

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет інформаційної безпеки та електронних комунікацій
(повне найменування факультету)

Кафедра інформаційних технологій електронних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ ПІДХОДІВ МОНІТОРИНГУ
ЗАЛИШКІВ ПРОДУКТІВ В РЕСТОРАНІ
RESEARCH AND IMPLEMENTATION OF APPROACHES TO MONITORING
FOOD RESIDUES IN A RESTAURANT

Виконав: студент(ка) 2 курсу, групи БК-612м
Спеціальності 151 Автоматизація та
комп'ютерно-інтегровані технології
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Автоматизація, мехатроніка та робототехніка

ПОЛОВІНЧУК Є.В.

(ПРІЗВИЩЕ та ініціали)

Керівник МИРОНОВА Н.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент КОРОТУН А.В.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет інформаційної безпеки та електронних комунікацій
Кафедра інформаційних технологій електронних засобів
Ступінь вищої освіти магістр
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і найменування)
Освітня програма (спеціалізація) Автоматизація, мехатроніка та робототехніка
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

В.о.завідувача кафедри ІТЕЗ, к.т.н, доц.
О.Ю. МАЛИЙ
“ ” 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПОЛОВІНЧУК Євгеній Вікторович

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та реалізація підходів моніторингу залишків продуктів в ресторані. Research and implementation of approaches to monitoring food residues in a restaurant

керівник проєкту (роботи) к.т.н., доц., МИРОНОВА Наталя Олексіївна,

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові,)

затверджені наказом закладу вищої освіти від “23” квітня 2024 року № 162

2. Строк подання студентом проєкту (роботи) 18 червня 2024 року

3. Вихідні дані до проєкту (роботи) програма повинна забезпечувати обробку та відображення даних в браузері чи мобільному застосунку

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ, 1 Аналіз предметної області та постановка задачі,, 2 Дослідження існуючих рішень та технологій для систем моніторингу залишків продуктів, 3 Проектування та реалізація системи, 4 Експериментальне дослідження розробленої системи, Висновки, Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) слайди презентації

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
Розділи 1-4	МИРОНОВА Н.О., доцент		
Нормоконтроль	ПОСПЕСЬВА І. Є., ст.викладач		

КАЛЕНДАРНИЙ ПЛАН

Студент(ка) _____ **Євгеній ПОЛОВІНЧУК**
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи) _____ **Наталя МИРОНОВА**
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
82 с., 14 табл., 10 рис., 1 дод., 29 джерел.

СИСТЕМА АВТОМАТИЗОВАНОГО МОНІТОРИНГУ, ЗАЛИШКИ
ПРОДУКТІВ, ТЕХНОЛОГІЯ ІНТЕРНЕТ РЕЧЕЙ, ТЕНЗОДАТЧИК,
МІКРОКОНТРОЛЕР, КЛАС, БАЗА ДАНИХ

Об'єкт дослідження – програмні та апаратні компоненти систем моніторингу залишків продуктів.

Предмет дослідження – система моніторингу залишків продуктів на основі технології Інтернету речей для ресторанної індустрії.

Мета роботи – розробка прототипу системи моніторингу залишками продуктів з використанням технології Інтернету речей, що дозволить автоматизувати процеси відстеження залишків продуктів з високою точністю.

Для реалізації програмної частини використовувався MERN-стек (MongoDB, Express.js, React, Node.js), мова програмування C++ та середовища розробки – Visual Studio Code та Arduino IDE. Мікроконтролер Arduino UNO з модулем ESP8266 та тензодатчиками використовувався для збору даних про вагу продукції та передачі їх на сервер. Сервер на базі Express.js приймає дані, перевіряє їх та зберігає у MongoDB.

В результаті були розроблені наступні моделі бази даних: Product, MicroController, User, Token. Система дозволяє автоматизувати процес моніторингу залишків продуктів та забезпечує можливість додавання нових мікроконтролерів і тензодатчиків для масштабування.

ABSTRACT

Explanatory note to the master's thesis: 82 pages, 14 tables, 10 figures, 1 appendix, 29 sources.

AUTOMATED MONITORING SYSTEM, PRODUCT RESIDUES, INTERNET OF THINGS TECHNOLOGY, LOAD CELL, MICROCONTROLLER, CLASS, DATABASE.

The object of research is the software and hardware components for product residue monitoring systems.

The subject of research is a monitoring system for product residuals based on the Internet of Things technology for the restaurant industry.

The purpose of the study is development of a prototype system for monitoring product residuals using the Internet of Things technology, which will automate the process of tracking product residuals with high accuracy.

To implement the software part, the MERN stack (MongoDB, Express.js, React, Node.js), the C++ programming language and development environments – Visual Studio Code and Arduino IDE were used. An Arduino UNO microcontroller with an ESP8266 module and load cells was used to collect data on the weight of goods and transmit it to the server. The Express.js-based server receives the data, verifies it, and stores it in MongoDB.

As a result, the following database models: Product, MicroController, User, Token were developed. The system automates the process of monitoring product residuals and provides the ability to add new microcontrollers and load cells for scalability.

ЗМІСТ

	С.
ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК	8
ВСТУП	9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ	11
1.1 Історія виникнення IoT	11
1.2 Основи IoT та їх застосування в ресторанній сфері	12
1.3 Проблеми управління запасами в ресторанах	13
1.4 Традиційні підходи до управління залишками продуктів у ресторанах ...	16
1.5 Переваги впровадження IoT-технологій у логістику ресторанів	18
1.6 Оптимальна модель IoT-системи для управління запасами	21
1.7 Висновки до розділу	22
2 ДОСЛІДЖЕННЯ ІСНУЮЧИХ РІШЕНЬ ТА ТЕХНОЛОГІЙ ДЛЯ СИСТЕМ МОНІТОРИНГУ ЗАЛИШКІВ ПРОДУКТІВ.....	24
2.1 Огляд готових рішень для систем моніторингу залишків продуктів	24
2.2 Аналіз стеків технологій для створення систем моніторингу залишків ...	30
2.3 Аналіз інструментів та компонентів для збору даних про вагу	38
2.4 Огляд та аналіз різних технологій передачі даних	46
2.5 Аналіз та огляд різних заходів безпеки IoT приладам	50
3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ	52
3.1 Загальна архітектура системи	52
3.2 Детальна схема підключення мікроконтролера до модулів	55
3.3 Проектування серверної частини	58
3.4 Опис моделей бази даних	60
3.5 Загальний огляд клієнтської частини	62
3.6 Висновки до розділу 3	64
4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РОЗРОБЛЕНОЇ СИСТЕМИ....	65
4.1 Тестування апаратної частини	65
4.2 Тестування серверної частини	65
4.3 Тестування клієнтської частини	66

4.4 Результати тестування	66
4.5 Можливості розширення та покращення системи.....	68
ВИСНОВКИ.....	71
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	73
Додаток А Текст програми.....	79

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
DB	– Database;
IDE	– Integrated Development Environment;
IOT	– Internet of Things;
HTTP	– Hypertext Transfer Protocol;
WS	– WebSocket;
USB	– Universal Serial Bus;
АЦП	– аналого-цифровий перетворювач;
МК	– мікроконтролер;
ЦАП	– цифро-аналоговий перетворювач;

ВСТУП

Актуальність теми. У сучасному світі управління запасами в ресторанах стає все більш важливим через необхідність ефективного використання ресурсів та мінімізації втрат. Інноваційні технології, такі як Інтернет речей (IoT), дозволяють автоматизувати процеси моніторингу та управління запасами, що значно покращує точність обліку та знижує ризики помилок. В умовах динамічного ринку ресторанного бізнесу автоматизація стає критично важливою для забезпечення конкурентоспроможності та підвищення якості обслуговування клієнтів.

Мета роботи. Основна мета цієї дипломної роботи полягає у створенні прототипу системи моніторингу залишків продуктів в ресторані з використанням технології Інтернету речей. Це дозволить автоматизувати процеси відстеження залишків продуктів з високою точністю.

Завдання дослідження. Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати існуючі технології реалізації систем моніторингу залишків продуктів;
- вивчити апаратні рішення на основі IoT;
- розробити проект та реалізувати прототип системи моніторингу залишків продуктів;
- провести експериментальну перевірку розробленої системи.

Об'єкт дослідження – програмні та апаратні компоненти систем моніторингу залишків продуктів.

Предмет дослідження – система моніторингу залишків продуктів на основі технології Інтернету речей для ресторанної індустрії.

Методи дослідження. Для реалізації прототипу системи моніторингу залишків продуктів використовувались технології Інтернету речей, мови програмування C++ та JavaScript.

Наукова новизна. Наукова новизна цієї роботи полягає у дослідженні методів проектування систем моніторингу залишків продуктів з використанням технології Інтернету речей, що сприяє автоматизації процесів моніторингу залишків продуктів в ресторані, знижує кількість помилок та підвищує точність даних.

Практична цінність. Практична цінність цієї роботи полягає у створенні прототипу системи моніторингу залишками продукції, що дозволяє автоматизувати процеси відстеження залишків продуктів з високою точністю.

За результатами роботи були подані тези на науково-практичну конференцію викладачів, науковців, молодих учених, аспірантів та студентів «Тиждень науки-2024».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Історія виникнення IoT

Провісником ідеї створення інтернету речей був Нікола Тесла, який висловлювався на цю тему ще в 1926 р. На його думку, радіо, удосконалюючись, поступово перетвориться на "величезний мозок", а всі інші інструменти та пристрої під'єднають до нього, вони стануть настільки компактними, що помістяться в кишеню.

Приблизно це і вийшло. Мільйони ПК та інших пристроїв за допомогою інтернету об'єднані у величезну мережу. "Дрібні інструменти" - це, в даному випадку, планшети і смартфони. Інтернетом речей, або Internet of Thing (IoT), називають загальносвітову обчислювальну мережу, до якої входять фізичні об'єкти різних типів, що взаємодіють один з одним і навколишнім світом.

Автором першої "інтернет-речі" був Джон Ромкі, який входив до числа розробників і винахідників TCP/IP. У 1990 р. йому вдалося під'єднати звичайний кухонний тостер до ПК і змусити цей прилад вмикатися і вимикатися за командою, відправленою з комп'ютера (Get і Set). Це дало змогу керувати тостером дистанційно і навіть програмувати його на автономну роботу. А ось термін був придуманий тільки в 1999 р. Однак особливо не використовувався, оскільки до 2010-х рр. у галузі інтернету речей не було ніяких особливих проривів, вона не розвивалася. Що загалом не дивно: на той час тільки настільні ПК, смартфони, ноутбуки та сервери мали всі потрібні інтерфейси, щоб законнектитися з інтернетом, і достатню обчислювальну потужність. Та й не було особливої потреби в комп'ютеризації побутової техніки та під'єднанні її до інтернету - це тільки зробило б її дорожчою. У міру того, як поширювалися бездротові технології, мікросхеми дешевшали і світ повсюдно глобалізувався, про концепцію інтернету речей почали згадувати дедалі частіше. Мікропроцесори ARM, чия енергоефективність перевершила процесори для десктопів, ставали дедалі

популярнішими. А головним зрушенням стало те, що великі корпорації почали втілювати цілком конкретні проекти в цій галузі.

1.2 Основи IoT та їх застосування в ресторанній сфері

IoT сприяє ефективному використанню ресурсів, особливо енергії, за допомогою інтелектуального моніторингу та управління споживанням. Це не тільки зменшує витрати, але й сприяє сталому розвитку, зменшуючи викиди вуглецю та заохочуючи екологічно чисті ініціативи.

Інтелектуальні системи Інтернету речей можуть адаптуватися до потреб та уподобань користувачів, надавати персоналізовані послуги та покращувати взаємодію з користувачами. Це досягається шляхом аналізу поведінки та уподобань користувача. У галузях і на підприємствах IoT може допомогти оптимізувати процеси, підвищити продуктивність і знизити виробничі витрати. За допомогою інтелектуальних систем ви можете автоматизувати повсякденні завдання, підвищити продуктивність і знизити ймовірність людських помилок. IoT також відкриває двері для нових бізнес-моделей та відкриває можливості для інноваційних стартапів та компаній. Він охоплює не тільки продукти та послуги, безпосередньо пов'язані з IoT, але й широкий спектр додаткових послуг, таких як аналіз даних, безпека та технічна підтримка. Але, незважаючи на багато переваг, розгортання Інтернету речей має проблеми, особливо проблеми безпеки, конфіденційності, стандартизації та відповідності. Це вимагає постійних зусиль для розробки стандартів, заходів безпеки та політики, що регулюють цю сферу. Загалом, IoT - це динамічна технологія, яка може впливати на різні аспекти нашого повсякденного життя та ділової діяльності та може кардинально змінити спосіб використання технологій. У той же час, щоб повністю реалізувати потенціал Інтернету речей, необхідно вирішити безліч проблем, особливо в області безпеки, конфіденційності та сумісності.

1.3 Проблеми управління запасами в ресторанах

Автоматичне відстеження запасів. Питання точного обліку запасів в сучасних ресторанах-одне з найважливіших завдань. Більшість інгредієнтів розраховуються за вагою, а не за масою, тому їх важко точно виміряти. Традиційні методи, засновані на ручних розрахунках або візуальних оцінках, часто помилкові і схильні до людських помилок. Наприклад, під час інвентаризації часто виникають ситуації, коли вага виробу оцінюється "на око", що призводить до значних відхилень від реальних даних. З впровадженням Інтернету речей стане можливим автоматизувати цей процес за допомогою вбудованих датчиків, які точно вимірюють вагу продукту і передають ці дані в централізовану систему управління в режимі реального часу. Це підвищує точність бухгалтерського обліку, знижує ризик помилок і дозволяє швидко реагувати на зміни в інвентарі.

Мінімізація втрат та оптимізація використання продуктів. Однією з найбільших проблем ресторанної індустрії є нераціональне використання інгредієнтів, що призводить до значних втрат. Часто шеф-кухар не може точно визначити необхідну кількість продукту, що призводить до надмірного вживання або, навпаки, до його нестачі. Впровадження Інтернету речей дозволяє ефективніше керувати запасами, аналізуючи дані про використання кожного компонента. Інтелектуальна система може прогнозувати потреби на основі історичних даних і поточного споживання, що може оптимізувати процес покупки і використання продукту, скоротити кількість відходів і підвищити прибутковість.

Підвищений захист від крадіжки та відповідальність. Крадіжка та несанкціоноване використання акцій є серйозною проблемою для багатьох ресторанів. Відсутність систем контролю може призвести до того, що співробітники будуть використовувати або красти продукти в особистих цілях. Технологія Інтернету речей може допомогти вирішити цю проблему, впровадивши системи відеоспостереження, інтегровані з датчиками ваги і

системами управління запасами. Це дозволяє не тільки відстежувати використання продукту, але й документувати всі транзакції, зменшуючи ризик крадіжки та підвищуючи відповідальність працівників.

Поліпшення процесу закупівель. Процес придбання вмісту часто є складним і вимагає постійного відстеження запасів. Людський фактор може призвести до помилок, якщо відповідальна особа забуває вчасно оформити замовлення або неправильно оцінює кількість необхідних товарів. Автоматизовані системи Інтернету речей можуть значно спростити цей процес, автоматично створюючи замовлення на основі останніх даних інвентаризації. Це забезпечує своєчасне поповнення запасів, знижує ризик дефіциту і дозволяє зосередитися на інших аспектах управління рестораном.

Поліпшення якості обслуговування. Завдяки точному відстеженню запасів та автоматизованому управлінню покупками Ресторани можуть запропонувати клієнтам високий рівень обслуговування. Постійне оновлення та своєчасне поповнення інвентарю дозволяє уникнути ситуацій, коли певні страви будуть недоступні через відсутність інгредієнтів. Це підвищить задоволеність клієнтів і зміцнить репутацію закладу.

Забезпечення відповідності стандартам. У ресторанній індустрії важливо дотримуватися гігієнічні норми і стандарти якості зберігання продуктів. Технологія ІОТ допомагає автоматизувати цей процес, постійно контролюючи умови зберігання, такі як температура та вологість. Це дозволяє своєчасно виявляти відхилення від норми і вживати необхідних заходів для забезпечення якості та безпеки продукції.

Покращена аналітика та прогнозування. Завдяки величезній кількості даних, зібраних датчиками Інтернету речей, Ресторани можуть проводити поглиблений аналіз та прогнозування потреб у продуктах. Це дозволяє точніше планувати покупки, оптимізувати використання ресурсів та зменшити витрати. Крім того, аналітичні інструменти можуть допомогти виявити тенденції та закономірності споживання продукції, полегшити прийняття обґрунтованих рішень та підвищити ефективність управління.

Таким чином, його впровадження в ресторанну індустрію може радикально змінити підхід до управління запасами, забезпечити більш точний облік, скоротити кількість відходів, підвищити ефективність і знизити витрати. Це дозволяє ресторанам зосередитися на наданні високоякісних послуг своїм клієнтам і залишатися конкурентоспроможними на сучасному ринку.

В статті [1] наведено, що проблема неправильного управління запасами товарів у ресторанній індустрії має більш масштабний характер. За оцінками FDA, лише у Сполучених Штатах 30-40% продовольства втрачається даремно. Це приблизно 35 мільйонів тонн щорічно. Або в більш людських масштабах це 219 фунтів відходів на людину щороку. Витрати, пов'язані з харчовими відходами, стосуються не лише споживачів, але й набагато ширшого кола проблем, пов'язаних з кліматом та навколишнім середовищем. За даними Агентства з охорони навколишнього середовища США (EPA), харчові відходи в Америці щороку спричиняють 170 мільйонів викидів парникових газів. Це дорівнює річному обсягу викидів 42 вугільних електростанцій. Цей тривожний надлишок CO₂ є ключовим фактором прискорення кліматичної кризи, що зачіпає кожен регіон країни. Від бурхливих лісових пожеж на західному узбережжі до руйнівних торнадо на південному сході - поширення історичних стихійних лих вимагає термінових дій для досягнення мети Паризької угоди щодо обмеження зростання глобальної температури на 1,5° C вище доіндустріального рівня.

1.4 Традиційні підходи до управління залишками продуктів у ресторанах

Управління запасами без використання технології IoT має кілька важливих способів, за допомогою яких ресторани можуть ефективно управляти запасами, підтримувати оптимальний рівень товарів і задовольняти потреби клієнтів [2].

Мінімальна / максимальна система інвентаризації. Система мінімальних / максимальних запасів є одним з найбільш поширених методів управління запасами, особливо якщо вона інтегрована з технологією IoT. Це допомагає ресторанам підтримувати оптимальний рівень запасів, задовольняючи потреби у виробництві та продажах, мінімізуючи витрати на зберігання.

У традиційному підході цей метод передбачає визначення мінімального та максимального рівня запасів для кожного продукту. Коли залишки продукції досягають мінімуму, це сигнал до замовлення нових товарів. Максимальний рівень визначає, коли замовлення потрібно закрити, щоб уникнути перевантаження складу та непотрібних витрат на зберігання. Такий підхід запобігає нестачі товарів, забезпечує безперервність бізнес-процесів і знижує ризики, пов'язані із застарілістю продукції і надлишком запасів.

Технологія точного часу (JIT). Методологія Jit спрямована на максимальну економію коштів за рахунок мінімізації кількості запасів на складі. Основна ідея полягає в тому, що виробництво або купівля товарів здійснюється тільки тоді, коли необхідно виконати певне замовлення. Це не тільки зменшує витрати на зберігання та управління запасами, але також допомагає зменшити ризики, пов'язані зі старінням товарів .

Традиційні підходи вимагають високої точності планування та надійної логістики для впровадження jit, оскільки затримки доставки можуть

призвести до нестачі необхідних компонентів. Але такий спосіб дозволяє оптимізувати витрати і зберегти свіжість продукту.

Регулярний огляд запасів. Регулярний огляд запасів складається з регулярної оцінки кількості та стану товарів на складі з певною періодичністю, щотижня або щомісяця. Такий підхід дозволяє виявити розбіжності декомунізації між фактичними запасами системи обліку та даними, які можуть бути використані для поповнення запасів або коригування наявної інформації.

Регулярні перевірки також можуть допомогти зменшити ризики, пов'язані з крадіжкою продукції. Інтеграція Інтернету речей у цей процес може значно підвищити ефективність, дозволяючи автоматично збирати дані інвентаризації, зменшуючи необхідність ручного втручання та підвищуючи точність.

Метод аналізу ABC. Аналіз ABC – це поширений підхід до управління запасами, який допомагає оптимізувати ресурси, класифікуючи продукти відповідно до їх важливості для бізнесу. Продукти діляться на 3 категорії: А (найбільш важливі), В (середньої важливості) і С (найменш важливі). Такий підхід дозволяє зосередити свої ресурси на управлінні найважливішими товарами, які матимуть найбільший вплив на дохід ресторану, та забезпечити достатню доступність на складі. Можна значно підвищити продуктивність, інтегруючи аналітику IoT та ABC, автоматизуючи збір даних та точно відстежуючи поведінку товару на складі та в процесі продажу.

Використання програмного забезпечення для управління запасами. Сучасні ресторани часто використовують спеціальне програмне забезпечення для управління запасами. Це дозволяє автоматизувати багато рутинних процесів. Такі системи можуть відстежувати рівень запасів у режимі реального часу, створювати замовлення відповідно до поточних потреб та надавати точні фінансові звіти. Програмне забезпечення допомагає зменшити витрати, підвищити точність бухгалтерського обліку та оптимізувати процес закупівель.

Інтеграція таких систем з IoT дозволяє автоматизувати збір даних в залежності від стану складів, транспортних засобів і навіть товарів, що знаходяться в розпорядженні споживачів. Це дозволяє менеджерам з логістики проводити детальний аналіз стану запасів, оптимізувати рівень запасів та зменшити витрати на зберігання. Програмне забезпечення також допомагає підвищити задоволеність клієнтів, забезпечуючи більш точне виконання замовлень та своєчасну доставку товарів.

Метод FIFO (перший прийшов першим обслужений). FIFO-це простий та ефективний метод управління запасами, який передбачає використання найстарішого інвентарю перед Новим. Це допоможе зберегти інгредієнти свіжими та запобігти їх псуванню. У ресторанах цей метод часто використовується для продуктів з коротким терміном зберігання.

Таким чином, хоча традиційні методи управління запасами в цілому ефективні, їх можна значно поліпшити з впровадженням технології Інтернету речей. Автоматизація та точний облік, що надаються рішеннями Інтернету речей, можуть зменшити людські помилки, оптимізувати процеси та підвищити загальну ефективність управління запасами.

1.5 Переваги впровадження IoT-технологій у логістику ресторанів

Впровадження технології IoT для відстеження та управління запасами може кардинально змінити підхід ресторану до управління запасами. Технологія IoT дозволяє автоматизувати збір даних про стан запасів, їх переміщення і споживання в режимі реального часу, забезпечуючи точність і оперативність інформації, необхідної для ефективного управління. Основними перевагами систем IOT для управління запасами є:

Автоматизоване відстеження запасів. Пристрої IoT, такі як датчики ваги та інші датчики, дозволяють автоматично відстежувати кількість інгредієнтів на вашому складі чи кухні. Це значно зменшує потребу в ручному введенні даних і зменшує ймовірність помилок. Наприклад, на

полицях з сировиною можуть бути встановлені датчики для безперервного вимірювання ваги і інформування про поточний рівень запасів в режимі реального часу.

Швидке реагування на зміни запасів. Завдяки моніторингу в режимі реального часу Ресторани можуть швидко реагувати на зміни рівня запасів, оптимізувати процес поповнення запасів і уникати ситуацій нестачі або надлишкового запасу. Система може автоматично генерувати замовлення на поставку товарів, коли залишки продукції досягають мінімального рівня, забезпечуючи безперервність бізнес-процесів і знижуючи витрати на зберігання.

Оптимізація логістики та зменшення витрат. IoT дозволяє аналізувати і оптимізувати логістичні маршрути і складські операції, знижуючи витрати на зберігання і транспортування товарів. Датчики можуть збирати дані про умови зберігання, такі як температура і вологість, що дозволяє підтримувати оптимальні умови зберігання продуктів і знижувати ризик їх псування.

Підвищення задоволеності клієнтів. Завдяки точному моніторингу запасів і своєчасному їх поповненню Ресторани завжди матимуть доступ до необхідних їм інгредієнтів, що дозволить уникнути ситуацій, коли певних страв не буде в наявності. Це підвищить задоволеність клієнтів і зміцнить репутацію закладу.

Прогнозування та аналіз. Збираючи великі обсяги даних, ви можете використовувати інструменти аналітики та прогнозування для вдосконалення стратегій управління запасами, планування попиту та оптимізації асортименту. Маючи дані про споживання та попит, ви можете робити точні прогнози та ефективніше планувати свої покупки.

Інтеграція з іншими системами. Система IoT може бути інтегрована з ERP, CRM та іншими інформаційними системами, створюючи єдину автоматизовану екосистему для управління всіма аспектами роботи ресторану. Це забезпечує більш ефективну взаємодію між різними підрозділами та спрощує процес управління.

Реалізація IoT системи. Впровадження IoT системи вимагає комплексного підходу, який включає вибір відповідного обладнання (сенсори, вагові датчики тощо), налаштування зв'язку між пристроями, розробку або адаптацію програмного забезпечення для збору, обробки та аналізу даних, а також інтеграцію з існуючими бізнес-процесами та системами.

Впровадження IoT системи складається з наступних етапів:

Етап 1. Вибір обладнання: включає вибір відповідних сенсорів і пристроїв для відстеження запасів. Важливо врахувати специфіку продуктів і умов їх зберігання.

Етап 2. Налаштування зв'язку: встановлення і налаштування мережі для забезпечення безперервного зв'язку між сенсорами і центральною системою управління.

Етап 3. Програмне забезпечення: розробка або адаптація програмного забезпечення для збору, обробки та аналізу даних з сенсорів.

Етап 4. Інтеграція з бізнес-процесами: інтеграція IoT системи з існуючими ERP, CRM та іншими системами для створення єдиної екосистеми управління.

Етап 5. Навчання персоналу: проводити навчання для співробітників з метою ознайомлення їх з новою системою і методами її використання.

Реалізація IoT системи може значно підвищити ефективність управління запасами, знижуючи витрати і покращуючи загальну продуктивність ресторану.

1.6 Оптимальна модель IoT-системи для управління запасами

Впровадження IoT в управлінні запасами в ресторанах вимагає добре продуманої моделі, яка враховує всі аспекти цього процесу, від збору даних до аналітики та інтеграції з існуючими бізнес-процесами. Оптимальна модель систем IoT для управління запасами включає кілька ключових компонентів, кожен з яких відіграє важливу роль у забезпеченні ефективності та точності управління.

До компонентів IoT-системи відносять:

а) сенсори і датчики

1) вагові сенсори: використовуються для постійного моніторингу ваги продуктів на складах і кухні. Це дозволяє автоматично відстежувати кількість наявних запасів у реальному часі;

2) датчики температури і вологості: важливі для контролю умов зберігання продуктів, що особливо актуально для швидкопсувних товарів. Вони допомагають підтримувати оптимальні умови зберігання, знижуючи ризики псування продуктів;

б) мережа передачі даних:

1) бездротові мережі (Wi-Fi, Zigbee, LoRa): Використовуються для передачі даних від сенсорів до центральної системи управління. Вибір мережі залежить від специфіки ресторану і умов зберігання;

2) проміжні пристрої (шлюзи): Забезпечують зв'язок між сенсорами та центральною системою, конвертуючи дані в потрібний формат для подальшої обробки;

в) центральна система управління:

1) сервери і хмарні обчислення: зберігання і обробка даних про залишків, зібраних сенсорами. Хмарні обчислення дозволяють масштабувати систему і забезпечувати доступ до даних з будь-якої точки;

2) програмне забезпечення для аналізу даних: спеціалізовані програми аналізують дані про запаси, прогнозують потреби і генерують замовлення на основі поточного стану запасів і прогнозів попиту;

г) інтерфейси користувача:

1) адміністративна панель: вебінтерфейс або мобільний застосунок для менеджерів ресторанів, що дозволяє моніторити запаси, отримувати сповіщення про низький рівень запасів і управляти замовленнями;

2) інтерфейси інтеграції: інтеграція з ERP, CRM та іншими бізнес-системами для забезпечення єдиного потоку даних і спрощення управління бізнес-процесами.

1.7 Висновки до розділу

Після розгляду традиційних методів ведення обліку залишків товарів на складі можна зробити висновок, що впровадження датчиків і автоматичних систем збору даних зможе суттєво покращити роботу ресторанної індустрії в цілому. Забезпечення високої точної інформації про стан запасів в режимі реального часу. Це знижує ймовірність помилок і забезпечує оперативне реагування на зміни запасів. Це дозволяє ресторанам точно контролювати кількість інгредієнтів на своїх складах і кухнях, уникаючи дефіциту і надлишків продуктів. Такі системи також вчасно виявляють відхилення від норм, швидко реагують на них, забезпечують оптимальні умови зберігання продуктів і зводять до мінімуму ризик їх псування.

Автоматизація процесу управління запасами дозволяє значно знизити експлуатаційні витрати, пов'язані з ручним обліком і управлінням запасами. Оптимізація умов зберігання і своєчасне поповнення запасів також допоможуть знизити втрати через псування продукції. Завдяки використанню інтернету речей Ресторани можуть автоматизувати багато повсякденних

процесів, зменшуючи потребу в ручній праці та дозволяючи працівникам зосередитися на більш важливих завданнях.

Інтеграція систем Інтернету речей з іншими бізнес-системами забезпечує безперервний потік даних і підвищує загальну ефективність управління рестораном. Менеджери можуть приймати обґрунтовані рішення, засновані на точних даних і прогнозах, і більш ефективно планувати закупівлі, розподіл ресурсів і роботу персоналу. Такі системи також можуть аналізувати дані про споживання та попит, допомагаючи вдосконалювати бізнес-процеси та підвищувати загальну продуктивність.

Своєчасне поповнення запасів і підтримання оптимальних умов зберігання продуктів забезпечить наявність всіх необхідних для приготування інгредієнтів, що підвищить задоволеність клієнтів і зміцнить репутацію ресторану. Завдяки точному моніторингу запасів і швидкому реагуванню на зміни Ресторани можуть уникнути ситуацій, коли певні страви недоступні через брак інгредієнтів, що в цілому покращує якість обслуговування клієнтів.

Оптимальні моделі систем IoT для управління запасами включають комплексне використання датчиків, мережевих технологій, програмного забезпечення для аналізу даних та інтеграцію з існуючими бізнес-системами. Впровадження такої системи дозволяє значно підвищити точність і оперативність управління запасами, знизити витрати, підвищити ефективність і поліпшити якість обслуговування клієнтів. Всі ці фактори сприяють підвищенню конкурентоспроможності ресторану і забезпечення його стабільного розвитку і зростання на ринку.

2 ДОСЛІДЖЕННЯ ІСНУЮЧИХ РІШЕНЬ ТА ТЕХНОЛОГІЙ ДЛЯ СИСТЕМ МОНІТОРИНГУ ЗАЛИШКІВ ПРОДУКТІВ

2.1 Огляд готових рішень для систем моніторингу залишків продуктів

Сьогодні вже існують різні платформи, які пропонують різні рішення для моніторингу та управління залишками продуктів в ресторанах. Вибір оптимальної системи залежить від конкретних потреб бізнесу, масштабу його діяльності та існуючої інфраструктури.

При виборі системи управління запасами важливо враховувати такі фактори, як можливість інтеграції з POS-системами, наявність аналітичних інструментів для прогнозування попиту, можливості хмарного розгортання і забезпечення моніторингу запасів в режимі реального часу. Деякі платформи також пропонують можливості контролю якості та оптимізації закупівель, що може бути дуже корисним для ресторанного бізнесу.

У цьому розділі буде проведено дослідження кількох основних систем управління запасами, які мають різні функції та орієнтовані на різні сегменти ринку. Серед розглянутих рішень: SensrTrx, ParTech, MarketMan. Кожна з цих систем має свої переваги та недоліки

2.1.1 Огляд платформи SensrTrx

SensrTrx – це програмне забезпечення для аналізу виробництва та продуктивності, яке дозволяє відстежувати стан виробничого процесу в режимі реального часу. Це допомагає скоротити час простою, поліпшити своєчасну доставку і підвищити прибутковість виробничих компаній. Ключові функції включають моніторинг ключових показників ефективності (KPI), візуалізацію даних, Аналіз витрат, декомунізацію та контроль якості. SensrTrx підтримує хмарні розгортання, тому ви можете легко масштабувати свою систему з одного заводу на глобальний рівень [3].

Можна виділити основні функції SensrTrx:

- моніторинг ключових показників ефективності (KPI): за допомогою SensrTrx можна відстежувати ключові показники ефективності, такі як продуктивність обладнання, обсяг виробництва та використання ресурсів. Це дозволяє адміністраторам приймати обґрунтовані рішення на основі реальних даних;

- візуалізація даних: програмне забезпечення забезпечує візуалізацію даних в режимі реального часу, дозволяючи користувачам швидко оцінити стан виробничого процесу. Візуальні звіти і графіки допоможуть вам виявити проблемні області і швидко зробити необхідні кроки;

- аналіз витрат: SensrTrx надає інструменти для поглибленого аналізу виробничих витрат для виявлення областей для зниження витрат та підвищення ефективності;

- аналіз часу простою: система дозволяє відстежувати час простою обладнання, аналізувати причини простою і вживати заходів щодо їх мінімізації. Це допомагає підвищити загальну продуктивність і скоротити втрати;

- контроль якості: SensrTrx забезпечує моніторинг виробничого процесу і контроль якості для виявлення і усунення дефектів на ранніх стадіях виробництва, що скорочує кількість дефектів і підвищує якість продукції;

- хмарне розгортання: SensrTrx підтримує хмарні розгортання, які дозволяють масштабувати системи від окремого підприємства до глобального рівня. Хмарні рішення забезпечують доступ до даних з будь-якої точки світу і спрощують інтеграцію з іншими системами.

Серед переваг можна виділити можливість SensrTrx надавати високоточні дані в режимі реального часу, щоб ви могли швидко реагувати на зміни у виробничому процесі. Система допомагає підвищити продуктивність за рахунок моніторингу ключових показників ефективності та аналізу часу простою. Впровадження Sensrtrx допомагає знизити

виробничі витрати за рахунок поглибленого аналізу витрат і контролю якості, що дозволяє виявляти і усувати дефекти на ранній стадії. Крім того, хмарне розгортання системи забезпечує гнучкість і масштабованість, що спрощує інтеграцію з іншими корпоративними системами. жовтень.Недоліки SensrTrx.

До недоліків можна віднести потребу значних ресурсів та часу, особливо для великих виробничих підприємств. Також висока вартість ліцензії та впровадження може бути бар'єром для малих та середніх підприємств.

2.1.2 Огляд платформи ParTech

ParTech пропонує рішення для управління ресторанами, включаючи системи моніторингу запасів. Програмне забезпечення par inventory допомагає автоматизувати процес інвентаризації, зменшуючи ручні операції та підвищуючи точність ваших даних. Система надає можливість інтеграції з POS-системою, що дозволяє швидко відстежувати використання інгредієнтів і формувати замовлення на основі реальних даних про продажі. Крім того, ParTech надає аналітичні інструменти для прогнозування попиту та оптимізації закупівель [4].

Можна виділити основні наступні основні функції ParTech:

- автоматизація процесу інвентаризації: за допомогою Par Inventory ви можете автоматизувати процес обліку запасів, зменшуючи необхідність ручного введення даних та ймовірність помилок. Система забезпечує точний облік використання сировини і залишків на складах;
- інтеграція з POS-системами: система забезпечує плавну інтеграцію з системою POS, дозволяючи Вам швидко відстежувати продаж і використання інгредієнтів. Це дозволить вам отримувати останні дані про запаси в режимі реального часу;

– аналіз та звітність: Par Inventory надає потужні аналітичні інструменти для аналізу даних про запаси, що дозволяє прогнозувати попит і оптимізувати закупівлі. Створюються детальні звіти, які допоможуть вам приймати обґрунтовані управлінські рішення;

– оптимізація закупівель: завдяки аналітичним інструментам система допомагає оптимізувати процес закупівель, знижує витрати і забезпечує своєчасне поповнення запасів;

– управління рецептами: Par inventory також дозволяє керувати рецептами, що забезпечить точний облік використання інгредієнтів для кожної страви та допоможе зменшити витрати на продукти.

До переваг можна віднести автоматизацію процесів інвентаризації, що дозволяє знизити кількість ручних операцій і підвищити точність даних. Система інтегрується з POS-системами, забезпечуючи актуальні дані про запаси в реальному часі, що дозволяє оперативно реагувати на зміни попиту і продажів. Аналітичні інструменти Par Inventory допомагають оптимізувати закупівлі та прогнозувати попит, знижуючи витрати і забезпечуючи своєчасне поповнення запасів. Управління рецептами дозволяє точно відстежувати використання інгредієнтів, що сприяє зниженню витрат на харчові продукти і підвищенню рентабельності ресторану.

Серед недоліків можна виділити значні початкові інвестиції, витрати часу на налаштування, особливо для невеликих ресторанів. Також вартість ліцензії може бути високою для малих підприємств.

2.1.3 Огляд платформи MarketMan

MarketMan – це хмарне програмне забезпечення для управління запасами та постачаннями, спеціально розроблене для ресторанів. Це дозволяє рестораторам відстежувати запаси, керувати замовленнями, контролювати витрати та зменшувати втрати [5]. MarketMan інтегрований з популярною системою POS, яка забезпечує точний облік використання інгредієнтів і своєчасне поповнення запасів. Система також надає можливості аналітики та звітності, які допомагають рестораторам приймати обґрунтовані рішення щодо управління запасами.

Основні функції Marketman наступні: управління запасами, інтеграція з POS-системами, замовлення та доставка, контроль витрат, аналіз та звітність. Переваги та недоліки такі самі як і у ParTech.

2.1.4 Висновки огляду платформ

В цілому, як можна було помітити, MarketMan і ParTech мають багато спільних функцій. Основні відмінності між ними полягають у специфіці цільової аудиторії та деталях інтеграції з постачальниками. MarketMan є потужним інструментом для ресторанів та харчових підприємств, орієнтованим на автоматизацію процесів управління запасами та оптимізацію витрат. ParTech пропонує ширший спектр рішень, включаючи готелі, та забезпечує комплексні інструменти для управління всім ресторанним бізнесом, з акцентом на безшовну інтеграцію з існуючими системами.

Чого не можна сказати про SensrTrx. Він відрізняється від MarketMan і ParTech тим, що орієнтований на виробничі підприємства і не інтегрується з POS-системами. SensrTrx фокусується на моніторингу виробничих процесів, аналізі продуктивності та управлінні якістю, що робить його ідеальним для виробничих компаній, які шукають способи оптимізувати свої виробничі

процеси. виробничі потреби. Порівняльна характеристика систем наведена у таблиці 2.1.

Таблиця 2.1 – Порівняльна таблиця систем

Функціональність	SensrTrx	ParTech	MarketMan
Управління запасами	Так	Так	Так
Інтеграція з POS-системами	Ні	Так	Так
Автоматизація інвентаризації	Так	Так	Так
Замовлення і постачання	Ні	Автоматичне створення замовлень, управління постачальниками	Інтеграція з постачальниками, автоматизація
Контроль витрат	Так	Так	Так
Аналітика та звітність	Потужні аналітичні інструменти	Потужні аналітичні інструменти	Потужні аналітичні інструменти
Управління рецептами	Ні	Так	Так
Мобільний доступ	Ні	Так	Так
Хмарне розгортання	Так	Так	Так
Підтримка користувачів	Так	Так	Так

2.2 Аналіз стеків технологій для створення систем моніторингу залишків

Існує багато різних підходів та способів розробки систем моніторингу запасів, кожен з яких має свої переваги та недоліки. Вибір технологічного стека для реалізації такого проекту є важливим і відповідальним завданням, яке впливає на продуктивність, масштабованість та загальну ефективність системи. У цьому розділі я розгляну кілька популярних технологічних стеків, які можуть бути використані для створення систем моніторингу запасів. Серед них: LAMP (Linux, Apache, MySQL, PHP), MEAN (MongoDB, Express.js, Angular, Node.js), MERN (MongoDB, Express.js, React js, Node.js) Django (Python, Django, PostgreSQL) та .NET (C#, ASP.NET, SQL Server).

Кожен з цих стеків має свої унікальні особливості та підходи до вирішення завдань розробки. Було проведено детальний аналіз кожного стека, оцінюючи їх переваги та недоліки, щоб визначити, яке рішення є найбільш оптимальним для мого проекту

2.2.1 Огляд стеку LAMP

LAMP – це класичний технологічний стек для розробки веб-додатків, який включає наступні компоненти [6]:

- Linux: операційна система, яка служить основою для інших компонентів стека [7];
- Apache: вебсервер, який обробляє запити від клієнтів і передає їх на серверну частину застосунку [8];
- MySQL: реляційна база даних, яка зберігає дані застосунку [9];
- PHP: серверна мова програмування, яка використовується для написання логіки застосунку [10].

Переваги технології LAMP:

- широка підтримка та довіра;

- відкритий код і безкоштовність;
- Apache: вебсервер Apache вважається надійним, швидким і безпечним. Він також є модульним, що робить його легко налаштованим;
- безпека: стек LAMP має архітектуру безпеки та шифрування корпоративного рівня;
- ефективність: використання стеку LAMP може скоротити час розробки додатків завдяки простоті його конфігурації. Наприклад, програмісти можуть почати з модуля Apache і модифікувати код за необхідності, замість того, щоб розробляти код повністю з нуля;
- масштабованість;
- LAMP стабільна і не потребує особливого обслуговування;
- розуміння: оскільки PHP і MySQL відносно прості у розумінні, розробка стеку LAMP є хорошим варіантом для початківців.

До недоліків можна віднести наступні:

- багатомовність: використання PHP і JavaScript вимагає наявності повноцінного розробника або команди розробників;
- обмежена підтримка ОС: підтримка обмежена системами на базі Linux;
- монолітна архітектура: LAMP менш гнучкий і масштабований, ніж хмарні архітектури.

2.2.2 Огляд стеку MEAN

MEAN – це сучасний технологічний стек для розробки веб-додатків, який включає наступні компоненти [6]:

- MongoDB: документо-орієнтована база даних NoSQL, яка забезпечує високу продуктивність і гнучкість у зберіганні даних [11];
- Express.js: вебфреймворк для Node.js, який спрощує створення серверної частини додатків [12];

- Angular або React: фронте́нд-фреймворк для створення динамічних веб-додатків з багатим інтерфейсом користувача [13,14];
- Node.js: платформа для виконання JavaScript на сервері, яка забезпечує високу продуктивність і асинхронність обробки запитів [15].

До переваг можна віднести наступні:

- використання однієї мови: MEAN або MERN вважається "повноцінним стеком", оскільки використовує JavaScript як єдину мову. Це спрощує переключення між клієнтською та серверною програмуванням та робить його більш ефективним. Наприклад, один розробник на JavaScript може побудувати цілий вебзастосунок;
- оновлення в реальному часі та демонстрації: Технології у стеку MEAN дозволяють надсилати оновлення в реальному часі до розгорнутих веб-додатків. Розробники також можуть швидко демонструвати функціональність вебзастосунків на етапі розробки;
- сумісність з хмарою: технології стеку MEAN або MERN можуть працювати з хмарними функціями, які використовуються в сучасних веб-службах (наприклад, виклик API для отримання даних);
- файли JSON: MEAN або MERN дозволяє зберігати документи у форматі JSON файлів, які призначені для швидкого обміну даними через мережі;
- ефективність: розробники можуть використовувати ресурси з публічних репозиторіїв та бібліотек, щоб скоротити час розробки веб-додатків. Це робить розробку на стеку MEAN або MERN економічно вигідним варіантом, який може зацікавити стартапи;
- швидке середовище виконання та легкість обслуговування: Середовище виконання Node.js є швидким і відповідає високим вимогам, тоді як фреймворк Angular.js легко підтримувати та тестувати;
- підтримка крос-платформ: MEAN або MERN - це крос-платформений стек, що означає, що його веб-додатки можуть функціонувати на різних операційних системах.

До недоліків можна віднести наступні:

- потенційна втрата даних: великомасштабні застосунки можуть втрачати дані через те, що MongoDB вимагає надмірної пам'яті для зберігання даних. Крім того, MongoDB не підтримує транзакційні функції; Час завантаження та несумісність: JavaScript може повільно завантажувати вебсайти або додатки на деяких пристроях, особливо старих або бюджетних. Вебзастосунки навіть можуть стати непрацездатними, якщо JavaScript вимкнено на пристрої. Крім того, впровадження MEAN може бути складним у існуючих архітектурах, оскільки старі застосунки ймовірно не використовують JavaScript;
- висока потреба в обслуговуванні: технології в стеку MEAN часто оновлюються, що вимагає частих оновлень вебзастосунків.

2.2.3 Огляд стеку Django

Django – це високорівневий вебфреймворк на мові програмування Python, який дозволяє швидко і ефективно розробляти вебзастосунки. Основні компоненти стека Django [16] включають:

- Python: високорівнева мова програмування з простим і зрозумілим синтаксисом, яка забезпечує високу продуктивність розробки [17];
- Django: вебфреймворк для Python, який слідує принципам DRY (Don't Repeat Yourself) і KISS (Keep It Simple, Stupid), що сприяє швидкому і зручному створенню вебзастосунків [18];
- PostgreSQL: потужна реляційна база даних з підтримкою складних запитів і транзакцій, яка забезпечує надійне зберігання даних [19].

До переваг можна віднести наступні:

- швидкість розробки: Django дозволяє швидко створювати прототипи і готові рішення завдяки великій кількості вбудованих функцій і готових компонентів;

- висока продуктивність: Python забезпечує високу продуктивність розробки, а PostgreSQL забезпечує швидке і ефективне зберігання даних;
- безпека: Django має вбудовані механізми для забезпечення безпеки застосунків, включаючи захист від SQL-ін'єкцій, XSS-атак та CSRF-атак;
- широка підтримка спільноти: Django має велику і активну спільноту, що забезпечує велику кількість ресурсів, документації і готових рішень;
- масштабованість: Django легко масштабується, що дозволяє використовувати його як для малих, так і для великих проектів.

До недоліків можна віднести наступні:

- висока крива навчання: для новачків може бути складно освоїти всі можливості Django, особливо якщо вони не мають досвіду роботи з Python;
- важкість налаштування: Django може вимагати значних зусиль для налаштування і оптимізації продуктивності, особливо для великих проектів;
- монотонність: хоча Django слідує принципу DRY, це може призвести до створення монотонного коду, що важко підтримувати і розширювати.

2.2.4 Огляд стеку .NET

.NET – це відкрита платформа [20] з інструментами, мовами програмування та бібліотеками для створення сучасних настільних, веб- і мобільних застосунків. Вона безперебійно працює на різних операційних системах. Підтримувані мови включають C# [21], F# [22] і Visual Basic, а також сумісні сторонні опції. Відома своєю легкістю, повторним використанням коду, сильною продуктивністю, безпекою та підтримкою спільноти, .NET є вищим вибором для розробки.

До переваг .NET можна віднести наступні:

- розробка крос-платформ: від легких вебзастосунків до міцних корпоративних рішень, .NET обслуговує все. Його ключова особливість полягає у можливості створення та виконання коду на різних платформах через єдине середовище програмування та набір інструментів;
- багаті ресурси та бібліотеки: доступні бібліотеки та зручні інструменти управління є сильними сторонами .NET. Розробники можуть покращити свої проєкти за допомогою багатих можливостей повторного використання відкритого коду, компонентів та модулів;
- висока безпека і надійність: Microsoft забезпечує безпеку та надійність .NET, офіційно підтримуючи його, надаючи рекомендації та реагуючи на загрози, коли вони матеріалізуються;
- доступ до кваліфікованих розробників: Однією з головних переваг .NET-фреймворку протягом багатьох років був великий ринок і легкий доступ до кваліфікованих талантів. У світі нараховується від 5 до 8 мільйонів розробників .NET, багато з яких є високопрофесійними фахівцями з понад 10 роками досвіду використання та розробки цієї технології. Це чудова новина для всіх, хто намагається найняти команду інженерів .NET з успішним досвідом.

До недоліків можна віднести наступні:

- вартість ліцензії: використання SQL Server і інших продуктів Microsoft може бути дорогим, що може бути бар'єром для малих і середніх підприємств;
- залежність від екосистеми Microsoft: Розробка на платформі .NET може обмежити вибір інструментів і технологій, оскільки більшість з них пов'язана з екосистемою Microsoft;
- крива навчання: для новачків може бути складно освоїти всі можливості платформи .NET, особливо якщо вони не мають досвіду роботи з C# та іншими продуктами Microsoft.

2.2.5 Обґрунтування вибору стеку технологій

Підбиваючи все вищеописане, було прийнято рішення обрати стек MERN для розробки програми. Головною причиною вибору є зручність використання єдиної мови програмування – JavaScript – для всіх частин додатка. Це значно спрощує розробку, обслуговування та співпрацю між розробниками. Node.js забезпечує високу продуктивність і масштабованість завдяки асинхронній обробці запитів. React дозволяє створювати динамічні та інтерактивні інтерфейси користувача, покращуючи користувацький досвід. MongoDB забезпечує гнучке та масштабоване зберігання даних. Велика і активна спільнота MERN надає доступ до численних ресурсів і бібліотек.

Отже, стек MERN є ідеальним вибором для розробки моєї системи моніторингу запасів, оскільки дозволяє створити сучасне, масштабоване та ефективне рішення. Порівняльна характеристика популярних стеків технологій наведена у таблиці 2.2.

Таблиця 2.2 – Порівнювальна таблиця популярних стеків технологій

Критерій	LAMP	Django	.NET	MEAN/ MERN
Мова	PHP	Python	C#, F#	JavaScript
База даних	MySQL	PostgreSQL	SQL Server	Mongo DB
Фреймворк	PHP	Django	ASP.NET	Espress.js, Angular або React
Продуктивність	Помірна	Висока	Висока	Висока
Гнучкість	Висока	Висока	Висока	Висока
Масштабованість	Обмежена	Висока	Висока	Висока
Безпека	Висока	Висока	Висока	Помірна
Підтримка транзакцій	Висока	Висока	Висока	Помірна
Легкість навчання	Помірна	Висока крива навчання	Помірна крива навчання	Крута крива навчання
Інтеграція з іншими системами	Висока	Висока	Висока	Висока
Вартість впровадження	Низька	Низька	Висока	Низька
Підтримка спільноти	Широка	Широка	Широка	Широка
Мобільний доступ	Обмежений	Високий	Високий	Високий

2.3 Аналіз інструментів та компонентів для збору даних про вагу

Крім адміністративної панелі, потрібні інструменти для збору тої самої інформації, яка буде відображатись для подальшого аналізу. У процесі розробки системи моніторингу залишків важливо вибрати найкращі інструменти та компоненти для збору даних про вагу. Це включає в себе вибір відповідної платформи мікроконтролера та датчика ваги, які будуть використовуватися для точного вимірювання ваги матеріалу. У цьому розділі буде проведено детальний аналіз різних мікроконтролерів та платформ датчиків ваги, щоб вибрати найбільш відповідне рішення для розробки програми.

2.3.1 Аналіз платформи RASPBERRY PI

Raspberry Pi – це популярна платформа мікроконтролера, яка використовується для різноманітних проектів, включаючи збір даних про вагу (рис.2.1).



Рисунок 2.1 – Мікроконтроллер Raspberry Pi[23]

До переваг можна віднести наступні:

- висока продуктивність: Raspberry Pi має потужний процесор, що дозволяє обробляти великі обсяги даних і виконувати складні обчислення;

- різноманітні інтерфейси підключення: підтримує численні інтерфейси, такі як GPIO, I2C, SPI, що дозволяє підключати різноманітні датчики і периферійні пристрої;
- велика спільнота та ресурси: завдяки популярності Raspberry Pi існує безліч ресурсів, документації та прикладів для розробників;
- можливість роботи з ОС: Raspberry Pi працює під управлінням різних операційних систем, таких як Raspbian, що дозволяє використовувати широкий спектр програмного забезпечення.

До недоліків можна віднести наступні:

- високе енергоспоживання: порівняно з іншими мікроконтролерами, Raspberry Pi має відносно високе енергоспоживання, що може бути критичним для проектів з автономним живленням;
- ціна: Raspberry Pi коштує дорожче за деякі інші мікроконтролери, що може збільшити загальні витрати на проект;
- великий розмір: розмір Raspberry Pi може бути занадто великим для компактних проектів або пристроїв з обмеженим простором.

2.3.2 Аналіз платформи ESP8266

ESP8266 – це мікроконтролер з вбудованим Wi-Fi модулем, який широко використовується для IoT проектів (рис.2.2).



Рисунок 2.2 – Мікроконтролер ESP8266 [24]

До переваг можна віднести наступні:

- низька вартість: ESP8266 є одним з найдешевших мікроконтролерів з вбудованим Wi-Fi, що робить його привабливим для багатьох проектів;
- низьке енергоспоживання: цей мікроконтролер споживає мало енергії, що є важливим для автономних пристроїв;
- компактний розмір: маленький розмір ESP8266 дозволяє використовувати його в компактних пристроях;
- Wi-Fi підключення: вбудований Wi-Fi модуль дозволяє легко підключати пристрій до інтернету для віддаленого моніторингу і управління.

До недоліків можна віднести наступні:

- обмежена обчислювальна потужність: ESP8266 має обмежену обчислювальну потужність порівняно з Raspberry Pi, що може бути недостатнім для деяких складних задач;
- менше інтерфейсів підключення: кількість доступних інтерфейсів підключення (GPIO, I2C, SPI) обмежена порівняно з Raspberry Pi;
- обмежена пам'ять: менший обсяг оперативної пам'яті і флеш-пам'яті може обмежувати можливості проекту.

2.3.3 Аналіз платформи Arduino UNO

Arduino UNO – популярний мікроконтролер, який широко використовується для різноманітних електронних проектів, включаючи збір даних про вагу (рис.2.3).



Рисунок 2.3 – Мікроконтроллер Arduino UNO [25]

Arduino UNO має безліч переваг, що роблять його популярним вибором серед початківців та досвідчених розробників. По-перше, цей мікроконтролер відомий своєю простотою використання, що значно полегшує процес освоєння для новачків. Додатково, величезна спільнота користувачів і розробників, що підтримують Arduino UNO, забезпечує доступ до великої кількості документації, прикладів і бібліотек. Це значно спрощує роботу з різними проектами. Arduino UNO також підтримує численні периферійні пристрої та датчики через інтерфейси GPIO, I2C та SPI, що робить його універсальним інструментом для різних застосувань. Низька вартість цього мікроконтролера робить його доступним для широкого кола користувачів, що також є вагомою перевагою.

Проте, існують і деякі недоліки, які варто враховувати. Arduino UNO має обмежену обчислювальну потужність, що може бути недостатньо для виконання складних завдань. Крім того, обмежений обсяг оперативної та флеш-пам'яті може обмежити можливості реалізації більш масштабних проектів. Ще одним недоліком є відсутність вбудованого Wi-Fi модуля, що може вимагати додаткових компонентів для підключення до інтернету. Порівняльна характеристика мікроконтролерів наведена у таблиці 2.3.

Таблиця 2.3 – Порівняльна характеристика мікроконтролерів

Критерій	Raspberry Pi	ESP8266	Arduino UNO
Вартість	Висока	Низька	Низька
Енергоспоживання	Високе	Низьке	Низьке
Розмір	Великий	Компактний	Компактний
Обчислювальна потужність	Висока	Обмежена	Обмежена
Пам'ять	Велика	Обмежена	Обмежена
Інтерфейси підключення	Різноманітні (GPIO, I2C, SPI)	Обмежені (GPIO, I2C, SPI)	Широкий вибір (GPIO, I2C, SPI)
Wi-Fi підключення	Може бути додано окремо	Вбудований	Відсутній
Простота використання	Складніший у налаштуванні	Простий у використанні	Дуже простий у використанні
Спільнота та ресурси	Велика і активна	Велика і активна	Велика і активна

2.3.4 Огляд датчиків

Тензометричні датчики – це найбільш поширені датчики ваги, які використовуються для точного вимірювання маси об'єктів. Вони працюють на основі зміни електричного опору матеріалу під впливом механічної деформації (рис.2.4).

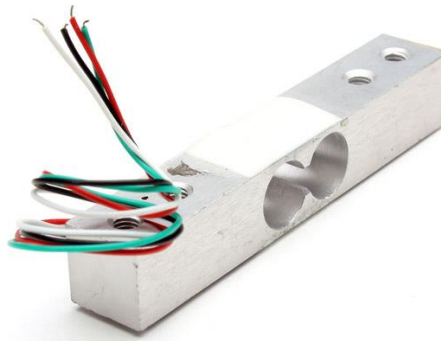


Рисунок 2.4 – Тензодатчик [26]

До переваг можна віднести наступні:

- висока точність: тензометричні датчики забезпечують високу точність вимірювань, що є важливим для багатьох додатків.
- надійність: ці датчики є надійними і довговічними, що робить їх ідеальними для тривалого використання;
- широкий діапазон вимірювань: тензометричні датчики можуть вимірювати масу від кількох грамів до кількох тонн, що робить їх універсальними.

До недоліків можна віднести наступні:

- чутливість до умов навколишнього середовища: Тензометричні датчики можуть бути чутливими до змін температури, вологості та інших факторів, що може вплинути на точність вимірювань;
- необхідність калібрування: для забезпечення точної роботи тензометричні датчики потребують регулярного калібрування;
- складність у налаштуванні: для правильної роботи тензометричних датчиків може знадобитися складне налаштування та інтеграція з іншими компонентами системи.

Крім тензометричних датчиків, існують і інші типи датчиків ваги, такі як ємнісні, індуктивні та п'єзоелектричні датчики (рис.2.5-2.7). Кожен з них

має свої переваги і недоліки, які варто враховувати при виборі датчика для конкретного проекту.



Рисунок 2.5 – Ємнісний датчик [27]



Рисунок 2.6 – П'єзоелектричний датчик[28]



Рисунок 2.7 – Індуктивний датчик [29]

До переваг можна віднести наступні:

- ємнісні датчики: висока чутливість і швидкий відгук, можуть працювати в агресивних середовищах;

- індуктивні датчики: висока надійність і довговічність, добре працюють в умовах високої температури і забрудненого середовища;
- п'єзоелектричні датчики: висока точність і швидкий відгук, особливо підходять для динамічних вимірювань.

До недоліків можна віднести наступні:

- ємнісні датчики: чутливість до вологості і забруднень, можуть потребувати регулярного очищення;
- індуктивні датчики: обмежений діапазон вимірювань, не підходять для вимірювання малих мас;
- п'єзоелектричні датчики: висока вартість, можуть бути складними у налаштуванні і інтеграції.

2.3.5 Обґрунтування вибору мікроконтролера і датчиків

Аналізуючи різні платформи мікроконтролерів та датчики ваги, було прийнято рішення, що для реалізації поставлених завдань найбільш оптимальним вибором буде використання мікроконтролера Arduino UNO у поєднанні з тензометричними датчиками ваги. Це рішення забезпечує оптимальний баланс між вартістю, точністю, енергоспоживанням і можливістю інтеграції з іншими компонентами системи. Arduino UNO є доступним, простим у використанні і має велику спільноту підтримки, що робить його ідеальним вибором для нашого проекту. Тензометричний датчик забезпечують високу точність і надійність вимірювань, що є критичним для моніторингу запасів у ресторані. Хоча мій проект потребує підключення до інтернету, я свідомо обрав звичайну Arduino UNO R3 з додатковим Wi-Fi модулем ESP8266 ESP-01 через те, що така конструкція є дешевшою та дозволяє зберегти гнучкість у розробці. Крім того, я вже маю досвід роботи з Arduino UNO під час навчання, тому ця платформа є зрозумілою та зручною для мене, що значно спрощує процес розробки і налаштування системи.

Порівняльна характеристика датчиків ваги наведена у таблиці 2.4.

Таблиця 2.4 – Порівнювальна таблиця датчиків ваги

Критерій	Тензометричні датчики	Ємнісні датчики	Індуктивні датчики	П'єзоелектричні датчики
Точність	Висока	Висока	Помірна	Висока
Надійність	Висока	Помірна	Висока	Висока
Діапазон вимірювань	Широкий	Обмежений	Обмежений	Помірний
Чутливість до середовища	Чутливість до температури та вологості	Чутливі до вологості	Висока стійкість до умов	Висока стійкість до умов
Необхідність калібрування	Потребують регулярного калібрування	Можуть потребувати частого очищення	Низька	Потребують регулярного калібрування
Вартість	Помірна	Висока	Висока	Висока
Складність у налаштуванні	Висока	Висока	Низька	Висока

2.4 Огляд та аналіз різних технологій передачі даних

Передача даних з пристроїв Інтернету речей в мережу є важливим аспектом для ефективної оптимізації процесів, збору даних і управління системою. Бездротові технології, такі як Wi-Fi, Bluetooth та Zigbee, відіграють ключову роль у цьому процесі, забезпечуючи швидкий та надійний обмін інформацією між пристроями та серверними або хмарними системами.

Однією з найпоширеніших бездротових технологій, що використовуються в домашніх та комерційних системах Інтернету речей, є Wi-Fi. Завдяки своїй здатності підтримувати високі швидкості передачі даних

і широкому охопленню, Wi-Fi вимагає передачі великих обсягів даних, таких як відеоспостереження і мультимедійні файли, ця технологія забезпечує надійне і стабільне з'єднання, що важливо для багатьох додатків в сучасних розумних будинках і офісах.

Ethernet також відіграє важливу роль у сфері Інтернету речей, особливо якщо мова йде про стабільні високошвидкісні дротові з'єднання. Ethernet забезпечує надійне з'єднання з високою швидкістю передачі даних. Це важливо для застосунків, де безперервність та швидкість передачі інформації є ключовими факторами. Використання Ethernet особливо важливо в промислових системах і центрах обробки даних, де потрібна висока пропускна здатність і мінімальна затримка.

Bluetooth – ще одна важлива бездротова технологія, ідеально підходить для створення невеликих мереж, де не потрібні великі відстані зв'язку і висока швидкість передачі даних. Bluetooth особливо корисний для носіїв та датчиків здоров'я, де економія енергії є ключовим фактором. Технологія забезпечує стабільне з'єднання на коротких відстанях і споживає мало енергії, тому пристрій може тривалий час працювати без підзарядки.

Технологія Zigbee, яка використовує малу потужність для передачі даних на короткі відстані, ідеально підходить для створення щільних бездротових мереж датчиків і виконавчих механізмів. Він часто використовується в системах домашньої автоматизації, що вимагають високої надійності і здатності до самоорганізації мережі. Завдяки своїй надійності і масштабованості, Zigbee добре підходить для "розумних" будинків, де десятки або сотні пристроїв повинні безперервно взаємодіяти, забезпечуючи користувачам високий рівень безпеки.

На додаток до цих технологій існують інші варіанти бездротового зв'язку, такі як LoRaWAN. Технологія забезпечує передачу даних на великі відстані при низькому енергоспоживанні, що робить її ідеальною для застосування в "розумних містах" і промислових рішеннях Інтернету речей. LoRaWAN дозволяє створювати масштабовану мережу з великою кількістю

пристроїв, які можуть обмінюватися даними на великих відстанях. Це особливо корисно для моніторингу стану міської інфраструктури та управління ресурсами.

Sigfox – ще одна важлива технологія, яка забезпечує дуже низьке енергоспоживання при передачі даних на великі відстані. Ця технологія особливо корисна для пристроїв, які передають невеликі обсяги даних і вимагають тривалого часу автономної роботи, таких як датчики для моніторингу навколишнього середовища та відстеження місцезнаходження активів.

Nb - IoT (вузькосмуговий Інтернет речей) – це технологія стільникового зв'язку, розроблена спеціально для пристроїв Інтернету речей, які потребують широкого покриття та низького енергоспоживання. Вона забезпечує надійне підключення у важкодоступних місцях, таких як підвали або віддалені місця розташування, що робить її ідеальною для моніторингу інфраструктури та управління ресурсами.

Протокол передачі телеметрії з чергою повідомлень (MQTT) – це спрощений протокол обміну повідомленнями, який використовується для передачі даних між пристроями та серверами в мережі IoT. Його модель публікації / підписки забезпечує ефективний обмін даними з мінімальною затримкою і ідеально підходить для додатків, які потребують швидкого реагування на зміни.

Використання цих бездротових технологій в IoT значно полегшує управління запасами та логістику, дозволяючи компаніям відстежувати місцезнаходження та стан товарів у режимі реального часу. Це дозволяє оптимізувати процеси зберігання та доставки, зменшити витрати, оптимізувати робочі процеси та забезпечити високий рівень задоволеності клієнтів.

Використання цих бездротових технологій в IoT значно полегшує управління запасами та логістику, дозволяючи компаніям відстежувати місцезнаходження та стан товарів у режимі реального часу. Це дозволяє

оптимізувати процеси зберігання та доставки, зменшити витрати, оптимізувати робочі процеси та забезпечити високий рівень задоволеності клієнтів. У даному проекті, враховуючи вимоги до стабільності та швидкості передачі даних, найбільш оптимальним вибором є використання Ethernet або Wi-Fi з протоколом HTTPS. Це забезпечує надійне та безпечне з'єднання, яке має вирішальне значення для наших систем збору даних та управління. Порівняльна характеристика технологій передачі даних наведена у табл. 2.5.

Таблиця 2.5 – Порівняльна таблиця технологій передачі даних

Технологія	Дальність зв'язку	Швидкість передачі даних	Енергоспоживання	Складність реалізації	Тип мережі
Wi-Fi	Середня (до 100 м)	Висока	Високе	Середня	Інфраструктура
Ethernet	Обмежена кабелем	Висока	Високе	Середня	Інфраструктура
Bluetooth	Коротка (до 10 м)	Низька	Низьке	Низька	Особиста мережа (PAN)
Zigbee	Середня (до 100 м)	Низька	Дуже низьке	Середня	Mesh/Зірка
LoraWAN	Велика (до 15 км)	Низька	Дуже низьке	Висока	Широка мережа(WAN)
Sigfox	Велика (до 50 км)	Низька	Дуже низьке	Висока	Широка мережа(WAN)
NB-IoT	Велика (до 35 км)	Середня	Низьке	Висока	Широка мережа(WAN)
MQTT	Залежить від транспортного протоколу	Залежить від транспортного протоколу	Залежить від транспортного протоколу	Низька	Транспортний протокол

2.5 Аналіз та огляд різних заходів безпеки IoT приладам

Технологія IoT зробила наше життя ефективнішим у всіх аспектах. Вони також стали невід'ємною складовою бізнес-операцій у всьому світі. Однак, з розвитком і ускладненням глобальної мережі розумних пристроїв продовжують з'являтися нові проблеми з безпекою.

Мережа фізичних пристроїв Інтернету речей інтегрована з датчиками, програмним забезпеченням та іншими технологіями. Ці пристрої підключені до інтернету для отримання і передачі інструкцій, а також для збору та обміну даними з іншими пристроями і системами через інтернет. Через що дуже вразливі до різних кіберзагроз. Тому не менш важливою складовою є забезпечення високого рівня захисту IoT пристроїв. Розглянемо наступні підходи захисту IoT пристроїв.

TLS/SSL: Протоколи безпеки на транспортному рівні (Transport Layer Security) і його попередник Secure Sockets Layer (SSL) забезпечують шифрування та аутентифікацію для захисту передачі даних. Використання HTTPS або захищеного MQTT (MQTT через TLS) є загальним підходом для забезпечення безпеки в IoT-системах. Це гарантує, що дані, які передаються між IoT пристроями та серверами, захищені від перехоплення та модифікації.

OAuth: Протокол OAuth використовується для безпечної аутентифікації та авторизації між пристроями та серверами IoT. Це дозволяє пристроям отримувати обмежений доступ до ресурсів без передачі паролів, що забезпечує додатковий рівень безпеки. OAuth гарантує, що тільки авторизовані пристрої можуть надсилати дані до сервера.

Ключі API та токени: Для аутентифікації з API сервера IoT пристрої можуть використовувати ключі API або токени. Токени можуть бути короткостроковими та змінюватися для додаткової безпеки. Цей метод забезпечує контроль доступу до ресурсів і дозволяє відстежувати використання API.

Цифрові підписи та сертифікати: IoT пристрої можуть підписувати дані цифровими підписами або використовувати цифрові сертифікати для підтвердження своєї ідентичності на сервері. Цифрові підписи гарантують, що дані не були змінені під час передачі, а сертифікати підтверджують автентичність пристроїв.

Брандмауери та системи виявлення вторгнень (IDS): Впровадження брандмауерів і систем виявлення вторгнень як на стороні пристрою, так і на стороні сервера може допомогти виявити і запобігти несанкціонованому доступу та атакам. Брандмауери блокують небажаний трафік, а IDS моніторять мережеву активність та виявляють підозрілі дії.

Вибір заходів безпеки або їх комбінації залежить від поставленого завдання. Якщо дані не дуже чутливі, то можна обмежитися і простим способом захисту. З огляду на те, що даній роботі дані про кількість продуктів на складі не є секретним, то можна використати https-протокол та статичний токена доступу, який присвоюватиметься мікроконтролеру і зберігатиметься в базі даних. Під час кожного надсилання даних на сервер, буде перевірятися цей токен.

3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Загальна архітектура системи

Архітектура системи моніторингу залишків продуктів у ресторані включає кілька ключових компонентів, які взаємодіють між собою для забезпечення ефективного збору, обробки та зберігання даних. Основні компоненти системи такі: клієнти, сервер, база даних, мікроконтролери та ваги. Нижче подана детальна схема архітектури системи (рис.3.1).

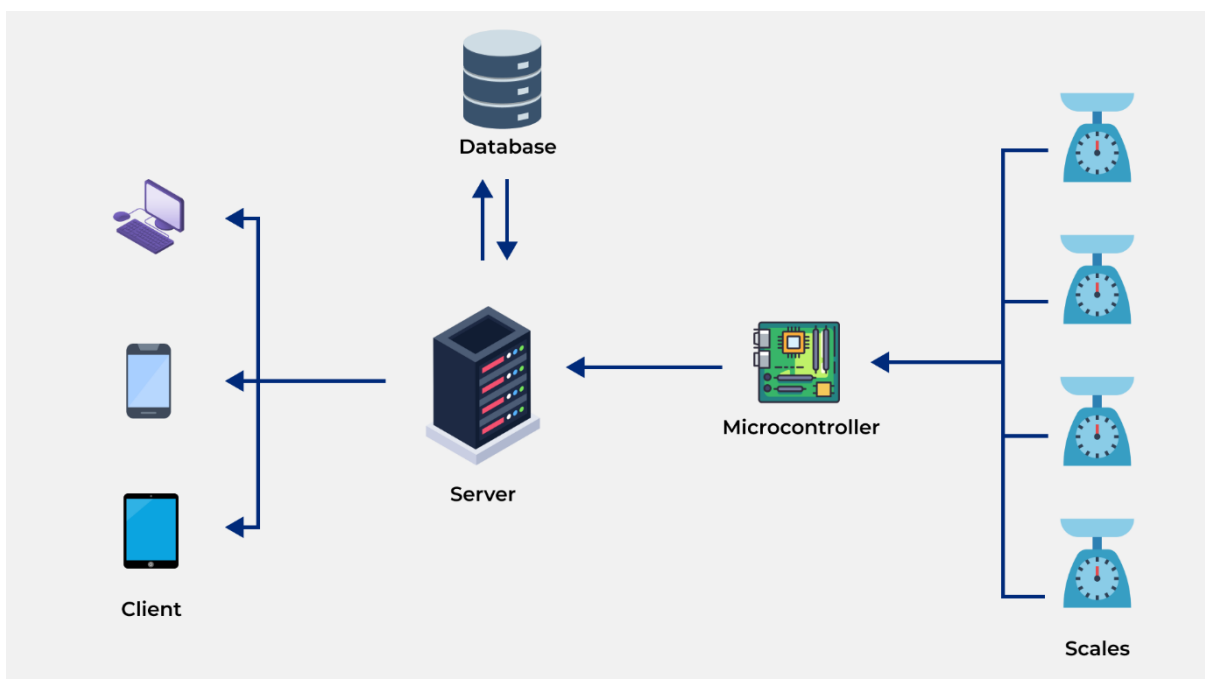


Рисунок 3.1 – Архітектура системи моніторингу залишків продуктів

Клієнти системи – це комп'ютери, смартфони та планшети, які використовуються адміністраторами ресторану для доступу до адміністративної панелі. Вони підключаються до сервера через інтернет для обміну даними. Це забезпечує адміністратору можливість переглядати, аналізувати і керувати запасами у реальному часі.

Сервер є центральною ланкою системи, що відповідає за обробку запитів від клієнтів та мікроконтролерів. Він забезпечує аутентифікацію користувачів, обробку отриманих даних і їх зберігання у базі даних. Сервер

побудований з використанням технологій Node.js та Express.js, що дозволяє досягти високої продуктивності і гнучкості у розробці та підтримці системи.

База даних використовується для зберігання інформації про запаси, даних з тензометричних датчиків ваги та інших необхідних для роботи системи даних. Використовується MongoDB, що забезпечує гнучке і масштабоване зберігання даних, яке може швидко адаптуватися до зміни обсягів і структури даних.

Мікроконтролери відповідають за збір даних з тензометричних датчиків ваги і передачу цих даних на сервер. Використовується Arduino UNO з додатковим модулем ESP8266 для підключення до Wi-Fi мережі. Мікроконтролери зчитують дані з датчиків ваги, обробляють їх та надсилають на сервер за допомогою HTTP-запитів.

Ваги є тензометричними датчиками, які вимірюють вагу продуктів у реальному часі. Дані з цих датчиків передаються через аналогово-цифрові модулі на мікроконтролери, які потім відправляють їх на сервер для подальшої обробки та зберігання. Це дозволяє в режимі реального часу відстежувати стан запасів, оптимізувати процеси зберігання та забезпечувати своєчасне поповнення запасів.

Загальна архітектура системи забезпечує безперервний потік даних між компонентами, що дозволяє в реальному часі відстежувати стан запасів, оптимізувати процеси зберігання і забезпечувати своєчасне поповнення запасів. Це сприяє підвищенню ефективності роботи ресторану, зниженню витрат та покращенню якості обслуговування клієнтів.

Загальну технологію роботи системи можна скласти з наступних етапів:

Етап 1. Приймання товару на склад. При надходженні нових продуктів на склад ресторану, кожна одиниця товару розміщується на ваги, підключені до системи. Тензометричні датчики зчитують вагу товару і передають ці дані на мікроконтролер Arduino UNO, який обробляє інформацію і надсилає її на сервер через модуль ESP8266.

Опис процесу приймання товару на склад:

- товар розміщується на ваги;
- тензOMETричні датчики вимірюють вагу товару;
- мікроконтролер Arduino UNO обробляє дані з датчиків;
- модуль ESP8266 надсилає дані на сервер через Wi-Fi.

Етап 2. Зберігання та моніторинг запасів. Після первинної обробки дані про вагу товару зберігаються в базі даних. Це дозволяє адміністраторам ресторану в реальному часі стежити за кількістю запасів, переглядаючи інформацію через адміністративну панель.

Опис процесу зберігання та моніторингу запасів:

- сервер отримує дані від мікроконтролера;
- дані зберігаються в базі даних MongoDB;
- адміністратори можуть переглядати актуальну інформацію про запаси через клієнтський інтерфейс (комп'ютери, смартфони, планшети).

Етап 3. Використання товару та оновлення даних. Коли товар використовується на кухні, його знову розміщують на ваги, щоб оновити дані про залишки. Мікроконтролер знову зчитує дані, які надсилаються на сервер для оновлення бази даних.

Опис процесу використання товару та оновлення даних:

- товар розміщується на ваги після використання;
- тензOMETричні датчики знову вимірюють вагу;
- мікроконтролер обробляє оновлені дані і надсилає їх на сервер;
- база даних оновлюється, відображаючи нову кількість запасів.

Етап 4. Автоматизація поповнення запасів. Система моніторингу запасів налаштована таким чином, щоб автоматично генерувати повідомлення або замовлення на поповнення запасів, коли кількість товару досягає встановленого мінімуму.

Опис процесу автоматизації поповнення запасів:

- сервер аналізує дані про запаси;

- якщо кількість товару досягає мінімального рівня, система генерує повідомлення про необхідність поповнення запасів;
- повідомлення надсилається відповідальним особам або автоматично створюється замовлення на постачання нових товарів.

Етап 5. Інвентаризація та звітність. Система дозволяє легко проводити інвентаризацію завдяки точному відстеженню ваги товарів у реальному часі. Адміністратори можуть генерувати звіти про стан запасів та аналізувати ефективність використання ресурсів.

Опис процесу інвентаризації та звітності:

- адміністратори запускають процес інвентаризації через адміністративну панель;
- система автоматично збирає дані з усіх ваг і мікроконтролерів;
- генеруються звіти про стан запасів, які можна переглянути через клієнтський інтерфейс.

Цей поетапний процес забезпечує ефективне управління запасами у ресторані, знижуючи витрати на зберігання та мінімізуючи ризики помилок. Автоматизація процесів дозволяє адміністраторам зосередитися на інших важливих аспектах роботи ресторану, підвищуючи загальну ефективність та якість обслуговування клієнтів.

3.2 Детальна схема підключення мікроконтролера до модулів

Детальна схема підключення компонентів системи управління запасами складається з мікроконтролера Arduino UNO, тензометричного датчика HX711, модуля ESP8266 для підключення до Wi-Fi, та 4-бітного цифрового світлодіодного модуля для візуалізації даних. Схема представлена на рисунку 3.2.

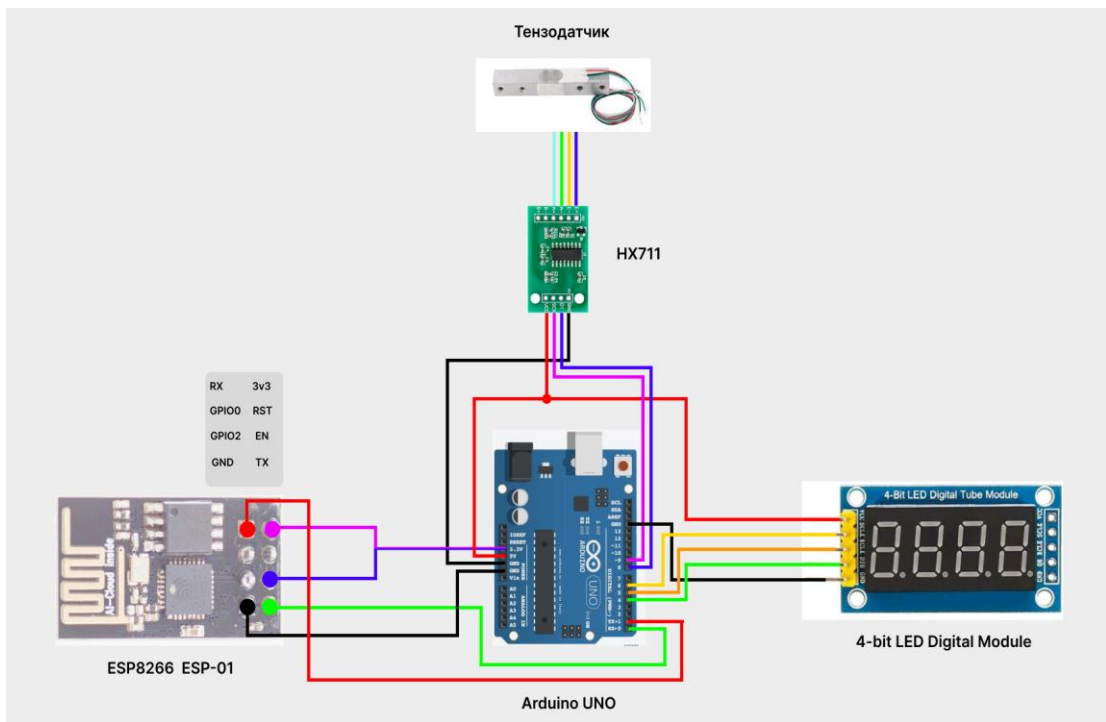


Рисунок 3.2 – Детальна схема підключення модулів до МК

Опис компонентів та їх підключення наведено нижче.

Тензометричний датчик підключається до підсилювача сигналу HX711. Чотири дроти з тензометричного датчика з'єднуються з HX711 відповідно до кольорового кодування: червоний (VCC) до E+, чорний (GND) до E-, білий (S-), зелений (S+).

Підсилювач HX711 підключається до Arduino UNO: VCC до 5V, GND до GND, DT до пін A0, SCK до пін A1, модуль ESP8266.

Модуль ESP8266 використовується для бездротової передачі даних на сервер через Wi-Fi: VCC до 3.3V, GND до GND, RX до пін 2 на Arduino, TX до пін 3 на Arduino, 4-бітний цифровий світлодіодний модуль.

Світлодіодний модуль підключається до Arduino UNO для візуалізації даних про вагу: VCC до 5V, GND до GND, DIO до пін 4 на Arduino, CLK до пін 5 на Arduino, STB до пін 6 на Arduino.

Процес підключення тензометричного датчика до HX711 виглядає наступним чином:

– з'єднання дротів тензометричного датчика з відповідними контактами на підсилювачі HX711;

- підключення HX711 до Arduino UNO;
- з'єднання контактів VCC, GND, DT та SCK підсилювача HX711 з відповідними піном Arduino UNO;
- підключення модуля ESP8266 до Arduino UNO;
- підключення модуля ESP8266 до Arduino, забезпечивши живлення від 3.3V і з'єднання пінів RX та TX з піном Arduino;
- підключення 4-бітного цифрового світлодіодного модуля до Arduino UNO;
- з'єднання VCC та GND модуля з 5V і GND на Arduino, а також з'єднання DIO, CLK та STB з відповідними піном Arduino.

Ця схема підключення забезпечує коректну роботу всіх компонентів системи управління запасами. Тензометричний датчик зчитує вагу товару, дані обробляються підсилювачем HX711 і передаються на мікроконтролер Arduino UNO. Модуль ESP8266 забезпечує передачу даних на сервер, а цифровий світлодіодний модуль дозволяє візуалізувати дані безпосередньо на місці.

Кінцевий прототип схеми підключення має наступний вигляд (рис.3.3).

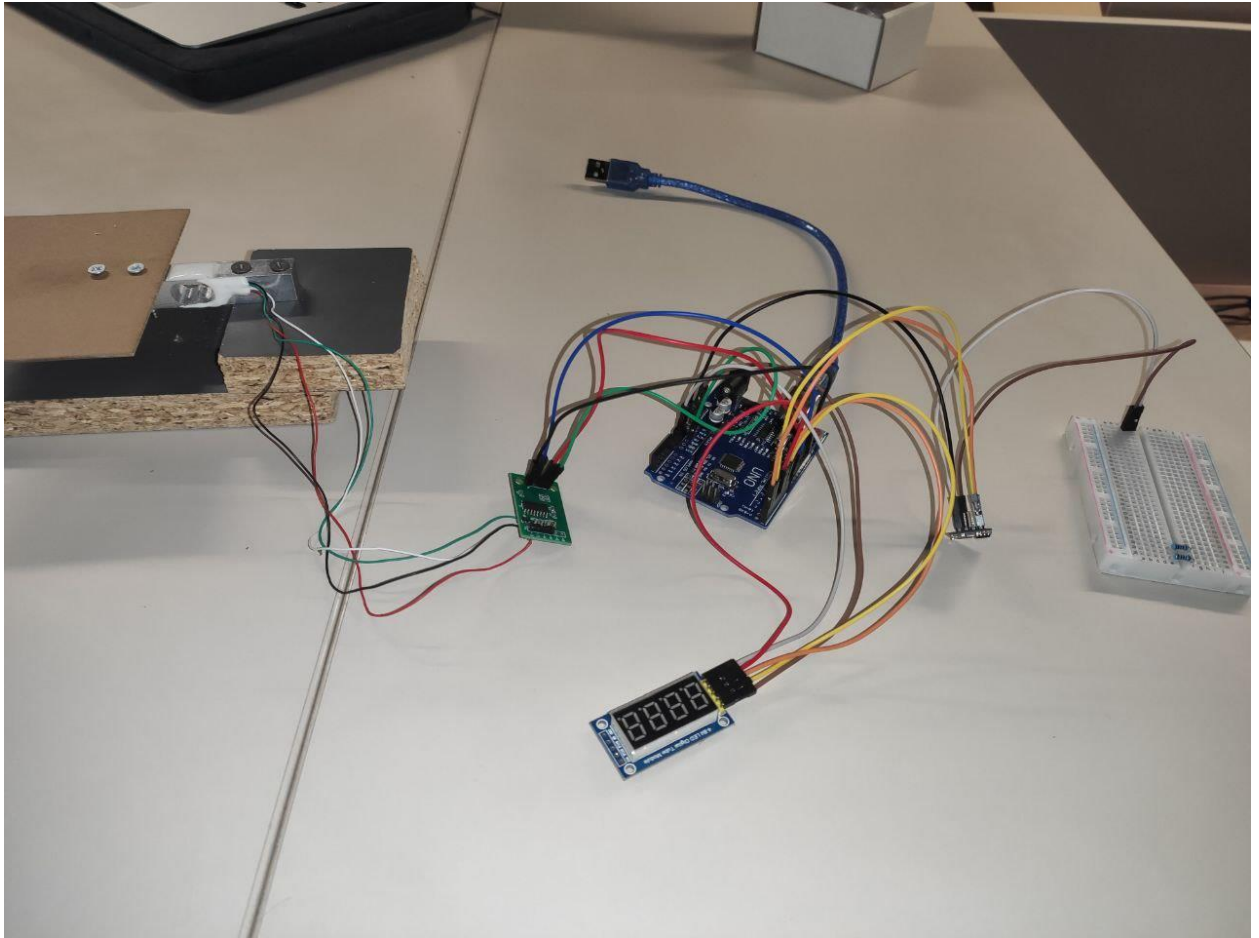


Рисунок 3.3 –Прототип системи

3.3 Проектування серверної частини

Серверна частина нашої системи моніторингу залишків продуктів базується на Express.js популярному вебфреймворку для Node.js. Основною функцією сервера є прийом, обробка та зберігання даних з мікроконтролера, а також забезпечення доступу до цих даних через клієнтський інтерфейс.

Після того, як мікроконтролер отримує дані з тензометричних датчиків, він надсилає їх на сервер за допомогою HTTP-запиту через модуль ESP8266. Запит містить вагу продукту, унікальний ідентифікатор продукту (ID), унікальний ідентифікатор мікроконтролера (microcontrollerID) та статичний токен для автентифікації.

Сервер Express.js отримує HTTP-запит на визначену кінцеву точку, наприклад, **“/update-weight”**, і обробляє його відповідним маршрутизатором у фреймворку Express.js. На цьому етапі сервер перевіряє автентичність запиту, використовуючи статичний токен, що міститься у запиті. Якщо токен валідний, сервер продовжує обробку запиту, перевіряючи коректність даних, таких як формат ідентифікатора продукту (ID), ідентифікатора мікроконтролера (microcontrollerID) та значення ваги. Це робиться для забезпечення того, що дані є точними і можуть бути безпечно збережені у базі даних.

Після успішної перевірки даних сервер зберігає їх у базі даних MongoDB. База даних організована таким чином, що кожен мікроконтролер та підключені до нього датчики мають свої власні записи. Це дозволяє системі відстежувати, які датчики підключені до конкретного мікроконтролера, і забезпечувати доступ до даних про всі продукти, пов'язані з цим мікроконтролером.

При надходженні даних від мікроконтролера сервер перевіряє, чи існує запис для даного мікроконтролера у базі даних. Якщо запис існує, сервер оновлює дані для відповідних датчиків. Якщо запису немає, створюється новий запис для мікроконтролера та пов'язаних з ним датчиків.

Для кожного мікроконтролера у базі даних зберігається унікальний ідентифікатор (microcontrollerID) та статичний токен для автентифікації. До мікроконтролера прив'язані записи тензометричних датчиків, кожен з яких має свій унікальний ідентифікатор (ID) та дані про вагу продукту. Це дозволяє легко відслідковувати, які датчики підключені до конкретного мікроконтролера, та отримувати інформацію про всі продукти, пов'язані з цим мікроконтролером.

Після успішного збереження даних у базі даних сервер відправляє відповідь назад на мікроконтролер, підтверджуючи, що дані були прийняті і збережені. Це дозволяє мікроконтролеру продовжувати свою роботу, знаючи, що дані були успішно передані.

3.4 Опис моделей бази даних

Далі буде розглянуто структуру моделей бази даних, які використовуються в системі моніторингу залишків продуктів. Усі моделі зберігаються в MongoDB і представляють різні аспекти системи, включаючи продукти, мікроконтролери, користувачів та токени авторизації. Нижче наведено описи чотирьох основних моделей: Product, MicroController, User та Token.

Модель Product представляє товари, які зберігаються на складі та контролюються за допомогою тензOMETричних датчиків. Кожен продукт має унікальний ідентифікатор, назву, вагу, одиниці виміру та категорію (табл.3.1).

Таблиця 3.1 – Опис моделі Product

Атрибут	Тип	Опис
Id	ObjectId	Унікальний ідентифікатор продукту
name	string	Назва продукту
weight	number	Вага продукту
measureUnits	string	Одиниці виміру ваги
category	string	Категорія продукту

Модель MicroController представляє мікроконтролери, які підключені до тензOMETричних датчиків і відповідають за збір даних. Кожен мікроконтролер має унікальний ідентифікатор, назву та статичний токен для автентифікації (табл.3.2).

Таблиця 3.2 – Опис моделі MicroController

Атрибут	Тип	Опис
id	ObjectId	Унікальний ідентифікатор мікроконтролера
name	string	Назва мікроконтролера
staticToken	string	Статичний токен для автентифікації

Модель User представляє користувачів системи, які мають доступ до даних про запаси та можуть взаємодіяти з системою через клієнтський інтерфейс. Кожен користувач має унікальний ідентифікатор, ім'я користувача, пароль та роль (табл.3.3).

Таблиця 3.3 – Опис моделі User

Атрибут	Тип	Опис
id	ObjectId	Унікальний ідентифікатор мікроконтролера
username	string	Ім'я користувача
password	string	Пароль користувача
role	string	Роль користувача(наприклад, адміністратор, менеджер)

Модель Token використовується для зберігання токенів авторизації користувачів, які забезпечують доступ до системи. Цей токен не має відношення до мікроконтролера і використовується виключно для авторизації користувачів(табл.3.4).

Таблиця 3.4 – Опис моделі Token

Атрибут	Тип	Опис
Id	ObjectId	Унікальний ідентифікатор мікроконтролера
value	string	Значення токена
userId	ObjectId	Ідентифікатор користувача, до якого прив'язаний токен

Моделі User та Token взаємопов'язані, оскільки кожен токен прив'язаний до конкретного користувача системи, забезпечуючи його автентифікацію. З іншого боку, модель MicroController має статичний токен, який використовується для автентифікації запитів від мікроконтролерів до сервера.

3.5 Загальний огляд клієнтської частини

Клієнтська частина системи моніторингу залишків продуктів реалізована за допомогою фреймворку React.js. Вона забезпечує зручний інтерфейс для взаємодії користувачів з системою, надаючи доступ до актуальної інформації про запаси, а також можливість додавання, редагування та видалення даних. Основними користувачами системи є адміністратори та менеджери, які мають різні рівні доступу до функціоналу.

Першим кроком для користувача є процес авторизації. Після введення імені користувача та пароля, клієнтська частина надсилає запит на сервер, де відбувається перевірка введених даних. Якщо авторизація успішна, користувач отримує токен доступу, який зберігається в браузері для подальших запитів до сервера. Це забезпечує захищений доступ до приватної частини застосунку.

Після авторизації користувач потрапляє на панель управління (dashboard). Це центральний елемент клієнтського інтерфейсу, де

відображається основна інформація про запаси та надаються інструменти для управління ними. Панель управління включає наступні компоненти:

- список категорій: відображає всі категорії товарів, наявних у системі. Користувач може переглядати деталі кожної категорії, редагувати їх або видаляти;

- список продуктів: відображає всі продукти, підключені до мікроконтролерів. Користувач може переглядати деталі кожного продукту, включаючи його вагу, одиниці виміру, категорію та останнє оновлення даних. Також є можливість редагувати та видаляти продукти;

- додавання нових категорій та продуктів: користувач може додавати нові категорії та продукти у систему. При додаванні нового продукту, користувач вводить назву, вагу, одиниці виміру, вибирає категорію та прив'язує продукт до відповідного мікроконтролера.

У панелі управління також є розділ для управління мікроконтролерами. Користувач може переглядати список всіх мікроконтролерів, зареєстрованих у системі, а також додавати нові мікроконтролери. При додаванні нового мікроконтролера, користувач вказує його назву та статичний токен, який буде використовуватися для автентифікації запитів від цього мікроконтролера.

Окрім основних функцій, клієнтська частина включає також наступні додаткові можливості:

- пошук та фільтрація: користувач може здійснювати пошук за назвою продукту або категорії, а також фільтрувати продукти за різними параметрами;

- сповіщення: система сповіщень інформує користувача про успішні або невдалі операції, а також про важливі події, такі як низький рівень запасів або помилки з мікроконтролерами;

- аналіз даних: панель управління надає інструменти для аналізу даних про залишки продукції, включаючи графіки та діаграми, які

допомагають візуалізувати динаміку змін ваги продуктів та інші важливі метрики.

3.6 Висновки до розділу 3

У третьому розділі розглянуто процес проектування та реалізації системи моніторингу залишків продуктів, використовуючи технології IoT. Визначено ключові компоненти системи та їх взаємодію, забезпечено надійне підключення мікроконтролерів до тензометричних датчиків, розроблено серверну частину на базі Express.js для обробки та зберігання даних у MongoDB, а також створено зручний кінтлієнтський інтерфейс на базі React.js. Це дозволило створити ефективну, безпечну та зручну у використанні систему, яка значно підвищує ефективність моніторингу залишків продуктів, знижує витрати та покращує обслуговування клієнтів.

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

Для оцінки точності та ефективності роботи розробленої системи моніторингу залишків продуктів, було створено тестове середовище, яке включає мікроконтролери з підключеними датчиками ваги, серверну частину на базі Express.js та клієнтську частину на основі React.js. Система була протестована в умовах, максимально наближених до реальних, щоб забезпечити точні результати та оцінку її продуктивності.

4.1 Тестування апаратної частини

Було проведено тестування апаратної частини системи, зокрема підключення тензодатчиків до мікроконтролерів та їх взаємодія з сервером.

Датчик ваги був протестован на:

- перевірку точності вимірювань ваги;
- вимірювання стабільності сигналу;
- оцінку похибок вимірювання.

Мікроконтролер був протестован на:

- перевірку стабільності передачі даних на сервер;
- вимірювання швидкості передачі даних;
- оцінку надійності підключення.

4.2 Тестування серверної частини

Тестування серверної частини включало перевірку коректності обробки запитів від мікроконтролерів, а також збереження та оновлення даних у базі даних MongoDB.

Тестування обробки запитів:

- перевірка коректності автентифікації та авторизації запитів;

- тестування обробки даних від різних мікроконтролерів.

Тестування бази даних:

- перевірка коректності збереження даних;
- оцінка продуктивності бази даних під навантаженням;
- вимірювання часу відповіді на запити.

4.3 Тестування клієнтської частини

Було проведено тестування клієнтського інтерфейсу для забезпечення зручності користування та коректного відображення даних.

Тестування інтерфейсу:

- перевірка функціональності форм додавання, редагування та видалення продуктів та категорій;
- тестування відображення даних у реальному часі;
- оцінка зручності користування інтерфейсом.

4.4 Результати тестування

Для оцінки точності вимірювань та коректності роботи системи були використані метрики з бібліотеки `sklearn.metrics`: `mean_absolute_error` та `mean_squared_error`.

Для оцінки точності вимірювань було розраховані наступні показники:

- *MAPE* – середня абсолютна похибка в процентах ;
- *MAE* – середня абсолютна похибка;
- *MSE* – середня квадратична похибка;
- *RMSE* – квадратний корінь з середньоквадратичної помилки;
- *ME* – середня похибка;
- *SD* – стандартне відхилення.

Середню абсолютну похибку в процентах $MAPE$ розраховують за формулою (4.1)

$$MAPE = \frac{1}{N} \sum_{t=1}^N \frac{|X(t) - \hat{X}(t)|}{X(t)} \cdot 100\%, \quad (4.1)$$

де $X(t)$ – фактичне значення показника;

$\hat{X}(t)$ – прогнозне значення показника.

Середню абсолютну похибку MAE розраховують за формулою (4.2)

$$MAE = \frac{1}{N} \sum_{t=1}^N |X(t) - \hat{X}(t)| \quad (4.2)$$

Середню квадратичну похибку MSE розраховують за формулою (4.3)

$$MSE = \frac{1}{N} \sum_{t=1}^N (X(t) - \hat{X}(t))^2 \quad (4.3)$$

Квадратний корінь з середньоквадратичної помилки $RMSE$ розраховують за формулою (4.4)

$$RMSE = \sqrt{MSE} \quad (4.4)$$

Середню похибку ME розраховують за формулою (4.5)

$$ME = \frac{1}{N} \sum_{t=1}^N (X(t) - \hat{X}(t)) \quad (4.5)$$

Стандартне відхилення SD розраховують за формулою (4.6)

$$SD = \sqrt{\frac{1}{N} \sum_{t=1}^N \left(\hat{X}(t) - ME \right)^2},$$

(4.6)

де ME – середня похибка,що визначена за формулою (545).

Результати оцінки точності вимірювань показані в таблиці 4.1

Таблиця 4.1 – Результати оцінки точності вимірювань

Параметр	Значення
MAE	0.05 кг
MSE	0.003 кг ²
RMSE	0.055 кг
ME	0.01 кг
SD	0.07 кг

4.5 Можливості розширення та покращення системи

Для покращення ефективності системи та її масштабованості, необхідно розглянути можливість розширення апаратної частини. Одним з ключових напрямків є збільшення кількості мікроконтролерів, до яких можна підключити більше тензодатчиків. Це дозволить системі обробляти дані від більшої кількості вагових платформ, що є критично важливим для великих складів або підприємств з великою кількістю різних продуктів. Впровадження додаткових мікроконтролерів також сприятиме підвищенню надійності системи, оскільки зменшується навантаження на кожен окремий мікроконтролер.

Для забезпечення автоматизації процесів закупівлі продукції, варто розглянути можливість впровадження алгоритмів, які будуть аналізувати дані про запаси та прогнозувати необхідність поповнення складу. Це може включати використання методів машинного навчання для прогнозування попиту та автоматичне генерування замовлень на основі аналізу історичних даних та поточного стану запасів. Такі алгоритми дозволять значно знизити витрати на управління запасами та уникнути ситуацій, коли продукція закінчується.

Адміністративна панель може бути розширена для надання більшого функціоналу менеджерам складу та адміністраторам системи. Наприклад, можна додати модуль для управління замовленнями клієнтів. Це дозволить клієнтам робити замовлення онлайн, вибираючи лише ті товари, які є в наявності. Система автоматично визначатиме доступність товарів на основі даних про запаси, що дозволить уникнути ситуацій, коли клієнти замовляють товари, яких немає на складі.

Крім того, можна впровадити систему сповіщень, яка буде інформувати менеджерів про критичні рівні запасів, необхідність поповнення або інші важливі події. Це підвищить оперативність реагування на зміну ситуації на складі та допоможе уникнути можливих проблем.

Для підвищення ефективності управління запасами варто розглянути можливість інтеграції системи з іншими інформаційними системами, такими як ERP (Enterprise Resource Planning) та CRM (Customer Relationship Management). Це дозволить забезпечити більш тісну взаємодію між різними аспектами бізнесу та автоматизувати більшу кількість процесів. Наприклад, інтеграція з ERP системою дозволить автоматично синхронізувати дані про залишки, замовлення та постачання, що зменшить ризик помилок та підвищить загальну ефективність роботи компанії.

Подальший розвиток системи може включати впровадження нових технологій, таких як штучний інтелект та інтернет речей. Наприклад, можна використовувати алгоритми штучного інтелекту для більш точного

прогнозування попиту та оптимізації запасів. Інтернет речей може бути використаний для підключення більшої кількості сенсорів та пристроїв, що дозволить отримувати більш точні та оперативні дані про стан запасів.

Розробка системи контролю залишків продукції – це лише перший крок для автоматизації процесів ресторану. Цей проект має великий потенціал для розширення та покращення. Впровадження додаткових мікроконтролерів, покращення серверної частини, розширення функціоналу адміністративної панелі та інтеграція з іншими системами дозволять зробити систему більш ефективною та надійною. Використання нових технологій, таких як штучний інтелект та інтернет речей, відкриває нові можливості для автоматизації та оптимізації процесів управління запасами, що сприятиме подальшому розвитку бізнесу та підвищенню його конкурентоспроможності.

ВИСНОВКИ

Результатом виконання дипломної кваліфікаційної роботи є система моніторингу залишків продукції за допомогою технологій IoT, тензодатчиків, мікроконтролерів Arduino UNO з модулем ESP8266, розроблена програма для прийому та обліку товарів.

У ході роботи були вирішені наступні завдання:

- дослідження існуючих технологій реалізації систем моніторингу залишків продукції;
- аналіз апаратних рішень на основі IoT;
- проектування та реалізація прототипу системи моніторингу залишків продукції;
- проведення експериментального дослідження розробленої системи.

У першому розділі був проведений огляд проблем моніторингу залишків продукції та знайдені шляхи їх вирішення; розглянуто способи управління запасами без використання IoT; визначено комплексний підхід до впровадження IoT та розглянуто загальну структуру системи.

У другому розділі було проаналізовано технологічні стеки для розробки подібних систем, таких як LAMP, MEAN, Django та .NET. Також було проведено аналіз апаратних рішень та різних платформ для збору даних про вагу, таких як Raspberry Pi та ESP8266, і датчиків ваги.

У третьому розділі були розроблені архітектура системи та загальна схема підключення мікроконтролера Arduino з модулями. Описано процес проектування серверної частини та моделі баз даних, а також клієнтську частину для адміністрування та управління системою.

У четвертому розділі було проведено експериментальне дослідження розробленої системи, включаючи тестування точності та надійності зібраних даних. Описано методи оцінки точності та проведені вимірювання, які підтвердили ефективність розробленої системи.

У підсумку, розроблено прототип системи моніторингу залишків продукції з використанням IoT технологій, що дозволяє автоматизувати процеси прийому та обліку товарів, знизити рівень помилок та підвищити точність даних. Система забезпечує точне та швидке оновлення інформації про залишки, що підвищує ефективність управління логістичними процесами. У майбутньому, система може бути розширена додаванням більшої кількості мікроконтролерів та датчиків, інтеграцією алгоритмів автоматичної закупівлі продукції, а також покращенням адміністративної панелі з можливістю онлайн-замовлень клієнтів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The Role of IoT Sensing-as-a-Service in Combating Food Waste [Electronic resource]. – Access mode: <https://blog.smartsense.co/sensing-as-a-service-combating-food-waste>
2. Методи управління запасами [Електронний ресурс]. – Режим доступу: <https://mltechsoft.com/insights/blog/inventory-control-methods-fifo-lifo-abc-analysis>
3. SensrTrx [Electronic resource]. – Access mode: <https://www.mingosmartfactory.com/sensrtrx-not-sensor-tracks-manufacturers/>
4. ParTech [Electronic resource]. – Access mode: <https://partech.com/about/>
5. MarketMan [Electronic resource]. – Access mode: <https://www.marketman.com/about-us>
6. LAMP vs. MEAN/MERN: Which development stack is right for you? [Electronic resource]. – Access mode: <https://www.remotely.works/blog/lamp-vs-mean-mern-which-development-stack-is-right-for-you>
7. Linux [Electronic resource]. – Access mode: <https://uk.wikipedia.org/wiki/Linux>
8. Apache [Electronic resource]. – Access mode: https://uk.wikipedia.org/wiki/Apache_HTTP_Server
9. MySQL [Electronic resource]. – Access mode: <https://uk.wikipedia.org/wiki/MySQL>
10. PHP [Electronic resource]. – Access mode: <https://uk.wikipedia.org/wiki/PHP>
11. Mongo DB [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/MongoDB>
12. Express js [Electronic resource]. – Access mode: <https://expressjs.com/>
13. Angular [Electronic resource]. – Access mode: <https://angularjs.org/>
14. 10. React js [Electronic resource]. – Access mode: <https://react.dev/>

15. Node js [Electronic resource]. – Access mode:
<https://nodejs.org/en/about>
17. Python [Electronic resource]. – Access mode:
<https://www.python.org/about/>
18. Django [Electronic resource]. – Access mode:
<https://www.djangoproject.com/foundation/>
19. PostgreSQL [Electronic resource]. – Access mode:
<https://www.postgresql.org/about/>
20. ASP.NET vs Django - A Detailed Comparison [Electronic resource]. –
Access mode: <https://flexiple.com/compare/asp-dot-net-vs-django>
21. C# [Electronic resource]. – Access mode:
[https://en.wikipedia.org/wiki/C_Sharp_\(programming_language\)](https://en.wikipedia.org/wiki/C_Sharp_(programming_language))
22. F# [Electronic resource]. – Access mode: <https://fsharp.org/>
23. Мікроконтролер Raspberry Pi [Electronic resource]. – Access mode:
<https://www.amazon.nl/-/en/Raspberry-Model-Desktop-900MHz-Linux/dp/B00T2U7R7I>
24. Мікроконтролер ESP8266 [Електронний ресурс]. – Режим доступу:
<https://joy-it.net/en/products/SBC-NodeMCU>
25. Мікроконтролер Arduino UNO [Електронний ресурс]. – Режим доступу:
https://elektronicavoorjou.nl/product/arduino-uno-rev3-atmega328/?gad_source=1&gclid=CjwKCAjw7NmzBhBLEiwAxrHQ-U9VNpaB8fIXMdc6PddeVPsSy3u0EWK1PFcQYxUwoZ6zn0S6b1SZthoC0RgQAvD_BwE
26. Тензометричний датчик [Електронний ресурс]. – Режим доступу:
<https://micronrobotics.com/product/3kg-load-cell-electronic-weighing-scale-sensor/>
27. Ємнісний датчик [Електронний ресурс]. – Режим доступу:
<http://www.tnstruments.com/ProductDetails.aspx?PID=19794siTlTE%3D&CType=ZgDIC937E8s%3D&CID=y57tQ3Va0b4%3D>

28. П'єзоелектричний датчик [Електронний ресурс]. – Режим доступу:
<https://www.kistler.com/HU/en/p/piezoelectric-press-force-load-cell-9323a/000000000018007665>

29. Індуктивний датчик [Електронний ресурс]. – Режим доступу:
<https://teko-com.ru/product/isbm-w48s8-31ns-1-5-lz-1h-v-7.html>

ДОДАТОК А

Текст програми

```
#include <HX711_ADC.h>
#include <EEPROM.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

// Дані для підключення до Wi-Fi
const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";

// Піни HX711
const int HX711_dout = 8; // мкк > піни HX711 dout
const int HX711_sck = 9; // мкк > піни HX711 sck
const int calVal_eeepromAdress = 0; // Адреса EEPROM для значення
калібрування
float calibrationValue; // значення калібрування

HX711_ADC LoadCell(HX711_dout, HX711_sck);

// Піни регістру зсуву
unsigned int ST_CP = 5; // RCLK
unsigned int SH_CP = 6; // SCLK
unsigned int DS = 4; // DIO

// Значення цифр для 7-сегментного дисплея
int hexDigitValue[] = {
    0xFC, // Сегменти для відображення 0
    0x60, // Сегменти для відображення 1
```

```

0xDA, // Сегменти для відображення 2
0xF2, // Сегменти для відображення 3
0x66, // Сегменти для відображення 4
0xB6, // Сегменти для відображення 5
0xBE, // Сегменти для відображення 6
0xE0, // Сегменти для відображення 7
0xFE, // Сегменти для відображення 8
0xF6  // Сегменти для відображення 9
};

unsigned int lastWeight = 0;
const int deviceId = 123; // Ідентифікатор пристрою
const String jwtToken = "your_jwt_token"; // JWT токен

void setup() {
  Serial.begin(57600);
  delay(10);
  Serial.println("Starting...");

  // Підключення до Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Ініціалізація HX711
  LoadCell.begin();
  unsigned long stabilizingtime = 2000;

```

```

boolean _tare = true;
LoadCell.start(stabilizingtime, _tare);
if (LoadCell.getTareTimeoutFlag() || LoadCell.getSignalTimeoutFlag()) {
    Serial.println("Timeout, check MCU>HX711 wiring and pin designations");
    while (1);
} else {
    LoadCell.setCalFactor(1.0);
    Serial.println("Startup is complete");
}

// Завантаження значення калібрування з EEPROM
EEPROM.begin(512);
EEPROM.get(calVal_eepromAdress, calibrationValue);
LoadCell.setCalFactor(calibrationValue);
Serial.print("Calibration value loaded from EEPROM: ");
Serial.println(calibrationValue);

// Встановлення піни регістру зсуву як виходи
pinMode(ST_CP, OUTPUT);
pinMode(SH_CP, OUTPUT);
pinMode(DS, OUTPUT);
}

void loop() {
    static boolean newDataReady = 0;
    const int serialPrintInterval = 1000; // Інтервал для виведення даних в
    мілісекундах
    static unsigned long t = 0;
    static unsigned long lastJsonSend = 0;

```

```
const unsigned long jsonSendInterval = 30000; // Інтервал для відправки даних
JSON в мілісекундах
```

```
// Перевірка нових даних/початок наступного перетворення
```

```
if (LoadCell.update()) newDataReady = true;
```

```
// Отримання згладженого значення з набору даних
```

```
if (newDataReady) {
```

```
    if (millis() > t + serialPrintInterval) {
```

```
        float weight = LoadCell.getData();
```

```
        Serial.print("Load_cell output val: ");
```

```
        Serial.println(weight);
```

```
        // Оновлення останнього значення ваги
```

```
        lastWeight = abs((int)weight);
```

```
        newDataReady = 0;
```

```
        t = millis();
```

```
    }
```

```
}
```

```
// Постійне відображення останнього значення ваги
```

```
displayWeight(lastWeight);
```

```
// Відправка даних JSON кожну хвилину
```

```
if (millis() - lastJsonSend >= jsonSendInterval) {
```

```
    sendJsonData(deviceId, lastWeight);
```

```
    lastJsonSend = millis();
```

```
}
```

```
}
```

```

/* Встановлення значення цифри на 7-сегментний дисплей
*
*   row = номер ряду цифри для відображення 0..3 де 3 - найбільш значуща
(лівий) цифра дисплея
*   digit = значення від 0 до 15
*   decimalPoint = true/false : чи світити десяткову точку
*/
void setDigit(unsigned int row, unsigned int digit, boolean decimalPoint) {
    unsigned int rowSelector;
    unsigned int data;

    rowSelector = bit(3-row)<<4;
    data = ~hexDigitValue[digit & 0xF];
    if (decimalPoint)
        data &= 0xFE;

    // Перші 8 бітів даних до другого 74HC595
    shiftOut(DS, SH_CP, LSBFIRST, data);

    // Тепер зрушуємо 4 біти ряду до першого 74HC595 і фіксуємо
    digitalWrite(ST_CP, LOW);
    shiftOut(DS, SH_CP, LSBFIRST, rowSelector);
    digitalWrite(ST_CP, HIGH);
}

/* Відображення числа як 4-значного десяткового числа на дисплеї
*   Примітка: це мультиплексування, тому потрібно постійно викликати цю
функцію,
*   інакше буде світитися лише одна цифра.

```

```

*/
void displayNumber(unsigned int number) {
    for (unsigned int i=0; i<4; i++) {
        setDigit(i, number % 10, false); // відображення крайньої правої цифри (1
цифра)
        number = number / 10; // перехід до наступної цифри
        delay(1);
    }
}

// Відображення ваги на 7-сегментному дисплеї
void displayWeight(int weight) {
    displayNumber(weight);
}

// Відправка даних JSON на серійний порт та через HTTP
void sendJsonData(int id, int weight) {
    String jsonData = "{\"id\": " + String(id) + ", \"weight\": " + String(weight) + "}";
    Serial.println(jsonData);

    if (WiFi.status() == WL_CONNECTED) { // Перевірка стану з'єднання Wi-Fi
        HTTPClient http;
        http.begin("http://localhost:7111/api/update-weight"); // Вказівка URL
призначення
        http.addHeader("Content-Type", "application/json"); // Вказівка заголовка
типу вмісту
        http.addHeader("Authorization", "Bearer " + jwtToken); // Додавання
заголовка авторизації з JWT токеном

        int httpResponseCode = http.POST(jsonData); // Відправка запиту

```

```
if (httpResponseCode > 0) {  
    String response = http.getString(); // Отримання відповіді на запит  
    Serial.println(httpResponseCode); // Друк коду відповіді  
    Serial.println(response); // Друк відповіді запиту  
} else {  
    Serial.print("Error on sending POST: ");  
    Serial.println(httpResponseCode);  
}  
  
http.end(); // Звільнення ресурсів  
} else {  
    Serial.println("Error in WiFi connection");  
}  
}
```