Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розроблення модуля моніторингу виробничих даних на приладобудівному виробництві
(тема)

Виконав:
здобувач 4 року навчання,
групи АКТСІ-21-3
Андрій ДИХТЕНКО
(власне ім'я, прізвище)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Системна інженерія
(повна назва освітньої програми)

Керівник Дмитро ГУРІН
(посада, власне ім'я, прізвище)

Допускається до захисту
Зав. кафедри КІТАР _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Я, Дихтенко Андрій Ігорович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на віповідне джерело.

" 24 " червень 2025 р.  Андрій ДИХТЕНКО

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва)
Тип програми Освітньо-професійна
Освітня програма Системна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« ___»_____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Дихтенко Андрію Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення модуля моніторингу виробничих даних на приладобудівному виробництві
Затверджена наказом по університету від 19.05 2025 р. № 391 СТ
2. Термін подання здобувачем роботи до екзаменаційної комісії 24.06. 2025 р.
3. Вихідні дані до роботи
3.1 Датчик DHT22 (температура: –40 до +80 °C; вологість: від 0 до 100 %)
3.2 Мікроконтролер ESP32 (підтримка Wi-Fi, живлення 3.3 В)
3.3 Веб-інтерфейс забезпечує перегляд даних у реальному часі, графіки, таблиці, аналітику та генерацію звітів через браузер.
4. Перелік питань, що потрібно опрацювати в роботі
Провести аналіз технічного завдання, обрати апаратно-програмні рішення, розробити програмне забезпечення, провести експериментальне дослідження роботи системи

5. Перелік графічного матеріалу із зазначенням, креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)
Графічний матеріал у вигляді презентації формату pptx 9 сторінок _____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Приміт-ка
1	Передатестаційна практика	02.05 – 17.05	Виконано
2	Аналіз технічного завдання	01.06 – 05.06	Виконано
3	Вибір компонентів та програмних засобів	06.06 – 10.06	Виконано
4	Розроблення програмного забезпечення	12.06 – 17.06	Виконано
5	Проведення експериментального дослідження роботи системи	17.06 – 21.06	Виконано
6	Оформлення пояснювальної записки	21.06 – 24.06	Виконано
7	Нормоконтроль	24.06	Виконано
	Подання роботи на перевірку Інтернет-сервісом StrikePlagiarism	25.06	Виконано
9	Подання роботи на рецензію	25.06	Виконано
10	Подання роботи на підпис зав. кафедри	25.06	Виконано
11	Подання кваліфікаційної роботи в ЕК	26.06	Виконано

Дата видачі завдання 02.05 2025 р.

Здобувач _____ Андрій ДИХТЕНКО
(підпис) (посада, власне ім'я, прізвище)

Керівник роботи _____ Дмитро ГУРІН
(підпис) (власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 63 с., 46 рис., 3 дод., 16 джерел.

МОНІТОРИНГ, ІНТЕРНЕТ РЕЧЕЙ, ВЕБ-ІНТЕРФЕЙС, АНАЛІЗ ДАНИХ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об’єкт розробки – процес моніторінгу виробничих процесів.

Предмет розробки – модуль моніторингу виробничих даних.

Мета роботи – підвищення ефективності моніторінгу та аналізу технологічних процесів шляхом розробки модуля для моніторингу виробничих даних на підприємстві

В першому розділі було проаналізовано предметну область, існуючі рішення та системи моніторингу.

В другому розділі було обрано та обґрунтовано апаратно – програмні рішення, а саме вибір компонентів, програмних засобів та описано підключення датчиків до мікроконтролера.

В третьому розділі було описано програмну логіку всієї системи. А саме передачу даних, серверну частину, інтерфейс користувача та надсилання повідомлень.

В четвертому розділі було продемонстровано роботу системи у повному обсязі.

У результаті розроблена повноцінна система моніторингу за виробничими даними за допомогою зручного веб-інтерфейсу.

ABSTRACT

The explanatory note: 63 p., 46 pict., 3 app., 16 names.

MONITORING, INTERNET OF THINGS, WEB INTERFACE, DATA ANALYSIS, SOFTWARE.

The object of development – is the process of monitoring production processes.

The subject of development – is the production data monitoring module.

The purpose of the work – is to increase the efficiency of monitoring and analysis of technological processes by developing a module for monitoring production data at the enterprise

The first section analyzed the subject area, existing solutions and monitoring systems.

The second section selected and justified hardware and software solutions, namely the selection of components, software and described the connection of sensors to the microcontroller.

The third section described the software logic of the entire system. Namely, data transfer, server part, user interface and sending messages.

The fourth section demonstrated the operation of the system in full.

As a result, a full-fledged production data monitoring system was developed using a convenient web interface.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЛК – програмований логічний контролер

СУБД – система управління базами даних

IoT – Internet of Things (Інтернет речей)

KPI – Key Performance Indicator (Ключові показники ефективності)

SCADA – Supervisory Control and Data Acquisition (Система диспетчерського контролю та збору даних)

Wi-Fi – Wireless Fidelity (Бездротова мережа)

ВСТУП

У сучасному, динамічно змінюваному світі, де розвиток технологій відбувається блисковично, промислові підприємства мусять пристосовуватися до дедалі вищих вимог щодо ефективності, стабільності та якості виробництва. При цьому, особливої ваги набуває здатність миттєво реагувати на зміни у параметрах технологічних процесів. Зрештою, надійний моніторинг стає не простим допоміжним інструментом, а критично важливою частиною будь-якої сучасної системи. Саме він дозволяє вчасно попереджати про несправності, знижувати помітно витрати та забезпечувати безперебійну роботу всього виробничого циклу.

Впровадження таких систем, здійснюється з використанням сучасних датчиків та мікроконтролерів. Варто зазначити, що це відкриває нові горизонти в можливостях контролю за станом обладнання та навколишнього середовища.

В межах цієї дипломної роботи було розроблено моніторингову систему. Вона забезпечує постійний збір даних з датчиків, їх збереження, подальшу обробку та візуалізацію через веб-інтерфейс у режимі реального часу. Додатково реалізовано функцію автоматичного сповіщення через популярний месенджер «Telegram», у випадку якщо дані, які надійшли до системи перевищують максимально допустиме значення. Такий підхід дозволяє забезпечити швидке реагування на зміни умов виробництва. Це підвищує загальну надійність та стійкість виробничих процесів на виробництві.

Ціллю сталого розвитку є ЦСР 9 – Промисловість, інновації та інфраструктура. Дана система сприяє цифровізації виробництва, що є частиною інноваційного розвитку.

Актуальність теми зумовлена кількома важливими факторами. Це нагальна потреба у мінімізації впливу людського фактору та прагнення до підвищення контролю на всіх етапах виробництва. Зберігання та обробка виробничих даних дозволяє підприємствам підвищити рівень автоматизації, покращити управління процесами та забезпечити необхідний рівень технологічної безпеки. Система може

бути масштабована та адаптовано до різних типів підприємств.

Об’єкт розробки – процес моніторінгу виробничих процесів.

Предмет розробки – модуль моніторингу виробничих даних.

Мета роботи – підвищення ефективності моніторінгу та аналізу технологічних процесів шляхом розробки модуля для моніторингу виробничих даних на підприємстві

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз існуючих систем моніторингу та існуючий рішень;

– здійснити вибір апаратно-програмних рішень для стабільної роботи системи;

– розробити програмне забезпечення – серверну частину яка забезпечить збір, зберігання та фільтрацію даних;

– розробити веб-інтерфейс для взаємодії системи з користувачем;

– провести експериментальне дослідження роботи системи;

– оформити кваліфікаційну роботу згідно ДСТУ 3008:2015 [1], а також з методичними вказівками з підготовки й оформлення кваліфікаційної роботи здобувачами першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп’ютерно-інтегровані технології освітньої програми «Системна інженерія» [2].

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Аналіз предметної області

У багатьох виробничих процесах сьогодні важливо мати змогу бачити, що відбувається прямо під час роботи. Якщо, скажімо, температура чи вологість змінюються – це може одразу вплинути на обладнання або якість продукції. Тому своєчасне виявлення будь-яких змін дозволяє уникнути збоїв, а іноді й повного зупинення виробництва. Крім того, це допомагає зменшити витрати, бо не треба ремонтувати те, що ще працює, або втрачати сировину через несподівані відхилення.

Моніторинг полягає в тому, щоб збирати показники з обладнання або навколишнього середовища, далі ці дані кудись передаються, зберігаються й обробляються. Але цього мало – потрібно, щоб інформація ще й зручно відображалась. Це все реалізується через спеціальні технічні та програмні рішення, які поєднуються з наявними системами на підприємстві. Залежно від того, що саме потрібно контролювати і як часто, система може бути дуже простою або більш складною.

У багатьох випадках важливо, щоб дані надходили без затримок. Особливо коли йдеться про температуру або інші швидкоплинні параметри. І добре, коли система не просто записує інформацію, а й одразу аналізує її, а при необхідності – надсилає сповіщення. Так користувач або технічний персонал можуть швидко зреагувати. Це допомагає не лише уникнути поломок, але й планувати обслуговування обладнання тоді, коли це справді потрібно, а не наосліп.

Серед основних параметрів, які зазвичай підлягають моніторингу, можна виділити:

- температуру (з метою запобігання перегріву);
- тиск (для контролю безпечності процесів);
- рівень рідин чи сипучих матеріалів;

– вологість;
– вібрації та шум (як індикатори механічних несправностей);
– споживання електроенергії та інші енергетичні параметри.

Для всіх зазначених параметрів, які потрібно контролювати, підбирається відповідний датчик. Важливо, щоб він давав достатньо точні результати, працював стабільно і легко підключався до основної системи. Після зчитування даних датчиком, вони передаються далі –або на мікроконтролер, або на сервер. Там дані проходять базову обробку: відсіюється або зберігається. Після цього вся інформація записується в базу даних, з якої можна проаналізувати її пізніше або побудувати звіт.

Зараз цифрові технології дозволяють значно більше. Наприклад, Інтернет речей (IoT) дає можливість з’єднувати багато пристроїв між собою. Дані можуть передаватися по Wi-Fi або мережі. І найголовніше – не обов’язково бути в тому самому приміщенні, щоб переглянути результати. Все це доступно через інтернет. Окремо варто згадати й нові алгоритми, які аналізують інформацію не тільки "постфактум", а й можуть прогнозувати, що буде далі.

Отже, тема моніторингу зараз охоплює цілу систему рішень: від самих датчиків до програм, які з ними працюють. Це не тільки про техніку, а й про те, як усе організовано. І саме завдяки таким системам можна краще керувати процесами, уникати непотрібних втрат і швидше реагувати на будь-які зміни навколо.

1.2 Аналіз існуючих рішень та систем моніторингу

У сучасному динамічно змінюваному світі, де виробничі процеси не лише стають більш складнішими, а й насиченішими та динамічнішими. Без належного, постійного контролю складно підтримувати їх стабільну роботу, майже неможливо. Саме тому, задля не допуску втрати контролю над ситуацією, підприємствам важливо та необхідно мати постійний доступ до актуальної інформації про те, що відбувається на їх виробничих майданчиках. Варто зазначити, що просте спостереження вже не дає потрібного ефекту, не вирішує ці

питання як раніше. А для ефективного уникнення збоїв або критичних ситуацій потрібно постійно контролювати виробництво. Саме тут і стає до нагоди автоматичний моніторинг, він у цьому плані відіграє, безумовно, важливу роль. Він дає постійну можливість слідкувати за технічним станом обладнання, умовами середовища, енергоспоживанням і багатьма іншими важливими моментами. Він дає реальну змогу бачити всі зміни на виробництві, дані збираються з різних джерел. Коли якісь показники виходять за максимально допустимі межі – це можна помітити вчасно й уникнути простоїв або серйозніших проблем. Як результат це дозволяє зменшити витрати, забезпечити безпеку для персоналу, створити нормальні умови праці та суттєво підвищити ефективність виробництва.

Власне, такі системи давно вже стали, органічно вписаною, частиною цифрової модернізації виробництва. Вони дозволяють не лише бачити дані, а й аналізувати їх, формувати звіти та ефективно ділитися інформацією між відділами. В результаті всі працівники на виробництві можуть знати що відбувається в режимі реального часу. Від оператора в цеху до технічного директора – можуть бачити, як працює система, і приймати рішення на основі актуальної інформації. Це дає можливість планувати роботи у подальшому та вносити зміни в реальному часі, приймати вдалі управлінські рішення, вчасно виконувати обслуговування, і в цілому – робити виробництво безпечнішим та ефективнішим.

Щодо даних, котрі збираються такими системами, то вони можуть бути дуже різноманітними. Наприклад, температура, вологість, тиск, кількість обертів або навіть цикли роботи обладнання. Важливо, що зібрана інформація може не просто зберігатися, а проходити фільтрацію, оброблятися, в залежності від потреб. Далі подається у зрозумілому вигляді для будь-якого користувача – таблиці, графіки, аналітика. Безперечно це набагато зручніше, через те, що людина бачить всю інформацію в одному місці й не витрачає час на ручний аналіз.

Запровадження таких рішень, має на меті здійснюється не лише заради контролювання, а заради того, щоб керівництво могло приймати обґрунтовані та виважені рішення базуючись на щось конкретне. Якщо бачити, які зміни працюють краще, коли саме зростає енергоспоживання або де виникають затримки — все це

можна врахувати під час планування. У такий спосіб моніторинг стає не просто технічним доповненням, а частиною ефективного управління.

Додатковою, але не достатньо важливою є перевага запровадження компактних моніторингових систем, адже вони не обов'язково мають бути громіздкими. Це означає що, їх можна впровадити як на великому підприємстві, так і в невеликому цеху. Їх можна підлаштувати під потреби, дозволяючи масштабувати їх у подальшому, додавати нові датчики, інтегрувати з іншими програмами або модулями, і завдяки цьому вони залишаються гнучкими.

У підсумку, головне, що дає така система – це надання достовірних й своєчасних даних, з якими дійсно можна працювати. Вони допомагають уникнути затримок, запобігти аваріям, планувати технічне обслуговування, і навіть прогнозувати, де може виникнути проблема в майбутньому. Це вже дає змогу не тільки контролювати ситуацію на виробництві та забезпечувати безпеку та ефективність виробництва, а повноцінне управління на основі фактів.

2 ВИБІР ТА ОБГРУНТУВАННЯ АПАРАТНО – ПРОГРАМНИХ РІШЕНЬ

2.1 Обґрунтування вибору компонентів

У процесі створення модуля моніторингової системи було важливо обрати такі компоненти, які б одночасно відповідали технічним вимогам і залишалися простими в реалізації, а саме: функціональні можливості, проста інтеграція до системи та економічний фактор.

Головним елементом став мікроконтролер ESP-WROOM-32. Передусім, через зручність: у ньому вже вбудований Wi-Fi модуль, а це дозволяє не тягнути зайвих дротів – зручно, особливо якщо розміщення модуля не фіксоване. Також, ESP32 має доволі сильну підтримку в спільноті, що полегшило роботу з ним – багато документації та бібліотек.

В основі цього модуля лежить мікросхема ESP32, яка розроблена як масштабована та адаптивна. Є 2 ядра ЦП якими можна індивідуально керувати або живити, а тактова частота регулюється від 80 МГц до 240 МГц. Користувач також може вимкнути центральний процесор і використовувати співпроцесор з низьким енергоспоживанням для постійного моніторингу периферійних пристроїв за зміни або перехід порогів. ESP32 об’єднує багатий набір периферійних пристроїв, починаючи від ємнісних сенсорних пристроїв датчики, датчики Холла, малошумні сенсорні підсилювачі, високошвидкісний SDIO/SPI, UART, I2S і I2C. Має WiFi 802.11n 2,4 Гц з максимальною швидкістю 150 Мбіт/с та Bluetooth v4.2 BR/EDR and BLE. ESP32 споживає незначну кількість енергії – в активному режимі це приблизно 80–240 мА, а у сплячому – одиниці мікроампер. Це дозволяє використовувати його в автономних або енергоощадних системах. Працює на живленні від 2,2 В – 3,6 В. Пам’яті в ESP32 достатньо для більшості завдань: це близько 520 КБ SRAM, 448 KBytes ROM і 4 МБ Flash. У нашому випадку цієї кількості цілком вистачає для зчитування показників, їх буферизації та передачі.

Має 38 pin (рис. 2.1).

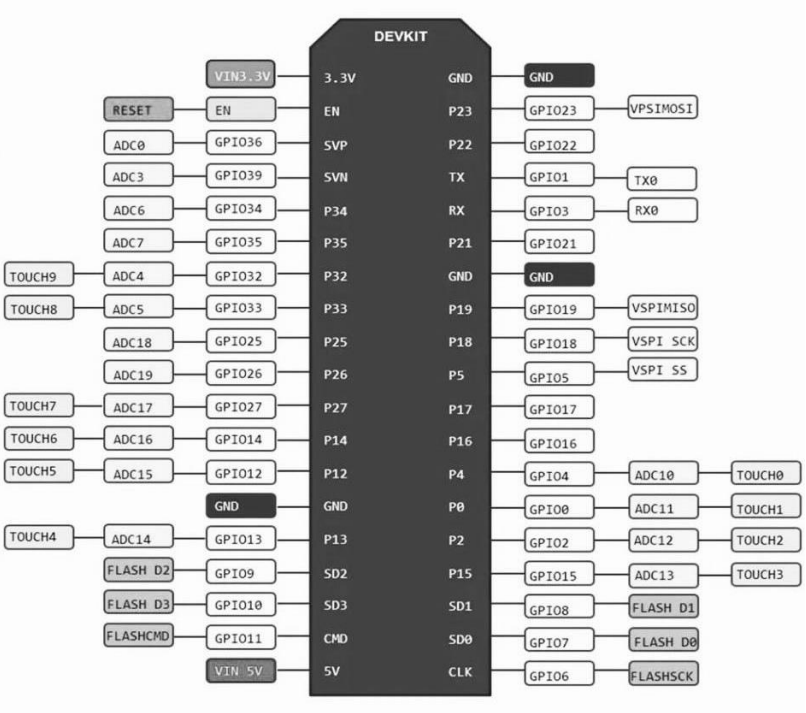


Рисунок 2.1 – Розпіновка ESP32

Отже, вважаю що цей модуль є оптимальним варіантом через його продуктивність, гнучкість та невелику ціну.

Для вимірювання даних, обрано датчик DHT22, який вимірює температуру та вологість повітря. Працює на живленні 3,3 В, тобто стабільно працює разом з ESP32. Виводить калібрований цифровий сигнал. Ця модель забезпечує вищу точність та ширший діапазон вимірювання у порівнянні з іншими датчиками в цьому ціновому сегменті. Отже, невеликий розмір, низьке енергоспоживання та простота взаємодії робить його оптимальним варіантом для цієї системи.

Загалом, елементи які були обрані, створюють стабільну систему здатну збирати та передавати дані для подальшої обробки.

2.2 Підключення до мікроконтролера

Підключення датчика до мікроконтролера відбулося за допомогою 3 пінів, а саме: GND – GND, 3.3V – 3.3V, OUT – G4. На схемі нижче (рис. 2.2) показано з'єднання між мікроконтролером та сенсорами.

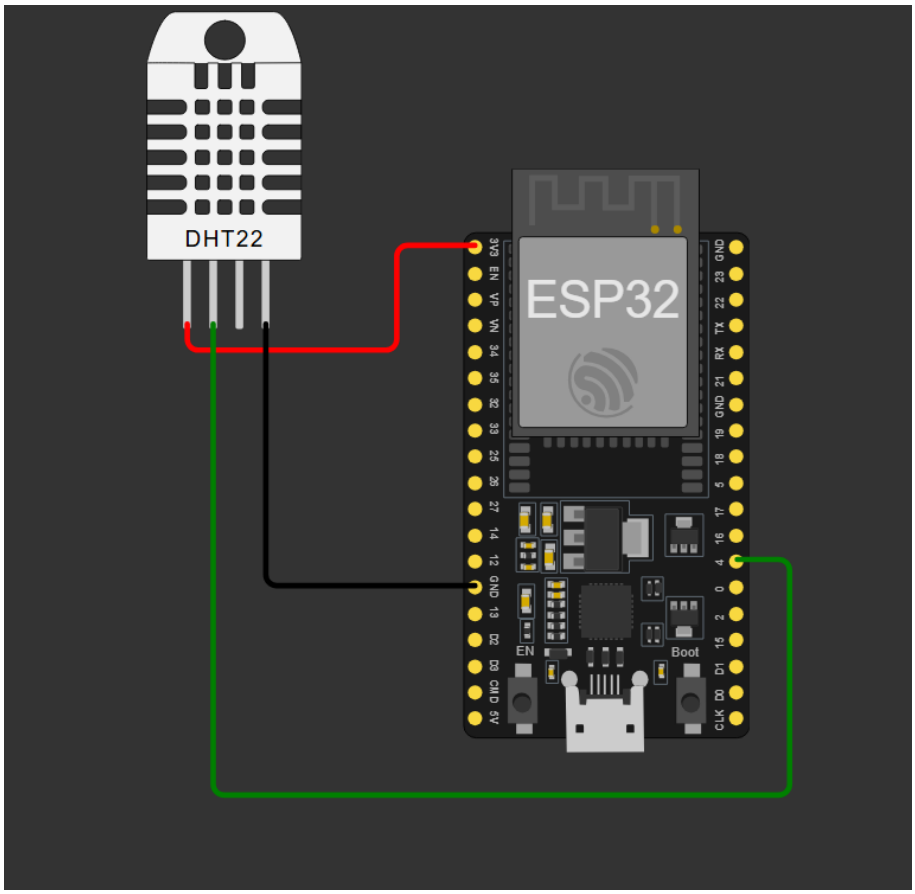


Рисунок 2.2 – Схема електричного підключення

2.3 Обґрунтування вибору програмних засобів

Для реалізації серверної, клієнтської та комунікаційної частин системи моніторингу було обрано набір технологій, які забезпечують швидкість розробки, стабільну роботу й легке масштабування.

Основою проекту було обрано фреймворк Flask для мови програмування Python. Має легку структуру, дає змогу швидко створити веб-сервер, має можливість для масштабування до складних програм. Важливим плюсом цього фреймворку є зручна інтеграція з Python бібліотеками та мінімальна залежність від сторонніх засобів. Це дозволило напряму використовувати бібліотеки для роботи з базами даних SQLite, також з бібліотекою для створення «.docx» файлів. Відразу працює коректно з HTML-шаблонами без додаткових пакетів.

У якості бази даних було використано SQLite – це компактна реляційна СУБД. На відміну від інших баз даних, вона не має окремого серверного процесу. Тобто зберігає всі дані у вигляді одного файлу. Це великий плюс, адже дозволяє уникнути розгортання окремого серверного програмного забезпечення для збереження даних. Вона працює швидко та стабільно. Продуктивність зазвичай не падає, навіть у середовищах з малим обсягом пам’яті. Форматом бази є один файл, який зберігається безпосередньо на комп’ютері. Через це і підвищується швидкість та продуктивність роботи. Також, важливим моментом є мінімалізм, якому притримується SQLite. Це додає простоту рішень та адміністрування. Отже, такий варіант чудово підходить для системи моніторингу.

Візуальну частину було реалізовано за допомогою класичних вебтехнологій. Для структури сторінок було обрано HTML, для оформлення CSS та JavaScript для оновлення графіків та перезавантаження сторінок.

Для забезпечення оперативного інформування користувачам про перевищення максимально допустимих значень температури, було реалізовано сповіщення в Telegram. Використання Telegram Bot API дає змогу автоматично надіслати повідомлення у вказану групу або чат. Це дає змогу швидко та зручно розробити спосіб надсилання повідомлень без створення додаткових додатків. Повідомлення миттєво надсилається користувачам через популярний месенджер, що розширює функціональні можливості системи.

2.4 Комп’ютерне моделювання системи автоматичного управління

До проекту може бути під’єднаний сервопривод. Він може виконувати роль виконавчого механізму в системі. Наприклад, при перевищенні температури буде відкриватися певний вентиляційний клапан. Для передбачення його роботи у складі системи, розрахуємо його стійкість.

Реакція сервоприводу не миттєва, він буде поступово змінювати положення, відповідно вхідному сигналу. Така поведінка є системою першого порядку, її можна описати передаточною функцією у вигляді:

$$W(s)=Ts+1K, \tag{2.1}$$

де T = 10 с – стала часу, що враховує механічну інерцію сервопривода;

K = 1 – коефіцієнт підсилення (нормований для спрощення).

Для оцінки стійкості потрібно визначити, що всі корені знаменника передаточної функції мають від’ємні дійсні частини.

$$10s+1=0\Rightarrow s=-0,1. \tag{2.2}$$

Можна побачити, що корінь рівняння має від’ємне значення, отже система стійка. За допомогою мови програмування Python, представлено реакцію на одиничний вплив, наприклад зростання температури, що призводить до роботи сервоприводу (рис. 2.3).

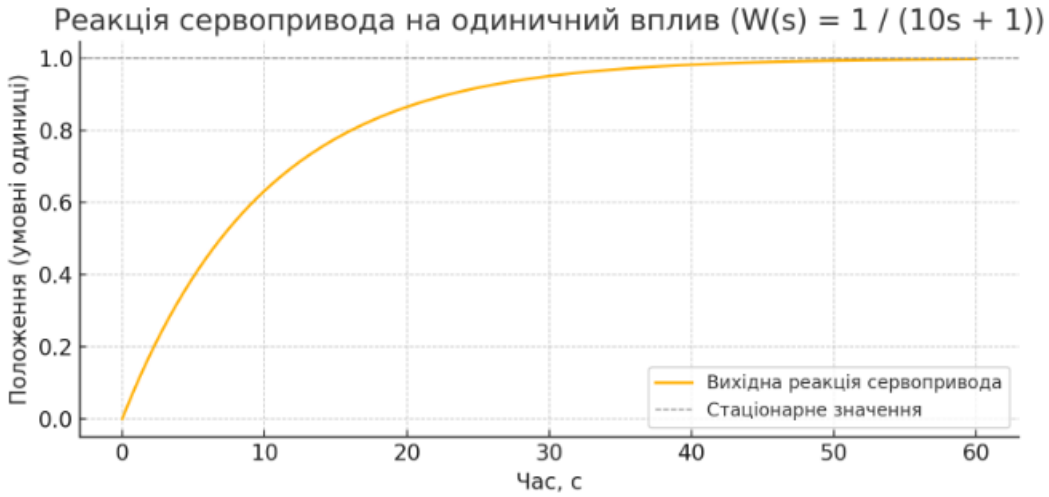


Рисунок 2.3 – Графік реакції впливу на сервопривод

Код програми:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

# Параметри системи
K = 1
T = 10

# Передаточна функція W(s) = K / (T*s + 1)
num = [K]
den = [T, 1]
system = signal.TransferFunction(num, den)

t = np.linspace(0, 60, 600)
t, y = signal.step(system, T=t)

plt.figure(figsize=(8, 4))
plt.plot(t, y, label='Вихідна реакція сервопривода')
plt.axhline(1, color='gray', linestyle='--', linewidth=0.8, label='Стаціонарне
```

```
значення')
    plt.title('Реакція сервопривода на одиничний вплив (W(s) = 1 / (10s + 1))')
    plt.xlabel('Час, с')
    plt.ylabel('Положення (умовні одиниці)')
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.show()
```

З отриманого графіку, можна побачити що система виходить на стабільне положення, тобто повну реакцію на вхідний сигнал поступово, з характерною інерцією.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧННЯ

3.1 Передача даних з ESP32

Для початку роботи з ESP32, потрібно було виконати підготовчу роботу. По-перше, модуль підключається до комп'ютера через USB-C кабель. Тут може виникнути проблема з відсутністю драйвера, оскільки деякі плати ESP32 використовують чіп CP2102 або CH340, для яких Windows не завжди автоматично встановлює підтримку. В моєму випадку довелося встановлювати драйвер з офіційного сайту Silicon Labs, після того, як драйвер був встановлений комп'ютер почав бачити плату як COM-порт, доступний у середовищі розробки.

Як середовище розробки було обрано Arduino IDE, вона проста у використанні, дозволяє швидко завантажити код на пристрій, має багато бібліотек та велику кількість інформації в інтернеті. Перед написанням коду потрібно додати підтримку ESP32 до Arduino IDE. Для цього у боковому меню ,потрібно обрати «Boards Manager» та встановити набір «esp32 by Espressif Systems» (рис. 3.1). Після цього у меню «Інструменти» обрати модель плати (рис. 3.2), в моєму випадку це ESP32 Dev Module та відповідний COM-порт.

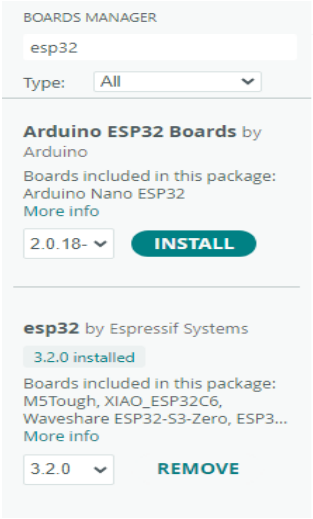


Рисунок 3.1 – «Boards Manager»

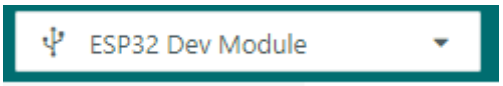


Рисунок 3.2 – «Інструменти»

Після цього можна починати роботу над кодом. Основне завдання ESP – зчитати показники температури та вологості з датчика DHT22 і передати ці дані на сервер. Для цього використовується бібліотека DHT.h для взаємодії з датчиком, а також WiFi.h і HTTPClient.h для підключення до мережі та відправлення HTTP-запитів (рис. 3.3).

```
1   #include <WiFi.h>
2   #include <HTTPClient.h>
3   #include "DHT.h"
4
```

Рисунок 3.3 – Підключені бібліотеки

Потім в коді задається ім’я Wi-Fi мережі, пароль та IP-адресу сервера до якого будуть надходити дані (рис. 3.4).

```
const char* ssid = "TP-Link_    ";
const char* password = "        ";
const char* serverUrl = "http://192.          :5000/data";
```

Рисунок 3.4 – Підключення до Wi-Fi мережі

Після, потрібно описати підключення датчику (рис. 3.5).

```
#define DHTPIN 4          // Ніжка, підключена до DATA
#define DHTTYPE DHT22     // Тип датчика
```

Рисунок 3.5 – Підключення датчику

Далі, у головному циклі зчитуються значення з DHT22, і формуються POST-запити до Flask-сервера, де ці значення передаються у форматі JSON (рис. 3.6).

Мікроконтролер відправляє HTTP POST, з тілом запиту {"temperature": 24.3, "humidity": 56.8}. Сервер, у свою чергу, приймає ці дані, розпізнає їх і записує до бази даних.

Також, додано HTTP вивід, для розуміння помилок ,які можуть виникнути під час роботи або підтвердження коректної роботи системи.

```
void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Помилка зчитування з DHT22");
  } else {
    Serial.printf("Температура: %.2f°C | Вологість: %.2f%%\n", temperature, humidity);

    if (WiFi.status() == WL_CONNECTED) {
      HTTPClient http;
      http.begin(serverUrl);
      http.addHeader("Content-Type", "application/json");

      String jsonData = "{\"temperature\": " + String(temperature, 2) +
                        ", \"humidity\": " + String(humidity, 2) + "}";

      int httpResponseCode = http.POST(jsonData);
      Serial.println("HTTP відповідь: " + String(httpResponseCode));
      http.end();
    } else {
      Serial.println("WiFi не підключено");
    }
  }
```

Рисунок 3.6 – Основна логіка відправки даних

Також, важливим моментом є затримка між запитами, їх необхідно виставити, щоб не перевантажити сервер частими зверненнями. В мене виставлено інтервал в 30 секунд (рис. 3.7). Цього значення достатньо для збору даних в режимі реального часу без перевантаження системи.

```
delay(30000);
```

Рисунок 3.7 – Інтервал збору даних

Після завантаження коду на ESP32, він починає працювати автономно. При ввімкненні підключається до Wi-Fi мережі та починає збір даних.

Отже, вся частина передачі даних з ESP32 містить підключення

мікроконтролера до комп’ютера, налаштування середовища та написання коду. Всі етапи є важливими для забезпечення стабільної роботи системи.

3.2 Серверна частина

Серверна частина проекту має головну роль у забезпеченні збору, обробки та збереженні даних, які надходять з ESP32. Основна задача – приймати HTTP-запити, отримувати з них інформацію про температуру та вологість, та потім додавати ці дані до бази даних для подальшої роботи з ними та відображенням на сайті.

Перш за все, потрібно встановити Flask через pip (pip install flask). Потім, створюється головний файл – app.py. В ньому описано основну логіку обробки запитів.

Після, здійснюється підключення бібліотек і модулів, які забезпечують роботу серверу (рис. 3.8). Flask для побудови веб-інтерфейсу, sqlite3 для взаємодії з базою даних. Також підключено модулі для роботи з датами, HTTP-запитами та для створення звітів та шаблонів.

```
from flask import Flask, render_template, jsonify, send_file, request
import sqlite3
from datetime import datetime, timedelta
from report_generator import generate_report
import requests
```

Рисунок 3.8 – Підключені бібліотеки

У Flask-додатку створено окремий маршрут /data, куди ESP32 надсилає HTTP POST-запити. У тілі цих запитів міститься JSON-об’єкт із двома змінними, які мікроконтролер отримує з датчика. Це температура ("temperature") та вологість ("humidity"). Далі, у Flask ці дані витягуються через request.get_json(), та до них додається мітка часу (автоматично). Та після цього ця інформація записується до бази даних (рис. 3.9).

```python
@app.route('/data', methods=['POST'])
def receive_data():
    data = request.json
    print(data)

    timestamp = datetime.now().isoformat()
    temperature = data.get('temperature')
    humidity = data.get('humidity')

    if temperature is None or humidity is None:
        return {'status': 'error', 'message': 'Missing temperature or humidity'}

    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    cursor.execute(
        "INSERT INTO data (timestamp, sensor, value) VALUES (?, ?, ?)",
        (timestamp, 'temp', temperature)
    )

    cursor.execute(
        "INSERT INTO data (timestamp, sensor, value) VALUES (?, ?, ?)",
        (timestamp, 'hum', humidity)
    )

    conn.commit()
    conn.close()

    try:
        if float(temperature) > 30.0:
            msg = f"🔒 Увага! Температура перевищила норму: {temperature}°C"
            send_telegram_message(msg)
    except ValueError:
        print("Не вдалося перетворити температуру у число.")
```

Рисунок 3.9 – Маршрут «/data»

При кожному зверненні до /data новий запис додається в таблицю бази даних. Її створено за допомогою окремого файлу, щоб описати структуру. Структура бази проста – одна таблиця, в якій є поля для температури, вологості, дати, типу датчика та часу (рис. 3.10).

```python
import sqlite3

conn = sqlite3.connect("database.db")
c = conn.cursor()

c.execute('''
    CREATE TABLE data (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        timestamp TEXT,
        sensor TEXT,
        value REAL,
        temperature REAL,
        humidity REAL
    )
''')

conn.commit()
conn.close()
```

Рисунок 3.10 – Створення бази даних

Також, важливою функцією у серверній частині є def get_data (рис. 3.11). Вона дістає дані з бази даних, але робить в залежності від того, які параметри їй передаються. Тобто, вона забирає результат запиту до бази даних у вигляді списку, для зручного швидкого отримування потрібних даних.

```
def get_data(interval_minutes=None, limit=None):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    if interval_minutes:
        cutoff_time = datetime.now() - timedelta(minutes=interval_minutes)
        cursor.execute("SELECT * FROM data WHERE sensor IN ('temp', 'hum')")
        data = cursor.fetchall()
        conn.close()

        filtered = []
        for row in data:
            timestamp = row[1]
            try:
                dt = datetime.fromisoformat(timestamp)
                if dt >= cutoff_time:
                    filtered.append(row)
            except ValueError:
                continue
        return filtered[::-1]

    elif limit:
        cursor.execute("SELECT * FROM data ORDER BY timestamp DESC LIMIT ?", (li
        data = cursor.fetchall()
        conn.close()
        return data[::-1]

    else:
        cursor.execute("SELECT * FROM data")
        data = cursor.fetchall()
        conn.close()
        return data[::-1]
```

Рисунок 3.11 – Функція def get_data

Крім прийому даних, серверна частина відповідає за за відправку інформації на клієнтську частину сайту. Для цього реалізовано маршрути, такі як /chart (рис. 3.13), /table_short (рис. 3.12), table_long, /analytics. Вони відповідають за відправку оброблених даних на відповідну HTML-сторінку. Залежно від запиту дані надсилаються з різними часовими інтервалами або обробляються для виведення у виді графіку або таблиці.

```
@app.route('/table_short')
def table_short():
    rows = get_data(limit=30)
    return render_template('table_short.html', rows=rows)
```

Рисунок 3.12 – Маршрут до таблиці

```python
@app.route('/chart')
def chart():
    return render_template('chart.html')
```

Рисунок 3.13 – Маршрут до графіку

Також, реалізовано завантаження звіту у форматі .docx, це зроблено за допомогою окремого файлу «report_generator», який викликається в основному файлі. Насамперед, підключено бібліотеки для роботи з часом, базою даних та документами (рис. 3.14).

```python
from docx import Document
from docx.shared import Pt
from docx.enum.table import WD_ALIGN_VERTICAL
from docx.enum.text import WD_PARAGRAPH_ALIGNMENT
from datetime import datetime
import sqlite3
```

Рисунок 3.14 – Підключені бібліотек

Після описана логіка формування файлу, а саме: додавання тексту, формування таблиць (рис. 3.15) та аналітичної частини (рис. 3.16).

```python
doc.add_paragraph(' ')
doc.add_heading('💧 Вологість (останні 30 значень)', level=1)
hum_table = doc.add_table(rows=1, cols=2)
hum_table.style = 'Light Grid Accent 1'
hum_table.rows[0].cells[0].text = 'Час'
hum_table.rows[0].cells[1].text = 'Значення (%)'

for ts, val in hum_data:
    row_cells = hum_table.add_row().cells
    row_cells[0].text = datetime.fromisoformat(ts).strftime('%d.%m.%Y %H:%M:
    row_cells[1].text = f"{val:.2f}"
```

Рисунок 3.15 – Додавання таблиці вологості

```
doc.add_paragraph(' ')
doc.add_heading(' Аналітика температури', level=2)
if temp_values:
    doc.add_paragraph(f'• Мінімальна: {min(temp_values):.2f} °C')
    doc.add_paragraph(f'• Максимальна: {max(temp_values):.2f} °C')
    doc.add_paragraph(f'• Середня: {sum(temp_values) / len(temp_values):.2f}
else:
    doc.add_paragraph('Немає даних для аналізу.')
```

Рисунок 3.16 – Додавання аналітичної частини температури

В кінці зберігається файл з назвою report.docx. Та завантажується коли він викликається в основному файлі (рис. 3.17).

```
@app.route('/download_report')
def download_report():
    generate_report()
    return send_file("report.docx", as_attachment=True)
```

Рисунок 3.17 – Маршрут для завантаження звіту

3.3 Інтерфейс користувача

Окрему увагу було приділено створенню зрозумілого та зручного веб-інтерфейсу. Щоб користувач міг швидко переглядати актуальну інформацію в різноманітних форматах, а саме: графіки, таблиці, зрозумілий аналітичний розділ та завантаження звітів. Для розмітки було використано HTML-шаблони, а для стилізації – CSS, з урахуванням сучасного мінімалістичного стилю. Кожний шлях до шаблонів описано в коді «app.py», також для коректної роботі іноді створювалися додаткові маршрути, такі як: @app.route('/chart/data') та @app.route('/chart/data').

Головною сторінкою є «dashboard.html» – «Панель управління», вона є навігаційним меню (рис. 3.18). З неї можна перейти до інших сторінок через кнопки. Кожна з кнопок відповідає за відкриття нової сторінки: графік, таблиця(30с), таблиця(20хв) та аналітична частина.

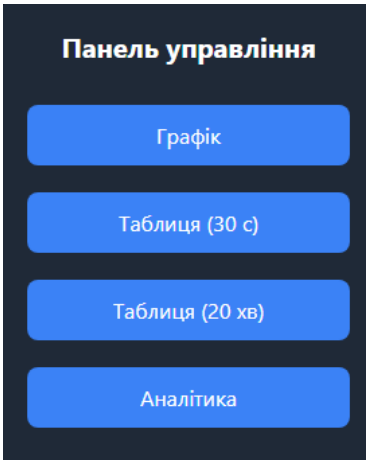


Рисунок 3.18 – «Панель управління»

Також, до «dashboard.html» під’єднано файл «base.html» з описом цих кнопок (рис. 3.19).

```
<div class="container">
  <div class="sidebar">
    <h3>Панель управління</h3>
    <ul>
      <li><a href="/chart">Графік</a></li>
      <li><a href="/table_short">Таблиця (30 с)</a></li>
      <li><a href="/table_long">Таблиця (20 хв)</a></li>
      <li><a href="/analytics">Аналітика</a></li>
    </ul>
  </div>
```

Рисунок 3.19 – Фрагмент коду в base.html

Кожна нова сторінка працює за допомогою окремих html файлів. Так в файлі «chart.html» реалізовано два окремих графіків з температурою та вологістю, які переключаються за допомогою двох кнопок (рис. 3.20).

```
const ctx = document.getElementById(canvasId).getContext('2d');
const chart = new Chart(ctx, {
    type: 'line',
    data: {
        labels: formattedTimestamps,
        datasets: [{
            label: sensor === 'temp' ? 'Температура (°C)' : 'Вологість (%)',
            data: data.values,
            borderColor: sensor === 'temp' ? '#f87171' : '#60a5fa',
            borderWidth: 2,
            fill: false
        }]
    },
```

Рисунок 3.20 – Фрагмент коду в «chart.html»

Таблиця з інтервалом оновлення даних в 30 секунд, також реалізована окремим файлом – «table_short.html». Тут так само є дві кнопки для перемикання між даними з різних датчиків (рис. 3.21). Виводяться в таблицю останні 30 значень.

```
<script>
function loadSensor(sensor) {
  fetch(`/table_short/data?sensor=${sensor}`)
    .then(res => res.text())
    .then(html => {
      document.getElementById('table-container').innerHTML = html;
    });
}

loadSensor('temp');
</script>
```

Рисунок 3.21 – Фрагмент коду в «table_short.html»

Також, ще створено додатковий файл «partial_table.html» з якого регулюється яку саме інформацію додавати до таблиці (рис. 3.22).

```
<table>
  <tr><th>Час</th><th>Значення</th></tr>
  {% for row in rows %}
    <tr><td>{{ row[0] }}</td><td>{{ row[1] }}</td></tr>
  {% endfor %}
</table>
```

Рисунок 3.22 – Фрагмент коду в «partial_table.html»

Наступною є таблиця в яку зберігаються дані з інтервалом 20 хвилин. Її реалізовано за допомогою «table_long.html» (рис. 3.23). Тут створюється дві окремі таблиці для зберігання даних з різних датчиків.

```
<h3>
  {% if sensor == 'temp' %}🌡 Температура
  {% elif sensor == 'hum' %}💧 Вологість
  {% else %}📟 {{ sensor|capitalize }}
  {% endif %}
</h3>
<table border="1">
  <tr><th>Час</th><th>Значення</th></tr>
  {% for row in rows %}
  <tr><td>{{ row[1] }}</td><td>{{ row[3] }}</td></tr>
  {% endfor %}
</table>
```

Рисунок 3.23 – Фрагмент коду «table_long.html»

Остання кнопка «Аналітика» – це аналітична частина, де в окремих осередках показується інформація щодо різних датчиків, такі як: середнє значення, максимальне та мінімальне. Також показується кількість вимірів. Реалізовану кнопку завантаження звіту в .docx форматі, в ньому відображено таблиці з інтервалом в 30 секунд та аналітична частина. Логіку цієї сторінки описано в «analytics.html» (рис. 3.24).

```
  <div style="flex: 1; min-width: 280px;">
    <h3 style="color:#2563eb; text-align:center;">💧 Вологість</h3>
    {% for item in hum_analytics %}
      <div class="analytics-card" style="margin-bottom: 20px;">
        <h4>{{ item.title }}</h4>
        <p>{{ item.value }}</p>
      </div>
    {% endfor %}
  </div>

</div>

<a href="/download_report" class="download-btn">⬇ Завантажити звіт</a>
```

Рисунок 3.24 – Фрагмент коду «analytics.html»

В результаті маємо зрозумілий та зручний веб-інтерфейс для перегляду даних в режимі реального часу.

3.4 Надсилання повідомлень за допомогою telegram bot`a

Окремою функцією системи є надсилання повідомлень при перевищенні максимально допустимих значень, які може виміряти датчик. Ця функція була реалізована за допомогою застосунку «Telegram», для надсилання повідомлень був створений чат-бот.

Першим кроком є створення бота в застосунку, для цього потрібно перейти до офіціального бота «BotFather» з адресою «@BotFather». Цей бот дозволяє створювати нових та редагувати вже створених. Для початку роботи потрібно відправити повідомлення з таким текстом: «/start». У відповідь надходить повідомлення з командами (рис. 3.25).

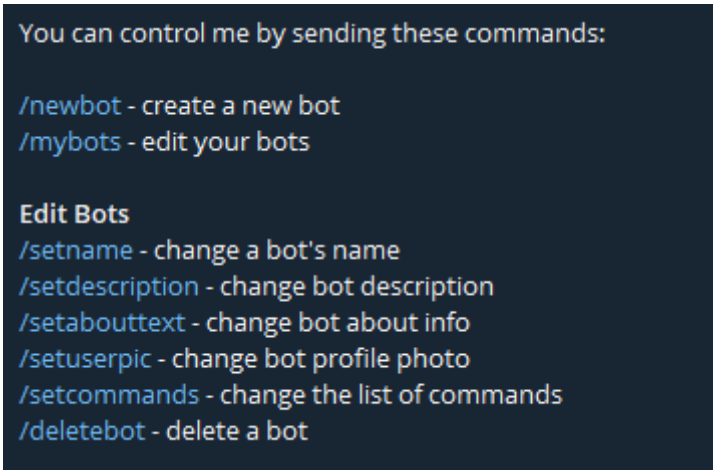


Рисунок 3.25 – Повідомлення з командами

Далі, потрібно відправити команду /newbot, у відповідь надійде повідомлення з проханням обрати назву та юзернейм. Мною було обрано назву «SensorBot» та адресу «mysensor213_bot». У відповідь надходить повідомлення з підтвердженням реєстрації бота та його токен. Токен це унікальний ідентифікатор, який потрібен для доступу до API Telegram і дозволяє програмі звертатися до серверів Telegram від імені цього бота. Коли серверна частина надсилає повідомлення через Telegram цей токен додається до запиту. Він має вигляд довгого рядка з літер та цифр.

Складається з двох частин розділених двокрапкою. Перша частина є id-ідентифікатором бота, частина після двокрапки є секретною частиною, яка і використовується в авторизації. Наприклад:

123456789 : AAHjKxJpB2pR8VpM2NczQj1TULcFDJXuY7A

Далі було вирішено створити групу, до якої можуть приєднуватися користувачі котрим повинно надходити повідомлення. Для цього в налаштуваннях обираємо «New Group» (рис. 3.26).

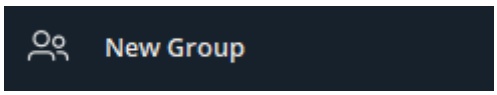


Рисунок 3.26 – Кнопка в налаштуваннях

Після вказуємо тип та назву групи. Було створено таку групу (рис. 3.27).

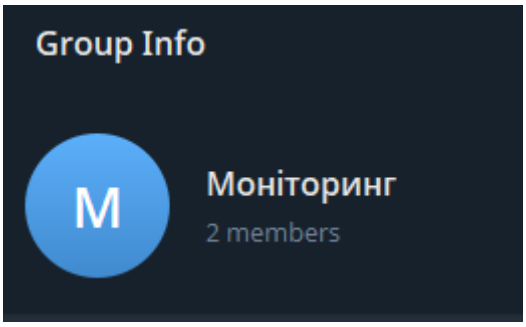


Рисунок 3.27 – Група в телеграм

Наступним кроком є додавання бота до цієї групи, для цього через меню додавання учасників приєднуємо бота (рис. 3.28), йому потрібно видати роль адміна в групі.

Рисунок 3.28 – Меню додавання учасників

Тепер потрібно дізнатися id цієї групи, щоб повідомлення надходили до неї. Це можна зробити за допомогою сайту:

«https://api.telegram.org/bot<токен>/getUpdates»

Для цього потрібно зі свого аккаунту надіслати повідомлення до групи, зміст повідомлення неважливий. На цьому сайті буде повідомлення про те, звідки надходили повідомлення куди приєднаний бот. Серед всієї інформації потрібно знайти повідомлення, яке має такий зміст (рис. 3.29).

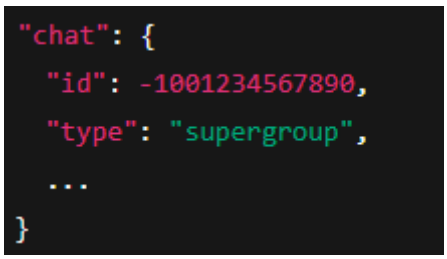


Рисунок 3.29 – Пошук id-групи

Нас цікавить набір цифр зі знаком «-», зазвичай починається на «-100». Це і є id-адресою групи яка нас цікавить. Після цього, коли вся потрібна інформація для взаємодії з ботом є, можна переходити до програмної частини. На початку коду сервера потрібно вказати токен бота та id групи (рис. 3.30).

```
BOT_TOKEN = '76  46  75:AAF3     Inq4H72ndT1rW_          '
CHAT_ID = '-1002    925 3'
```

Рисунок 3.30 – Вказані токен та id

Після, в функцію «def receive_data» потрібно додати такий код (рис. 3.31).

```
try:
    if float(temperature) > 30.0:
        msg = f"🚨 Увага! Температура перевищила норму: {temperature}°C"
        send_telegram_message(msg)
except ValueError:
    print("Не вдалося перетворити температуру у число.")

return {'status': 'received'}
```

Рисунок 3.31 – Логіка надсилання повідомлень

Максимально допустиме значення вказуємо у рядку: if float(temperature) > 30.0:

Як результат отримаємо повідомлення від бота до групи (рис. 3.32).

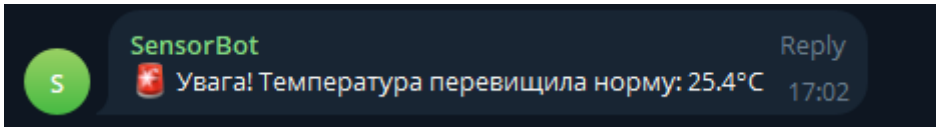


Рисунок 3.32 – Повідомлення від бота

4 ЕКСПЕРЕМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РОБОТИ СИСТЕМИ

4.1 Демонстрація роботи реалізованої системи

Коли на ESP32 подається живлення, він починає збирати дані за допомогою датчика DHT22. Переконатися у правильності збору даних можна через «Serial Monitor» у середовищі Arduino IDE (рис. 4.1).

Рисунок 4.1 – Повідомлення в «Serial Monitor»

Після цього вмикаємо сервер «app.py», при правильній роботі буде таке повідомлення в консолі (рис. 4.2).

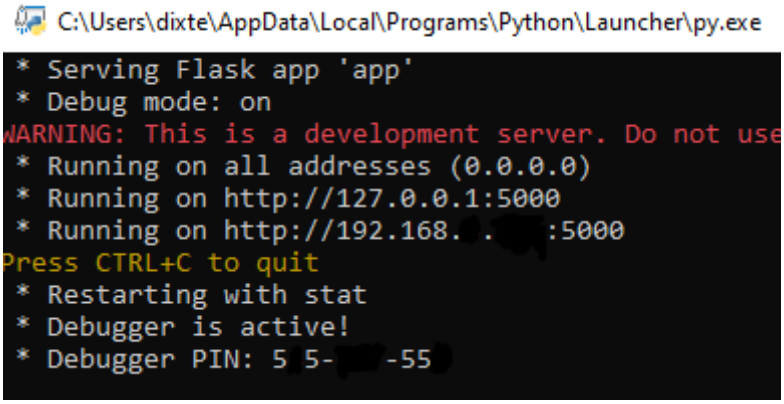


Рисунок 4.2 –Повідомлення в консолі

Тепер при вірній передачі даних з ESP32 до серверу HTTP відповідь буде 200 (рис. 4.3).

Рисунок 4.3 – Правильне надсилання даних на сервер

Тепер дані вдало надходять на сервер. За допомогою веб-інтерфейсу можна переглядати інформація у вигляді таблиць, графіків та аналітику у режимі реального часу. Далі, показано графіки температури (рис. 4.4) та вологості (рис. 4.5), які автоматично оновлюються.

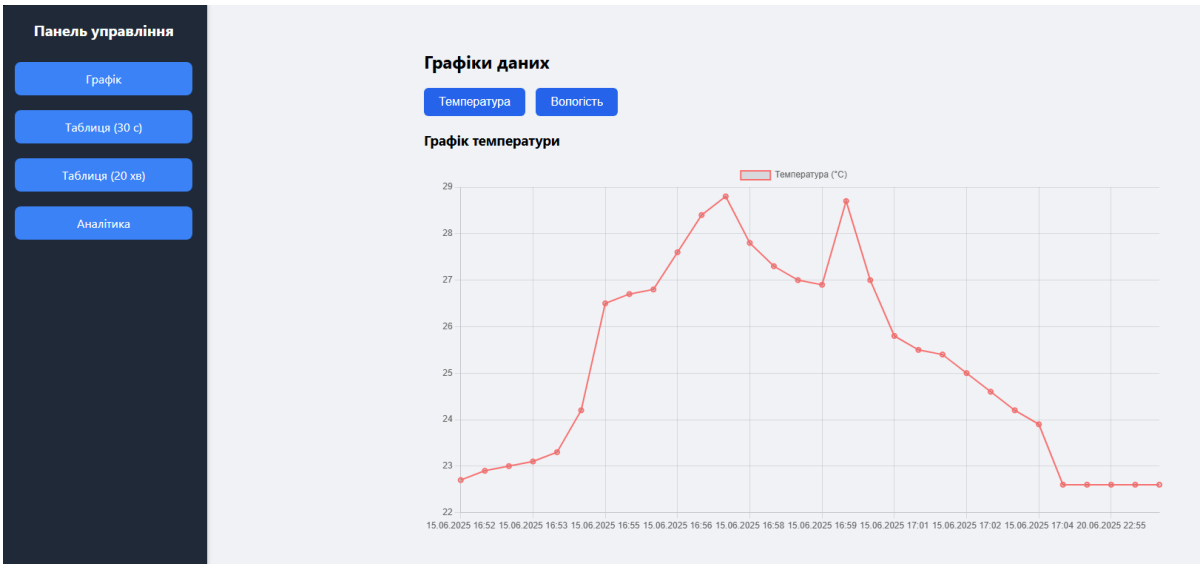


Рисунок 4.4 – Графік температури

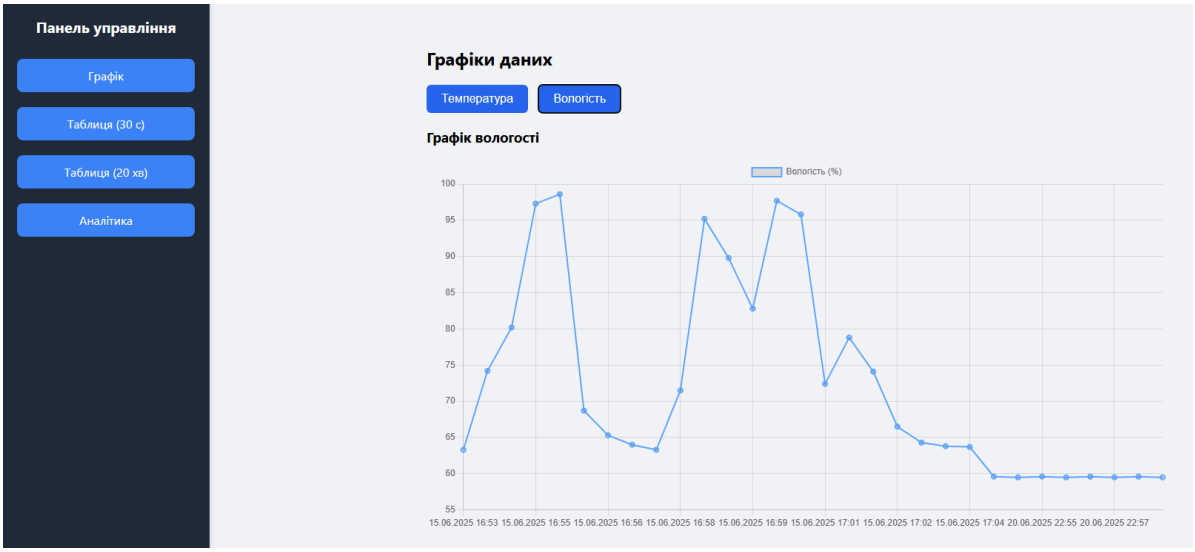


Рисунок 4.5 – Графік вологості

Наступним можна переглянути таблиці: з інтервалом оновлення в 30 секунд (рис. 4.6) та 20 хвилин (рис. 4.7).

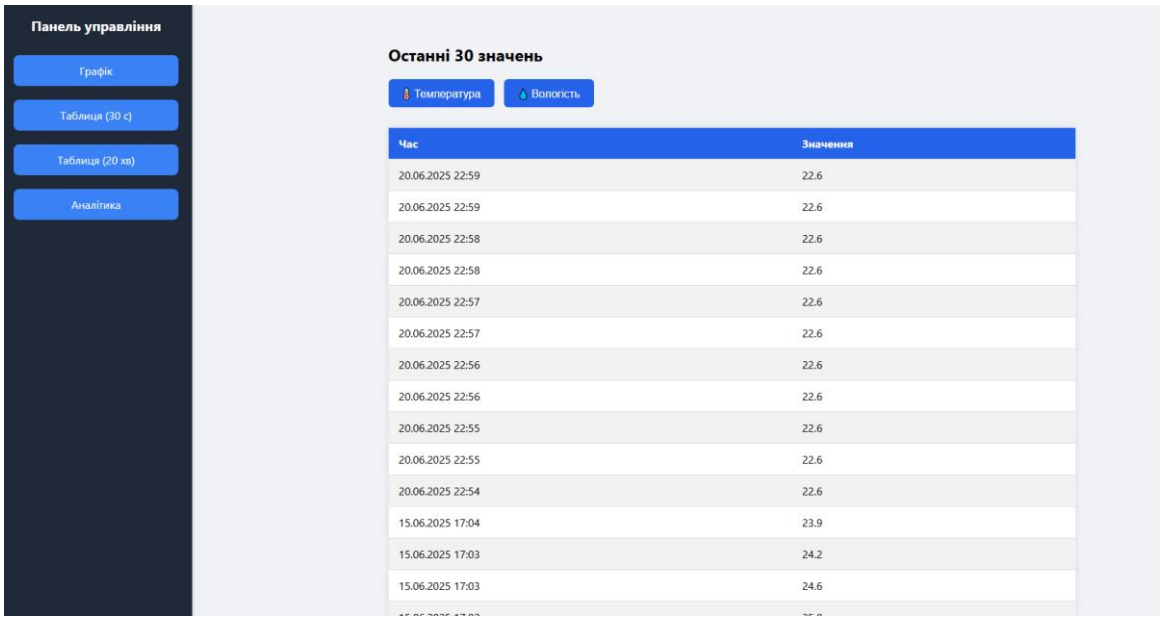


Рисунок 4.6 – Таблиця температури(30 секунд)

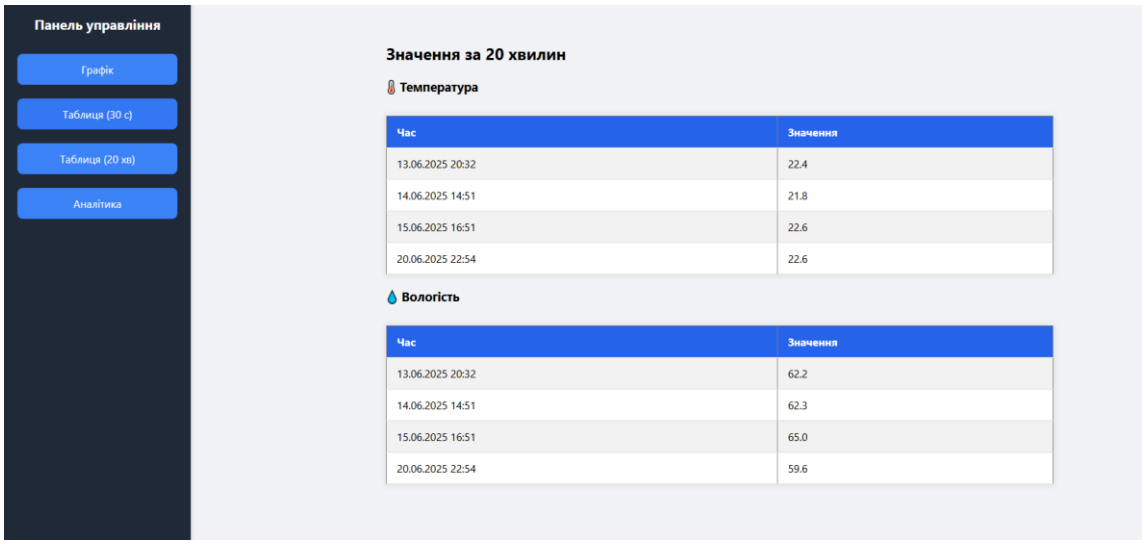


Рисунок 4.7 – Таблиці(20 хвилин)

Остання кнопка – «Аналітика», дозволяє продивитися інформацію в окремих секціях, а саме: максимальне, мінімальне та середнє значення (рис. 4.8). Також є загальна кількість вимірів та можливість завантаження звіту (рис. 4.9).

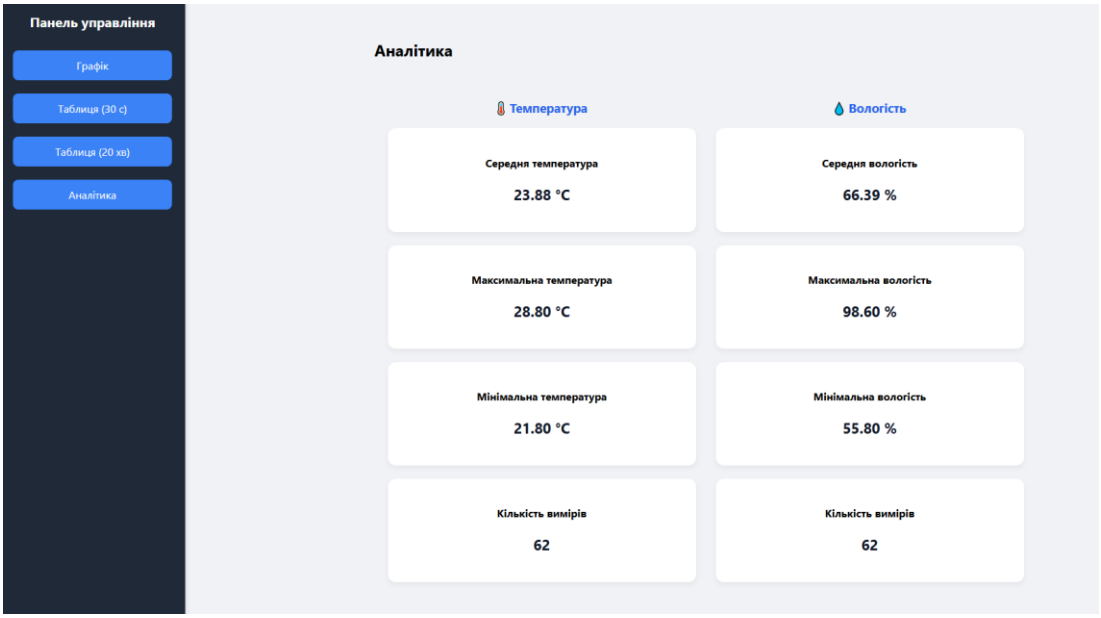


Рисунок 4.8 – Аналітична частина

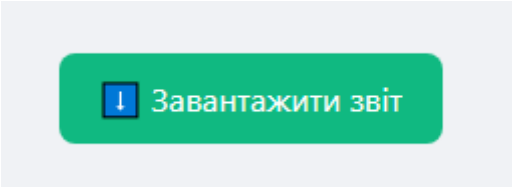
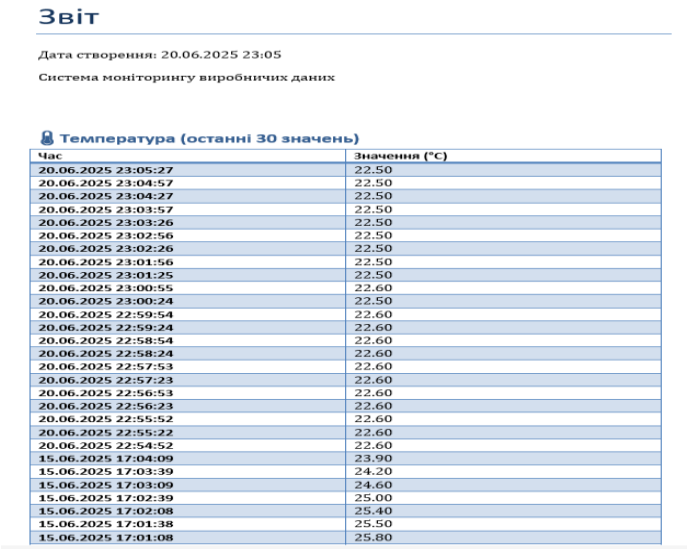


Рисунок 4.9 – Кнопка завантаження звіту

Звіт завантажується у форматі .docx та має вигляд файлу з двома таблицями (рис. 4.10) та окремою аналітичною частиною (рис. 4.11).



Звіт

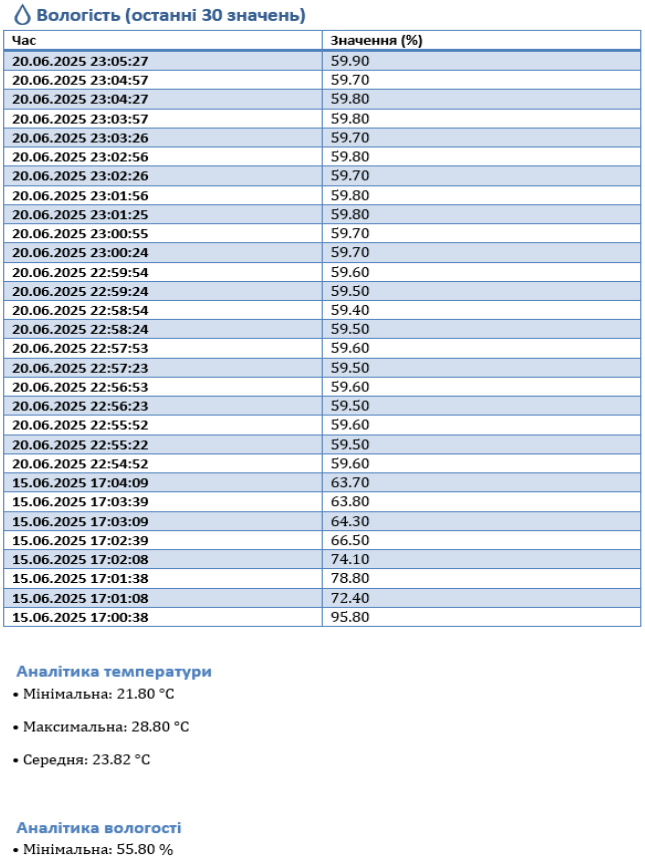
Дата створення: 20.06.2025 23:05

Система моніторингу виробничих даних

Температура (останні 30 значень)

Час	Значення (°C)
20.06.2025 23:05:27	22.50
20.06.2025 23:04:57	22.50
20.06.2025 23:04:27	22.50
20.06.2025 23:03:57	22.50
20.06.2025 23:03:26	22.50
20.06.2025 23:02:56	22.50
20.06.2025 23:02:26	22.50
20.06.2025 23:01:56	22.50
20.06.2025 23:01:25	22.50
20.06.2025 23:00:55	22.60
20.06.2025 23:00:24	22.50
20.06.2025 22:59:54	22.60
20.06.2025 22:59:24	22.60
20.06.2025 22:58:54	22.60
20.06.2025 22:58:24	22.60
20.06.2025 22:57:53	22.60
20.06.2025 22:57:23	22.60
20.06.2025 22:56:53	22.60
20.06.2025 22:56:23	22.60
20.06.2025 22:55:52	22.60
20.06.2025 22:55:22	22.60
20.06.2025 22:54:52	22.60
15.06.2025 17:04:09	23.90
15.06.2025 17:03:39	24.20
15.06.2025 17:03:09	24.60
15.06.2025 17:02:39	25.00
15.06.2025 17:02:08	25.40
15.06.2025 17:01:38	25.50
15.06.2025 17:01:08	25.80

Рисунок 4.10 – Вигляд першої сторінки



Вологість (останні 30 значень)

Час	Значення (%)
20.06.2025 23:05:27	59.90
20.06.2025 23:04:57	59.70
20.06.2025 23:04:27	59.80
20.06.2025 23:03:57	59.80
20.06.2025 23:03:26	59.70
20.06.2025 23:02:56	59.80
20.06.2025 23:02:26	59.70
20.06.2025 23:01:56	59.80
20.06.2025 23:01:25	59.80
20.06.2025 23:00:55	59.70
20.06.2025 23:00:24	59.70
20.06.2025 22:59:54	59.60
20.06.2025 22:59:24	59.50
20.06.2025 22:58:54	59.40
20.06.2025 22:58:24	59.50
20.06.2025 22:57:53	59.60
20.06.2025 22:57:23	59.50
20.06.2025 22:56:53	59.60
20.06.2025 22:56:23	59.50
20.06.2025 22:55:52	59.60
20.06.2025 22:55:22	59.50
20.06.2025 22:54:52	59.60
15.06.2025 17:04:09	63.70
15.06.2025 17:03:39	63.80
15.06.2025 17:03:09	64.30
15.06.2025 17:02:39	66.50
15.06.2025 17:02:08	74.10
15.06.2025 17:01:38	78.80
15.06.2025 17:01:08	72.40
15.06.2025 17:00:38	95.80

Аналітика температури

- Мінімальна: 21.80 °C
- Максимальна: 28.80 °C
- Середня: 23.82 °C

Аналітика вологості

- Мінімальна: 55.80 %

Рисунок 4.11 – Вигляд другої сторінки звіту

Отже, в цьому розділі було продемонстровано роботу системи у повному обсязі. Від збору даних до їх збереження та аналізу.

4.2 Охорона праці

Охорона праці це система організаційно-технічних, правових, лікувальних та профілактичних заходів для збереження життя, здоров’я та працездатності у процесі трудової діяльності людини. Передбачає створення безпечних умов для запобігання травматизму та захворювань.

Під час виконання роботи потенційно небезпечним фактором була робота з електричною схемою (напруга 3,3 В).

При роботі за комп’ютером потрібно дотримуватися норм роботи з ПК. Потрібно пам’ятати про вентиляцію приміщення, освітленням, ергономікою

робочого місця та відпочинку. Освітленість становила 400 лк, що відповідає вимогам для лабораторних приміщень (норма – не менше 300 лк). Рівень шуму також не перевищував норму (не вище 65 дБ), був на рівні 40 – 50 дБ.

Для забезпечення безпечних умов праці, необхідно розрахувати об’єм повітрообміну. Вимірюється за формулою:

$$L = N \cdot q, \qquad (2.1)$$

де L – об’єм повітрообміну (м³/год);

N – кількість осіб у приміщенні;

q – норма повітрообміну на одну особу (60 м³ / год).

Кількість людей у приміщенні – N = 1.

$$L = 1 \cdot 60 = 60 \text{ м}^3 / \text{год}.$$

Тобто, потрібно забезпечити об’єм повітрообміну на рівні 60 м³ / год.

Для запобігання виникненню пожеж потрібно вжити комплексні заходи. Важливо забезпечити належний розрахунок електричних навантажень, використання запобіжників. Потрібно мати засоби пожежогасіння, включаючи вогнегасники на робочих місцях та, за необхідності, автоматичні системи пожежогасіння у приміщеннях.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було спроектовано та реалізовано повноцінну систему моніторингу виробничих параметрів на приладобудівному виробництві. Основою став мікроконтролер ESP32, датчик DHT22, серверна частини була реалізована за допомогою мікрофреймворку Flask та бази даних SQLite. Розроблена система дозволяє збирати, зберігати, обробляти та візуалізувати показники температури та вологості в режимі реального часу. Продивитися оброблені дані можна через зручний вебінтерфейс. Приділено увагу зручності та зрозумілій подачі цих даних користувачеві, було зроблено кілька варіантів перегляду інформації: таблиць, графіків та аналітична панель.

Проаналізовано предметну область, існуючі рішення та системи моніторингу. Це дозволило дізнатися потрібну інформацію та структуровано підійти до розробки. Яку роль відіграють системи моніторингу у виробництві та які задачі вирішують. Що саме входить до систем моніторингу, які компоненти та програмне забезпечення використовується.

Обрано такі виробничі дані: температура та вологість. Ці показники безпосередньо впливають на стабільність обладнання , якість продукції та безпеку на виробництві, тому їхній вибір був найбільш доцільним для демонстрації роботи системи. Також, розроблена система моніторингу є доволі гнучкою, та за потреби можна адаптувати вимірювані параметри зважаючи на потреби конкретного виробництва.

Обрано програмні та технічні рішення. Як основу макету було використано мікроконтролер ESP32 та датчик температури та вологості DHT22. Вибір будувався на стабільній та зручній взаємодії програм та компонентів.

Розроблено програмне забезпечення, яке дозволило реалізувати задуману систему. Було використано мікрофреймворк Flask, базу даних SQLite, для сповіщень Telegram Bot API та стандартні вебтехнології. Для структури сторінок

було обрано HTML, для оформлення CSS та JavaScript для оновлення графіків та перезавантаження сторінок.

В результаті, досягнуто мети роботи, розроблено модуль для моніторингу виробничих даних на підприємстві задля підвищення ефективності моніторінгу та аналізу технологічних процесів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. Структура та правила оформлення. Введ. 2015-06-22. К. Держстандарт України,2017. – 29 с.

2. Методичні вказівки з підготовки кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології освітньої програми «Системна інженерія» / Упоряд.: І.Ш. Невлюдов, О.М. Цимбал, О.В. Токарєва, А.І. Бронніков. Харків: ХНУРЕ, 2023. 65 с.

3. Lyashenko, V., Abu-Jassar, A.T., Yevsieiev, V., Maksymova, S. Automated Monitoring and Visualization System in Production, Int. Res. J. Multidiscip. Technovation, 5(6) 2023 09-18.

4. Невлюдов, І.Ш. Автоматичне управління технологічними об'єктами [Текст]: підручник/І.Ш. Невлюдов, О.В.Токарєва. – Харків: ХНУРЕ, 2018.–190 с.

5. Nevliudov, I., & et al.. (2020). Method of Algorithms for Cyber-Physical Production Systems Functioning Synthesis. International Journal of Emerging Trends in Engineering Research, 8(10), 7465-7473

6. Danylenko M. M. Comparative analysis of modern SCADA packages for production automation / M. M. Danylenko, S. V. Sotnik // International Journal of Academic Engineering Research (IJAER). – 2025. – Vol. 9. – 2. – pp. 26-34.

7. Sotnik S. Overview of Modern Accelerometers // International Journal of Engineering and Information Systems (IJEAIS) / S. Sotnik, V. Lyashenko. – 2022. – Vol. 6, Issue 1. – pp. 57-64. 7. Tan X. et al. Battery Management System and Its Applications. – John Wiley & Sons, 2022 – 389 p.

8. Khalimonov Y. I. Monitoring and optimising conditions in production environment / Y. I. Khalimonov, I. K. Sezonova, S. V. Sotnik // Інформаційні технології і автоматизація – 2024 : матеріали XVII міжнародної науково-практичної конференції, 31 жовтня-1 листопада 2024 р. – Одеса : Видавництво ОНТУ, 2024 р. – С. 256-258.

9. Невлюдов І. Ш. Виробничі процеси та обладнання об’єктів автоматизації: Підручник для студентів вищих навчальних закладів / І. Ш. Невлюдов та інш. Кривий Ріг: Криворізький коледж НАУ, 2017 р. – 444 с.

10. ESP-WROOM-32 Datasheet [Електронний ресурс]: espressif.com – Режим доступу:https://espressif.com/sites/default/files/documentation/esp_wroom_32_datasheet_en.pdf

11. Автоматизація та комп’ютерно-інтегровані технології освітньої програми «Системна інженерія» / Упоряд.: І.Ш. Невлюдов, О.М. Цимбал, О.В.Токарєва, А.І. Бронніков. Харків: ХНУРЕ, 2022. 66 с.

12. SQLite Documentation // SQLite [Електронний ресурс]: Офіційна документація. – Режим доступу: https://sqlite.org/index.html

13. Flask Documentation // Pallets Projects [Електронний ресурс]: Офіційна документація. – Режим доступу: https://flask.palletsprojects.com/en/stable/

14. DHT22 Temperature-Humidity Sensor Datasheet // SparkFun Electronics. [Електронний ресурс]: Офіційна документація. – Режим доступу: https://cdn.sparkfun.com/assets/f/7/d/9/c/DHT22.pdf.

15. Комплекс навчально-методичного забезпечення навчальної дисципліни «Безпека праці в індустрії ІТ-технологій» підготовки освітнього рівня бакалавр усіх спеціальностей та усіх напрямів університету [http://catalogue.nure.ua/knmz] / ХНУРЕ; розроб.: Т. Є. Стиценко, Г. В. Пронюк, Н. М. Сердюк. – Харків, 2017. – 122 с.

16. Теорія автоматичного управління (збірник задач) [Текст]: навч.посіб. для студентів спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології / І.Ш. Невлюдов, О.В. Токарєва; Харків. нац. ун-т радіоелектроніки. - Харків: Панов А.М., 2020. – 240 с.