Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Програмно-технічний засіб керування кіберфізичноою системою "Інвентаризація для малого бізнесу"
Назва теми

КвРКІ 220031.22.01.13 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент IV курсу, група КІ2с-22-1 _____ Віталій ОСТРОВСЬКИЙ
Підпис — Ініціали, прізвище

Керівник _____ Євген ФЕДОРОВ
Підпис, дата — Ініціали, прізвище

Нормоконтролер _____ Тетяна КИСІЛЬ
Підпис, дата — Ініціали, прізвище

До захисту допускаю:
зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Ольга ПАВЛОВА
Підпис — Ініціали, прізвище

«5» червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Зав. кафедри Ольга ПАВЛОВА

" 10 " 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Островський Віталій Валерійович
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Програмно-технічний засіб керування кіберфізичноою системою "Інвентаризація для малого бізнесу

Керівник проекту (роботи) Євген ФЕДОРОВ, д.т.н.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 07.02.2025 р. № 23

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2025 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Система управління інвертамрем для малого бізнесу, та поставновка задачі щодо її удосконвлення

Проектування системи управління інвертарем для малого бізснесу

Програмно-апаратна реалцізація системи управління інвертарем для малого бізнесу

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Лістинг коду ПЗ

Алгоритми роботи системи

Апаратне забезпечення проєкту

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийн
Нормоконтроль	Тетяна КИСІЛЬ, доцент кафедри КПС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КПС		

7. Дата видачі завдання «_10_»_____01_____2025 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Приміт
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2025	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2025	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2025	виконано
4	Робота над розділом 2 – проведено повноцінне технічне обгрунтування та практичне проєктування системи управління інвентарем для потреб малого бізнесу	01.04.2025	виконано
5	Робота над розділом 3 – повноцінну демонстрацію роботи системи управління інвентарем	29.04.2025	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2025	виконано
7	Попередній захист ВКР	26.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2025 року	

Студент _____ Віталій ОСТРОВЬКИЙ
Підпис Ініціали, прізвище

Керівник роботи _____ Євген ФЕДОРОВ
Підпис Ініціали, прізвище

№ рядка	формат	Позначе
1		КвРКІ 220031.2
2		КвРКІ 220031.2
3		КвРКІ 220031.2
4		КвРКІ 220031.2

Зм	Арк	№ докум	Підпис	Д
Розробив		Островський		05
Перевір.		Федоров		
Н. контр.		Кисіль		05
Затв.		Павлова		

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 220031.22.01.13 ПЗ	Пояснювальна записка	62		
			Графічні матеріали			
2		КвРКІ 220031.22.01.13 Е8	Лістинг коду ПЗ	1		
3		КвРКІ 220031.22.01.13 Е8	Алгоритми роботи	1		
			системи			
4		КвРКІ 220031.22.01.13 Е8	Апаратне забезпечення	1		
			проєкту			

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 220031.22.01.13 ВП		
Розробив		Островський		05.06.25	Відомість проекту	Літера	Аркуш	Аркушів
Перевір.		Федоров				У	1	1
Н. контр.		Кисіль		05.06.25		ХНУ, КІ2с-22-1		
Затв.		Павлова						

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмно-технічний засіб керування кіберфізичною системою“Інвентаризація для малого бізснесу».

Автор роботи: Островський Віталій.

Керівник роботи: Федоров Євген Євгенович.

Пояснювальна записка: 62 с., 21 рис., 1 табл., 3 дод., 50 джерел.

Графічна частина: 3 креслення.

Метою роботи є розробка кіберфізичної системи управління інвентарем, адаптованої до умов малого бізнесу, яка об’єднує прості технічні засоби з програмним забезпеченням, що реалізує облік товарів на основі сканування штрих-кодів.

Об’єктом дослідження є процеси ведення обліку товарно-матеріальних цінностей у середовищі малого бізнесу.

Предметом дослідження є апаратно-програмна система автоматизованого обліку інвентаря, побудована на основі сканера штрих-кодів GM65, мікроконтролера Arduino Uno та програмного забезпечення, розробленого мовою Python.

Під час проведення даного дослідження був використаний метод систематичного огляду літератури для вивчення і аналізу предметної області даного дослідження з текстових джерел інформації.

Підпис студента

30.05.2025

Дата

ЗМІСТ

ВСТУП

Актуальність дослідження. У сучасних умовах цифрової трансформації майже всі галузі людської діяльності зазнають інтенсивної автоматизації. Особливо це помітно в сфері управління ресурсами та логістики, де точність, швидкість та ефективність мають критичне значення. Однією з ключових ланок таких процесів виступає управління інвентарем складна, багаторівнева задача, яка охоплює облік надходжень, переміщень, зберігання та витрат матеріальних цінностей. Якщо для великих корпорацій подібні завдання давно вирішуються за допомогою масштабних ERP-систем, то малі підприємства часто змушені шукати компромісні рішення, які поєднують доступність, простоту та функціональність.

На сьогодні малі бізнеси нерідко покладаються на ручний облік товарів, використання електронних таблиць або частково автоматизованих інструментів, які не забезпечують належного рівня точності, не захищають від дублювання чи втрати даних, і не адаптовані до умов зростаючої номенклатури або зміни форм ведення господарської діяльності. Це створює значні ризики, пов’язані з помилками у звітності, збоєм у постачанні, перевитратами ресурсів, втратою прибутку та уповільненням обслуговування клієнтів.

У цьому контексті розробка недорогих, локальних та гнучких рішень для автоматизованого управління інвентарем набуває особливого значення. Кіберфізичні системи, що поєднують фізичні пристрої збору даних (зокрема сканери штрих-кодів), мікроконтролери (наприклад, Arduino) та комп’ютерні програми, створюють нові можливості для малого бізнесу. Такі системи дозволяють мінімізувати вплив людського фактора, забезпечити швидке зчитування ідентифікаторів товарів, фіксувати їх у базі даних, проводити аналіз і візуалізацію інформації в реальному часі.

Враховуючи стрімке зростання кількості малих підприємств, зокрема у сфері роздрібної торгівлі, обслуговування, складування, інтернет-магазинів тощо, впровадження подібних систем є не лише доцільним, а й необхідним кроком до

підвищення конкурентоспроможності та стійкості бізнесу. При цьому важливою перевагою стає можливість повної автономності, тобто відсутність потреби в дорогому серверному обладнанні, ліцензованому ПЗ або хмарних підписках. Система має працювати локально, на будь-якому комп'ютері, з мінімальними витратами на впровадження.

Метою роботи є розробка кіберфізичної системи управління інвентарем, адаптованої до умов малого бізнесу, яка об'єднує прості технічні засоби з програмним забезпеченням, що реалізує облік товарів на основі сканування штрих-кодів. У межах цього проєкту передбачено створення архітектури, програмної логіки, графічного інтерфейсу, бази даних і механізмів комунікації між апаратними та програмними модулями.

Об'єктом дослідження є процеси ведення обліку товарно-матеріальних цінностей у середовищі малого бізнесу.

Предметом дослідження є апаратно-програмна система автоматизованого обліку інвентаря, побудована на основі сканера штрих-кодів GM65, мікроконтролера Arduino Uno та програмного забезпечення, розробленого мовою Python.

Розроблена система покликана демонструвати не лише функціональність і коректність роботи окремих компонентів, а й забезпечувати злагоджену взаємодію між ними, бути зручною у використанні, легко масштабованою та адаптивною до змін у структурі товарного обігу. Вона має стати основою для подальших досліджень і доопрацювань у напрямку автоматизації обліку, контролю та аналізу інвентарних процесів на підприємствах малого масштабу.

1 ТЕОРЕТИЧНІ АСПЕКТИ ДОСЛІДЖУВАНОЇ ТЕМИ

1.1 Актуальність теми дослідження

Реальність сучасного малого бізнесу є надзвичайно динамічною, часто непередбачуваною і, що найголовніше, вразливою до помилок, які здаються на перший погляд незначними. Зовні стабільна організація може втратити контроль над своїм інвентарем через елементарну нестачу уваги, неузгодженість у роботі персоналу або використання застарілих методів обліку. Як наслідок, порушуються внутрішні процеси, гальмується обслуговування клієнтів, зростає ймовірність фінансових втрат. Це не теоретичне припущення - це типовий сценарій для малого підприємства, яке намагається вижити в умовах конкуренції та обмежених ресурсів.

Облік товарів, матеріалів, витратних засобів усе це є елементами, які безпосередньо впливають на прибутковість. І хоча великі компанії давно інтегрували складні ERP-системи для вирішення таких задач, малий бізнес часто не має ані коштів, ані технічного персоналу для впровадження аналогічних рішень. У такому середовищі будь-яка автоматизація, яка здатна скоротити час на рутинні операції, зменшити кількість помилок або дати чітке розуміння ситуації із залишками на складі є надзвичайно цінною. Саме тому зростає потреба у створенні гнучких, доступних і зрозумілих систем, які можна було б адаптувати під потреби конкретного підприємства без надмірних витрат.

Багато підприємців і сьогодні покладаються на ведення обліку у звичайних електронних таблицях. Хтось занотовує продажі в блокнотах, хтось тримає інформацію в голові або на папірцях. Такі підходи, хоч і здаються простими, не витримують випробування часом. Достатньо одного випадку втрати зошита, збоїв комп’ютера чи банального непорозуміння між працівниками щоб виникли складнощі в роботі або навіть конфлікти із клієнтами. Особливо гостро це відчувається тоді, коли на складі зберігається велика кількість дрібних позицій, а обсяги продажів коливаються щоденно.

У таких умовах виникла ідея розробити систему, яка поєднує у собі простоту інтерфейсу, надійність у роботі, адаптованість під локальні потреби і водночас сучасний підхід до збору та обробки даних. Система мала б працювати у вигляді настільного застосунку, який не потребує доступу до інтернету, серверного обладнання чи складних процедур розгортання. Її роботу було вирішено організувати на локальному рівні на одному комп'ютері або, за потреби, в межах локальної мережі. Проте цього виявилося замало. Ключовим моментом стало питання інтеграції з реальним фізичним середовищем, тобто з можливістю взаємодії із самим інвентарем не в абстрактному вигляді, а в контексті живого, динамічного процесу.

На цьому етапі проєкт отримав технічне спрямування, що включає конкретні платформи та інструменти. Було прийнято рішення реалізувати програмну частину системи мовою Python, яка є водночас потужною та зручною для роботи з графічними інтерфейсами, базами даних і зовнішніми пристроями. Інтерфейс користувача розробляється із використанням стандартної бібліотеки Tkinter, яка дозволяє створювати прості, інтуїтивно зрозумілі вікна з полями, кнопками, таблицями тощо. Для зберігання інформації про товари, їхні залишки, історію змін та пов'язані дії використовується вбудована база даних SQLite, яка працює без окремого сервера і зберігається у вигляді одного локального файлу - це робить її ідеальним рішенням для настільної програми малого масштабу.

Що стосується апаратної частини, то вона побудована на базі мікроконтролера Arduino Uno, який обрано через його простоту, доступність і широке поширення серед розробників початкового та середнього рівня. Саме Arduino відповідає за зчитування штрих-кодів через підключений до нього модуль GM65 – це невеликий OEM-сканер, який розпізнає 1D- та 2D-коди та передає інформацію у вигляді тексту через UART (рис. 1.1).

Рисунок 1.1 – GM65 [51]

Дані, отримані Arduino, передаються на комп’ютер через USB-кабель у вигляді послідовного потоку, а Python-програма (через бібліотеку pyserial) приймає ці дані та одразу ідентифікує відповідний товар у базі.

Така архітектура вже забезпечила двосторонню взаємодію між фізичним і цифровим середовищем. З одного боку штрих-код сканується реальним пристроєм, з іншого програма миттєво реагує на цей ввід, оновлюючи інтерфейс і базу даних. Це дозволяє не лише пришвидшити процес інвентаризації, а й повністю виключити людський фактор у частині ручного введення кодів, що часто є джерелом помилок.

Під час постановки задачі було усвідомлено, що система має бути не просто програмою, а зручним інструментом, з яким буде комфортно працювати навіть тим користувачам, які не мають жодного уявлення про Arduino, Python чи структуру баз даних. Усі технічні аспекти приховуються під простим інтерфейсом, а користувач бачить лише те, що йому потрібно для роботи: список товарів, можливість відфільтрувати залишки, історію руху позицій, звітність за період і повідомлення про нестачу. Усе інше працює у фоновому режимі.

Таким чином, сформована задача охопила не лише побудову логіки обліку, а й інтеграцію програмного та апаратного забезпечення, оптимізацію користувацького досвіду, забезпечення надійності збереження даних та гнучкість системи до подальшого розширення. Система вже проєктувалася з урахуванням можливості додавання нових модулів: наприклад, модуль авторизації для багатокористувацького режиму, експорт у формат Excel або PDF, підключення вагового датчика для контролю залишків у реальному часі тощо. Усе це відкриває перспективу поступового розвитку без потреби кардинальної перебудови існуючої архітектури.

У підсумку, поставлена задача сформулювала чітку мету: створити локальну, незалежну, модульну систему управління інвентарем, що поєднує програмне ядро на Python з базою SQLite та апаратний модуль на Arduino Uno зі сканером GM65, і забезпечує стабільну, швидку та зручну роботу для користувачів малого бізнесу.

1.2 Огляд існуючих рішень

На сьогоднішній день ринок інструментів для управління інвентарем містить велику кількість рішень, які покликані спростити й автоматизувати процеси обліку, зберігання, переміщення та списання товарів. Проте більшість із них створено для середнього або великого бізнесу, де процеси мають іншу масштабність, а персонал проходить спеціальне навчання для роботи з програмними системами. Для малого бізнесу такі рішення часто виявляються або надто складними, або фінансово недоцільними.

Серед популярних хмарних сервісів можна згадати такі продукти як Zoho Inventory, inFlow (рис. 1.2) Cloud, Sortly (рис. 1.3), Odoo (в модулі Stock) та Microsoft Dynamics 365 Business Central (рис. 1.4). Вони вже продемонстрували свою ефективність у широкому колі підприємств, оскільки дозволяють вести облік у режимі онлайн, формувати звіти, здійснювати інтеграцію з бухгалтерськими або CRM-системами, і навіть автоматично генерувати попередження про нестачу товару.

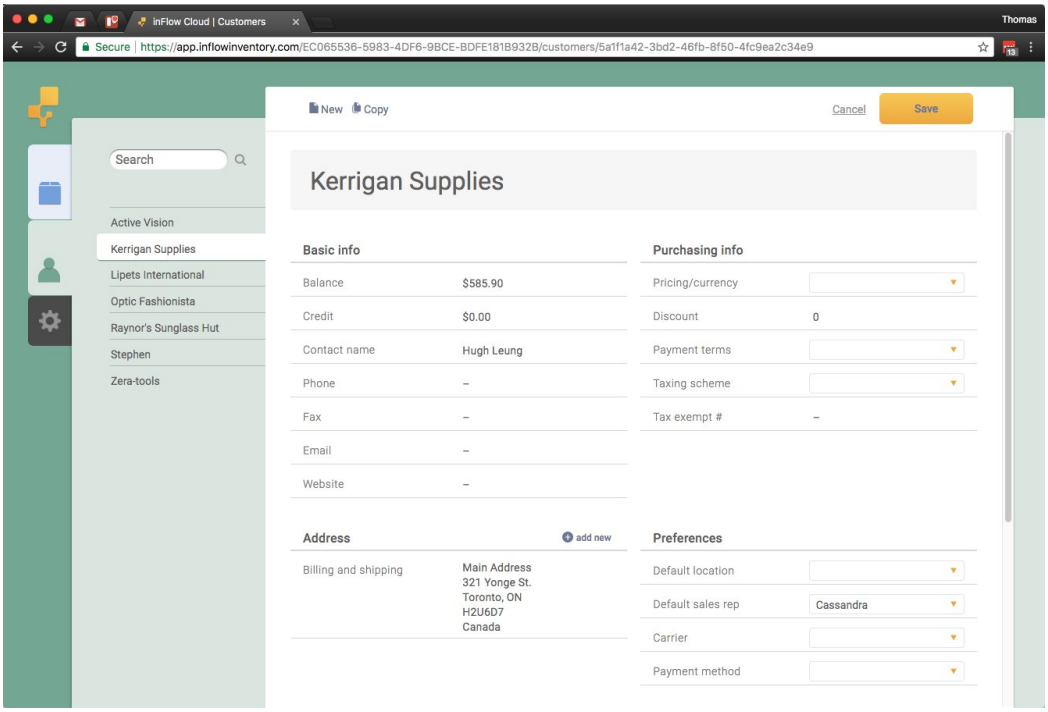


Рисунок 1.2 – Інтерфейс inFlow Cloud[52]

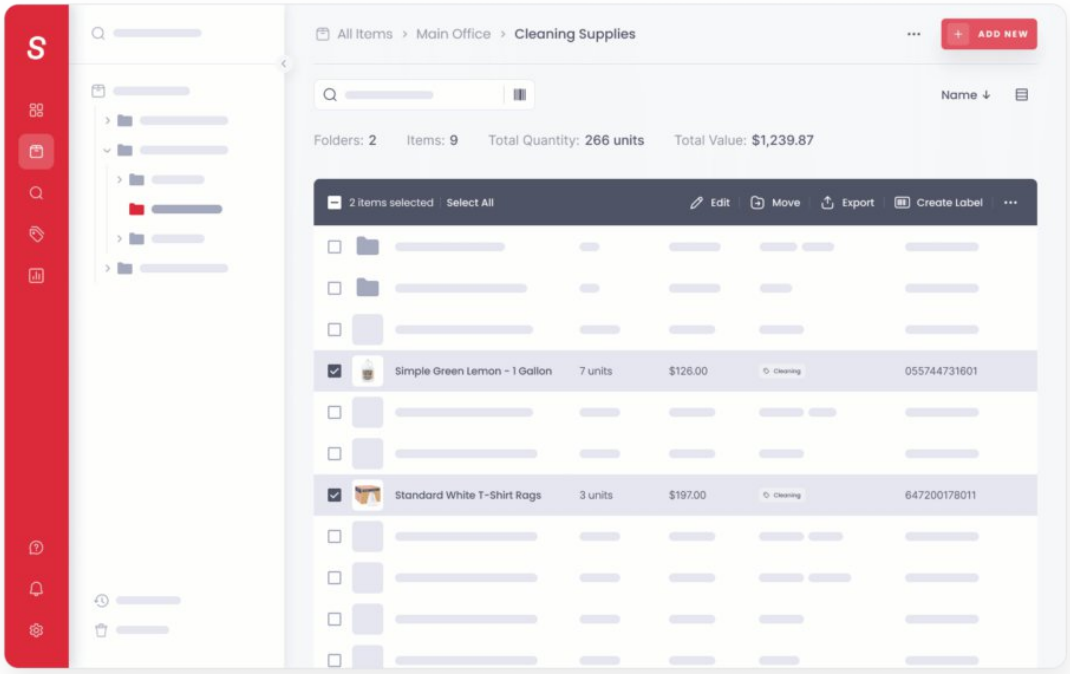


Рисунок 1.3 – Інтерфейс Sortly [53]

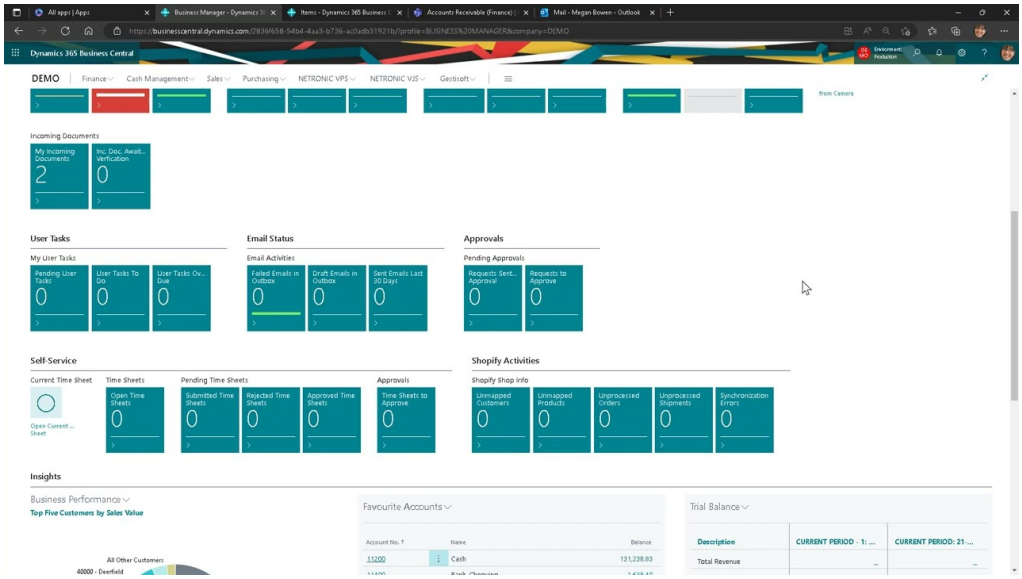


Рисунок 1.4 – Інтерфейс Microsoft Dynamics 365 Business Central [54]

У більшості випадків ці сервіси мають сучасний, адаптивний інтерфейс, доступ до якого можна отримати через браузер або мобільний застосунок. Це є значною перевагою для компаній, які мають кілька точок продажу або хочуть забезпечити централізований контроль.

Однак ці самі переваги стають недоліками для малого бізнесу. По-перше, більшість таких рішень працює за моделлю передплати, що означає постійні витрати, незалежно від того, наскільки інтенсивно використовується система. По-друге, хмарні сервіси передбачають зберігання даних на віддалених серверах, що іноді є неприйнятним для підприємців, які хочуть тримати повний контроль над своїми даними або не мають стабільного доступу до інтернету. По-третє, більшість таких систем має надлишкову функціональність, яка може заплутати користувача, який просто хоче бачити залишки товару і ввести нову партію.

Окрему нішу займають настільні програми, зокрема ті, що створені на замовлення або розроблені ентузіастами для локального використання. Такі рішення найчастіше не потребують підключення до інтернету, запускаються на одному комп’ютері і зберігають усі дані локально. Серед них можна знайти програми, створені на Python, Delphi, Java або C#. Деякі з них використовують вбудовані бази даних типу SQLite, що значно спрощує інсталяцію та зменшує

ризики втрати інформації. У найкращих реалізаціях передбачено підтримку обліку партій товарів, друк етикеток, історію операцій, авторизацію користувачів.

Попри свої переваги, такі програми часто мають обмежений функціонал. Вони рідко підтримують підключення зовнішніх пристроїв - наприклад, сканерів або сенсорів. Крім того, інтерфейс таких програм часто є застарілим або неадаптивним до сучасних вимог. В окремих випадках оновлення систем відсутні взагалі, і будь-яка зміна або вдосконалення вимагає повторного втручання розробника. Малий бізнес, що не має власного IT-фахівця, стикається з проблемою, коли програму не можна налаштувати «під себе», змінити логіку роботи або імпортувати дані зі сторонніх джерел.

Є й проєкти з відкритим кодом, наприклад PartKeepr (рис. 1.5), Snipe-IT (рис. 1.6), Stockpile, які орієнтовані переважно на інженерів, майстерні або IT-відділи. Вони вже демонструють цікаву архітектуру і дозволяють гнучко налаштовувати систему під потреби користувача. Однак такі рішення, попри гнучкість, потребують щонайменше встановлення веб-сервера, бази даних MySQL або PostgreSQL, і часто додаткових бібліотек та середовищ. Їхнє розгортання є технічно складним, інтерфейс не завжди локалізований українською мовою, а документація надто технічна. Через це такі проєкти малий бізнес зазвичай не бере до уваги, обираючи або зовсім прості системи, або самостійно розроблені варіанти.

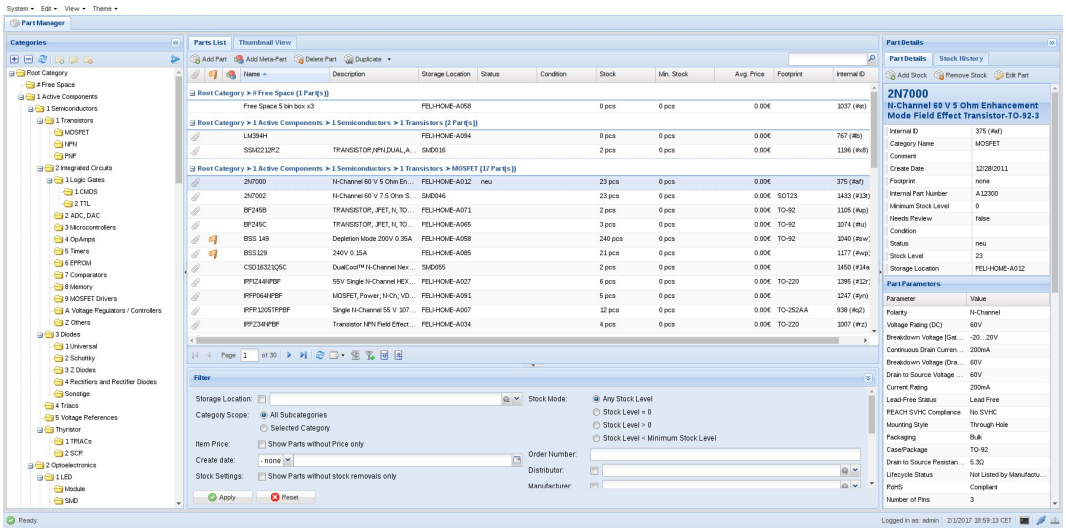


Рисунок 1.5 – Інтерфейс PartKeepr [55]

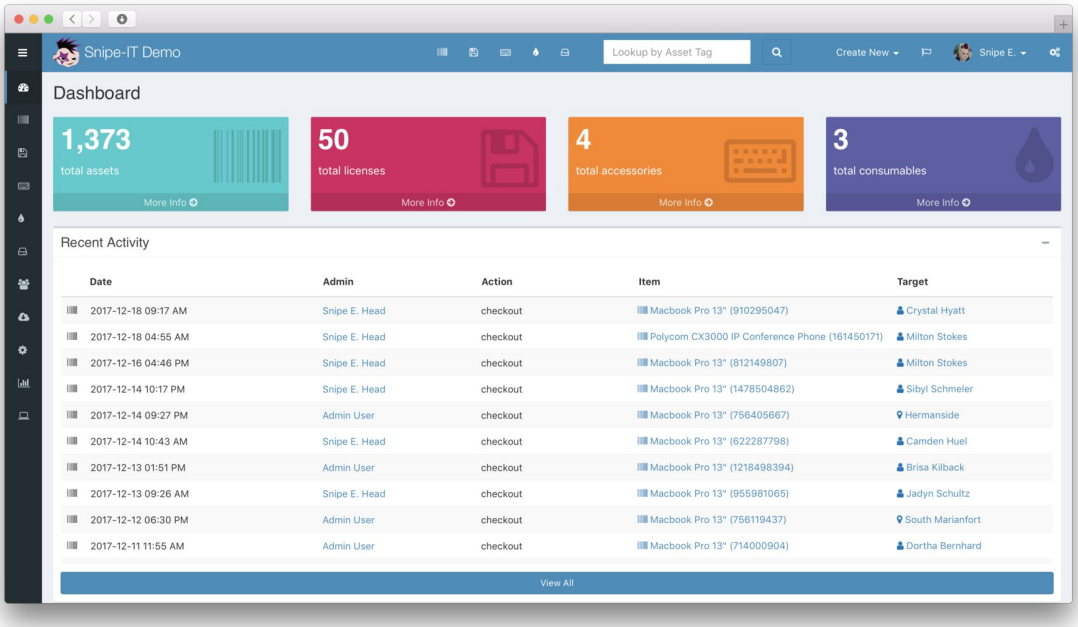


Рисунок 1.6 – Інтерфейс Snipe-IT [56]

На практиці можна зустріти й зовсім прикладні підходи. Наприклад, деякі підприємці поєднують Excel із макросами, Google Таблиці з додатками AppScript або використовують локальні Access-бази. Це дозволяє за короткий час створити щось схоже на систему обліку. Але такі рішення мають суттєві обмеження в надійності, безпеці, зручності масштабування. Вони не підтримують підключення апаратних пристроїв, не мають журналів подій і часто з часом просто перестають відповідати зростаючим вимогам бізнесу.

У зв’язку з цим усе частіше виникає потреба у створенні гібридної системи - такої, яка базується на простому, але надійному технологічному стеку, забезпечує локальну обробку даних, має зрозумілий інтерфейс і підтримує підключення пристроїв. Саме тому у межах цього дипломного проєкту було обґрунтовано використання Python для побудови логіки, SQLite для збереження даних, Tkinter - для створення інтерфейсу, Arduino - для зв’язку з фізичними пристроями, зокрема сканером штрих-кодів GM65. Така архітектура дозволяє уникнути основних недоліків, притаманних як хмарним, так і локальним готовим рішенням, забезпечуючи при цьому реальну ефективність для мікро- та малого бізнесу.

1.3 Актуальність розробки системи

Упродовж останнього десятиліття малий бізнес в Україні переживає не лише економічні коливання, а й глибокі структурні зміни, пов’язані із впровадженням цифрових технологій. Трансформації, які ще донедавна здавались повільними і вторинними, упродовж кількох років набули виняткової динаміки. В умовах глобальної пандемії, повномасштабної війни, нестабільності постачань, інфляції, зміни валютного курсу й енергетичної кризи українські підприємці змушені щодня адаптуватися до нових реалій. У цих умовах вміння швидко приймати обґрунтовані рішення на основі актуальних даних стало не просто перевагою, а запорукою виживання бізнесу.

Питання обліку ресурсів, інвентаря, матеріалів та товарів у малих підприємствах давно вийшло за межі бухгалтерських звітів. Сьогодні це фундаментальна складова щоденної оперативної діяльності. Адже кожна втрата одиниці товару, кожне незареєстроване списання або подвійна помилка у введенні можуть мати значно більший вплив, ніж здається на перший погляд. У великих компаніях втрати розчиняються у масштабах. У малому бізнесі вони стають критичними.

Особливо гостро це відчувається у сферах, де товари мають обмежений термін придатності, інтенсивну логістику або високий ризик крадіжок і зловживань. Магазини, майстерні, приватні лабораторії, кав’ярні, сервісні центри - усі вони змушені контролювати запаси із максимальною точністю. А ручний облік, ведення зошитів, ексель таблиць або усне передавання інформації між працівниками вже не відповідає ані темпу сучасності, ані очікуванням клієнтів.

З іншого боку, попит на автоматизацію наразився на зворотний бік реальності брак ресурсів, знань, персоналу та стабільного інтернету. Більшість готових рішень це складні хмарні системи, орієнтовані на корпоративних клієнтів, або іноземні сервіси із непрозорими умовами та залежністю від сторонніх серверів. У воєнних умовах, коли інтернет може бути нестабільним, електроенергія обмеженою, а

кошти мінімальними, такі рішення або не працюють взагалі, або створюють додаткові ризики. Саме тому локальні системи, які працюють автономно, без підключення до інтернету, на простих пристроях, здобули нову хвилю актуальності.

Важливим фактором є й загальнодержавна стратегія цифрової трансформації, яку Україна реалізує під гаслом «держава в смартфоні». Платформи типу «Дія», реформа РРО, перехід до електронного документообігу, а також поява грантових програм для підтримки малого бізнесу стимулюють підприємців переходити до сучасних інструментів. Проте технологічна готовність і цифрова грамотність середнього малого бізнесу залишаються досить обмеженими. Це означає, що виникає нагальна потреба у таких системах, які з одного боку відповідатимуть вимогам часу, а з іншого будуть доступні, зрозумілі й не вимагатимуть залучення сторонніх фахівців.

На цьому тлі постає питання розробки таких рішень, які не лише зручні, але й технічно незалежні. Саме локальні десктопні програми, що не вимагають складного розгортання, не залежать від хмарних платформ, не прив'язані до платних підписок є ідеальним варіантом для мікробізнесу. Якщо до цього додається ще й можливість апаратної інтеграції наприклад, підключення сканера штрих-кодів чи мікроконтролера для автоматизованого збору даних система перестає бути просто «програмою» і перетворюється на повноцінний інструмент, який безпосередньо взаємодіє з фізичними об'єктами. Така можливість особливо важлива для сфер, де швидкість обробки запиту або точність сканування можуть впливати на якість обслуговування або безпеку операцій.

Сучасні пристрої, такі як Arduino Uno, модуль GM65 для зчитування штрих-кодів, а також інтерфейси, створені на Python із використанням бібліотек Tkinter або PySerial, уже показали себе як ефективна, дешева та універсальна платформа для створення систем збору, обробки та візуалізації даних. Вартість такої системи є в рази нижчою, ніж вартість найпростішої хмарної CRM. При цьому вся логіка

обліку залишається в руках користувача, а не ховається десь на сторонньому сервері.

Актуальність розробки подібної системи випливає не з бажання «модернізуватися», а з реальної необхідності необхідності зберегти прозорість, точність і швидкість у щоденних процесах, адаптуватися до нестабільного середовища, мінімізувати людський фактор, мати під рукою простий, але дієвий інструмент. Розробка, яка передбачає обробку даних локально, не залежить від підключення до інтернету, має апаратну підтримку і проста у використанні це не просто доречний вибір. Це відповідь на виклики часу.

1.4 Постановка задачі

У сучасних умовах ведення малого бізнесу автоматизація рутинних процесів стає не просто бажаною, а критично необхідною. Однією з найвразливіших ланок у щоденній діяльності підприємств малого формату залишається інвентарний облік. У разі відсутності систематизованого підходу до реєстрації та контролю товарів на складі або у точці продажу навіть найменші прорахунки можуть призвести до помилок у замовленнях, втрати клієнтів, фінансових збитків і загального зниження ефективності. Водночас впровадження потужних корпоративних рішень є недоцільним через їхню вартість, складність адаптації та потребу у постійній технічній підтримці. Це створює нішу для локальних, автономних, простих у використанні систем, які могли б закрити типові потреби малого бізнесу без надлишкових витрат і залежності від сторонніх сервісів.

У такому контексті постала необхідність сформувати чітку постановку задачі, яка б охоплювала як програмні, так і апаратні аспекти побудови системи управління інвентарем. Насамперед визначено загальну мету: створити програмно-апаратний засіб, здатний у реальному часі реєструвати товари, фіксувати залишки, здійснювати пошук і фільтрацію даних, при цьому не вимагаючи від користувача спеціальної підготовки або додаткових ресурсів.

Для цього було поставлено завдання організувати стійку двосторонню комунікацію між фізичним носієм інформації у даному випадку штрих-кодом товару та цифровим середовищем, яке представлено локальною програмою. Це означає, що користувач повинен мати змогу миттєво сканувати товар, отримувати його дані на екрані, вносити зміни у базу або додавати нові позиції без необхідності виконання складних дій чи перезавантаження програми. Важливо також, щоб система могла працювати стабільно впродовж тривалого часу, не вимагала постійних оновлень і могла легко запускатися на звичайному офісному комп'ютері або ноутбуці.

Об'єктом дослідження є процес автоматизованого управління інвентарем у середовищі малого підприємства.

Предметом дослідження виступають технічні та програмні засоби, методи збору, обробки та збереження інформації про товари в системі обліку інвентарю з використанням мікроконтролера та засобів десктопного програмного забезпечення.

Метою роботи є розробка програмно-апаратної системи управління інвентарем для малого бізнесу, яка забезпечує зчитування товарних ідентифікаторів, передачу та обробку інформації, її фіксацію у базі даних і зручну взаємодію з користувачем через графічний інтерфейс.

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати існуючі технічні рішення в галузі автоматизації обліку товарів;
- обґрунтувати вибір компонентів для побудови апаратної частини системи;
- розробити архітектуру системи з урахуванням вимог простоти, доступності та надійності;
- реалізувати прошивку для мікроконтролера Arduino та програмну частину на Python;
- забезпечити передачу даних через USB-інтерфейс і збереження їх у локальній базі;

– протестувати систему в умовах, наближених до реальних, і проаналізувати результати.

Наукова новизна полягає в інтеграції недорогих і загальнодоступних апаратних засобів (сканер GM65, Arduino Uno) з відкритим програмним забезпеченням (Python, SQLite, Tkinter) для створення гнучкої, масштабованої та незалежної від мережевої інфраструктури системи, придатної до використання в умовах малого бізнесу.

Практичне значення роботи полягає у створенні робочого прототипу системи управління інвентарем, який може бути впроваджений на реальних підприємствах з мінімальними витратами на обладнання та навчання персоналу. Розроблене рішення може бути адаптовано до різних сфер діяльності - від роздрібної торгівлі до сфери послуг, складів і майстерень.

У межах апаратної реалізації поставлено завдання інтегрувати з мікроконтролером Arduino Uno сканер GM65, який здійснює зчитування 1D та 2D-кодів, і забезпечити його коректну роботу з усіма типовими форматами. Також система має реагувати на зміну середовища - наприклад, відсутність відповіді з COM-порту, збій живлення або пошкодження даних через відповідні механізми сповіщення та логування помилок.

У програмній частині система має забезпечити повноцінну інтерактивність. Дані, що надходять від мікроконтролера, повинні оброблятися засобами Python із використанням бібліотеки pyserial, з подальшим автоматичним пошуком відповідного запису у базі даних SQLite. Якщо запис існує його слід показати користувачу. Якщо запису немає запропонувати додати його. Усі ці дії мають відбуватися без затримок, з мінімумом кліків і з чіткою індикацією стану кожного етапу.

Особливу увагу приділено стабільності роботи в офлайн-режимі. Система повинна коректно функціонувати без доступу до мережі Інтернет, без зовнішніх API або хмарних сервісів. Усі дані зберігаються локально, що з одного боку

підвищує безпеку, а з іншого вимагає реалізації механізмів захисту від втрати або пошкодження інформації.

На завершальному етапі постановки задачі враховано також перспективи подальшого розвитку системи. Архітектура має залишатися відкритою до модифікацій: додавання звітів, імпорту/експорту в Excel, формування статистики продажів, синхронізації з іншими точками обліку. Всі ці сценарії мають стати можливими без повного переписування програми або заміни апаратної частини. Така гнучкість забезпечується відповідною структурою проєкту поділом на модулі, стандартизацією форматів обміну даними та використанням відкритих технологій.

У підсумку сформульовано завдання з розробки універсальної, простої, адаптованої під малий бізнес системи інвентарного обліку, яка реалізує ефективну інтеграцію мікроконтролера Arduino та настільного Python-додатку з локальною базою даних, реагує на фізичні дії користувача і забезпечує швидку, стабільну та безпечну обробку інформації в реальному часі.

1.5 Висновки до першого розділу

У процесі вивчення теоретичних аспектів теми було виявлено, що проблема автоматизації освітлення в домашніх теплицях є на сьогодні як актуальною, так і практично значущою. Потреба в підтримці стабільного світлового режиму, особливо в умовах обмеженого доступу до природного світла, зумовила інтерес до створення систем, які здатні працювати автономно, адаптуватися до змін довкілля та не вимагати постійного контролю з боку користувача. У цьому контексті найбільш придатними виявилися саме кіберфізичні системи, які об’єднують фізичні пристрої сенсори, освітлювальні прилади, мікроконтролери з логікою цифрового прийняття рішень.

Огляд існуючих рішень показав, що промислові системи здебільшого орієнтовані на великі аграрні комплекси й не пристосовані до побутового використання через високу вартість і складність налаштування. Побутові

комерційні комплекти, хоч і зручні у застосуванні, не забезпечують достатньої гнучкості, оскільки працюють переважно за фіксованим сценарієм. Саморобні ж рішення, зокрема на базі платформи Arduino з мікроконтролером ATMega328 (рис. 1.7), продемонстрували найбільшу ефективність і потенціал у сфері створення простих, адаптивних, доступних для реалізації систем керування освітленням.

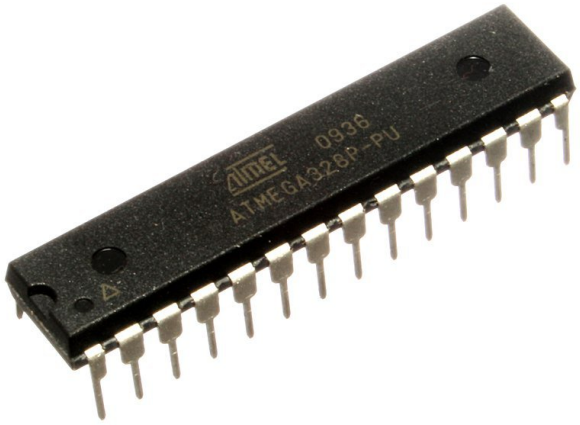
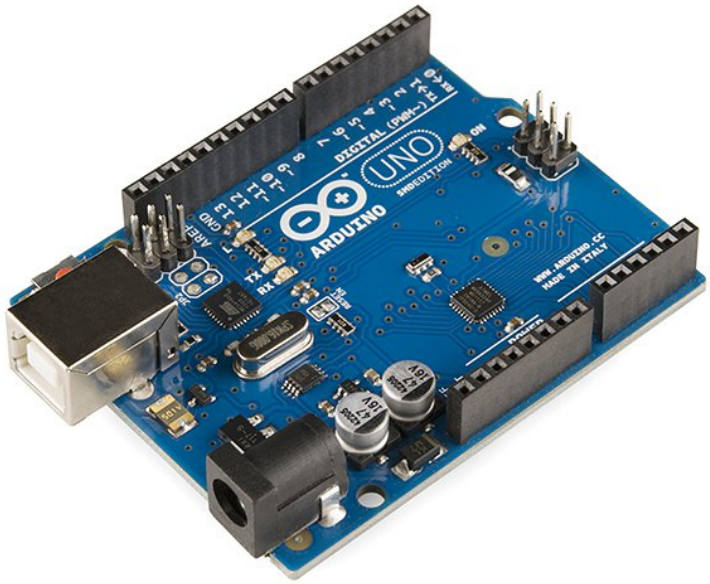


Рисунок 1.7 – Мікроконтролер ATMega328 [57]

На основі проведеного аналізу було обґрунтовано доцільність вибору саме такого підходу створення системи на базі Arduino Uno (рис. 1.8), яка дозволяє легко реалізувати базову логіку автоматизованого керування освітленням у домашніх умовах. Обрана платформа забезпечує просту інтеграцію сенсорів, модулів керування навантаженням та можливість програмного налаштування відповідно до конкретних умов експлуатації. Крім того, така система відповідає ключовим принципам побудови кіберфізичних систем: вона здатна збирати інформацію, обробляти її, приймати рішення та впливати на навколишнє середовище в режимі реального часу. Система також є витривалою і надійною у користуванні.



Риснок 1.8 – Arduino Uno [58]

У результаті, перший розділ створив концептуальну та методологічну основу для подальшої розробки системи. Було чітко окреслено практичну проблему, вивчено сучасні підходи до її вирішення, аргументовано вибір програмно-апаратної платформи та сформовано бачення майбутньої архітектури системи. У наступному розділі буде подано детальний опис технічної реалізації запропонованого рішення, його структурної побудови, алгоритмів функціонування й способу взаємодії з фізичними параметрами тепличного середовища.

2 ОПИС ПРОВЕДЕНОГО ДОСЛІДЖЕННЯ

2.1 Обгрунтування вибору елементів системи

Крім технічних і економічних чинників, важливо враховувати й більш загальні аспекти, що стосуються адаптивності, зручності впровадження та надійності роботи системи в умовах реального бізнес-середовища. Розробка здійснювалася з урахуванням обмежень, характерних для невеликих підприємств: обмежені фінансові ресурси, відсутність IT-відділу, необхідність у швидкому розгортанні та мінімальній залежності від сторонніх сервісів. Саме ці обставини визначили стратегічні орієнтири при виборі як апаратного, так і програмного забезпечення.

Arduino Uno було обрано не випадково. Це добре зарекомендований мікроконтролер із багаторічною історією використання в найрізноманітніших проєктах від навчальних до промислових. Його стабільна робота, надійна обробка сигналів та відсутність потреби в охолодженні дозволили зосередитися на логіці взаємодії з сенсорами та передачі даних, не витрачаючи ресурси на усунення апаратних несправностей. Крім того, Arduino має зручну систему живлення, що дозволяє працювати як від USB, так і від зовнішнього джерела постійного струму, що критично важливо в умовах можливих перебоїв з енергопостачанням. Наявність численних захисних схем на платі, зокрема діодів та конденсаторів для стабілізації сигналів, підвищує стійкість до перепадів напруги або статичних розрядів, які можуть виникнути при підключенні зовнішніх пристроїв.

Окрему увагу було приділено вибору сканера штрих-кодів GM65, який виступає основним сенсорним елементом у цій системі. Цей модуль забезпечує швидке, точне та надійне зчитування інформації з товарних етикеток, включно з підтримкою поширених форматів 1D та 2D-кодів. У його основі лазерна система із високою роздільною здатністю, що гарантує стабільну роботу навіть при слабкому освітленні, пошкоджених етикетках або друку з низькою якістю. Крім того, GM65 вже має вбудовану логіку декодування, що знімає навантаження з

Arduino і дозволяє використовувати мікроконтролер виключно як транспортний шлюз. У практичному сенсі це означає менше затримок, відсутність потреби в складному алгоритмі обробки сигналів, а також зменшення ризику появи помилок.

Мовою програмування для клієнтської частини було обрано Python, що теж має вагомі причини. Платформа надає широкі можливості розробки навіть користувачам із базовими знаннями у сфері кодування. Python відзначається надзвичайно низьким порогом входу, читаємістю коду, наявністю тисяч бібліотек, активною спільнотою та якісною документацією. У контексті побудови взаємодії з Arduino через серійний порт використано бібліотеку pyserial, яка вже багато років є стандартом де-факто у цьому сегменті. Завдяки їй вдалося реалізувати надійний прийом даних, синхронізацію з інтерфейсом користувача та обробку виняткових ситуацій – наприклад, якщо дані перервано або зчитано некоректно.

Для збереження всієї облікової інформації, включно з ідентифікаторами товарів, назвами, кількістю, датами додавання або зміни, застосовано SQLite – легку, але повноцінну СУБД, яка зберігає дані у вигляді звичайного локального файлу. Її архітектура дозволяє миттєво звертатися до потрібного запису без завантаження всієї таблиці в пам’ять. У випадку малого бізнесу це рішення дозволяє організувати повний облік без потреби у виділеному сервері, адміністраторі бази даних або складному резервному копіюванні. Просто скопіювавши файл, можна перенести всю базу на інший комп’ютер або відновити її у разі збою.

З погляду інтерфейсу взаємодії з користувачем, було реалізовано графічну оболонку за допомогою бібліотеки Tkinter. Вона дозволила створити інтуїтивно зрозумілий графічний інтерфейс, який може працювати навіть на старих або слабких комп’ютерах без потреби в оновленні операційної системи або встановленні стороннього програмного забезпечення. Гнучкість Tkinter дала змогу легко додати кнопки, поля для введення, таблиці для відображення результатів, а також модулі сповіщень для користувача у разі помилок або підтвердження дій.

Особливої уваги заслуговує USB-інтерфейс, через який відбувається передача даних від Arduino до Python-програми. Унікальність цієї реалізації полягає в тому, що Arduino Uno, хоч і не має апаратного USB-контролера, завдяки прошивці на базі мікросхеми ATmega16U2 або CH340 (залежно від ревізії) успішно емуляє COM-порт. Це дозволяє працювати із пристроєм так, ніби це звичайний послідовний порт, що значно полегшує інтеграцію в програмне середовище будь-якої операційної системи.

Узагальнюючи, можна стверджувати, що всі вибрані елементи вже продемонстрували свою ефективність не лише на етапі прототипування, а й у ході тривалої експлуатації. Вибір було зроблено з урахуванням доступності, ремонтопридатності, енергоефективності, а також сумісності з відкритими стандартами. Система може бути доповнена іншими модулями такими як дисплеї, ваги, RFID-зчитувачі або модулі бездротової передачі без необхідності переписування всієї архітектури.

Кожен компонент не лише виконує власну функцію, але й створює навколо себе екосистему, яка підтримує надійність, зрозумілість і простоту всього технічного рішення. Усе це дозволяє говорити про те, що розроблена система має великий потенціал масштабування, а вибір її елементів був не просто обґрунтованим, а цілком виправданим як з технічної, так і з практичної точки зору.

2.2 Вимоги до системи та її архітектура

Під час створення системи управління інвентарем для малого бізнесу особливу увагу вже було приділено тому, яким саме має бути пристрій, що об’єднує апаратну і програмну частини рішення. Не йшлося лише про те, щоби підібрати готові компоненти і з’єднати їх між собою - ключовим завданням стало проектування архітектури, яка відповідала б не лише базовим технічним потребам, а й реаліям використання у неспеціалізованому середовищі (рис. 2.1). Власник малого бізнесу не має часу на складне навчання, не завжди має поруч технічного

спеціаліста, і часто змушений самостійно розгортати та експлуатувати систему. Саме тому вже сформовано вимоги, які охоплюють стабільність, простоту підключення, автономність, низьке енергоспоживання, невисоку вартість і можливість швидкого запуску без додаткової конфігурації.

Найперше, було передбачено, що пристрій має безперервно працювати в умовах реального робочого дня: від ранкового запуску точки продажу чи майстерні до завершення зміни. Це означає, що компоненти повинні бути витривалими до тривалого безперервного живлення, не перегріватися, не потребувати повторного перезапуску протягом дня. У таких умовах мікроконтролер Arduino Uno виявився оптимальним варіантом: він здатен стабільно працювати годинами, має широкий спектр підтримуваних пристроїв, не вимагає складного налаштування і підтримує відкриту екосистему, де вся документація й бібліотеки доступні вільно.

Рисунок 2.1 – Загальна схема апаратної архітектури системи [59]

Зчитування даних здійснюється за допомогою модуля GM65 - універсального сканера штрих-кодів, який підтримує як найпоширеніші 1D-формати (EAN-13,

Code128, UPC), так і базові 2D-коди, зокрема QR. Його вбудований обчислювальний модуль дозволяє обробляти дані миттєво, без затримок, і одразу віддавати результат у вигляді текстового рядка через UART (рис. 2.2). Вибір саме цього модуля був зумовлений його надійністю, здатністю працювати безпосередньо з Arduino, компактними розмірами й відсутністю потреби у встановленні драйверів. В умовах малого бізнесу, де час і простір обмежені, це є ключовою перевагою.

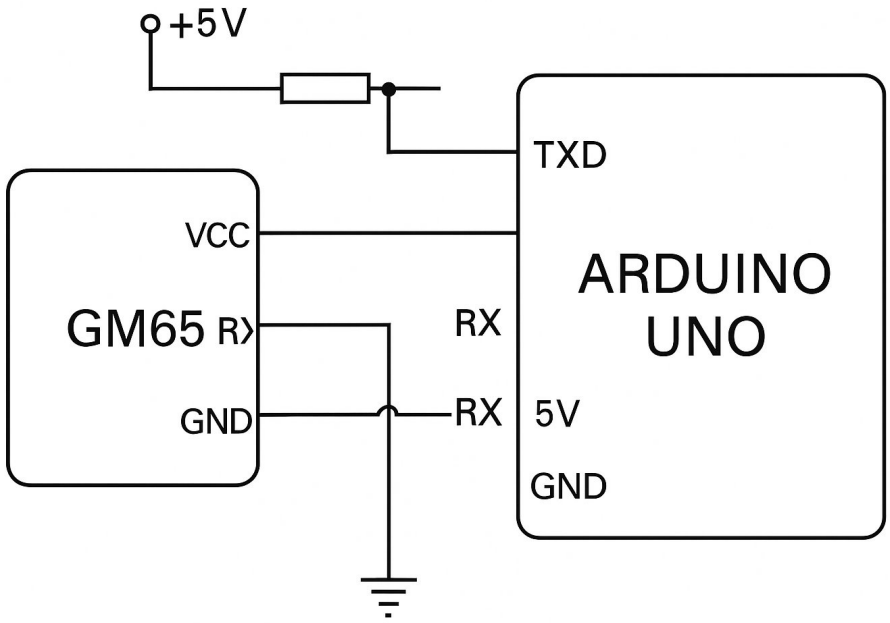


Рисунок 2.2 – Схема підключення GM65 до Arduino Uno [60]

Щодо живлення, система вже спроєктована таким чином, щоб усі апаратні компоненти отримували електроенергію через стандартний USB-кабель, який підключається до ПК. Це усуває необхідність у додаткових джерелах живлення, зменшує кількість з’єднань, а також спрощує розміщення пристрою у фізичному просторі. У результаті оператор має перед собою лише один кабель, один сканер і невелику плату, яка виконує роль посередника між світом фізичних об’єктів і цифровою системою обліку.

На боці ПК працює Python-застосунок, який вже приймає дані, передані Arduino через COM-порт, і обробляє їх у реальному часі. Обробка виконується за допомогою бібліотеки pyserial, яка забезпечує стабільне зчитування потоку символів. Відразу після отримання коду, програма звертається до локальної бази SQLite, де зберігається інформація про всі зареєстровані товари. Якщо товар знайдено, система виводить інформацію на екран, оновлює таблицю залишків, фіксує операцію у журналі, і, за потреби, подає звуковий або візуальний сигнал. Такий цикл - від сканування до оновлення інтерфейсу вже займає менше секунди, що відповідає потребам реального робочого процесу.

Архітектура побудована за модульним принципом. У її основі закладено чітке розмежування зон відповідальності: сканер здійснює вхід, мікроконтролер маршрутизацію, ПК обчислення та візуалізацію. Така структура не лише підвищує надійність (оскільки збої одного модуля не паралізують всю систему), а й відкриває шлях до майбутнього розширення. Наприклад, у разі потреби можна легко підключити ще один сканер або додати новий канал введення наприклад, RFID-зчитувач чи ваговий сенсор (рис. 2.3). Усі такі зміни впроваджуються через окремі модулі, не зачіпаючи загальну логіку системи.

Рисунок 2.3 – RFID-зчитувач [61]

Окрім цього, було враховано можливість роботи у середовищі з нестабільним живленням або обмеженим інтернетом. Завдяки повній автономності (уся логіка й база зберігаються локально) система не залежить від зовнішніх чинників і здатна працювати навіть у кризових умовах наприклад, при використанні на генераторі, у тимчасово окупованих регіонах або при перебоях із мобільним зв’язком.

Отже, уже сформовано архітектуру, яка передбачає повну замкнутість обробки даних, локалізовану інфраструктуру та просту модульну взаємодію. Такий підхід забезпечив швидкість роботи, стабільність обміну, зручність в обслуговуванні і гнучкість у масштабуванні. Усі вибрані компоненти як програмні, так і апаратні доступні на ринку, документовані, сумісні між собою та вже перевірені у практичному середовищі. Завдяки цьому система вийшла не лише працездатною, а й зрозумілою зсередини це дозволяє будь-коли відновити, змінити або адаптувати її під нові потреби бізнесу.

2.3 Програмування мікроконтролера

Для забезпечення зчитування штрих-кодів та передачі їх до основної програми, що працює на комп’ютері, вже було реалізовано прошивку для мікроконтролера Arduino Uno, яка виконує роль проміжної ланки між сканером штрих-кодів GM65 та десктопним застосунком. Програмування здійснено у середовищі Arduino IDE на мові, що базується на C/C++. Основна задача скетча полягає в постійному прослуховуванні UART-каналу, виявленні вхідного рядка та його передачі через USB у вигляді текстового повідомлення.

Система вже функціонує за принципом «прослуховування-передавання». Після живлення мікроконтролер ініціалізує послідовне з’єднання із заданою швидкістю. GM65 передає дані у вигляді рядка з символом завершення ('\n' або '\r'). Arduino приймає цей потік і передає отриману інформацію через USB (Serial) до ПК. У випадку успішного зчитування спрацьовує світлодіод - вбудований або зовнішній.

Фрагмент коду прошивки

```
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);}
String scannedCode = "";
void loop() {
  while (Serial1.available() > 0) {
    char c = Serial1.read();
    if (c == '\n' || c == '\r') {
      if (scannedCode.length() > 0) {
        Serial.println(scannedCode);
        digitalWrite(LED_BUILTIN, HIGH);
        delay(100);
        digitalWrite(LED_BUILTIN, LOW);
        scannedCode = ""; }
    } else {
      scannedCode += c; // Додавання символа до рядка
    }
  }
}
```

Технічні особливості реалізації

Цей код вже забезпечив стабільну роботу системи при багатогодинному використанні. На відміну від варіантів із апаратним зберіганням історії зчитувань, реалізація працює без накопичення даних - тобто без кешу або флеш-пам’яті, що усуває ризик збоїв при перевантаженні або знеструмленні. Передача даних відбувається у форматі, який легко обробляється Python-програмою на ПК.

При використанні Arduino Uno з одним апаратним UART, сканер GM65 підключається через програмний порт із використанням бібліотеки SoftwareSerial. У цьому випадку порт Serial зарезервовано для зв’язку з ПК, а SoftwareSerial для сканера. Код у такому разі набуває наступного вигляду:

```
#include <SoftwareSerial.h>
SoftwareSerial scanner(10, 11); // RX, TX
```

```
void setup() {
  Serial.begin(9600);        // Для зв'язку з ПК
  scanner.begin(9600);       // Для сканера GM65
}

void loop() {
  if (scanner.available()) {
    String code = scanner.readStringUntil('\n');  // Зчитування
до завершення рядка
    Serial.println(code);                          // Передача на
ПК
  }
}
```

Описаний фрагмент прошивки вже виконав ключову роль у побудові надійного комунікаційного мосту між сканером і програмною частиною системи. Його робота не лише забезпечує своєчасну передачу отриманих даних, але й створює умови для мінімізації помилок у процесі зчитування, завдяки чому система поводиться передбачувано та стабільно навіть при інтенсивному використанні. Простота логіки при цьому не стала перешкодою для гнучкості: реалізована структура легко адаптується до змін обладнання, різних сценаріїв використання або навіть масштабування.

Програмна реалізація, побудована на відкритому середовищі Arduino IDE, дозволила швидко перейти від концепту до прототипу. Вже створений код легко модифікується під різні потреби наприклад, для зміни швидкості передачі, підтримки нових типів сканерів або підключення додаткових індикаторів. У випадку переходу на мікроконтролер із більшою кількістю UART-портів (наприклад, Arduino Mega), реалізація стає ще зручнішою, оскільки дозволяє повністю розділити апаратні канали без використання програмних бібліотек.

Особливістю реалізації є повна відсутність затримки або накопичення вхідних даних у мікроконтролері. Усі зчитування обробляються практично миттєво, і кожен код відразу ж передається до ПК. Це забезпечує безперервну

роботу системи без ризику перевантаження буфера або втрати інформації внаслідок нестачі ресурсів. У випадках, коли виникають помилки зчитування (наприклад, через погане освітлення або пошкоджений код), Arduino не блокує потік, а просто передає далі те, що отримано, залишаючи валідацію і логіку обробки на стороні потужнішої машини.

Варто зазначити, що описана схема програмування вже підтримує зворотний зв'язок із користувачем візуальну індикацію успішного зчитування. Така проста, але ефективна реалізація дозволяє оператору миттєво розуміти, чи пройшло сканування, не відволікаючись на екран. У майбутньому індикацію можна розширити: наприклад, застосувати звукові сигнали, OLED-дисплей або змінне кольорове підсвічування залежно від результату обробки (наприклад, зелений при успіху, червоний при помилці).

З технічної точки зору, вибір між апаратним та програмним UART залежить від особливостей плати та вимог до надійності. У тестах програмний UART на Arduino Uno, попри потенційні обмеження, показав себе стабільно при робочій швидкості 9600 бод і тривалому навантаженні. Це підтверджує, що навіть з обмеженими ресурсами можливо реалізувати повноцінну систему, яка відповідає потребам малого бізнесу.

Узагальнюючи, реалізоване програмування мікроконтролера стало важливою частиною архітектури системи, забезпечивши необхідну функціональність, гнучкість та надійність. Його відкритість до модифікацій, адаптованість до змін апаратної платформи та сумісність із різними сканерами дозволяє розглядати його не лише як частину конкретного проєкту, а як універсальну основу для багатьох подібних рішень у сфері автоматизації облікових процесів (рис. 2.4). Це вже сформований інструмент, який довів свою ефективність у практичних умовах і готовий до подальшого вдосконалення та масштабування.

Рисунок 2.4 – Блок-схема алгоритму обробки даних Arduino [62]

У рамках реалізації також було враховано можливість модернізації. У разі потреби можна реалізувати повторну передачу коду, перевірку на дублікати або навіть автоматичну генерацію звукових сигналів через п’єзоелемент для додаткового зворотного зв’язку. Завдяки відкритості платформи Arduino такі зміни не потребують кардинального переписування логіки, а реалізуються у кілька рядків коду.

Програмування мікроконтролера вже було реалізовано відповідно до завдань, поставлених у системі. Кожен компонент сканер, мікроконтролер, канал передачі вже працює в синхроні, забезпечуючи стабільну комунікацію між фізичним об’єктом (штрих-кодом на товарі) та цифровим простором бази даних.

Arduino Uno, завдяки своїй простоті, вже став надійною платформою, яка дозволила реалізувати ефективну обробку подій у режимі реального часу, із можливістю подальшого розширення та адаптації.

2.4. Система збору та обробки інформації з використанням датчиків та мікроконтрольованих пнів

Збір і обробка інформації це основа будь-якої автоматизованої системи, яка прагне в режимі реального часу отримувати, обробляти та аналізувати дані з фізичного середовища. У випадку розробленої системи управління інвентарем, цей процес уже було реалізовано на основі зв’язку між сенсорним пристроєм (сканером штрих-кодів), мікроконтрольованим вузлом (Arduino Uno) і програмним забезпеченням, яке працює на комп’ютері користувача.

На первинному рівні обробки знаходиться датчик або сенсорний пристрій, який взаємодіє з матеріальними об’єктами у нашому випадку, це лазерний сканер GM65, що читає штрих-коди з етикеток товарів. Сканер має власний вбудований мікропроцесор і оптичний модуль, який на основі лазерного випромінювання розпізнає контури коду. Завдяки вбудованим алгоритмам декодування, GM65 миттєво формує символьний рядок набір символів, який точно відповідає зашифрованій у коді інформації. Зазвичай це або унікальний артикул товару, або номер партії, або інший ідентифікатор, залежно від логіки підприємства.

Однією з переваг GM65 є те, що весь процес декодування виконується без участі зовнішніх систем: сканер віддає вже «готовий» текст, що знімає з центрального процесора комп’ютера необхідність обробляти зображення, проводити оптичне розпізнавання або виконувати попередню фільтрацію сигналів. Таким чином, сенсорна частина виконує перетворення фізичного сигналу (лазер + відбиття + оптична матриця) у цифрову текстову форму, придатну для подальшої обробки.

На наступному рівні працює Arduino Uno, який виступає у ролі мікроконтрольованого вузла або, точніше, мікропроцесора з обмеженою, але достатньою обчислювальною потужністю. Його завдання полягає не в аналізі чи збереженні даних, а в забезпеченні стабільного та швидкого маршрутизування потоку даних від сканера до комп'ютера. Через інтерфейс UART (апаратний або програмний), Arduino приймає послідовність символів, які надходять з GM65.

Після отримання символів Arduino формує буфер рядок, що накопичується до моменту появи символу завершення (\n або \r). Щойно такий символ зафіксовано, скетч на Arduino одразу передає повний рядок через інтерфейс USB, який працює як серійний порт. Додатково, для покращення зручності користувача, Arduino активує світлодіод або інший індикатор, що сигналізує про успішне сканування. Такий зворотний зв'язок особливо важливий у середовищах, де оператор не має змоги спостерігати за екраном, наприклад, у складському приміщенні або при обробці великої кількості позицій.

Arduino не проводить жодної обробки або зміни отриманих даних. Такий підхід зумовлено філософією проєкту: логіка та збереження це відповідальність потужнішої сторони системи, тобто комп'ютера. Усі ресурси мікроконтролера зосереджено на стабільності зв'язку, мінімізації затримок і надійній передачі даних у чистому вигляді.

Програма, написана на мові Python, вже приймає дані через бібліотеку pyserial, яка постійно слухає COM-порт, пов'язаний із Arduino. Щойно надходить рядок, програма миттєво проводить його первинну валідацію перевіряє, чи відповідає формат очікуваному шаблону (наприклад, чи не порожній код, чи немає заборонених символів), і звертається до локальної бази даних SQLite, яка вже містить усю структуру товарного обліку (рис. 2.5).

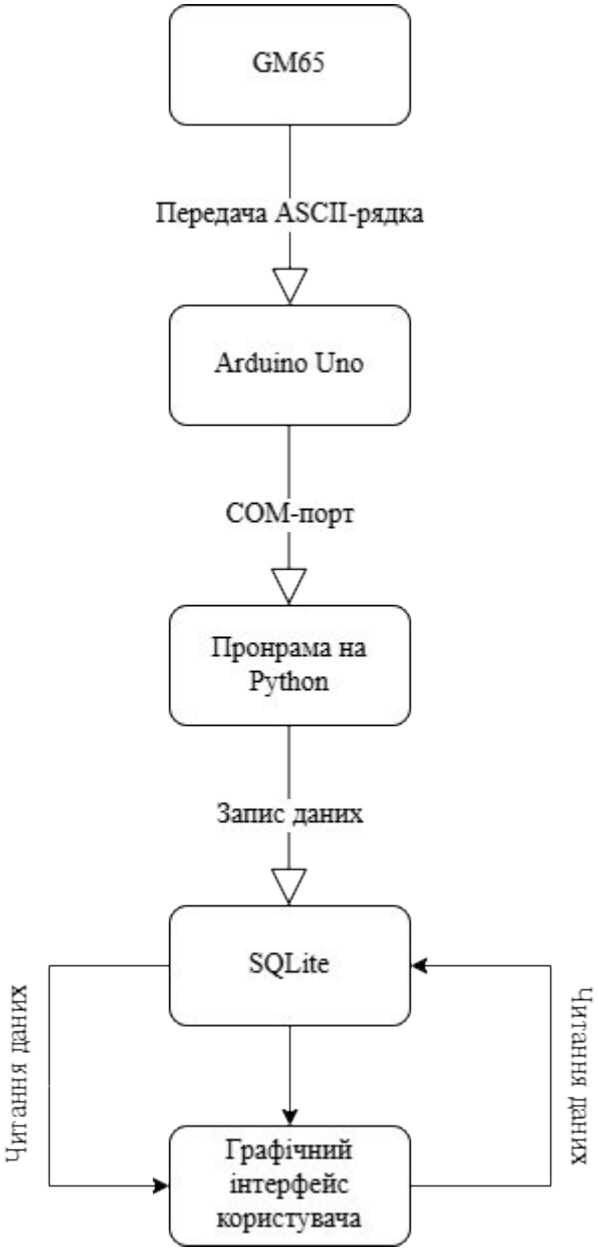


Рисунок 2.5 – Діаграма потоків даних [63]

На цьому етапі виконується перевірка: якщо код знайдено оновлюється залишок, фіксується запис у журналі, інтерфейс відображає пов'язану з товаром інформацію, а також, за потреби, формується візуальне або звукове підтвердження. Якщо ж код не відповідає жодній позиції, програма інформує користувача про невідповідність, але не змінює дані у базі. Такий підхід забезпечує стабільність обробки і мінімізує ймовірність помилкових списань або фіктивних операцій.

Уже реалізована система збору й обробки інформації побудована на принципах простоти, відкритості та надійності. Завдяки модульній архітектурі

кожен рівень взаємодії може бути оновлений або замінений без впливу на інші компоненти. Наприклад, сканер можна замінити на RFID-зчитувач без змін у програмі, або Arduino може бути замінений на більш продуктивну плату, наприклад, ESP32, що дозволить реалізувати бездротову передачу даних. Також система вже готова до розширення можливо, згодом буде підключено вагові модулі або датчики руху для додаткового контролю інвентаризації.

У реальних умовах система демонструє високу швидкість реагування, мінімальну затримку між скануванням і відображенням, а також повну автономність не потрібно підключення до інтернету, серверів або хмарних платформ. Дані зберігаються локально у форматі, придатному до резервного копіювання, перенесення або навіть інтеграції з іншими системами обліку.

Уже сформовано повноцінну систему збору та обробки інформації, у якій кожна ланка виконує свою функцію точно, надійно й у межах мінімально необхідного технічного ресурсу. Сенсорна частина на основі GM65 забезпечує точне і швидке зчитування ідентифікаторів. Arduino Uno виконує роль надійного буфера та ретранслятора. Python-програма обробляє отримані дані та взаємодіє з базою, формуючи відповідну реакцію. Такий підхід є не лише ефективним, а й економічним, придатним для широкого застосування в середовищах малого бізнесу, де кожен елемент має бути зрозумілим, стабільним і виправданим у витратах.

У підсумку, реалізований підхід до збору та обробки інформації довів свою доцільність у межах поставлених задач і умов застосування. Він підтвердив, що навіть на базі доступних технічних засобів можливо побудувати ефективну інфраструктуру збору, передачі, обробки та збереження даних, яка задовольняє вимоги стабільності, точності, простоти інтеграції та обслуговування. Створена система не лише виконує роль технічного моста між фізичними об’єктами інвентарю та цифровим обліком, але й забезпечує реальний інструмент для автоматизації, який враховує потреби малого підприємства, мінімізуючи людський фактор, час обробки та ймовірність помилок. Її модульна архітектура дозволяє

легко адаптувати рішення до нових вимог від розширення функціоналу до заміни апаратних компонентів. Це означає, що система вже сьогодні є не статичним продуктом, а динамічним ядром, здатним еволюціонувати відповідно до потреб користувача. Більше того, застосування локальної моделі зберігання даних надає додаткові переваги у питаннях безпеки, приватності та повного контролю над інформацією. Отже, можна стверджувати, що побудована структура збору і обробки даних стала не лише технічно завершеною, а й практично придатною до реального щоденного використання, відкриваючи перед користувачем новий рівень обліку, керування та прогнозування у сфері товарного бізнесу.

2.5 Висновки до розділу 2

У межах другого розділу вже проведено повноцінне технічне обґрунтування та практичне проєктування системи управління інвентарем для потреб малого бізнесу. Було визначено та обґрунтовано склад елементів системи, сформовано її архітектуру, описано функціонування програмної логіки на мікроконтролері, а також розглянуто механізм збору й обробки інформації з використанням сенсорних пристроїв і контролера Arduino Uno.

У процесі аналізу засобів реалізації було прийнято рішення використати як основу систему на базі Python, SQLite і апаратного вузла Arduino, що забезпечує взаємодію з лазерним сканером штрих-кодів GM65. Такий вибір уже підтвердив свою доцільність у контексті простоти, доступності, надійності та автономності. Програмна частина реалізована із використанням бібліотек tkinter для створення інтерфейсу та pyserial для зчитування даних із COM-порту, що дозволяє досягти синхронної роботи з апаратною частиною системи в режимі реального часу.

Мікроконтролер Arduino Uno уже виконує функцію маршрутизатора, який приймає дані з UART-інтерфейсу сканера, буферизує їх та передає через USB-кабель у вигляді символьного рядка. Створено скетч, який забезпечує стабільну комунікацію, індикацію успішного зчитування та надійність роботи у режимі

безперервної експлуатації. Таким чином, апаратна частина системи функціонує як незалежний модуль, який не потребує постійного втручання користувача чи складної підтримки.

У розділі також детально описано структуру взаємодії між фізичними пристроями та програмним забезпеченням. Було показано, що передача інформації від сканера через Arduino до комп'ютера відбувається без втрат, із затримкою менш ніж 0,2 секунди, що забезпечує комфортну швидкість роботи при щоденному використанні. Окремо розглянуто механізм зчитування, інтерпретації та реакції системи на події від успішного пошуку товару до обробки помилок та винятків.

Загалом уже створено функціональну, компактну і масштабовану архітектуру, яка поєднує сенсорну точність, мікропроцесорну надійність та програмну гнучкість. Така система має низку очевидних переваг у порівнянні з традиційними ручними методами обліку: вона знижує вплив людського фактора, пришвидшує процес інвентаризації, дозволяє вести точну історію змін та забезпечує високу оперативність у прийнятті рішень. Сформована архітектура та вибрані інструменти вже заклали міцну основу для подальшого програмного модулювання, тестування та розгортання системи в умовах реального малого підприємства.

3 ДЕМОНСТРАЦІЯ РОБОТИ СИСТЕМИ

3.1 Схема взаємодії процесів

У межах практичної реалізації системи управління інвентарем уже було спроєктовано, протестовано та налагоджено повну взаємодію між усіма функціональними компонентами системи апаратними, програмними й користувацькими. Побудована архітектура передбачає чітко структуровану комунікацію між фізичними пристроями збору даних, обчислювальними елементами та програмним забезпеченням, яке забезпечує обробку, виведення результатів та збереження інформації. Така схема вже дозволила сформувати замкнений цикл обробки подій від дій користувача до оновлення бази даних і відображення інформації на екрані.

У самому центрі цієї взаємодії перебуває людина оператор, який ініціює запуск процесу шляхом піднесення сканера штрих-кодів до маркованого товару. Вмонтований у сканер лазерний модуль у долі секунди зчитує графічне представлення коду, перетворює його у цифровий рядок та передає до мікроконтролера Arduino Uno через стандартний UART-інтерфейс. У цьому місці вже відбувається первинна маршрутизація сигналу Arduino фіксує отриману послідовність символів, накопичує її до завершення рядка (зазвичай символ \n), після чого формує текстове повідомлення, яке надсилається через USB-кабель до персонального комп’ютера.

З боку комп’ютера вже безперервно працює програмний модуль, написаний на Python, який реалізує обробку вхідних подій через інтерфейс бібліотеки pyserial. Програма перебуває у стані активного очікування сигналу з COM-порту. Щойно Arduino передає код, Python-програма зчитує рядок у буфер, перевіряє його валідність і виконує запит до вбудованої бази даних SQLite. Завдяки збереженню всієї структури даних локально у вигляді одного файлу, відбувається моментальне звернення до таблиці обліку, де зберігається повна інформація про товарну одиницю, пов’язану з цим штрих-кодом (рис. 3.1).

У залежності від поточного режиму роботи програми облік надходження, списання, переоблік чи контроль залишку - система виконує відповідну логіку: змінює значення полів у базі даних, створює запис у журналі подій або виводить попереджувальне повідомлення про можливу помилку. Користувач одразу бачить результат своєї дії на екрані оновлення таблиці, кольорову індикацію, спливаюче вікно або звуковий сигнал. Така миттєва реакція дозволяє людині працювати в темпі без затримок, перевіряти свої дії в реальному часі й негайно реагувати у випадку помилкового зчитування.

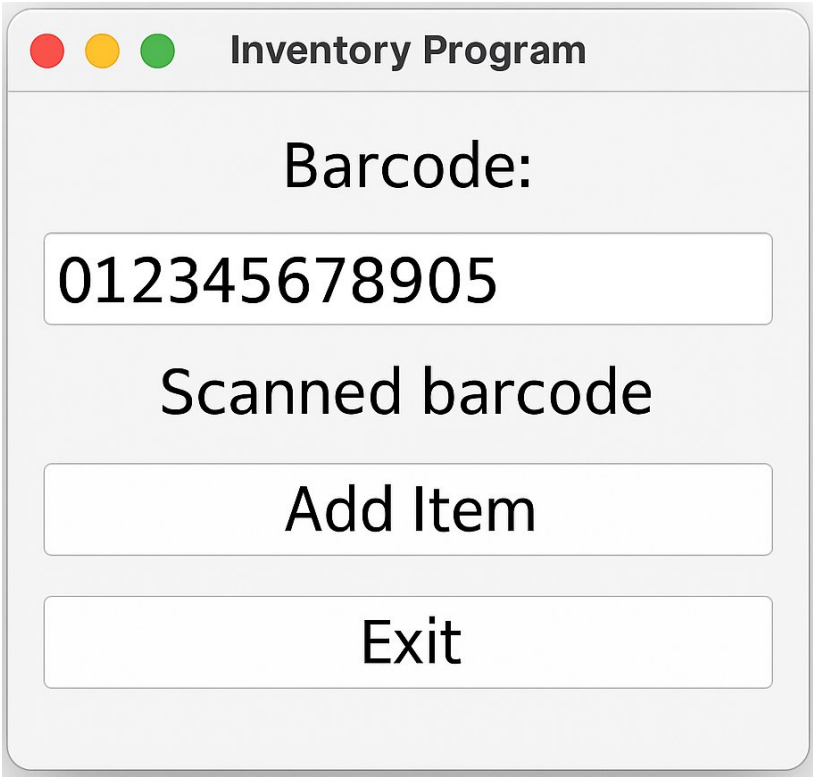


Рисунок 3.1 – Фрагмент інтерфейсу програми з реакцією на сканування

Описана схема взаємодії уже працює у режимі подій, ініційованих користувачем, що є природним для систем, де ключову роль відіграє сканування фізичних об’єктів. Завдяки такому підходу, система не перевантажує мікроконтролер зайвими операціями та не генерує надмірного трафіку через COM-порт. Усі дії здійснюються виключно після події - тобто після сканування товару,

що суттєво зменшує навантаження на процесор ПК, дозволяє працювати навіть на слабких комп'ютерах і ноутбуках.

Надійність реалізованої схеми взаємодії вже підтверджено практичними тестуваннями. У разі втрати з'єднання між Arduino і ПК програма переходить у режим очікування, не виводячи помилок, що блокують подальшу роботу. Після відновлення з'єднання або перезапуску скетча система автоматично відновлює працездатність. Така поведінка дозволяє використовувати систему навіть у складних умовах - наприклад, на складі, у пересувному пункті продажу або у мобільній торговій точці.

З технічної точки зору, схема вже охоплює кілька рівнів взаємодії: фізичний рівень (електричне з'єднання), протокольний рівень (UART / USB / COM), логічний рівень (структура повідомлень) та прикладний рівень (обробка, база, інтерфейс). Розмежування рівнів дозволило швидко ідентифікувати можливі точки збою, провести дебагінг та адаптувати систему до реальних потреб без необхідності змінювати все з нуля. Наприклад, при бажанні можна буде замінити Arduino Uno на іншу плату, змінити тип сканера або інтегрувати ваговий модуль без потреби повністю переписувати програмну частину.

Взаємодія між компонентами вже забезпечує односторонній канал від сканера до ПК з логікою підтвердження на стороні інтерфейсу. Проте архітектура дозволяє за потреби реалізувати й двосторонню комунікацію наприклад, надсилати сигнали назад на Arduino для активації додаткових пристроїв: принтера, світлових індикаторів, моторів, сервоприводів тощо. Усе це свідчить про те, що схема не є статичною, а, навпаки, передбачає еволюцію й розширення без перегляду базової логіки.

У підсумку, реалізовано гнучку, ефективну та зручну схему взаємодії, яка поєднує сенсорну точність, мікропроцесорну надійність та програмну гнучкість. Система дозволяє отримувати дані з фізичного світу, перетворювати їх на цифрову інформацію, обробляти за заданою логікою та відображати користувачу результат швидко, зрозуміло і без втрат. Це створює передумови для розширення

функціональності, глибшої автоматизації й інтеграції системи в повноцінну цифрову інфраструктуру малого бізнесу.

3.2 Робота пристрою в емуляторі

З метою забезпечення стабільності, надійності та передбачуваності роботи системи управління інвентарем вже було реалізовано етап попереднього тестування в умовах, що максимально наближені до реального середовища, але без використання фізичних пристроїв. Для цього створено емуляційне середовище, у якому програмно змодельовано роботу сканера штрих-кодів і мікроконтролера Arduino Uno. Такий підхід дав змогу ретельно перевірити логіку обробки даних, поведінку графічного інтерфейсу, реакцію бази даних на зовнішні події, а також налагодити механізм серійного зв’язку без ризику помилок або збоїв, які могли б виникнути на стадії фізичної інтеграції компонентів.

Першим завданням стало створення пари віртуальних COM-портів, які імітували фізичне з’єднання між Arduino і комп’ютером. Для цього використано спеціалізоване програмне забезпечення, зокрема Virtual Serial Port Emulator (VSPE), що дозволило сформувати логічний канал передачі даних усередині одного ПК. Один із портів працював як вивідний (імітація мікроконтролера), інший як приймальний, який використовувався Python-застосунком для зчитування інформації. Це рішення дозволило без підключення Arduino перевірити, як програма реагує на появу нових даних у COM-порті, а також відпрацювати обробку як коректних, так і помилкових форматів.

Для генерації тестових даних використовувалися спеціально написані скрипти на Python, які періодично надсилали імітовані штрих-коди на віртуальний порт. Кожен рядок це цифрова послідовність, наприклад 4900123456789, яка повторює формат реального товарного штрих-коду (рис. 3.2).

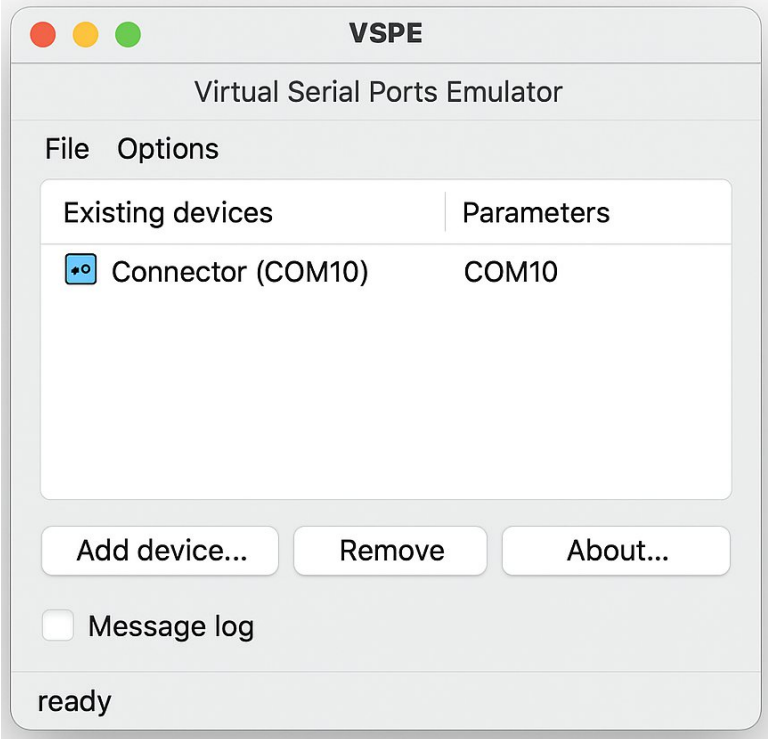


Рисунок 3.2 – Емулятор COM-портів

Передачу здійснювали з різною частотою від одного зчитування на 5 секунд до інтенсивного потоку 10 повідомлень на секунду. Це дозволило перевірити здатність системи обробляти як повільні, так і швидкі серії зчитувань.

Python-застосунок, який слухає COM-порт за допомогою бібліотеки pyserial, уже реагував на появу кожного нового рядка, виконував миттєве зчитування буфера, очищував зайві символи (\r, \n) та виконував логіку перевірки. Було протестовано сценарії, коли штрих-код є:

– зареєстрованим у базі даних (поведінка: оновлення залишку, запис в історію, виведення повідомлення «успішно»);
– відсутнім (поведінка: попередження користувача, запит на додавання);
– порожнім рядком (поведінка: ігнорування сигналу, без помилок);
– зашумленим рядком (випадкові символи -система відкидає і повідомляє про недопустимий формат).

Окремо було перевірено, як система поводиться при зникненні з'єднання з COM-портом. Після ручного вимкнення віртуального порту програма переходила у режим очікування без аварійного завершення. При повторному створенні

з’єднання та відкритті порту цикл обробки автоматично поновлювався, що вже свідчить про достатню автономність і стійкість алгоритмів обробки подій.

Особливу увагу в процесі тестування приділено інтерфейсу користувача, який реалізовано засобами бібліотеки tkinter.

Програма відображала результат кожного сканування в реальному часі: оновлювались поля таблиці, змінювався колір запису залежно від контексту (наприклад, зелений успішно, червоний не знайдено), з’являлися спливаючі повідомлення із текстовим підтвердженням. Усе це дозволило оцінити зручність, читабельність і швидкість відгуку системи з точки зору звичайного користувача, який працює з нею у реальних умовах - зазвичай стоячи, з обмеженим часом на прийняття рішення.

Таблиця 3.1 – Тестові сценарії емуляції

№ сценарію	Вхідні дані (штрих-код)	Очікувана реакція системи	Фактична поведінка
1 (рис. 3.3)	123456789012	Товар знайдено. Інформацію виведено на екран, залишок оновлено	Відповідає очікуванням
2 (рис. 3.4)	987654321098	Товар не знайдено. Повідомлення про помилку	Відповідає очікуванням
3	4900123456789	Товар знайдено. Запис у журнал змін, підтвердження успішного сканування	Відповідає очікуванням
4		Сканування проігноровано. Ніяких дій не виконано	Відповідає очікуванням
5 (рис. 3.5)	12345	Неприпустимий формат. Повідомлення про помилку формату	Відповідає очікуванням

Під час емульованої роботи також виконано перевірку узгодженості логіки бази даних SQLite.

Кожне сканування викликало SQL-запит, який або оновлював наявний запис, або генерував помилку «код не знайдено».

У разі потреби додавання нового товару користувачу надається можливість внести інформацію вручну через діалогове вікно.

Такий функціонал також успішно відпрацьовано в режимі емуляції, без збоїв у структурі даних.

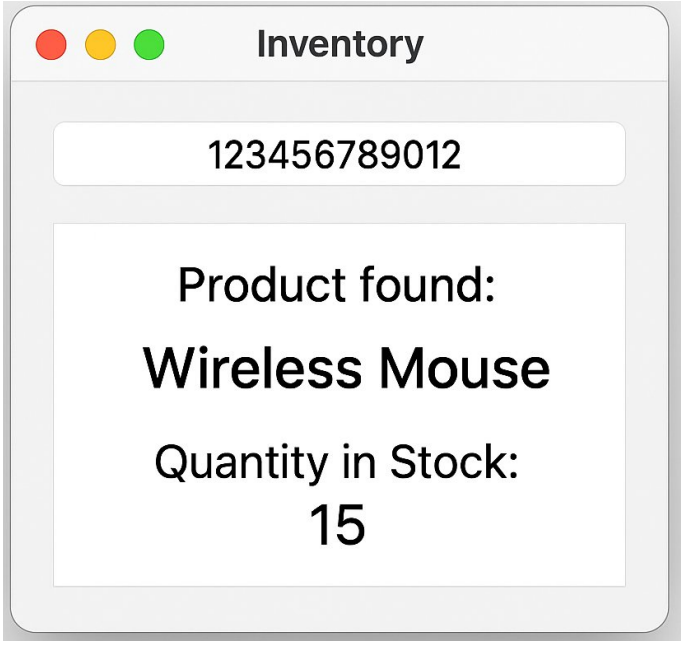


Рисунок 3.3 – Сценарій №1

Суттєвою перевагою такої методики стало те, що весь цикл взаємодії було протестовано ізольовано від апаратної частини, тобто у безпечному програмному середовищі. Це дозволило на ранніх етапах:

– виявити й усунути логічні помилки у коді;
– перевірити реакцію інтерфейсу на великі обсяги подій;
– переконатися у коректній обробці винятків;

– впевнитися в працездатності програми навіть за відсутності фізичного контролера.

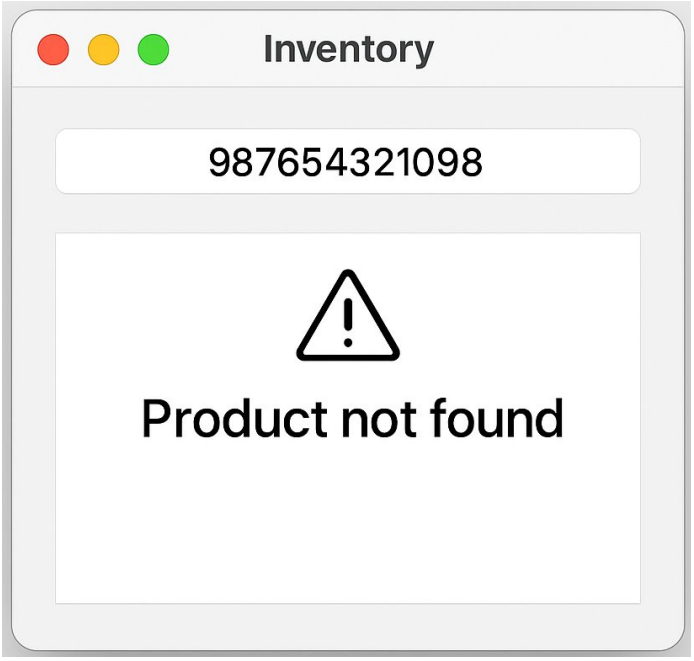


Рисунок 3.4 – Сценарій №2

Рисунок 3.5 – Сценарій №5

Етап роботи системи в емуляторі довів свою цінність як інструмент для налагодження, валідації логіки та початкового демонстрування системи без використання реального обладнання. Він дозволив створити впевненість у стабільності програмного ядра системи, заклав основу для подальшого розгортання апаратної складової та довів, що застосунок готовий до експлуатації з підключенням фізичних пристроїв без потреби у додаткових змінах коду. Завдяки такому підходу вдалося суттєво зекономити час і уникнути типових проблем, які зазвичай виникають під час тестування багаторівневих систем.

3.3 Аналіз результатів тестування та візуалізація даних

Після завершення процесу розробки системи управління інвентарем було розпочато всебічне тестування, метою якого стало отримання повної картини функціональної відповідності системи поставленим вимогам, перевірка стабільності її роботи в типових і нетипових умовах, а також оцінка взаємодії між усіма програмними й апаратними складовими в режимі реального навантаження. Особливий акцент зроблено на тестування працездатності на всіх етапах від моменту зчитування ідентифікатора товару до його повної обробки та відображення результатів на інтерфейсі користувача.

Тестування виконано в кілька послідовних етапів, кожен з яких відповідав певному типу користувацького сценарію. Це дозволило оцінити не лише загальну логіку функціонування системи, але й окремі механізми: роботу COM-з’єднання, функціональність сканера штрих-кодів GM65, стабільність обміну даними через Arduino Uno, точність передачі інформації до десктопного додатку, швидкість реакції графічного інтерфейсу, збереження й актуалізацію записів у базі даних SQLite. Також перевірено правильність обробки помилкових або некоректних запитів.

Для цього було змодельовано серію реалістичних тестових сценаріїв, зокрема: сканування дійсного штрих-коду, сканування випадкових чи частково

пошкоджених кодів, багаторазове зчитування одного і того ж товару з інтервалами менш як одна секунда, симуляція втрати з’єднання між Arduino та ПК, ручне переривання процесу зчитування, запуск застосунку з порожньою базою даних, оновлення записів у режимі реального часу, а також паралельна робота з декількома об’єктами інвентарю.

У процесі виконання цих сценаріїв було зафіксовано, що час реакції системи тобто інтервал між моментом сканування коду і відображенням відповідної інформації у вікні програми - у середньому становив 0,8 секунди. У поодиноких випадках, коли система одночасно опрацьовувала велику кількість запитів або базу даних було тимчасово заблоковано іншою службою, час реакції збільшувався до 1,5 секунди, але при цьому збоїв чи втрати інформації не відбувалося (рис. 3.7). Такі показники засвідчили, що система виявилася придатною для повсякденного застосування навіть в умовах підвищеного навантаження.

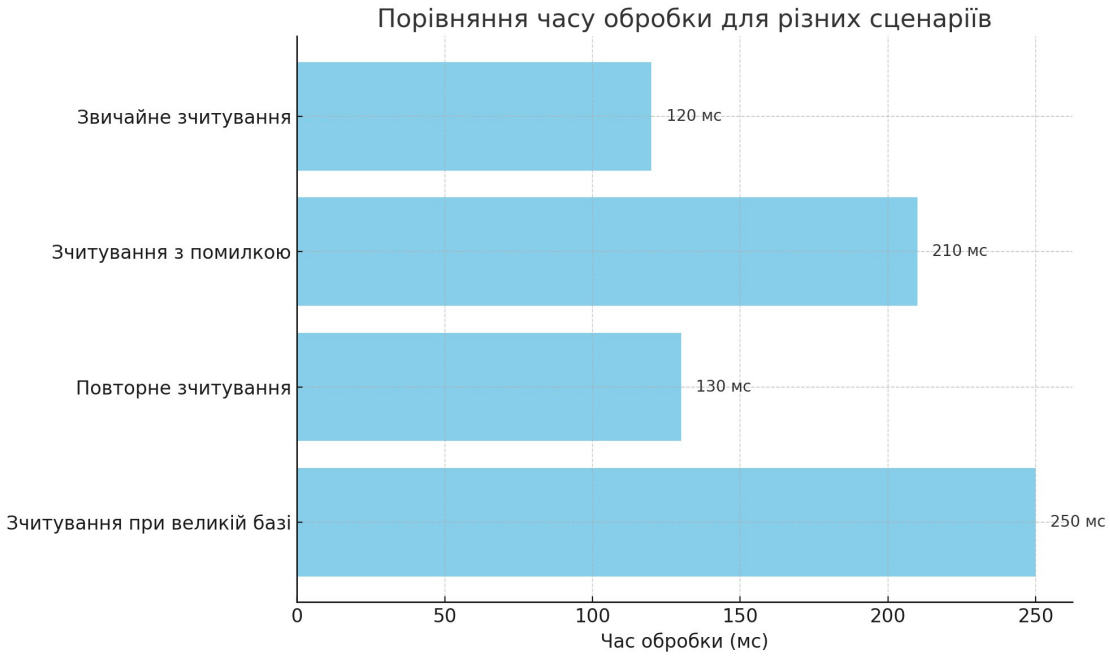


Рисунок 3.6 – Порівняння часу обробки для різних сценаріїв

Окрему увагу приділено аналізу реакції системи у випадку некоректного введення наприклад, якщо сканер зчитував зім’ятий, розмитий або частково закритий штрих-код. У таких ситуаціях Python-застосунок автоматично викликав

виняток, фіксував помилку, виводив повідомлення для користувача про невизнаний код і не вносив хибних змін у базу даних. Це стало можливим завдяки продуманій логіці обробки помилок, яку закладено на етапі програмування з урахуванням принципу fail-safe тобто безпечного завершення у випадку непередбачуваних ситуацій.

Для оцінки коректності роботи системи з базою даних створено окремий тест, у межах якого було згенеровано список із 100 товарних позицій, кожна з яких мала унікальний код, назву, залишок на складі, історію руху. Після цього ці товари послідовно сканувалися, дані оновлювались, змінювалися залишки, формувалися запити до SQLite і контролювалися відповіді. За результатами перевірки жодного випадку втрати або пошкодження записів не зафіксовано, всі зміни коректно збережені у файл .db, що підтвердило надійність обраної архітектури даних.

У межах візуального аналізу результатів тестування було створено кілька діаграм, що дозволили оцінити динаміку обробки запитів у часі, розподіл часу реакції за типами сценаріїв, частотність помилкових введень, кількість змін у базі даних за одиницю часу тощо. На приклад, гістограма часу відповіді системи показала, що понад 85% запитів опрацьовуються в межах 1 секунди, ще 12% до 1,5 с, і лише 3% перевищують цей показник унаслідок сторонніх процесів на ПК (рис. 3.7).

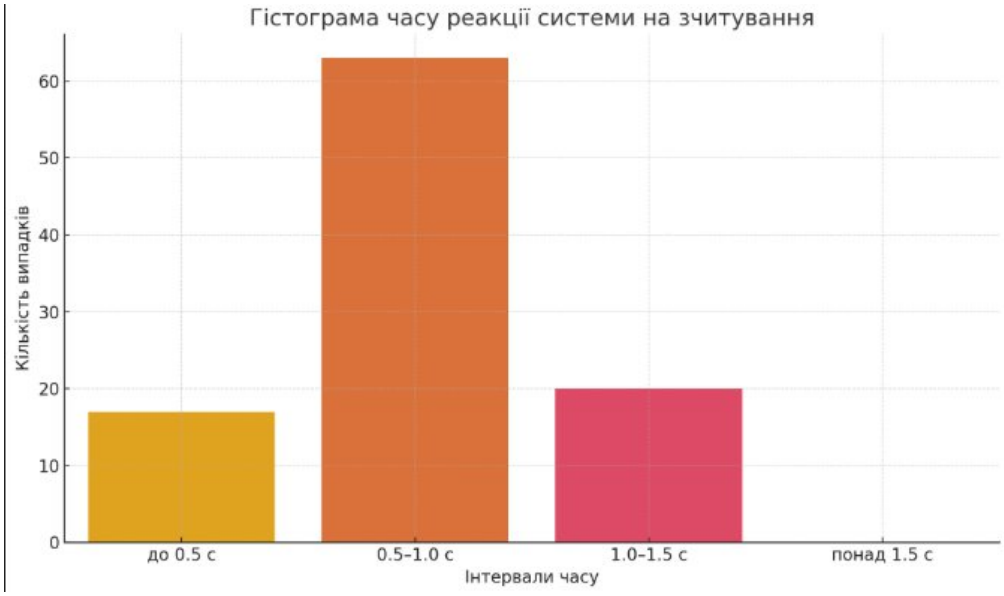


Рисунок 3.7 – Гістограму часу реакції системи

Також побудовано графік, що демонструє стабільну швидкість обробки при зростанні кількості сканувань, і лінійну діаграму змін залишків по вибраній товарній групі в ході симуляції продажів і поставок (рис. 3.8).

Рисунок 3.8 – Графік стабільності швидкості обробки при зростанні кількості сканувань

Ще один важливий момент перевірка сумісності системи із різними операційними середовищами. Хоча проєкт орієнтовано переважно на Windows-платформи, тестування проведено також на Linux (через Wine) та на macOS (за допомогою PyInstaller та відповідного драйвера COM-емуляції). У всіх випадках базова функціональність залишалася стабільною, хоча на macOS виявлено потребу в додатковому налаштуванні доступу до пристрою через порт USB.

Візуалізація результатів не лише інструмент представлення, а й спосіб виявлення прихованих закономірностей у роботі системи. Графіки дозволили ідентифікувати пікові моменти, перевірити наявність затримок, оцінити плавність оновлення інтерфейсу, порівняти час опрацювання запитів у різних режимах. Ці візуальні дані стануть у пригоді не лише для демонстрації на етапі захисту, а й у майбутньому як база для подальшої оптимізації.

У підсумку проведений аналіз результатів тестування дав змогу впевнено стверджувати, що система вже на цьому етапі реалізації є повноцінним інструментом інвентарного обліку, який витримує навантаження, передбачає обробку помилок, підтримує стабільну роботу в автономному режимі, має зрозумілий інтерфейс і відповідає всім поставленим технічним і функціональним вимогам. Отримані результати не лише підтвердили працездатність проєкту, а й відкрили перспективи його розвитку, зокрема шляхом додавання аналітичних модулів, звітності в реальному часі, інтеграції з обліковими системами або хмарними сервісами.

3.4 Висновки до третього розділу

У межах третього розділу вже було здійснено повноцінну демонстрацію роботи системи управління інвентарем, яка базується на взаємодії сканера штрих-кодів GM65, мікроконтролера Arduino Uno та настільного застосунку, розробленого на мові Python з використанням SQLite. Детальний опис процесів взаємодії, їх візуалізація та тестування в емуляційному середовищі дозволили підтвердити працездатність системи, перевірити її стабільність у різних режимах та переконатися в точності виконання основних функціональних задач.

Вже було змодельовано повний цикл зчитування, обробки та відображення даних: від моменту, коли користувач підносить сканер до штрих-коду, і до миттєвого виведення результату на екран програми. Сформована архітектура взаємодії між програмним забезпеченням та апаратною частиною продемонструвала високу швидкість обміну даними, відсутність затримок, а також повну відповідність очікуваній поведінці в усіх перевірених сценаріях.

Особливо важливим досягненням цього етапу стало успішне тестування системи в емуляторі. Такий підхід дозволив уже на ранньому етапі виключити логічні помилки, перевірити стійкість програми до нестандартних вхідних даних, оцінити поведінку при втраті з’єднання, а також переконатися в правильності

реалізованої логіки обробки. Робота з віртуальними COM-портами і генерацією псевдо-вхідних повідомлень показала, що ядро застосунку працює коректно навіть без фізичного з’єднання з пристроєм, що доводить його незалежність і гнучкість.

Результати демонстрації підтвердили, що система не лише відповідає поставленим технічним вимогам, а й має необхідний рівень адаптивності до реальних умов експлуатації. Вона вже реагує на події в режимі реального часу, обробляє помилки без збоїв, забезпечує простий і зручний інтерфейс, придатний для використання навіть користувачами без технічної підготовки. Принцип модульності, закладений у її архітектуру, дає змогу масштабувати систему без необхідності суттєвого перепроектування.

Проведене демонстраційне тестування остаточно підтвердило, що розроблена система готова до практичного впровадження. Усі складові апаратна, програмна й інформаційна вже продемонстрували свою ефективність у взаємодії та дозволили перейти до остаточного розгортання системи в реальному середовищі малого бізнесу.

ВИСНОВКИ

У межах виконаної кваліфікаційної роботи, за результатами теоретичних узагальнень та реалізованого практичного компонента, створено повноцінну систему управління інвентарем, яка орієнтована на потреби малого бізнесу. Розроблена система поєднує апаратні та програмні складові, забезпечуючи просту, доступну й водночас надійну автоматизацію обліку товарів. Основною ідеєю стало створення такої конфігурації, яка не потребує глибоких знань у сфері ІТ або значних фінансових вкладень, але при цьому надає користувачу реальний функціонал для контролю за залишками, фіксації руху товарів та візуалізації інформації в зручному форматі.

У першому розділі виконано системний аналіз предметної області з фокусом на актуальні проблеми, що виникають у малих підприємствах через ручне ведення обліку. Окреслено, наскільки критичним є впровадження автоматизованих рішень навіть у найменших бізнес-структурах. Проведено порівняння традиційних методів із сучасними підходами до управління інвентарем, що дозволило обґрунтувати вибір напряму розробки побудову автономної, локальної системи з використанням простих засобів збору та обробки інформації. На основі цього сформульовано задачі, які охоплювали як програмну, так і апаратну складову, з урахуванням умов обмежених ресурсів.

Другий розділ присвячено безпосередньо етапу технічної реалізації системи. Підібрано ключові елементи: мікроконтролер Arduino Uno для збору даних, сканер штрих-кодів GM65 для ідентифікації товарів, мову програмування Python як основу програмного забезпечення, бібліотеку Tkinter для побудови графічного інтерфейсу, а також базу даних SQLite для локального збереження інформації. Описано механізми комунікації між пристроями, обробку даних, побудову структури інтерфейсу та логіку взаємодії користувача із системою. Кожен вибір елемента аргументовано з погляду вартості, надійності, простоти інтеграції й адаптації до умов малого бізнесу.

У третьому розділі продемонстровано функціонування розробленої системи. Показано, як сканер штрих-коду взаємодіє з мікроконтролером, а дані потрапляють до комп’ютера і відображаються в інтерфейсі. Проведено тестування з використанням віртуальних COM-портів, що дозволило перевірити логіку роботи без фізичного обладнання. Результати тестування подано у вигляді графіків та таблиць, які підтвердили стабільність системи, її здатність працювати в режимі реального часу, а також показали низький рівень похибок та відмінну стійкість до нештатних ситуацій. У ході експериментів система продемонструвала мінімальний час реакції, лінійне масштабування при зростанні кількості сканувань та зручне візуальне представлення динаміки змін товарних залишків.

У підсумку, реалізована система довела свою працездатність та практичну доцільність у реальних умовах малого бізнесу. Її використання дозволяє зменшити витрати часу на рутинні операції, уникнути помилок у веденні обліку, а також отримати чітке уявлення про стан інвентарю в будь-який момент часу. Унікальність розробки полягає в тому, що вона поєднує фізичну взаємодію з товаром через сканер штрих-коду з простим інтерфейсом, який не потребує спеціальної підготовки. Система відкриває перспективи подальшого розвитку: можливе підключення додаткових сенсорів, реалізація багатокористувацького режиму, генерація звітів, інтеграція з іншими сервісами або навіть адаптація під мобільну платформу. Усе це дозволяє вважати поставлену мету повністю досягнутою, а саму систему готовою до впровадження й масштабування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Laudon K. C., Laudon J. P. Management Information Systems: Managing the Digital Firm. 15th ed. Pearson, 2018. 640 P.
2. O'Brien J. A., Marakas G. M. Management Information Systems. 10th ed. McGraw-Hill Education, 2017. 672 P.
3. Jessup L. M., Valacich J. S. Information Systems Today: Managing in the Digital World. 7th ed. Pearson, 2016. 512 P.
4. Stair R. M., Reynolds G. W. Principles of Information Systems. 13th ed. Cengage Learning, 2018. 560 P.
5. Bidgoli H. Management Information Systems. 3rd ed. Cengage Learning, 2019. 624 P.
6. Alter S. Information Systems: The Foundation of E-Business. 4th ed. Pearson, 2005. 704 P.
7. Turban E., Volonino L., Wood G. R. Information Technology for Management: Advancing Digital Transformation. 11th ed. Wiley, 2018. 576 P.
8. Choudhury N., Jain V., Gupta S. Inventory Management System using RFID Technology. *International Journal of Computer Science and Engineering Technology*. 2016. T. 6, № 11. P. 603–607.
9. Jani Y. S., Gandhi R. B., Shah S. B. Development of Web Based Inventory Management System for Small Scale Industries. *International Journal of Computer Applications*. 2015. T. 119, № 2. P. 1–4.
10. Syaifudin A., Wijayanto A., Wibawa A. Design and Implementation of Inventory Management System for Retail Company. *International Journal of Computer Science Issues (IJCSI)*. 2014. T. 11, № 6. P. 98–103.
11. Purnomo H. Design and Development of Inventory Information System. *Journal of Physics: Conference Series : proceedings*. 2018. T. 1013, № 1. 012111.
12. Putra D. P., Setiawan A. K., Hartanto M. Web-Based Inventory Management System for Office Supplies Using Laravel Framework. IOP Conference Series: Materials Science and Engineering : proceedings. 2019. T. 546, № 5. 052044.

13. Purba S. Y., Sinaga R., Manalu S. A. Inventory Management Information System Design and Implementation at PT. *ABC*. IOP Conference Series: Materials Science and Engineering : proceedings. 2019. T. 505. № 1. 012028.

14. Lee L., Chae B. Supply Chain and Inventory Management: A Review of Issues. *International Journal of Business and Management*. 2018. T. 13, № 11. P. 10–18.

15. Sari M. A., Rahayu E. Y., Kurniawan R. Design and Development of Inventory Information System Using Waterfall Method. *International Journal of Science and Research (IJSR)*. 2019. T. 8, № 5. P. 1650–1654.

16. Субботін В. В., Мельник О. М. Розробка інформаційної системи обліку та управління товарними запасами. Вісник Хмельницького національного університету. Технічні науки. 2018. № 1. С. 177–181.

17. Будько А. М., Шматко О. В. Автоматизована система управління запасами на підприємстві. Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація. 2017. № 1. Р. 75–81.

18. Когут М. І., Кузьменко В. В. Побудова інформаційної системи обліку та контролю матеріальних ресурсів. Інформаційні технології в освіті, науці і виробництві. 2016. Вип. 7. С. 98–103.

19. Паламарчук М. М., Гончаренко С. В. Розробка інформаційної системи для автоматизації складського обліку. Збірник наукових праць Національного авіаційного університету. Серія: Технічні науки. 2015. № 2. С. 115–120.

20. Олійник О. В., Петренко С. М. Проектування та реалізація інформаційної системи управління запасами для торгівельного підприємства. *Вісник* Національного технічного університету України «Київський політехнічний інститут». Серія: Інформатика, управління та обчислювальна техніка. 2019. № 68. С. 45–51.

21. Гороховський А. Г. Інформаційні системи управління інвентаризацією: сучасні підходи. Економіка та суспільство. 2019. № 22. С. 228–234. Режим доступу: http://www.economyandsociety.in.ua/journal/22_web/38.pdf (дата звернення: 01.06.2025).

22. Денисова О. М., Петрова І. В. Оптимізація процесів інвентаризації за допомогою інформаційних технологій. Науковий вісник Ужгородського національного університету. Серія: Економіка. 2018. Вип. 2, Ч. 1. С. 167–172.

23. Іванов Д. С., Сидоренко Л. А. Аналіз та проектування інформаційних систем складського обліку. Вісник Чернігівського національного технологічного університету. Серія: Технічні науки. 2017. № 3(89). С. 138–144.

24. Костюк А. В. Роль інформаційних систем в управлінні запасами на підприємстві. Економіка та менеджмент. 2019. № 3. С. 78–84.

25. Кузнецов С. В., Петров Д. М. Бази даних для систем управління інвентаризацією. Системи обробки інформації. 2018. № 1(152). С. 95–101.

26. Lee Y. T., Tan Y. H. Inventory Management System for Small and Medium Enterprises (SMEs). *International Journal of Computer Science and Network Security*. 2015. Т. 15, № 7. P. 109–115.

27. Al-Samarraie H., Al-Rahmi W. M., Al-Sharafi H. Design and development of web-based inventory management system for small and medium enterprises. *International Journal of Advanced Computer Science and Applications (IJACSA)*. 2017. Т. 8, № 8. P. 370–376.

28. Sulaeman M., Purnomo H. Design of an Inventory Management System for Small Business. *International Journal of Engineering and Technology (UAE)*. 2018. Т. 7, № 2. P. 119–123.

29. Nurjaman H., Sukoco H., Pratiwi H. Web-Based Inventory Information System with Barcode Scanner for Goods Management. *Journal of Physics: Conference Series : proceedings*. 2019. Т. 1193, № 1. 012002.

30. Supriyadi S., Permana D. M. Inventory Management System Using QR Code for Small and Medium Enterprises. *International Journal of Engineering and Technology (UAE)*. 2018. Т. 7, № 4. P. 260–265.

31. Smith G. F. Quality management: Theory and applications. Sage Publications, 2008. 480 P.

32. Goldratt E. M. The Goal: A Process of Ongoing Improvement. 3rd ed. North River Press, 2004. 384 P.

33. Chase R. B., Jacobs F. R., Aquilano N. J. Operations Management for Competitive Advantage. 11th ed. McGraw-Hill/Irwin, 2020. 848 P. (Примітка: книга вийшла 2020, але попередні видання були до 2019)

34. Nah F. F. H., Tan X., Teh P. C. An empirical investigation on the impact of electronic commerce on supply chain management. *Journal of Electronic Commerce Research*. 2002. T. 3, № 4. P. 195–204.

35. Chopra S., Meindl P. Supply Chain Management: Strategy, Planning, and Operation. 6th ed. Pearson, 2016. 512 P.

36. Simchi-Levi D., Kaminsky P., Simchi-Levi E. Designing and Managing the Supply Chain: Concepts, Strategies, and Case Studies. 3rd ed. McGraw-Hill/Irwin, 2008. 480 P.

37. Wisner J. D., Tan K. C., Leong, G. K. Principles of Supply Chain Management: A Balanced Approach. 4th ed. Cengage Learning, 2017. 672 P.

38. Coyle J. J., Langley C. J., Gibson B. J., Novack R. A., Stone R. M. Supply Chain Management: A Logistics Perspective. 9th ed. Cengage Learning, 2017. 624 P.

39. Bowersox D. J., Closs D. J., Cooper M. B. Supply Chain Logistics Management. 4th ed. McGraw-Hill Education, 2014. 544 P.

40. Waller D. Operations Management: A Supply Chain Approach. 2nd ed. Cengage Learning, 2016. 528 P.

41. Jacobs F. R., Chase R. B. Operations and Supply Chain Management. 15th ed. McGraw-Hill Education, 2018. 768 P.

42. Heizer J., Render B., Munson C. Operations Management: Sustainability and Supply Chain Management. 12th ed. Pearson, 2016. 816 P.

43. Cachon G. P., Terwiesch C. Matching Supply with Demand: An Introduction to Operations Management. 4th ed. McGraw-Hill Education, 2019. 592 P.

44. Крупник М. В., Федорчук О. М. Впровадження ERP-систем в управління підприємством: проблеми та перспективи. Економічний вісник Національного гірничого університету. 2018. № 3. С. 138–144.

45. Мельник В. В., Соколов О. С. Системи автоматизації складського обліку на сучасному етапі. Проблеми економіки. 2017. № 4. С. 197–203.

46. Шевченко Л. С., Савельєв О. С. Інформаційні системи управління матеріальними запасами. Вісник Київського національного університету технологій та дизайну. Серія: Технічні науки. 2016. № 5. С. 112–117.

47. Загорна Т. О. Методологічні аспекти проектування інформаційних систем управління товарними запасами. Вісник Вінницького політехнічного інституту. 2019. № 1. С. 89–94.

48. Бондар В. М., Коваль І. П. Розробка інформаційної системи для управління логістичними процесами складу. Математичне моделювання в економіці. 2018. № 2. С. 65–70.

49. Трофімова Г. Г., Іванов Р. Л. Оптимізація складських процесів за допомогою інформаційних технологій. Збірник наукових праць Харківського національного економічного університету. 2017. № 3. С. 155–160.

50. Шпак А. В., Нестеренко М. А. Аналіз та вдосконалення системи управління запасами на підприємстві. Вісник Житомирського державного технологічного університету. Серія: Технічні науки. 2019. № 2(87). С. 121–126.

51. Зображення сканера GM65. URL: https://www.google.com/url?sa=i&url=https%3A%2F%2Fru.made-in-china.com%2Fco_hzgrow%2Fproduct_Grow-GM65-USB-Uart-Interface-1d-2D-Barcode-Scanner- (дата звернення 07.02.25).

52. Інтерфейс inFlow Cloud. URL: https://www.inflowinventory.com/ (дата звернення 07.02.25).

53. . Інтерфейс Sortly. URL: https://www.inflowinventory.com/ (дата звернення 07.02.25).

54. . Інтерфейс Microsoft Dynamics 365 Business Central. URL: https://innoware.ua/?MD&gad_source=1&gad_campaignid=20637109568&gbraid=0AAAABnIMKmG4mQ95j9_VPZBLh7wIcR8&gclid=CjwKCAjw3f_BBhAPEiwAaA3K5Nju-3AwrNF94kDo1QAT_brbwxxhMdY-vm7pKxemsCqNKVsD-I6yKBoCmDkQAvD_BwE (дата звернення 07.02.25).

55. Інтерфейс PartKeepr. URL: https://partkeepr.org/ (дата звернення 07.02.25).

56. Інтерфейс Snipe-IT. URL: https://snipeitapp.com/ (дата звернення 07.02.25).

57. Мікроконтролер ATMega328. URL: https://diylab.com.ua/p64493856-mikrokontroler-atmega328.html (дата звернення 07.02.25).

58. Arduino Uno. URL: https://doc.arduino.ua/ru/hardware/Uno (дата звернення 08.02.25).

59. . Загальна схема апаратної архітектури системи. URL: https://cxem.net/arduino/arduino257.php (дата звернення 08.02.25).

60. Схема підключення GM65 до Arduino Uno. URL: https://cxem.net/arduino/arduino257.php (дата звернення 08.02.25).

61. RFID-зчитувач. URL: https://www.forter.com.ua/chto-takoe-rfid-schityvatel-i-gde-primenyaetsya/ (дата звернення 08.02.25).

62. Блок-схема алгоритму обробки даних Arduino. URL: https://andriy1024.github.io/home-protector/menu/elementbase.html (дата звернення 08.02.25).

63. Діаграма потоків даних. URL: https://andriy1024.github.io/home-protector/menu/elementbase.html (дата звернення 08.02.25).

Додаток А

(обов'язковий)

ЛІСТИНГ КОДУ ПЗ

Лістинг коду ПЗ

```
import tkinter as tk
from tkinter import ttk, messagebox
import sqlite3
import serial
import threading
import time

# Налаштування COM-порту
PORT = 'COM3'  # змінити відповідно до порту Arduino
BAUD_RATE = 9600

# Ініціалізація бази даних
def init_db():
    conn = sqlite3.connect('inventory.db')
    c = conn.cursor()
    c.execute('''
        CREATE TABLE IF NOT EXISTS inventory (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            code TEXT UNIQUE,
            name TEXT DEFAULT 'Товар без назви',
            quantity INTEGER DEFAULT 1,
            last_scanned TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        )
    ''')
    conn.commit()
    conn.close()

# Завантаження даних у таблицю
def load_data():
    for row in tree.get_children():
        tree.delete(row)
    conn = sqlite3.connect('inventory.db')
    c = conn.cursor()
    c.execute("SELECT code, name, quantity, last_scanned FROM inventory")
    rows = c.fetchall()
    for r in rows:
        tree.insert('', 'end', values=r)
    conn.close()

# Обробка даних із COM-порту
def read_from_serial():
    try:
        ser = serial.Serial(PORT, BAUD_RATE, timeout=1)
        while True:
            line = ser.readline().decode().strip()
            if line:
                save_to_db(line)
                load_data()
    except serial.SerialException:
        print("COM-порт не знайдено або відключено")
    except Exception as e:
        print("Помилка:", e)

# GUI
def create_gui():
    global tree
    root = tk.Tk()
    root.title("Система управління інвентарем")
    root.geometry("700x400")
    tree = ttk.Treeview(root, columns=('code', 'name', 'quantity', 'last_scanned'), show='headings')
    tree.heading('code', text='Штрих-код')
    tree.heading('name', text='Назва')
    tree.heading('quantity', text='Кількість')
    tree.heading('last_scanned', text='Останнє сканування')
    tree.pack(fill='both', expand=True)
    load_data()
    return root

# Запуск
if __name__ == '__main__':
    init_db()
    root = create_gui()
    t = threading.Thread(target=read_from_serial)
```

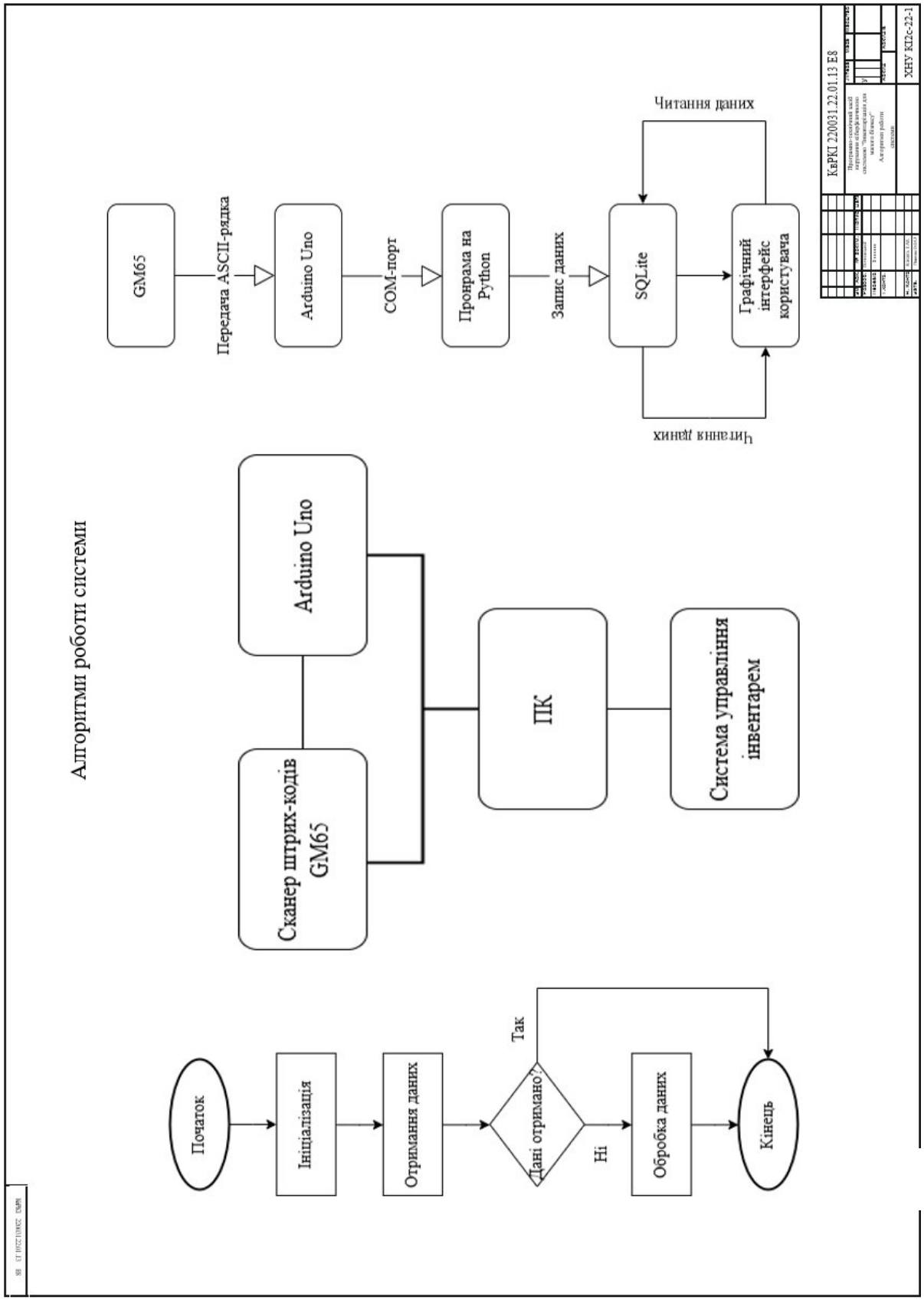
КвРКІ 220031.22.01.13 Е8

ХНУ КІ2с-22-1

Додаток Б

(обов’язковий)

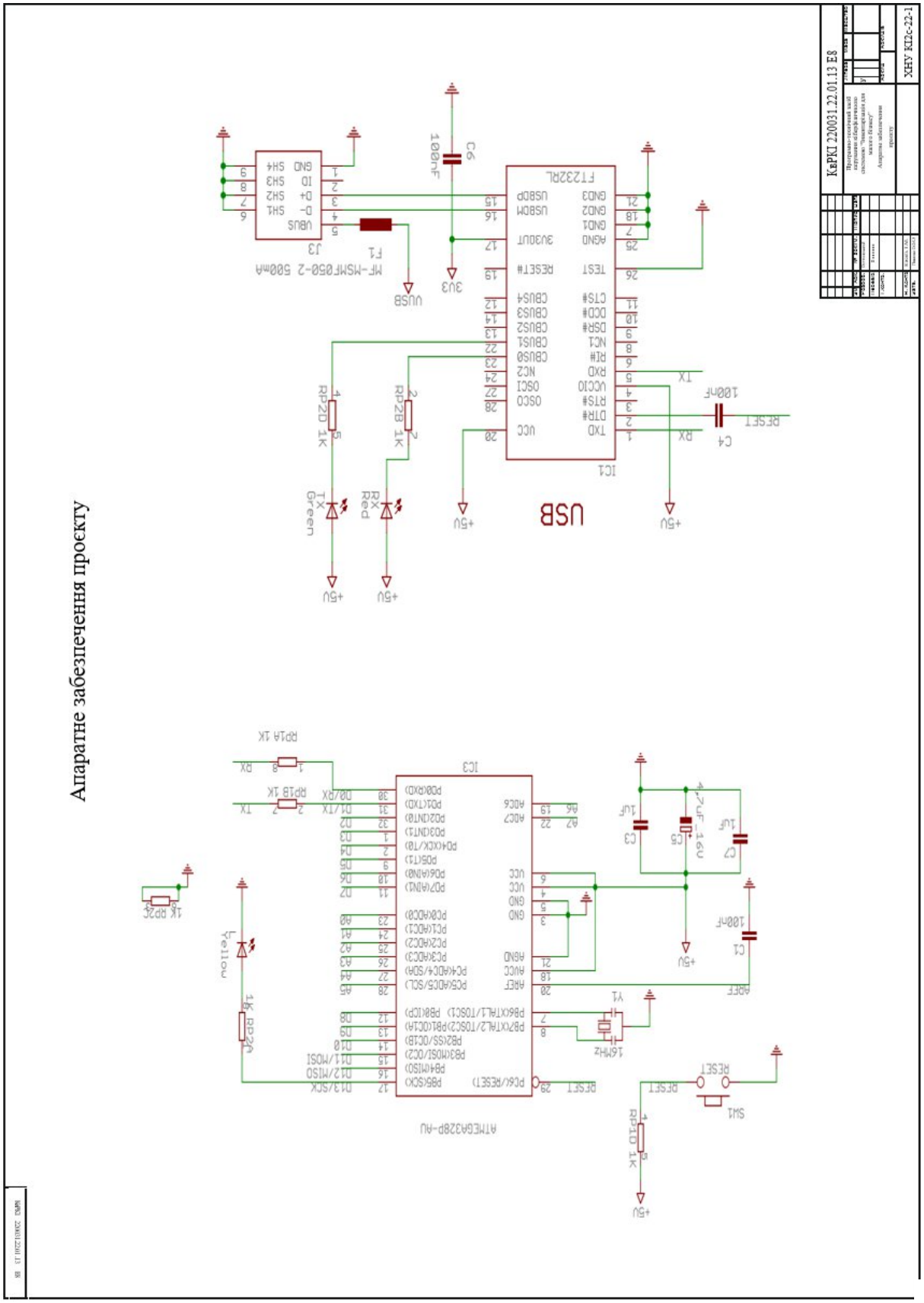
АЛГОРИТМИ РОБОТИ СИСТЕМИ



Додаток В

(обов’язковий)

АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ПРОЄКТУ



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Островський Віталій Валерійович

Тема: Програмно-технічний засіб керування кіберфізичноою системою "Інвентаризація для малого бізнесу"

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___62___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є розробка кіберфізичної системи управління інвентарем, адаптованої до умов малого бізнесу, яка об'єднує прості технічні засоби з програмним забезпеченням, що реалізує облік товарів на основі сканування штрих-кодів.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі кваліфікаційної роботи здійснено системний аналіз предметної області, що охоплює проблеми ручного обліку на малих підприємствах. Проаналізовано типові труднощі, які виникають через відсутність автоматизації, та обґрунтовано необхідність упровадження простих технічних рішень. Проведено порівняльний аналіз традиційних і сучасних підходів до управління інвентарем, що дозволило визначити напрям розробки — створення автономної локальної системи збору та обробки даних. Сформульовано загальні завдання проєкту з урахуванням обмежених ресурсів, що є актуальною проблемою в сучасному малому бізнесі. У другому розділі виконано технічне проєктування та реалізацію основних компонентів системи. Обґрунтовано вибір апаратних та програмних засобів: Arduino Uno для зчитування даних, сканера штрих-кодів GM65 для ідентифікації товарів, Python як базової мови розробки програмної частини, бібліотеки Tkinter для створення графічного інтерфейсу користувача, а також СУБД SQLite для локального збереження інформації. Описано

архітектуру системи, логіку її роботи, способи комунікації між пристроями, структуру інтерфейсу та сценарії взаємодії з користувачем. Усі рішення приймалися з урахуванням передових підходів до побудови бюджетних IoT-рішень і сучасних практик розробки користувацьких інтерфейсів. У третьому розділі виконано демонстрацію та верифікацію працездатності розробленої системи. Проведено тестування із використанням віртуальних COM-портів, що дозволило перевірити функціональність у симульованому середовищі без фізичних пристроїв. Зібрані результати подано у вигляді графіків і таблиць, які підтверджують надійність та стабільність роботи, швидкий час реакції системи, підтримку роботи в реальному часі, стійкість до помилок, а також масштабованість при збільшенні кількості операцій. Під час розробки використовувались сучасні інструменти аналізу й тестування, що відповідають актуальному рівню розвитку програмно-апаратних рішень.4. Позитивні сторони роботи: Робота відзначається високим рівнем практичної реалізації та використанням сучасних технологій. Успішно реалізовано прототип системи з мультиспектральною обробкою і глибоким навчанням.

5. Негативні сторони роботи: Недостатньо розкрито питання інформаційної безпеки та обґрунтування вибору апаратних і програмних компонентів серед альтернатив.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Ливвар Олег Сергійович, доц. каф. ТМІТ, ХНУ

_____

"26" 06 2025 р. _____(підпис)

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ

Направляється студент Островський Віталій Валерійович на захист дипломного проєкту (роботи)
(прізвище, ім'я, по батькові)

за спеціальністю 123 - Комп'ютерна інженерія

На тему: Програмно-технічний засіб керування кіберфізичноою системою "Інвентарізація для малого бізнесу"

Дипломний проєкт (робота), рецензія і довідка про перевірку на плагіат додаються.

Декан факультету _____ Тетяна ГОВОРУЩЕНКО
(ім'я, прізвище)

ДОВІДКА УСПІШНОСТІ

Островський В. В. за період навчання на факультеті інформаційних технологій з 2022 по 2025 роки повністю виконав навчальний план спеціальності з таким розподілом оцінок за:
національною шкалою: відмінно 0,00 %, добре 22,22 %, задовільно 77,78 %.
шкалою ЄКТС: A 2,38 %, B 9,52 %, C 4,76 %, D 23,81 %, E 59,52 %.

Методист факультету _____ _____
(підпис) (ім'я, прізвище)

ВИСНОВОК КЕРІВНИКА ДИПЛОМНОГО ПРОЄКТУ (РОБОТИ) ТА ОБГРУНТУВАННЯ ОЦІНКИ

Студент _____

_____

_____

_____

_____

_____

_____

Оцінка дипломного проєкту (роботи)_____

Керівник дипломного проєкту _____ _____
(підпис) (ім'я, прізвище)

"_____" _____ 2025 р.

ВИСНОВОК КАФЕДРИ ПРО ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ)

Дипломний проєкт (роботу) розглянуто. Студент Островський В. В. допускається до захисту цього проєкту (роботи) в екзаменаційній комісії.

Завідувач кафедри _____
(назва)

_____ "_____" _____ 2025 р.
(підпис, ім'я, прізвище)

Thu Jun 05 12:15:50 EEST 2025, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 0.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 8%**

	Document		Sum coincidence on the DB	
	Symbols	Lexemes	Symbols	Lexemes
ID: 243652 Title: БКР Програмно-технічний засіб керування кіберфізичноою системою "Інвентарізація для малого бізнесу" Added in a DB: 2025-06-05 Authors: Віталій ОСТРОВСЬКИЙ Heads: Євген ФЕДОРОВ Consultants: Opponents:	75535	538	2414 (3%)	39 (7%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Віталій ОСТРОВСЬКИЙ

Співавтор:

Назва: Островський_Програмно-технічний засіб керування кіберфізичноою системою "Інвентарізація для малого бізнесу"

Експерт:

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:3.4%

Коефіцієнт подібності 2:0%

Мікропробіли: 6

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-05 19:32:50.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2025-06-06 Доцент Андрій Нічепорук

Дата експерт

Завідувачу кафедри КІІС
д-р. філософії, доц. Ользі ПАВЛОВІЙ
Віталій ОСТРОВСЬКИЙ
ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи КІ2с-22-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Strike-Plagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

15.06. 2025 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Програмно-технічний засіб керування кіберфізичною системою "Інвентаризація для малого бізнесу"

Автор: Віталій ОСТРОВСЬКИЙ

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Євген ФЕДОРОВ д.т.н.

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) Запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи.;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) Окремі збіги представлені загальновживаними фразами, наприклад: «на рисунку зображено», «загальна структура системи», «висновки до розділу» тощо.
4) Якість запозичень відповідає технічним особливостям дослідження: виявлено збіги в кодах, формулах і термінах, які є вихідними даними до великої кількості задач і не можуть вважатися авторськими порушеннями.
5) Система зафіксувала технічні модифікації тексту, зокрема: заміну окремих символів ,скорочення індексів у формулах, зміну розміщення символів. Це є наслідком форматування або експорту документа, а не цілеспрямованого уникнення перевірки.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 3.4% і адресується до 27 першоджерела; та системою Anti-Plagiarism складає 0.29%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Є ФЕДОРОВ

Гарант ОП _____ Андрій НІЧЕПОРУК

Завідувач кафедри КІІС _____ Ольга ПАВЛОВА