НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

на тему: Апаратно-програмний комплекс для обліку та контролю збору врожаю

Виконав : студент __4__ курсу, групи __ІО-02__
(шифр групи)

Литвиненко Данило Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник професор, д.т.н., доц. Клименко І.А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) ст. викладач Виноградов Ю.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент доцент, к.т.н., доц. Клятченко Я.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2024 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Литвиненка Данила Олександровича

1. Тема проєкту *Апаратно-програмний комплекс для обліку та контролю збору врожаю*
керівник проєкту Клименко Ірина Анатоліївна, доцент, д.т.н.,проф. кафедри ОТ ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 27 травня 2024 року № 2112-с
2. Термін здачі студентом закінченого проєкту *3 червня 2024 р.*
3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)
Огляд існуючих рішень.
Аналіз технологій для реалізації системи для обліку та контролю збору врожаю.
Розроблення програмного-апаратного комплексу для обліку та контролю збору врожаю.
Реалізація дослідного зразку пристрою для збирання даних про врожай.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) схема структурна, діаграма даних (схема функціональна), алгоритм роботи програми (схема принципова).
6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Виноградов Ю.М.		

7. Дата видачі завдання «01» грудня 2023 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Вибір теми проєкту*	*01.12.2023-15.12.2023*	
2.	*Вивчення та аналіз літературних джерел*	*16.12.2023-25.03.2024*	
3.	*Розробка архітектури програмного забезпечення*	*26.03.2024-30.04.2024*	
4.	*Розробка апаратного забезпечення*	*01.04.2024-06.05.2024*	
5.	*Програмна реалізація користувацького інтерфейсу*	*07.05.2024-14.05.2024*	
6.	*Розробка прототипу системи та експерименти*	*15.05.2024-22.05.2024*	
7.	*Оформлення пояснювальної записки*	*23.05.2024-01.06.2024*	
8.	*Проходження нормоконтролю та перевірка на плагіат*	*08.06.2024*	
9.	*Захист*	*17.06.2024*	

Студент-дипломник _____ Данило ЛИТВИНЕНКО
(підпис)

Керівник проєкту _____ Ірина КЛИМЕНКО
(підпис)

АНОТАЦІЯ

Дана дипломна робота присвячена розробці апаратно-програмного комплексу для обліку та контролю збору врожаю. У роботі вирішено проблему автоматизації процесу збору даних про врожайність сільськогосподарських культур та їх подальшого обліку. У роботі проведений аналіз існуючих систем обліку врожаю та їх недоліків. На основі отриманих даних розроблено апаратну частину системи, що включає в себе датчики для збору даних, мікроконтролер для їх обробки та передачі на сервер. Також розроблено програмне забезпечення для візуалізації та аналізу даних на сервері. Програма має зручний інтерфейс користувача, який дозволяє переглядати дані, а також зберігати і аналізувати їх. Отримані результати показують ефективність розробленого комплексу у порівнянні з традиційними методами обліку врожаю, зокрема, зменшення часу на збір даних, підвищення точності та зручність управління процесом. Програмний продукт був створених на мовах Go, C++, TypeScript та з використанням багатьох зовнішніх бібліотек.

ANNOTATION

This thesis is devoted to the development of a hardware and software complex for accounting and control of harvesting. The work solves the problem of automating the process of collecting data on crop yields and their subsequent accounting. The paper analyzes existing crop accounting systems and their shortcomings. Based on the data obtained, the hardware part of the system was developed, which includes sensors for data collection, a microcontroller for processing and transferring them to the server. Software for visualizing and analyzing data on the server was also developed. The program has a user-friendly interface that allows you to view data, as well as store and analyze it. The results obtained show the effectiveness of the developed complex compared to traditional methods of crop accounting reducing the time for data collection, increasing the accuracy and convenience of process management. The software product was created in Go, C++, TypeScript and using many external libraries.

№ рядка	Формат	Позначення	Найменування	Кількість	Примітка
1			Документація загальна		
2			Знову розроблена		
3					
4	А4	ІАЛЦ.467200.001 ОА	Апаратно-програмний комплекс для	1	
5			обліку та контролю збору врожаю		
6			Опис альбому		
7					
8	А4	ІАЛЦ.467200.002 ТЗ	Апаратно-програмний комплекс для	3	
9			обліку та контролю збору врожаю		
10			Технічне завдання		
11					
12	А4	ІАЛЦ.467200.003 ПЗ	Апаратно-програмний комплекс для	103	
13			обліку та контролю збору врожаю		
14			Пояснювальна записка		
15					
16	А3	ІАЛЦ.467200.004 Е1	Апаратно-програмний комплекс для	1	
17			обліку та контролю збору врожаю		
18			Схема структурна		
19					
20					
21	А4	ІАЛЦ.467200.005 Е2	Апаратно-програмний комплекс для	1	
22			обліку та контролю збору врожаю		
23			Діаграма даних		
24			Схема функціональна		
25					
26	А3	ІАЛЦ.467200.006 Е3	Апаратно-програмний комплекс для	1	
27			обліку та контролю збору врожаю		
28			Алгоритм роботи програми		
29			Схема принципова		
30					
31	А4	ІАЛЦ.467200.007 Е4	Апаратно-програмний комплекс для	12	
32			обліку та контролю збору врожаю		
33			Текст програмного коду		
34					
35					

Змн.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.001 ОА		
Розроб.		Литвиненко Д.О.			Апаратно-програмний комплекс для обліку та контролю збору врожаю Опис альбому	Літ.	Арк. 1 / Аркушів 1
Перевір.		Клименко І.А.					
Н. Контр.		Виноградов Ю.М.				КПІ ім. Ігоря Сікорського, ФІОТ, Група ІО-02	
Затверд.							

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Апаратно-програмний комплекс для обліку та контролю збору врожаю*»

Київ – 2024

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

Це технічне завдання поширюється на розробку апаратно-програмного комплекс для обліку та контролю збору врожаю. Область застосування: альтернатива існуючим системам такого типу.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», який був затверджений факультетом «Інформатики та обчислювальної техніки» кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут імені Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням даної роботи є розробка апаратно-програмного комплексу для обліку та контролю збору врожаю для фермерських господарств.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є науково-технічна література, публікації в періодичних виданнях, офіційна документація, публікації в Інтернеті на дану тему.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

- Система може автономно працювати протягом робочого дня (12 годин).
- Пристрій для збору даних повинен мати достатньо малі габарити, щоб його могла перенести одна людина.
- Дані повинні зберігатися на пристрої навіть при поломці контролера або у вимкненому стані.
- Пристрій може витримати падіння з висоти 1 метра.
- Пристрій достатньо герметичний від потрапляння бруду на плату.
- Несанкціонований доступ до даних про прибутки повинен бути неможливим, або не релевантним для зловмисника.

5.2. Вимоги до програмного забезпечення

- Операційна система Windows, Mac, Linux.
- Nginx версії 1.10.3 і новіші

5.3. Вимоги до апаратної частини

- Комп’ютер на базі процесора x64
- Оперативної пам’яті не менше 2 ГБ
- Вільний простір жорсткого диска не менше 4 ГБ
- Підключення до Інтернету

6. ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Вибір теми проєкту	01.12.2023-15.12.2023
Вивчення та аналіз літературних джерел	16.12.2023-25.03.2024
Розробка архітектури програмного забезпечення	26.03.2024-30.04.2024
Розробка апаратного забезпечення	01.04.2024-06.05.2024
Програмна реалізація користувацького інтерфейсу	07.05.2024-14.05.2024
Розробка прототипу системи та експерименти	15.05.2024-22.05.2024
Оформлення пояснювальної записки	23.05.2024-01.06.2024

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Апаратно-програмний комплекс для обліку та контролю збору врожаю*»

Київ – 2024

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АПК	Апаратно-програмний комплекс
RTC	(Real-Time Clock) Годинник реального часу
SDA	(Serial DAta) Лінія даних
SCL	(Serial CLock) Лінія тактового сигналу
LCD	(Liquid Crystal Display) Рідкокристалічний дисплей
BLE	(Bluetooth Low Energy) Bluetooth з низьким енергоспоживанням
UART	(Universal Asynchronous Receiver/Transmitter) Універсальний асинхронний приймач/передавач
SPI	(Serial Peripheral Interface) Послідовний периферійний інтерфейс
SD	(Secure Digital Memory Card) Захищена цифрова карта пам’яті
MOSI	(Master Output Slave Input) Вихід ведучого, вхід веденого
MISO	(Master Input Slave Output) Вхід ведучого, вихід веденого
AP	(Access Point) Точка доступу
API	(Application Programming Interface) Прикладний програмний інтерфейс
TCP	(Transmission Control Protocol) Протокол управління передачею
FTP	(File Transfer Protocol) Протокол передавання файлів
SPIFFS	(Serial Peripheral Interface Flash File System) Файлова система флеш-пам'яті, що підключається за послідовним периферійним інтерфейсом
IoT	(Internet of Things) Інтернет речей

ВСТУП

Кожен день ми стикаємося з проблемами різного масштабу і сільське господарство є основою не лише нашого життя, а й економічного розвитку країни в цілому. Воно є джерелом харчової безпеки та сприяє соціальній стабільності. Однак, для досягнення максимальних результатів та вирішення актуальних проблем сільськогосподарського сектору, потрібно постійно вдосконалювати методи обліку та контролю за збором врожаю.

У контексті цього, дана дипломна робота присвячена розробці апаратно-програмного комплексу, який спрямований на удосконалення процесу обліку та контролю збору врожаю. Цей комплекс поєднує в собі технології апаратної та програмної інженерії для забезпечення точного та ефективного збору даних про врожайність сільськогосподарських культур.

Впровадження такого комплексу може мати значний вплив на покращення ефективності господарювання та збільшення прибутковості сільського господарства, що в свою чергу позитивно позначиться на розвитку аграрного сектору в цілому.

Впровадження апаратно-програмного комплексу для обліку та контролю збору врожаю може стати важливим кроком у подоланні викликів майбутнього. За допомогою цього комплексу фермери матимуть можливість отримувати точні дані про врожайність та ефективно використовувати ресурси, що дозволить їм оптимізувати виробництво та забезпечити стабільний дохід. Шляхом збору та аналізу даних про врожаї, виробники зможуть забезпечити доступ до інформації про якість та кількість продукції, що сприятиме підвищенню довіри споживачів до сільськогосподарських товарів.

Отже, апаратно-програмний комплекс для обліку та контролю збору врожаю є важливим інструментом для модернізації сільського господарства. Він не лише допоможе фермерам підвищити продуктивність та оптимізувати виробництво, але й сприятиме створенню стійкої та конкурентоспроможної аграрної галузі, яка відповідає сучасним вимогам та викликам.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Актуальність та доцільність розробки апаратно-програмного комплексу для обліку та контролю збору врожаю

Сільське господарство є невід'ємною частиною суспільства, оскільки забезпечує харчову безпеку, сприяє економічному розвитку, створює соціальну стабільність, впливає на збереження навколишнього середовища та є полем для інновацій та досліджень.

Для забезпечення однієї з головних потреб людства, важливо підтримувати та розвивати цей сектор, бо галузь стикається з багатьма викликами, такими як зростання населення, зміна клімату та дефіцит ресурсів. Впровадження інформаційних технологій та нових видів обчислювальної техніки може стати ключовим фактором подолання цих проблем та забезпечення стійкого та ефективного розвитку агробізнесу.

Останнім часом спостерігається стрімке зростання цін на овочі та фрукти. Зростання витрат на транспорт, на енергоносії, на добрива та засоби захисту рослин. Незважаючи на це, ціна на зернові культури залишається відносно стабільною, що змушує фермерів переходити на городину, яка є рентабельнішою у перспективі.

Вирощування овочів та фруктів потребує значної кількості ручної праці. Потрібно швидко та якісно обліковувати товар, для унеможливлення довгого простою у роботі, що призводить до зайвих витрат і погіршення якості товару.

Багато фермерів шукають способи покращити ефективність та прибутковість своїх господарств. Тому зараз існує зростаючий попит на інноваційні рішення для сільського господарства, одними з яких є системи обліку врожаю.

Система обліку врожаю може забезпечити доступ до даних про запаси та рух продукції всім зацікавленим сторонам, що поліпшить комунікацію та

координацію в ланцюжку постачання. Це особливо корисно для кооперативів та великих господарств.

Але не потрібно забувати і за малий агробізнес. Його часка завжди була важливою складовою світового харчового ринку. Для невеликих фермерських господарств розробка власної автоматизованої системи обліку є не вигідною через високі капіталовкладення та потребу у навчанні персоналу.

Апаратно-програмний комплекс для обліку та контролю збору врожаю націлений саме на малих фермерів, щоб допомогти їм за відносно малі кошти залишатися перспективним бізнесом. Замість витрат на окремі рішення для кожного етапу виробництва, апаратно-програмний комплекс для обліку та контролю збору врожаю може об’єднати різні функції. Це дозволить зменшити витрати на впровадження та обслуговування системи, а також спростить процес управління господарством.

Крім того, апаратно-програмний комплекс для обліку та контролю збору врожаю може забезпечити доступ до важливих даних про виробництво, які можуть бути використані для прийняття стратегічних рішень. Наприклад, аналіз даних про врожайність різних культур може допомогти фермерам оптимізувати розподіл землі та вибрати найбільш прибуткові культури для вирощування. Також дані про запаси та рух продукції можуть бути використані для планування поставок та виробництва, що дозволить уникнути надлишкових запасів і зменшити витрати.

Загалом, апаратно-програмний комплекс для обліку та контролю збору врожаю може стати потужним інструментом для оптимізації виробництва на малих фермерських господарствах, допомагаючи зменшити витрати, підвищити продуктивність і зробити бізнес більш конкурентоспроможним на ринку.

1.2 Огляд та приклади відомих систем обліку товарів

У даному розділі зосередимося на аналізі та порівнянні деяких відомих систем автоматизації збору врожаю в Україні та світі, що працюють у сфері торгівлі та сільського господарства.

Кожна з цих систем має свої особливості, переваги та недоліки, які були враховані при проектуванні дипломного проекту. Розглянуто функціональність та можливості систем, але повну інформацію про будову пристроїв, фреймворки і код не можливо отримати через комерційну таємницю.

1.2.1. МініСофт Урожай (Україна)

Апаратно-програмний комплекс на базі програмного забезпечення “Міні-Софт Урожай” призначений для обліку та управління врожаєм ягід, фруктів і овочів, розрахунку та виплати заробітної плати працівникам, зайнятим на збиранні та обліку зібраної продукції. [1]

Крім рослинництва МініСофт Урожай може застосовуватися і в інших галузях агропромислового комплексу. Наприклад, у виробництві молока, меду та лікарських трав, де необхідно вести облік надходження продукції для обліку продуктивності кожного працівника. [1]

Система доступна в серверній і локальній версіях і може бути встановлена на комп’ютер без синхронізації з сервером. Крім основних функцій, як-от управління базами даних товарів, співробітників, контейнерів тощо, система містить такі функції, як створення унікальних штрих-кодів за допомогою спеціального принтера, розрахунок заробітної плати співробітників, управління врожаєм і синхронізація баз даних між кількома станціями (комп’ютерами).

Вартість використання цього програмного забезпечення залежить від необхідної версії і починається від 3000 гривень за серверну версію платформи. Загальна вартість програмного забезпечення, включно з необхідним обладнанням (програмне забезпечення, ваги, сканер штрих-кодів і

принтер етикеток), становить 22 000 гривень, з урахуванням знижок по 500 гривень. Крім того, за 500 гривень можна замовити телеграм-бота для системи. [1, 2]

Апаратна частина складається з принтера чеків, принтера етикеток, сканера штрих-кодів, вагів, термінала збору даних, РРО, картридера.[1, 3] Усі вони не є розробкою ТОВ “МініСофт”, а лише налаштовуються ПЗ даної компанії. На рис. 1.1 можна побачити приклад вікна програми.

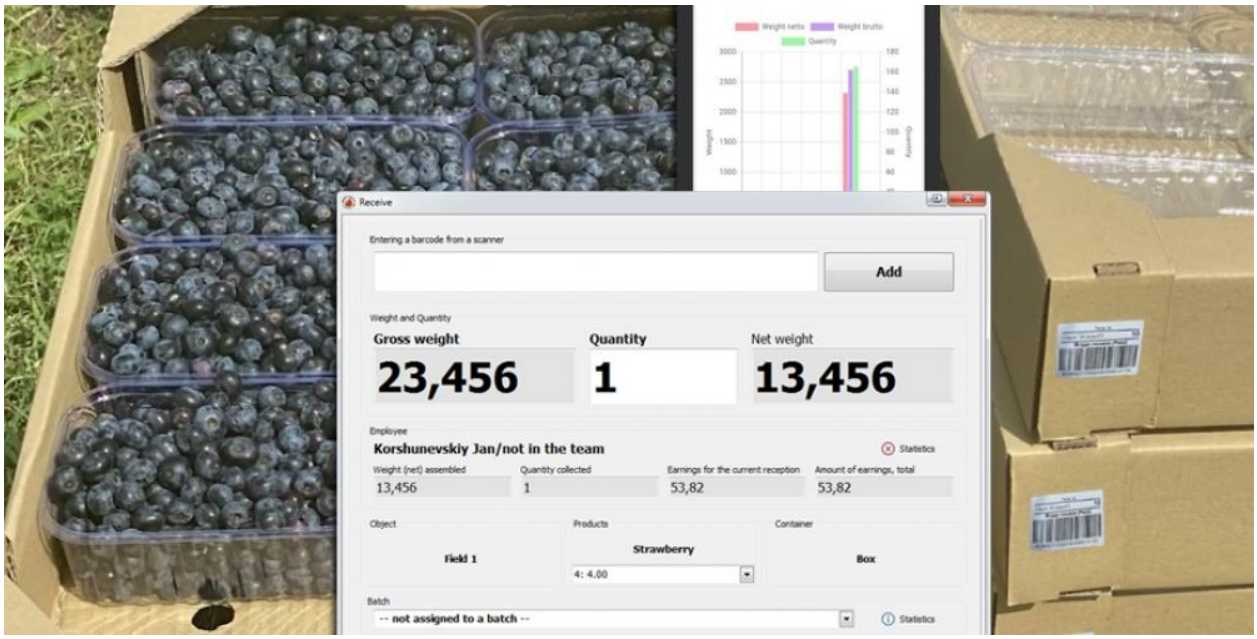


Рисунок 1.1. – Вікно програми і приклад робочої зони [4]

Із плюсів можна відмітити:

- створення, публікацію, редагування товарів
- контроль зарплати і виробітку працівників
- інструменти візуалізації даних
- ролі та рівні доступу
- фільтри та пошук
- подання аналітики
- підтримка української мови
- присутня ліценція

Щодо мінусів:

– документація не є повною
– зворотній зв’язок є платним
– інтеграція з російським програмним забезпеченням

1.2.2. AgSquared (США)

AgSquared пропонує персоналізовані налаштування та постійну підтримку, щоб допомогти орієнтуватися в постійно мінливих вимогах фермерського бізнесу. Якщо ваші пріоритети змінюються і ви потребуєте нових рішень, AgSquared готові підтримати вас на кожному кроці. [5]

Кожне фермерське господарство відрізняється від інших і має унікальні пріоритети та плани розвитку. AgSquared створили як висококонфігуровану платформу, яка не є універсальною. Але AgSquared має достатньо різноманітний функціонал, який представлений на рис. 1.2.

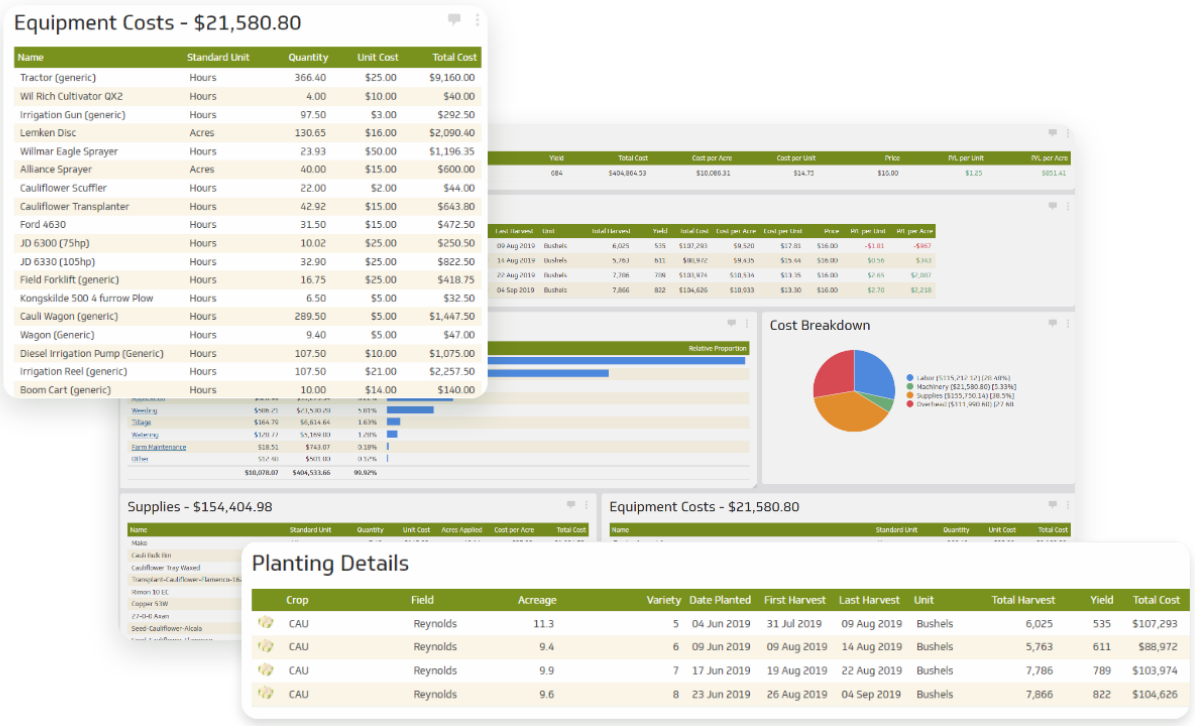


Рисунок 1.2. – Приклад вікон AgSquared [6]

У рамках партнерства NATIVE та AgSquared можуть використовуватися разом для передачі даних з поля в ланцюжок поставок. Коли фермери керують своїми полями, вони можуть використовувати AgSquared для реєстрації оперативних даних про посіви, обприскування та обробіток ґрунту прямо на своїх телефонах. Звідти NATIVE дозволяє покупцям простежити шлях від насіння до продажу, щоб фермери могли знайти покупців, готових платити більшу ціну за врожай з доданою вартістю, ніж у випадку з даними. [7]

Загалом, AgSquared може допомогти фермерам керувати своїми операціями, покращити свою продуктивність та підвищити свою рентабельність. Однак воно може бути дорогим та складним у налаштуванні, і воно може не інтегруватися з іншим програмним забезпеченням, яке використовують фермери України.

1.2.3. Aegro (Бразилія)

Aegro – це платформа для управління сільським господарством, яка дозволяє здійснювати повне планування врожаю, від бюджетів витрат на посадку до цілей продуктивності бізнесу. [8] Функції цієї платформи також включають контроль над шкідниками та хворобами, супутникові знімки майна для дистанційної перевірки вирощування, одночасне використання платформи кількома користувачами та касову книгу для упорядкованого та актуального відображення фіскальних зобов’язань фермерського господарства.

Також, Aegro підключається до Climate FieldVie, John Deere Operations Center та Stara для автоматичної передачі агрономічних даних, що забезпечує повне розуміння операцій. Одне з вікон програми ви можете побачити на рис. 1.3.

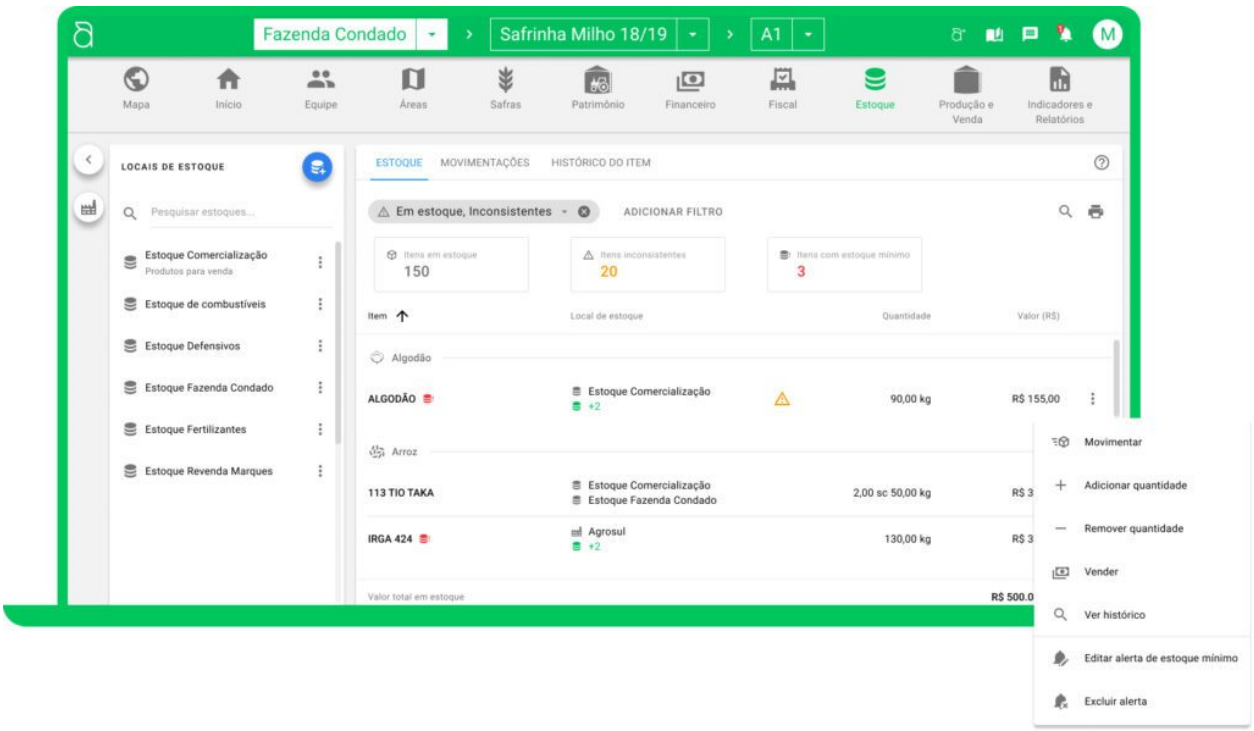


Рисунок 1.3. – Приклад вікна Aegro [9]

Показником перспективності проекту є те, що бразильська платформа для управління фермерськими господарствами Aegro залучила 12 млн реалів ($2,22 млн) в рамках раунду фінансування під керівництвом існуючих інвесторів SP Ventures та ABSeed. [10]

1.2.4. POS Vector (Україна)

POS Vector [11] – це компанія, яка пропонує рішення з автоматизації для роздрібних торгових підприємств. Вони пропонують програмне та апаратне забезпечення, спеціально розроблене для різних потреб роздрібної торгівлі. Дана компанія не націлена на фермерські господарства, але пропонує схожий функціонал, що й інші розглянуті компанії.

По-перше, POS Vector має попередньо налаштовані набори, щоб допомогти підприємствам швидко розпочати роботу.

По-друге, включає широкий спектр програмного забезпечення для роздрібної торгівлі, включаючи POS-системи, програмне забезпечення для управління запасами та програмне забезпечення для лояльності клієнтів.

По-третє, пропонує широкий спектр апаратного забезпечення для роздрібної торгівлі, включаючи сенсорні екрани, сканери штрих-кодів та принтери розписок.

По-четверте, надає технічну підтримку та обслуговування клієнтів, щоб допомогти підприємствам отримати максимальну віддачу від своїх інвестицій.

1.3. Проблематика розроблення системи для обліку та контролю збору врожаю

1.3.1. Переваги систем для обліку та контролю збору врожаю

Система автоматизує багато завдань, пов’язаних зі збором врожаю, що призводить до значної економії часу. АПК (апаратно-програмний комплекс) використовується для збору даних про врожайність у режимі реального часу. Ці дані можуть використовуватися для оптимізації маршрутів збирання та підвищення загальної ефективності.

Також, система забезпечує більш точні дані про врожайність, ніж традиційні методи збору даних. Це допомагає приймати кращі рішення щодо управління полями.

Програма допомагає фермерам покращити якість свого врожаю, відстежуючи умови навколишнього середовища та інші фактори, які можуть впливати на ріст рослин.

Всі ці системи допомагають у виконанні вимог GlobalG.A.P., які пропонують набір стандартів для безпечних та відповідальних виробничих процесів у сільському господарстві, аквакультурі та квітникарстві, доповнюючи їх пакетом послуг для ефективного впровадження. [12]

1.3.2. Недоліки систем для обліку та контролю збору врожаю

Система може бути дорогою у придбанні та складною в експлуатації, що може бути перешкодою для деяких фермерів, особливо для дрібних.

АПК може збирати дані, які є чутливими для фермерів, тому важливо переконатися, що ці дані захищені та використовуються лише за призначенням.

Впровадження АПК може потребувати внесення змін до існуючих процесів та методів роботи, що може бути складним і вимагати часу.

1.3.3. Функціонал характерний системам для обліку товарів

Для фермерських господарств, які вирощують овочі:

- Реєстрація площі землі, виділеної під вирощування овочів, планування сівозміни та вибір сортів овочів для вирощування.
- Облік та реєстрація введення саджанців або насіння для вирощування овочів.
- Моніторинг процесу вирощування овочів, включаючи полив, добрива, захист від шкідників та хвороб.
- Реєстрація та контроль збору врожаю овочів з поля.
- Контроль за кількістю та якістю зібраного врожаю на складі.
- Організація та облік транспортування овочів до ринків або пунктів продажу.
- Моніторинг втрат під час збору, транспортування та зберігання овочів, включаючи переробку відходів.
- Якщо на фермі здійснюється переробка, то важливо вести облік процесу виробництва консервів або заморожених овочів.

Для фермерських господарств, які вирощують худобу на забій:

- Фермерське господарство повинно мати систему для реєстрації та обліку прибуття худоби на м’ясокомбінат. Кожна тварина має бути маркована з унікальним номером (номер вушка) для подальшого відстеження.
- Кожен забій тварини та вилучення субпродуктів, таких як органи, кістки, шкіра тощо, має бути детально зареєстрованим для подальшого використання та виробництва.

- Контроль тварин, які не призначені для забою, для їх продажу живою вагою.
- Процес обвалювання туш має бути контрольованим та оптимізованим для максимального виходу якісного м’яса та субпродуктів.
- М’ясо може бути перероблене в різноманітні продукти, такі як ковбаси та делікатеси. Цей процес також має бути врахований в системі виробництва та обліку.
- Шкіра тварин також може мати комерційну цінність і повинна бути врахована в системі обліку для подальшого використання або продажу.

Для фермерських господарств, які вирощують велику рогату худобу:

- Реєстрація та приймання сирої молока від фермерами або інших постачальників.
- Розділення молока на складові частини, такі як сировина з певною жирністю.
- Контроль за використанням матеріалів, таких як наповнювачі, стабілізатори, закваски, необхідних для виробництва молочної продукції.
- Процес переробки сирої сировини в готову молочну продукцію, таку як молоко, йогурт, сир тощо.
- Контроль за використанням та облік пакувальних матеріалів для упаковки готової продукції.

Постановка завдань до розробки

Для таких систем характерна велика кількість різноманітних технологій, які доповнюють одна одну. Загалом, нема точної статистичної інформації про технології, які використовуються при проектуванні таких систем, тому підприємства мають різні технологічно, але подібні функціонально системи. Тому проблематику можна розглядати з боку функціоналу готового продукту, а не складності реалізації технічних завдання.

Важливо, щоб система була достатньо точною та надійною у вимірюванні обсягу та якості зібраного врожаю. Помилки у вимірюваннях можуть призвести до недооцінки або переоцінки врожаю.

Апаратно-програмний комплекс повинен бути здатний полегшувати збір даних про врожай, уникати необхідності ручного пропису всіх складових та обробки інформації.

Для оптимальної ефективності, комплекс повинен мати принаймні технічну спроможність інтегруватися з існуючими системами управління господарством, такими як системи моніторингу клімату, системи зберігання та транспортування врожаю тощо.

Оскільки зібрані дані можуть містити конфіденційну інформацію про врожай та господарство, важливо, щоб апаратно-програмний комплекс був забезпечений відповідними заходами захисту даних для запобігання несанкціонованому доступу та витоку інформації.

Щоб забезпечити широке прийняття та використання системи, важливо, щоб вона була зручною у використанні та мала інтуїтивно зрозумілий інтерфейс для користувачів.

Розробка та впровадження комплексу повинні бути економічно доцільними для фермерських господарств та агробізнесу загалом, забезпечуючи позитивний ефект від інвестицій у вигляді збільшення продуктивності та ефективності господарювання.

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі цієї дипломної роботи розглянуто популярні системи для обліку та контролю збору врожаю та аналоги, їх призначення, складові, та принципи функціонування. Досліджено проблематику створення таких систем, головні передумови та причини застосування систем для обліку та контролю збору врожаю.

Згідно з отриманими результатами, можна зазначити, що системи для обліку та контролю мають як позитивні, так і негативні аспекти. Тому важливо, щоб система була стійкою та гнучкою, а функціонал зрозумілим для адміністратора і збирача. Потрібно, щоб система була незалежною від обсягу даних та забезпечувала швидку їх обробку. Технологічні деталі розробки програмного забезпечення для цієї системи та пристрою для збору даних будуть детально описані у наступному розділі дипломної роботи.

РОЗДІЛ 2
АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ ДЛЯ ОБЛІКУ ТА КОНТРОЛЮ ЗБОРУ ВРОЖАЮ

2.1. Комплектуючі пристрою для збору даних збору врожаю

2.1.1. Датчик ваги (тензодатчик)

Тензодатчик – це пристрій, призначений для вимірювання деформації. [13] Зовні тензодатчики для вимірювання невеликої маси виглядають як металеві пластини або бруски, які кріпляться до корпусу ваг.

Основний принцип роботи тензодатчика полягає в тому, що він складається з тензочутливого елементу, який зазвичай складається з декількох витків провідників, що розташовані на тонкому пластинчастому датчику. На рис. 2.1 зображений тензодатчик для вимірювання не великих мас.

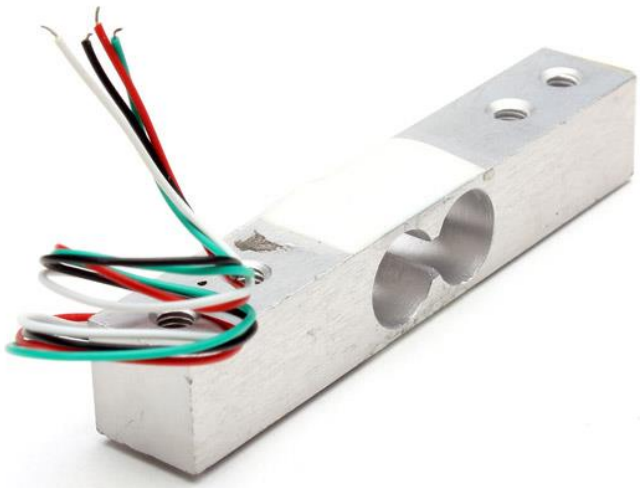


Рисунок 2.1. – Тензодатчик на 20 кг [13]

Під дією зовнішньої сили (наприклад, стискання або розтягнення), форма датчика змінюється, що призводить до зміни опору провідників. [14] Ця

зміна опору вимірюється електронними пристроями, такими як, наприклад, HX711, і конвертується в певне числове значення.

Тензодатчики мають високу точність вимірювань і широкий діапазон застосувань. Вони використовуються у різних галузях, і також в сільському господарстві. Тензодатчик є одним з ключових компонентів даного проекту.

До переваг можна віднести простоту інтеграції та конструкції, невисоку ціну відносно ціни проекту, а також надійність навіть після тисячі зважувань. Щодо мінусів, то для різної маси та точності вимірювання потрібно обирати відповідний датчик, тому чим більша дозволена вага, тим менша точність.

2.1.2. Двоканальний модуль датчиків ваги HX711

Модуль HX711 є популярним інтерфейсним засобом для підключення тензодатчиків до мікроконтролерів або одноплатних комп'ютерів. Цей модуль дозволяє зчитувати дані від двох тензодатчиків одночасно. На рис. 2.2 зображений даний модуль.

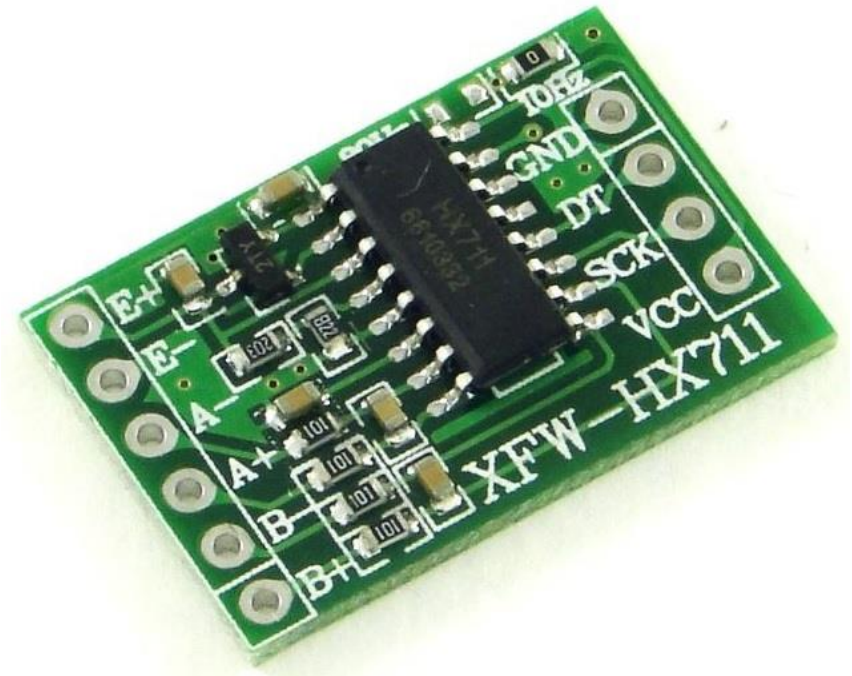


Рисунок 2.2. – Модуль HX711 [15]

Точність перетворення дорівнює 24 біт [16], що повністю задовольняє проект з вагами у 50 кг з точністю 5 г. Хоча 24-бітна точність може здатися

високою, вона не завжди означає, що датчик є високоточним у реальному світі. Інші фактори, такі як шум, помилки вимірювання та неточності в самому датчику, можуть вплинути на реальну точність системи.

Стандартна постійна напруга у 5 В, але може бути і від 2.6 до 5.5 В, і робочій струм менше 1.5 мА [16] дають можливість використовувати модуль у автономних системах, так як він може працювати годинами і від звичайної батарейки. Модуль має вбудований підсилювач, який дозволяє збільшити вихідний сигнал тензодатчика. Дані з датчика можуть оновлюватися 80 разів за секунду і це є досить високим показником для більшості застосувань.

HX711 може використовуватися з різними тензодатчиками, наприклад, з датчиками ваги на 1 кг, 10 кг, 50 кг та іншими. Максимальна вага, яку може виміряти HX711 обмежена тільки самим тензодатчиком.

Модуль має компактний розмір 25x15x3 мм, що полегшує його використання в різних проектах. Він також може бути легко підключений до інших електронних компонентів, що робить його універсальним інструментом для розробки різноманітних систем вимірювання ваги.

Загалом, HX711 – дешевий прилад китайського виробництва, який цілком задовольняє вимоги проєкта.

2.1.3. Матрична клавіатура

Матрична клавіатура – це тип клавіатури, який використовує матричну схему для виявлення натискання клавіш. Це один з найпоширеніших типів клавіатур, який зазвичай використовується в більшості видів техніки.

Основний принцип роботи матричної клавіатури полягає у тому, що клавіші розташовані у вигляді матриці рядків і стовпців. Кожна клавіша має свою унікальну адресу, яка визначається перетином рядка і стовпця. Коли користувач натискає на клавішу, мікроконтролер зчитує адресу натиснутої клавіші та ініціює відповідну дію, наприклад, введення символу на екран. [17]

На рис. 2.3 зображений зворотній бік матричної клавіатури, на якому можна побачити струмопровідні доріжки і припаяні кнопки.

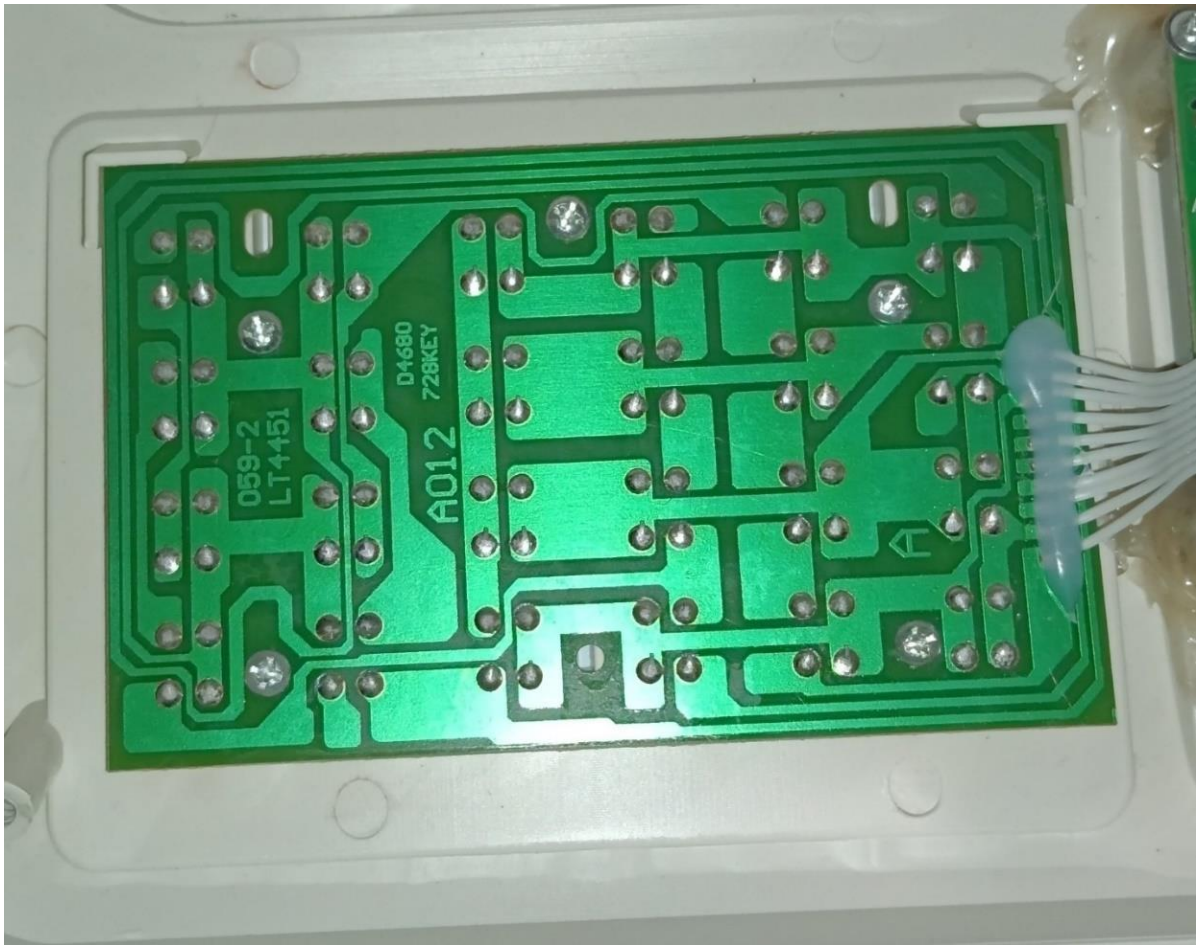


Рисунок 2.3. – Зворотній бік матричної клавіатури, яка знаходилась у вагах

У матричних клавіатур зазвичай використовуються два типи матеріалів для датчиків: мембранні та механічні. Мембранні клавіатури мають тонкі гнучкі плівки, які замиканням контактів утворюють електричне з’єднання, визначаючи натискання клавіші. У механічних клавіатурах кожна клавіша має свій власний механічний перемикач, який здійснює контакт при натисканні.

Однією з переваг матричних клавіатур є їх низька вартість виробництва та довгий термін служби. Вони також можуть мати компактний дизайн, що дозволяє їх використовувати в пристроях з обмеженим простором. Однак матричні клавіатури мають свої обмеження. Вони можуть бути вразливими до вироблення помилок при одночасному натисканні декількох клавіш. Також

вони не забезпечують такого ж рівня комфорту при натисканні клавіш, як механічні клавіатури.

У підсумку, матричні клавіатури є популярними та ефективними рішеннями для багатьох сучасних пристроїв, забезпечуючи зручне та надійне введення даних для користувачів.

2.1.4. RTC модуль DS1302

DS1302 – це модуль реального часу (RTC), який забезпечує точний показ часу для мікроконтролерних систем. [18] Цей модуль відносно простий у використанні, надійний і ефективний, що робить його популярним серед аматорів і професіоналів у сфері електроніки. На рис. 2.4 зображений даний модуль.

Рисунок 2.4. – RTC Модуль DS1302 [19]

У основі DS1302 лежить кварцовий генератор, стабілізувальним елементом частоти якого є кварцовий резонатор, який генерує стабільний сигнал тактової частоти. Цей сигнал використовується для вимірювання часу в одиницях секунд. Секундомір відстежує час, використовуючи тактовий сигнал від кристалу. Кожен раз, коли секундомір зараховує секунду, він збільшує значення регістру секунди.

Модуль побудовано на базі мікросхеми з 31 байтом статичної оперативної пам’яті.[18] Чіп DS1302 є поліпшеним аналогом DS1202. Основні відмінності між ними – додаткові 7 байт оперативної пам’яті та два виводи для під’єднання живлення на платі.

DS1302 може відображати інформацію про поточну дату і визначати кількість днів у місяці, включно з днями високосного року. Модуль підтримує 24-годинне і 12-годинне відображення. Інформація може передаватися і прийматися у вигляді 1-байтових або 31-байтових пакетів. [18]

Модуль може керуватися різноманітними платформами та мікропроцесорними пристроями через інтерфейс I2C.

Для підключається DS1302 до мікроконтролера на платі є п’ять виходів: VCC і GND використовуються для живлення плати від зовнішнього джерела, вихід CLK – тактовий сигнал, DAT – виведення даних для подальшого опрацювання в мікроконтролері, RST – скидання даних. [18]

DS1302 має дуже низьке споживання енергії, що дозволяє йому працювати в автономному режимі від батарей живлення протягом довгого періоду часу.

Загалом, DS1302 – дешевий модуль китайського виробництва, технології якого відомі вже багато років, який виконує свої функції на відмінно.

2.1.5. Рідкокристалічний дисплей 20x4 з шиною I2C

Рідкокристалічний дисплей 20x4 – це пристрій виведення інформації, який має по 20 символів у кожному з 4 рядків. Цей тип дисплеїв часто використовується в різних проектах на базі мікроконтролерів для відображення текстової інформації.

Шина I2C (Inter-Integrated Circuit) забезпечує спосіб зв’язку між мікроконтролером та дисплеєм за допомогою всього двох проводів: лінії даних (SDA) та лінії тактового сигналу (SCL). [20] Це робить підключення дисплея до пристрою керування набагато простішим та зменшує кількість необхідних пінів.

Рідкокристалічний дисплей може виводити букви, цифри, спеціальні символи та навіть створювати просту графіку шляхом керування окремими символами в певних позиціях. Також, цей дисплей має підсвітку, що забезпечує

хорошу видимість інформації в умовах низької освітленості. На рис. 2.5 зображений використаний у проекті рідкокристалічний дисплей.

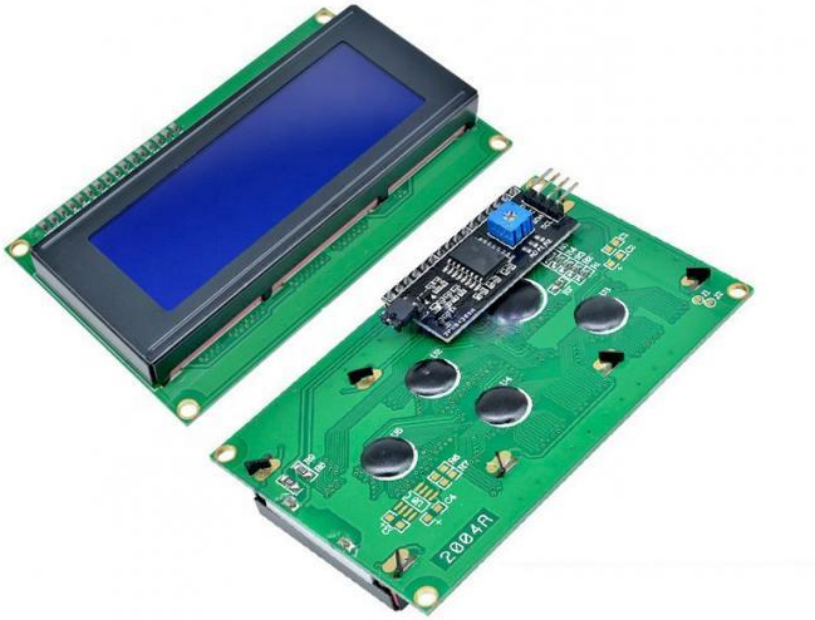


Рисунок 2.5. – LCD (liquid crystal display) дисплей 20x4 з шиною I2C і підтримкою кирилиці [21]

На практиці, даний дисплей може працювати і при нижчій напрузі за стандартну, але яскравість істотно зменшується.

2.1.6. Wi-Fi модуль DevKit V1 з ESP-32

30-контактний модуль розробника DevKit V1 побудований на новому компактному високопродуктивному модулі компанії Espressif, ESP-WROOM-32, і призначений для широкого спектра застосувань. Модуль містить усі периферійні пристрої, необхідні для швидкого і зручного оперування ESP-WROOM-32.

Цей модуль підтримує dual mode Bluetooth, що означає, що він працює як в режимі “classic”, так і в режимі BLE, що розширює можливості підключення. Також є швидкий Wi-Fi 802.11 b/g/n до 150 Мбіт/с. [22]

З мінімальною чутливістю -98 дБм та широким діапазоном робочих температур від -40°C до +125°C, цей модуль може працювати в різних умовах навколишнього середовища. Енергоефективність до 20 мкА у режимі глибокого сну дозволяє створювати автономні проекти.

Щодо технічних характеристик, модуль оснащений USB-UART конвертером CP2102 та має максимальний струм стабілізатора напруги 800мА.

На рис. 2.6 зображений модуль DevKit V1 з ESP-32, який був використаний у проєкті.

Рисунок 2.6. – Wi-Fi модуль DevKit V1 з ESP-32 [23]

Компактні розміри плати і програмне забезпечення з різноманітними режимами та захистом роблять DevKit v1 ідеальним вибором для розробки даного дипломного проекту.

2.1.7. Модуль microSD карти

Модуль для microSD карт представляє собою пристрій, спеціально розроблений для використання microSD. На рис. 2.7 представлено даний модуль.

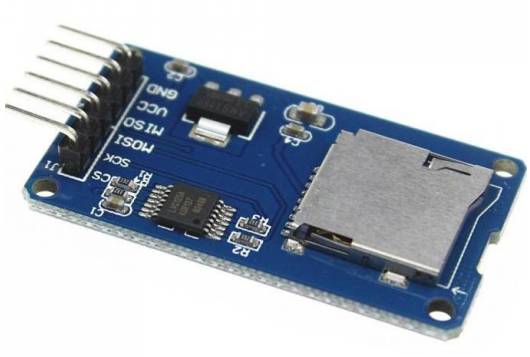


Рисунок 2.7. – Модуль microSD карти [24]

Модуль може працювати в діапазоні напруги від 4.5 В до 5.5 В для сумісності з різними джерелами живлення. Модуль має вбудований стабілізатор AMS1117-3.3 – це стабілізатор напруги, який забезпечує стабільне живлення з напругою 3.3 В.

Модуль спілкується через SPI (Serial Peripheral Interface) – це протокол зв’язку, який дозволяє обмінюватися даними між мікроконтролером та периферійними пристроями.

Модуль має перетворювач рівнів 74LVC125 – це пристрій, який перетворює логічні рівні між різними пристроями, у цьому випадку між контролерами Arduino різних моделей.

Модуль може споживати струм від 0.2 мА до 200 мА, що робить його енергоефективним. Підтримуються логічні рівні 3.3 В та 5 В, що дозволяє модулю працювати з різними типами контролерів.

Модуль призначений для використання з microSD картами (об’ємом до 2 ГБ) і microSDHC картами (об’ємом до 32 ГБ), що робить його універсальним засобом зберігання даних. Модуль має розміри 42x24x12 мм і важить приблизно 5 грам, що робить його компактним і легким для використання в різних проектах.

2.1.8. MicroSD карта

SanDisk Extreme Pro microSDHC 32GB [25] є карткою пам’яті, яка пропонує високу продуктивність і надійність для різноманітних пристроїв. Ця карта пам’яті має ємність 32 ГБ і використовує технологію NAND (not and) Flash для забезпечення швидкої передачі даних. За допомогою адаптера SD (Secure Digital), включеного в комплект, вона також може бути використана з пристроями, що підтримують формат SD. Карта підтримує швидкості зчитування даних до 100 МБ/с і має рейтинг швидкості Class 10.

Також варто відзначити, що ця карта має рейтинг продуктивності програм Class A1, що означає, що вона забезпечує швидкий доступ до додатків на пристроях Android. У комплекті йде програмне забезпечення Rescue Pro Deluxe, яке допомагає відновити втрачені або випадково видалені файли.

Щодо фізичних характеристик, карта має розміри 15 мм у довжину, 11 мм у ширину та 1 мм у висоту. Вона вироблена з високоякісних матеріалів. Ця карта живиться від номінальної напруги 3.3 В.

2.2. Технології розробки вбудованого програмного забезпечення пристрою для збору даних збору врожаю

2.2.1. Бібліотека RtcDS1302 by Makuna

Бібліотека RtcDS1302 by Makuna [26] призначена для використання з годинниками реального часу (RTC) на базі чіпів DS1302, які використовуються для точного вимірювання часу у мікроконтролерних проектах.

Першим кроком у використанні бібліотеки є підключення модуля RTC до мікроконтролера і налаштування необхідних пінів. Бібліотека RtcDS1302 дозволяє зручно встановлювати зв’язок з RTC за допомогою простих функцій ініціалізації.

Крім часу, бібліотека RtcDS1302 також дозволяє працювати з датою. Можна зчитувати поточну дату з RTC і встановлювати нові значення дати за потреби.

DS1302 має кілька режимів роботи, таких як 24-годинний або 12-годинний режими, формат виводу дати тощо. Бібліотека RtcDS1302 дозволяє зручно встановлювати та змінювати ці налаштування в залежності від потреб.

Багато RTC модулів, зокрема DS1302, мають вбудовану EEPROM (Electrically Erasable Programmable Read-Only Memory), яка може використовуватися для зберігання додаткової інформації. Бібліотека RtcDS1302 надає зручний доступ до цієї EEPROM для зберігання додаткових даних.

Як і багато популярних бібліотек для мікроконтролерів, RtcDS1302 має добре написану документацію і активну спільноту користувачів, що робить його використання ще більш зручним.

2.2.2. Бібліотека LiquidCrystal I2C by Frank de Brabander

Бібліотека LiquidCrystal I2C by Frank de Brabander [27] – це інструмент для роботи з рідкокристалічними дисплеями (LCD) через шину I2C на мікроконтролерах.

LiquidCrystal I2C надає зручний і простий інтерфейс для роботи з LCD. З його допомогою ви можете легко виводити текст, створювати власні символи, керувати курсором і багато іншого, необхідного для роботи з LCD.

Бібліотека дозволяє легко використовувати LCD в різних режимах, включаючи 4-бітний та 8-бітний режими передачі даних. Деякі версії бібліотеки також підтримують виведення кириличних символів на LCD, що робить її ідеальним вибором для проектів, які потребують виведення тексту українською.

Деякі варіанти бібліотеки дозволяють зручно налаштовувати контрастність та яскравість LCD дисплея безпосередньо з мікроконтролера. LiquidCrystal I2C має добре написану документацію і активну спільноту користувачів.

Простий приклад використання бібліотеки LiquidCrystal I2C для відображення тексту на LCD дисплеї зображений на рис. 2.8.

```
#include <Wire.h>  // Підключаємо бібліотеку для роботи з шиною I2C
#include <LiquidCrystal_I2C.h>  // Підключаємо бібліотеку LiquidCrystal I2C

// Ініціалізуємо об'єкт LiquidCrystal_I2C з адресою 0x27 та розміром 20x4
LiquidCrystal_I2C lcd(0x27, 20, 4);

void setup() {
  // Ініціалізуємо LCD дисплей
  lcd.init();
  // Встановлюємо курсор в лівий верхній кут
  lcd.setCursor(0, 0);
  // Виводимо текст на LCD дисплей
  lcd.print("Hello, World!");
}

void loop() {
  // Нічого не робимо в циклі loop, оскільки ми просто виводимо текст один раз при запуску
}
```

Рисунок 2.8. – Код прикладу використаніня бібліотеки LiquidCrystal I2C

2.2.3. Бібліотека Keypad by Mark Stanley

Бібліотека Keypad by Mark Stanley [28] – це інструмент, який забезпечує просту та ефективну взаємодію з матричними клавіатурами на мікроконтролерах. Завдяки цій бібліотеці, робота з клавіатурою стає максимально зручною та гнучкою. Дозволяючи зчитувати введені користувачем символи та реагувати на події, вона стає невід'ємною частиною проектів, де потрібно взаємодія з користувачем через кнопки.

Бібліотека Keypad дозволяє налаштувати різні параметри клавіатури, такі як розмір матриці та взаємозв'язок між клавішами та символами. Keypad може працювати з клавіатурами різних розмірів та конфігурацій. Бібліотека Keypad дозволяє виявляти не тільки сам факт натискання клавіші, а й момент її відпускання.

Приклад використання зображений на рис. 2.9.

```cpp
#include <Keypad.h>
const byte ROWS = 4; // Кількість рядків клавіатури
const byte COLS = 4; // Кількість стовпців клавіатури
char keys[ROWS][COLS] = { // Матриця символів, яка визначає клавіатурну сітку
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

byte rowPins[ROWS] = {9, 8, 7, 6}; // Піни, що використовуються для підключення рядків клавіатури
byte colPins[COLS] = {5, 4, 3, 2}; // Піни, що використовуються для підключення стовпців клавіатури

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup() {
  Serial.begin(9600); // Ініціалізація з'єднання з монітором портів
}

void loop() {
  char key = keypad.getKey(); // Зчитуємо натискану клавішу

  if (key != NO_KEY) { // Перевіряємо, чи натиснута будь-яка клавіша
    Serial.println(key); // Виводимо натискану клавішу в монітор портів
  }
}
```

Рисунок 2.9. – Код прикладу використання бібліотеки Keypad

2.2.4. Бібліотека HX711.h

Бібліотека HX711.h [29] використовується для роботи з платою HX711. Основна функція цієї бібліотеки – спрощення взаємодії з ваговими датчиками, що забезпечує зчитування великої кількості даних від вагових сенсорів і конвертує їх у вагу.

Бібліотека HX711.h написана мовою C++, і вона надає зручний інтерфейс для взаємодії з датчиками ваги. Вона дозволяє легко зчитувати дані з вагових сенсорів через інтерфейс SPI (Serial Peripheral Interface).

Основні функції, які надає бібліотека HX711.h, включають:

- ініціалізація датчика ваги;
- зчитування ваги;
- калібрування датчика;
- налаштування параметрів датчика, таких як режим роботи і рівень підсилення.

2.2.5. Бібліотека SPI.h

Бібліотека SPI.h [30] дозволяє легко взаємодіяти з пристроями, які підтримують протокол SPI. SPI є популярним протоколом зв'язку, який дозволяє обмін даними між мікроконтролером (наприклад, ESP32) та іншими пристроями, такими як сенсори, дисплеї, пам'ять та інші периферійні пристрої.

Бібліотека дозволяє ініціалізувати і налаштувати пристрій з відповідними параметрами, такими як швидкість передачі даних, режим роботи, порядок передачі бітів тощо.

Вона надає функції для передачі даних з контролера до підключених пристроїв, а також має функції для отримання даних від підключених пристроїв по шині SPI.

Протокол SPI забезпечує обмін даними в обох напрямках за допомогою трьох основних ліній: SCK (Serial Clock), MOSI (Master Output Slave Input) та MISO (Master Input Slave Output). Деякі пристрої також можуть мати додаткові лінії, такі як SS (Slave Select), які використовуються для вибору конкретного пристрою у мережі SPI.

Кожна мережа SPI має один мікроконтролер, який виконує роль master, тобто ініціює та керує обміном даними, і один або кілька пристроїв, які виконують роль slave, тобто приймають команди від майстра та відправляють дані відповідно до цих команд.

Master генерує сигнал тактового імпульсу SCK, який використовується для синхронізації передачі даних між пристроями. Частота цього сигналу визначається майстром і повинна відповідати характеристикам пристроїв у мережі SPI.

Контролер відправляє дані на пристрої за допомогою лінії MOSI, а пристрій відправляє дані майстру за допомогою лінії MISO. Ці лінії забезпечують двонаправлену передачу даних між майстром та ведучим.

SPI має кілька режимів передачі даних, які визначають спосіб передачі та прийому бітів даних. Ці режими включають різні комбінації параметрів тактового сигналу та фази даних.

Переваги SPI полягають у його високій швидкості передачі даних, простоті реалізації та низькій затримці передачі, але чим довший дріт SPI, тим більше може впливати шум і втрати сигналу.

2.2.6. Бібліотека SD.h

Бібліотека SD.h [31] – це набір функцій, який дозволяє взаємодіяти з SD картами. Бібліотека дозволяє легко ініціалізувати підключену SD картку, надає функції для зчитування даних з файлів на SD картці та запису нових даних.

Бібліотека дозволяє створювати нові файли на SD картці та видаляти існуючі файли за потреби, також має функції для навігації по каталогам.

Бібліотека дозволяє перевірити, чи доступна SD картка для зчитування та запису даних. Вона має механізми для обробки помилок, що можуть виникати під час роботи з SD карткою, таких як недостатня пам’ять або неможливість зчитати файл.

У даній роботі бібліотека SD.h відіграє ключову роль для роботи з даними збору.

2.2.7. Бібліотека wifi.h

Бібліотека wifi.h [32] дозволяє забезпечити підтримку бездротового зв’язку Wi-Fi на мікроконтролерах. Бібліотека дозволяє легко здійснити підключення до доступних мереж Wi-Fi, введення пароля, якщо потрібно, та налаштування параметрів підключення.

Вона також надає можливість створити свою власну мережу Wi-Fi, до якої можуть підключатися інші пристрої. Бібліотека дозволяє створювати веб-сервери, які можуть обслуговувати HTTP-запити від клієнтів. Вона також

дозволяє створювати клієнтів, які можуть здійснювати HTTP-запити до віддалених серверів.

Бібліотека wifi.h підтримує створення як UDP, так і TCP з'єднань для обміну даними з іншими пристроями через мережу Wi-Fi. Вона дозволяє легко взаємодіяти з хмарними сервісами, такими як ThingSpeak, Firebase, AWS і т. д., для зберігання та обробки даних. Бібліотека підтримує різні методи шифрування та аутентифікації, що дозволяють забезпечити безпеку підключення до Wi-Fi мережі.

Розуміння принципу роботи бездротової мережі Wi-Fi корисно для ефективного використання бібліотеки wifi.h. Основними елементами бездротової мережі Wi-Fi є точки доступу (AP), клієнти (кінцеві пристрої) та мережева інфраструктура, яка дозволяє їм взаємодіяти.

Точки доступу – це пристрої, які створюють бездротові мережі і забезпечують доступ до Інтернету або локальних ресурсів. Вони виступають у ролі центральних вузлів мережі, до яких можуть підключатися інші пристрої, такі як клієнти.

Клієнти – це бездротові пристрої, які підключаються до точок доступу для доступу до мережі. Це можуть бути комп'ютери, смартфони, мікроконтролери та інші пристрої.

Мережна інфраструктура Wi-Fi включає в себе різні компоненти, такі як маршрутизатори, комутатори та інші мережні пристрої, які дозволяють точкам доступу обмінюватися даними між собою та із зовнішніми мережами.

Загалом, порядок роботи мікроконтролера у мережі Wi-Fi за допомогою бібліотеки wifi.h зазвичай включає такі кроки:

1. Пошук доступних мереж Wi-Fi
2. Аутентифікація та обмін ключами
3. Отримання IP-адреси
4. Взаємодія з мережею (Обмін даними з іншими пристроями у мережі, виконання HTTP-запитів, інше)

5. Обробка помилок та перепідключення
6. Від’єднання від мережі

2.2.8. Бібліотека Async TCP

Бібліотека Async TCP [33] – це бібліотека, яка дозволяє створювати асинхронні TCP-з’єднання (Transmission Control Protocol) на мікроконтролерах. Вона відіграє дуже важиву роль у проекті.

Вона дозволяє створювати як TCP-сервери, які приймають з’єднання від інших пристроїв, так і TCP-клієнти, які підключаються до віддалених серверів. Вона може обробляти одночасно декілька з’єднань з різних клієнтів або до різних серверів, що дозволяє створювати багатоклієнтські або багатосерверні застосунки.

Бібліотека працює у асинхронному режимі, що означає, що вона може обробляти дані в одному з’єднанні, тоді як інші з’єднання можуть продовжувати працювати без блокування. Вона має можливість встановлювати callback-функції, які викликаються при подіях, таких як отримання нових даних, втрата з’єднання або успішне підключення.

Деякі версії бібліотеки можуть підтримувати TLS (Transport Layer Security) для забезпечення безпеки мережевих з’єднань. Також бібліотека надавати підтримку вищих рівнів мережевих протоколів, таких як HTTP, FTP, MQTT, що дозволило у проекті створити якісний веб сервер на базі контролера.

2.2.9. Файлова система SPIFFS

SPIFFS (Serial Peripheral Interface Flash File System) [34] – це легка файлова система, спеціально розроблена для мікроконтролерів, таких як ESP32, які мають обмежену ємність пам’яті та використовують флеш-пам’ять. Вона ефективно взаємодіє з флеш-чіпом SPI ESP32, забезпечуючи стійке зберігання даних, які не зникають після вимкнення живлення.

SPIFFS пропонує фундаментальні операції файлової системи, такі як читання, запис, видалення та створення файлів і каталогів. Це дозволяє ефективно керувати та організовувати дані.

Розмір SPIFFS налаштовується розробником. Він може займати як 0 Мб так і 4 Мб, можливі й інші варіанти. SPIFFS може зберігати параметри конфігурації для проекту, такі як облікові дані Wi-Fi, маркери доступу або налаштування користувача. Це усуває необхідність повторної конфігурації пристрою після вимкнення живлення.

Доступ до флеш-пам’яті зазвичай повільніший, ніж до RAM (Random Access Memory), що є мінусом, але розмір її в рази більший, тому файли html, файли для друку на термопринтері зберігаються саме на ній.

2.2.10. Бібліотека Bluetooth Serial

Бібліотека Bluetooth Serial [35] – це набір функцій, який дозволяє забезпечити бездротове з’єднання і обмін даними між Bluetooth-пристроями.

Ця бібліотека використовує простий інтерфейс, що дозволяє легко інтегрувати її в будь-який проект. Вона надає можливість встановлювати бездротові з’єднання навіть для початківців у програмуванні. Завдяки її функціональності, можливо легко здійснювати передачу даних, керувати пристроями чи створювати додатки для IoT (Internet of Things).

Bluetooth Serial може працювати як з класичним Bluetooth, так і з Bluetooth Low Energy (BLE), що розширює її можливості. Вона дозволяє налаштовувати різні параметри з’єднання, такі як шифрування даних чи автоматичне з’єднання з відомими пристроями.

2.2.11. Мова програмування C++

C++ [36] – це мова програмування загального призначення. Вона еволюціонувала з мови C, додаючи до неї можливості об’єктно-орієнтованого програмування, узагальнення та інші функції. C++ широко використовується в

різних сферах, включаючи розробку системного програмного забезпечення, вбудованих систем, ігрових програм, графічних застосунків та наукових обчислень.

C++ підтримує об'єктно-орієнтоване програмування, яке дозволяє розробникам моделювати реальні світові сутності за допомогою програмних об'єктів. C++ підтримує узагальнення, що дозволяє створювати код, який може працювати з різними типами даних без необхідності повторного написання коду. C++ забезпечує високий ступінь контролю над пам'яттю та ресурсами комп'ютера, що робить її придатною для розробки ефективних програм. C++ код може бути компільований та виконуватися на різних платформах з мінімальними змінами. C++ має широкий спектр функцій, що робить її придатною для вирішення складних задач.

Переваги: швидкість, гнучкість, широке застосування, велика база матеріалів, велике співтовариство. Недоліки: складність, можливе виникнення важковловимих помилок, погана модульність.

2.3. Технології розробки сайту для обліку товарів

2.3.1. Мова програмування Golang

Golang [37], також відома як Go – це мова програмування загального призначення, розроблена компанією Google.

Go має простий і лаконічний синтаксис, подібний до C. Змінні в Go завжди передаються за значенням, що робить код більш передбачуваним і менш схильним до помилок. Go використовує інтерфейси для визначення наборів методів, які можуть реалізовувати різні типи. Це сприяє поліморфізму та розширюваності коду. Go має вбудований сміттєзбірник, який автоматично керує виділенням і звільненням пам'яті. Це спрощує роботу і зменшує ризик витоків пам'яті.

Go має багато переваг, які роблять її популярним вибором для розробників програмного забезпечення. До них належать: простота,

продуктивність, масштабованість, велика кількість стандартних бібліотек, велика та активна спільнота. Однак є і недоліки: порівняно молода мова, не має деяких функцій, які є звичайними в інших мовах.

Деякі мови програмування, такі як Python або Java, можуть бути кращим вибором для певних проектів. Однак, завдяки своїм численним перевагам, Golang була обрана для розробки бекенду.

2.3.2. Вебфреймворк Gin

Gin [38] – це високопродуктивний вебфреймворк для Go, який поєднує в собі простоту, гнучкість та високу продуктивність. Він швидко став одним з найпопулярніших фреймворків Go завдяки своїм численним перевагам.

Gin має простий і лаконічний API, який легко вивчити та використовувати. Він пропонує зручні функції для маршрутизації, обробки запитів, рендерингу шаблонів та багато іншого. Він підтримує різні стилі програмування. Gin оптимізований для продуктивності та використовує ефективні алгоритми маршрутизації та рендерингу. Він може обробляти велику кількість запитів з низькою затримкою. Gin має вбудовану підтримку CORS (Cross-Origin Resource Sharing), JSON та HTML. Він підтримує middleware, що дозволяє розширювати функціональність фреймворку. Gin має активну спільноту розробників, які надають підтримку та ресурси. Gin часто порівнюють з іншими популярними фреймворками Go, такими як Echo та Buffalo. Gin має ряд переваг перед цими фреймворками, тому було обрано саме його.

2.3.3. Бібліотека Gorm

Gorm [39] – це бібліотека для мови програмування Go, що допомагає спростити і вдосконалити взаємодію з базами даних. Розроблена на основі активних записів, вона дозволяє розробникам працювати з базами даних за допомогою структур даних, що відображають таблиці баз даних.

Gorm підтримує широке число баз даних, включаючи MySQL, PostgreSQL, SQLite, MariaDB, Oracle та SQL Server. Gorm має простий та інтуїтивно зрозумілий API, який легко вивчити та використовувати. Gorm може автоматично створювати та оновлювати схеми баз даних на основі моделей Go.

Бібліотека дозволяє визначати відносини між таблицями, такі як один до одного, один до багатьох, багато до багатьох, і автоматично виконує необхідні запити для роботи з цими відносинами. Gorm підтримує транзакції, що дозволяє вам виконувати атомарні операції над даними.

Gorm можна використовувати для: створення та оновлення даних, отримання даних з бази даних, видалення даних з бази даних, виконання складних запитів, робота з транзакціями, моделювання зв'язків між моделями.

Gorm часто порівнюють з іншими популярними ORM для Go, такими як xorm та Beego orm. Gorm має ряд переваг перед цими ORM, включаючи його простоту використання, потужні функції та активну спільноту.

2.3.4. Мова програмування TypeScript

TypeScript [40] – це мова програмування, яка базується на мові JavaScript і додає до неї сильну типізацію.

TypeScript дозволяє визначати типи для змінних, параметрів функцій та значень, підтримує концепції класів та інтерфейсів, може використовувати існуючий код JavaScript, надає можливість налаштовувати різні параметри компіляції.

У TypeScript розробники можуть явно оголошувати типи даних для змінних, параметрів функцій і значень, що повертаються. Наприклад, змінні можуть бути оголошені з певним типом, як-от числовий, строковий, булевський, масив або користувацький тип, визначений в інтерфейсі або класі. Така статична типізація допомагає виявити помилки, пов'язані з типами, на етапі розроблення, до виконання коду. Система типів TypeScript може виявити розбіжності між очікуваними та фактичними типами даних, унаслідок чого код

стає більш передбачуваним і менш схильним до помилок. На рис. 2.9 представлено популярність мови TypeScript.

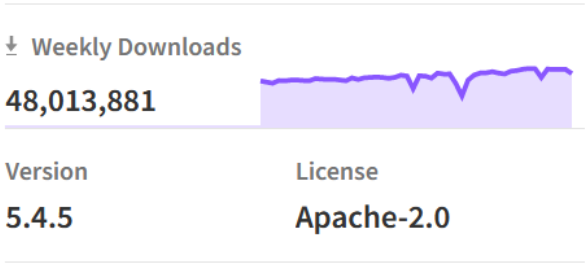


Рисунок 2.9. – Інформація про популярність TypeScript [41]

Кількість завантажень в тиждень перевищує 48 мільйонів, остання версія 5.4.5, і нові стабільні версії з'являються кожний місяць, розробницькі – кожен день.

2.3.5. Фреймворк розробки вебдодатків Angular

Angular [42] є одним з найпопулярніших фреймворків для розробки вебдодатків на сучасному ринку програмного забезпечення. Його виникнення пов'язане з необхідністю в ефективному та структурованому підході до розробки складних клієнтських застосунків.

Angular базується на ідеї компонентів, що дозволяє побудувати додаток з невеликих незалежних компонентів, які можна повторно використовувати та модульнізувати. Angular надає потужні інструменти для створення динамічних HTML-шаблонів з використанням директив, які розширюють функціональність HTML.

Angular має механізм dependency injection, який дозволяє розробникам вводити залежності в компоненти, що полегшує тестування та створення зв'язків між компонентами. Angular надає зручні засоби для управління маршрутизацією в додатках, що дозволяє створювати односторінкові додатки (SPA) з легкістю. Angular має потужний модуль форм, який дозволяє легко створювати та валідувати форми в додатках.

Чому саме Angular був обраний у проекті? Бо він є повноцінним фреймворком з MVC і має двосторонню прив’язку даних, що дуже покращує подання аналітичних даних з гнучкою зміною параметрів і пошуком клієнтом.

2.3.6. База даних PostgreSQL

PostgreSQL [43] є однією з найпопулярніших відкритих реляційних баз даних у світі. PostgreSQL пропонує широкий спектр функцій, які роблять його відмінним вибором для багатьох сценаріїв розробки програмного забезпечення. PostgreSQL підтримує багато типів даних, включаючи географічні дані, JSON, XML, і багатовимірні масиви. Система дозволяє створювати користувацькі функції та типи даних, що розширює її можливості. PostgreSQL має високу підтримку транзакцій, що забезпечує цілісність даних і відмінну стійкість до відмов. PostgreSQL підтримує різні мови програмування для написання функцій, включаючи SQL, PL/pgSQL, Python, Perl, і багато інших. Вона надає різні типи індексів та механізми оптимізації запитів для покращення продуктивності. PostgreSQL має можливості для високої доступності та резервного копіювання для забезпечення безпеки даних.

Враховуючи його безкоштовність та високу продуктивність, PostgreSQL зазвичай виявляється ефективним з точки зору витрат на обслуговування та розвиток. PostgreSQL легко масштабується з ростом обсягу даних та навантаження. PostgreSQL надає багато параметрів конфігурації, що дозволяє налаштувати його під конкретні потреби проекту.

Він не має дефіциту розробників, які знайомі з СУБД. Головна причина обрання – безкоштовність і легкість використання на VPS (Virtual Private Server), що полегшить масштабування та зменшить бюджет.

ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі дипломної роботи проведено детальний аналіз та огляд технологій, необхідних для реалізації апаратно-програмного комплексу для обліку та контролю збору врожаю. Було вибрано комплектуючі для розробки пристрою, технології для розробки вбудованого програмного забезпечення та технології для розробки вебсайту.

Для розробки пристрою обрано наступні компоненти: тензодатчик, HX711 для зчитування сигналу від тензодатчика, клавіатура матрична 6x4 для введення даних, DS1302 для роботи з реальним часом, LCD дисплей 20x4 з шиною I2C для відображення інформації, Wi-Fi модуль DevKit V1 з ESP-32 для забезпечення бездротового зв’язку, модуль microSD для збереження даних.

Для розробки вбудованого програмного забезпечення використано наступні технології та бібліотеки: RtcDS1302 для роботи з реальним часом, Keypad для обробки введених даних з клавіатури, HX711 для зчитування ваги, SPI та SD для роботи з SD картою, Wi-Fi та Async TCP для мережевого зв’язку та HTTP-сервера, SPIFFS для роботи з файловою системою на мікроконтролері, Bluetooth Serial для можливості використання бездротового зв’язку з Bluetooth.

Для розробки вебсайту обрано наступні технології: Golang як мову програмування, Gin та Gorm для розробки серверної частини, TypeScript та Angular для розробки клієнтської частини, PostgreSQL як систему управління базами даних.

Вибір вказаних компонентів та технологій базується на їхній придатності для конкретних завдань розробки апаратно-програмного комплексу для обліку та контролю збору врожаю, а також на їхній широкій популярності та підтримці спільнотами розробників.

РОЗДІЛ 3
РОЗРОБЛЕННЯ ПРОГРАМНОГО-АПАРАТНОГО КОМПЛЕКСУ ДЛЯ ОБЛІКУ ТА КОНТРОЛЮ ЗБОРУ ВРОЖАЮ

3.1. Проектування пристрою для збору даних про урожай

3.1.1. Проектування електричної схеми та конструкції пристрою

Підключення необхідних модулів до ESP32 було важливим етапом розробки проекту. Перше, що було зроблено – це підключення SD-карти до ESP32 для зберігання та доступу до даних. Спочатку було важливо визначити правильні піни для з'єднання з ESP32. Після цього проводилося підключення карти згідно з встановленими пінами. Перевірка роботи SD-карти була проведена через перевірку доступу до збережених файлів.

Наступним етапом було підключення модуля RTC для забезпечення точного відображення часу на пристрої. Цей модуль був підключений до ESP32 через визначені піни, які забезпечували зв'язок з контролером. Для правильної роботи RTC було також необхідно вставити батарейку, що дозволяла модулю продовжувати роботу навіть без живлення.

Підключення LCD-екрану було ще одним важливим кроком у розробці проекту. Цей екран використовувався для відображення інформації для користувача. Підключення проводилось через інші піни зображені у Додатку 1, які передавали відповідні сигнали з ESP32 на екран. Після підключення було перевірено правильність роботи екрану через відображення тесту.

Матрична клавіатура була включена до проекту для взаємодії з користувачем та введення даних. Її підключення вимагало встановлення зв'язку між рядами та стовпцями клавіш і мікроконтролером ESP32. Кожен ряд та кожен стовпчик клавіатури були підключені до відповідних пінів на ESP32.

Тензодатчик був підключений до ESP32 через модуль HX711, який дозволяв зчитувати сигнали з датчика та обробляти їх.

Модуль HX711 був підключений до ESP32 через піни, які забезпечували зв’язок з мікроконтролером. Після підключення необхідно було правильно сконфігурувати модуль HX711 та налаштувати його для оптимального зчитування сигналів з тензодатчика.

Для живлення пристрою використовувався акумулятор. Він був підключений до ESP32 через вхід для зовнішнього живлення. Кнопка була використана для управління живленням пристрою. Натискання кнопки включало або вимикало живлення, дозволяючи користувачеві керувати роботою пристрою за потреби.

Під час підключення модулів до ESP32 виникла проблема з неправильним розумінням кольорів проводів. На початку я вважав, що червоний провід позначає плюс, а чорний – мінус. Однак, під час перевірки працездатності пристрою виявив, що підключення було неправильним. Розглядаючи цю проблему, я звернув увагу на те, що різні виробники можуть використовувати різні кольори проводів для позначення плюса та мінуса. Щоб вирішити цю проблему, я запитав експерта та дослідив рекомендації виробників. Це дозволило правильно ідентифікувати кольори проводів для підключення кожного модуля.

Також, була проблема з модулем microSD картки, так як він містить лінійний стабілізатор AMS1117 3.3В, який повинен підтримувати напругу на рівні 3.3 В для правильної роботи картки, так як картки працюють від 3.3 В, а модуль може забезпечуватися живленням у 5 В. Але плата ESP32 дає струм напругою 3.3 В, а стабілізатор перетворює її на приблизно 2.5 В, що не достатньо для живлення. Тому було вирішено прибрати даний компонент з модуля. Після цього все запрацювало стабільно.

3.1.2. Розробка вбудованого програмного забезпечення

Програмне забезпечення створювалось поступово, а кількість бібліотек змінювалась, залежно від потреб або при знаходженні більш кращих варіантів.

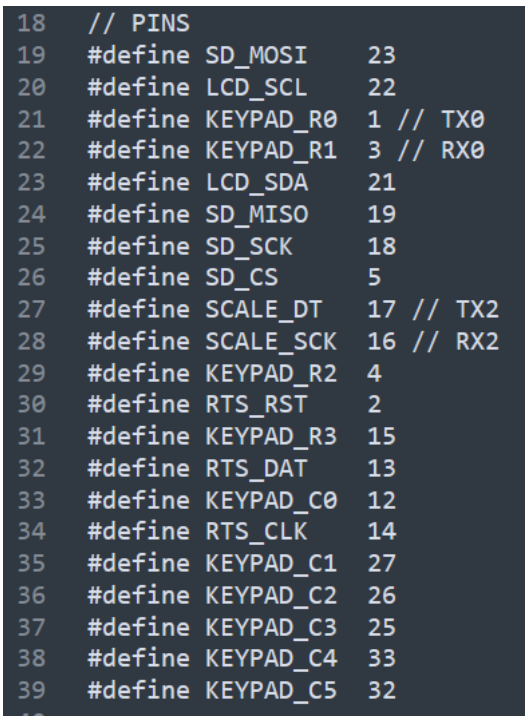
Але структура завчасно була відома. Тому поступово буде розгорнуто описано реалізований програмний код. Функціонал представлено у Додатку 3.

Спочатку за допомогою директиви #include додаються всі бібліотеки як зображено на рис. 3.1.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h> // LiquidCrystal I2C by Frank de Brabander
#include <Keypad.h> // Keypad by Mark Stanley
#include <ThreeWire.h> // Rtc by Makuna
#include <RtcDS1302.h> // Rtc by Makuna
#include <HX711.h> // HX711 Arduino library by Bogdan Necula
#include <FS.h>
#include <SPI.h>
#include <SD.h> // SD by Arduino, SparkFun
#include <Preferences.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <SPIFFS.h>
#include <BluetoothSerial.h>
#include <time.h>
```

Рисунок 3.1. – Частина коду де додаються всі бібліотеки

Далі поглянемо рис. 3.2, де визначено контакти, пов’язані з модулем SD-карти (SD_MOSI, SD_MISO, SD_SCK, SD_CS). Вказано контакти для взаємодії з LCD-дисплеєм (LCD_SCL, LCD_SDA). Виводи для матричної клавіатури (KEYPAD_R0, KEYPAD_R1, KEYPAD_R2, KEYPAD_R3, KEYPAD_C0, KEYPAD_C1, KEYPAD_C2, KEYPAD_C3, KEYPAD_C4, KEYPAD_C5) визначені для з’єднань рядків і стовпців. Визначено контакти, пов’язані з модулем тензодатчика (SCALE_DT, SCALE_SCK). Визначено контакти, пов’язані з модулем RTC (RTS_RST, RTS_DAT, RTS_CLK).



```
// PINS
#define SD_MOSI    23
#define LCD_SCL    22
#define KEYPAD_R0  1 // TX0
#define KEYPAD_R1  3 // RX0
#define LCD_SDA    21
#define SD_MISO    19
#define SD_SCK     18
#define SD_CS      5
#define SCALE_DT   17 // TX2
#define SCALE_SCK  16 // RX2
#define KEYPAD_R2  4
#define RTS_RST    2
#define KEYPAD_R3  15
#define RTS_DAT    13
#define KEYPAD_C0  12
#define RTS_CLK    14
#define KEYPAD_C1  27
#define KEYPAD_C2  26
#define KEYPAD_C3  25
#define KEYPAD_C4  33
#define KEYPAD_C5  32
```

Рисунок 3.2. – Частина коду де визначено всі контакти

Далі в коді визначено константу PROJECT_NAME зі значенням “apricotka”. Це дозволяє звертатися до цього значення в інших частинах коду, замість повторного введення тексту “apricotka”. Аналогічно, визначаємо константу TZ зі значенням “EET-2EEST,M3.5.0/3,M10.5.0/4”, яка потрібна для позначення часової зони.

Далі в коді перевірено чи не визначено CONFIG_BT_SPP_ENABLED, CONFIG_BT_ENABLED, CONFIG_BLUEDROID_ENABLED, і якщо ні (тобто, якщо Bluetooth не доступний або не ввімкнений), компілятор генерує помилку з повідомленням про це.

Далі в коді визначаємо кількість рядків та стовпців на клавіатурі (KEYPAD_ROW_NUM, KEYPAD_COLUMN_NUM). У цьому випадку, клавіатура має 4 рядки і 6 стовпців. Визначаємо масиви для збереження пінів, які використовуються для рядків та стовпців клавіатури відповідно.

Далі в коді визначаємо двовимірний масив символів, де кожен символ відповідає кожній кнопці клавіатури.

За звичайних умов кожен рядок представляє один рядок клавіатури, а кожний стовпчик представляє одну кнопку в рядку, але через особливості клавіатури, для економії, плата має доріжки лише з одного боку. Щоб все працювало правильно, було змінено порядок символів у матриці в коді, як показано на рис. 3.3.

Рисунок 3.3. – Старий і новий порядок визначення клавіш клавіатури

Далі в коді створюємо об’єкт типу LiquidCrystal_I2C, який використовується для управління рідкокристалічним дисплеєм з інтерфейсом I2C. Параметри конструктора: адреса дисплея (0x27), кількість символів у рядку (20) і кількість рядків (4).

Далі в коді створюємо об’єкт типу Keypad, який використовується для управління клавіатурою. Параметри конструктора включають мапування клавіш makeKeymap(keys), масиви пінів pinRows і pinCols, а також кількість рядків і стовпців клавіатури.

Далі в коді створюємо об’єкт типу ThreeWire, який використовується для зв’язку через SPI-інтерфейс. Параметри конструктора: піни DAT, CLK і RST, які використовуються для зв’язку. Створюємо об’єкт типу RtcDS1302, який використовується для роботи з модулем реального часу DS1302. Параметр конструктора myWire вказує на об’єкт ThreeWire.

Далі в коді створюємо об’єкт типу HX711, який використовується для роботи з ваговим датчиком HX711. Немає параметрів конструктора, оскільки вони за замовчуванням відповідають потрібним налаштуванням.

Далі в коді створюємо об’єкт типу Preferences, який використовується для зберігання налаштувань у вигляді ключ-значення на мікроконтролері ESP32.

Ініціалізація BluetoothSerial в коді відбувається через об’єкт SerialBT. Це означає створення засобу для здійснення безпровідного зв’язку по Bluetooth. У даному випадку використовується BluetoothSerial, що дозволяє забезпечити спілкування з іншими пристроями за допомогою протоколу Bluetooth. Ініціалізація відбувається на початку програми, перед будь-якими іншими діями. Оскільки це ініціалізація, а не саме з’єднання з конкретним пристроєм, то це є лише підготовчий етап. Це дозволяє системі готуватися до можливого обміну даними через Bluetooth у майбутньому.

Далі в коді створюємо об’єкт server, який представляє собою веб-сервер. Це здійснюється за допомогою конструктора AsyncWebServer, який приймає порт 80 для прослуховування вхідних з’єднань. Також визначається функція notFound, яка буде викликатися у випадку, якщо запит користувача не вдалося знайти. Ця функція відправляє клієнту відповідь з кодом 404 та повідомленням “Not found”.

Далі розглянемо функції внутрішнього програмного забезпечення.

Функція читання файлів readFile призначена для отримання вмісту файлу з файлової системи. Параметрами є посилання на файлову систему, з якої потрібно читати файл, а також шлях до файлу, який потрібно прочитати.

Спочатку функція намагається відкрити файл за вказаним шляхом для читання. Якщо файл не відкривається або він є каталогом, функція виводить повідомлення про помилку та повертає пустий рядок. У разі успішного відкриття файлу функція читає його вміст посимвольно, додаючи кожен символ до рядка fileContent. Після завершення читання вмісту файлу, вміст рядка fileContent виводиться в Seral Monitor для відладки. Функція повертає прочитаний вміст файлу у вигляді рядка.

Функція запису в файл writeFile призначена для запису контенту у файл у файловій системі. Параметри: посилання на файлову систему, в якій буде виконано запис, шлях до файлу, у який буде виконаний запис, рядок, який буде записаний у файл. Функція спочатку спробує відкрити файл за вказаним шляхом для запису, використовуючи fs.open(path, “w”). Якщо файл не існує, він буде створений. Якщо відкриття файлу не вдалося, виведеться повідомлення про помилку. Якщо файл вдало відкрито для запису, функція виконає запис в нього за допомогою file.print(message). Якщо запис успішно виконано, виведеться повідомлення про успішний запис. У випадку невдалого запису виведеться повідомлення про помилку запису. Незалежно від результату запису, файл буде закрито за допомогою file.close(), щоб звільнити ресурси.

Функція getDateTimeString форматує об’єкт RtcDateTime у рядок, що представляє дату та час. Використовується для виведення повної інформації про дату та час, включаючи рік, місяць, день, годину, хвилину та секунду. Формат рядка: “DD.MM.YYYY HH:MM:SS”, де “DD” – день, “MM” – місяць, “YYYY” – рік, “HH” – година, “MM” – хвилина, “SS” – секунда.

Функція getDateString також форматує об’єкт RtcDateTime у рядок, але вона виводить лише інформацію про дату без часу. Формат рядка: “DD.MM.YYYY”, де “DD” – день, “MM” – місяць, “YYYY” – рік.

Функція rtcWork викликається для роботи з RTC. Спочатку вона отримує поточну дату та час з RTC. Потім вона форматує дату у рядок типу String, використовуючи функцію getDateString, яка отримує день, місяць та рік. Якщо дата є дійсною, ця функція виводить дату на LCD екран. У випадку, якщо дата не є дійсною, на екрані виводиться повідомлення про помилку.

Функція isNumber приймає рядок s в якості вхідного параметра. Вона перевіряє кожен символ у рядку, використовуючи цикл for, щоб перевірити, чи є він цифрою. Для цього використовується вбудована функція isDigit. Якщо будь-який символ у рядку не є цифрою, функція повертає false, що означає, що рядок не є числовим. Якщо всі символи у рядку є цифрами, функція повертає

true, підтверджуючи, що рядок представляє числове значення. Ця функція допомагає програмі визначити, чи коректно введені дані, які мають бути числовими, і може бути використана для перевірки коректності введених користувачем значень перед їхнім подальшим використанням у програмі.

Функція clearLCDLine призначена для очищення вказаного рядка на LCD-дисплеї. Вона приймає параметр x, який вказує номер рядка, який потрібно очистити. Після виклику цієї функції весь текст на заданому рядку буде замінений пробілами, щоб очистити його. Пробіли обрані не просто так, а для уникнення блимання екрану.

Функція inputFromKeypad призначена для отримання введених даних з клавіатури або ключових кнопок та їх відображення на LCD-дисплеї. Вона приймає параметри poss (позиція курсора на рядку), row (номер рядка, на якому потрібно вивести введені дані) та isFloat (логічне значення, яке вказує, чи очікується введення дробового числа). Під час виклику цієї функції користувач може вводити символи з клавіатури. Якщо користувач натискає кнопку “<”, то введений символ видаляється. Якщо введені дані перевищують обмеження по довжині рядка, вони не будуть відображені на екрані. Якщо користувач натискає кнопку “>”, введення завершується, і функція повертає рядок, який був введений користувачем.

Функція readFromFile виконує операцію читання з файлу у файловій системі. Приймає посилання на файлову систему, з якою буде взаємодіяти функція, та шлях до файлу, який потрібно прочитати. Функція відкриває файл за вказаним шляхом для читання. Потім вона зчитує вміст файлу у рядок. Якщо файл успішно відкрито і зчитано, вміст файлу повертається як результат функції. Якщо файл не вдалося відкрити або прочитати, повертається рядок із повідомленням про помилку.

Функція listDir призначена для отримання списку файлів у вказаній директорії. Приймає параметри fs, що вказує на тип файлової системи (у даному випадку SD-карта), dirname, який вказує шлях до директорії, та levels,

який вказує на глибину рекурсивного перегляду піддиректорій. Відкриває вказану директорію та проходиться по всіх файлам і піддиректоріях, формуючи рядок з їх назвами та типами (файл або директорія) та розміром, і повертає цей рядок.

Функція printDirectory допоміжна для функції listDir та призначена для виведення списку файлів та піддиректорій у вигляді HTML-таблиці. Приймає параметри dirname, який вказує шлях до директорії, та levels, який вказує на глибину рекурсивного перегляду піддиректорій. Відкриває вказану директорію та формує рядок HTML-коду для кожного файлу та піддиректорії, включаючи назву, тип, розмір файлу та кнопки дій (читання, завантаження, видалення).

Функція шаблонного процесора processor використовується для заміни заповнювачів в шаблонах на збережені значення. Ця функція отримує рядок var, який є заповнювачем, що потрібно замінити на вміст. Функція перевіряє, чи збігається рядок var з яким-небудь з визначених заповнювачів. Якщо такий збіг виявлено, вона повертає відповідне значення. Функція перевіряє, який саме заповнювач збігається і виконує відповідну дію:

- Якщо var дорівнює “language”, функція повертає мову, збережену в налаштуваннях.
- Якщо var дорівнює “dateTime”, функція повертає рядок з поточною датою та часом з часового модуля.
- Якщо var дорівнює “slaveName”, функція повертає ім’я Bluetooth пристрою, яке зберігається в налаштуваннях.
- Якщо var дорівнює “bluetoothPin”, функція повертає пін-код Bluetooth, який зберігається в налаштуваннях.
- Якщо var дорівнює “routerSSID”, функція повертає SSID Wi-Fi маршрутизатора, який також зберігається в налаштуваннях.
- Якщо var дорівнює “routerPassword”, функція повертає пароль Wi-Fi маршрутизатора.

– Якщо var дорівнює “calFactor”, функція повертає коефіцієнт калібрування.
– Якщо жоден з заповнювачів не збігається з var, функція повертає порожній рядок.

Функція bluetoothConnect відповідає за з’єднання по Bluetooth. Спочатку вона завершує будь-яке існуюче з’єднання сервера, відключається від мережі Wi-Fi та вимикає режим Wi-Fi. Потім функція отримує пін-код та назву пристрою, які зберігаються у пам’яті, або використовує значення за замовчуванням, якщо такі відсутні. На дисплеї виводиться інформація про те, що Wi-Fi вимкнено та почався процес підключення до пристрою з вказаною назвою.

Далі починається процес ініціалізації Bluetooth з використанням заданого імені проекту і встановленням пін-коду. Після цього виконується спроба підключення до пристрою з вказаною назвою slaveName. Після виконання спроби підключення на дисплеї виводиться відповідне повідомлення. У випадку успішного з’єднання на екрані відображається повідомлення про успішне підключення та назву підключеного пристрою. У випадку невдачі виводиться повідомлення про неможливість підключитися, і наголошується на необхідності перевірити доступність та зону покриття пристрою. Функція повертає значення типу bool, що вказує на успішність з’єднання, яке можна обробити.

Функція startServer – це частина програми, яка створює веб-сервер за допомогою мови програмування C++ та бібліотеки AsyncWebServer. Кожен обробник маршрутів відповідає на певний URL-шлях та виконує певні дії при отриманні запиту.

Маршрут “/” відповідає кореневому URL. При отриманні GET-запиту на корінь вебсайту, сервер відправляє клієнту файл “/index.html”. Маршрут “/style.css” обробляє запити на файл стилів CSS і відправляє файл “/style.css” клієнту з вказаним типом вмісту “text/css”. Маршрут “/favicon.ico” відповідає

запитам на іконку вебсайту і відправляє файл “/favicon.png” клієнту з вказаним типом вмісту “image/png”. Маршрут “/language” приймає дані про мову, які відправляються клієнтом через GET-запит. Ці дані зберігаються у налаштуваннях мікроконтролера. Маршрут “/sd” працює з файловою системою SD-карти. Він здійснює завантаження, читання та видалення файлів з карти SD, а також відображення списку файлів у кореневій директорії. Маршрут “/time/update/pc” використовується для оновлення часу на мікроконтролері за допомогою значення часової мітки, яке передається через GET-запит. Маршрут “/time/update/ntp” використовується для синхронізації часу на мікроконтролері з сервером часу NTP. Маршрут “/bluetooth” приймає дані про ім’я та пін Bluetooth-пристрою, які зберігаються у налаштуваннях мікроконтролера. Маршрут “/printer” відправляє клієнту файл “/label.txt”, який може використовуватися для друку на принтері з етикетками. Маршрут “/printer/update” оновлює вміст файлу “/label.txt” на сервері з вказаним в GET-запиті вмістом. Маршрут “/wifi” приймає дані про ім’я та пароль мережі Wi-Fi, які зберігаються у налаштуваннях мікроконтролера. Маршрут “/nvs” приймає дані для калібрування тензодатчика і зберігає їх у пам’яті мікроконтролера. Маршрут “/notFound” обробляє запити на маршрути, які не відповідають жодному зазначеному маршруту, і відправляє клієнту повідомлення про помилку “404 Not Found”. У кінці функції викликається метод server.begin(), який розпочинає роботу веб-сервера.

Функція setup() виконує початкове налаштування для пристрою. Ініціалізація з’єднання з послідовним портом за швидкістю 115200 біт на секунду, об’єкта для зберігання налаштувань проекту. Ініціалізація файлової системи SPIFFS для роботи з файлами у вбудованій пам’яті. Ініціалізація та увімкнення підсвічування для LCD дисплею. Ініціалізація SD карти за допомогою SPI зазначеними пінами. Потім перевіряється, чи вдалося ініціалізувати SD карту. Якщо ні, виводиться відповідне повідомлення на LCD та через послідовний порт, і програма завершується. Ініціалізація модуля

реального часу, перевіряється, чи правильно налаштований час в RTC, якщо ні – встановлюється час компіляції програми. Перевіряється, чи RTC не заблокований на запис, чи не відключений. Порівнюється час RTC з часом компіляції. Встановлення та калібрування параметрів ваги. Якщо збережений калібрувальний коефіцієнт не дорівнює нулю, то встановлюється цей коефіцієнт.

Функція, яка називається loop, є основною частиною програми, яка виконується безперервно після ініціалізації. Функція спочатку перевіряє, чи була натиснута будь-яка кнопка на клавіатурі. У залежності від кнопки виконується відповідна дія.

Якщо користувач натискає кнопку “X” на клавіатурі, програма починає процес введення цих даних. Перш ніж вводити нові дані, екран LCD очищається від попередніх відомостей. Програма виводить на екран повідомлення “Employee:” і очікує введення ID працівника. Користувач вводить ID за допомогою клавіатури. Якщо введене значення є числом, воно зберігається у змінну “employee”. Якщо введене значення не є числом, програма продовжує очікувати коректного введення. Після введення ID працівника програма переходить до введення ID продукту. Так само, як і з ID працівника, користувач вводить ID продукту через клавіатуру, і введене значення зберігається у змінну “product”. Після введення ID продукту програма пропонує користувачеві ввести масу зібраного врожаю. На екран виводиться повідомлення “Mass:” (Маса:), і користувач вводить масу через клавіатуру. Після введення маси програма переходить до наступного кроку.

Після введення всіх необхідних даних програма отримує поточну дату та час і формує рядок JSON, що містить всі введені дані, а також дату та час збору врожаю. Цей рядок JSON зберігається у файлі на SD карті. Якщо файл вже існує, нові дані додаються до вже існуючих. Після збереження даних на SD карту програма пропонує користувачу вибір чи потрібно надрукувати ярлик. Якщо користувач натискає “PRINT”, програма спробує підключитися до

Bluetooth і надіслати дані для друкування. Якщо користувач натискає “>”, програма завершує без друку і повертається до очікування нових команд.

Коли користувач натискає кнопку “=”, програма перевіряє доступність HX711. Це необхідно для того, щоб переконатися, що зважувальний пристрій підключений та готовий до використання. Якщо HX711 виявлена, програма переходить до кроку калібрування. Калібрування полягає у встановленні відповідності між виміряною вагою та реальною масою предмета. Користувачу відображається повідомлення на дисплеї, яке просить його ввести масу відомого об’єкта для калібрування. Після введення маси програма автоматично розпочинає процес калібрування ваги. Цей процес полягає у тому, що HX711 збирає серію вимірів. Після завершення калібрування програма розраховує коефіцієнт калібрування, який використовується для перетворення вимірів в масу в реальному часі. Калібрувальний коефіцієнт зберігається у пам’яті мікроконтролера для подальшого використання. Після успішного завершення калібрування користувачу відображається повідомлення про готовність ваги до використання на дисплеї, і програма очікує подальших дій користувача або натискання інших кнопок.

Коли користувач натискає кнопку “T”, програма викликає функцію scale.tare(). Ця дія відбувається в режимі калібрування HX711, що забезпечує скидання поточного значення нульової точки ваги. Коли вага обнуляється, вона стає відносною, тобто будь-які подальші вимірювання будуть відображати різницю між поточною вагою і значенням, яке було зараховано як “нульове” в режимі тарування.

Коли користувач натискає кнопку “+”, програма виконує виклик функції rtcWork().

Коли натиснуто кнопку “W”, програма виводить на екран текстові повідомлення, які пропонують користувачеві вибрати між двома режимами мережі: режимом точки доступу Wi-Fi (AP) або режимом підключення до існуючої Wi-Fi мережі (STA). Якщо користувач обирає режим точки доступу

Wi-Fi (AP), програма встановлює відповідний режим мережі та створює нову Wi-Fi мережу з певним ім’ям та паролем. Після цього вона відображає на екрані інформацію про те, що Bluetooth вимкнено, Wi-Fi ввімкнено, а також вказує IP-адресу точки доступу, за якою можна підключитися. Якщо користувач обирає режим підключення до існуючої Wi-Fi мережі (STA), програма вводить ім’я і пароль від цієї мережі (ці дані попередньо зберігаються у програмі) та намагається підключитися до неї. Якщо підключення вдале, на екрані виводиться інформація про те, що Bluetooth вимкнено, Wi-Fi ввімкнено, а також показується IP-адреса пристрою в мережі. Якщо ж підключення до Wi-Fi мережі невдале, програма повідомляє про це користувача на екрані. Після вибору режиму мережі відповідно до дій користувача програма виконує відповідні налаштування і дії для забезпечення доступу до мережі.

Якщо натиснуто “B”, програма намагається підключитися до Bluetooth.

3.1.3. Встановлення вбудованого програмного забезпечення

Після компіляції, вихідне програмне забезпечення має розмір більше 1 мегабайта, що не підходить для звичайної схеми розділів флеш-пам’яті ESP32.

На рис. 3.4 позначені розділи та їх розмір у стандартному форматі та після форматування. App1 служить тільки для оновлення програмного забезпечення через Wi-Fi, тому ним можна знехтувати і віддати його обсяг пам’яті під app0. Також розміру SPIFFS у 896 кілобайт цілком достатньо і навіть залишається багато вільного місця.

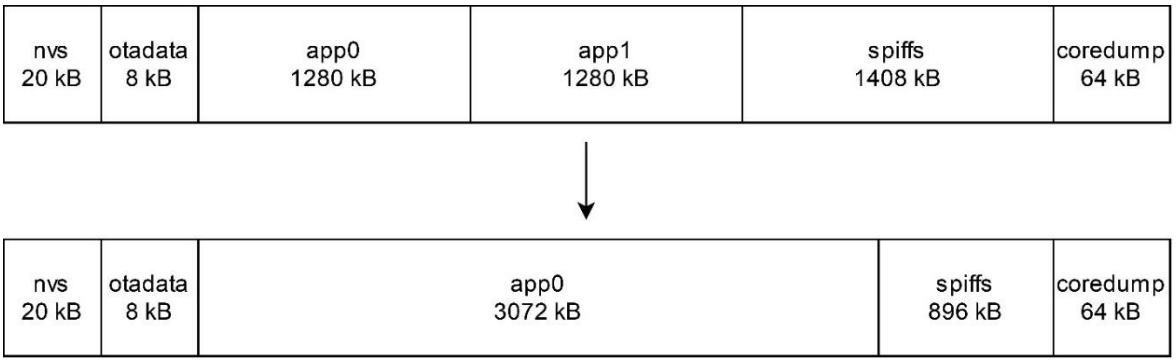


Рисунок 3.4. – Розділи SPIFFS та їх розмір у стандартному форматі та після форматування

Щодо функцій розділів, NVS (Non-Volatile Storage) зберігає необхідні конфігураційні дані, такі як параметри Wi-Fi, налаштування Bluetooth і інші системні параметри. NVS дозволяє зберігати ці дані постійно, навіть після вимкнення живлення. Otadata містить інформацію про версію прошивки для OTA оновлень. Otadata зберігає необхідні дані для коректного визначення та виконання процедури оновлення. App0 містить виконуючий код програмного забезпечення, який може бути виконаний на ESP32. SPIFFS – це маловитратна файлова система, яка може працювати безпосередньо з Flash пам'яті мікроконтролера ESP32. Вона часто використовується для зберігання веб-сторінок, налаштувань, даних сенсорів тощо. Coredump використовується для зберігання інформації про дампи ядра у випадку критичних помилок або аварійного завершення роботи програми.

Встановлення відбувається шляхом компіляції та завантаженням у флеш пам'ять контролера. Для того щоб веб-сервер працював, а принтер етикеток знав що друкувати, потрібно відповідні файли. Вони окремо від програми завантажуються в SPIFFS і використовуються за потреби.

3.2. Проектування вебсайту

3.2.1. Створення серверної частини вебсайту

Серверна частина є масштабним програмним забезпеченням, розробленим на мові програмування Go, яке включає різноманітні компоненти для забезпечення комплексної функціональності. Серверна частина поділена на кілька основних модулів, кожен з яких відповідає за певний аспект функціональності, наприклад, обробка зображень, управління користувачами, обробка замовлень тощо.

На рис. 3.5 показано основні директорії та файли. Оглядати всі не потрібно, бо деякі з них згенеровані середовищем розробки.

Директорія client містить компільовані файли клієнтської частини, вони надсилаються серверною частиною під час запитів користувачів.

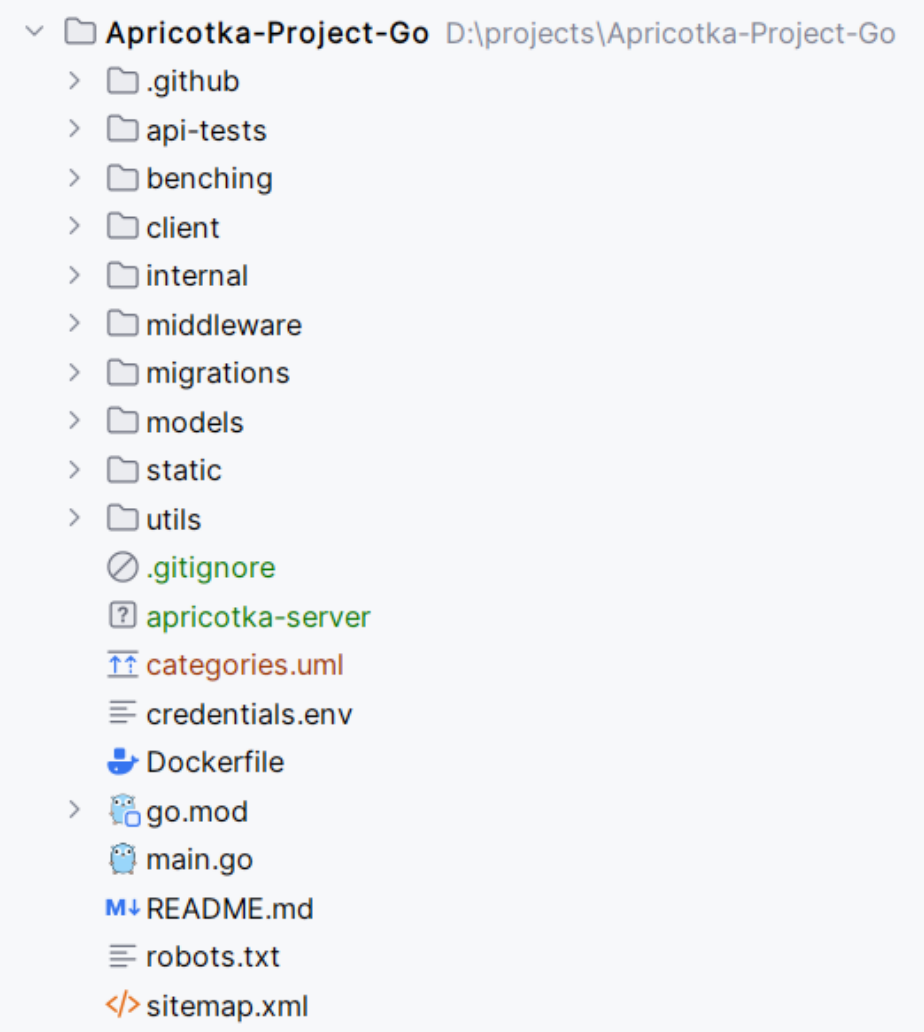


Рисунок 3.5. – Основні директорії та файли серверної частини вебсайту

Файл credentials.env містить конфіденційні дані, такі як ключі доступу та паролі. Забезпечує зберігання конфіденційної інформації в безпечному місці, так як на сервері знаходиться інший файл з більш секретними даними.

Файл go.mod визначає залежності модуля Go та їх версії. Допомагає керувати зовнішніми бібліотеками та забезпечує стабільність проекту. Файл go.sum містить контрольні суми для перевірки залежностей, зазначених у go.mod.

Головний файл програми на мові Go – це main.go, який містить точку входу для виконання додатку. Виконує ініціалізацію та запуск основних компонентів таких як база даних, змінні середовища, роутер та запуск сервера на зазначеному порті.

Директорія internal містить основні модулі проекту, які реалізують різні функціональні компоненти системи. Кожен модуль організований у піддиректорії з відповідними файлами для обробки запитів (handlers), взаємодії з базою даних (repositories) та бізнес-логіки (services). На рис. 3.6 можна побачити структуру даного каталога більш детально.

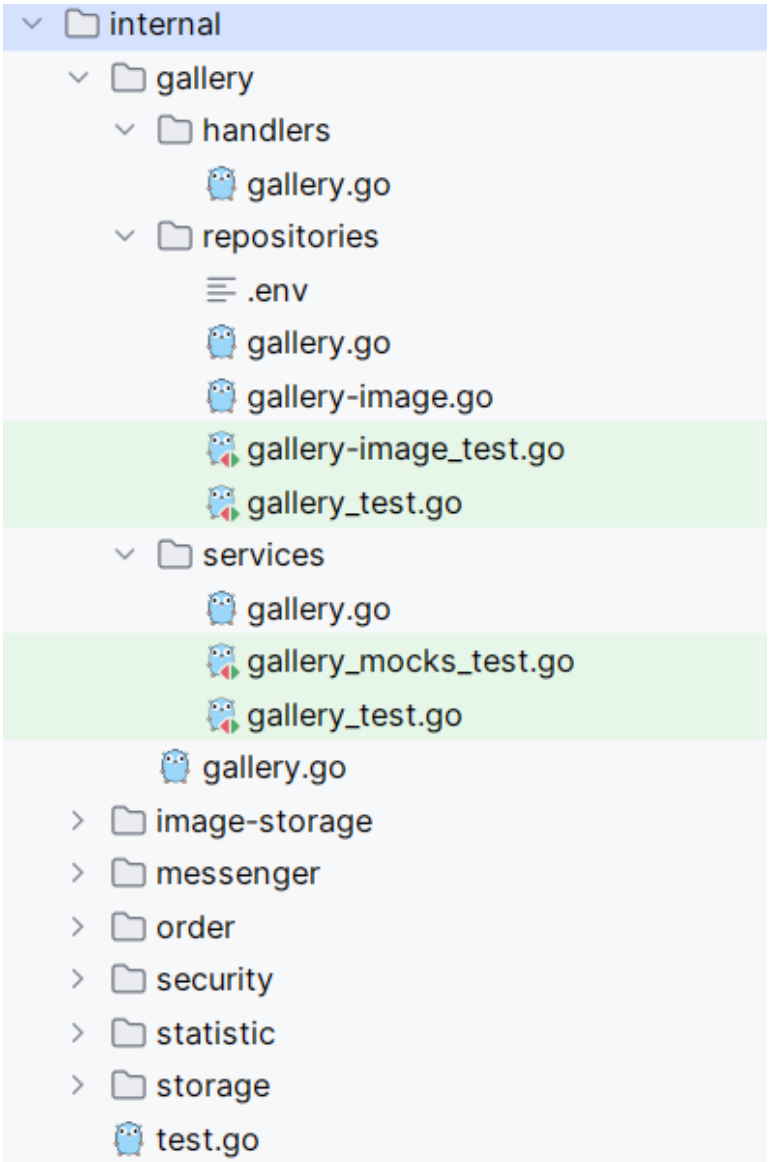


Рисунок 3.6. – Структура директорії internal

Сервіс gallery відповідає за управління галереєю підприємства. Він реалізує операції створення, читання, оновлення та видалення світлин галереї, використовуючи транзакції для забезпечення цілісності даних при завантажені нових зображень. Авторизація для редагування галереї дозволена лише

адміністраторам, забезпечуючи таким чином контроль доступу. Основний файл gallery.go ініціалізує репозиторії, сервіси та хендлери для роботи з галереєю. У handlers/gallery.go визначено логіку обробки HTTP-запитів, пов'язаних з галереєю. У repositories/gallery-image.go забезпено взаємодію з базою даних для збереження та отримання зображень. Тестові файли перевіряють коректність виконання операцій сервісу.

Сервіс image-storage відповідає за зображення товарів.

Сервіс messenger використовується для обміну повідомленнями. Він розсилає електронні листи користувачам. Він використовується для інформування про стан замовлення та інших повідомлень, пов'язаних з надсиланням повідомлень клієнтам.

Сервіс orders відповідає за замовленнями. Він реалізує створення, оновлення, видалення та обробку замовлень. Цей модуль забезпечує облік замовлень, їх статусів та всі операції, пов'язані з управлінням замовленнями.

Сервіс security реалізує авторизацію користувачів на сайті. Він забезпечує безпеку та контроль доступу до різних частин системи, відповідає за реєстрацію, аутентифікацію та управління правами доступу користувачів.

Сервіс statistic відповідає за збереження статистичних даних про збір врожаю. Сам він нічого не вираховує, так як це покладено на клієнтську частину, через велику завантаженість під час розрахунків великих обсягів інформації від різних людей.

Сервіс storage відповідає за товарами на складі. Він надає можливість додавати товари, розподіляти їх по групах, обробляти різні ціни приходу.

Окрім основних елементів, які я іменував сервісами, також є допоміжні, такі як проміжне програмне забезпечення, моделі, файли міграції, утиліти. Проміжне програмне забезпечення надважливе у даній роботі, так відповідає за десеріалізацію користувачів різних рівнів доступу при кожному запиті.

Алгоритм дії перевірки звичайного користувача у middleware/deserialize-user.go представлений на рис. 3.7.

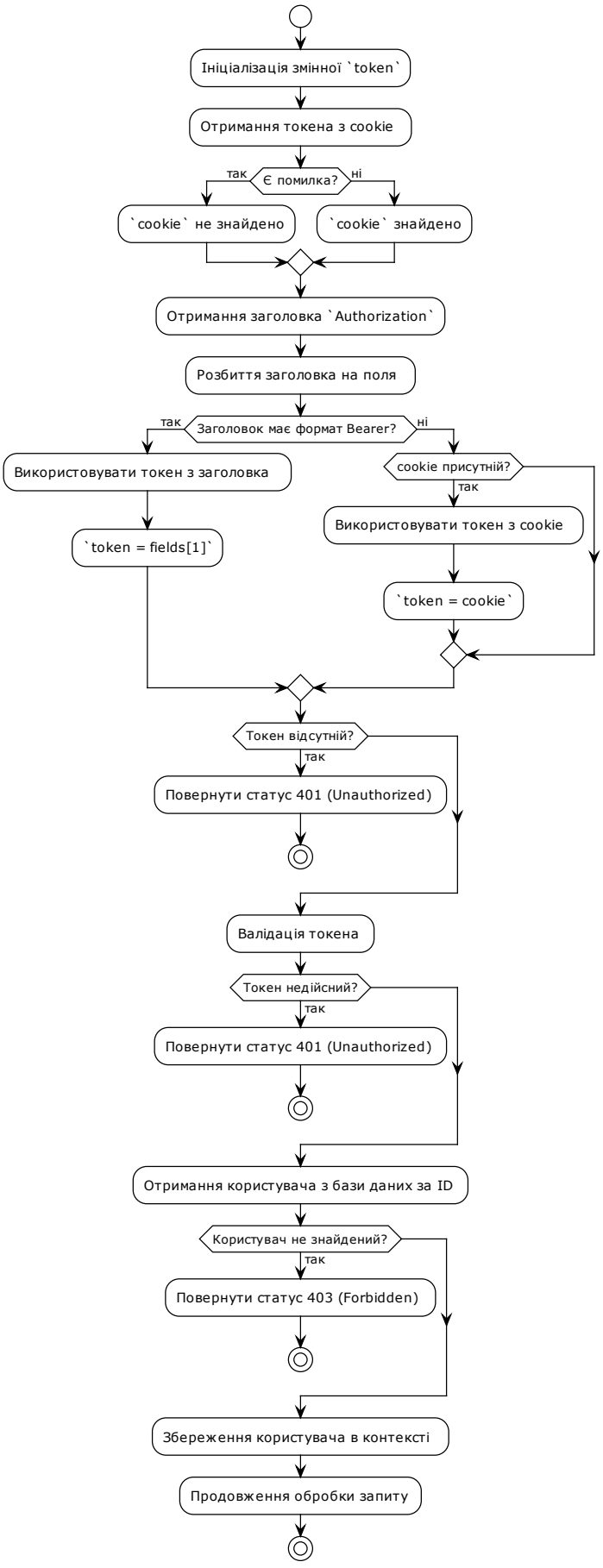


Рисунок 3.7. – Алгоритм дії перевірки звичайного користувача

Як можна побачити, користувач може авторизуватися як через кукі так і через заголовок, але якщо під час перевірки хоч щось піде не так, користувач не зможе пройти далі. Відмінність перевірки на адміністратора лише у значенні поля роль користувача.

Щодо файлів міграції. Вони потрібні для застосування початкової міграції, що створює необхідні таблиці та інші об'єкти бази даних. Забезпечує налаштування бази даних для роботи з додатком. А також для відкату початкової міграції, що видаляє створені таблиці та інші об'єкти бази даних. Використовується для відновлення попереднього стану бази даних. За це відповідають файли з форматом .up.sql та .down.sql відповідно. На рис. 3.8 можна побачити приклад частини вмісту даних файлів.

```
CREATE TABLE IF NOT EXISTS product_images
(
    id bigint not null primary key,
    product_id bigint
        constraint fk_products_product_images
            references products
            on update cascade on delete set null,
    image_file text
);
alter table product_images
    add constraint product_images_products_fk
        foreign key (product_id) references products (id);
alter table product_images
    alter column id add generated by default as identity;
insert into product_images (id,product_id,image_file) values (default,1,'https://live.staticflickr.com/65535/5211920641
insert into product_images (id,product_id,image_file) values (default,1,'https://live.staticflickr.com/65535/5211917928
insert into product_images (id,product_id,image_file) values (default,2,'https://live.staticflickr.com/65535/5211814862
```

Рисунок 3.8. – Частина вмісту файла міграції БД

Як можна побачити, це звичайні sql команди. Директорія models містить визначення моделей даних, які використовуються у проекті. Звичайними моделями, які мають значення тільки на рівні серверної частини є available.go, email-details.go, setup.go. Щодо сутностей з піддиректорії entities, то вони напряму пов'язані з таблицями бази даних і є типами даних, які представляють записи таблиць, для легшої маніпуляції даними.

Остання згадана частина структури проекту є директорія utils. Вона містить всього 2 файли які відповідають за генерацію токена автентифікації та авторизацію через google.

3.2.2. Написання програмного коду клієнтської частини вебсайту

Через велику потребу в зручному маніпулюванні даними сайту, кількість компонентів проекту є достатньо великою, а їх основний перелік зображено на рис. 3.9.

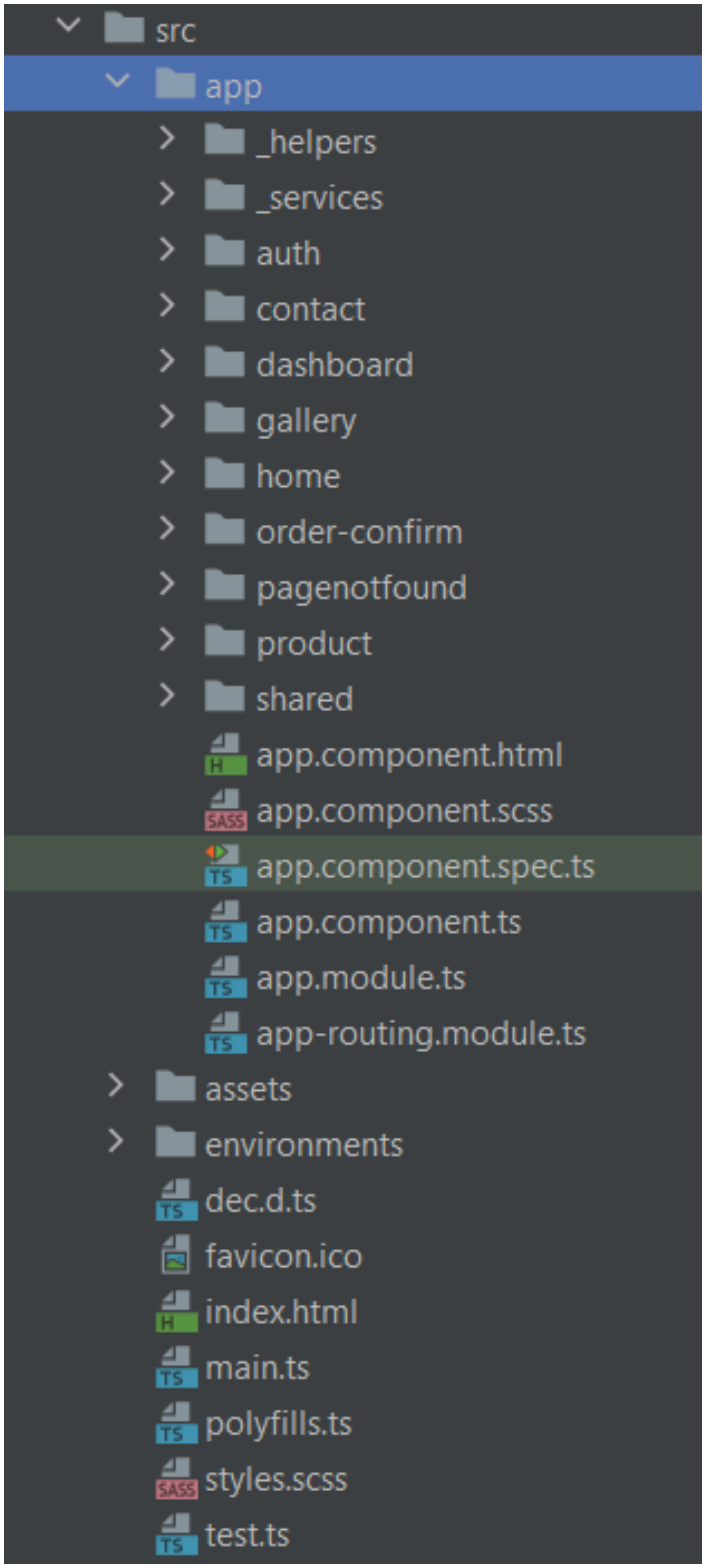


Рисунок 3.9. – Основні директорії та файли клієнтської частини вебсайту

Директорія _helpers містить один файл auth.interceptor.ts. У ньому реалізовано перехоплювач HTTP-запитів, який перехоплює всі HTTP-запити, що виходять із програми Angular. Потім, він отримує токен авторизації зі сховища токенів TokenStorageService. Якщо токен існує, перехоплювач клонує оригінальний запит і додає до його заголовків ключ Authorization зі значенням Bearer <token>. Змінений запит з токеном відправляється на сервер.

У папці app/_services містяться сервіси, які використовуються в різних частинах програми для виконання певних функцій та взаємодії з бекендом. Розглянемо кожен із сервісів детальніше. api.service.ts є основним сервісом для взаємодії з API бекенду. Відповідає за надсилання запитів на сервер (GET, POST, PUT, DELETE) та обробку відповідей. Має велику кількість методів. auth-guard.service.ts реалізує захист маршрутів, що вимагають авторизації. Перевіряє, чи користувач авторизований, і перенаправляє на сторінку входу, якщо ні. Сервіс admin-auth-guard.service.ts реалізує захист маршрутів адміністративної частини програми. Він перевіряє, чи користувач авторизований та чи має роль адміністратора. Якщо ні, перенаправляє на головну сторінку. Сервіс auth.service.ts призначений для авторизації та реєстрації користувачів. Має два методи, перший з яких login, який відправляє запит на вхід з email та паролем, а другий – register, який відправляє запит на реєстрацію нового користувача. Сервіс cart.service.ts призначений для роботи з кошиком покупок. Зберігає товари в кошику в локальному сховищі браузера, розраховує загальну суму замовлення, надає методи для додавання, видалення та зміни кількості товарів. Файл token-storage.service.ts – сервіс для зберігання токену авторизації та інформації про користувача в локальному сховищі браузера. Файл user.service.ts – сервіс для отримання даних користувача з бекенду. Містить методи для отримання публічного контенту, а також контенту, доступного лише користувачам, модераторам та адміністраторам. Найцікавішим є сервіс novaposhta.service.ts для взаємодії з API Нової Пошти. Використовується для пошуку населених пунктів при оформленні замовлення.

Директорія app/auth – це модуль, який відповідає за автентифікацію та реєстрацію користувачів у фронтенд-додатку Apricotka. Він містить компоненти та маршрутизацію, необхідні для реалізації процесу входу, реєстрації та відображення інформації про успішність цих операцій. Також використовуються сервіси з app/_services. Всього є 3 компоненти. Компонент login відповідає за форму входу користувача. Обробляє дані, введені користувачем (email та пароль), та відправляє їх на сервер для автентифікації за допомогою AuthService. Зберігає токен та дані користувача у TokenStorageService після успішного входу. Компонент signup відповідає за форму реєстрації нового користувача. Обробляє дані, введені користувачем (ім’я, email, пароль), та відправляє їх на сервер для реєстрації за допомогою AuthService. Останній компонент info відображає інформацію про успішність входу або реєстрації користувача, отримує статус входу через параметри маршруту та виводить відповідне повідомлення. Маршрутизація auth-routing має всього 3 маршрути: “/auth/login”, “/auth/register”, “/auth/info” – кожен з яких веде до однойменного компоненту.

Директорія contact в цьому Angular проекті, стосується модуля ContactModule, який відповідає за відображення контактної інформації та інтерактивної карти на сторінці контактів. Він має маршрут “/contact”.

Директорія dashboard – це розділ вебсайту, призначений для авторизованих користувачів (як звичайних, так і адміністраторів). Він надає інтерфейс для управління особистими даними, замовленнями, а також, у випадку адміністратора, інструменти для керування контентом сайту (категоріями, товарами, замовленнями, працівниками) та перегляду статистики. Компонент dashboard-index відображає список замовлень користувача. Дозволяє переглядати деталі кожного замовлення (товар, кількість, ціну). Показує статус замовлення (обробляється, відправлено тощо). Компонент dashboard-profile містить основну інформацію профілю користувача (ім’я, електронна пошта), та надає можливість редагувати ім’я

користувача. Компонент dashboard-admin-categories (лише для адміністратора) відображає таблицю категорій товарів. Дозволяє додавати, редагувати та видаляти категорії. Має функцію фільтрації категорій за назвою. Компонент dashboard-admin-products (лише для адміністратора) відображає таблиці з товарами, зображеннями товарів та інформацією про складські надходження. Дозволяє додавати, редагувати та видаляти товари, зображення та надходження. Підтримує завантаження даних про збір врожаю. Має функції сортування та фільтрації. Компонент dashboard-admin-orders (лише для адміністратора) відображає таблицю замовлень та дозволяє переглядати деталі замовлення та змінювати його статус. Компонент dashboard-admin-employees (лише для адміністратора) відображає таблицю працівників (збирачів). Дозволяє додавати, редагувати та видаляти працівників. Має функцію фільтрації за ПІБ. Компонент dashboard-admin-statistics (лише для адміністратора) містить три вкладки: “Чек дня”, “Збирачі” та “Товари”. “Чек дня” дозволяє розрахувати та відобразити таблицю з інформацією про заробіток збирачів за певний день. “Збирачі” відображає продуктивність кожного збирача за вибраний період у вигляді графіків. “Товари” показує статистику зібраного врожаю за вибраний період у вигляді кругової, стовпчикової або лінійної діаграми.

Директорія gallery – це модуль, який відповідає за відображення галереї зображень. Шлях “/gallery” веде до компонента gallery, який керує відображенням галереї.

Модуль home є частиною проекту, призначеною для відображення головної сторінки вебсайту. Він включає кілька файлів, кожен з яких виконує певну функцію. Файл home-routing.module.ts визначає маршрут для головної сторінки, який відображається компонентом HomeComponent. Шаблон головної сторінки міститься у файлі home.component.html, де розташовано карусель зображень з рекламними зображеннями та підписами, а також компонент HomeProductsComponent, який відповідає за відображення списку

товарів. Стилі для елементів головної сторінки, зокрема для каруселі та її вмісту, містяться у файлі home.component.scss. Клас компонента HomeComponent знаходиться у файлі home.component.ts, де здійснюється ініціалізація компонента.

Директорія order-confirm містить компонент OrderConfirmComponent, який відповідає за логіку та відображення сторінки підтвердження замовлення в інтернет-магазині. Компонент отримує дані про товари в кошику від сервісу CartService та відображає їх у вигляді списку. Для кожного товару вказується назва, кількість, ціна та загальна вартість. Також розраховується і відображається загальна сума замовлення. Користувач повинен ввести свій ПІБ, адресу електронної пошти та номер телефону. Якщо користувач авторизований, ці поля можуть бути попередньо заповнені з його профілю. Для введення адреси використовується інтеграція з API “Нової Пошти” для автоматичного доповнення та перевірки адреси. Користувач може вибрати один з двох способів оплати: “Після отримання товару”, “Карткою”. Після заповнення всіх необхідних даних користувач може підтвердити замовлення. Компонент відправляє дані замовлення на сервер за допомогою сервісу ApiService, очищає кошик CartService.removeCart() та відображає результат оформлення замовлення (успішно створено або сталася помилка). Компонент використовує Angular Forms для обробки форм введення даних. Для відображення списку товарів та інших елементів інтерфейсу використовуються компоненти Angular Material: MatListModule, MatFormFieldModule, MatInputModule, MatRadioModule та інші.

Компонент pagenotfound відповідає за відображення сторінки помилки 404, коли користувач намагається отримати доступ до неіснуючого маршруту в додатку.

Модуль product складається з 2 компонентів: product-list, product-details. Компонент ProductListComponent відповідає за відображення списку продуктів, отриманих з бекенду. Він надає функціональність пошуку товарів,

включаючи розширений пошук із фільтрами за ціною. Кожен товар у списку представлений у вигляді картки з зображенням, назвою, ціною та кнопками “Детальніше” та “Замовити”. Крім того, компонент реалізує механізм додавання товарів у кошик. Компонент ProductDetailsComponent відображає детальну інформацію про конкретний продукт. Він отримує дані про продукт з бекенду на основі ідентифікатора продукту, вказаного в URL. Компонент показує галерею зображень продукту, його назву, опис, ціну та наявну кількість.

Модуль product використовує сервіси ApiService та CartService для отримання даних про продукти та управління кошиком покупок відповідно. Він також використовує Angular Material для створення елементів інтерфейсу користувача, таких як картки, кнопки, діалоги та інші. NgxSlider використовується для реалізації повзунків у фільтрах розширеного пошуку. Крім того, модуль product використовує бібліотеку ng-gallery для відображення галереї зображень продукту в компоненті ProductDetailsComponent. Також є підтримка анімації та завантаження зображень у лінивому режимі.

Директорія shared містить компоненти, модулі та інші ресурси, які використовуються в різних частинах програми, забезпечуючи повторне використання коду та модульність. У папці components знаходяться спільні візуальні елементи, що використовуються в усьому додатку. HeaderComponent відповідає за верхню частину сторінки, відображаючи логотип, меню навігації та елементи керування, такі як авторизація та кошик покупок. FooterComponent містить нижній колонтитул сайту з інформацією про авторські права та посиланнями. BaseLayoutComponent слугує основним макетом для сторінок програми, визначаючи загальну структуру та включаючи такі компоненти, як HeaderComponent та FooterComponent. LoaderComponent відображає індикатор завантаження під час отримання даних. SidenavComponent надає бічну панель навігації для компактних пристроїв, дублюючи посилання з HeaderComponent.

ShoppingCartComponent реалізує спливаюче вікно кошика покупок, дозволяючи користувачам переглядати та керувати своїми замовленнями.

Директорія data містить статичні дані, що використовуються в додатку. Файл menus.ts визначає структуру меню навігації, що включає шляхи та назви елементів меню. Файли product.ts та products.ts містять класи та функції для отримання даних про товари з API, включаючи деталі окремих товарів та списки товарів.

Директорія interfaces визначає інтерфейси TypeScript, які описують структуру даних, що використовуються у програмі. Ці інтерфейси забезпечують узгодженість даних у різних компонентах та сервісах. Наприклад, інтерфейс Product визначає структуру об’єкта товару, що включає ідентифікатор, назву, опис, ціну та інші атрибути. На рис. 3.10 можна побачити весь перелік інтерфейсів.

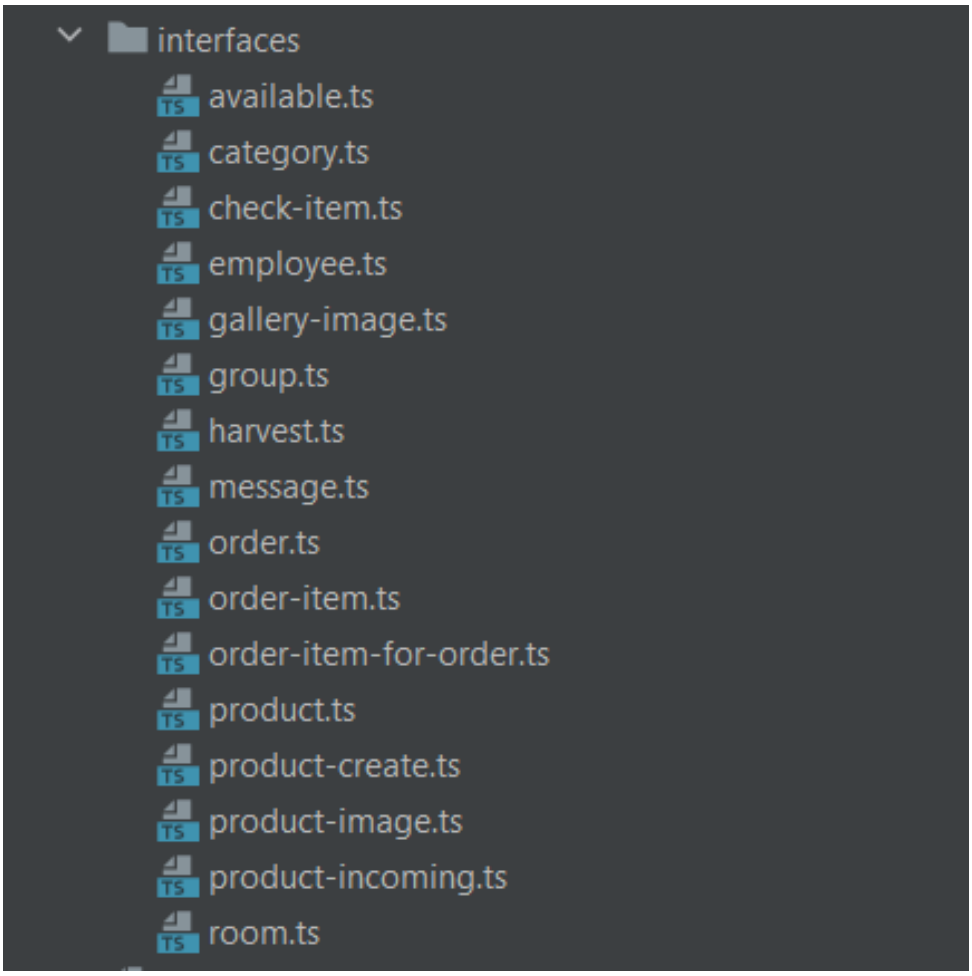


Рисунок 3.10. – Перелік інтерфейсів клієнтської частини вебсайту

3.2.3. Опис програмної логіки роботи сховища даних

База даних представлена у додатку 2. Вона складається з 11 таблиць, призначених для зберігання інформації, пов'язаної з галереєю зображень, категоріями продуктів, продуктами, замовленнями, користувачами та іншими пов'язаними даними.

Таблиця gallery зберігає інформацію про галереї зображень, включаючи ідентифікатор галереї (id) та її назву (title). Таблиця gallery_images містить дані про зображення в галереях, включаючи ідентифікатор зображення (id), ідентифікатор галереї, до якої воно належить (gallery_id), та шлях до файлу зображення (file). Таблиця categories зберігає інформацію про категорії продуктів, такі як "Абрикоси", з унікальним ідентифікатором (id) та назвою (name). Таблиця employees містить дані про співробітників, включаючи їх ідентифікатор (id) та унікальне ім'я (name). Таблиця products зберігає інформацію про продукти, включаючи ідентифікатор продукту (id), ідентифікатор категорії (category_id), до якої він належить, опис (info), назву (name) та ціну (price). Таблиця product_images містить дані про зображення продуктів, включаючи ідентифікатор зображення (id), ідентифікатор продукту (product_id), до якого воно належить, та шлях до файлу зображення (image_file). Таблиця users зберігає інформацію про користувачів, включаючи їх ідентифікатор (id), ім'я (name), унікальну адресу електронної пошти (email), пароль (password), роль (role), статус підтвердження (verified), постачальника (provider), дату створення (created_at) та дату оновлення (updated_at). Таблиця incoming_product_details зберігає деталі про надходження продуктів, включаючи ідентифікатор (id), ідентифікатор продукту (product_id), постачальника (supplier), початкову ціну (initial_price), кількість (quantity) та дату і час (timestamp). Таблиця orders зберігає інформацію про замовлення, включаючи ідентифікатор замовлення (id), тип (type), статус (status), дату і час (timestamp), ідентифікатор користувача (user_id), ім'я користувача (username), адресу електронної пошти (email), номер телефону (phone), адресу доставки

(address), тип оплати (payment_type) та дані про оплату (payment). Таблиця order_items містить дані про товари в замовленні, включаючи ідентифікатор (id), ідентифікатор продукту (product_id), початкову ціну (initial_price), кількість (quantity), ціну (price) та ідентифікатор замовлення (order_id), до якого він належить. Таблиця harvests зберігає інформацію про збори врожаю, включаючи ідентифікатор (id), дату і час (timestamp), масу (mass), ідентифікатор співробітника (employee_id), який здійснив збір, та ідентифікатор продукту (product_id).

Зв'язок gallery_images з gallery встановлює приналежність зображень до певних галерей. Кожне зображення може бути пов'язане лише з однією галереєю, але галерея може містити багато зображень. Якщо галерея видаляється, пов'язані зображення також видаляються. Зв'язок products з categories визначає, до якої категорії належить кожен продукт. Кожен продукт може бути пов'язаний лише з однією категорією, але категорія може містити багато продуктів. Якщо категорія видаляється, пов'язані з нею продукти не видаляються, але їх значення category_id встановлюється в NULL. Зв'язок product_images з products пов'язує зображення з відповідними продуктами. Кожне зображення може бути пов'язане лише з одним продуктом, але продукт може мати багато зображень. Якщо продукт видаляється, пов'язані зображення також видаляються. Зв'язок incoming_product_details з products встановлює зв'язок між деталями надходження товару та самим продуктом. Кожен запис деталей надходження товару може бути пов'язаний лише з одним продуктом. Якщо продукт видаляється, пов'язані з ним деталі надходження товару також видаляються. Зв'язок order_items з orders визначає, які товари входять до кожного замовлення. Кожен товар може бути пов'язаний лише з одним замовленням. Якщо замовлення видаляється, пов'язані з ним товари також видаляються. Зв'язок harvests з employees та products відстежує, який співробітник зібрав який продукт і в якій кількості. Кожен запис про збір врожаю може бути пов'язаний лише з одним співробітником та одним

продуктом. Якщо співробітник або продукт видаляється, пов’язані записи про збір врожаю не видаляються, але їх значення employee_id або product_id встановлюється в NULL.

3.2.4. Налаштування серверу та запуск вебсайту

Для запуску сайту потрібно спочатку компілювати серверну та клієнтські частини. Виконуємо “ng build” для отримання показаних на рис. 3.11 файлів вебдодатку створених за допомогою фреймворку Angular.

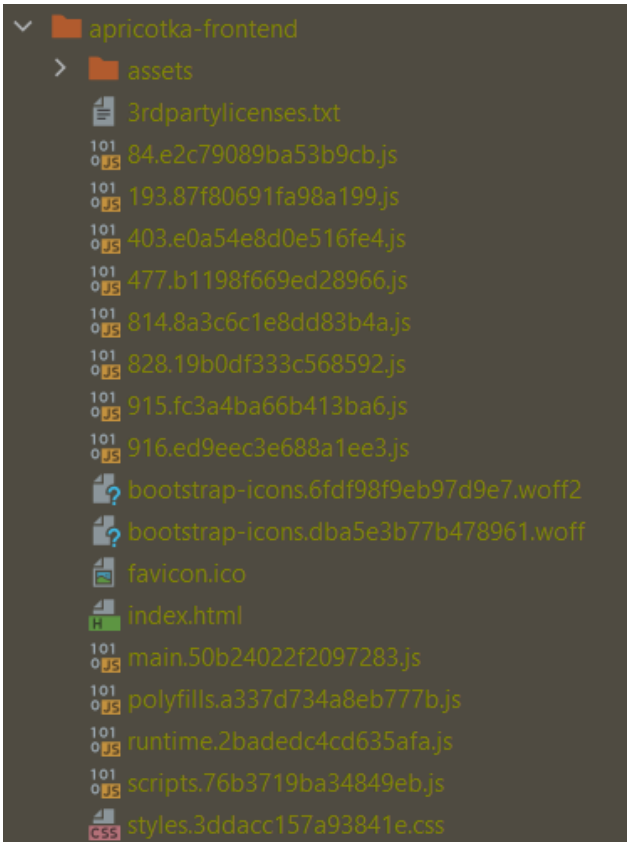


Рисунок 3.11. – Результат компіляції клієнтської частини вебсайту

Далі виконуєм команду зображену на рис. 3.12 і отримуємо один виконуваний файл серверної частини apricotka-server.

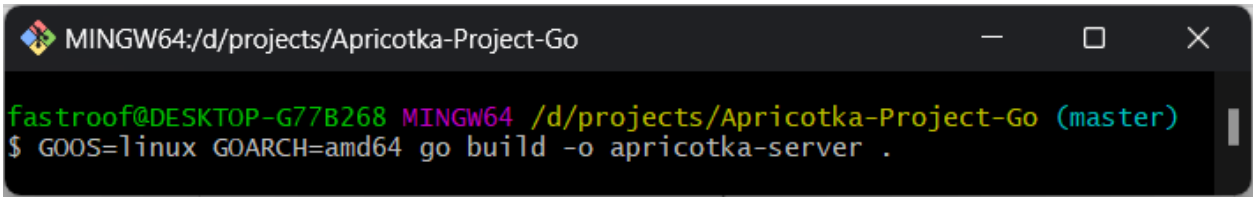


Рисунок 3.12. – Командний рядок з командою для збірки серверної частини

Тепер, потрібно під’єднатися до серверу та завантажити згенеровані файли. А також додати файл змінних середовища, вміст якого є конфіденційний. Також, потрібно провести міграцію бази даних. Демонстрація роботи представлена у розділі 4.

Файл robots.txt представлений на рис. 3.13, дає вказівки пошуковим роботам щодо індексації сайту. Використовується для контролю доступу до різних частин вебсайту.

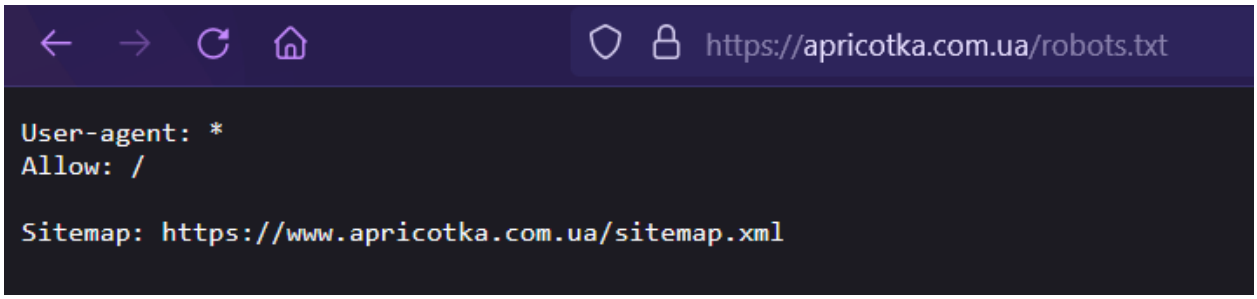


Рисунок 3.13. – Файл robots.txt

Файл sitemap.xml містить карту сайту для пошукових систем, допомагаючи їм ефективно індексувати сторінки. Містить перелік всіх URL сайту з додатковою інформацією. Фрагмент файлу представлений на рис. 3.14.

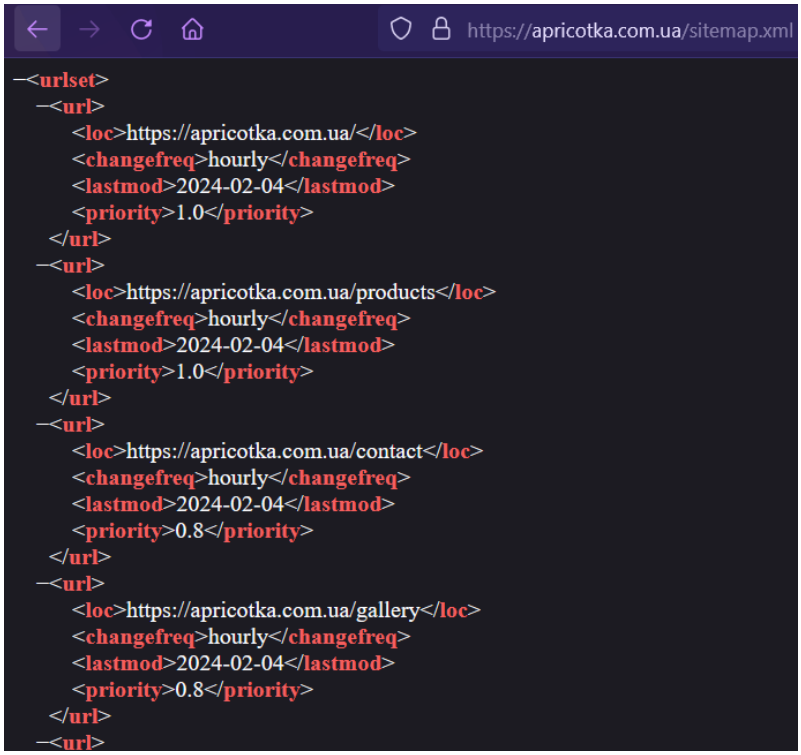


Рисунок 3.14. – Фрагмент файлу sitemap.xml

ВИСНОВОК ДО РОЗДІЛУ 3

Було проведено проектування електричної схеми пристрою, що визначило основні компоненти та їх з’єднання для досягнення бажаних функцій. Також було розроблено конструкцію пристрою, враховуючи ергономіку та функціональні вимоги. Було розроблено програмне забезпечення, яке вбудовується безпосередньо у пристрій. Це програмне забезпечення відповідає за керування пристроєм та виконання основних функцій. Після розробки програмного забезпечення, його було встановлено та налаштовано на пристрої, щоб забезпечити його коректну роботу та сумісність з іншими компонентами системи.

Також виконано проектування сайту, що доповнює функціонал системи. Були розроблені структурні елементи та написаний програмний код. Для зберігання даних сайту було розроблено програмну логіку, яка визначає спосіб їхнього збереження та обробки. Було проведено налаштування веб-серверу та запуск сайту, щоб забезпечити його доступність для користувачів.

Загалом, проведена робота дозволила успішно розробити та впровадити систему з урахуванням всіх вимог та потреб користувачів.

РОЗДІЛ 4
РЕАЛІЗАЦІЯ ДОСЛІДНОГО ЗРАЗКУ ПРИСТРОЮ ДЛЯ ЗБИРАННЯ ДАНИХ ПРО УРОЖАЙ

4.1. Демонстрація роботи пристрою

Вставляємо всі потрібні елементи такі як батарейка для RTC та microSD картка у модуль. На рис. 4.1 можна побачити як виглядає пристрій у середині.

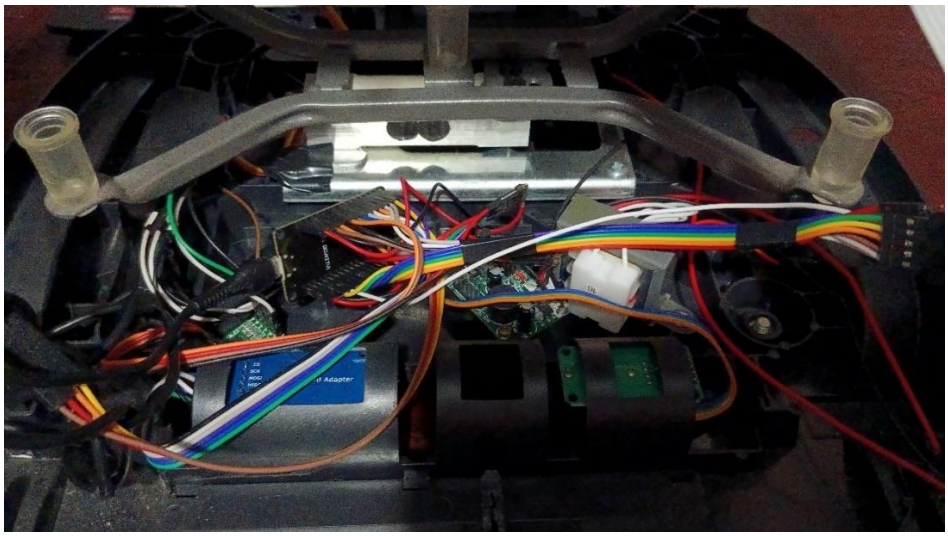


Рисунок 4.1. – Внутрішній вигляд розробленого пристрою

Вмикаємо пристрій та перевіряємо, що не виведено повідомлення про помилку як на рис. 4.2.

Рисунок 4.2. – Початковий стан увімкненого пристрою

Для керування буде використана клавіатура на рис. 4.3, всі кнопки мають різне призначення.

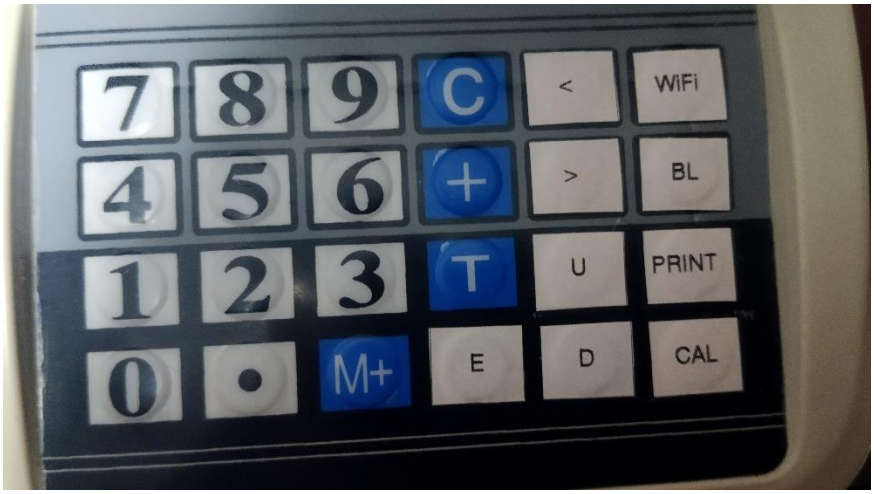


Рисунок 4.3. – Клавіатура пристрою

Для початку відкалібруємо пристрій. Натиснемо на клавішу “CAL” і побачимо напис на екрані зображений на рис. 4.4. Потім за допомогою цифр уводимо відому вагу об’єкта для калібрування у грамах.

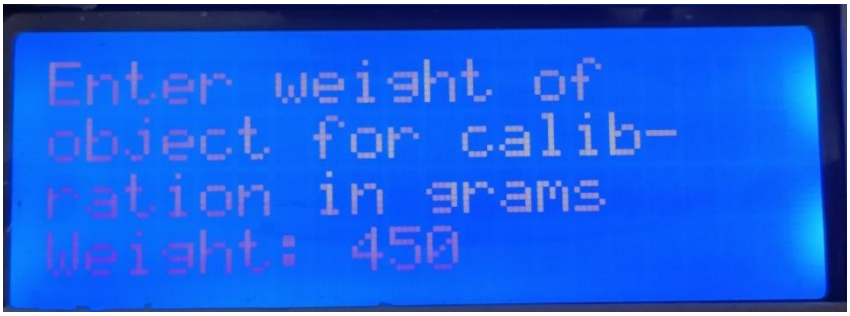


Рисунок 4.4. – Початок калібрування

Натискаємо “>”, та прибираємо об’єкти з вагів. Про це зазначено на рис. 4.5.

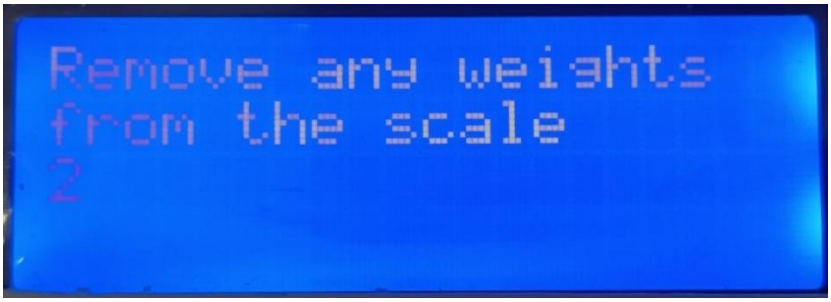


Рисунок 4.5. – Повідомлення про те, що потрібно прибрати об’єкти з вагів

Далі, після того як відбудеться тарування як на рис. 4.6, та потрібно за 10 секунд покласти потрібний об’єкт калібрування.

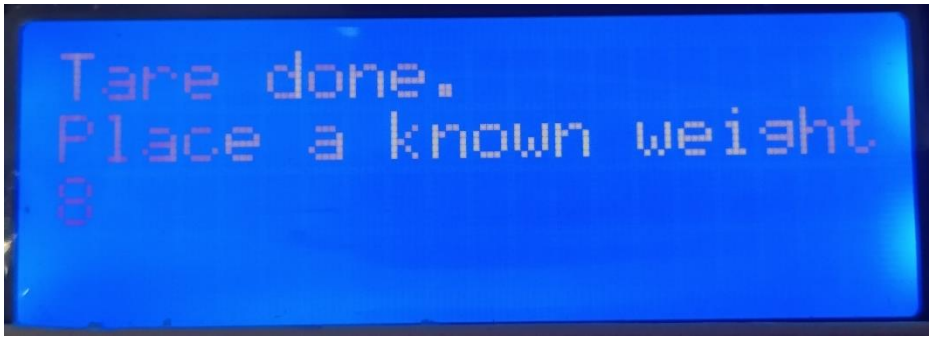


Рисунок 4.6. – Повідомлення про вдале тарування та потребу покласти об’єкт калібрування

Потім, потрібно трішки почекати, поки пройду декілька десятків зчитування значення з тензодатчика та вирахування коефіцієнта калібрування, про це буде написано на екрані як на рис. 4.7.

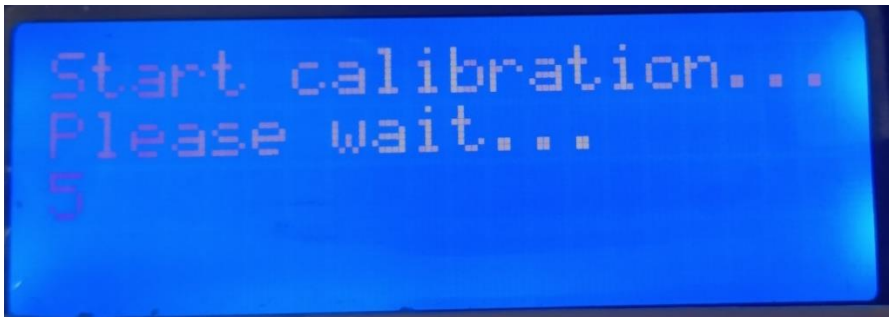


Рисунок 4.7. – Повідомлення про прохання очікувати закінчення калібрування

У кінці, відомий коефіцієнта калібрування буде виведений на екран як на рис. 4.8, а також збережений у внутрішню пам’ять контроллера.

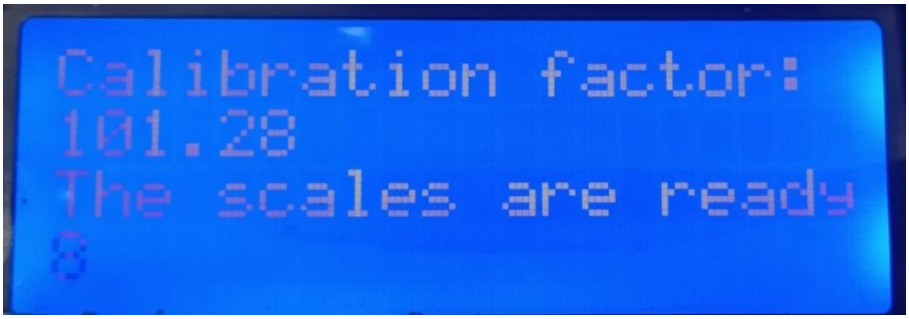


Рисунок 4.8. – Повідомлення про вдале калібрування та результат

Далі можна спробувати перевірити підключення через Bluetooth. Для цього натискаємо на кнопку “BL”, та отримуємо повідомлення як на рис. 4.9.

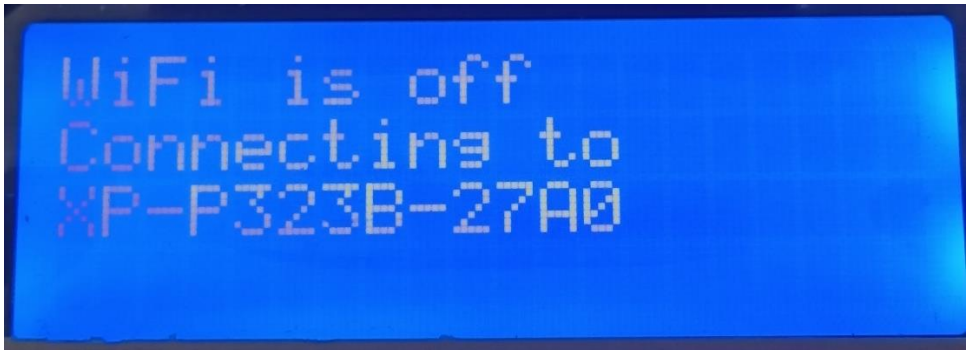


Рисунок 4.9. – Повідомлення про відключення Wi-Fi та початок під’єднання до принтера етикеток

Якщо під’єднання вдале, то буде отримано повідомлення зображене на рис. 4.10.

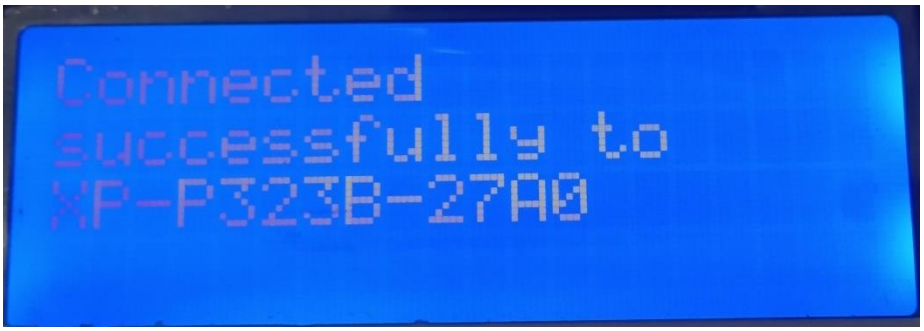


Рисунок 4.10. – Повідомлення про вдале під’єднання до принтера етикеток

Далі спробуємо виконати головну функцію пристрою, а саме додання нового запису про збір врожаю. Для цього натиснемо на кнопку “M+”. Та почнемо вводити номер збирача як на рис. 4.11.

Рисунок 4.11. – Уведення номеру збирача

Натискаємо “>” та вводимо номер товару, який зібраний. Натискаємо “>” та кладемо предмет, який зважується. Пристрій сам виміряє масу. Результат можна побачити на рис. 4.12.

Рисунок 4.12. – Результат уведення даних збору

Натискаємо “>” та маємо вибір між друкувати етикетку чи ні, повідомлення зображене на рис. 4.13.

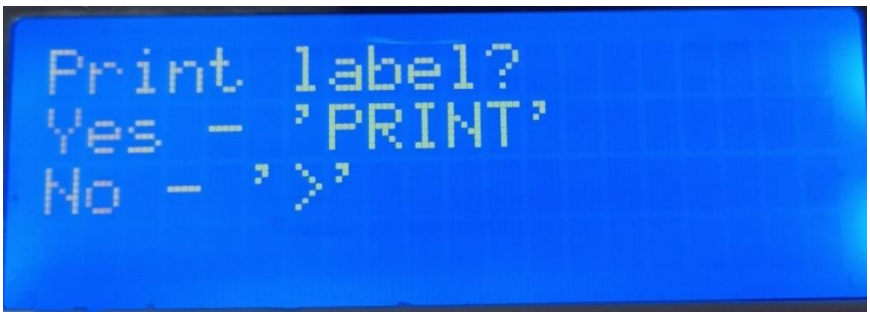


Рисунок 4.13. – Повідомлення про обрання чи друкувати етикетку

Натискаємо кнопку “PRINT” та принтер друкує етикетку як на рис. 4.14.

Рисунок 4.14. – Результат друку етикетки принтером

Тепер, давайте під’єднаємось до Wi-Fi. Для цього натискаємо на кнопку “WiFi” та отримуємо повідомлення на рис. 4.15 про можливість між під’єданням до існуючої мережі та створенням своєї. Обираємо потрібний варіант у залежності від наявності вже існуючої мережі.

Рисунок 4.15. – Повідомлення про вибір типу Wi-Fi

У результаті, якщо не буде проблем, отримуємо повідомлення як на рис. 4.16 про відключення Bluetooth та під’єднання до мережі.

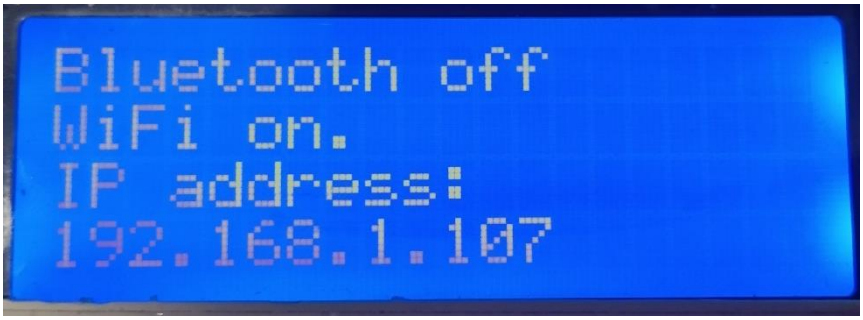


Рисунок 4.16. – Повідомлення про відключення Bluetooth та під’єднання до мережі

Також виводиться IP адреса пристрою за якою можна під’єднатися до вебсайту на якому є багато налаштувань, одна з них зображена на рис. 4.17.

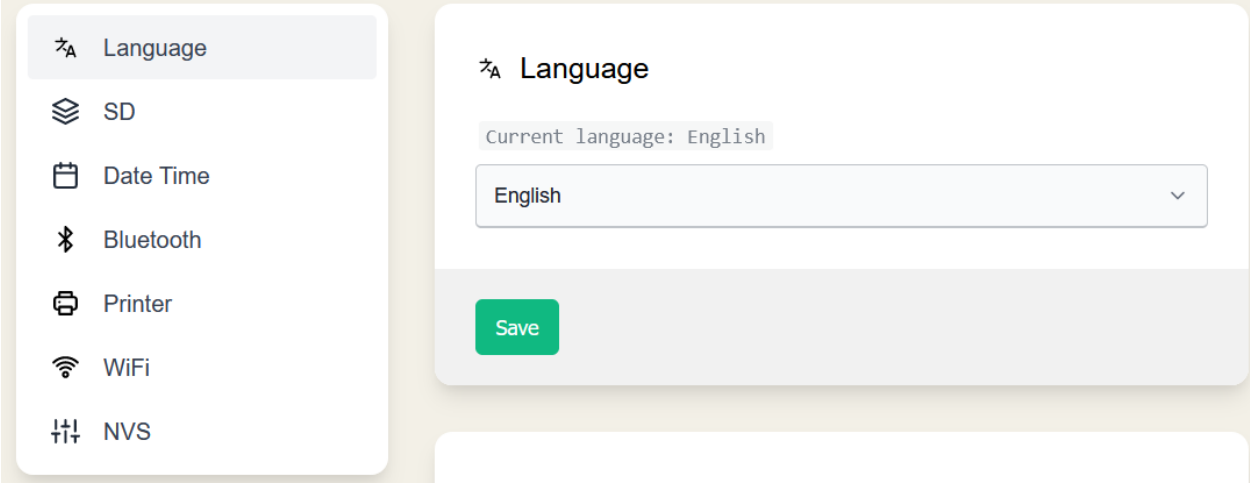


Рисунок 4.17. – Вибір мови інтерфейсу

Також, є можливість переглянути вміст microSD картки без витягування з пристрою. Для цього натискаємо на чорну кнопку на рис. 4.18.

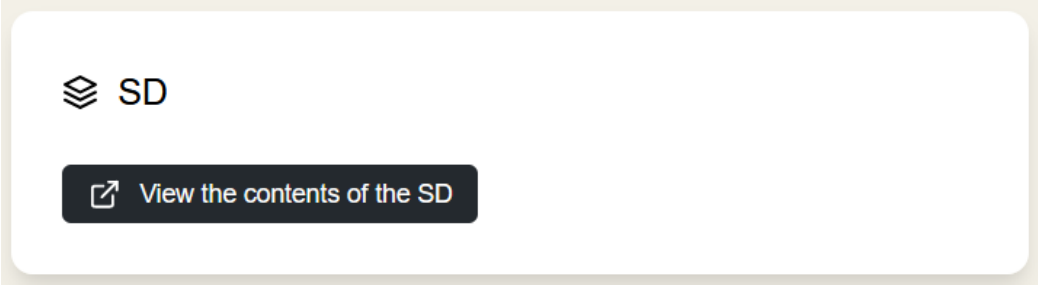


Рисунок 4.18. – Кнопка переходу до меню SD картки

Переходимо до меню SD картки як на рис. 4.19.

Name/Type	Type File/Dir	File Size		
Dir	System Volume Information			
17.01.2024.json	File	201 B	Read	Download
15.01.2024.json	File	135 B	Read	Download
20.01.2024.json	File	202 B	Read	Download
23.04.2024.json	File	83 B	Read	Download
28.05.2024.json	File	245 B	Read	Download

Рисунок 4.19. – Меню SD картки

Натискаємо на “Read” файлу 28.05.2024.json та можемо переглянути його вміст, який зображено на рис. 4.20.

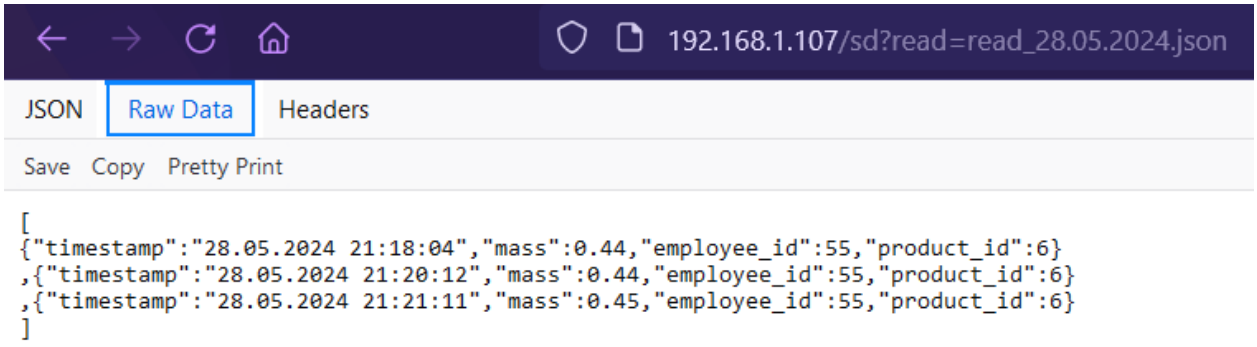


Рисунок 4.20. – Вміст файлу 28.05.2024.json

Наступний розділ налаштувань, який зображений на рис. 4.21 – це значення дати та часу на RTC модулі. Його можна оновити натиснувши на одну із 2 кнопок. Зараз час на модулі не відповідає дійсності, тому потрібно це виправити.

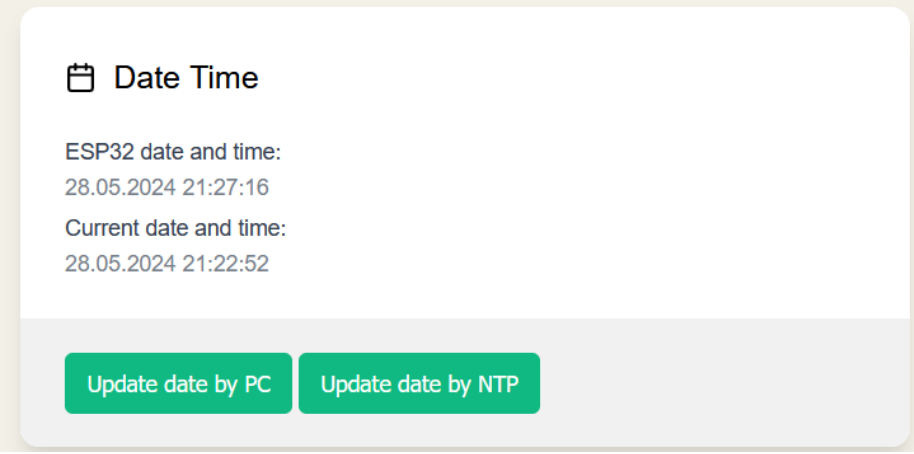


Рисунок 4.21. – Розділ налаштування часу

Натискаємо на “Update date by PC” та отримуємо результат як на рис. 4.22.

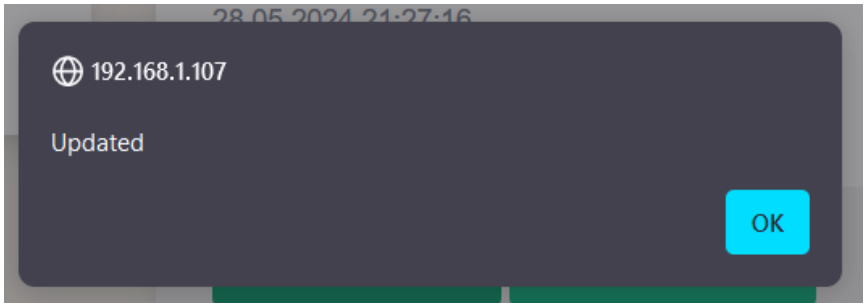


Рисунок 4.22. – Результат натискання на “Update date by PC”

Якщо все пройде без помилок, то час синхронізується і результат можна побачити на рис. 4.23.

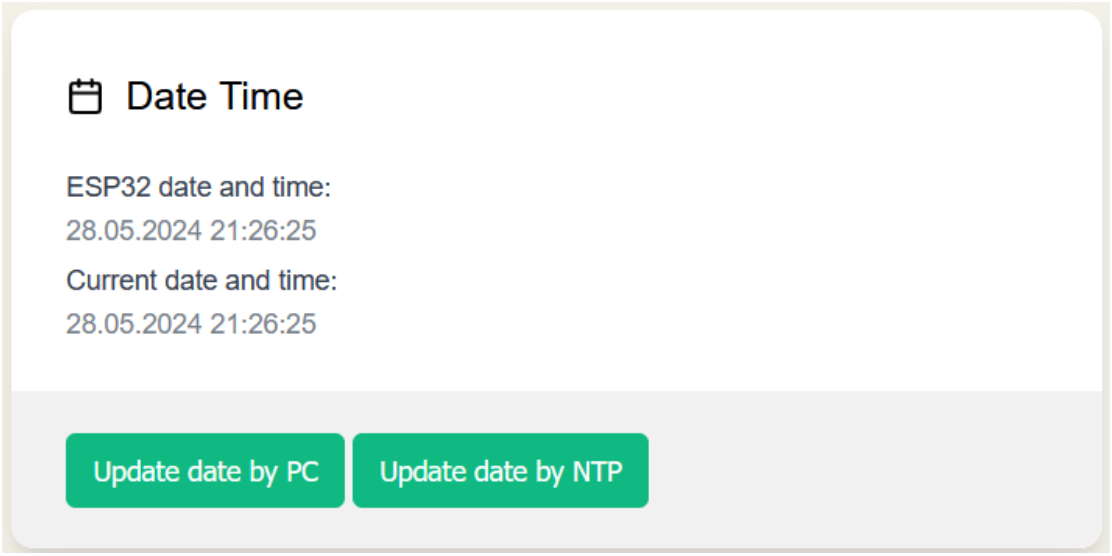


Рисунок 4.23. – Результат синхронізації часу

На рис. 4.24 представлене наступне меню, на якому можна налаштувати назву Bluetooth пристрою та PIN-коду

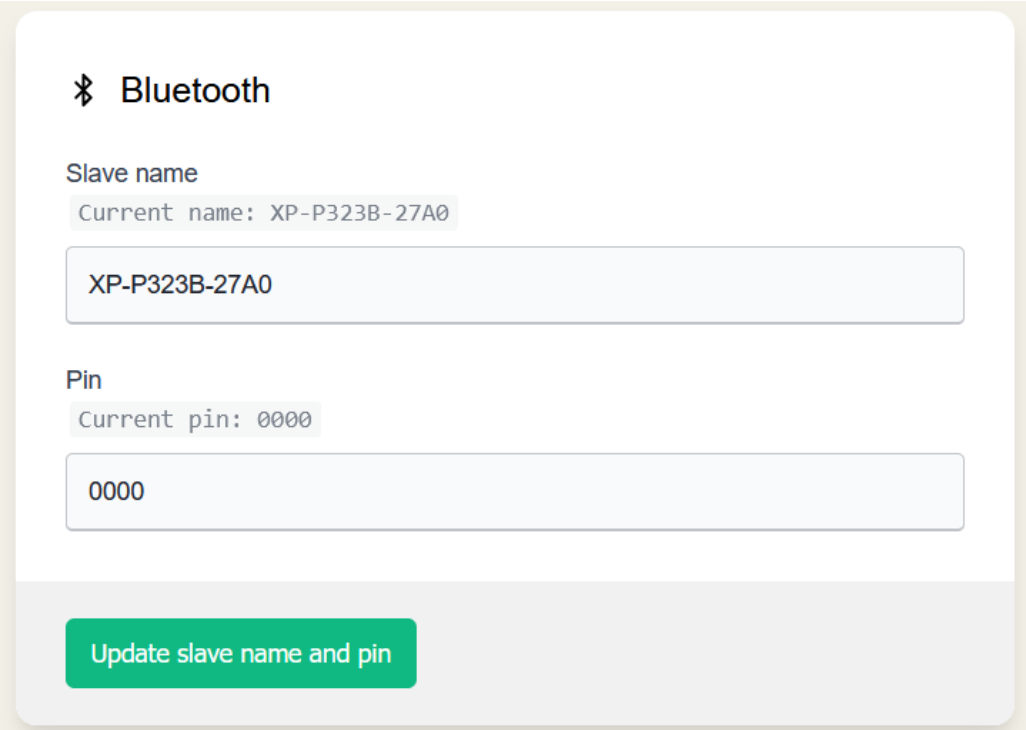


Рисунок 4.24. – Меню налаштування Bluetooth пристрою та PIN-коду

Далі розглянемо рис. 4.25, на якому зображено розділ призначений для керування принтером етикеток. Кнопка “View the current label” дозволяє переглянути поточний код етикетки яка друкується. Текстове поле “Lable code” призначене для введення або редагування коду, який визначає вміст і зовнішній вигляд етикетки.

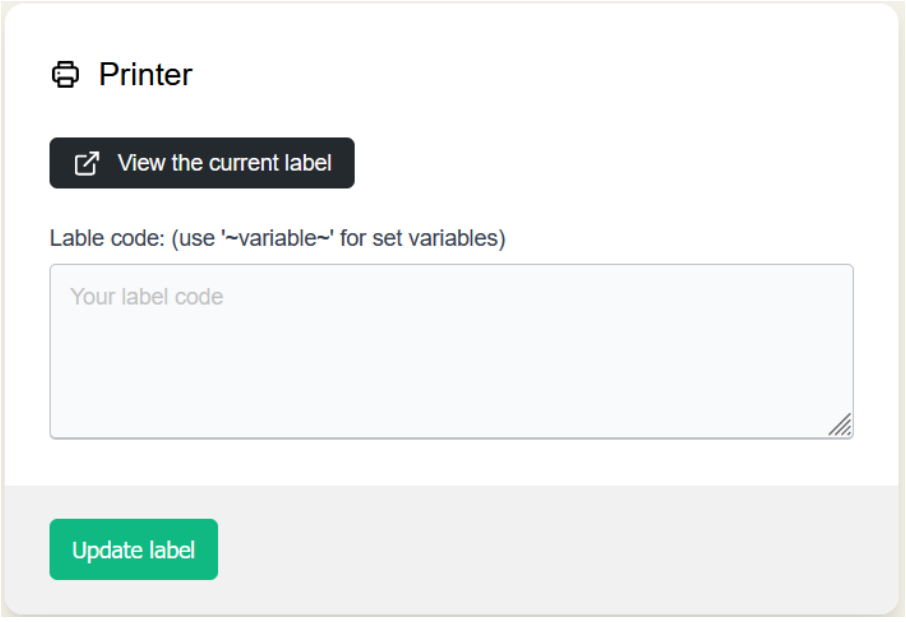


Рисунок 4.25. – Меню керування принтером етикеток

На рис. 4.26 зображено завантажений файл з кодом етикетки, його опрацювати та модернізувати на власний розсуд. Використовується мова програмування Zebra.

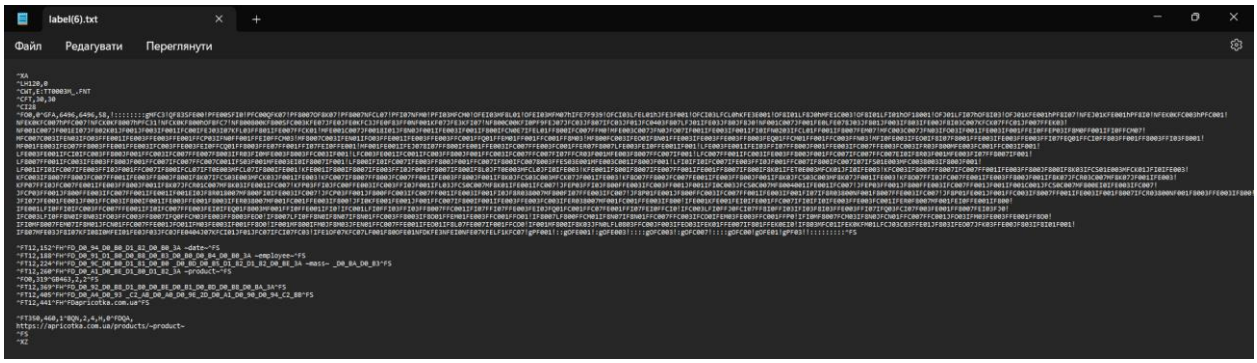


Рисунок 4.26. – Код етикетки

На рис. 4.27 показано інтерфейс налаштування Wi-Fi. Він містить поле для введення назви мережі Wi-Fi, поточне значення – “kahovka_2.4g” та поле для введення пароля до мережі Wi-Fi.

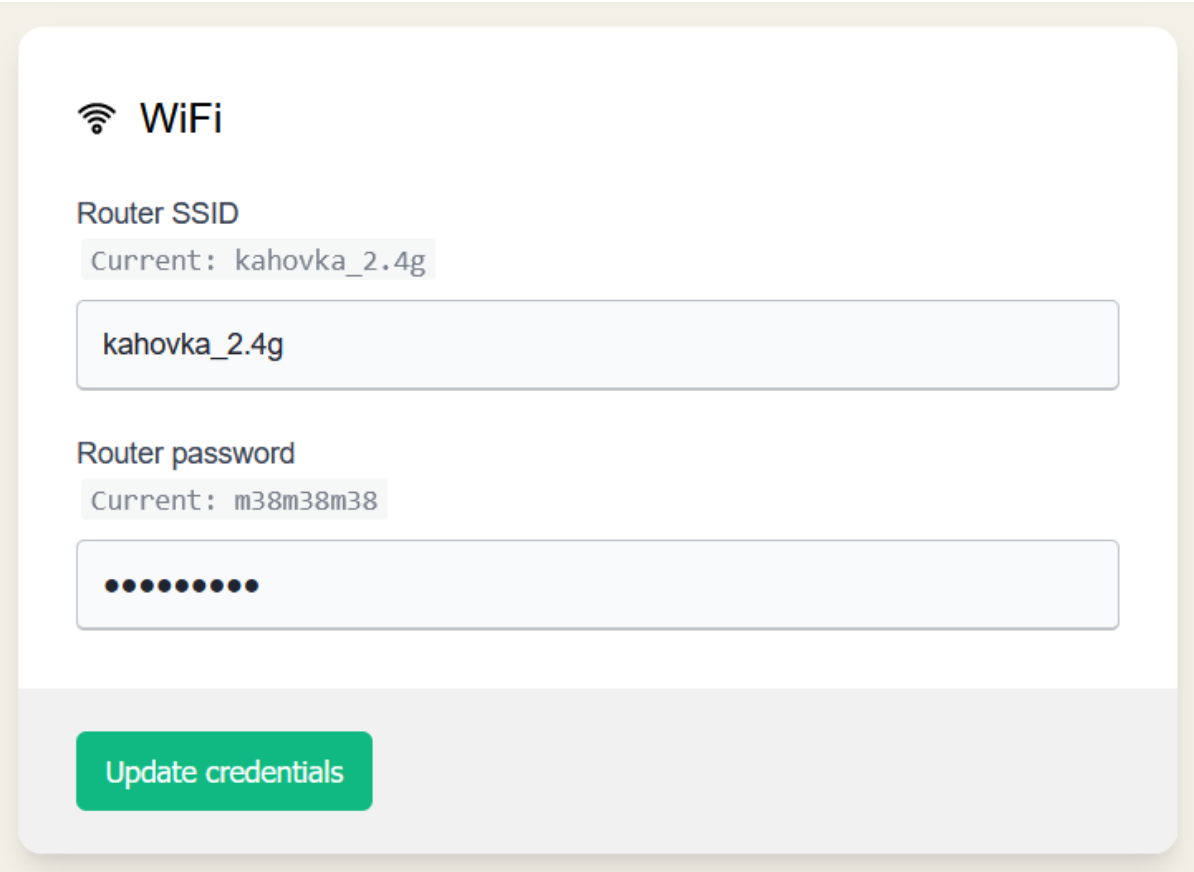


Рисунок 4.27. – Меню керування налаштуваннями Wi-Fi

Останній розділ представлений на рис. 4.28, у ньому можна налаштувати коефіцієнтом калібрування просто увівши його у поле.

Рисунок 4.28. – Меню керування налаштуваннями коефіцієнтом калібрування

4.2. Демонстрація роботи сайту

При початку роботи з сайтом, користувач переходить на головну сторінку, на якій зображено карусель із зображеннями саду, яку можна перемикати, щоб побачити фотографії в різні сезони. Далі на сторінці можна знайти товари, відсортовані за походженням: українські, французькі, американські, канадські та європейські сорти. На рис. 4.29 показано вигляд головної сторінки

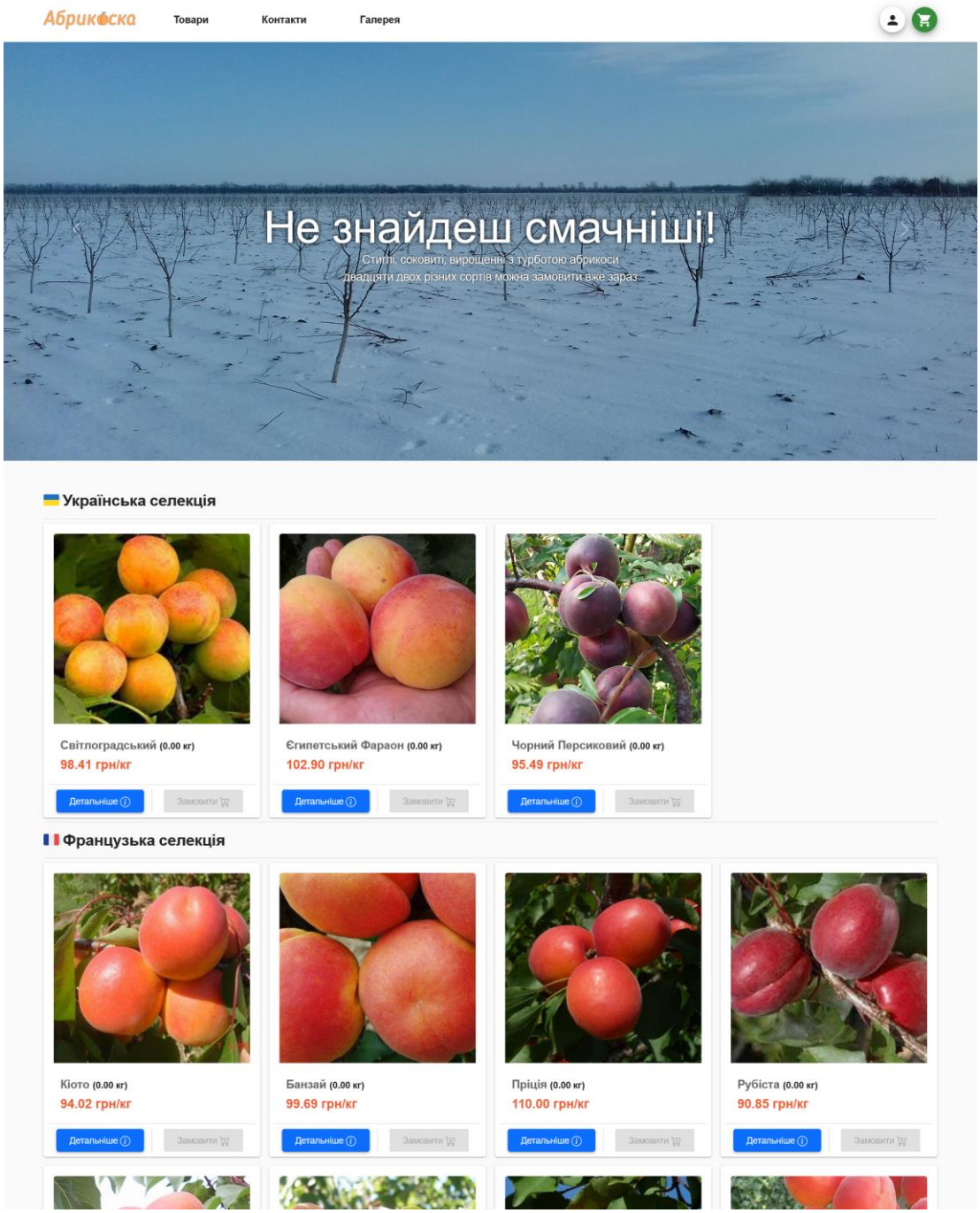


Рисунок 4.29. – Головна сторінка вебсайту

Далі переходимо до сторінки “Товари”, де можна переглянути всі товари сайту без поділу на категорії. Також доступний пошук за назвою товару і розширений пошук за ціною. На рис. 4.30 показано вигляд сторінки “Товари”.

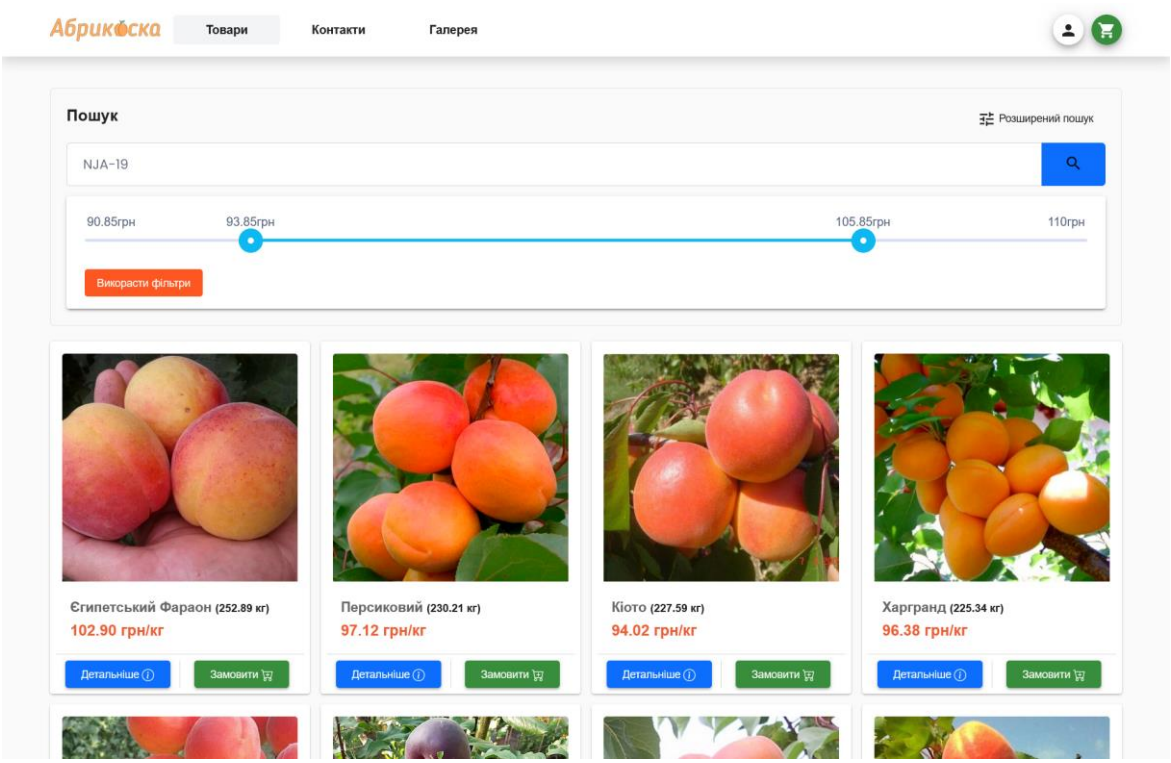


Рисунок 4.30. – Сторінка “Товари”

Перейдемо до розділу “Контакти”. Тут можна дізнатися трохи більше про фермерське господарство, а також знайти контактну інформацію, включаючи номер телефону, електронну пошту та місцезнаходження господарства. На рис. 4.31 показано вигляд сторінки “Контакти”.

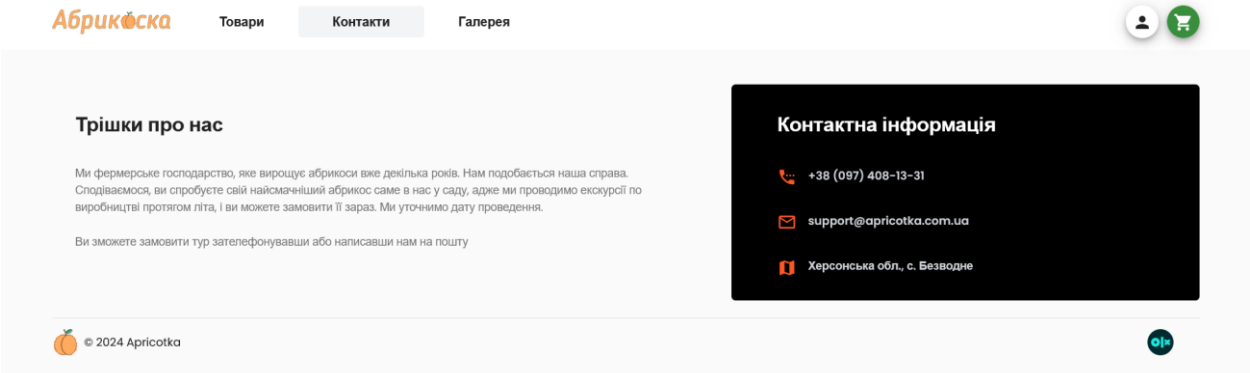


Рисунок 4.31. – Сторінка “ Контакти”

Перейдемо до сторінки “Галерея”. Тут ми бачимо фотографії, що відображають історію саду та господарства. Є можливість фільтрувати зображення за різними категоріями, такими як догляд за садом, дерева, плоди

та цвітіння. Крім того, можна натискати на будь-яке фото, щоб відкрити його у режимі слайд-шоу. На рис. 4.32 представлено вигляд сторінки “Галерея”, а на рис. 4.33 – вигляд слайд шоу галереї.

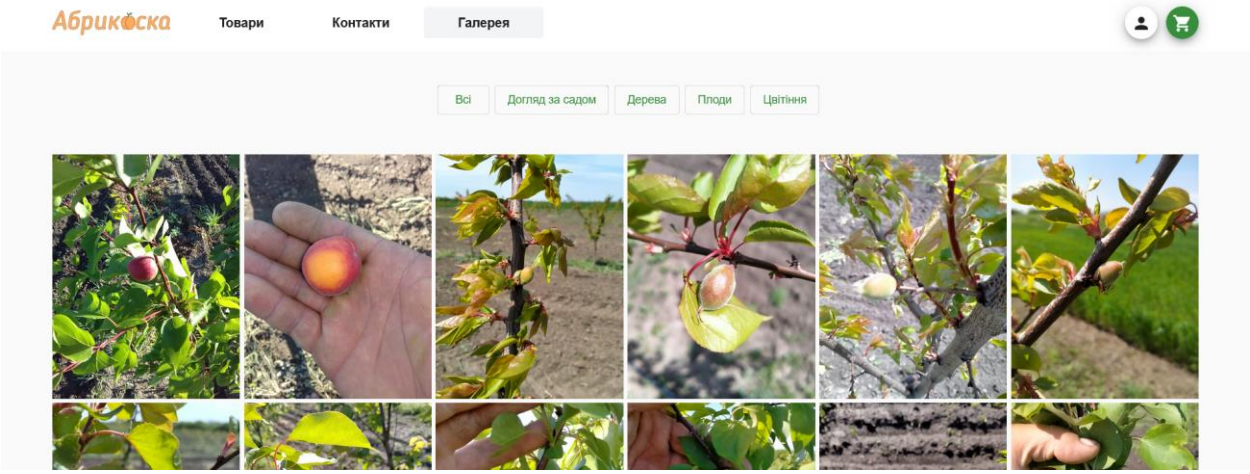


Рисунок 4.32. – Сторінка “Галерея”

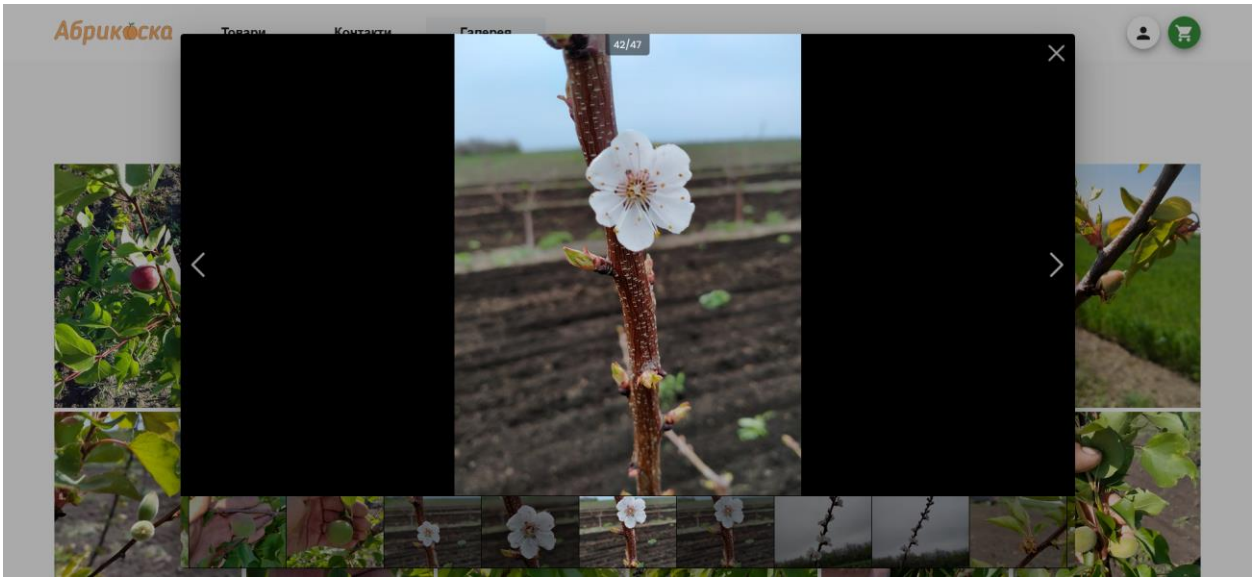


Рисунок 4.33. – Слайд шоу галереї

На головній сторінці або у вкладці “Товари” можна натиснути на будь-який товар, щоб отримати детальну інформацію про нього, переглянути більше фотографій та прочитати опис. На рис. 4.34 представлено вигляд сторінки з детальною інформацією.

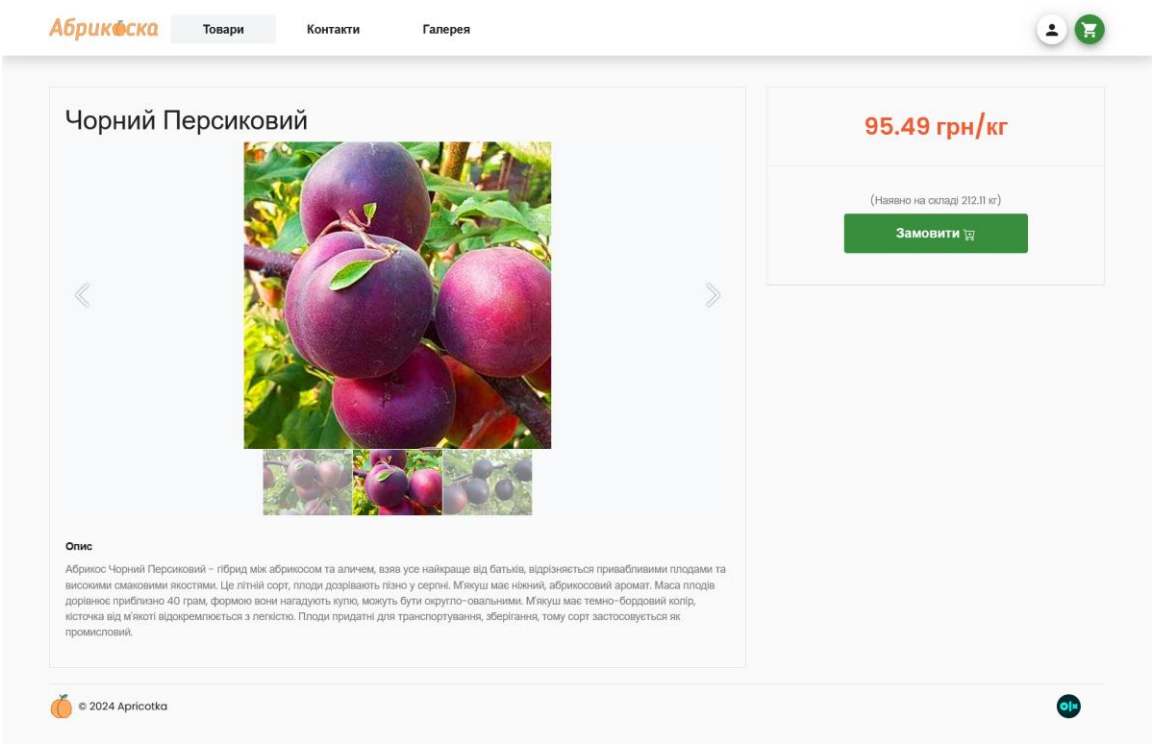


Рисунок 4.34. – Сторінка з детальною інформацією про товар

Давайте перейдемо і зареєструємося як звичайний користувач. Для цього треба натиснути на кнопку “Авторизація”, яка зображена у правому куті на рис. 4.35.

Рисунок 4.35. – Шапка сайту

А далі на напис “Зареєструйтесь”, який можна побачити на рис. 4.36.

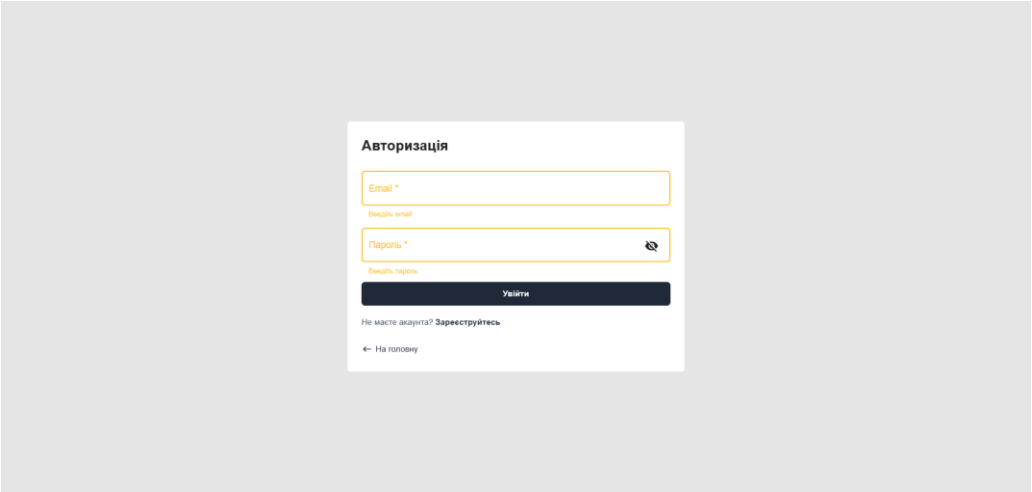


Рисунок 4.36. – Сторінка авторизації

Уводимо данні користувача та натискаємо на кнопку “Зареєструватися” як на рис. 4.37.

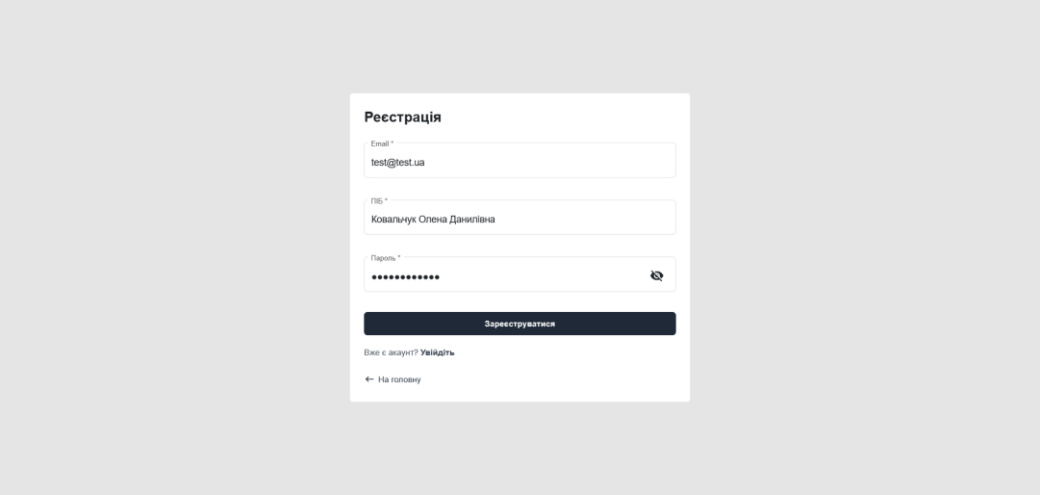


Рисунок 4.37. – Сторінка реєстрації

Отримуємо повідомлення про успішність операції. Його вигляд можна побачити на рис. 4.38.

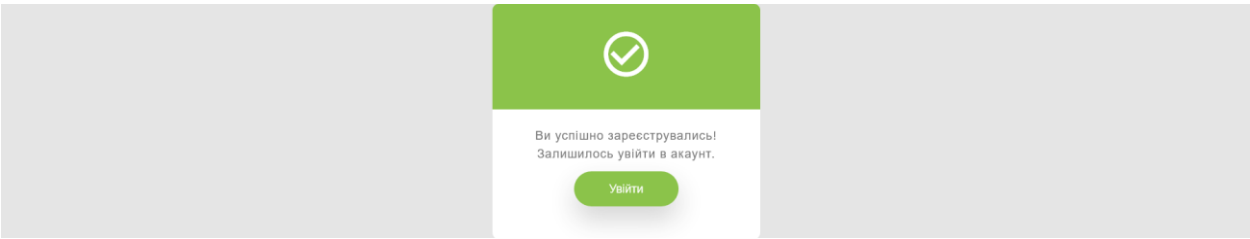


Рисунок 4.38. – Повідомлення про успішність операції реєстрації

Після успішного входу переходимо до особистого кабінету та можемо переглядати статус замовлень та особисті дані. Особисті дані можна редагувати. На рис. 4.39 представлено замовлення, на рис. 4.40 – профіль користувача.

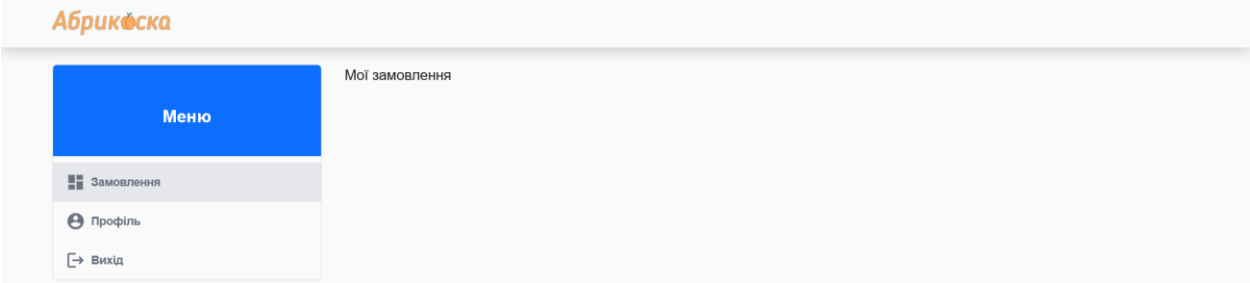


Рисунок 4.39. – Сторінка перегляду власних замовлень

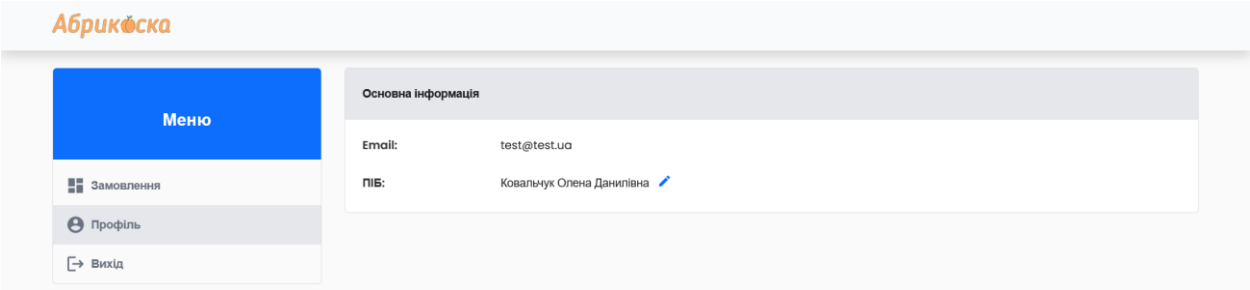


Рисунок 4.40. – Сторінка перегляду та редагування особистої інформації

Давайте продовжимо і авторизуємось під адміністратором. Після успішної авторизації перейдемо в особистий кабінет. На рис. 4.41 багато вкладок доступних адміністратору. У розділі “Керування категоріями” зараз є лише одна категорія – “Абрикоси”, але ми можемо додати нову категорію, наприклад, “Помідори” як на рис. 4.42. Також ми можемо редагувати та видаляти категорії.

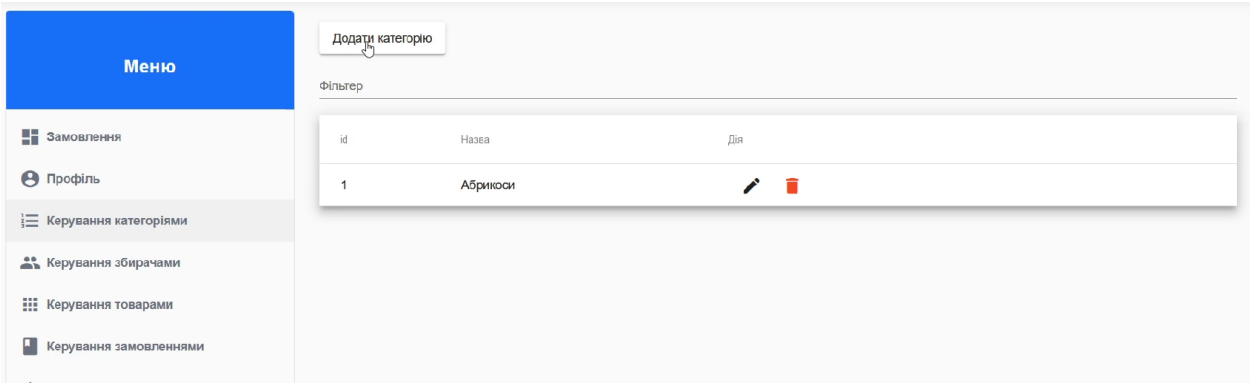


Рисунок 4.41. – Сторінка керування категоріями

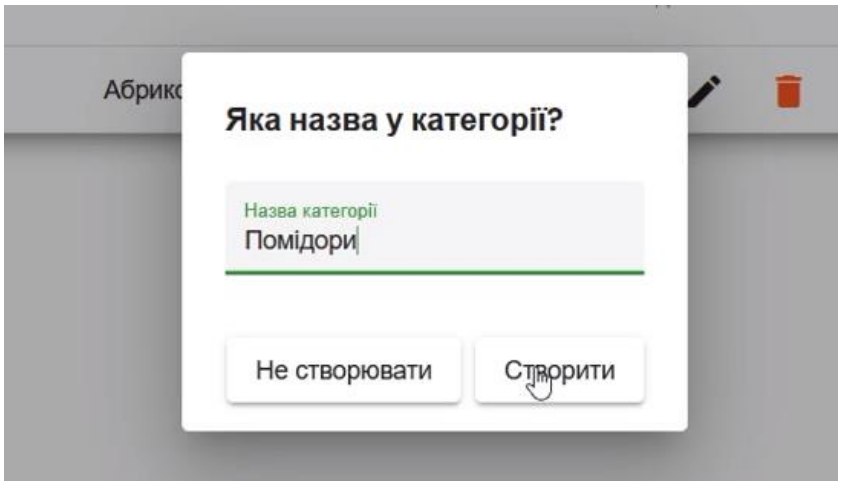


Рисунок 4.42. – Вікно створення нової категорії

У розділі “Керування збирачами”, який представлено на рис. 4.43 ми можемо зберігати інформацію про збирачів, додавати їх, редагувати та видаляти.

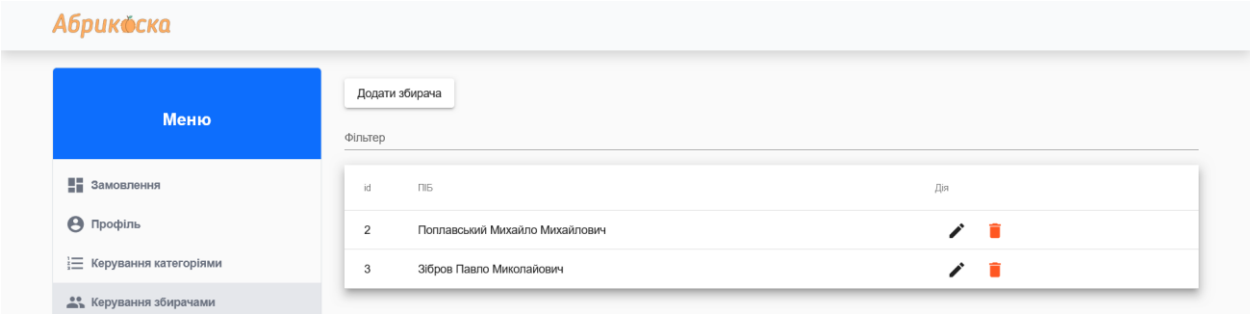


Рисунок 4.43. – Сторінка керування збирачами

У розділі “Керування товарами”, який представлено на рис. 4.44 ми можемо переглянути велику кількість товарів, які були завантажені за допомогою міграції бази даних. Ми можемо фільтрувати товари за ціною, назвою та застосовувати інші фільтри. Тут можна додавати, редагувати та видаляти товари.

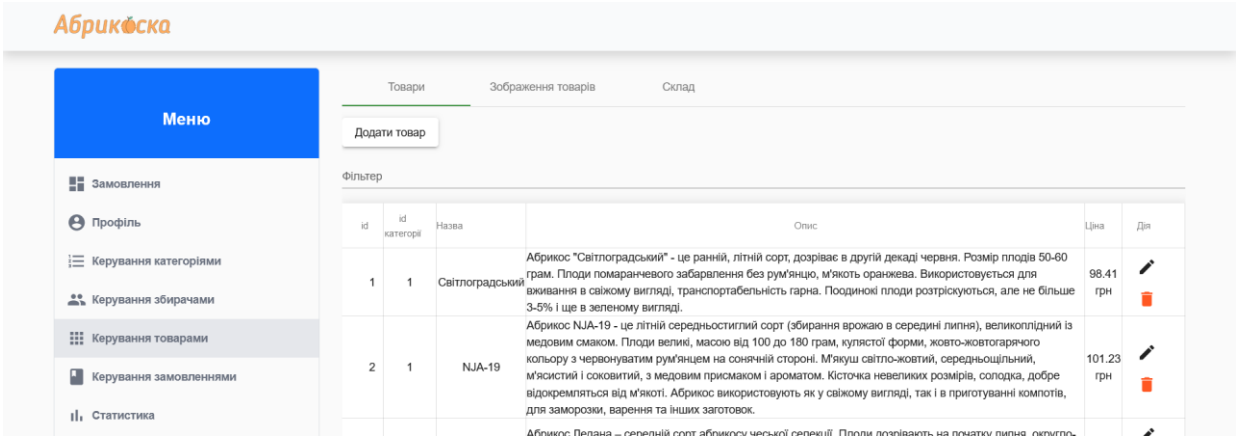


Рисунок 4.44 – Сторінка керування товарами

Окрім інформації про товари, є їх зображення, які представлені на рис. 4.45.

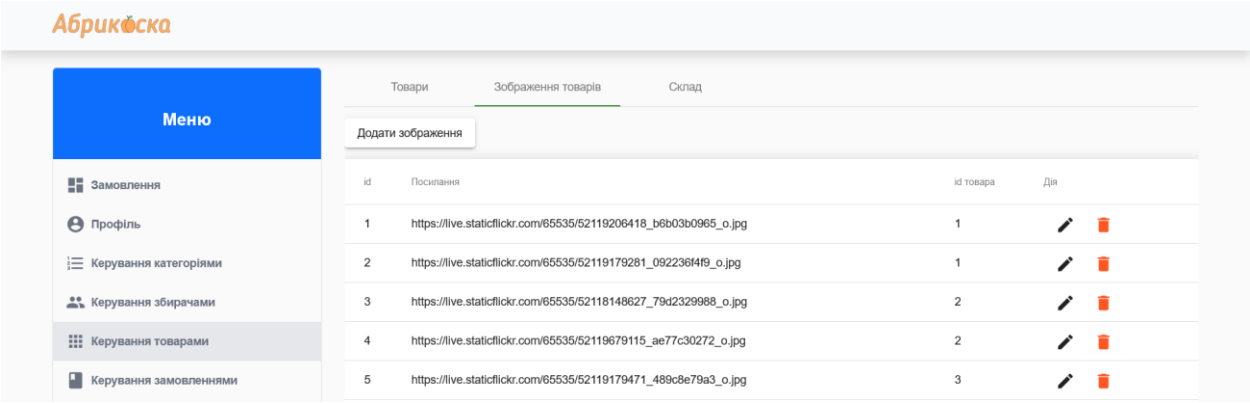


Рисунок 4.45. – Меню керування зображеннями товарів

На рис. 4.46 показана вкладка “Склад” де можна вручну увести товар, як показано на рис. 4.47, або завантажити дані збору з пристрою.

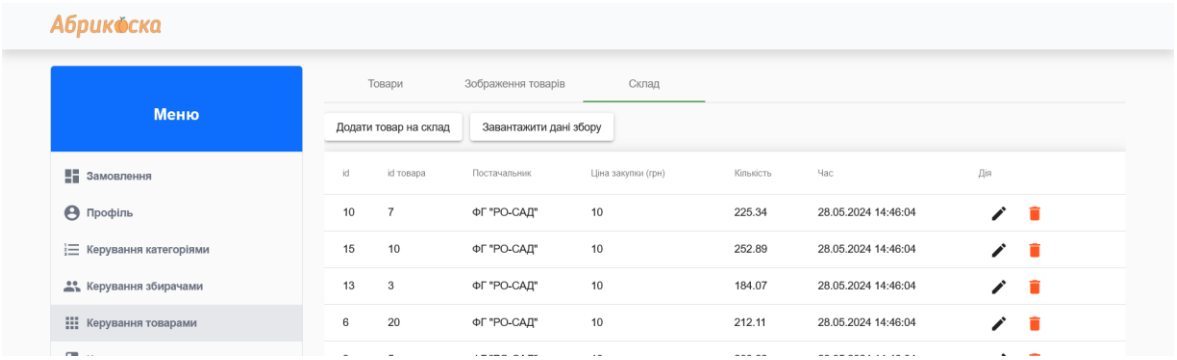


Рисунок 4.46. – Меню керування складом товарів

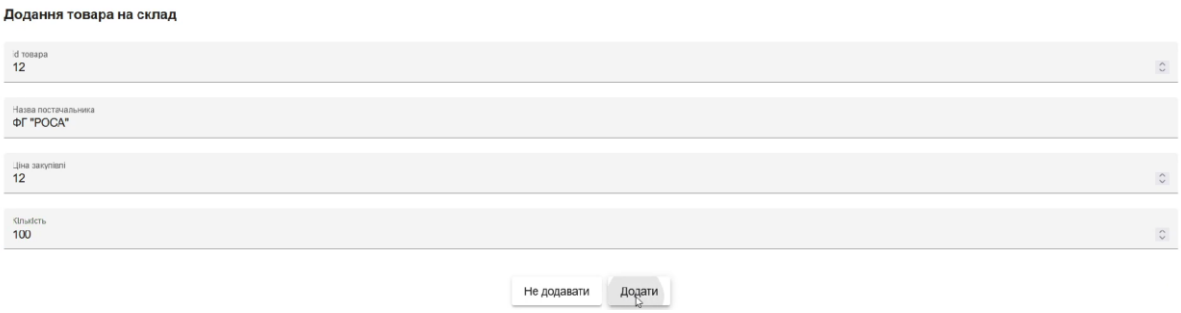


Рисунок 4.47. – Вікно додання нового товару на склад

Спробуємо замовити товари. Почнемо з додавання в кошик як на рис. 4.48. Далі натискаємо “Оформити замовлення” і переходимо до розділу на рис. 4.49, де потрібно ввести особисті дані, обрати спосіб оплати, а також місце доставки. У розділі “Доставка” автоматично вираховується назва пункту за

допомогою сервісу “Нова пошта”. Наприклад, коли ми починаємо вводити інформацію, “Нова пошта” пропонує вибір відповідного пункту.

Рисунок 4.48. – Вікно кошику замовлення

Рисунок 4.49. – Частина вікна оформлення замовлення

Після формування замовлення, у особистому кабінеті. На рис. 4.50 ми можемо переглянути його прогрес та побачити посилання на вибрані товари. Статус замовлення наразі є лише в процесі обробки.

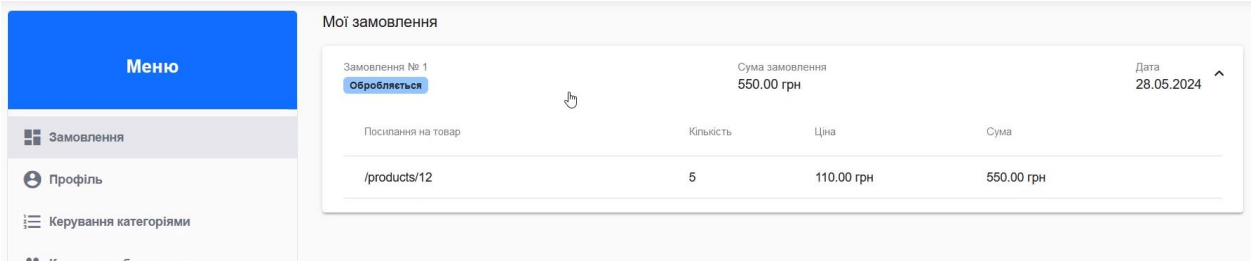


Рисунок 4.50. – Сторінка особистих замовлень з новим замовленням

Увійдемо в розділ складу і завантажимо дані збирання. Оберемо оплату за кг у розмірі 10 гривень та виберемо файл для завантаження, як на рис 4.51.

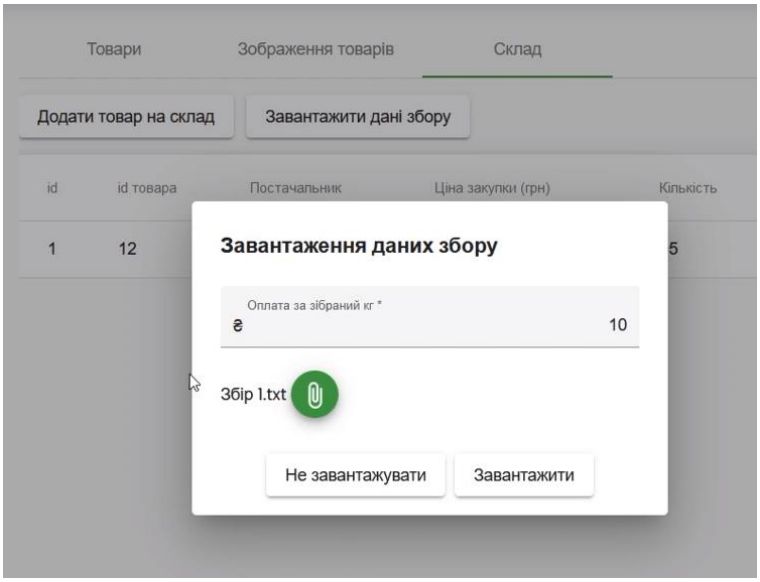


Рисунок 4.51. – Вікно завантаження даних збору

Після завантаження, ми можемо побачити дані збору врожаю. Завантаження відбулося швидко – за кілька секунд. Ми повторюємо процес для наступного завантаження. Після завершення ми бачимо, що інформація успішно збережена.

Тепер переходимо до основної статистики, вибираємо, наприклад, день 14 травня, та переглядаємо “Чек дня”. На рис. 4.52 ми можемо побачити, що Поплавському повинні заплатити 518.3 гривень, а зібраний товар становить 51.83 кг.

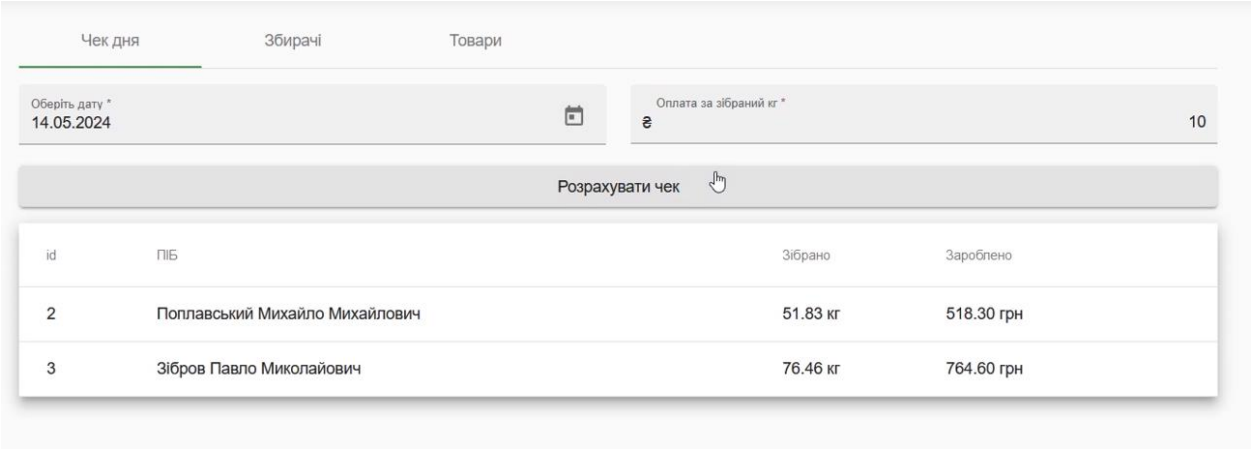


Рисунок 4.52. – Розділ “Чек дня”

Далі переглядаємо статистику збирачів, наприклад, за період з 9 квітня по 30 травня, ми можемо побачити продуктивність Поплавського на рис. 4.53.

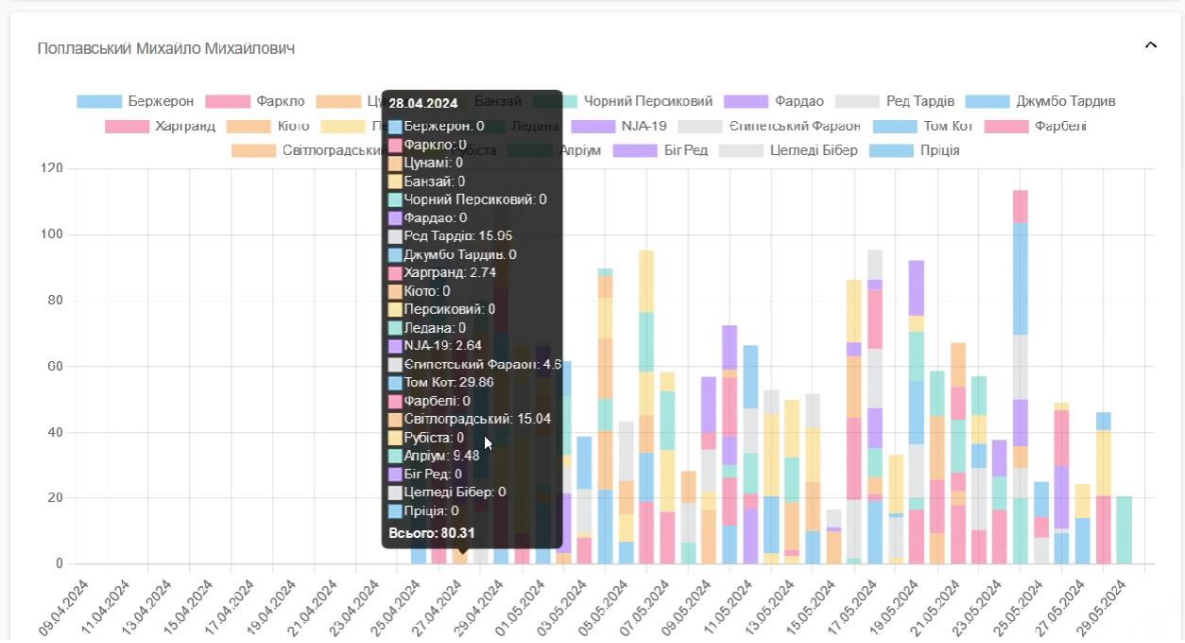


Рисунок 4.53. – Продуктивність збирача за період

Далі переходимо до товарів, вибираємо той самий період і розраховуємо. Спочатку ми бачимо на рис. 4.54 кругову діаграму за весь період – скільки всього зібрано певних товарів в кілограмах та відсотках.

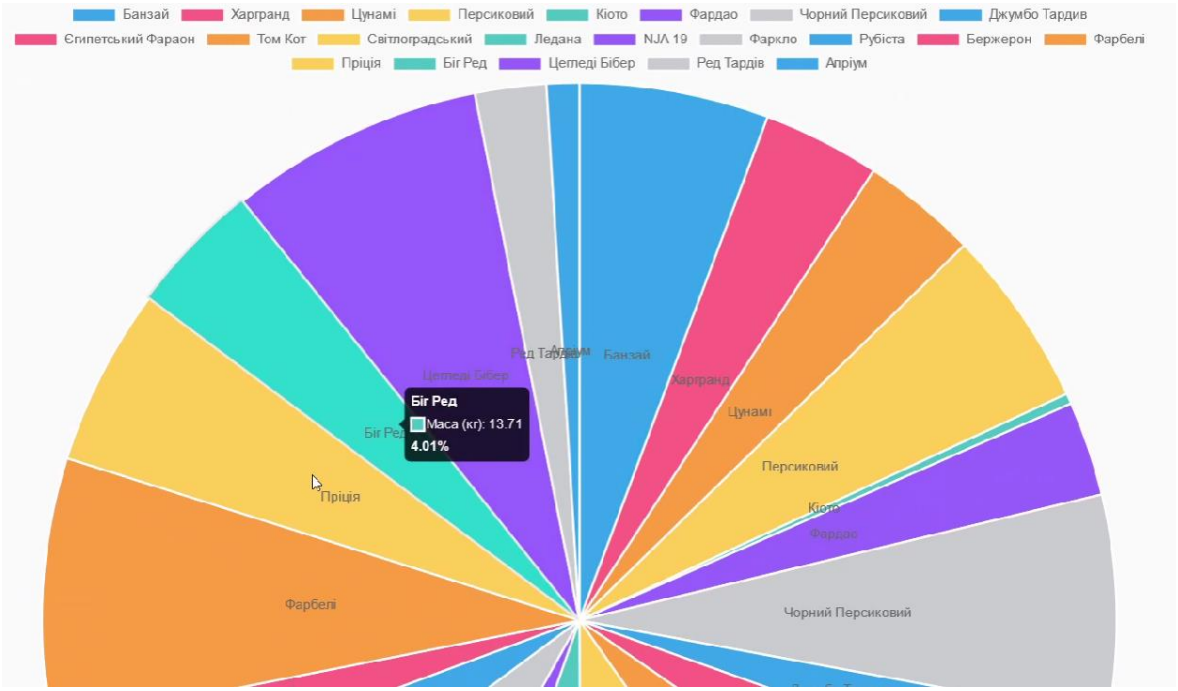


Рисунок 4.54. – Кругова діаграма зібраних товарів за період

Потім на рис. 4.55 – гістограма по днях, яка показує кількість товарів, які надійшли на склад за певний день.

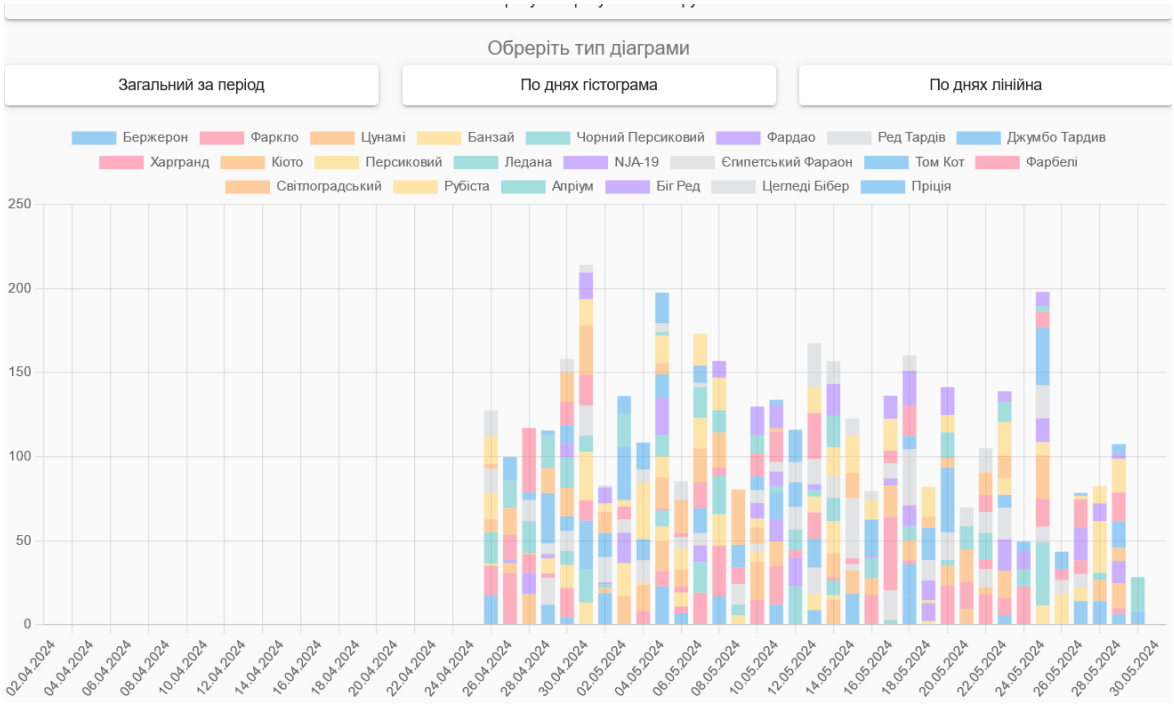


Рисунок 4.55. – Гістограма зібраного врожаю по днях

Далі на рис. 4.56 лінійна діаграма, що дозволяє нам визначити прибутковість різних сортів і час їх збирання.

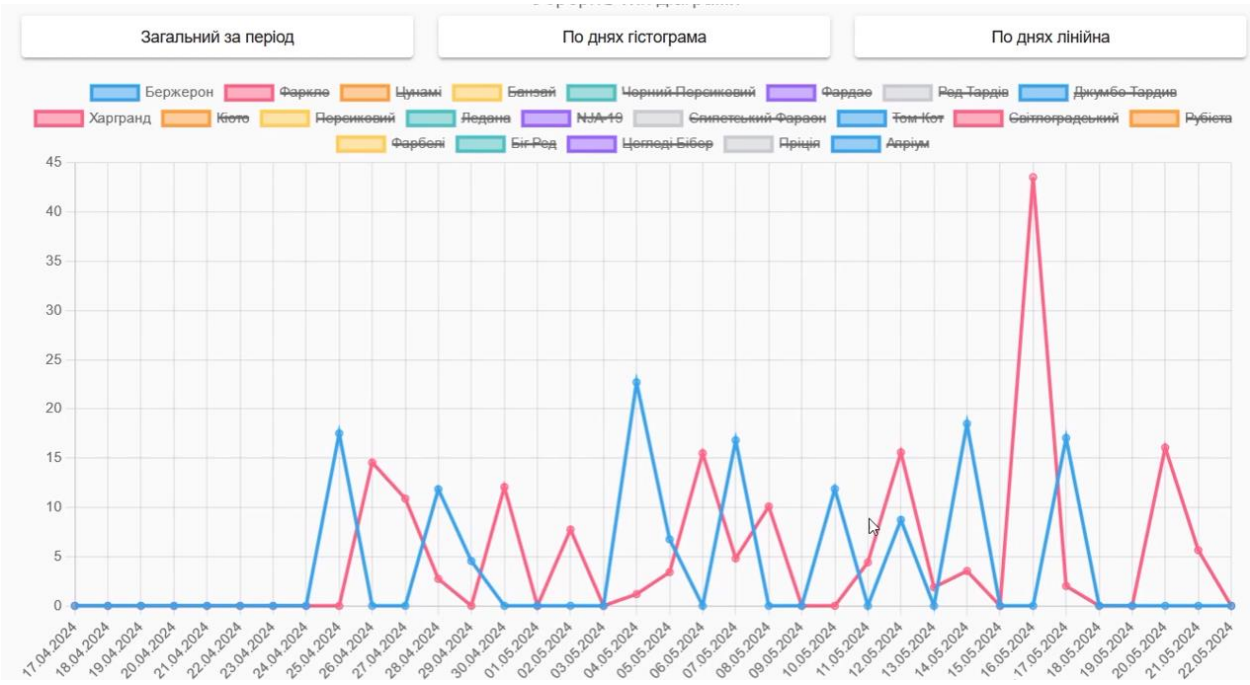


Рисунок 4.56. – Лінійна діаграма зібраного врожаю

ВИСНОВОК ДО РОЗДІЛУ 4

У даному розділі було наочно продемонстровано роботу розробленого апаратно-програмного комплексу для обліку та контролю збору врожаю. Зокрема, було детально описано процес взаємодії з пристроєм та веб-сайтом, що входять до складу комплексу.

Під час демонстрації роботи пристрою було показано основні етапи його використання. Було розглянуто калібрування ваги для забезпечення точності вимірювань. Також було продемонстровано, як відбувається процес додавання нового запису про збір врожаю, включаючи введення ідентифікатора працівника, ідентифікатора продукту та вимірювання маси зібраного продукту. Крім того, було показано, як пристрій може підключатися до Bluetooth для взаємодії з принтером етикеток, що дозволяє автоматично друкувати етикетки з інформацією про зібраний урожай. Також було продемонстровано процес підключення пристрою до мережі Wi-Fi, що надає можливість віддаленого налаштування та керування пристроєм через веб-інтерфейс.

Демонстрація роботи вебсайту розпочалася з огляду головної сторінки, яка містить інформацію про фермерське господарство та його продукцію. Було показано, як користувачі можуть переглядати товари, ознайомлюватися з контактною інформацією та переглядати галерею зображень. Особливу увагу було приділено процесу реєстрації та авторизації користувачів на сайті, а також функціоналу особистого кабінету. Для звичайних користувачів було продемонстровано можливість оформлення замовлень та відстеження їхнього статусу, а для адміністраторів – розширений функціонал з керування категоріями товарів, збирачами, товарами на складі, замовленнями та перегляду статистики. Було показано, як дані про збір врожаю, зібрані пристроєм, можуть бути завантажені на сайт для подальшого аналізу та формування звітів. Були продемонстровані різні види статистики, які можна отримати за допомогою вебсайту, такі як “Чек дня”, продуктивність збирачів та статистика зібраного врожаю за різними параметрами.

На мою думку, даний розділ надав вичерпну інформацію про роботу розробленого апаратно-програмного комплексу, підкресливши його функціональність, зручність використання та потенційну користь для фермерських господарств та агробізнесу.

ВИСНОВКИ

У дипломній роботі було розроблено апаратно-програмний комплекс для обліку та контролю збору врожаю.

Апаратна частина – це спеціалізований пристрій на базі мікроконтролера ESP32, який дозволяє збирачам вводити дані про зібраний урожай за допомогою клавіатури та зчитувати вагу за допомогою тензодатчика. Пристрій також має можливість підключення до Wi-Fi для передачі даних на сервер та Bluetooth для підключення до принтера етикеток.

Програмна частина – це веб-сайт, розроблений з використанням технологій Go, TypeScript та Angular, який надає інтерфейс для управління даними про збір врожаю, працівників, товари та замовлення. Він також надає аналітичні інструменти для візуалізації та аналізу даних про зібраний урожай.

У першому розділі було проведено огляд існуючих рішень для обліку та контролю збору врожаю. Було виявлено, що більшість існуючих систем мають високу вартість та складність в налаштуванні, що робить їх недоступними для багатьох малих фермерських господарств.

У другому розділі було проаналізовано технології, які були використані для розробки комплексу. Було обрано мікроконтролер ESP32 завдяки його широким можливостям та доступності. Для розробки вебсайту було обрано мову Go завдяки її продуктивності та масштабованості, а також фреймворки Angular та Gin для спрощення розробки.

У третьому розділі було детально описано процес розробки апаратної та програмної частин комплексу. Було розроблено електричну схему пристрою, підібрано необхідні компоненти та написано програмний код для мікроконтролера ESP32. Також було створено вебсайт з використанням обраних технологій.

У четвертому розділі було продемонстровано роботу розробленого комплексу. Було показано, як збирачі можуть використовувати пристрій для введення даних про зібраний урожай, як ці дані передаються на сервер та як

адміністратор може використовувати вебсайт для управління даними та перегляду статистики.

Загалом, розроблений апаратно-програмний комплекс успішно вирішує проблему обліку та контролю збору врожаю на малих фермерських господарствах. Він є доступним, простим у використанні та надає всі необхідні інструменти для ефективного управління процесом збору врожаю. Завдяки цьому комплексу фермери зможуть підвищити ефективність своєї роботи, зменшити витрати та збільшити прибутки.

Технічне завдання на дипломний бакалаврський проєкт виконано повністю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. МініСофт Урожай. *МініСофт*. URL: https://minisoft.ua/po/minisoft-urozhay (дата звернення: 29.04.2024).
2. Ціни на програми та обслуговування. *МініСофт - програми та обладнання для автоматизації торгівлі: магазинів, кафе, аптек, автомийок, салонів*. URL: https://minisoft.ua/page/cini-na-programi-ta-obslugovuvannya (дата звернення: 29.04.2024).
3. Торгівельне обладнання. *МініСофт - програми та обладнання для автоматизації торгівлі: магазинів, кафе, аптек, автомийок, салонів*. URL: https://minisoft.ua/shop (дата звернення: 29.04.2024).
4. Automated Harvest Workflows by Minisoft Harvest. *MiniSoft Harvest - automating business processes of harvesting berries, fruits and vegetables.* URL: https://minisoftharvest.com/blog/automated-harvest-workflows-minisoft-harvest (дата звернення: 30.04.2024).
5. How We Work – AgSquared. *AgSquared.* URL: https://agsquared.com/how-we-work/ (дата звернення: 30.04.2024).
6. How it works outcomes. *Home – AgSquared.* URL: https://agsquared.com/wp-content/uploads/2023/06/how-it-works-outcomes.png (дата звернення: 30.04.2024).
7. AgSquared and NATIVE AGTECH Partner to Provide Complete Seed-to-Sale Visibility for the Food Supply Chain. *Markets insider*. URL: https://markets.businessinsider.com/news/stocks/agsquared-and-native-agtech-partner-to-provide-complete-seed-to-sale-visibility-for-the-food-supply-chain-1028583793 (дата звернення: 30.04.2024).
8. Aegro Pricing, Alternatives & More 2024. *Capterra.* URL: https://www.capterra.com/p/216112/Aegro/ (дата звернення: 17.05.2024).
9. Controle de estoque – Aegro. *Aegro*. URL: https://aegro.com.br/funcionalidades/controle-de-estoque (дата звернення: 01.05.2024).
10. EXCLUSIVE: Brazilian farm management platform Aegro scores seed funding, eyes fintech opportunity. *AgFunderNews.* URL: https://agfundernews.com/aegro-farm-management-platform-scores-funding-eyes-fintech-opportunity (дата звернення: 01.05.2024).
11. Комп'ютерні програми для автоматизації торгівлі та обліку торгівлі. *POS Vector.* URL: https://www.pos-vector.net/torhivlia/pz/ (дата звернення: 01.05.2024).
12. GLOBALG.A.P. | What we offer. *GLOBALG.A.P. | Smart farm assurance solutions.* URL: https://www.globalgap.org/what-we-offer/ (дата звернення: 01.05.2024).

13. Датчик ваги (тензодатчик) 20 кг. *Arduino в Україні*. URL: https://arduino.ua/prod4484-datchik-vesa-tenzodatchik-20-kg (дата звернення: 02.05.2024).
14. Папушин Ю. Л. Основи автоматизації гірничого виробництва : курс лекцій / Ю. Л. Папушин, В. С. Білецький ; Дон. нац. техн. ун-т, Донецьке відділення НТШ. – Донецьк : Східний видавничий дім, 2007. С. 154 – 155.
15. HX711 Двоканальний модуль датчиків ваги (тензодатчиків). *Arduino в Україні*. URL: https://arduino.ua/prod1147-hx711-dual-channel-weighing-sensor-module (дата звернення: 02.05.2024).
16. HX711 Datasheet(PDF). *ALLDATASHEET.COM - Electronic Parts Datasheet Search.* URL: https://pdf1.alldatasheet.com/datasheet-pdf/view/1132222/AVIA/HX711.html (дата звернення: 02.05.2024).
17. Kane P. Working with Matrix Keypads. *Jameco Electronics.* URL: https://www.jameco.com/Jameco/workshop/JamecoBuilds/working-with-matrix-keypads.html (дата звернення: 03.05.2024).
18. DS1302 Datasheet. *ALLDATASHEET.COM - Electronic Parts Datasheet Search.* URL: https://pdf1.alldatasheet.com/datasheet-pdf/view/447115/TGS/DS1302.html (дата звернення: 03.05.2024).
19. DS1302 Real Time Clock. *SZHJW Technology Store.* URL: https://www.aliexpress.com/item/1005005796411444.html (дата звернення: 04.05.2024).
20. I2C-bus specification and user manual - UM10204. *Automotive, IoT & Industrial Solutions | NXP Semiconductors.* URL: https://www.nxp.com/docs/en/user-guide/UM10204.pdf (дата звернення: 04.05.2024).
21. LCD дисплей 20x4 шина I2C Синій (з підтримкою кирилиці). *Arduino в Україні.* URL: https://arduino.ua/prod3886-lcd-displei-20x4-shina-i2c-sinii-s-podderjkoi-kirillici (дата звернення: 04.05.2024).
22. ESP32-WROOM-32 Datasheet. *Espressif Systems.* URL: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf (дата звернення: 05.05.2024).
23. Wi-Fi модуль DevKit V1 з ESP-32. *Arduino в Україні.* URL: https://arduino.ua/prod3990-wi-fi-modyl-devkit-v1-s-esp-32 (дата звернення: 05.05.2024).
24. Модуль microSD карти. *Arduino в Україні.* URL: https://arduino.ua/prod1601-modyl-micro-sdtf-card (дата звернення: 05.05.2024).
25. SDSQXAF-032G-GN6MA. *ALLDATASHEET.COM - Electronic Parts Datasheet Search.* URL: https://pdf1.alldatasheet.com/datasheet-

pdf/view/1246243/ETC1/SDSQXAF-032G-GN6MA.html (дата звернення: 05.05.2024).
26. Miller M. Makuna/Rtc: Arduino Library for RTCs, Ds1302, Ds1307, Ds3231, Ds3232, Ds3234 and Pcf8563/BM8563 with deep support. *GitHub*. URL: https://github.com/Makuna/Rtc (дата звернення: 07.05.2024).
27. Frank de Brabander. fdebrabander/Arduino-LiquidCrystal-I2C-library: Library for the LiquidCrystal LCD display connected to an Arduino board. *GitHub*. URL: https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library (дата звернення: 08.05.2024).
28. Stanley M., Brevig A. Chris--A/Keypad: A version of the keypad library found in Wiring. *GitHub*. URL: https://github.com/Chris--A/Keypad (дата звернення: 08.05.2024).
29. Necula B. bogde/HX711: An Arduino library to interface the Avia Semiconductor HX711 24-Bit Analog-to-Digital Converter (ADC) for Weight Scales. *GitHub*. URL: https://github.com/bogde/HX711 (дата звернення: 10.05.2024).
30. Arduino. Arduino & Serial Peripheral Interface (SPI). *Arduino Documentation*. URL: https://docs.arduino.cc/learn/communication/spi (дата звернення: 10.05.2024).
31. Greiman W., SparkFun Electronics. arduino-libraries/SD: SD Library for Arduino. *GitHub*. URL: https://github.com/arduino-libraries/SD (дата звернення: 10.05.2024).
32. Arduino LLC. arduino-libraries/WiFi: WiFi Library for Arduino. *GitHub*. URL: https://github.com/arduino-libraries/WiFi (дата звернення: 11.05.2024).
33. Gochkov H. dvarrel/AsyncTCP: Async TCP Library for ESP32. *GitHub*. URL: https://github.com/dvarrel/AsyncTCP (дата звернення: 13.05.2024).
34. SPIFFS Filesystem. *Technical Documents | Espressif Systems*. URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/storage/spiffs.html (дата звернення: 14.05.2024).
35. Copercini E. Bluetooth. *Technical Documents | Espressif Systems*. URL: https://docs.espressif.com/projects/arduino-esp32/en/latest/api/bluetooth.html (дата звернення: 14.05.2024).
36. Lospinoso J. C++ Crash Course: A Fast-Paced Introduction. No Starch Press, 2019. 792 с.
37. Documentation - The Go Programming Language. *The Go Programming Language*. URL: https://go.dev/doc/ (дата звернення: 14.05.2024).
38. Documentation | Gin Web Framework. *Gin Web Framework*. URL: https://gin-gonic.com/docs/ (дата звернення: 14.05.2024).
39. GORM Guides. *GORM*. URL: https://gorm.io/docs/ (дата звернення: 14.05.2024).

40. Documentation - TypeScript for the New Programmer. *TypeScript: JavaScript With Syntax For Types.* URL: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html (дата звернення: 17.05.2024).
41. Typescript. *npm.* URL: https://www.npmjs.com/package/typescript (дата звернення: 17.05.2024).
42. Introduction to the Angular docs. *Angular.* URL: https://angular.io/docs (дата звернення: 17.05.2024).
43. PostgreSQL 16.3 Documentation. *PostgreSQL.* URL: https://www.postgresql.org/docs/16/index.html (дата звернення: 17.05.2024).

ДОДАТОК 1

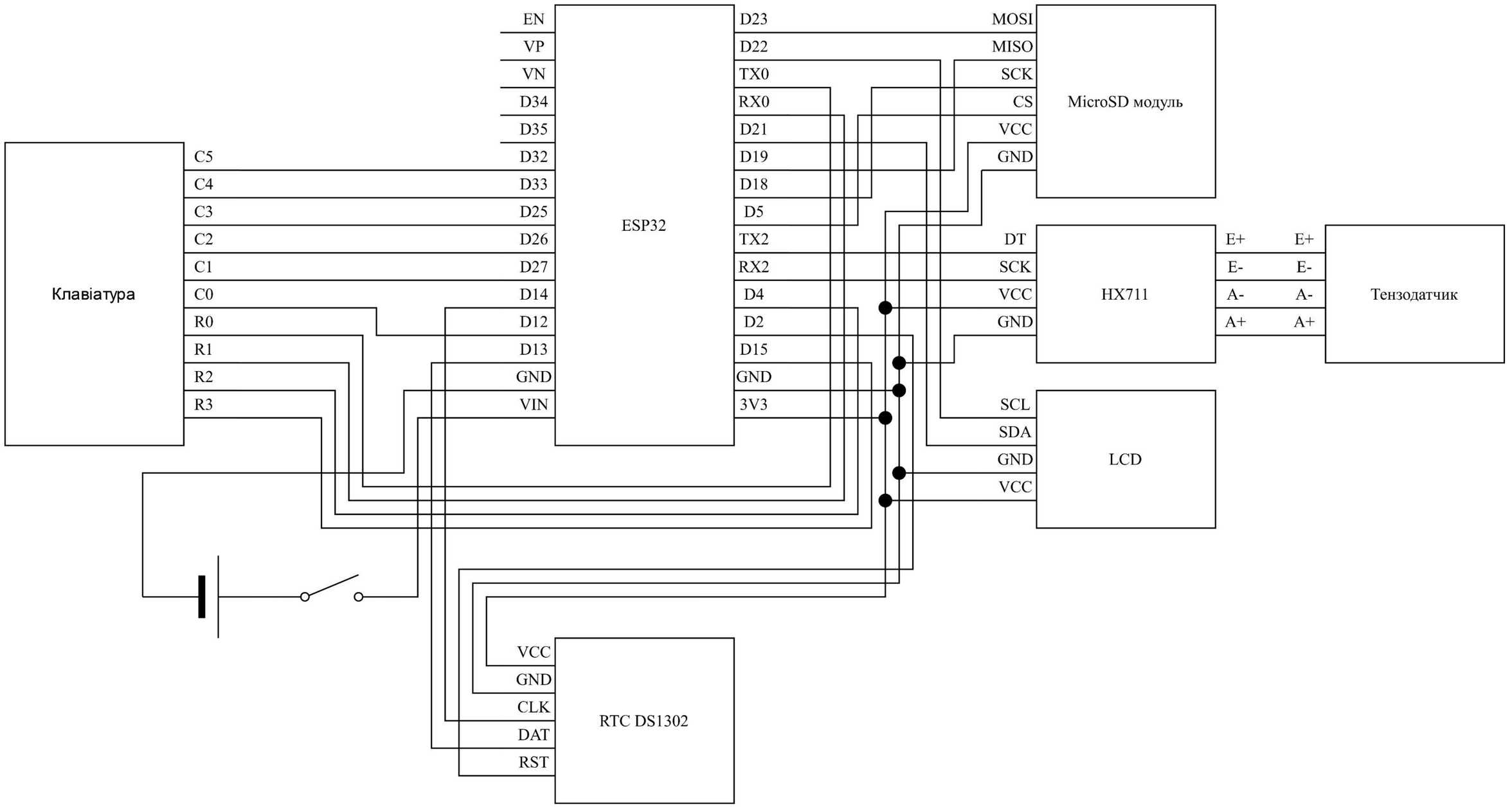
Апаратно-програмний комплекс для обліку та контролю збору врожаю

Схема структурна

ІАЛЦ.467200.004 Е1

Аркушів 1

Київ – 2024 р



					ІАЛЦ.467200.004 Е1		
Змн	Арк.	№ докум.	Підпис	Дата		Літ.	Маса / Масштаб
Розроб.		Литвиненко Д.О.			*Апаратно-програмний комплекс для обліку та контролю збору врожаю*		
Перевірив		Клименко І.А.			Схема структурна	Аркуш 1	Аркушів 1
						КПІ ім. Ігоря Сікорського, ФІОТ, Група ІО-02	
Н. Контр.		Виноградов Ю.М.					
Затверд.							

ДОДАТОК 2

Апаратно-програмний комплекс для обліку та контролю збору врожаю

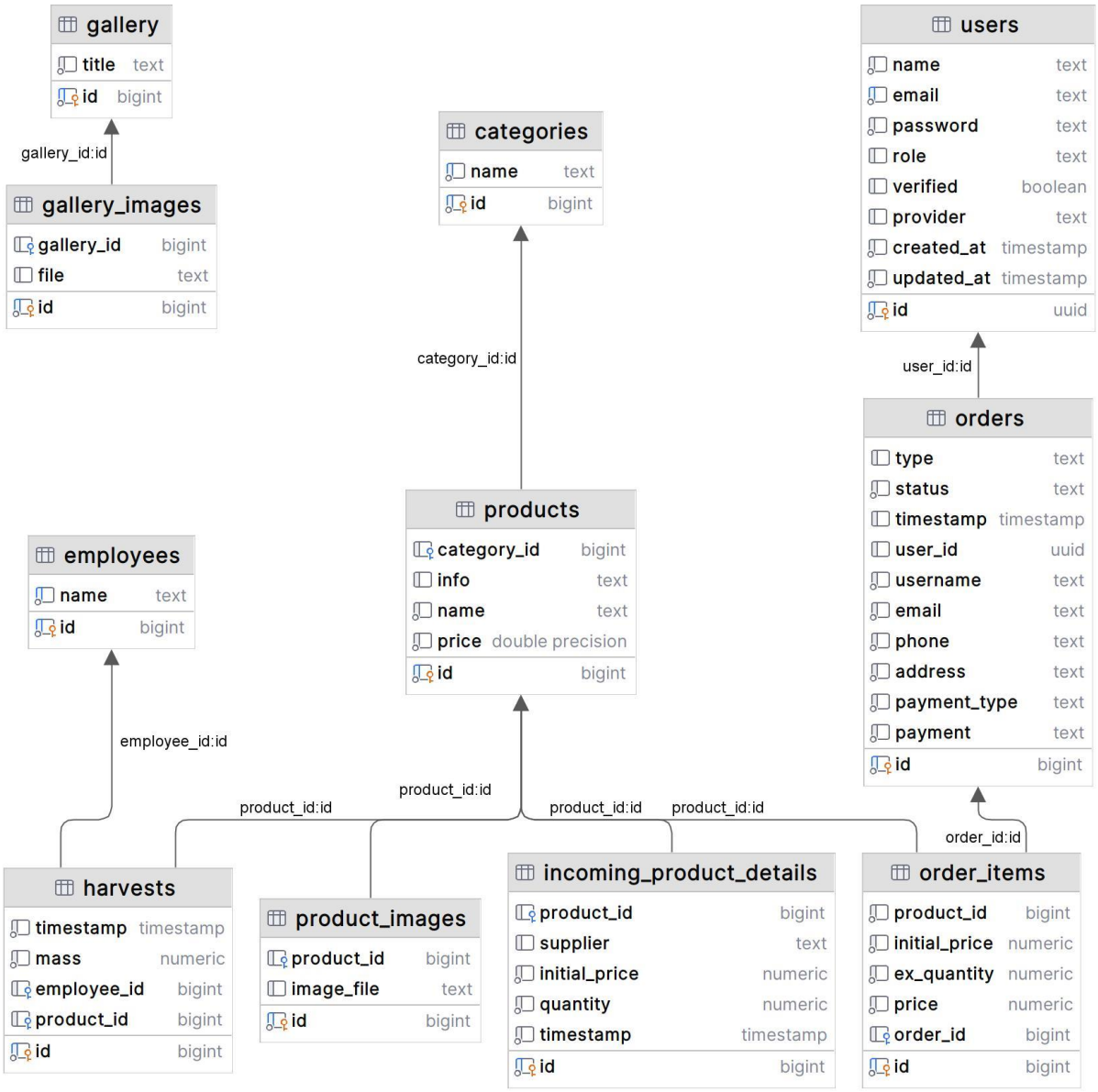
Діаграма даних

Схема функціональна

ІАЛЦ.467200.005 Е2

Аркушів 1

Київ – 2024 р



					ІАЛЦ.467200.005 Е2		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Литвиненко Д.О.			*Апаратно-програмний комплекс для обліку та контролю збору врожаю* Діаграма даних Схема функціональна	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Клименко І.А.					
Н. Контр.		Виноградов Ю.М.				КПІ ім. Ігоря Сікорського, ФІОТ, Група ІО-02	
Затвердив							

ДОДАТОК 3

Апаратно-програмний комплекс для обліку та контролю збору врожаю

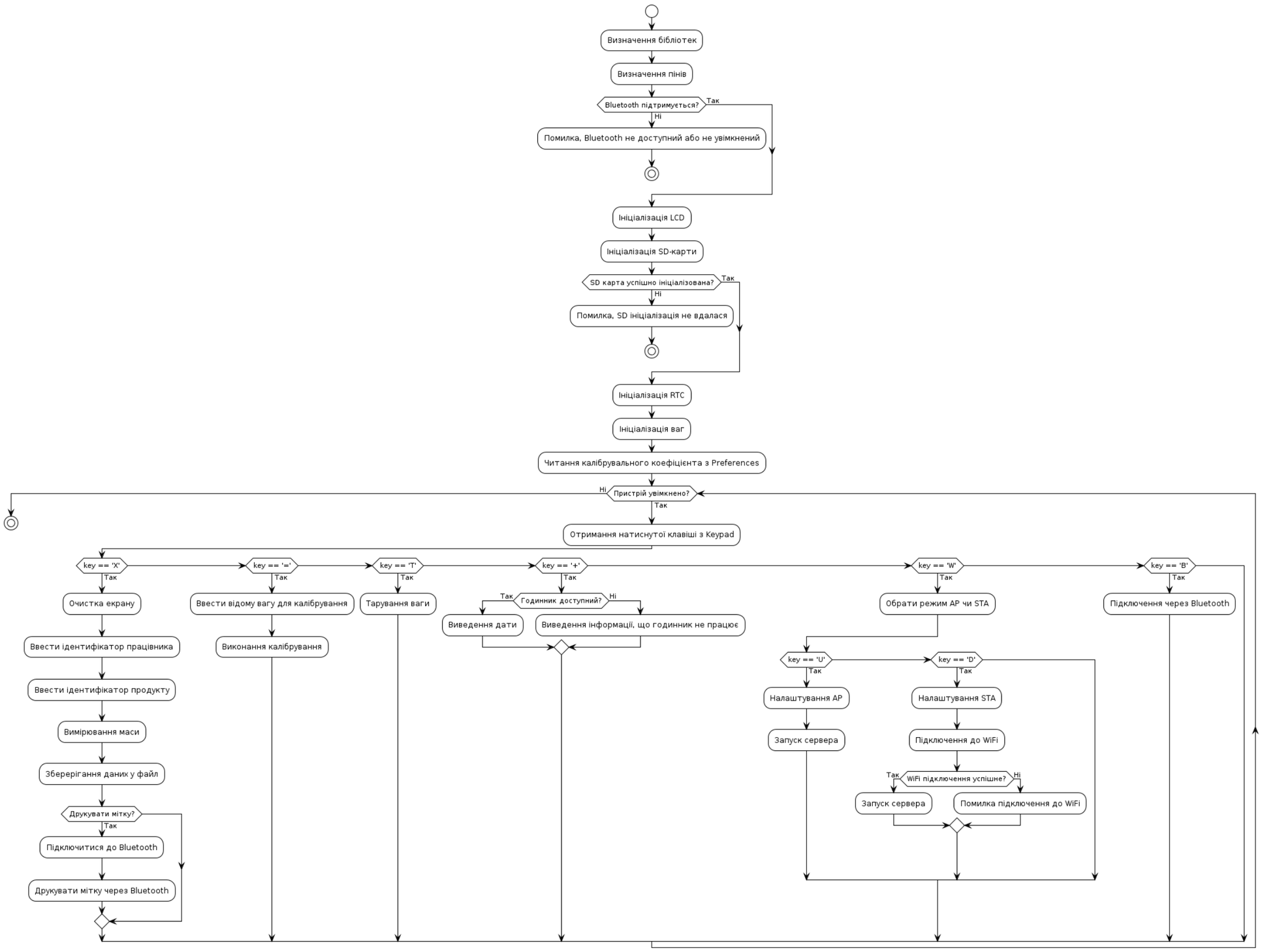
Алгоритм роботи програми

Схема принципова

ІАЛЦ.467200.006 Е3

Аркушів 1

Київ – 2024 р



					ІАЛЦ.467200.006 Е3	
Змн	Арк.	№ докум.	Підпис	Дата		
Розроб.		Литвиненко Д.О.			*Апаратно-програмний комплекс для обліку та контролю збору врожаю*	Літ. / Маса / Масштаб
Перевірив		Клименко І.А.				Аркуш 1 Аркушів 1
					Алгоритм роботи програми	КПІ ім. Ігоря Сікорського, ФІОТ, Група ІО-02
Н. Контр.		Виноградов Ю.М.			Схема принципова	
Затверд.						

ДОДАТОК 4

Апаратно-програмний комплекс для обліку та контролю збору врожаю

Текст програмного коду

ІАЛЦ.467200.007 Е4

Аркушів 12

Київ – 2024 р

Модуль ініціалізації системи setup.ino

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h> // LiquidCrystal I2C by Frank de
Brabander
#include <Keypad.h> // Keypad by Mark Stanley
#include <ThreeWire.h> // Rtc by Makuna
#include <RtcDS1302.h> // Rtc by Makuna
#include <HX711.h> // HX711 Arduino library by Bogdan Necula
#include <FS.h>
#include <SPI.h>
#include <SD.h> // SD by Arduino, SparkFun
#include <Preferences.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <SPIFFS.h>
#include <BluetoothSerial.h>
#include <time.h>

// PINS
#define SD_MOSI    23
#define LCD_SCL    22
#define KEYPAD_R0  1 // TX0
#define KEYPAD_R1  3 // RX0
#define LCD_SDA    21
#define SD_MISO    19
#define SD_SCK     18
#define SD_CS      5
#define SCALE_DT   17 // TX2
#define SCALE_SCK  16 // RX2
#define KEYPAD_R2  4
#define RTS_RST    2
#define KEYPAD_R3  15
#define RTS_DAT    13
#define KEYPAD_C0  12
#define RTS_CLK    14
#define KEYPAD_C1  27
#define KEYPAD_C2  26
#define KEYPAD_C3  25
#define KEYPAD_C4  33
#define KEYPAD_C5  32
```

```
#define PROJECT_NAME "apricotka"
#define TZ "EET-2EEST,M3.5.0/3,M10.5.0/4"

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only
available for the ESP32 chip.
#endif
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and
enable it
#endif

// Keypad
#define KEYPAD_ROW_NUM     4 // four rows
#define KEYPAD_COLUMN_NUM  6 // six columns
byte pinRows[KEYPAD_ROW_NUM]={KEYPAD_R0,KEYPAD_R1,KEYPAD_R2,KEYPAD_R3};
byte
pinCols[KEYPAD_COLUMN_NUM]={KEYPAD_C0,KEYPAD_C1,KEYPAD_C2,KEYPAD_C3,KEYP
AD_C4,KEYPAD_C5};
char keys[KEYPAD_ROW_NUM][KEYPAD_COLUMN_NUM]={
  {'4','7','W','1','0','P'},
  {'5','8','<','2','.','U'},
  {'6','9','B','3','X','='},
  {'+','C','>','T','E','D'}
};
// 4 7 WIFI 1 0 PRINT
// 5 8 < 2 . UP
// 6 9 BLUETOOTH 3 X(ADD NEW) =(CALIBRATE)
// + C > T E(>0<) DOWN

// Init of main parts
LiquidCrystal_I2C lcd(0x27, 20, 4);
Keypad keypad = Keypad(makeKeymap(keys), pinRows, pinCols,
KEYPAD_ROW_NUM, KEYPAD_COLUMN_NUM );
ThreeWire myWire(RTS_DAT, RTS_CLK, RTS_RST); // DAT, CLK, RST
RtcDS1302<ThreeWire> Rtc(myWire);
HX711 scale;
Preferences preferences;

// Varialbles
unsigned long timing;
int weight;
```

```
char key;
char s = 'f';
String bluetoothLabel = "";

BluetoothSerial SerialBT;
// Set web server port number to 80
AsyncWebServer server(80);

void notFound(AsyncWebServerRequest *request) {
  request->send(404, "text/plain", "Not found");
}

// Replaces placeholder with stored values
String processor(const String& var){
      //Serial.println(var);
      if(var == "language"){
            return preferences.getString("language", "English");
      }
      else if(var == "dateTime"){
            return getDateTimeString(Rtc.GetDateTime());
      }
      else if(var == "slaveName"){
            return preferences.getString("slaveName", "error");
      }
  else if(var == "bluetoothPin"){
            return preferences.getString("bluetoothPin", "0000");
      }
      // else if(var == "label"){
      //    return readFile(SPIFFS, "/label.txt");
      // }
  else if(var == "routerSSID"){
            return preferences.getString("routerSSID", "error");
      }
  else if(var == "routerPassword"){
            return preferences.getString("routerPassword", "error");
      }
  // else if(var == "hotspotSSID"){
      //    return preferences.getString("hotspotSSID", "Apricotka
Harvest");
      // }
  // else if(var == "hotspotPassword"){
      //    return preferences.getString("hotspotPassword", "12345678");
      // }
  else if(var == "calFactor"){
```

```
            return String(preferences.getFloat("calFactor", 100.96));
        }
        return String();
}

void setup() {
  Serial.begin(115200);
  Serial.println();

  preferences.begin(PROJECT_NAME, false);

  if(!SPIFFS.begin(true)){
    Serial.println("An Error has occurred while mounting SPIFFS");
    return;
  }

  // LCD
  lcd.init();
  lcd.backlight();

  // SD
  SPI.begin(SD_SCK, SD_MISO, SD_MOSI, SD_CS);
  Serial.print("Initializing SD card...");
  if (!SD.begin()) {
    Serial.println("SD initialization failed!");
    lcd.setCursor(0, 0);
    lcd.print("SD isn't available.");
    lcd.setCursor(0, 1);
    lcd.print("Please insert SD,");
    lcd.setCursor(0, 2);
    lcd.print("and restart device");
    while (true);
  }
  Serial.println("SD initialization done.");
  uint8_t cardType = SD.cardType();
  if(cardType == CARD_NONE){
    lcd.setCursor(0, 0);
    lcd.print("No SD card attached");
    while (true);
  }

  // RTC
  Rtc.Begin();
  RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
```

```
  if (!Rtc.IsDateTimeValid()) {
      Serial.println("RTC lost confidence in the DateTime!");
      Rtc.SetDateTime(compiled);
  }
  if (Rtc.GetIsWriteProtected()) {
      Serial.println("RTC was write protected, enabling writing now");
      Rtc.SetIsWriteProtected(false);
  }
  if (!Rtc.GetIsRunning()) {
      Serial.println("RTC was not actively running, starting now");
      Rtc.SetIsRunning(true);
  }

  RtcDateTime now = Rtc.GetDateTime();
  if (now < compiled) {
      Serial.println("RTC is older than compile time!");
      Rtc.SetDateTime(compiled);
  } else if (now > compiled) {
      Serial.println("RTC is newer than compile time.");
  } else if (now == compiled) {
      Serial.println("RTC is the same as compile time!");
  }

  // Scale
  scale.begin(SCALE_DT, SCALE_SCK);
  scale.set_scale();
  scale.tare();
  float temp = preferences.getFloat("calFactor", 100.96); // Get
calibration factor
  if (temp != 0) {
    scale.set_scale(temp);
  }
}
```

Модуль основного потоку програми loop.ino

```
void loop() {
  key = keypad.getKey();
  if (key) {
    if (key == 'X') {
      lcd.clear();
      int employee;
      int product;
      float mass;

      String employeeStringForLCD = "Employee: ";
      while (true) {
        clearLCDLine(0);
        lcd.setCursor(0, 0);
        lcd.print(employeeStringForLCD);
        String employeeString =
inputFromKeypad(employeeStringForLCD.length(), 0, false);
        if (isNumber(employeeString)) {
          employee = employeeString.toInt();
          break;
        }
      }

      String productStringForLCD = "Product: ";
      while (true) {
        clearLCDLine(1);
        lcd.setCursor(0, 1);
        lcd.print(productStringForLCD);
        String productString =
inputFromKeypad(productStringForLCD.length(), 1, false);
        if (isNumber(productString)) {
          product = productString.toInt();
          break;
        }
      }

      String massStringForLCD = "Mass: ";
      lcd.setCursor(0, 2);
      lcd.print(massStringForLCD);
      unsigned long massTiming;
      while (true) {
        if (scale.is_ready()) {
          key = keypad.getKey();
```

```
    if (key == '>') {
      break;
    } else if (key == 'T') {
      scale.tare();
    }
    if (millis() - massTiming > 1000) {
      lcd.setCursor(massStringForLCD.length(), 2);
      mass = scale.get_units(10) / 1000.0;
      //if(weight<0) { weight=0.00;}
      lcd.print(mass);lcd.print("    ");
      massTiming = millis();
    }
  }
}

RtcDateTime now = Rtc.GetDateTime();
String dateOfHarvest = getDateString(now);
String dateTimeOfHarvest = getDateTimeString(now);

String filename = "/" + dateOfHarvest + ".json";
String text = "{\"timestamp\":\"" + dateTimeOfHarvest +
"\",\"mass\":" + mass + ",\"employee_id\":" + employee +
",\"product_id\":" + product + "}";

bool fileExists = SD.exists(filename);
File file = SD.open(filename, FILE_WRITE);
if (fileExists) {
  text = "," + text;
  file.seek(file.size()-1);
} else {
  text = "[\n" + text;
}
text = text + "\n]";
file.print(text);
file.close();

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Print label?");
lcd.setCursor(0, 1);
lcd.print("Yes - 'PRINT'");
lcd.setCursor(0, 2);
lcd.print("No - '>'");
```

```
      while (true) {
        key = keypad.getKey();
        if (key == '>') {
          break;
        } else if (key == 'P') {
          if (!SerialBT.isClosed() && !SerialBT.connected()) {
            if (!bluetoothConnect()){
              break;
            }
          }
          bluetoothLabel = readFile(SPIFFS, "/label.txt");
          bluetoothLabel.replace("~date~", dateOfHarvest);
          bluetoothLabel.replace("~employee~", String(employee));
          bluetoothLabel.replace("~mass~", String(mass));
          bluetoothLabel.replace("~product~", String(product));
          uint8_t buf[bluetoothLabel.length()];
          memcpy(buf,bluetoothLabel.c_str(),bluetoothLabel.length());
          SerialBT.write(buf,bluetoothLabel.length());
          SerialBT.println();
          break;
        }
      }

    } else if (key == '=') {
      if (scale.is_ready()) {
        int weightOfObjectForCalibration;
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Enter weight of");
        lcd.setCursor(0, 1);
        lcd.print("object for calib-");
        lcd.setCursor(0, 2);
        lcd.print("ration in grams");
        String weightStringForLCD = "Weight: ";

        while (true) {
          clearLCDLine(3);
          lcd.setCursor(0, 3);
          lcd.print(weightStringForLCD);
          String weightString =
inputFromKeypad(weightStringForLCD.length(), 3, false);
          if (isNumber(weightString)) {
            weightOfObjectForCalibration = weightString.toInt();
            break;
```

```
  }
}

scale.set_scale();
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Remove any weights");
lcd.setCursor(0, 1);
lcd.print("from the scale");
for (byte i = 5; i > 0; i--) {
  lcd.setCursor(0, 2);
  lcd.print(i);
  delay(1000);
}

scale.tare();
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Tare done.");
lcd.setCursor(0, 1);
lcd.print("Place a known weight");
for (byte i = 9; i > 0; i--) {
  lcd.setCursor(0, 2);
  lcd.print(i);
  delay(1000);
}

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Start calibration...");
lcd.setCursor(0, 1);
lcd.print("Please wait...");
int reading = scale.get_units(10);
for (byte i = 5; i > 0; i--) {
  reading = (scale.get_units(10) + reading) / 2;
  lcd.setCursor(0, 2);
  lcd.print(i);
  delay(1000);
}

float cf = (float) reading / (float)
weightOfObjectForCalibration;
preferences.putFloat("calFactor", cf);
scale.set_scale(cf);
```

```
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Calibration factor:");
      lcd.setCursor(0, 1);
      lcd.print(cf);
      lcd.setCursor(0, 2);
      lcd.print("The scales are ready");
      for (byte i = 9; i > 0; i--) {
        lcd.setCursor(0, 3);
        lcd.print(i);
        delay(1000);
      }
      lcd.clear();
    } else {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("HX711 not found.");
      lcd.setCursor(0, 1);
      lcd.print("Try one more time");
    }
  } else if (key == 'E') {
  } else if (key == 'T') {
    scale.tare();
  } else if (key == '+') {
    rtcWork();
  } else if (key == 'W') {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Press button");
    lcd.setCursor(0, 1);
    lcd.print("WiFi AP - 'U'");
    lcd.setCursor(0, 2);
    lcd.print("WiFi STA - 'D'");
    lcd.setCursor(0, 3);
    lcd.print("To menu - 'E'");
    while (true) {
      key = keypad.getKey();
      if (key == 'E') {
        lcd.clear();
        break;
      } else if (key == 'U' || key == 'D') {
        SerialBT.end();
        server.end();
```

```
            WiFi.disconnect(true);
            WiFi.mode(WIFI_OFF);

            if (key == 'U') {
              String ssid = "apricotka-harvest";
              String pass = "11111111";

              lcd.clear();
              lcd.setCursor(0, 0);
              lcd.print("Bluetooth off");
              lcd.setCursor(0, 1);
              lcd.print("Runing AP");
              lcd.setCursor(0, 2);
              lcd.print(ssid);

              WiFi.mode(WIFI_AP);
              WiFi.softAP(ssid, pass);

              lcd.clear();
              lcd.setCursor(0, 0);
              lcd.print("Bluetooth off");
              lcd.setCursor(0, 1);
              lcd.print("WiFi on.");
              lcd.setCursor(0, 2);
              lcd.print("IP address:");
              lcd.setCursor(0, 3);
              lcd.print("192.168.4.1");

              startServer();
            } else {
              String ssid = preferences.getString("routerSSID",
"kahovka_2.4g");
              String pass = preferences.getString("routerPassword",
"m38m38m38");

              lcd.clear();
              lcd.setCursor(0, 0);
              lcd.print("Bluetooth off");
              lcd.setCursor(0, 1);
              lcd.print("Connecting to");
              lcd.setCursor(0, 2);
              lcd.print(ssid);

              WiFi.mode(WIFI_STA);
```

```
            WiFi.begin(ssid, pass);
            lcd.setCursor(0, 3);
            bool wifiOn = false;
            for (int i = 0; i < 20; i++) {
              if (WiFi.status() == WL_CONNECTED) {
                wifiOn = true;
                break;
              }
              delay(500);
              lcd.print('*');
            }

            if (wifiOn) {
              lcd.clear();
              lcd.setCursor(0, 0);
              lcd.print("Bluetooth off");
              lcd.setCursor(0, 1);
              lcd.print("WiFi on.");
              lcd.setCursor(0, 2);
              lcd.print("IP address:");
              lcd.setCursor(0, 3);
              lcd.print(WiFi.localIP());

              startServer();
            } else {
              lcd.clear();
              lcd.setCursor(0, 0);
              lcd.print("Connection failed");
            }
          }
          break;
        }
      }
    } else if (key == 'B') {
      bluetoothConnect();
    } else if (key == 'P') {
    }
  }

}
```