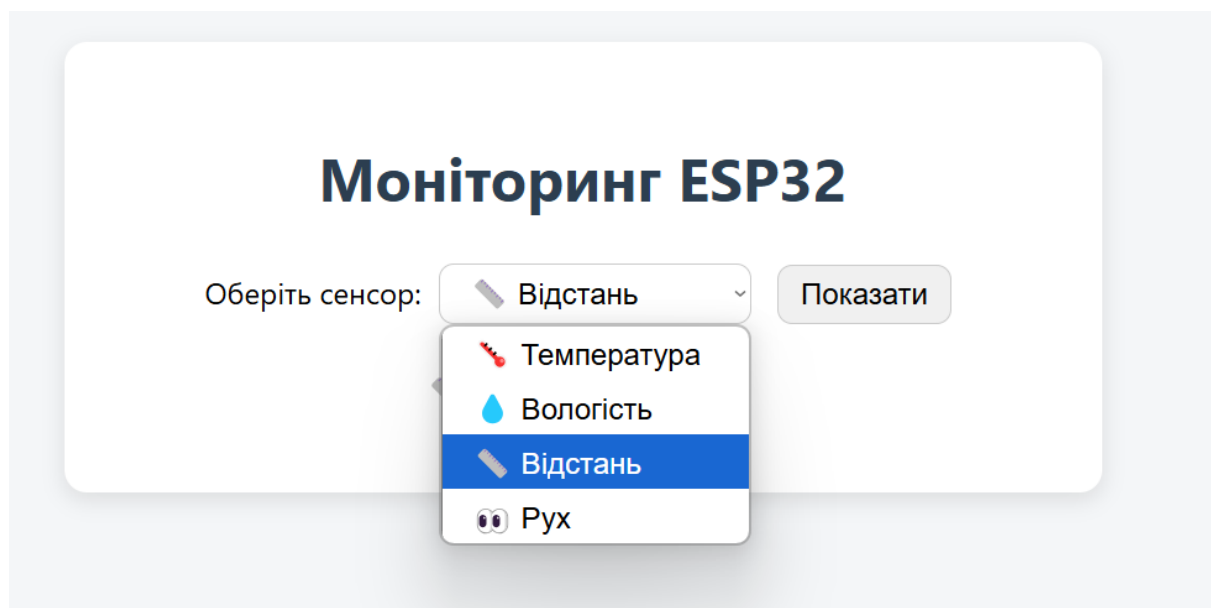


Рисунок 3.5 – Зовнішній вигляд інтерфейсу вебсайту

Відображення результатів є динамічним — залежно від обраного сенсора показується лише відповідна інформація. Наприклад, якщо обрано «температура», виводиться значення температури з відповідним піктограмою. Аналогічно працює для інших сенсорів, що зменшує навантаження на користувача.

Для покращення візуального сприйняття інформації. Оформлення включає світле тло, заокруглені контейнери з тінями, читабельні шрифти та іконки біля назв параметрів. Інформація групується у компактні блоки, що підвищує зрозумілість інтерфейсу. У разі виникнення помилки (наприклад, сенсор не підключений), виводиться повідомлення червоним кольором, що чітко сигналізує користувачу про проблему.

Таким чином, вебінтерфейс системи забезпечує ефективний та доступний спосіб моніторингу параметрів середовища, зібраних через ESP32.

Інтерфейс системи керування складається з кількох основних елементів, які забезпечують зручність взаємодії користувача з пристроєм ESP32 та його сенсорами. Завдяки правильній організації та візуальному оформленню, користувач може легко обирати необхідну інформацію та швидко реагувати на зміни у середовищі або помилки в системі.

Центральним елементом взаємодії є меню вибору сенсора — це HTML-елемент `<select>`, який дозволяє користувачу обрати, які саме дані потрібно переглянути. Меню містить варіанти: температура, вологість, відстань та рух. Це меню дає змогу динамічно фільтрувати дані на сторінці, показуючи лише значення з обраного сенсора. Це не тільки спрощує користування, а й зменшує кількість зайвої інформації на екрані.

Після вибору сенсора та отримання даних від ESP32, відповідне значення виводиться на сторінці в естетичному блоці.

Ці значення розміщуються всередині візуально виокремленого контейнера з фоном, тінями та заокругленими кутами. Такий підхід підвищує зручність сприйняття інформації.

у

	Н				ІАЛЦ.467200.004 ПЗ	Арк.
						44
Змн.	Нарк.	№ докум.	Підпис	Дата		

У разі виникнення проблем із отриманням даних — наприклад, відсутність сенсора або його фізичне відключення — інтерфейс відображає відповідне повідомлення. Це повідомлення має червоний колір і відображається чітко під заголовком або замість даних сенсора. Повідомлення базуються на статусах, які надсилає ESP32 через JSON, і обробляються у Flask-шаблоні.

Ці елементи разом створюють дружній, інформативний і надійний інтерфейс для моніторингу.

3.4 Тестування програми

Після завершення основної розробки програмної та апаратної частин системи моніторингу на базі ESP32 було проведено детальне тестування, щоб переконатися у коректності роботи всіх компонентів, стабільності передачі даних і правильному відображенні інформації у вебінтерфейсі.

Для перевірки коректності запуску системи було проведено тестування початкової ініціалізації апаратної та серверної частин. Зокрема, перевірка охоплює правильне завантаження прошивки на ESP32 та виведення діагностичних повідомлень у термінал.

У середовищі Wokwi після запуску симуляції ESP32 виконує стандартну послідовність ініціалізації, яка включає: ініціалізацію серійного порту, початок з'єднання з WiFi, очікування встановлення з'єднання.

Після успішного з'єднання з WiFi ESP32 отримує локальну IP-адресу, виводить її у серійний термінал і повідомляє що плата підключилась до WiFi й запустила вебсервер. Ці два повідомлення є ключовими індикаторами того, що: ESP32 успішно підключено до мережі, вбудований вебсервер готовий до обробки HTTP-запитів. Якщо один з етапів не відпрацьовує повідомлення не з'являться, а система не відповідатиме на запити з Flask або браузера. Таким чином, вивід у термінал використовується як основний діагностичний інструмент для підтвердження готовності пристрою до роботи.

					ІАЛЦ.467200.004 ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

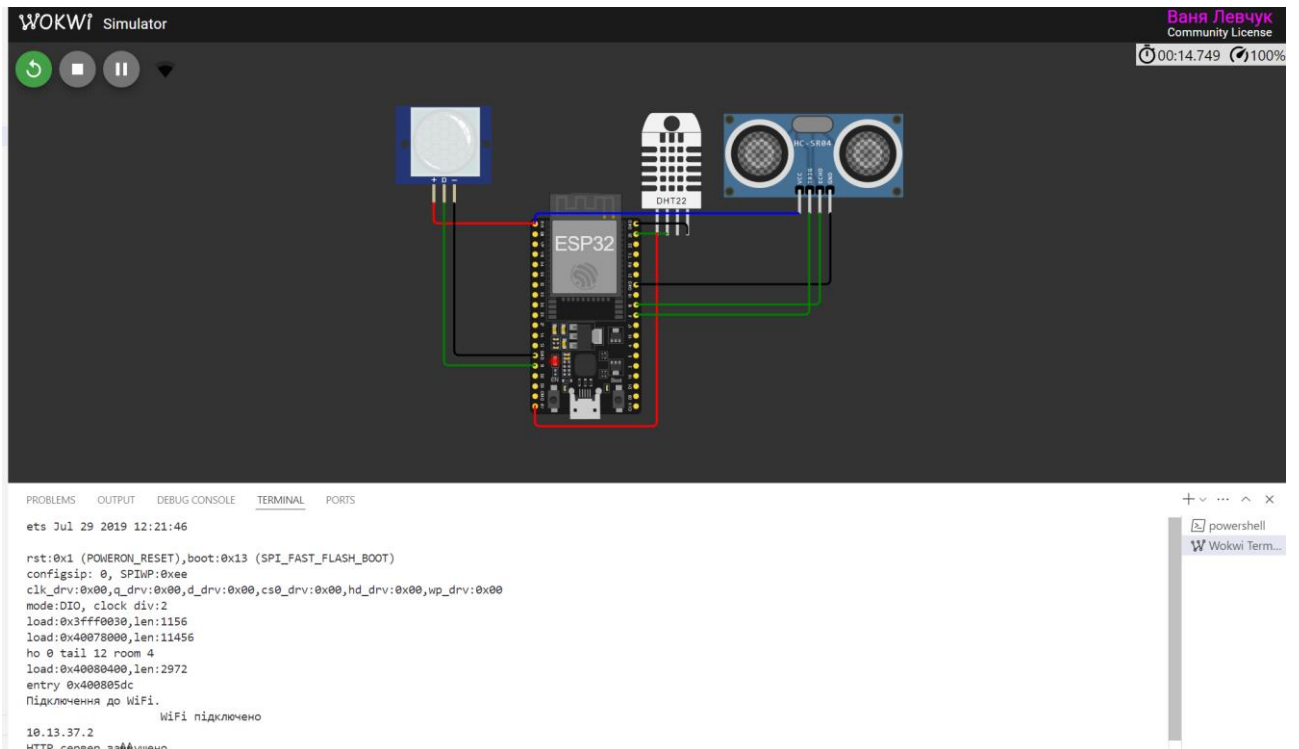


Рисунок 3.6 – Правильний запуск ESP32

Після успішного запуску ESP32 та його HTTP-сервера, наступним етапом є запуск вебзастосунку на Flask, який працює на локальному сервері. Цей застосунок формує HTML-інтерфейс для користувача, який взаємодіє з ESP32 шляхом надсилання HTTP-запитів та відображення отриманих даних.

Після відкриття вебсторінки у браузері, Flask автоматично виконує запит типу GET до IP-адреси ESP32. ESP32 формує відповідь у форматі JSON, яка включає поточні значення всіх підключених сенсорів, а також статуси підключення відповідних сенсорів.

За замовчуванням вебінтерфейс налаштований так, що при першому завантаженні сторінки у випадаючому меню автоматично обрано сенсор температури. Якщо датчик DHT22 підключено правильно і він функціонує коректно, то на сторінці буде виведено значення температури, отримане з ESP32.

Після цього користувач може за допомогою випадаючого меню <select> змінити тип сенсора для перегляду. Під час кожної зміни обраного сенсора на

					ІАЛЦ.467200.004 ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

клієнтській стороні JavaScript-обробник формує новий запит до ESP32, отримує актуальні дані і динамічно оновлює блок виводу на сторінці.

У випадку, якщо обраний сенсор недоступний або відключений, інтерфейс коректно обробляє це і виводить інформативне повідомлення про помилку, не порушуючи роботу всієї сторінки. Така логіка забезпечує гнучку і стабільну взаємодію користувача з системою в реальному часі.

На рисунку 3.7 видно, як у середовищі Wokwi встановлюються значення температури (39.3 °C) та вологості (52.5 %) через слайдери емуляції датчика DHT22, підключеного до плати ESP32. Це дозволяє створити тестове

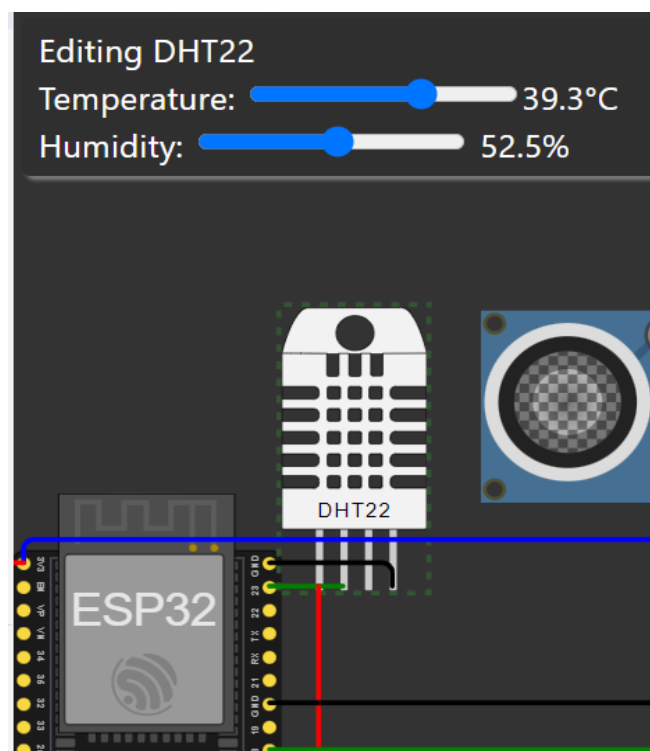


Рисунок 3.7 – Дані які передає на вебсервер сенсор температури і

середовище для подальшої перевірки відображення даних на вебсторінці

На рисунку 3.8 представлено приклад виводу даних із сенсора DHT22 у вебінтерфейсі після запуску Flask-серверу та підключення до ESP32.

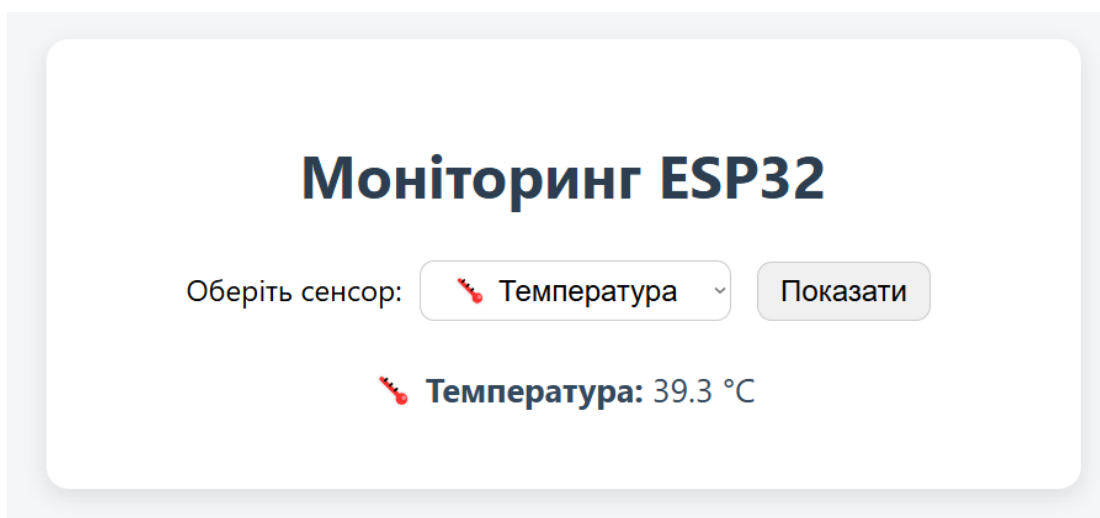


Рисунок 3.8 – Дані температури виведені вебсайтом

. У верхній частині вікна браузера видно заголовок "Моніторинг ESP32" та інтерфейс взаємодії з користувачем — випадаюче меню для вибору сенсора, кнопка "Показати" та область відображення даних.

Початково у меню обрано сенсор "Температура", і нижче система виводить значення 39.3 °C. Це число повністю збігається з тим, що було задано у вікні симуляції Wokwi для DHT22

На зображенні, після вибору сенсора "Вологість" та натискання кнопки "Показати", вебсторінка оновлює вміст без перезавантаження всієї сторінки — динамічно підставляється значення 52.5 %. Це демонструє, що дані були отримані з вебсерверу коректно.

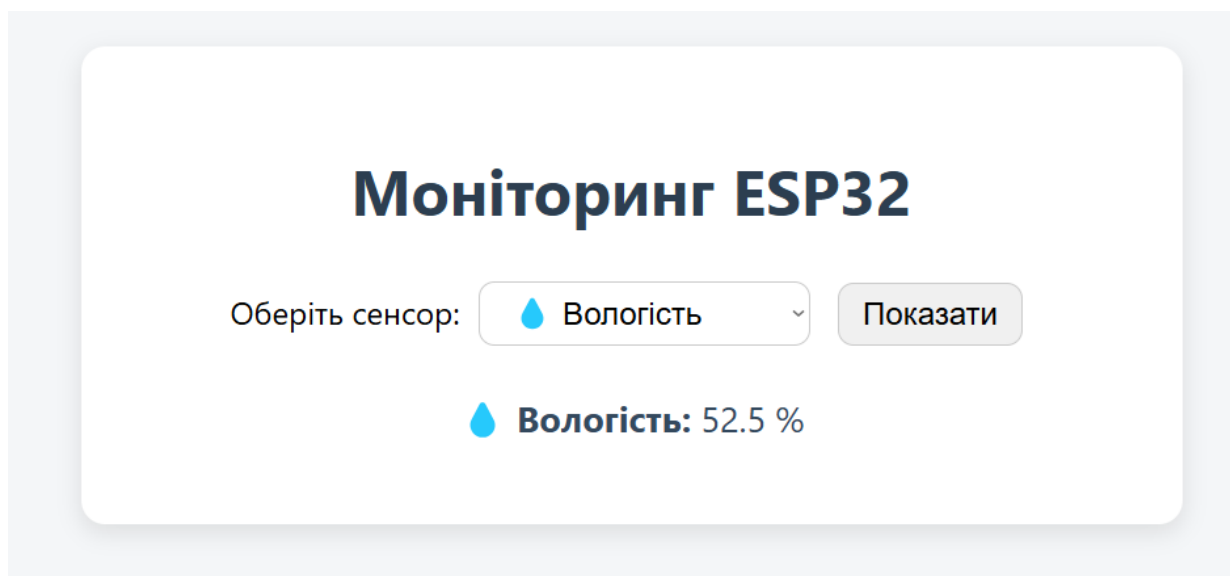


Рисунок 3.9 – Дані вологості виведені вебсайтом

На наступному етапі було проведено перевірку правильності зчитування та виводу даних з датчика відстані HC-SR04, підключеного до ESP32.

У середовищі Wokwi в параметрах симуляції для сенсора було вручну встановлено значення відстані 311 см, яке імітує фактичний вимір, який надалі має передаватися на вебінтерфейс.

На рисунку 3.10 демонструється процес встановлення контрольного значення 311 см для ультразвукового сенсора HC-SR04 у середовищі моделювання Wokwi.

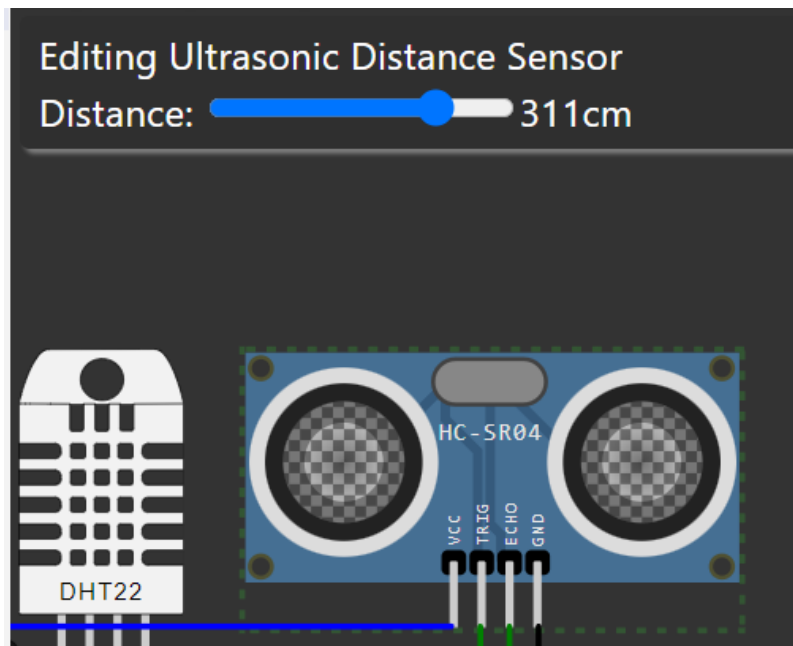


Рисунок 3.10 – Дані які передає на вебсервер сенсор дистанції

Значення виставляється за допомогою відповідного повзунка налаштувань симульованого сенсора, що дозволяє вручну задати відстань до уявного об'єкта. Це значення використовується як еталон для подальшого тестування коректності обміну даними між ESP32 і вебсервером.

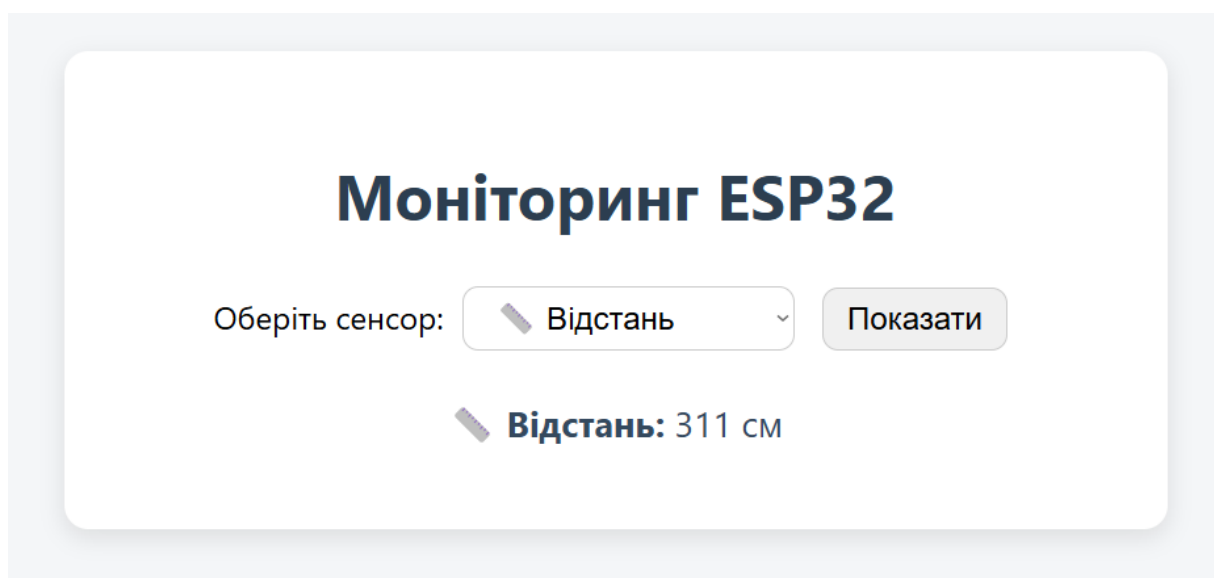


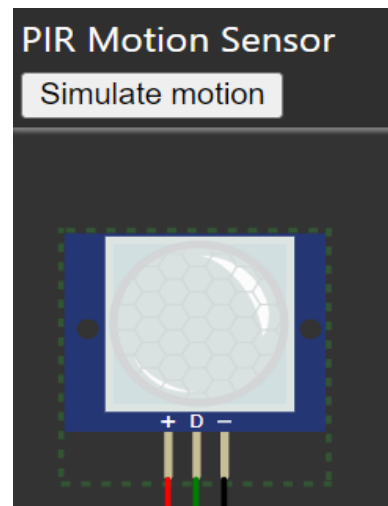
Рисунок 3.11 – Дані відстані виведені вебсайтом

На рисунку 3.11 видно, що при виборі в інтерфейсі сенсора “Відстань” вебзастосунок, створений на Flask, отримує ці дані з ESP32 через HTTP-запит, обробляє JSON-відповідь та виводить на сторінку значення 311 см. Це свідчить про успішну комунікацію між апаратною частиною, серверною

					ІАЛЦ.467200.004 ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

логікою та вебінтерфейсом. Таким чином, тест підтверджує коректну інтеграцію симульованого сенсора HC-SR04, точність обчислень на мікроконтролері та правильність відображення результатів у користувацькому інтерфейсі.

Сенсор pir. На рисунку 3.12 показано інфрачервоний сенсор руху PIR, підключений у середовищі моделювання Wokwi. У цьому середовищі PIR-сенсор емулює виявлення або відсутність руху за допомогою перемикача стану. Принцип роботи PIR-сенсора полягає у виявленні змін інфрачервоного випромінювання, які виникають, коли в полі його зору з'являється об'єкт



з температурою, вищою за фон. У середовищі Рисунок 3.12 – Сенсор PIR моделювання Wokwi цей сенсор працює так що: при натисненні на клавішу “simulate motion” змінюється інформація сенсору з 0 на 1 якщо до натиснення було 0 і з 1 на 0 якщо було 1. Для тестування коректності зчитування даних було натиснуто на клавішу двічі, тобто зараз сенсор руху передає 0, що значит що руху не виявлено.

На рисунку 3.13 показано вебінтерфейс після вибору сенсора "Рух" у випадяючому меню, після натискання кнопки “Показати”. Оскільки виведено що руху немає, то все працює коректно

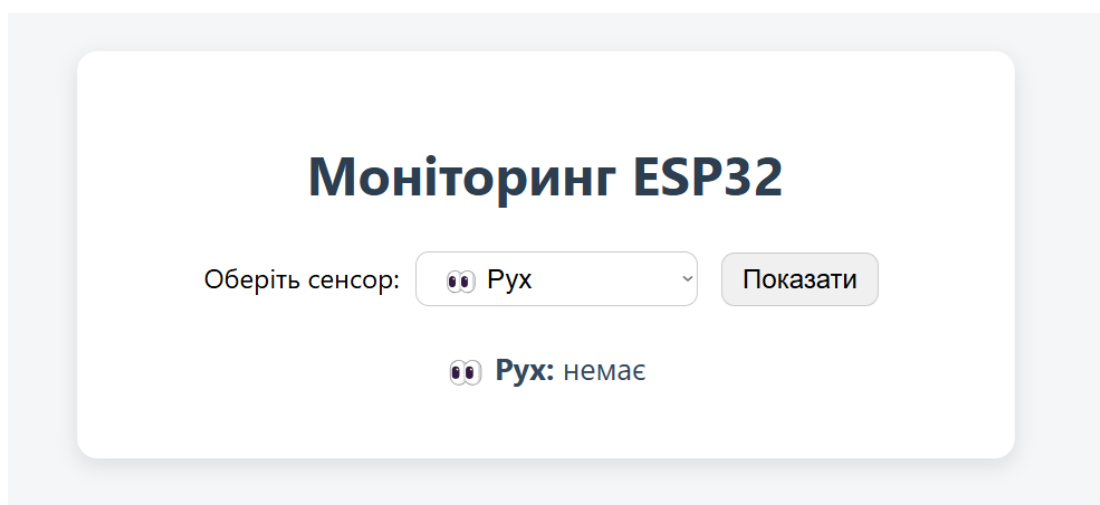


Рисунок 3.13 – Дані сенсора руху виведені вебсайтом

					ІАЛЦ.467200.004 ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

У рамках розробки пристрою моніторингу було передбачено, що не всі сенсори можуть бути підключені одночасно або бути постійно доступними. Така ситуація є типовою для модульних систем, де окремі компоненти можуть встановлюватись залежно від потреб користувача чи доступності обладнання. У зв'язку з цим у програмному коді ESP32 реалізовано перевірку стану кожного сенсора окремо, а також відповідну обробку помилок, якщо сенсор відсутній або не відповідає.

Щоб перевірити, як система поводить себе в умовах частково підключених сенсорів, було проведено тестування з навмисним відключенням одного з датчиків. Для даного тесту було відключено ультразвуковий датчик HC-SR04.

На рисунку 3.14 представлено результат запуску емуляції системи в середовищі Wokwi з умисно відключеним одним із сенсорів — зокрема, ультразвуковим датчиком відстані HC-SR04.

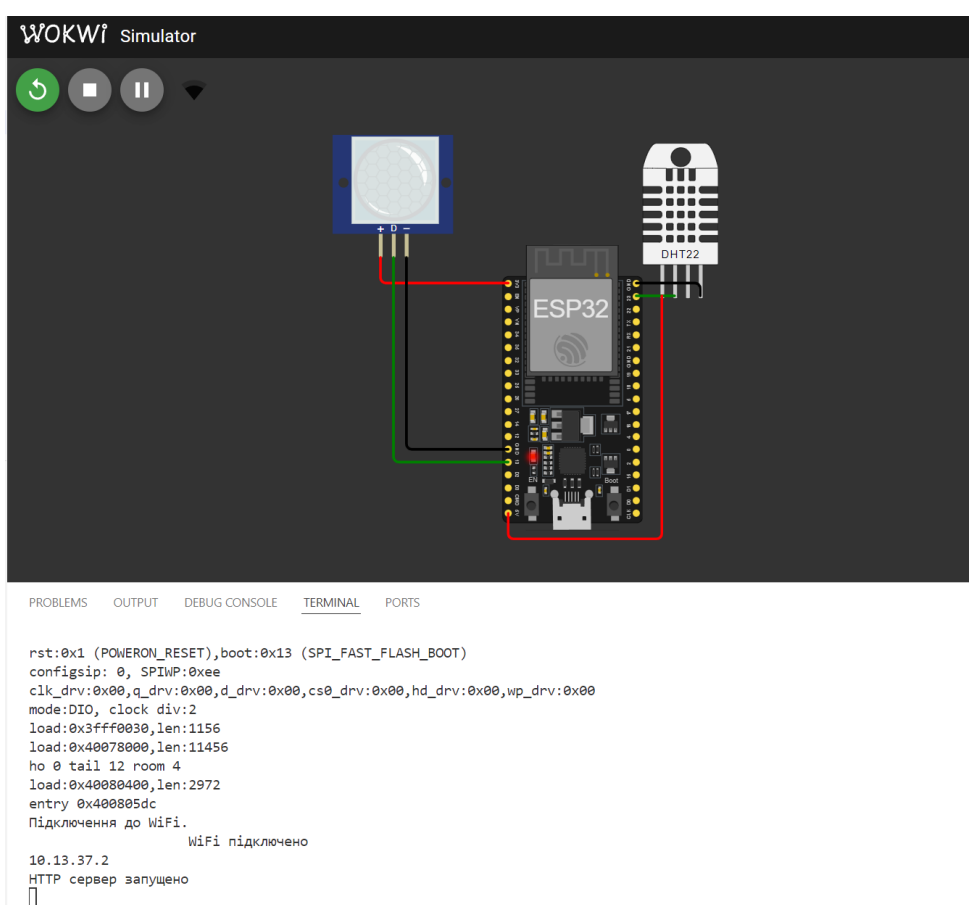


Рисунок 3.14 – Плата з відключеним датчиком HC-SR04

Незважаючи на його фізичну відсутність у схемі, ініціалізація ESP32 відбулася коректно, що підтверджується виводом у серійному моніторі. Отже, система здатна адаптуватися до змін апаратної конфігурації, залишаючись працездатною навіть у разі часткової комплектації, що є важливою перевагою для практичного використання.

На рисунку 3.15 зображено результат роботи вебінтерфейсу в умовах, коли один із сенсорів не підключений до ESP32 — зокрема, ультразвуковий сенсор HC-SR04.

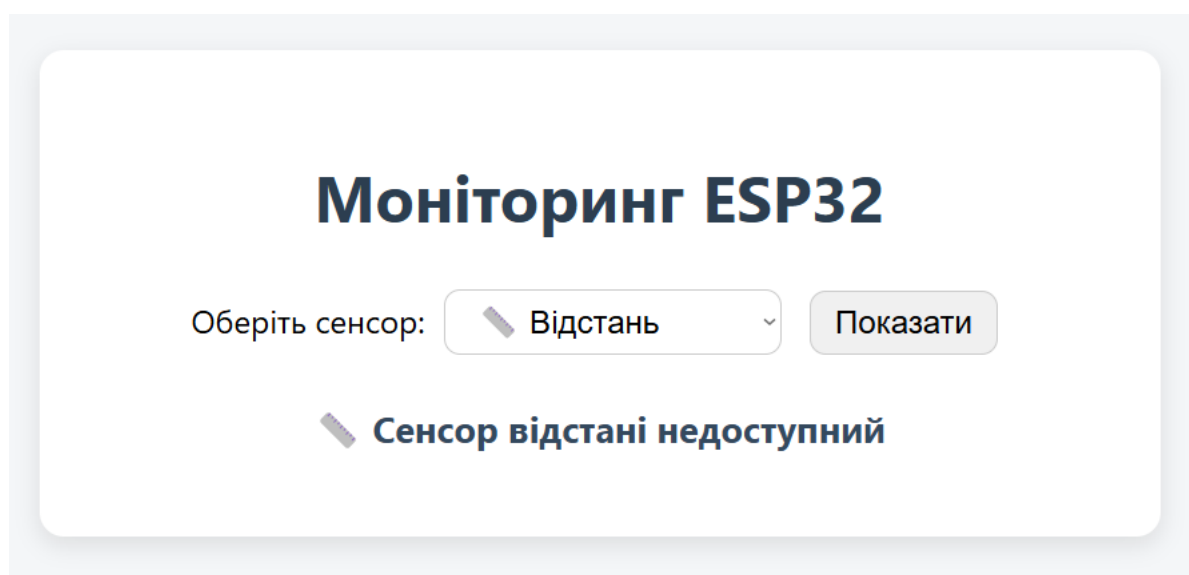


Рисунок 3.15 – Дані сенсора руху виведені вебсайтом

Після запуску Flask-додатку та переходу користувача на сторінку моніторингу, було обрано у випадаючому списку сенсор “Відстань”. Фреймворк Flask при обробці цієї відповіді виконує перевірку статусу "distance_status".

Відповідно до умов у HTML-шаблоні (Jinja2), якщо статус не "ok", вебінтерфейс не виводить некоректні значення, а замість цього динамічно формує повідомлення: Сенсор відстані недоступний.

На наступному етапі було проведено тестування сценарію, коли вебінтерфейс не може встановити з’єднання з мікроконтролером ESP32. У цьому випадку, при завантаженні вебсторінки Flask-додаток намагається виконати HTTP-запит до заданої IP-адреси ESP32.

Моніторинг ESP32

Оберіть сенсор:

Температура

Показати

Помилка з'єднання з ESP32

Рисунок 3.16 – Дані про помилку виведені вебсайтом

Якщо запит не вдається виконати протягом заданого тайм-ауту, генерується виняток. Цей виняток перехоплюється у блоці try-ехсерт на стороні Flask, і замість спроби обробити дані, додаток передає до шаблону змінну error з повідомленням: “Помилка з'єднання з ESP32”, як на рисунку 3.16.

					ІАЛЦ.467200.004 ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У результаті виконання дипломного проєкту було розроблено та протестовано багатофункціональний пристрій для моніторингу параметрів середовища на базі ESP32.

Розроблена система забезпечує зчитування даних з сенсорів температури та вологості, руху та відстані з подальшою передачею цих даних через вбудований вебсервер у форматі JSON.

Застосування мікроконтролера ESP32 забезпечило високу гнучкість і можливість локального бездротового обміну даними без зовнішніх серверів. Серверна частина на Flask дозволила зручно обробляти дані та відображати їх у вебінтерфейсі. Проведені тести підтвердили стабільну роботу пристрою та правильність обробки даних.

Отримані результати свідчать про перспективність використання даної розробки для створення інтегрованих систем моніторингу в розумних будинках, офісах чи виробничих приміщеннях.

					ІАЛЦ.467200.004 ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Espressif Announces the Launch of ESP32 Cloud on Chip and Funding by Fosun Group [Електронний ресурс]. – Режим доступу до ресурсу: https://web.archive.org/web/20170401061525/http://espressif.com/en/media_overview/news/espressif-announces-launch-esp32-cloud-chip-and-funding-fosun-group
2. Blynk [Електронний ресурс]. – Режим доступу до ресурсу: <https://docs.blynk.io/en>
3. Home Assistant [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.home-assistant.io/docs/>
4. Shelly [Електронний ресурс]. – Режим доступу до ресурсу: <https://shelly-api-docs.shelly.cloud>
5. ESP32 [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.espressif.com/en>
6. Flask [Електронний ресурс]. – Режим доступу до ресурсу: <https://flask.palletsprojects.com/en/stable/>
7. Geeksforgeeks Flask [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/flask-tutorial/>
8. Wokwi [Електронний ресурс]. – Режим доступу до ресурсу: <https://docs.wokwi.com>
9. PlatformIO [Електронний ресурс]. – Режим доступу до ресурсу: <https://docs.platformio.org>
10. HTML [Електронний ресурс]. – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTML>
11. CSS [Електронний ресурс]. – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/CSS>
12. ESP32 Simulation [Електронний ресурс]. – Режим доступу до ресурсу: <https://docs.wokwi.com/guides/esp32>

					ІАЛЦ.467200.004 ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Wokwi-dht22 [Електронний ресурс]. – Режим доступу до ресурсу: <https://docs.wokwi.com/parts/wokwi-dht22>
14. Wokwi-hc-sr04 [Електронний ресурс]. – Режим доступу до ресурсу: <https://docs.wokwi.com/parts/wokwi-hc-sr04>
15. Wokwi-pir-motion-sensor [Електронний ресурс]. – Режим доступу до ресурсу: <https://docs.wokwi.com/parts/wokwi-pir-motion-sensor>

					ІАЛЦ.467200.004 ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

