ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «АВТОМАТИЗОВАНА СИСТЕМА ОБЛІКУ НА ВИРОБНИЦТВІ З ВИКОРИСТАННЯМ ARDUINO ТА МОВИ ПРОГРАМУВАННЯ C#»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Іван БРАТАШОВ*

Виконав:
здобувач вищої освіти
група ІСДМ-61 Іван БРАТАШОВ

Керівник:
науковий ступінь, вчене звання Оксана ТКАЛЕНКО
к.т.н., доцент

Рецензент:
науковий ступінь, вчене звання _____
Ім’я, ПРІЗВИЩЕ

Київ – 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти магістр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна прорама Інформаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедрою ІСТ
_____Каміла СТОРЧАК

“____” _____2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Браташову Івану Руслановичу
(прізвище, ім’я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Автоматизована система обліку на виробництві з використанням Arduino та мови програмування C#
керівник кваліфікаційної роботи: Оксана ТКАЛЕНКО к.т.н., доцент
(ім’я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від “30 ” жовтня 2025 р. № 467.

2. Строк подання кваліфікаційної роботи «26» грудня 2025 р.

3. Вихідні дані кваліфікаційної роботи:
1. Автоматизація виробництва.
2. Методи автоматизації системи обліку на виробництві.
3. Науково-технічна література.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
1. Дослідження можливостей автоматизації обліку на виробництві.
2. Огляд методів автоматизації обліку виробничого процесу.
3. Аналіз результатів впровадження автоматизованої системи обліку.

5.Перелік ілюстраційного матеріалу: *презентація*

6. Дата видачі завдання «30» жовтня 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Підбір технічної літератури	30.10.2025	
2.	Дослідження виробничих машин	03.11.2025	
3.	Ознайомлення з функціоналом Arduino та C#	12.11.2025	
4.	Розробка автоматизованої системи обліку на виробництві	20.11.2025-28.11.2025	
5.	Висновки по роботі	03.12.2025	
6.	Розробка демонстраційних матеріалів, доповідь	11.12.2025	
7.	Оформлення магістерської роботи	25.12.2025	

Здобувач вищої освіти _____ Іван БРАТАШОВ
(підпис) *(ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Оксана ТКАЛЕНКО
(підпис) *(ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття ступня магістр: 76 стор., 63 рис., 36 джерел.

Мета роботи – розробка автоматизованої системи обліку на виробничому підприємстві.

Об'єкт дослідження – процес автоматизованої системи обліку.

Предмет дослідження – автоматизована система обліку.

Короткий зміст роботи. Ця кваліфікаційна робота слугує ілюстративним прикладом того, як можна створити автоматизовану систему обліку для промислового цеху, яка може автоматично рахувати вироблену продукцію та часи роботи і простою виробничих машин.

У процесі дослідження було звернено увагу на мікроконтролер Arduino, мову програмування C# та C Arduino, а також певні датчики, що необхідні для працездатності описаної системи. Розглядалися їхні можливості, архітектурні рішення та способи взаємодії між ними.

В результаті цього дослідження було зроблено висновки щодо доцільності використання технологій Arduino та C#.

КЛЮЧОВІ СЛОВА: ІНТЕРНЕТ РЕЧЕЙ, АВТОМАТИЗОВАНА СИСТЕМА, ОБЛІК, РОЗРОБКА, БАЗА ДАНИХ, МІЖМАШИННА ВЗАЄМОДІЯ, АВТОМАТИЗАЦІЯ, C#, ARDUINO, ПОЛІГРАФІЯ.

ABSTRACT

Text part of the qualification work for the degree of Master: 76 pages, 63 figures, 36 sources.

The purpose of the work – develop an automated accounting system at a manufacturing enterprise.

The object of the study – the process of an automated accounting system.

The subject of the study – an automated accounting system.

Summary of the work. This qualification work serves as an illustrative example of how to create an automated accounting system for an industrial workshop that can automatically count the products produced and the operating and downtime of production machines.

During the study, attention was paid to the Arduino microcontroller, the C# and C Arduino programming languages, as well as certain sensors that are necessary for the operation of the described system. Their capabilities, architectural solutions and methods of interaction between them were considered.

As a result of this study, conclusions were drawn regarding the feasibility of using Arduino and C# technologies.

KEYWORDS: INTERNET OF THINGS, AUTOMATED SYSTEM, ACCOUNTING, DEVELOPMENT, DATABASE, INTER-MACHINE INTERACTION, AUTOMATION, C#, ARDUINO, PRINTING.

ЗМІСТ

ВСТУП

Актуальність теми. Створення автоматизованої системи обліку на підприємстві дозволяє уникнути помилок і спростити роботу особи відповідальної за облік продукції. Суть у тому, щоб створити електронну базу даних, у якій буде вся інформація про продукцію. Описана в цій дипломній роботі система дозволяє проводити точний облік продукції в реальному часі, а також контролювати час роботи і простою обладнання, що дозволить краще контролювати процес виробництва та виявляти слабкі місця виробничого ланцюга та подальшу ліквідацію недоліків виробництва.

Також, запропонована система дає можливість контролювати час роботи та простою обладнання, даючи можливість записувати моменти, коли обладнання включали, а коли виключали, що дає можливість контролювати виробничий процес.

Мета роботи – Створення автоматизованої системи обліку на виробничому підприємстві, за допомогою мікроконтролера Arduino та комп'ютерних програм, написаних мовою C#.

Для досягнення мети, у магістерській роботі успішно виконано наступні завдання:

– Розробка системи та написання схем;

– написання коду Arduino;

– створення програм мовою C#;

– перевірка працездатності створеної системи.

Об'єкт дослідження – процес автоматизованої системи обліку.

Предмет дослідження – автоматизована система обліку.

Методи дослідження. Під час написання магістерської кваліфікаційної роботи були використані методи написання схем підключення розробленої системи, програмування мікроконтроллера Arduino, написання програм мовою C# та приведення прикладу інтеграції розробленої системи у життя виробництва на базі пакетоформувальної машини, що використовується у

напрямку поліграфії.

Наукова новизна одержаних результатів. У ході дослідження описано новий підхід для створення автоматизованої системи контролю та передачі інформації на центральний комп’ютер, безпосередньо від виробничих машин.

Практична значущість одержаних результатів. Запропонована система забезпечує ефективне рішення проблеми недостачі контролю за виробничим обладнанням, відповідальності операторів обладнання за їх конкретний час роботи, виявлення слабких місць в системі виробництва, для подальшої ліквідації недоліків виробничого ланцюга.

Апробація результатів магістерської роботи. Основні положення і результати магістерської роботи доповідались на науково практичних конференціях, що проходили на базі Державного університету інформаційно-комунікаційних технологій та Державного університету «Житомирська політехника».

1. ОГЛЯД КЛЮЧОВИХ ІНСТРУМЕНТІВ ПРОЄКТУ

1.1 Підбір підходячої моделі Arduino

Arduino (Ардуіно) − апаратна обчислювальна платформа для аматорського конструювання, основними компонентами якої є плата мікроконтролера з елементами вводу/виводу та середовище розробки Processing/Wiring на мові програмування, що є спрощеною підмножиною C/C++. Arduino може використовуватися як для створення автономних інтерактивних об'єктів, так і підключатися до програмного забезпечення, яке виконується на комп'ютері (наприклад: Processing, Adobe Flash, Max/MSP, Pure Data, SuperCollider) [1].

У системі, яка пропонується в цьому магістерському дипломному проєкті, не потрібна потужна модель Arduino, адже завдання Arduino – зчитувати сигнал та відправляти дані на комп'ютер, який відіграє роль бази даних для окремого виробничого цеху.

Для системи, що описується, відмінно підійде модель Arduino NANO. Її завдання – передавати інформацію про роботу машини на комп'ютер у вигляді логів, які потім будуть вноситися до бази даних.

Arduino NANO відрізняється від інших моделей своєю компактністю та відносно дешевою ціною. Для описуваної системи вона цілком підходить.

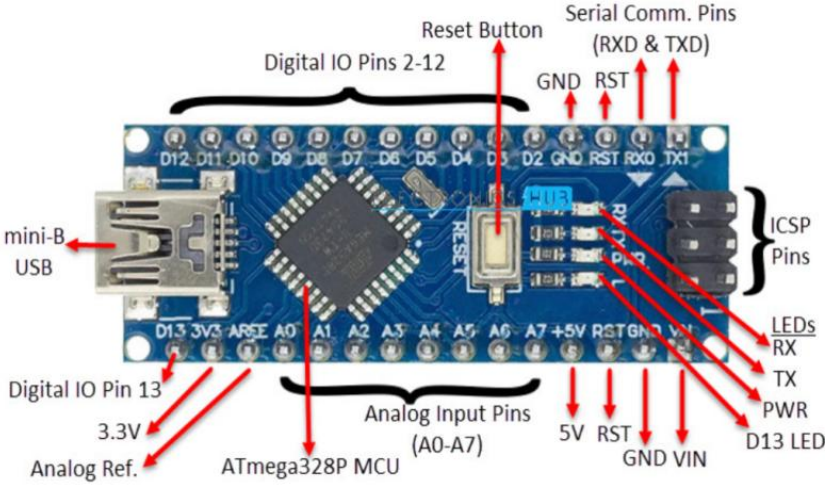

Рис. 1.1 Arduino NANO [2]

1.2 Підбір необхідних датчиків, іншого обладнання та програм для розробки

Для описаної в цій дипломній роботі системи, необхідний будь-який датчик, який працює за принципом кнопки, наприклад – датчик холу, що реагує на магніт, або інфрачервоний датчик, що реагує на появу перешкоди за кілька сантиметрів від нього, або будь-який інший датчик, який працюватиме за принципом кнопки.

Також нам знадобиться твердотільне реле, яке має давати сигнал на Arduino, про запуск або зупинку машини. Твердотільне реле повинно спрацьовувати тоді, коли оператор машини натискатиме кнопку старту процесу роботи, і розриватиме ланцюг, при зупинці машини, тим самим сигналізуючи системі Arduino про моменти запуску та зупинки машини, які Arduino має запам'ятовувати і вказувати в логах, які будуть відправлятися на центральний комп'ютер виробничого цеху. В цій кваліфікаційній дипломній роботі було наведено приклад підключення твердотільного реле з керуючим каналом 220В та силовим каналом 5В. Це зроблено для того, щоби при запуску та зупинці виробничого процесу машини, твердотільне реле пропускало живлення 5В з Arduino на цифровий пін Arduino, відповідальний за контроль вмикання та вимикання виробничого процесу машини.

Ще нам знадобиться комп'ютер, який відіграватиме роль центрального цехового сервера, який прийматиме інформацію від плат Arduino, вбудованих у виробничі машини. Комп'ютер може бути не дуже потужний, але обов'язково з операційною системою Windows 10 або Windows 11. Завдання комп'ютера – запускати програму на C#, яка повинна давати платам Arduino інформацію про поточний час, контролювати виробничий процес, приймати інформацію від плат Arduino, після чого особа відповідальна за цех, повинна вносити дані з логів, виключно через ще одну програму на C#, яка буде описана нижче.

Для написання необхідних для описаної системи програм мовою C#, було використано Microsoft Visual Studio 2026.

Рис. 1.2 Інфрачервоний датчик [3]

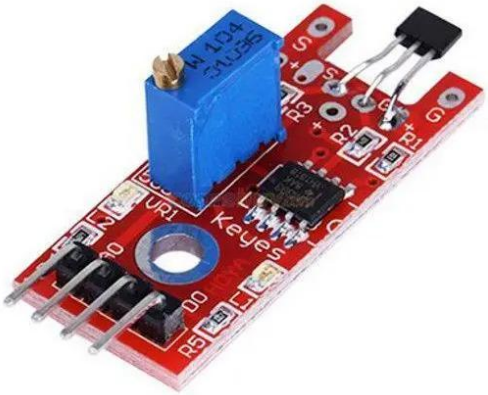

Рис. 1.3 Датчик холу [4]

Рис. 1.4 Твердотільне реле [5]

1.3 Виробнича машина, що застосована в системі автоматизації обліку виробничого підприємства

В даній дипломній роботі було вирішено взяти за основу пакетоформувальну машину, яка використовується в напрямку поліграфії. Ця машина призначена для формування пакетів з поліетиленової плівки.

Рис. 1.5 Пакетоформувальна машина

Для більшої наочності, було вирішено створити спрощену 3D-модель пакетоформувальної машини, яка буде містити всі ключові вузли реальної пакетоформувальної машини.

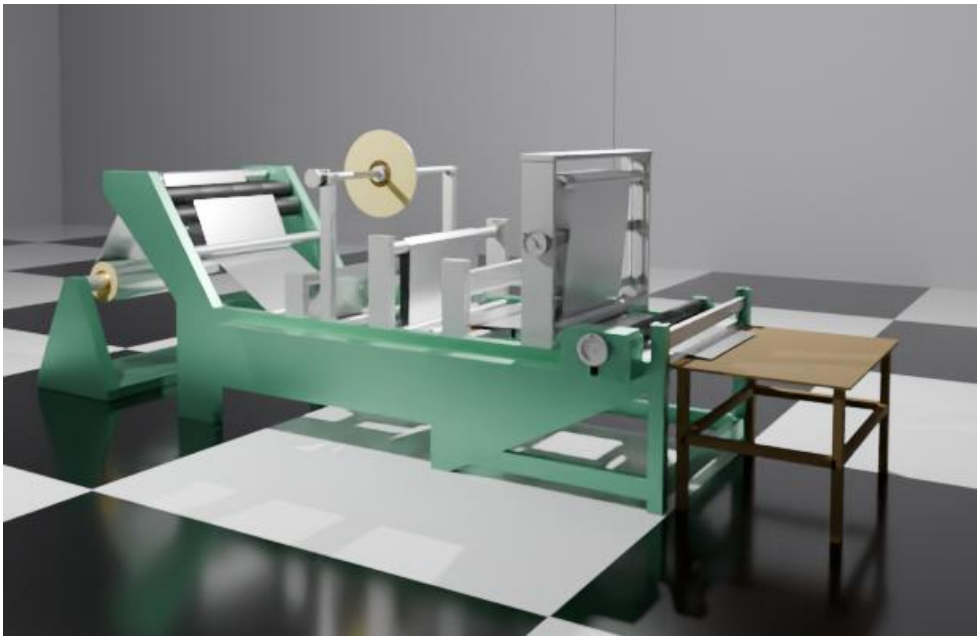

Рис. 1.6 3D-модель пакетоформувальної машини

1.4 Принцип роботи пакетоформувальної машини

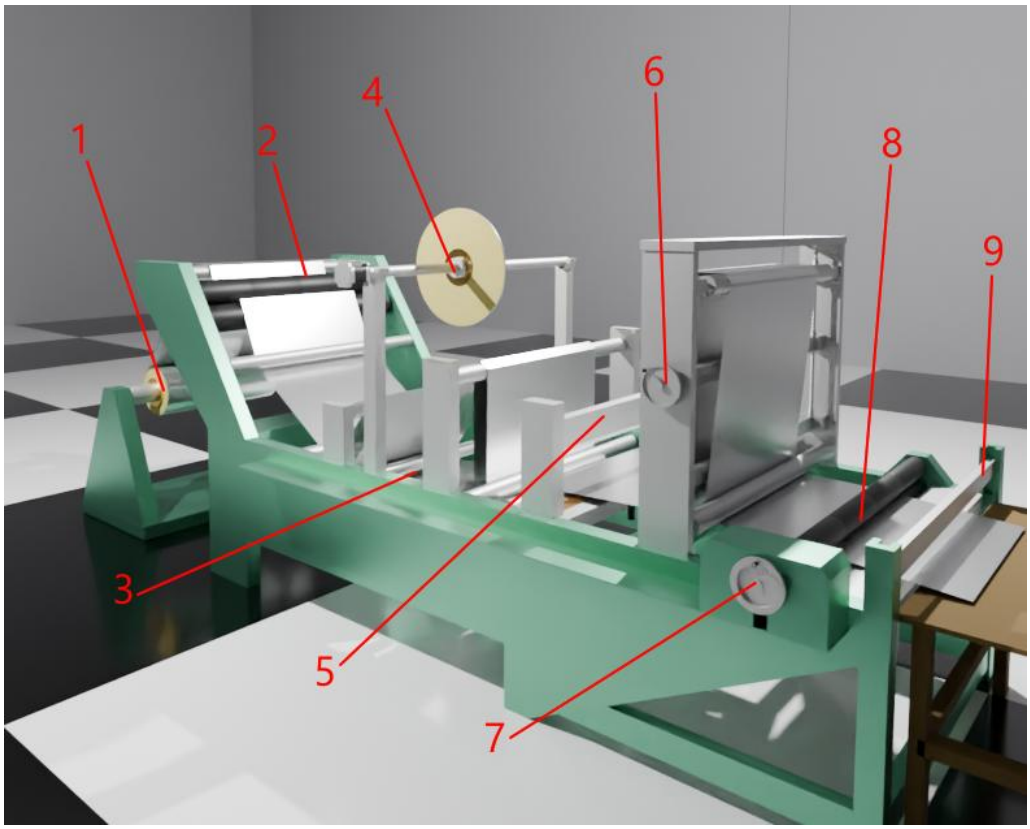

Рис. 1.7 Деталі пакетоформувальної машини

- Вузол розмотки;
- перший протяжний вузол;
- вузол подачі полотна;
- вузол клейкої стрічки;
- вузол пайки;
- вузол налаштування лінії різу;
- другий протяжний вузол;
- протяжні вали;
- гільйотина.

Вузол розмотки – це одна з частин пакетоформувальної машини, основне завдання якої утримання завчасно звернутого рулона з полотном у конкретному положенні, а також для подальшого розмотування рулону, шляхом руху іншими вузлами пакетоформувальної машини в процесі її роботи.

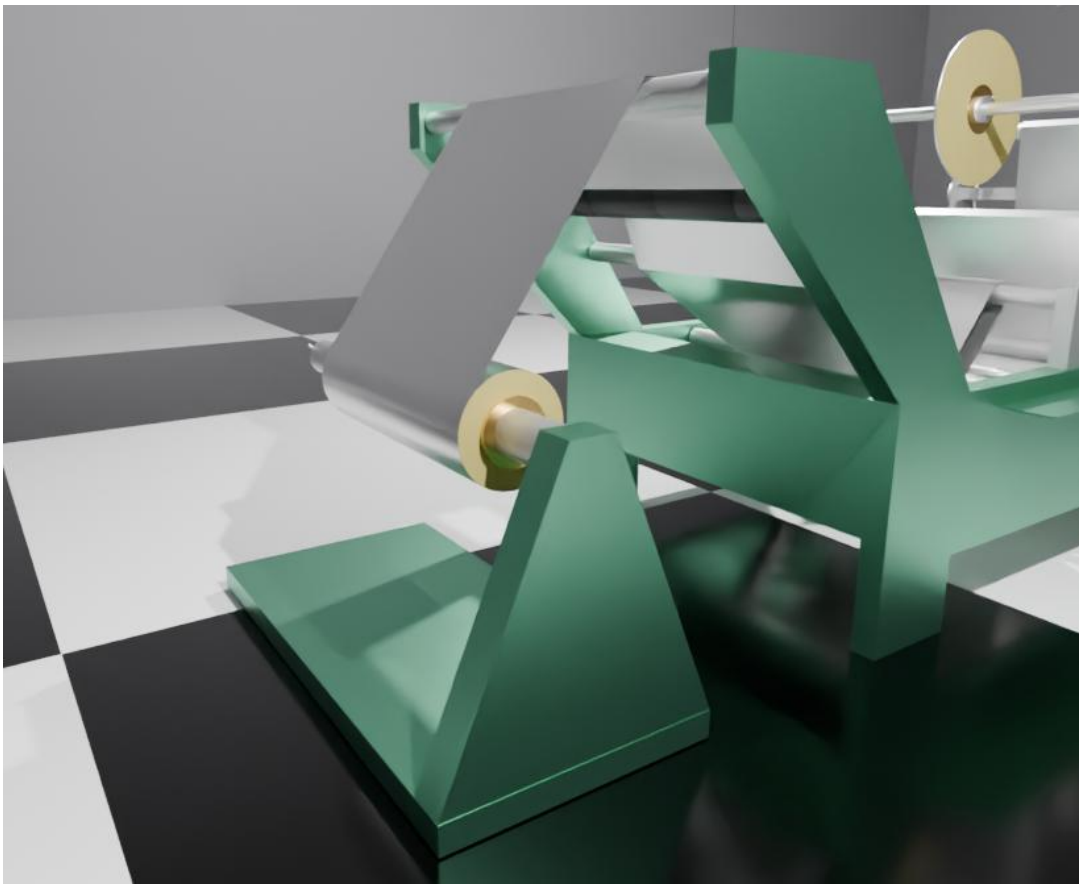

Рис. 1.8 Вузол розмотки (ракурс з лівої сторони)

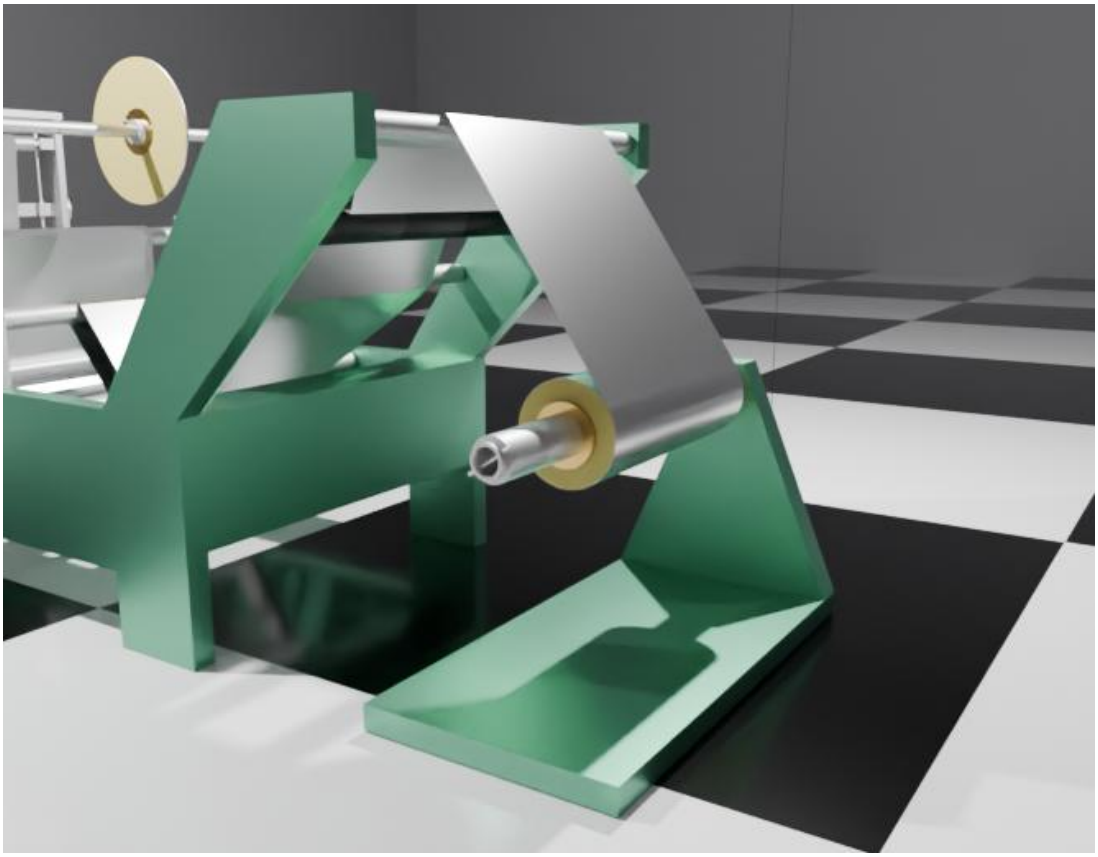

Рис. 1.9 Вузол розмотки (ракурс з правої сторони)

Перший протяжний вузол – це вузол пакетоформуювальної машини, що приймає згорнуту плівку (полотно), що подається з вузлу розмотки, та направляє до наступних вузлів машини.

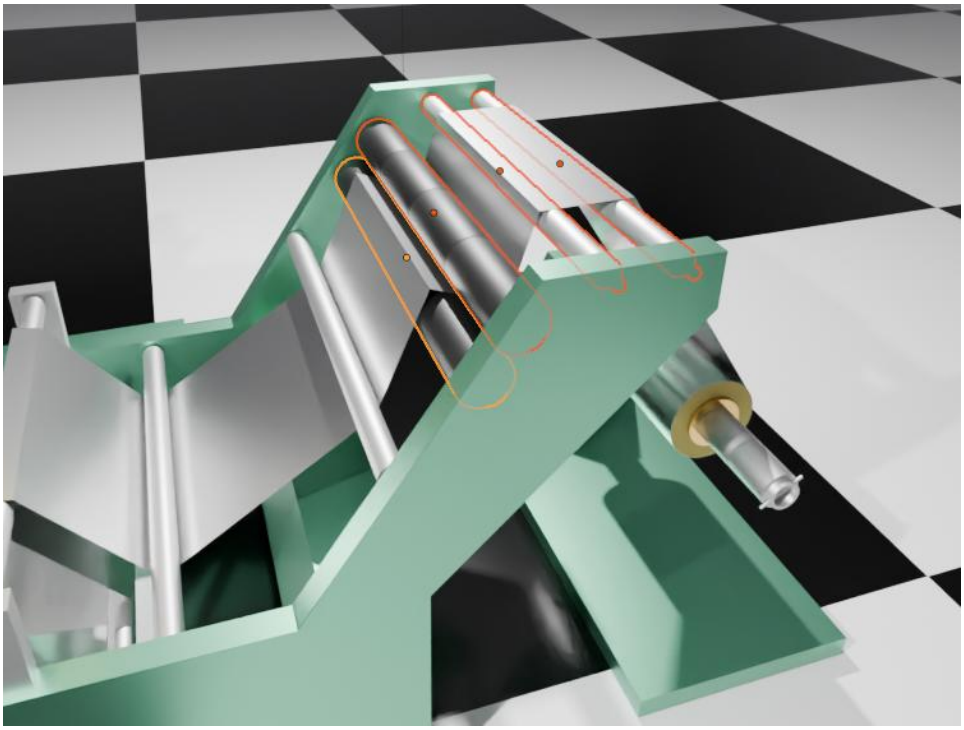

Рис. 1.10 Перший протяжний вузол (ракурс з правої сторони)

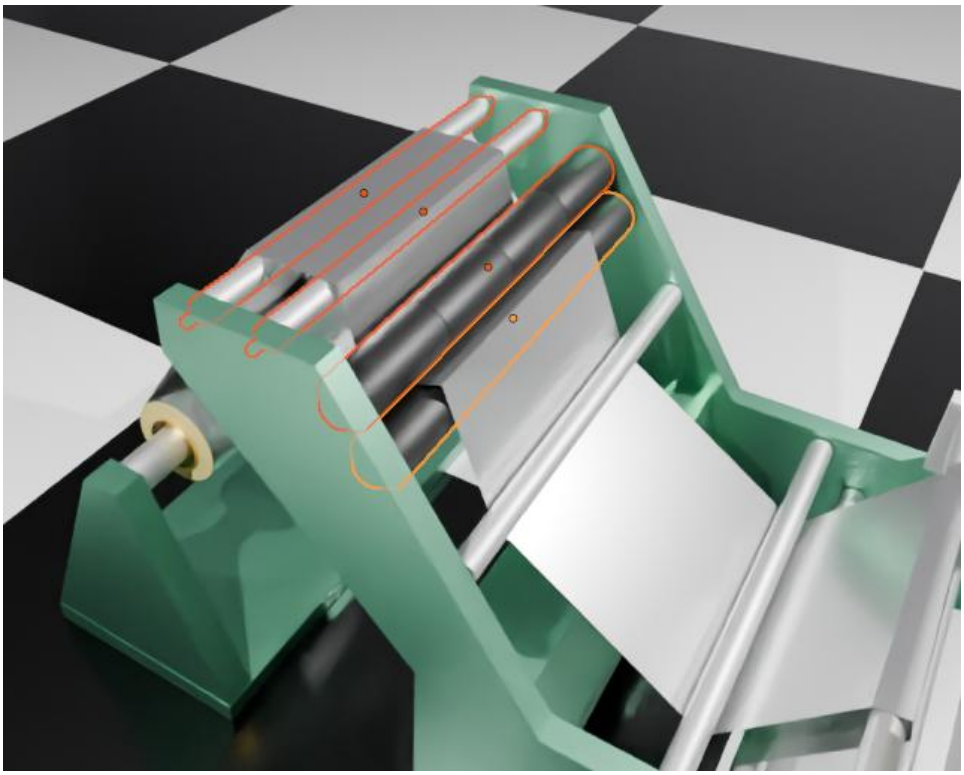

Рис. 1.11 Перший протяжний вузол (ракурс з лівої сторони)

Вузол подачі полотна – це вузол пакетоформувальної машини, задачою якого є відтягувати полотно на певну, налаштовану кількість сантиметрів, котра формує ширину пакета.

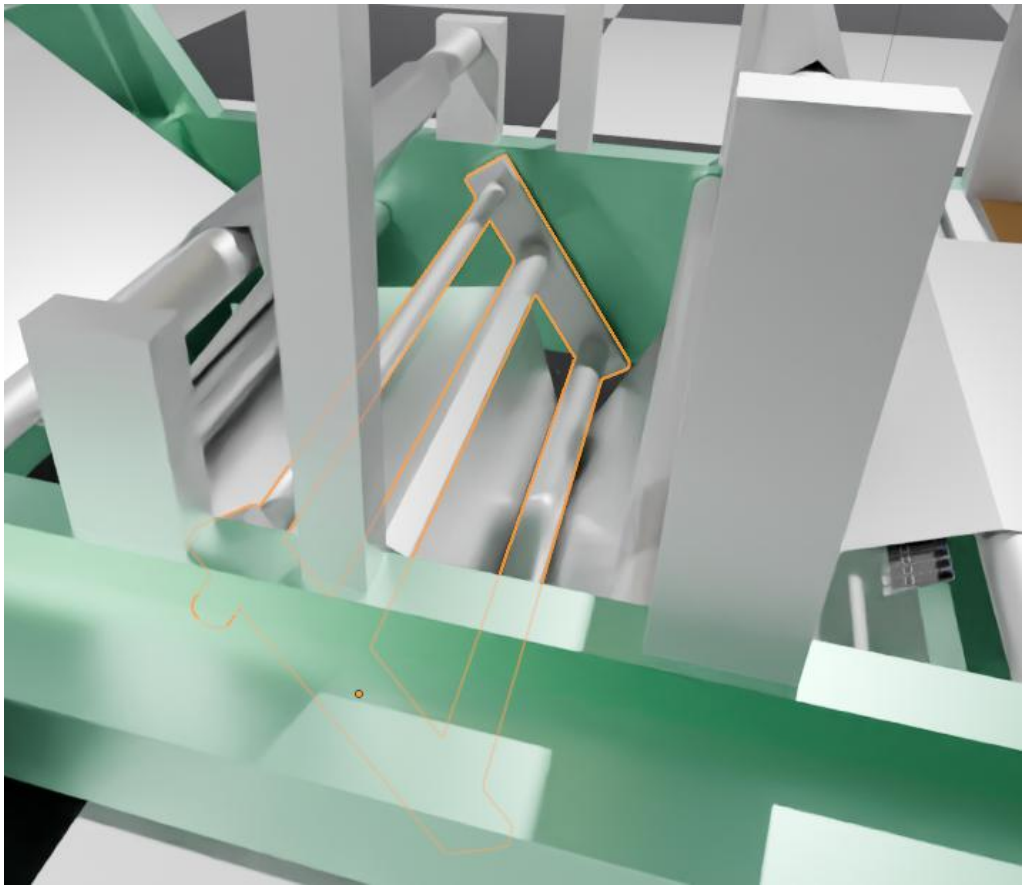

Рис. 1.12 Вузол подачі полотна (ракурс з лівої сторони)

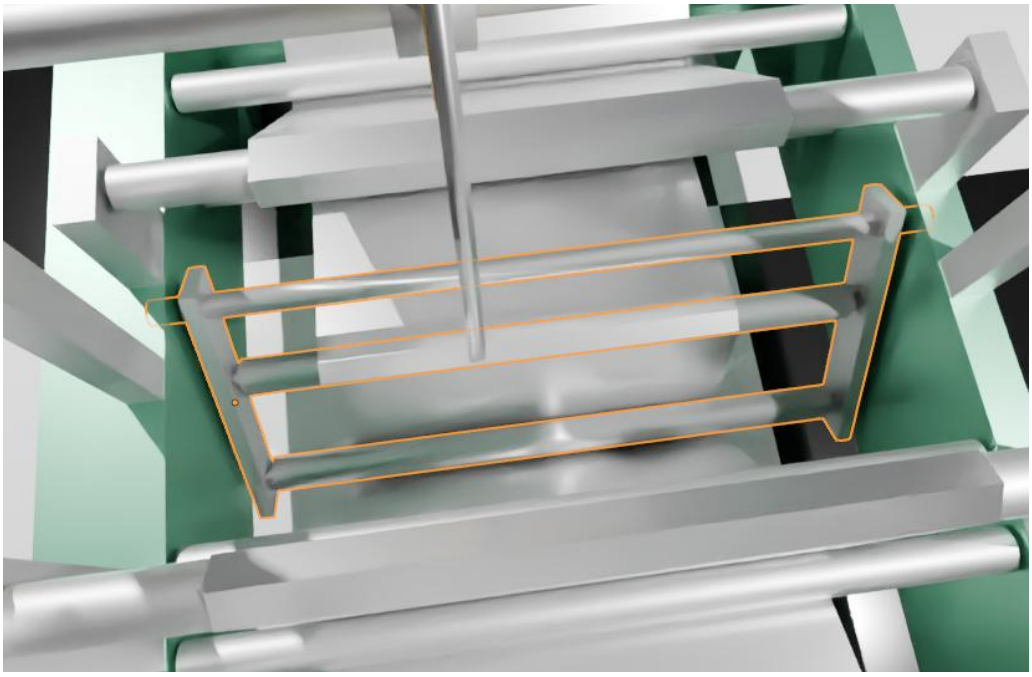

Рис. 1.13 Вузол подачі полотна (ракурс з верхньої сторони)

Вузол клейкої стрічки – це вузол пакетоформувальної машини, який відповідає за нанесення на полотно клейкої стрічки. Цей вузол використовується тоді, коли є замовлення на пакети з клейкою стрічкою.

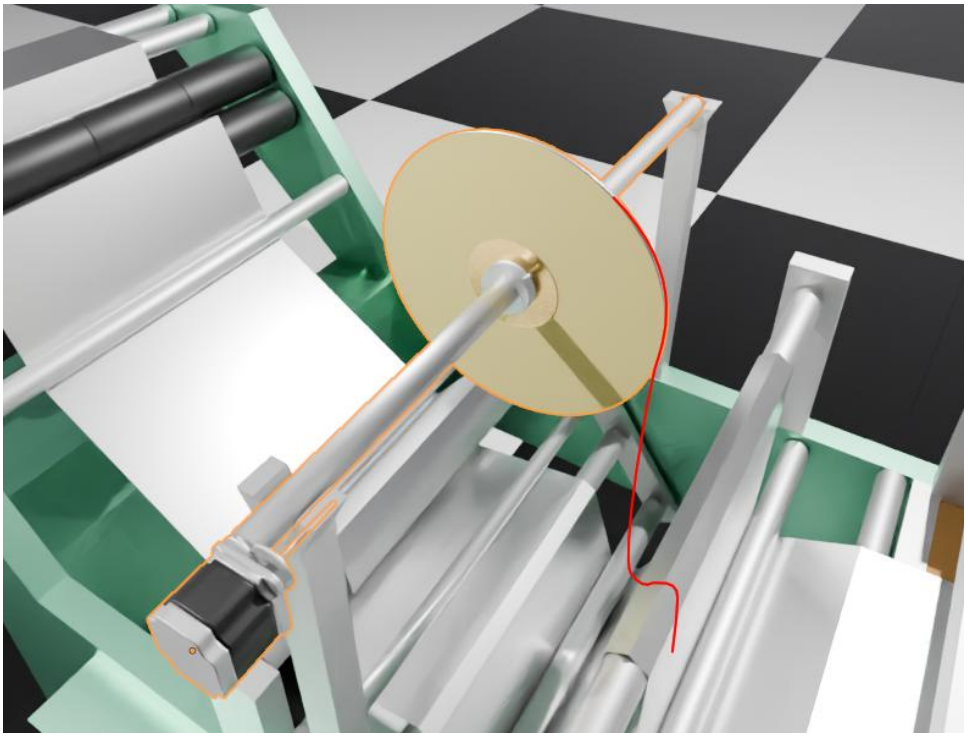

Рис. 1.14 Вузол клейкої стрічки (ракурс з лівої сторони, червоним позначена клейка стрічка, що кріпиться на полотні)

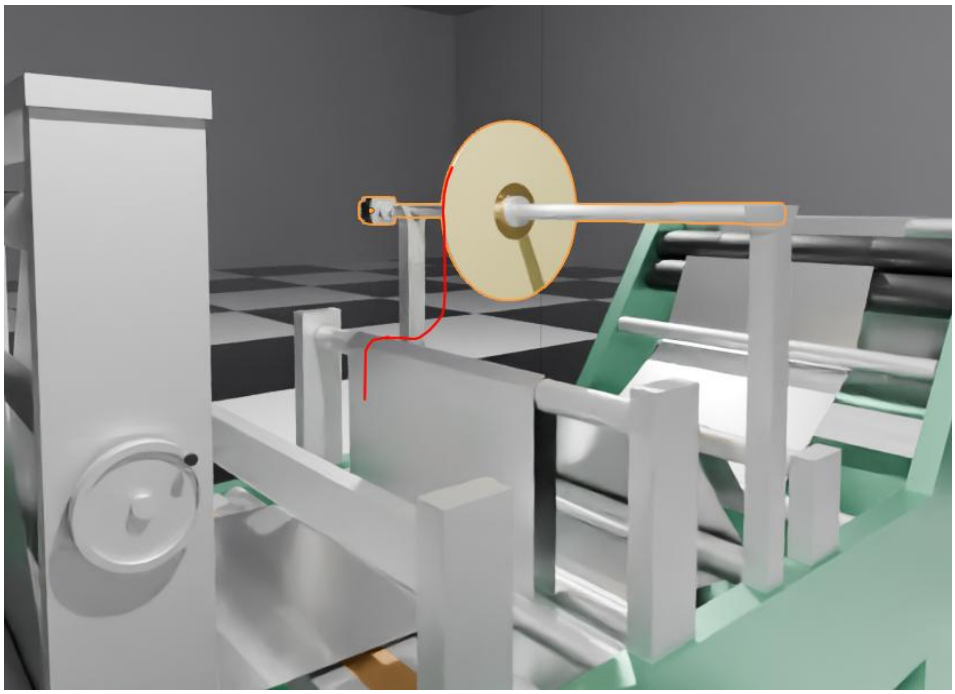

Рис. 1.15 Вузол клейкої стрічки (ракурс з правої сторони, червоним позначена клейка стрічка, що кріпиться на полотні)

Вузол пайки – це вузол пакетоформувальної машини, задачею якого є спаювання згорнутого у два шари полотна таким чином, щоб вийшли стінки пакетів.

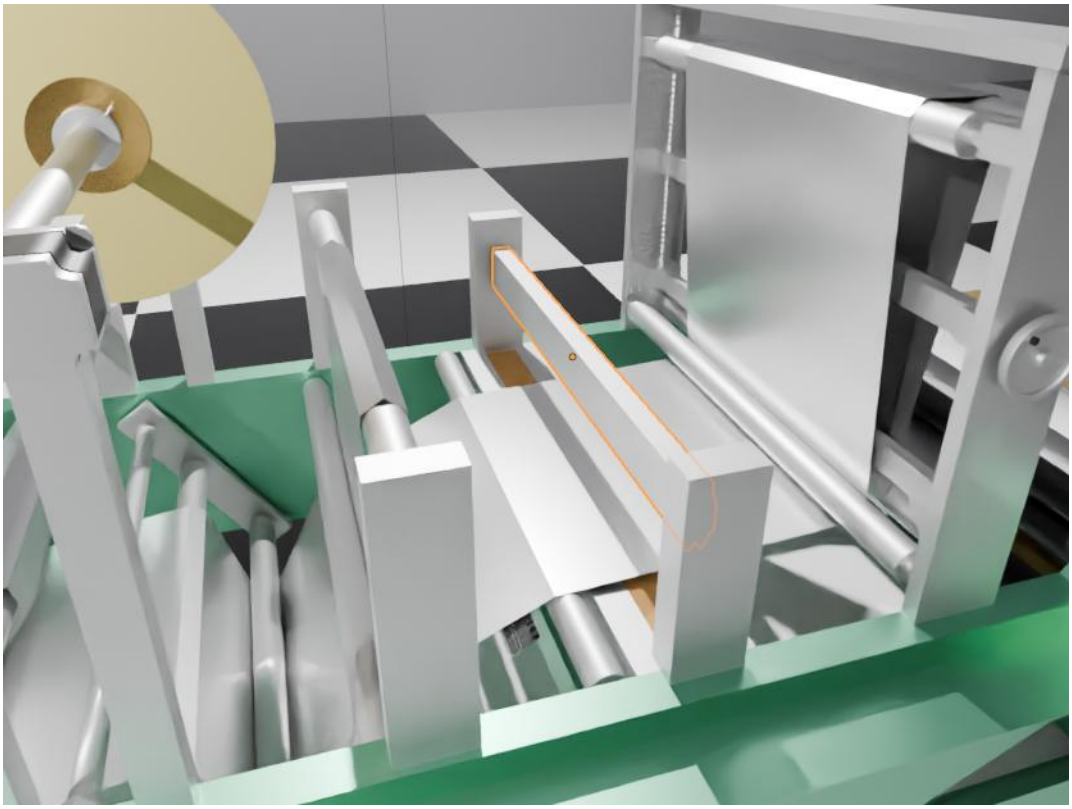

Рис. 1.16 Вузол пайки (ракурс з лівої сторони)

Форма паяльника зроблена таким чином, щоб за один дотик робити одночасно два шви, між якими потім має проходити лінія різу. Таким чином формуються одночасно два краї сусідніх між собою пакетів.

Деякі пакетоформувальні машини мають паяльник з одним елементом дотику, щоб відразу з пайкою різати полотно на окремі пакети, шляхом розплавлення місця, де має відрізатись пакет, але ця машина має інший спосіб формування пакетів.

Температура паяльника залежить від швидкості машини та товщини плівки, з якої виробляються пакети. Чим товще пакет, тим краще спаюються його краї, тому тут температура має бути нижче, аби не перепаяти полотно, та воно не відірвалося. Чим швидше йде подача полотна, тим менше проміжок часу, за який паяльник дотикається по пакету, тому тут температура має бути вижчою. Зазвичай, температура знаходиться в проміжку між 140 °C та 150 °C.

За температурою паяльника стежить термодатчик, що прикріплений до нагрівального елементу паяльника. Системі контролю температури задається еталонне значення температури, система включає нагрів тоді, коли температура менше заданої, та відключає нагрів тоді, коли температура стає більше, ніж задана.

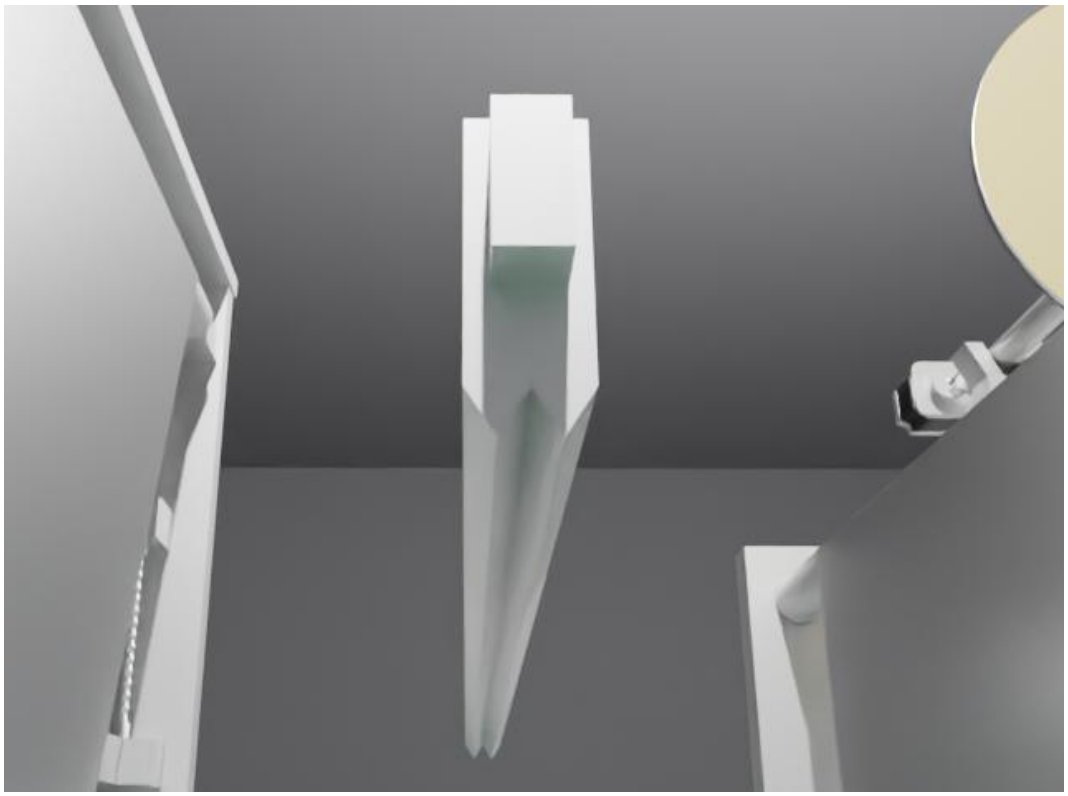

Рис. 1.17 Вузол пайки (форма паяльника)

Температура паяльника залежить від швидкості машини та товщини плівки, з якої виробляються пакети. Чим товще пакет, тим краще спаюються його краї, тому тут температура має бути нижче, аби не перепаяти полотно, та воно не відірвалося. Чим швидше йде подача полотна, тим менше проміжок часу, за який паяльник дотикається по пакету, тому тут температура має бути вижчою. Зазвичай, температура знаходиться в проміжку між 140 °C та 150 °C.

За температурою паяльника стежить термодатчик, що прикріплений до нагрівального елементу паяльника. Системі контролю температури задається еталонне значення температури, система включає нагрів тоді, коли температура менше заданої, та відключає нагрів тоді, коли температура стає більше, ніж задана.

Також, під паяльником має проходити термопрокладка, яка має прижимати з нижньої сторони плівку до паяльника. Це зроблено для того, аби тепло від паяльника не розповсюджувалося на корпус машини, а також м’якость прокладки потрібна для того, щоб не псувалась форма паяльника.

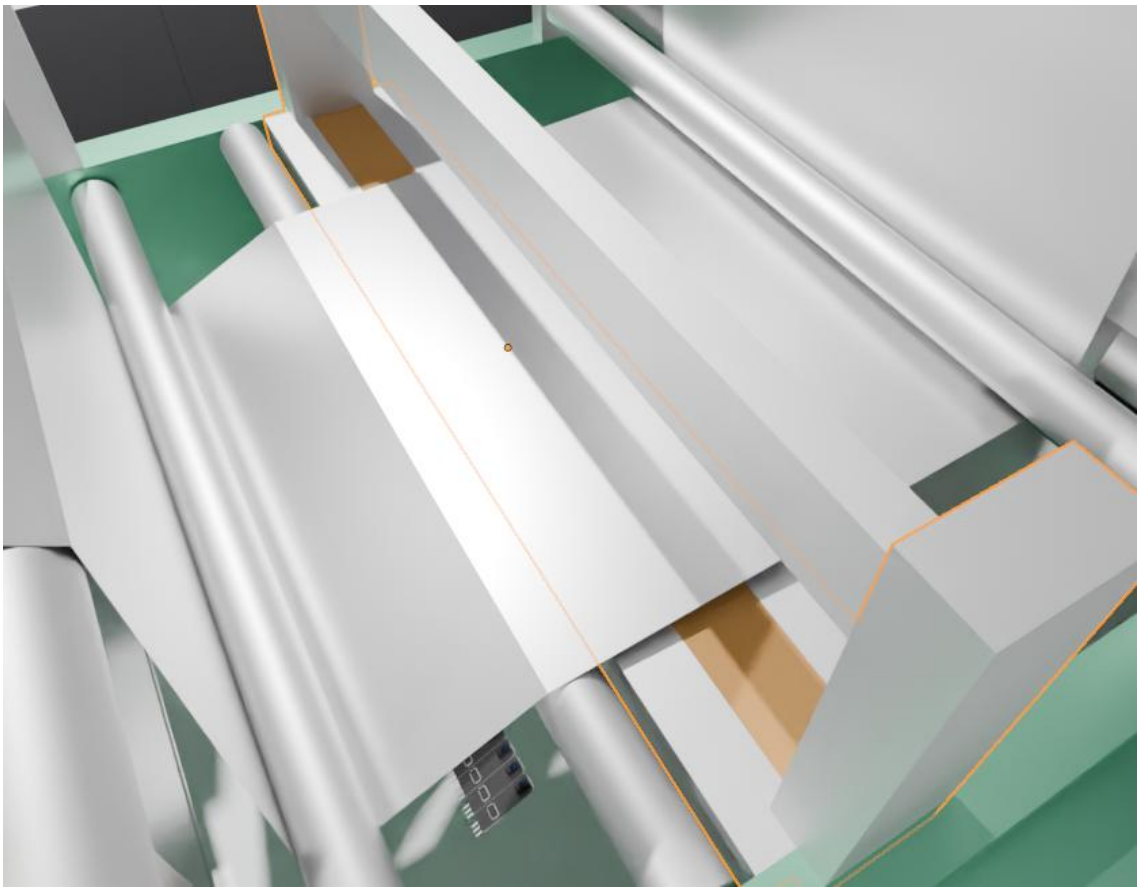

Рис. 1.18 Вузол пайки (термопрокладка має коричневий колір)

Швидкість паяльника залежить від швидкості решти машини, оскільки він рухається синхронно з гільйотиною, яка має нарізати полотно на окремі пакети. Чим швидше темп машини, тим швидше рухається паяльник.

Бувають ситуації, коли треба екстрено зупинити машину, наприклад, коли починає вироблятись бракована продукція та оператор не може в процесі роботи машини впоратись з вирішенням цієї проблеми. В таких ситуаціях треба обов’язково зупиняти машину лише тоді, коли паяльник піднятий, бо інакше існує серйозний ризик перепаяти полотно, розплавивши його у місці спайки. Якщо таке сталося, то доводиться протягувати полотно заново, що робиться доволі довго, чим витрачається зайвий час.

Вузол налаштування лінії різу – це вузол пакетоформувальної машини, що відповідає за налаштування місця, де гільйотина має різати полотно на окремі пакети. Це місце, де має проходити лінія різу, знаходиться між двома швами, які формує вузол пайки. Таким чином, формуються окремі поліетиленові пакети.

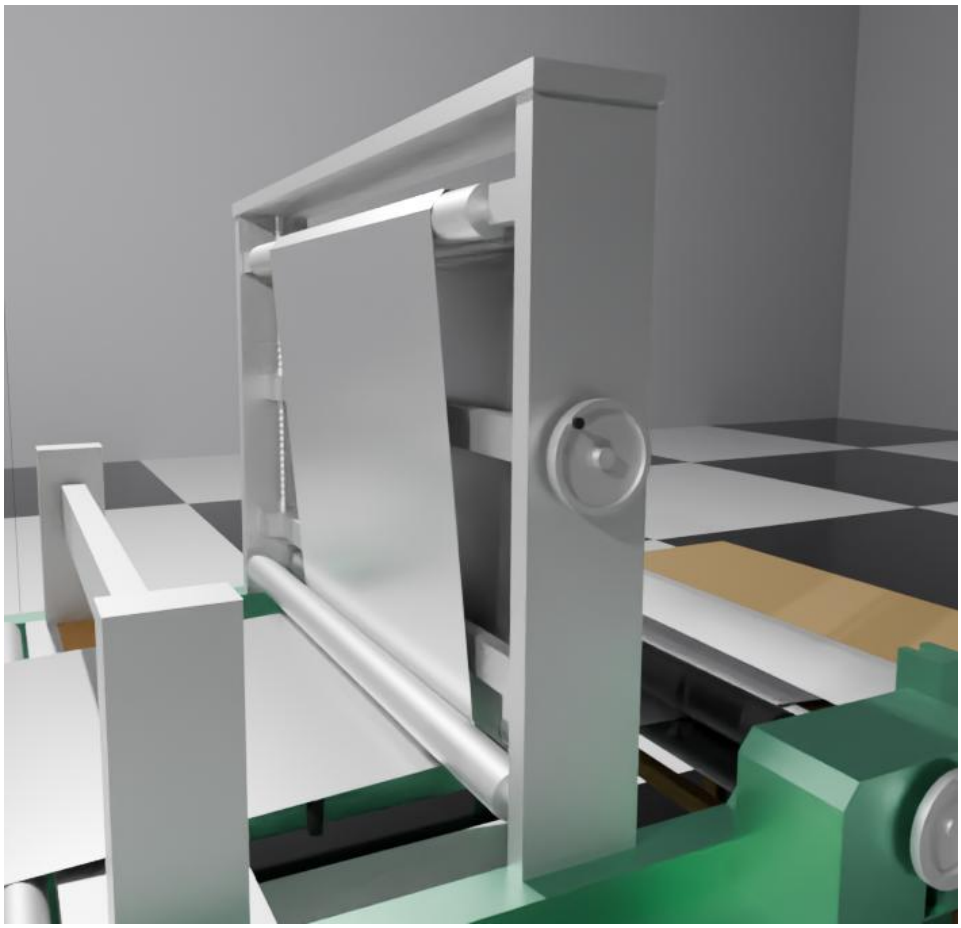

Рис. 1.19 Вузол налаштування лінії різу (ракурс з лівої сторони)

Він складається з рами, всередині якої знаходяться дві різьбові шпильки, які можна обертати вручну за допомогою вентилів. по яким вгору та вниз їздять вали, що опускаються або підіймаються. Таким чином, полотно на лінії різу їздить вліво чи вправо, що можна регулювати вентилем на вузлі налаштування лінії різу.

Цей вузол є одним із найбільш затребуваних в обслуговуванні, тому що лінія різу доволі часто з'їжджає і з ним доводиться дуже часто взаємодіяти, щоб налагодити лінію різу. Особливо часто це трапляється тоді, коли формат пакетів або занадто великий, або занадто маленький.

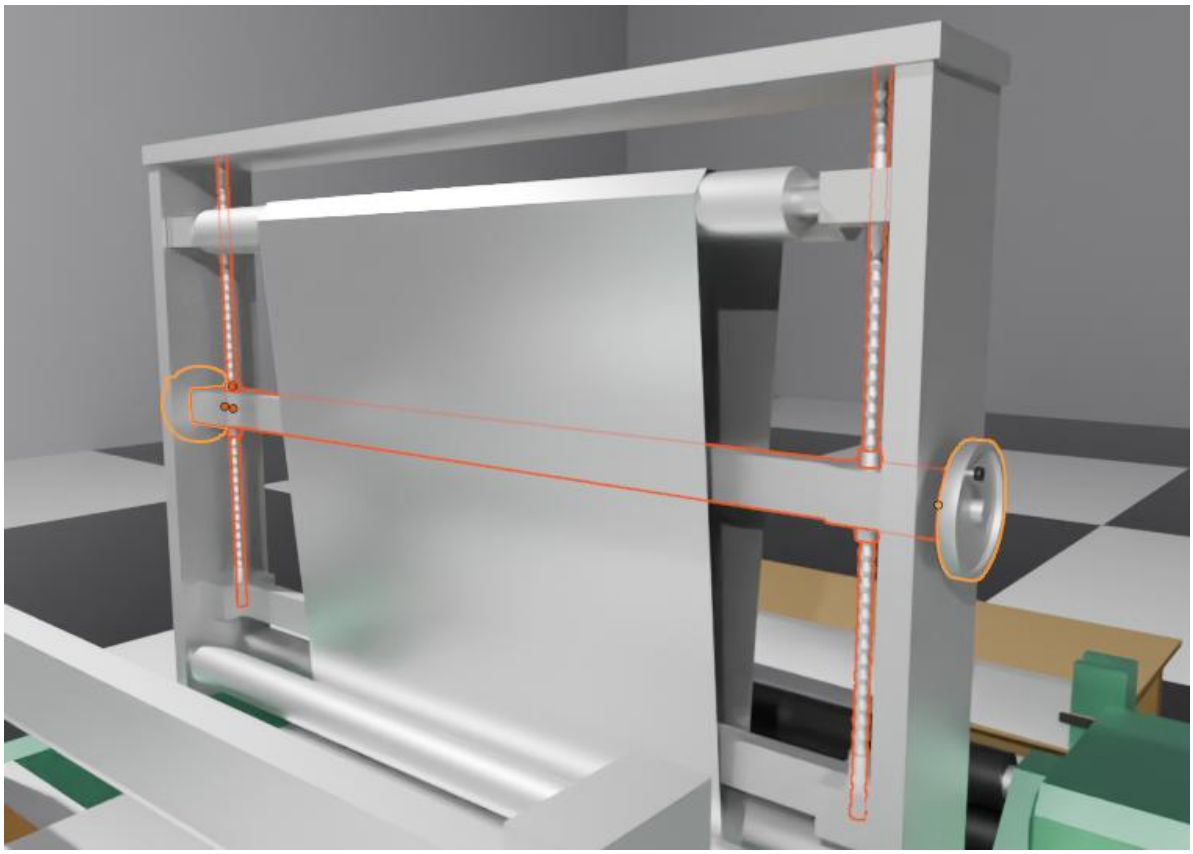

Рис. 1.20 Вузол налаштування лінії різу (з виділеним механізмом підйому та опускання регулювальних валів)

Саме регулювальні вали відповідають за регулювання положення полотна на лінії різу. Вони складаються з двох валів, вставлених в раму, яка в свою чергу кріпиться на різьбових шпильках, які при обертанні змушують цю конструкцію переміщуватись.

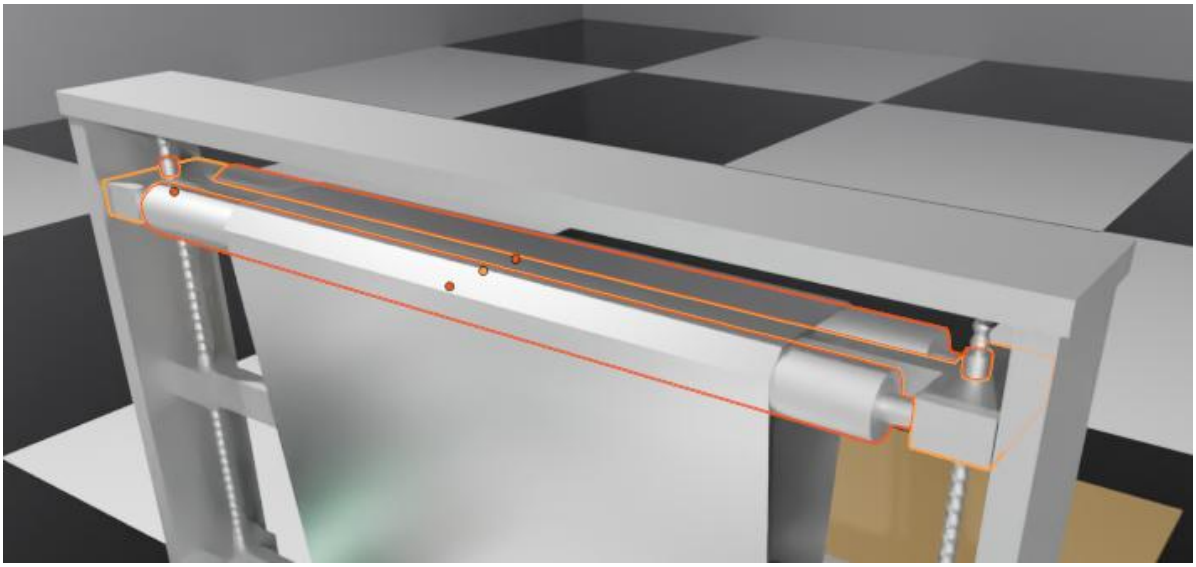

Рис. 1.21 Регулювальні вали у вузлі налаштування лінії різу

Другий протяжний вузол – це вузол, який відповідає за протягування полотна на лінію різу за допомогою протяжних валів. Протяжні вали зроблені з гуми, щоб було краще зчеплення з полотном.

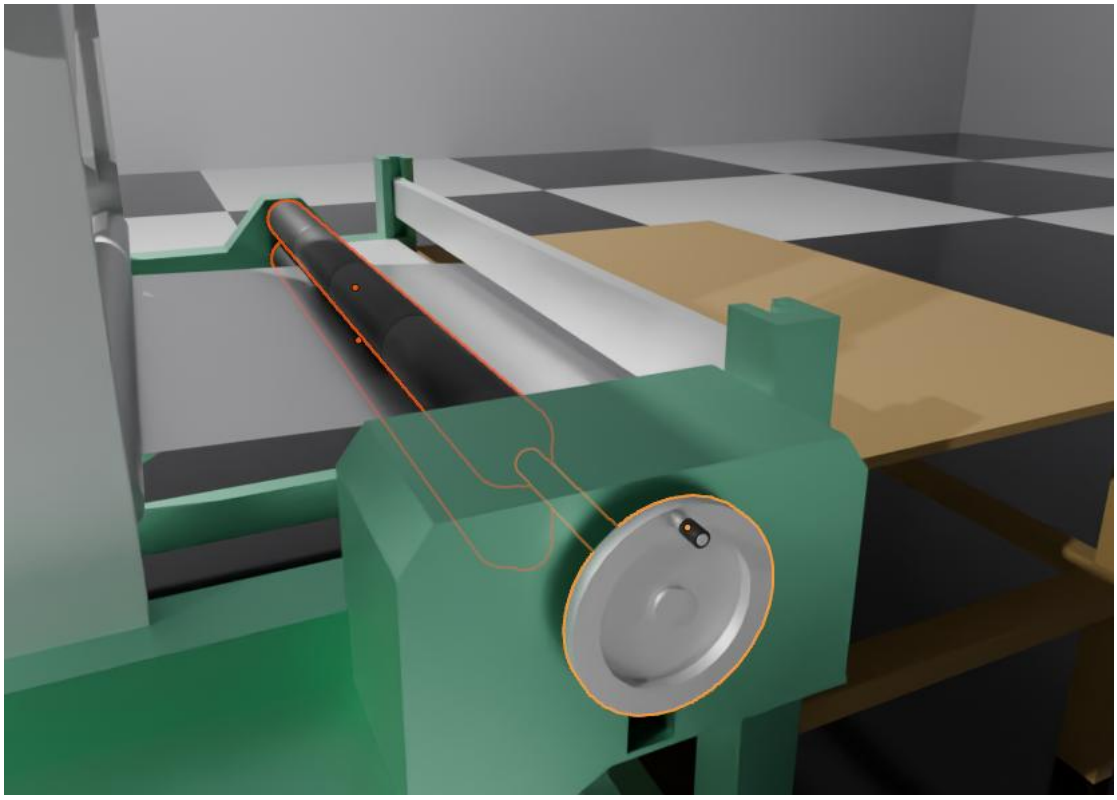

Рис. 1.22 Другий протяжний вузол (ракурс з лівої сторони)

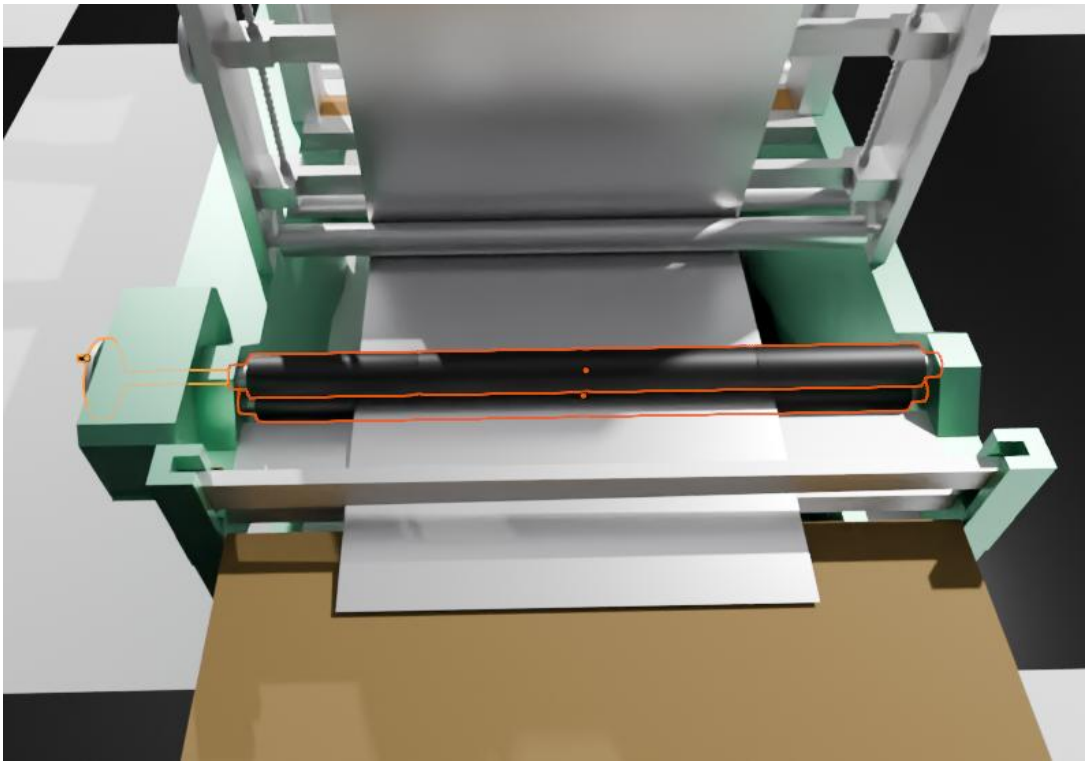

Рис. 1.23 Другий протяжний вузол (ракурс з верхньої сторони)

Гільйотина – це вузол пакетоформуючої машини, що відповідає за нарізку полотна на окремі пакети. Це останній вузол пакетоформувальної машини, відповідаючий за створення готової продукції.

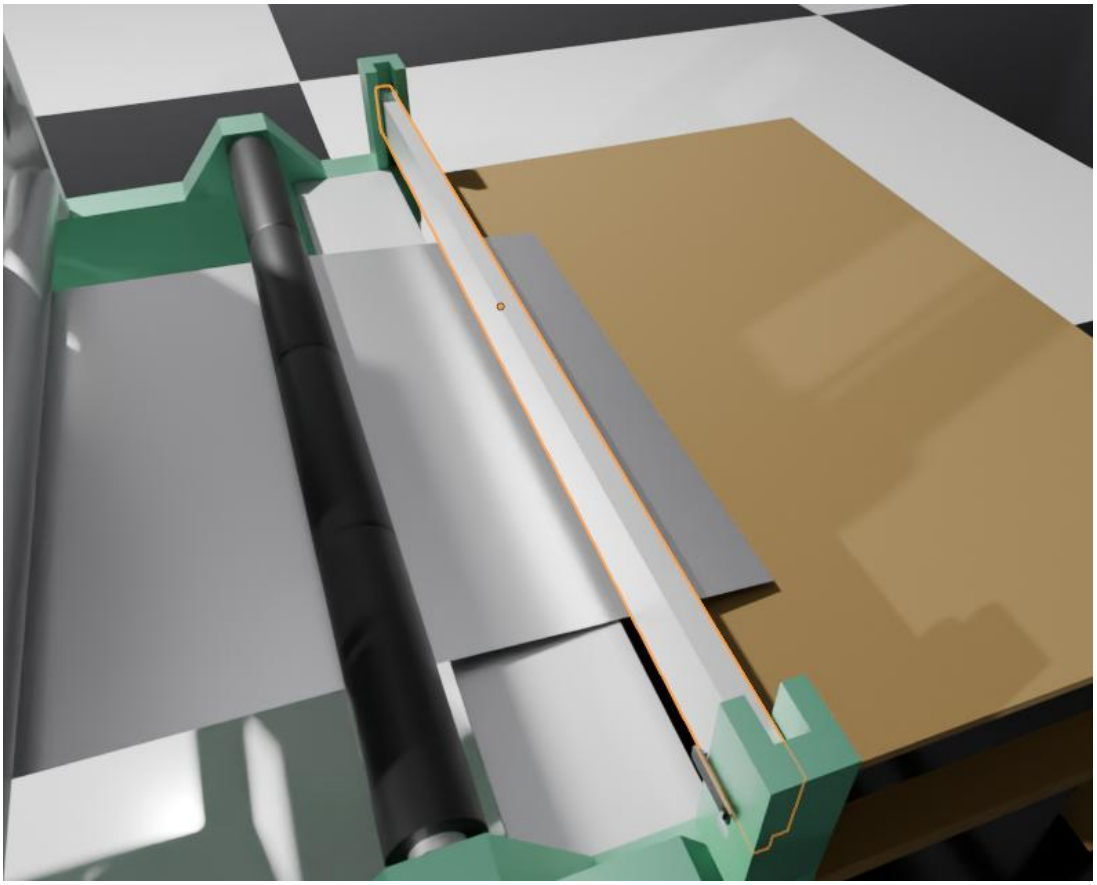

Рис. 1.24 Гільйотина (ракурс з верхньої сторони)

Для того, щоб гільйотина мала змогу нормально різати, вона має бути злегка перекошена, щоби емітувати ефект ножиць між самою гільйотиною зверху та корпусом пакетоформувальної машини знизу.

Гільйотина рухається синхронно з паяльником, тому по положенню гільйотини можна визначати й положення паяльника, оскільки їх обох рухає один двигун. По правилам безпеки, машина має зупинятися лише тоді, коли гільйотина має піднятий стан, оскільки в цей момент паяльник теж має бути в піднятому стані, що було описано вище.

Краї гільйотини мають направляючі колоди, що дозволяють рухатись гільйотині не змінюючи її траєкторію. Направляючі колоди потрібно змащувати олією щоби уникнути зайвого зносу.

Рис. 1.25 Гільйотина (ракурс, демонструючий перекошеність гільйотини)

Лінія різу – це лінія, куди б’є гільйотина, нарізаючи полотно на окремі, готові пакети.

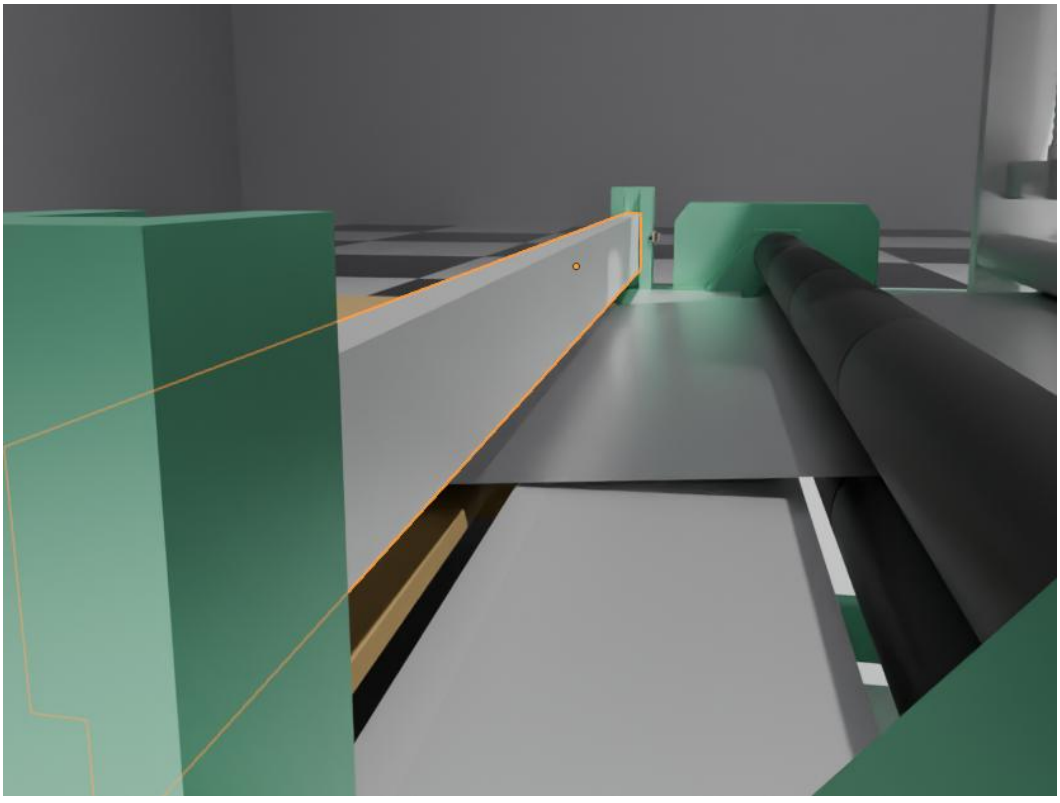

Рис. 1.26 Лінія різу

2. РОЗРОБКА ПРОЄКТУ АВТОМАТИЗАЦІЇ СИСТЕМИ ОБЛІКУ ВИРОБНИЧОГО ПРОЦЕСУ

2.1 Створення схеми автоматизації системи обліку виробничого процесу

В цій магістерській дипломній роботі було вирішено описати розроблену систему контролю виробляємої продукції, на основі виробничого цеху, що виробляє продукцію напрямку поліграфії, а саме поліетиленові пакети.

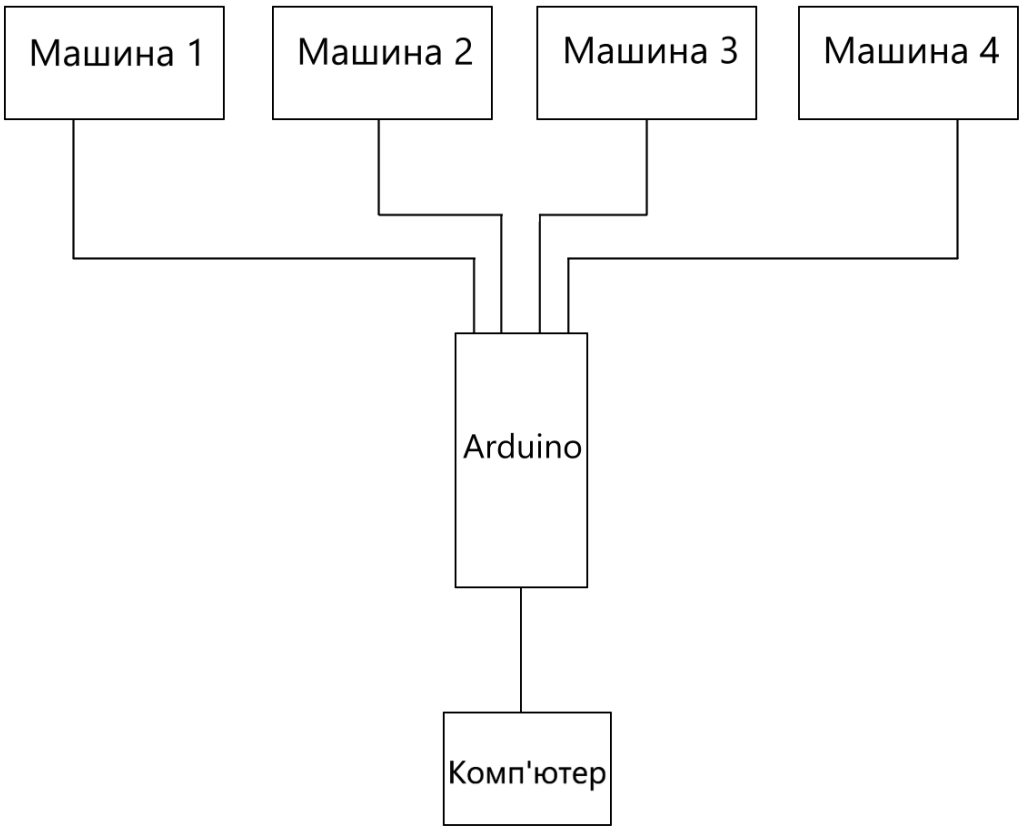

Рис. 2.1 Спрощена схема роботи системи

Система являє собою комп’ютер, який виступає в ролі бази даних цеху, мікроконтролера Arduino, який виконує основну роль для збору інформації, що поступатиме від датчиків, які розташовані на машинах, а також самі інфрачервоні датчики чи датчики Холлу, які працюють за принципом кнопки.

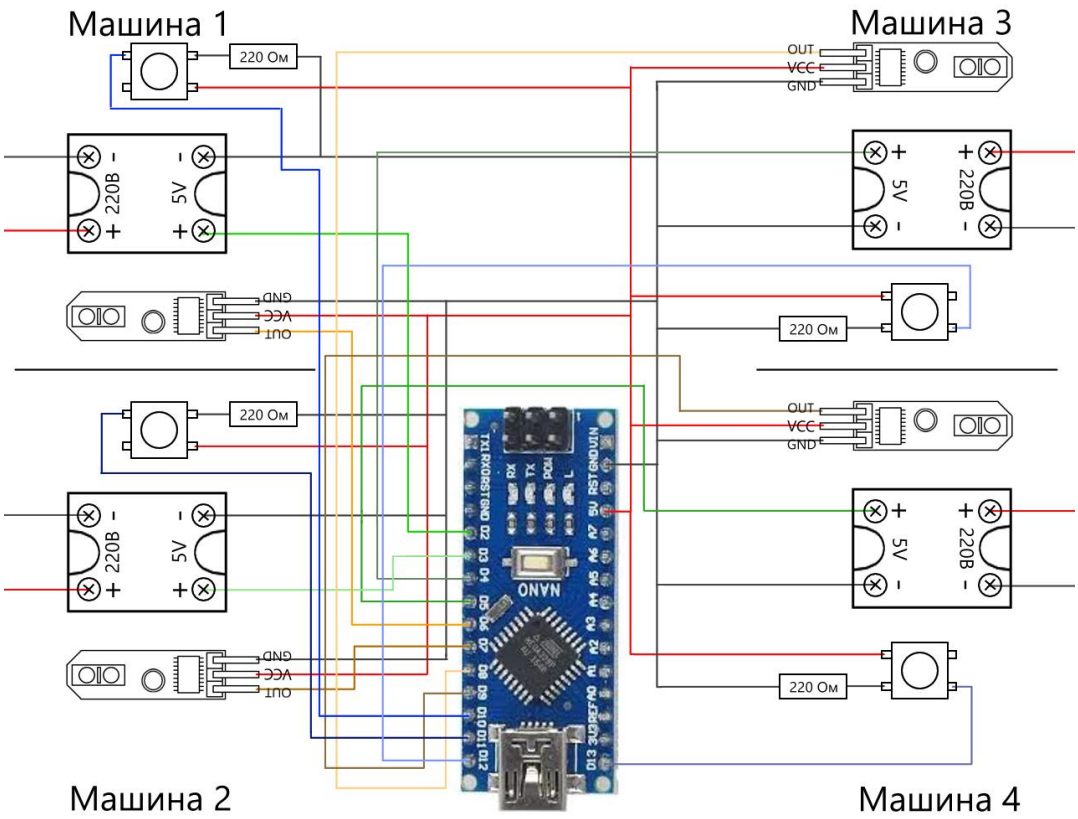

Рис. 2.2 Схема підключення датчиків

На схемі зображено підключення датчиків. Система отримує сигнал про запуск та зупинку машини через твердотільне реле, яке має управляючий канал 220В та підключається послідовно після тумблеру запуску та зупинки роботи машини, де при старті процесу виробництва на машині, струм йде через канал 220В, що запускає проток струму на каналі 5В, який дає сигнал для Arduino, що процес виробництва стартував.

Таким-же чином, зупинка виробничого процесу зупиняє проток струму на каналі 220В, чим закриває канал 5В та розриває сигнал, через що Arduino розуміє, що система зупинена та виводить потрібну строку в монітор порту.

Інфрачервоні датчики, що зображені на схемі підключення датчиків, відповідають за рахунок кількості виробленої продукції. В системі пакетоформувальної машини, інфрачервоний датчик можна прикріпити до корпусу пакетоформувальної машини таким чином, щоб інфрачервоний датчик був повернутий в бік гільйотини та реагував на кожний її рух вниз, таким чином рахуючи кожен відрізаний від решти полотна окремий пакет.

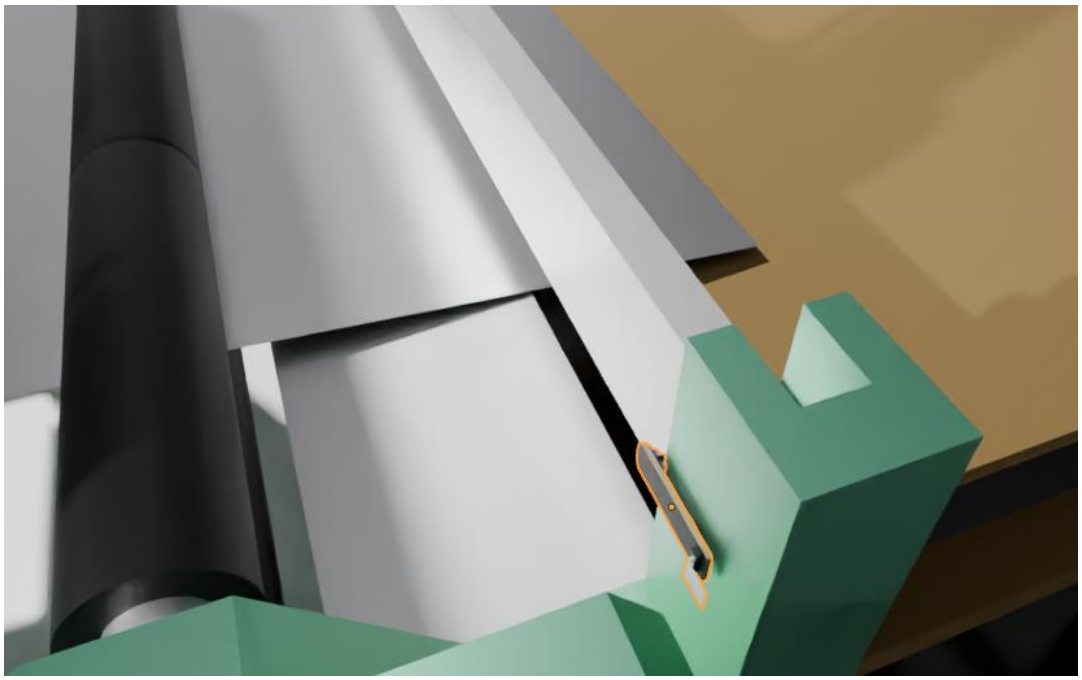

Рис. 2.3 Приклад положення інфрачервоного датчику

Також нам знадобиться звичайна кнопка, яку можна встановити на пульті керування машиною. Ця кнопка буде відповідати за створення рядку в лог-файлі, в якому мають бути прочерки замість чисел кількості продукції та інших колонок. Ці прочерки означають кінець заказу. Цю кнопку має натискати оператор власноруч, коли він зробив достатньо продукції аби закрити замовлення, щоби наглядно виділити в таблиці кінець замовлення, як облегшення роботи для відповідальної особи, яка повинна підраховувати підсумки у кінці зміни.

2.2 Написання програмного коду для Arduino NANO

Для роботи системи, потрібно написати код для плати Arduino NANO. Задачею плати Arduino є збирання інформації з датчиків, що будуть встановлені на виробничих машинах та писати логи в монітор порту.

В коді для Arduino було вирішено прописати окрім основної виробничої машини, запасні місця. Місця, які дозволять в майбутньому приєднати до системи до чотирьох машин. Щоб приєднати більше машин, потрібно взяти

модель Arduino, яка має більше пінів, на кшталт Arduino Uno, чи Arduino MEGA2560, проте для цього прикладу вистачить і Arduino NANO. Також, в коді прописаний стандартний час «00:00:00», який буде замінюватись на поточний час, програмою «ArduinoLogger», написаний на мові C#, яка буде встановлена на центральному комп’ютері цеху та забезпечувати працездатність системи. В коді, для кожної з чотирьох машин прописані три піна, до яких має бути приєднано три датчика, які мають працювати по принципу кнопки. Один пін має відповідати за відстеження початку та зупинки процесу виробництва продукції на машині, наприклад через твердотільне реле, яке буде вмикатись при запуску робочого процесу на машині та вимикатись при зупинці робочого процесу на машині. Другий пін має відповідати за рахунок кількості виробленої продукції, рахуючи свої спрацьовування. Третій пін має відповідати за кінець замовлення, спрацьовуючи по кнопці, яку має натискати оператор тоді, коли він зробив достатньо продукції для закривання замовлення.

В системі проблеми з розпізнаванням кирилічного тексту, тому писати доведеться лише латиницею.

Код Arduino для автоматизованої системи обліку наведений нижче.

```
const int machine1Pin = 2;
const int machine2Pin = 3;
const int machine3Pin = 4;
const int machine4Pin = 5;

const int button1Pin = 6;
const int button2Pin = 7;
const int button3Pin = 8;
const int button4Pin = 9;

const int orderEndButton1 = 10;
const int orderEndButton2 = 11;
```

```
const int orderEndButton3 = 12;
const int orderEndButton4 = 13;

String currentTime = "00:00:00";  // Час за змовчуванням
int productCount[4] = {0, 0, 0, 0};
bool machineState[4] = {false, false, false, false};

void setup() {
  Serial.begin(9600);

  pinMode(machine1Pin, INPUT);
  pinMode(machine2Pin, INPUT);
  pinMode(machine3Pin, INPUT);
  pinMode(machine4Pin, INPUT);

  pinMode(button1Pin, INPUT_PULLUP);
  pinMode(button2Pin, INPUT_PULLUP);
  pinMode(button3Pin, INPUT_PULLUP);
  pinMode(button4Pin, INPUT_PULLUP);

  pinMode(orderEndButton1, INPUT_PULLUP);
  pinMode(orderEndButton2, INPUT_PULLUP);
  pinMode(orderEndButton3, INPUT_PULLUP);
  pinMode(orderEndButton4, INPUT_PULLUP);

  Serial.println("Gotovo do roboty.");
}

void loop() {
  // Отримуємо поточний час від комп'ютера через Serial
```

```
    if (Serial.available()) {
      currentTime = Serial.readStringUntil('\n');
      Serial.flush();
    }

    checkMachine(machine1Pin, button1Pin, orderEndButton1, 0, "Mashina_1");
    checkMachine(machine2Pin, button2Pin, orderEndButton2, 1, "Mashina_2");
    checkMachine(machine3Pin, button3Pin, orderEndButton3, 2, "Mashina_3");
    checkMachine(machine4Pin, button4Pin, orderEndButton4, 3, "Mashina_4");

    delay(500);
  }

  void checkMachine(int machinePin, int buttonPin, int orderEndPin, int index, const char* machineName) {
    bool currentState = digitalRead(machinePin) == HIGH; // HIGH = увімкнено, LOW = вимкнено
    delay(50); // Антидребезг

    if (digitalRead(machinePin) != currentState) return; // Перевірка стабільності сигналу

    if (currentState != machineState[index]) {
      if (!currentState) {
        // Машина вимикається, виводимо кількість продукції
        logProductCount(machineName, productCount[index]);
```

```
            productCount[index] = 0; // Зкидуємо лічильник
        }
        machineState[index] = currentState;
        logMachineEvent(machineName, currentState);
    }

    // Якщо машина працює, наглядаємо за кнопкою
    if (currentState) {
        static bool lastButtonState[4] = {false, false, false, false};
        bool buttonState = digitalRead(buttonPin) == LOW; // LOW = натискання
        delay(50); // Антидребезг кнопки

        if (digitalRead(buttonPin) != buttonState) return; // Перевірка стабільності сигналу

        if (buttonState && !lastButtonState[index]) {
            productCount[index]++;
        }
        lastButtonState[index] = buttonState;
    }

    // Перевіряємо кнопку завершення заказу
    static bool lastOrderButtonState[4] = {false, false, false, false};
    bool orderButtonState = digitalRead(orderEndPin) == LOW; // LOW = натискання
    delay(50);

    if (digitalRead(orderEndPin) != orderButtonState) return;
```

```
    if (orderButtonState && !lastOrderButtonState[index]) {
      logOrderEnd(machineName);
    }
    lastOrderButtonState[index] = orderButtonState;
}

void logMachineEvent(const char* machineName, bool state) {
    Serial.print(machineName);
    Serial.print(state ? " VKL " : " VYKL ");
    Serial.println(currentTime);
    Serial.flush();
}

void logProductCount(const char* machineName, int count) {
    Serial.print(machineName);
    Serial.print(" Produkciya: ");
    Serial.println(count);
    Serial.flush();
}

void logOrderEnd(const char* machineName) {
    Serial.print(machineName);
    Serial.println(" - -");
    Serial.flush();
}
```

2.3 Створення програми «ArduinoLogger»

Для функціонування описуваємої системи знадобляться дві програми, які було вирішено писати мовою програмування C#. Перша програма буде мати

назву «ArduinoLogger». Ця програма відповідтиме за формування логів, в яких буде написаний робочий цикл усіх машин, на котрих стоятимуть датчики які контролюватиме описана вище Arduino NANO. «ArduinoLogger» має зчитувати монітор порту, до якого підключена Arduino. Сам «ArduinoLogger» працюватиме у вигляді консолі, де буде відображатись інформація, яку він має зчитувати з Arduino. Окрім консолі, «ArduinoLogger» матиме свій значок на панелі задач Windows. Натискаючи праву клавішу миши по цьому значку, відкриватиметься віконце, де можна буде вибрати COM-порт, до якого приєднана плата Arduino. Тоді у відкритої консолі мають з’явитися повідомлення, що має писати Arduino в моніторі свого порту, але вже з поточним часом, таким, який встановлений в ОС Windows на цьому комп’ютері.

Оскільки основною задачею програми є формування логів про роботу машин, до яких приєднана Arduino, то програма «ArduinoLogger» має закладену функцію кожен день о 21:00 формувати лог-файл, який матиме назву «Report_дата_час», наприклад лог-файл сформований двадцять дев’ятого листопада, о 21:00, матиме назву «Report_29.11.2025_21.00».

Проте, якщо потрібно терміново сформувати лог-файл, то можна скористатися поєднанням клавіш «Ctrl+L», яке сформує терміновий лог-файл.

Програму «ArduinoLogger» було вирішено писати за допомогою «Microsoft Visual Studio 2026», мовою C#.

Код C# для створення програми «ArduinoLogger» наведений нижче.

```
using System;
using System.Drawing;
using System.IO;
using System.IO.Ports;
using System.Runtime.InteropServices;
using System.Runtime.Remoting.Messaging;
using System.Timers;
using System.Windows.Forms;
```

```
class Program : Form
{
    private static SerialPort arduino;
    private static System.Timers.Timer logTimer;
    private static System.Timers.Timer timeSyncTimer;
    private static string desktopPath = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
    private NotifyIcon trayIcon;
    private ToolStripComboBox portSelector;
    private static string logBuffer = "";

    [DllImport("user32.dll")]
    private static extern bool RegisterHotKey(IntPtr hWnd, int id, int fsModifiers, int vk);
    [DllImport("user32.dll")]
    private static extern bool UnregisterHotKey(IntPtr hWnd, int id);

    private const int HOTKEY_ID = 1;
    private const int MOD_CTRL = 0x0002;
    private const int VK_L = 0x4C;

    [STAThread]
    static void Main()
    {
        Console.Title = "Arduino Logger";
        Console.WriteLine("Очікування даних від Arduino...");
        System.Windows.Forms.Application.Run(new Program());
    }

    public Program()
```

```
        {
            SetupTrayIcon();
            RegisterHotKey(this.Handle, HOTKEY_ID, MOD_CTRL, VK_L);
            InitializeArduino();

            logTimer = new System.Timers.Timer(60000); // Перевірка кожну хвилину
            logTimer.Elapsed += CheckLogTime;
            logTimer.Start();

            timeSyncTimer = new System.Timers.Timer(5000); // Синхронизація часу кожні 5 секунд
            timeSyncTimer.Elapsed += SyncTimeWithArduino;
            timeSyncTimer.Start();
        }

        private void InitializeArduino()
        {
            string[] ports = SerialPort.GetPortNames();
            if (ports.Length > 0)
            {
                arduino = new SerialPort(ports[0], 9600);
                arduino.DataReceived += Arduino_DataReceived;
                try { arduino.Open(); }
                catch { }
            }
        }

        private void Arduino_DataReceived(object sender, SerialDataReceivedEventArgs e)
```

```
{
    try
    {
        string data = arduino.ReadLine().Trim();
        logBuffer += data + "\n";
        Console.WriteLine(data);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Помилка при зчитуванні даних: " + ex.Message);
    }
}

private void SyncTimeWithArduino(object sender, ElapsedEventArgs e)
{
    if (arduino != null && arduino.IsOpen)
    {
        string currentTime = DateTime.Now.ToString("HH:mm:ss");
        arduino.WriteLine(currentTime);
    }
}

private void SetupTrayIcon()
{
    trayIcon = new NotifyIcon()
    {
        Icon = SystemIcons.Application,
        Visible = true,
        ContextMenuStrip = new ContextMenuStrip()
    };
```

```
            portSelector = new ToolStripComboBox();
            portSelector.Items.AddRange(SerialPort.GetPortNames());
            portSelector.SelectedIndexChanged                                    +=
PortSelector_SelectedIndexChanged;
            if (portSelector.Items.Count > 0)
               portSelector.SelectedIndex = 0;

            trayIcon.ContextMenuStrip.Items.Add("Порт:", null, null);
            trayIcon.ContextMenuStrip.Items.Add(portSelector);
            trayIcon.ContextMenuStrip.Items.Add("Вихід", null, (s, e) => ExitApp());
         }

         private void PortSelector_SelectedIndexChanged(object sender, EventArgs
e)
         {
            if (arduino != null && arduino.IsOpen)
            {
               arduino.Close();
            }
            arduino = new SerialPort(portSelector.SelectedItem.ToString(), 9600);
            arduino.DataReceived += Arduino_DataReceived;
            try
            {
               arduino.Open();
               MessageBox.Show($"Порт змінений на {arduino.PortName}",
"Успішно", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
               MessageBox.Show($"Помилка підключення до {arduino.PortName}:
```

```
{ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void ExitApp()
        {
            logTimer.Stop();
            timeSyncTimer.Stop();
            UnregisterHotKey(this.Handle, HOTKEY_ID);
            trayIcon.Visible = false;
            System.Windows.Forms.Application.Exit();
        }

        private static void CheckLogTime(object sender, ElapsedEventArgs e)
        {
            if (DateTime.Now.Hour == 21 && DateTime.Now.Minute == 0)
                SaveLog();
        }

        private static void SaveLog()
        {
            string fileName = $"Report_{DateTime.Now:dd.MM.yyyy_HH.mm}.txt";
            string filePath = Path.Combine(desktopPath, fileName);
            try
            {
                File.WriteAllText(filePath, logBuffer);
                MessageBox.Show($"Лог збережений: {fileName}", "Логування",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
```

```
            {
                MessageBox.Show($"Помилка запису логу: {ex.Message}",
"Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        protected override void WndProc(ref Message m)
        {
            if (m.Msg == 0x0312 && m.WParam.ToInt32() == HOTKEY_ID)
                SaveLog();
            base.WndProc(ref m);
        }
    }
```

2.4 Створення програми «ExcelReportParser»

Друга програма називається «ExcelReportParser». Ця програма відповідає за коректне заповнення таблиці Excel. Вона автоматично вносить зміни в таблицю Excel.

При відкритті програми «ExcelReportParser», відкрівається вікно, де потрібно вибрати лог-файл, попередньо сформований програмою «ArduinoLogger». Після вибору лог-файлу, відкриється наступне вікно, в якому потрібно буде вибрати Excel-файл. Після чого дані з лог-файлу запишуться в таблиці Excel, а лог-файл знищиться.

В самій таблиці має бути декілька листів, в нашому випадку, це чотири листа, для кожної машини окремо. Вони мають називатися «Machine_1», «Machine_2», «Machine_3» та «Machine_4». Кожна з таблиць має містити колонки з наступними назвами:

- Дата та час звіту;
- ВКЛ;

- ВИКЛ;
- Час роботи;
- Кількість виробленої продукції.

При записі лог-файлів у таблицю Excel через «ExcelReportParser», рядки з лог-файлів переносяться у таблицю Excel по принципу, якщо є щось написане в одному чи декількох рядках, то програма автоматично пропускає цей рядок та записує інформацію у наступному. Таким чином, не треба турбуватись за те, що «ExcelReportParser» перепише старі рядки, бо він формуватиме лише нові.

Код C# для створення програми «ExcelReportParser» наведений нижче.

```
using OfficeOpenXml;
using System;
using System.ComponentModel;
using System.IO;
using System.Linq;
using System.Windows.Forms;

class Program
{
   [STAThread]
   static void Main()
   {
      OpenFileDialog openFileDialog = new OpenFileDialog();
      openFileDialog.Title = "Виберіть лог-файл";
      openFileDialog.Filter = "Текстові файли|*.txt";

      if (openFileDialog.ShowDialog() != DialogResult.OK)
         return;

      string logFilePath = openFileDialog.FileName;
```

```
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.Title = "Виберіть Excel-файл";
            saveFileDialog.Filter = "Excel файли|*.xlsx";

            if (saveFileDialog.ShowDialog() != DialogResult.OK)
                return;

            string excelFilePath = saveFileDialog.FileName;

            ProcessLogFile(logFilePath, excelFilePath);

            File.Delete(logFilePath); // Видаляємо лог після обробки
        }

        static void ProcessLogFile(string logFilePath, string excelFilePath)
        {
            FileInfo fileInfo = new FileInfo(excelFilePath);
            ExcelPackage.LicenseContext =
OfficeOpenXml.LicenseContext.NonCommercial;
            using (ExcelPackage package = new ExcelPackage(fileInfo))
            {
                string[] lines = File.ReadAllLines(logFilePath);
                string reportDateTime =
Path.GetFileNameWithoutExtension(logFilePath).Replace("Report_", "");

                foreach (string machineName in new string[] { "Mashina_1",
"Mashina_2", "Mashina_3", "Mashina_4" })
                {
                    ExcelWorksheet worksheet =
package.Workbook.Worksheets.FirstOrDefault(ws => ws.Name == machineName);
```

```
if (worksheet == null)
    worksheet = package.Workbook.Worksheets.Add(machineName);

if (worksheet.Dimension == null)
{
    worksheet.Cells[1, 1].Value = "Дата та час звіту";
    worksheet.Cells[1, 2].Value = "ВКЛ.";
    worksheet.Cells[1, 3].Value = "ВИКЛ.";
    worksheet.Cells[1, 4].Value = "Час роботи";
    worksheet.Cells[1, 5].Value = "Кількість виробленої продукції";
}

int row = worksheet.Dimension?.Rows + 1 ?? 2;
string startTime = "";
string endTime = "";
int productionCount = 0;

foreach (var line in lines)
{
    string[] parts = line.Split(' ');
    if (parts.Length >= 3 && parts[0] == machineName)
    {
        if (parts[1] == "VKL")
        {
            startTime = parts[2];
            productionCount = 0;
        }
        else if (parts[1] == "VYKL")
        {
            endTime = parts[2];
```

```
                        TimeSpan workDuration = TimeSpan.Zero;

                        if (TimeSpan.TryParse(startTime, out TimeSpan start) &&
TimeSpan.TryParse(endTime, out TimeSpan end))
                        {
                            workDuration = end - start;
                        }

                        worksheet.Cells[row, 1].Value = reportDateTime;
                        worksheet.Cells[row, 2].Value = startTime;
                        worksheet.Cells[row, 3].Value = endTime;
                        worksheet.Cells[row, 4].Value = workDuration.ToString();
                        worksheet.Cells[row, 5].Value = productionCount;
                        row++;
                    }
                    else if (parts[1] == "Produkciya:" && int.TryParse(parts[2], out
int count))
                    {
                        productionCount = count;
                    }
                }
            }
        }

        package.Save();
    }
}
```

2.5 Демонстрація результатів роботи розробленої системи

Для початку приєднуємо завчасно запрограмовану Arduino NANO до комп’ютера, Arduino NANO було приєднано до порту COM4. Після цього потрібно запустити «ArduinoLogger». Після чого, відкриється консоль з написом «Очікування даних від Arduino…».

Сам «ArduinoLogger» нагадує своїм за зовнішнім виглядом командний рядок Windows. В цьому командному рядку має прописуватись інформація, яку програма має зчитувати безпосередньо з монітору порту, до якого приєднана Arduino. Саме цю інформацію і формує програма, що зашита в мікроконтролер Arduino. В цих рядках мають бути прописані моменти запуску та зупинки виробничого процесу машин та кількість виробленої їми продукції за час їх сеансу.

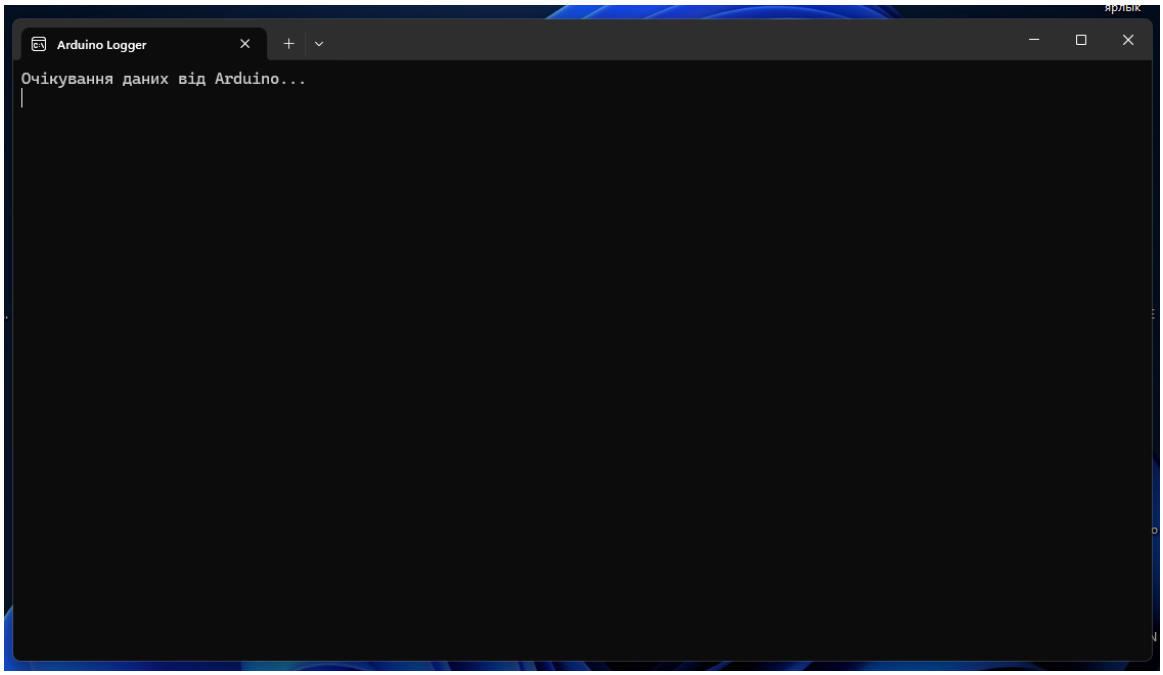

Рис. 2.4 Вікно «ArduinoLogger»

Після цього, потрібно вибрати порт, до якого була приєднана Arduino NANO, щоби «ArduinoLogger» міг зчитувати дані, що присилає йому в порт Arduino NANO.

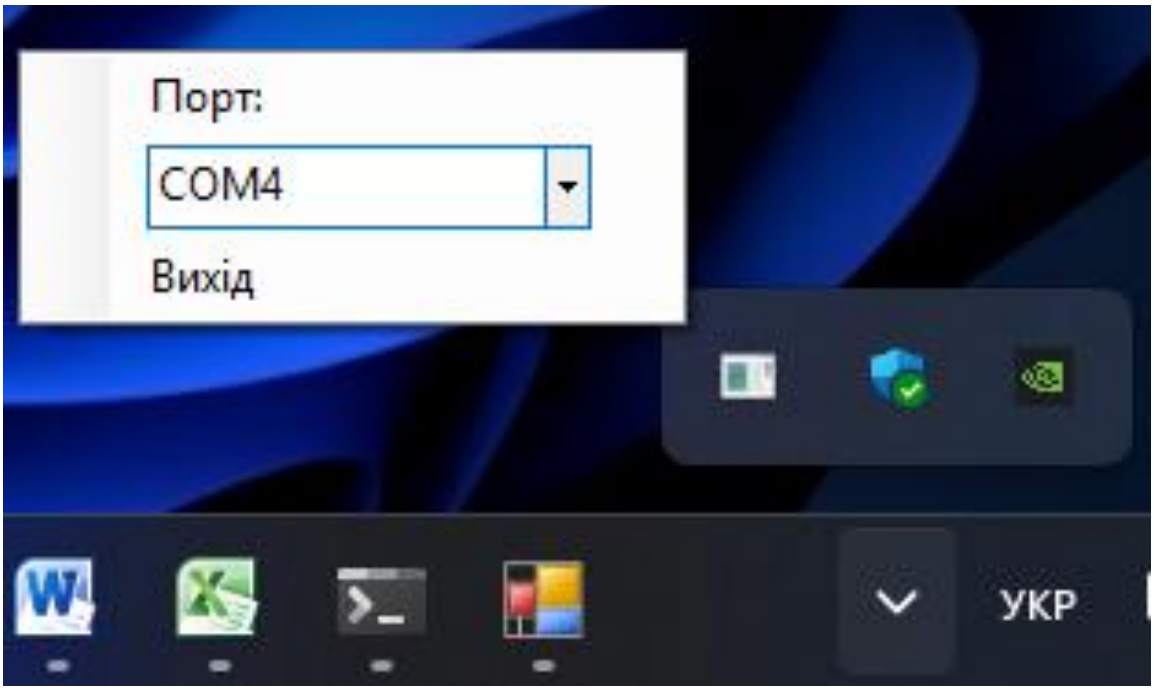

Рис. 2.5 Вікно «ArduinoLogger» на панелі задач з вибором COM-порту

Після вибору потрібного порту, комп’ютер починає сприймати дані напряму з Arduino та виводити їх у консоль. Ми можемо бачити напис «Gotovo do roboty» латиницею, бо кирилицю Arduino виводить проблематично.

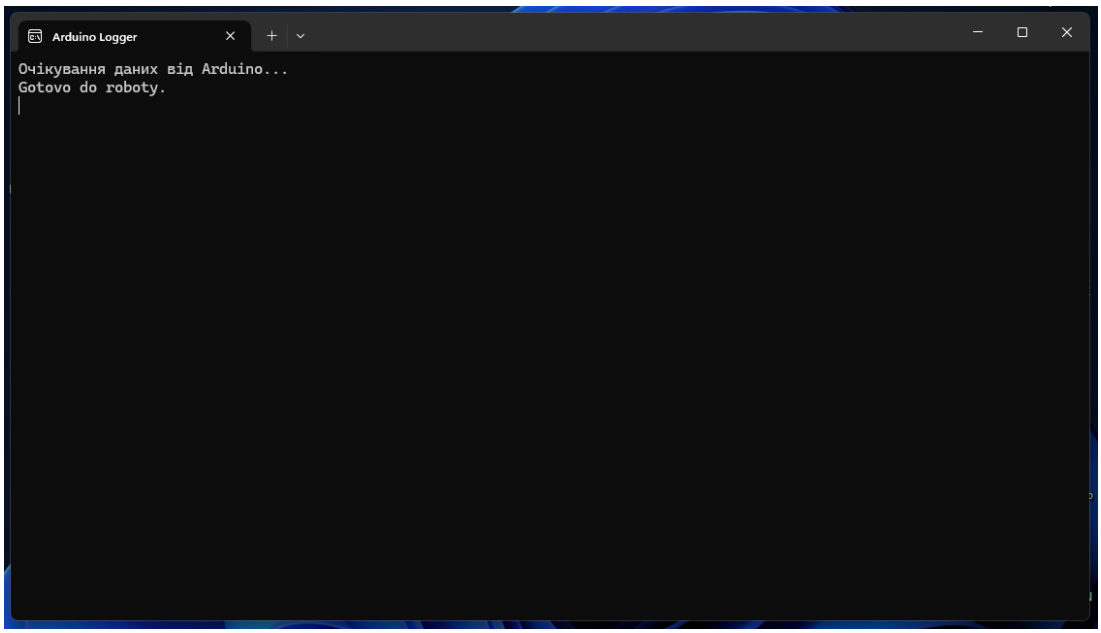

Рис. 2.5 Вікно «ArduinoLogger» на панелі задач з вибором COM-порту

Щоби показати працездатність системи, нам потрібно запустити одну з машин, запуск яких відстежує твердотільне реле. Для прикладу, активуємо

твердотільне реле машини під номером чотири, щоби з’явилося повідомлення про те, що машина включена.

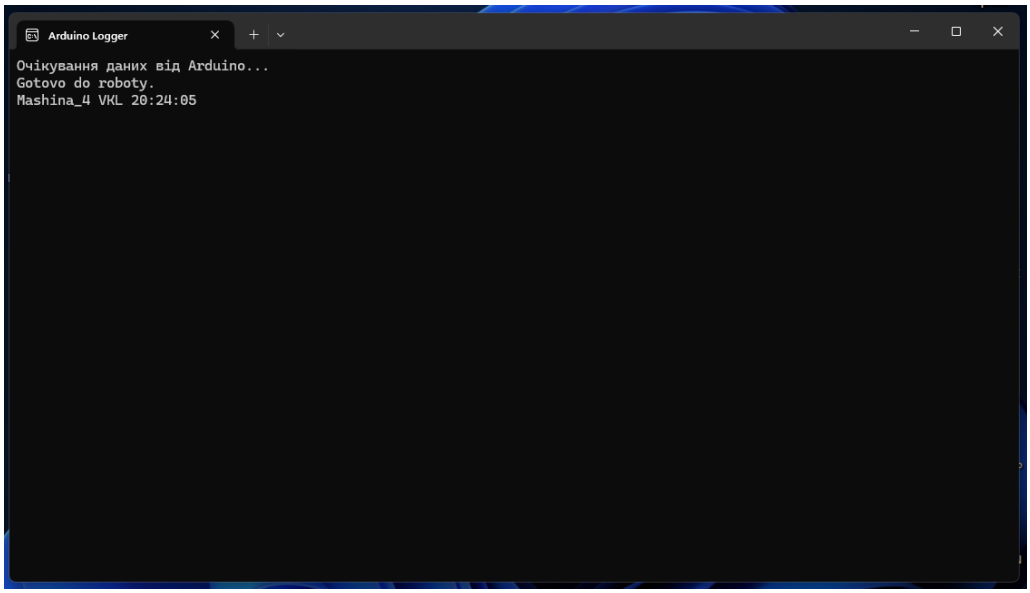

Рис. 2.6 Вікно «ArduinoLogger» з запущеною машиною під номером чотири

Можемо бачити в консолі повідомлення, що машина під номером чотири, стартувала свій процес роботи о 20:24 на п’ятій секунді. Далі активовуємо і деактивовуємо інфрачервоний датчик п’ятнадцять разів, щоби датчик нарахував п’ятнадцять зроблених пакетів, після чого деактивовуємо твердотільне реле, щоби Arduino відчула, що машина зробила зупинку.

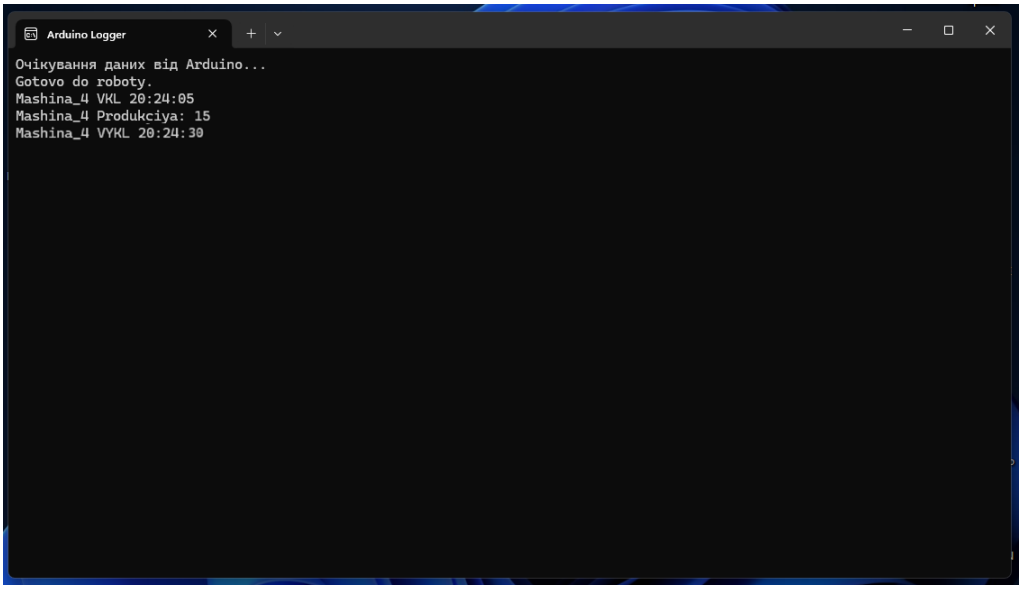

Рис. 2.7 Вікно «ArduinoLogger» з завершеним циклом роботи машини під номером чотири

В результаті можемо бачити в консолі написи, що свідчать про те, що було зроблено п’ятнадцять пакетів та зупинка відбулася о 20:24, на тридцятій секунді.

Тепер для перевірки примусового формування лог-файлу, треба зажати поєднання клавіш «Ctrl+L», після чого можна буде побачити, що лог-файл успішно створений. Назва лог-файлу має дату та час, коли було створено лог-файл, а саме 29 листопада 2025 року, о 20:41. Лог-файл автоматично створиться на робочому столі ОС Windows.

Оскільки в програмі «ArduinoLogger» прописано, що кожен день о 21:00 має формуватися лог-файл, це дозволяє автоматизувати процес формування лог-файлів, які формуватимуться кожен день перед закінченням робочого дня у особи, що відповідальна за контроль внесення лог-файлів в таблицю Excel. Звичайно, 21:00 це лише тестовий час, тому його можна змінити на будь-який інший, щоби підлаштувати систему під конкретний графік роботи співробітників підприємства.

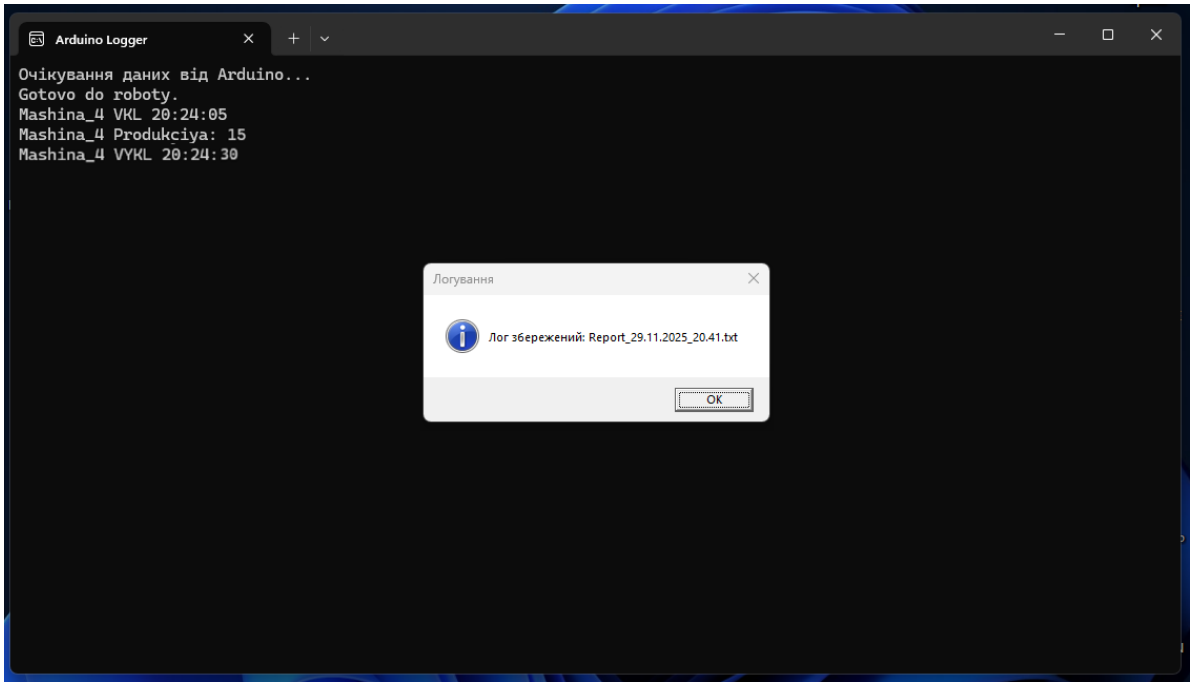

Рис. 2.8 Вікно «ArduinoLogger» з виведеним повідомленням про формування лог-файлу

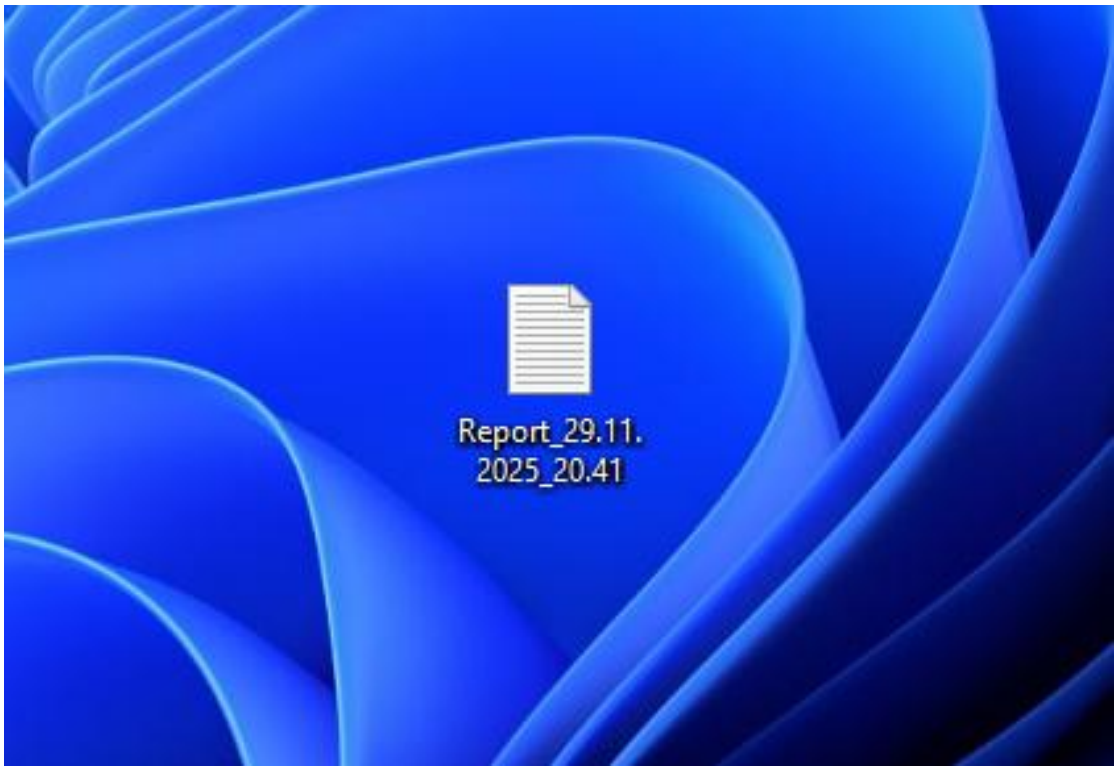

Рис. 2.9 Текстовий документ лог-файлу на робочому столі ОС Windows

Відкривши лог-файл «Report_29.11.2025_20.41», можемо бачити все, що було записано у консолі, окрім самої верхнього рядку «Очікування даних від Arduino…».

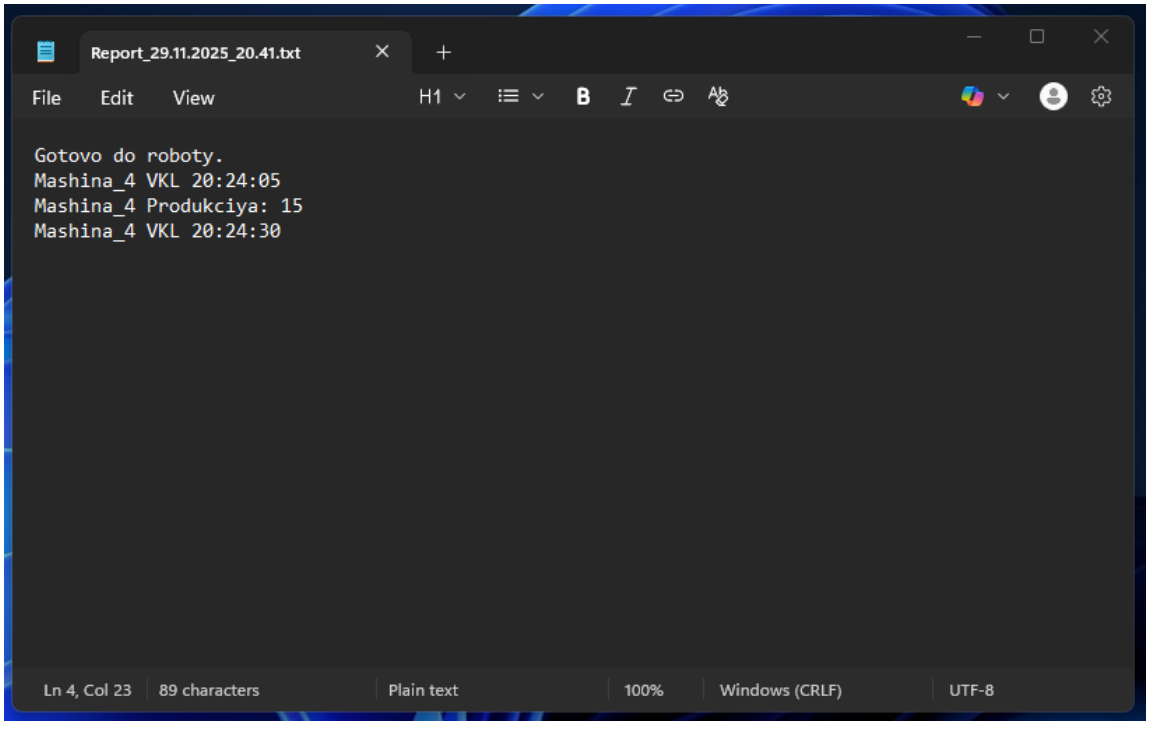

Рис. 2.10 Вміст лог-файлу «Report_29.11.2025_20.41»

Тепер, потрібно завантажити дані з лог-файлу у таблицю Excel. Для цього нам знадобиться програма «ExcelReportParser». Завчасно створимо Excel-файл, його можна назвати будь-як, дамо їй назву «Test» та напишемо там назви колонок майбутньої таблиці, після чого створимо ще три листа, кожен з котрих матиме назву «Machine_1», «Machine_2», «Machine_3» та «Machine_4», відповідно кожній з машин. При завантаженні лог-файлів у Excel-файл, лог-файл знищується, але всі дані з лог-файлу, окрім першого рядку з написом «Gotovo do roboty.» переносяться в таблицю, разом з датою та часом у назві файлу.

Кожний лист у Excel-файлі матиме аналогічні таблиці з аналогічними назвами, в залежності від номеру машини, прописаної в програмі Arduino, її дані будуть розміщуватись на відповідному листі в Excel-файлі. Таким чином, інформація про діяльність різних машин не змішується та читати цю інформацію стає комфортніше.

В колонці «Час роботи» записується розрахований час одного виробничого сеансу машини. Таким чином документуються факти запуску та зупинки машин, а також можна наглядно бачити скільки часу машина працювала, а скільки простоювала.

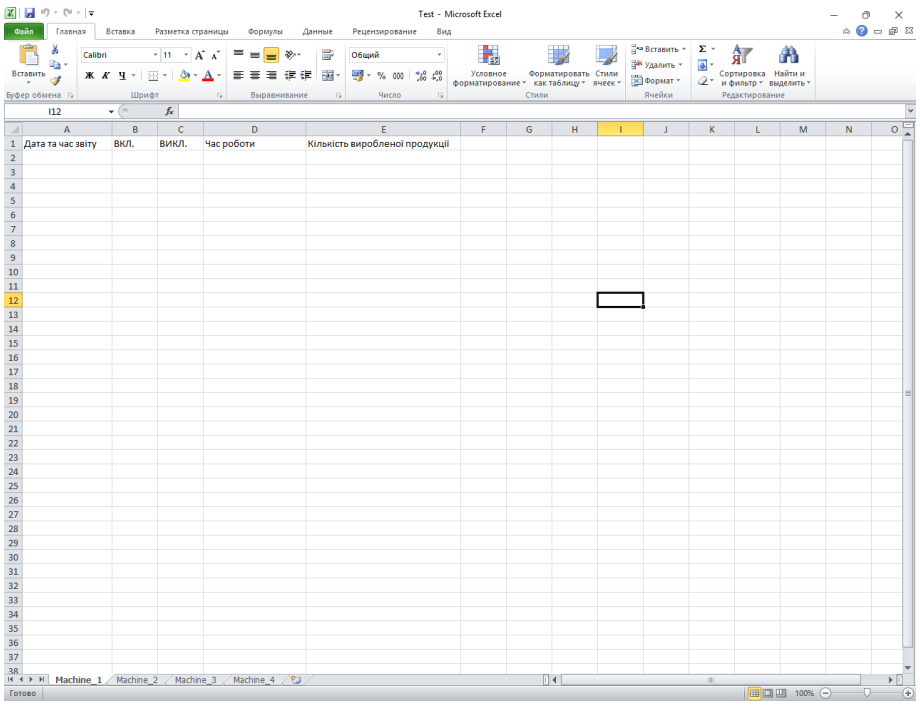

Рис. 2.11 Таблиця Excel з пустою таблицею на листі «Machine_1»

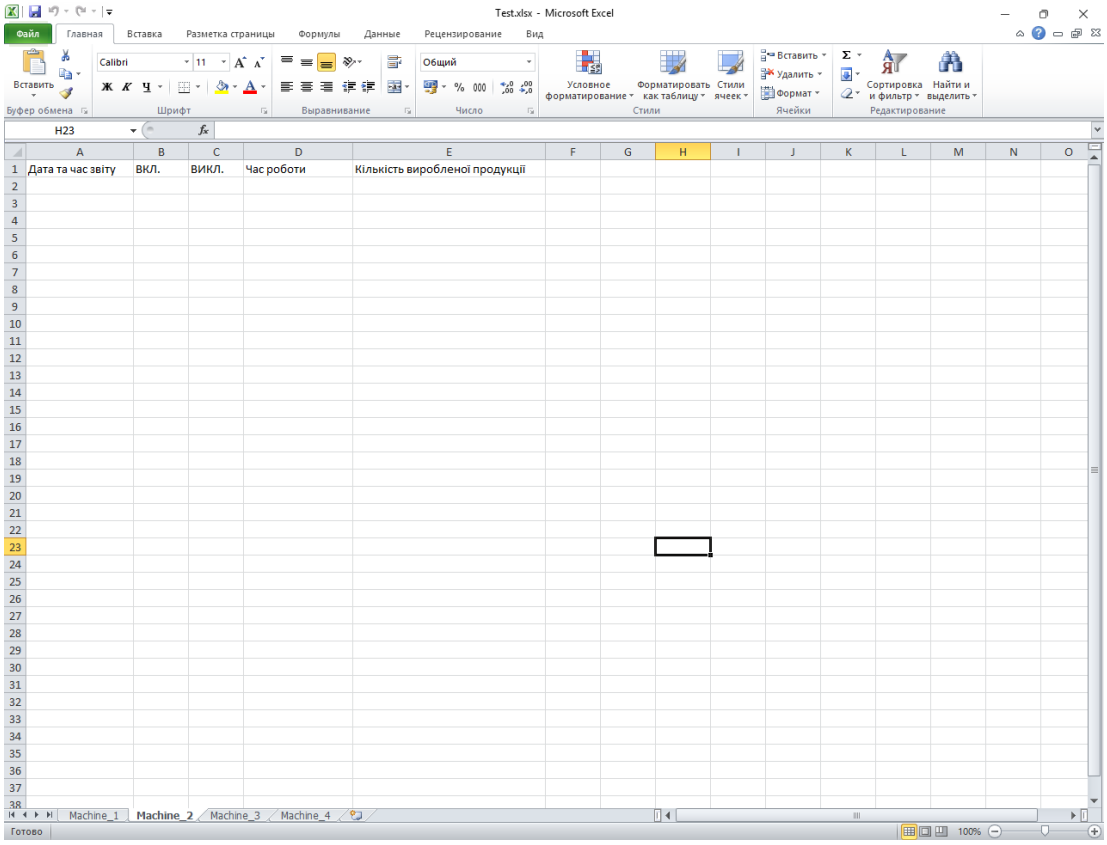

Рис. 2.12 Таблиця Excel з пустою таблицею на листі «Machine_2»

Запустимо програму «ExcelReportParser», щоби завантажити інформацію з лог-файлу до таблиці Excel. При запуску «ExcelReportParser», відкривається вікно, де треба вибрати лог-файл.

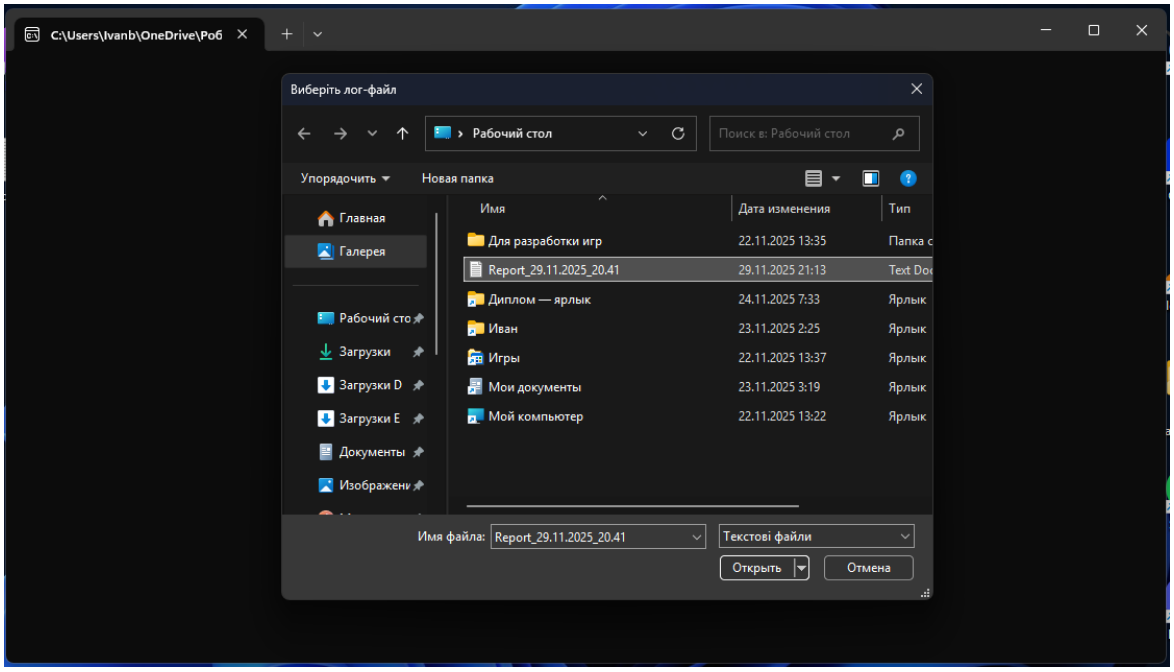

Рис. 2.13 Вікно вибору лог-файлу у програмі «ExcelReportParser»

Після вибору лог-файлу, у програмі «ExcelReportParser» з’явилося нове вікно, в якому на цей раз треба вибрати файл Excel, у який буде перезаписуватись.

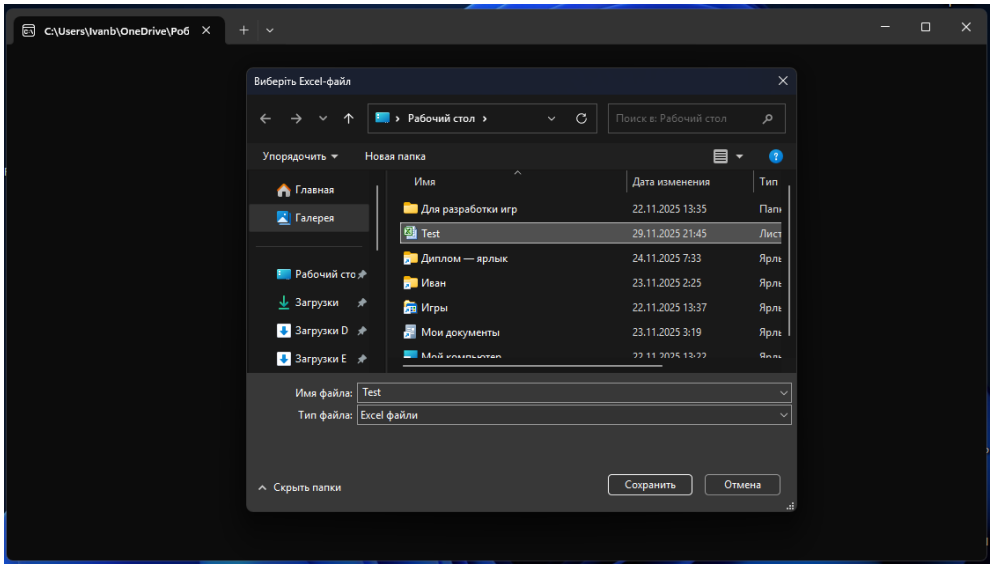

Рис. 2.14 Вікно вибору файлу Excel у програмі «ExcelReportParser»

Після вибору файлу Excel, лог-файл знищився, а у таблиці Excel з’явилася строка з датою та часом створення лог-файлу, часом запуску та зупинки робочого процесу, кількість виробленої за цикл продукції та час, скільки тривав виробничий цикл.

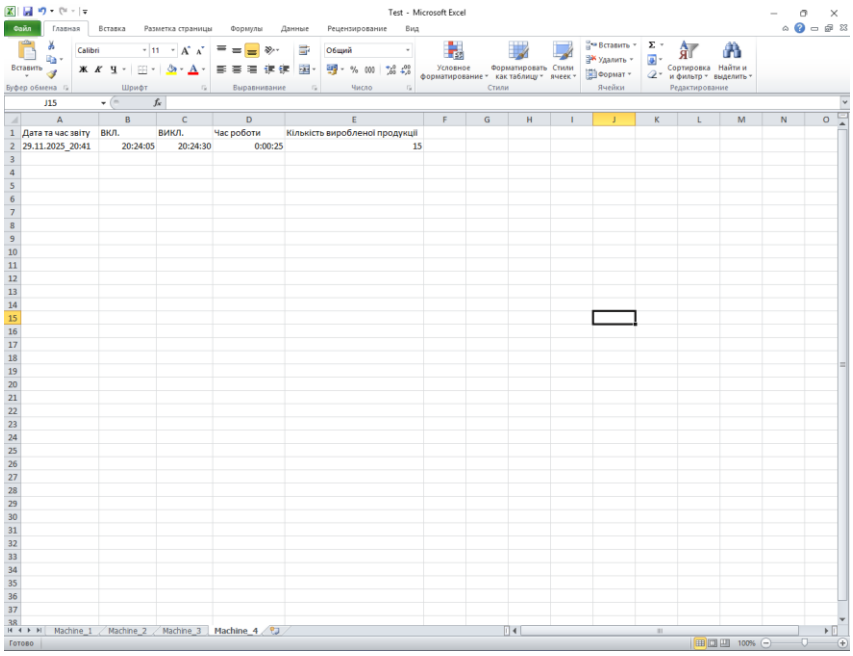

Рис. 2.15 Таблиця Excel після завантаження лог-файлу «Report_29.11.2025_20.41»

3. ПРИКЛАДИ ЗАСТОСУВАННЯ СИСТЕМИ НА БАЗІ ІНШОЇ ВИРОБНИЧОЇ МАШИНИ З ІНШИМ ПРИНЦИПОМ РОБОТИ

3.1 Принцип роботи бобінорізки

Бобінорізка – це виробнича машина, яка застосовується у напрямку поліграфії, чиєю метою є нарізка плівки на окремі рулони. На відміну від пакетоформувальної машини, бобінорізка не виробляє продукцію поштучно. Принцип її рахунку полягає в рахуванні метражу плівки на окремий рулон. Швидкість бобінорізки вимірюється в метрах у хвилину (м/х), а довжина плівки згорнутої у рулони вимірюється у метрах погонних (м.п.).

Для більшої наочності була розроблена спрощена 3D-модель бобінорізки, яка має в собі ключові вузли свого оригіналу.

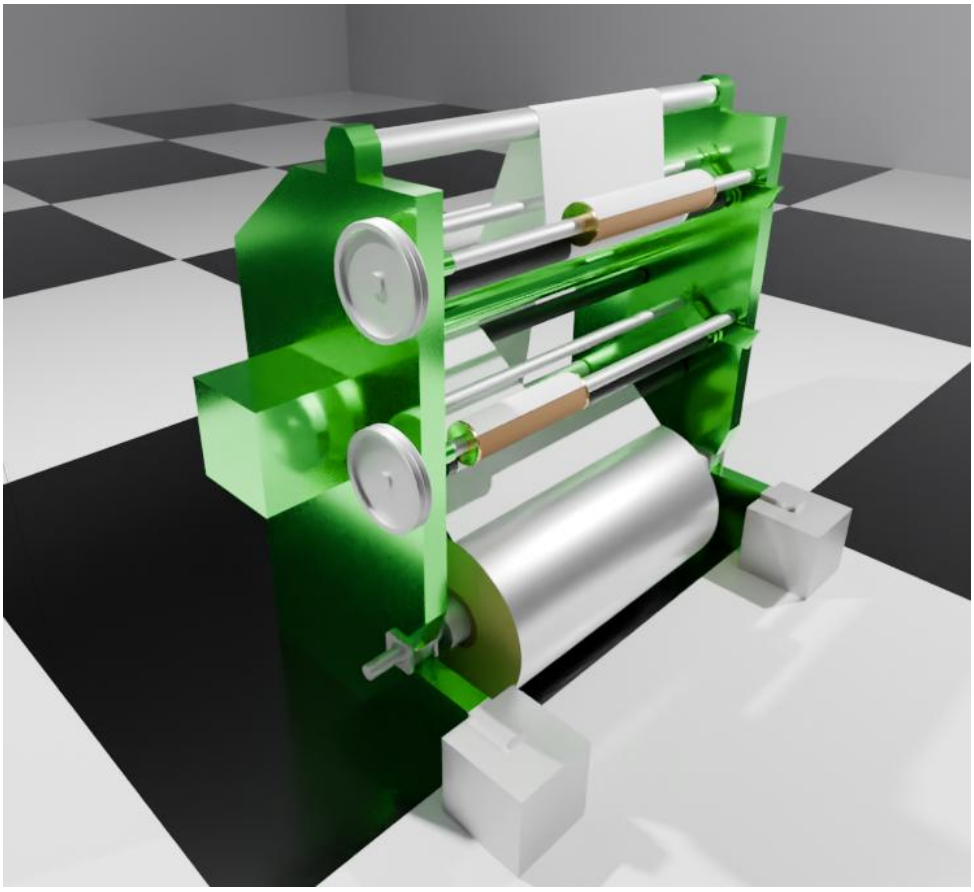

Рис. 3.1 Спрощена 3D-модель бобінорізки (ракурс з переднього боку)

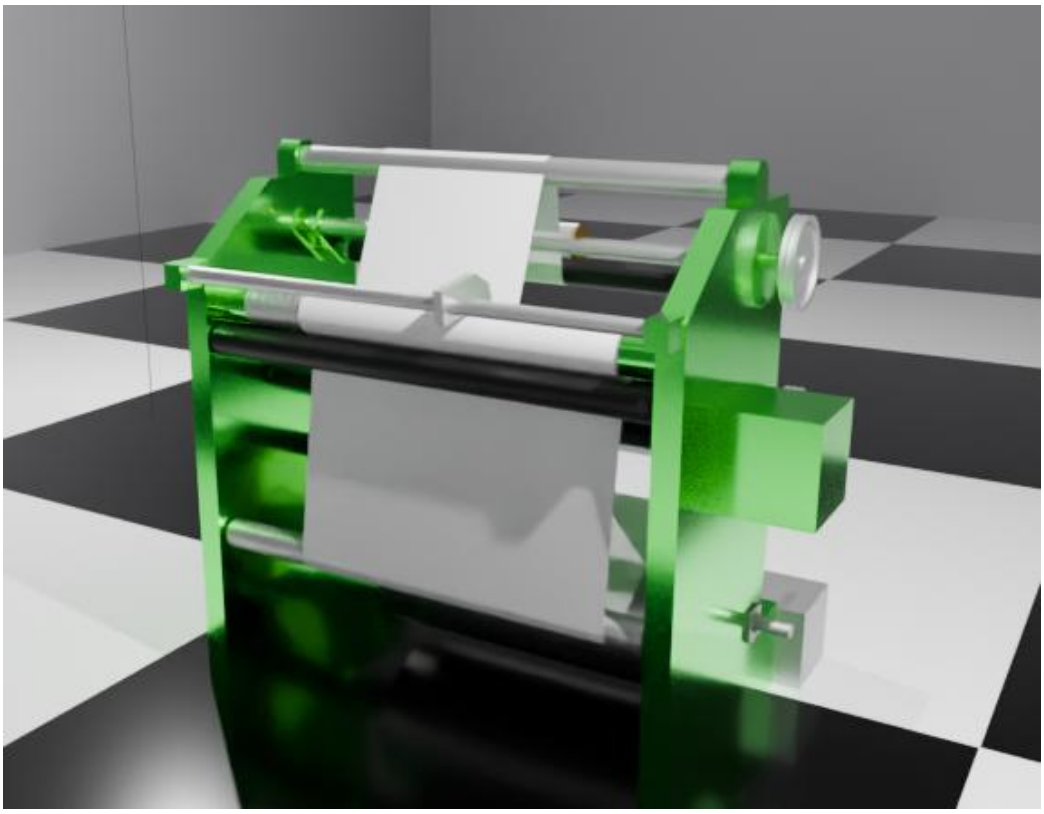

Рис. 3.2 Спрощена 3D-модель бобінорізки (ракурс з заднього боку)

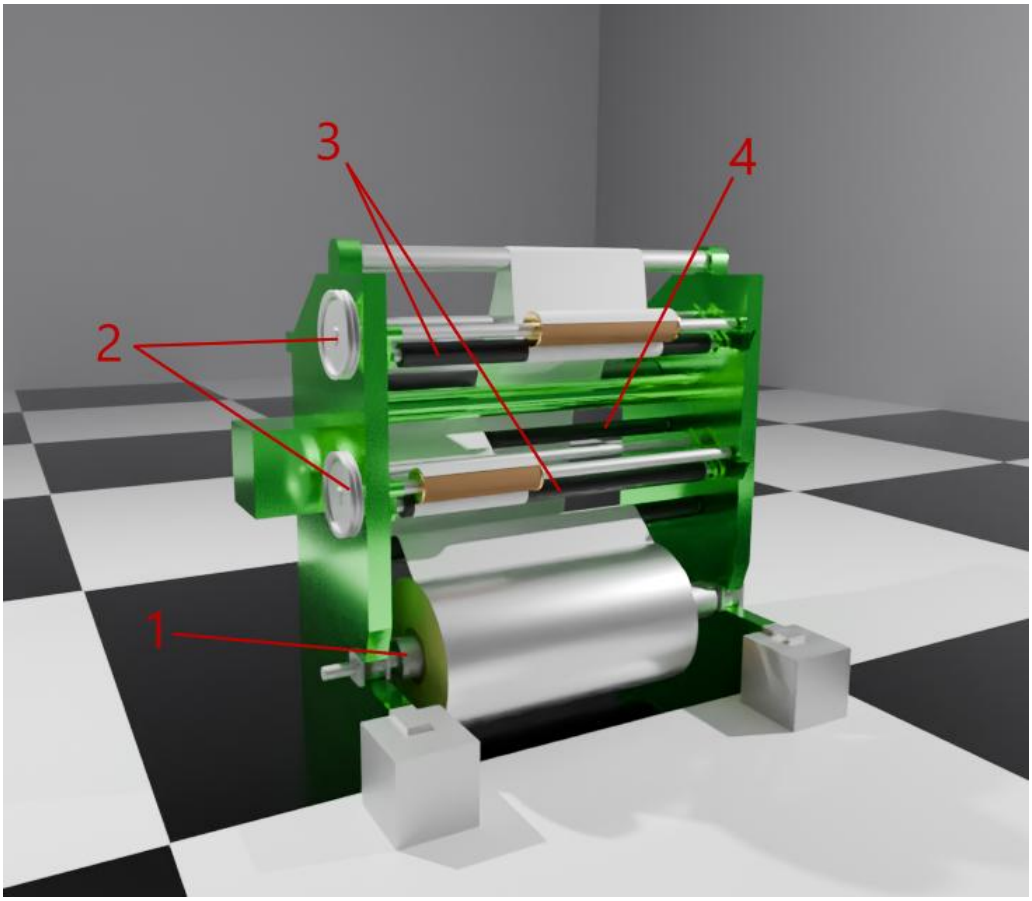

Рис. 3.3 Ключові елементи бобінорізки (з переднього боку)

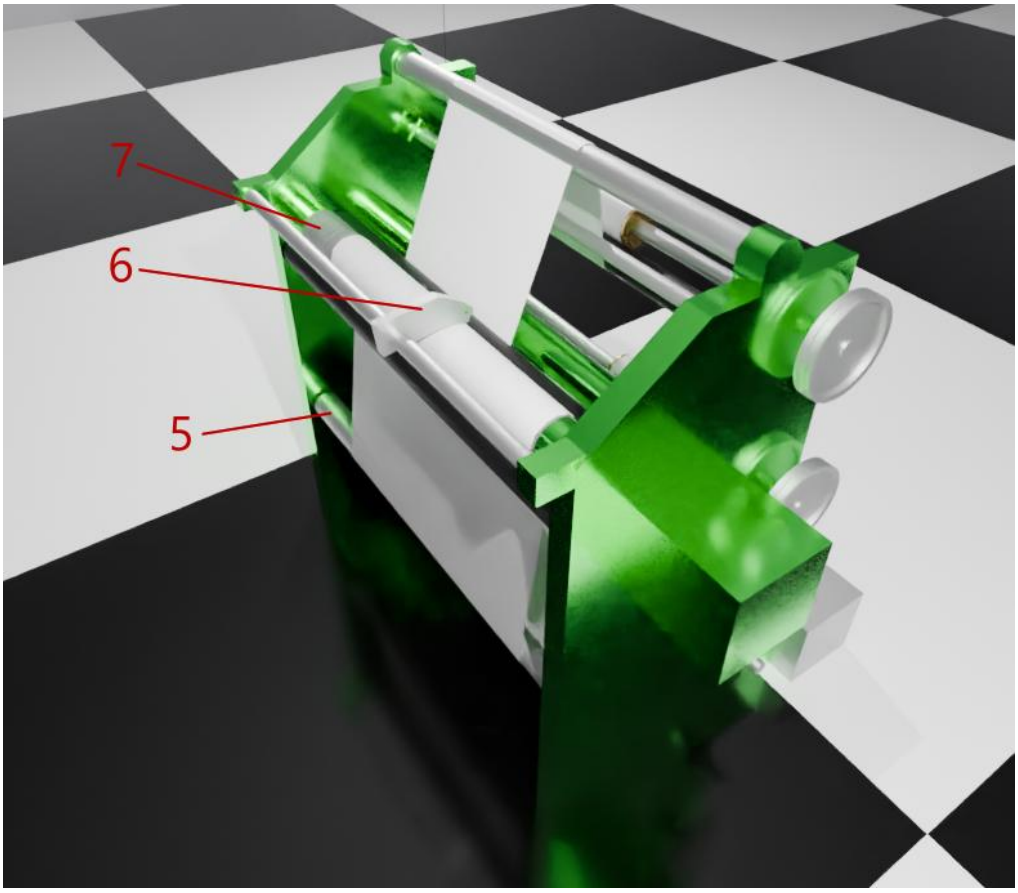

Рис. 3.4 Ключові елементи бобінорізки (з заднього боку)

- Бобіна;
- намотчики;
- прижимні вали;
- рахувальний вал;
- тензо-вал;
- лезо;
- противоніж.

Бобіна – це вал, порожній всередині, який має висувні елементи, щоб фіксувати рулон з сировинною плівкою. Висувні елементи видуваються зжатим повітрям з компресору, яке вдувається у бобіну через клапан. Висувні елементи впиваються у втулку, висячого на бобіні рулону, та не дозволяють рулону їздити на бобіні в процесі роботи бобінорізки.

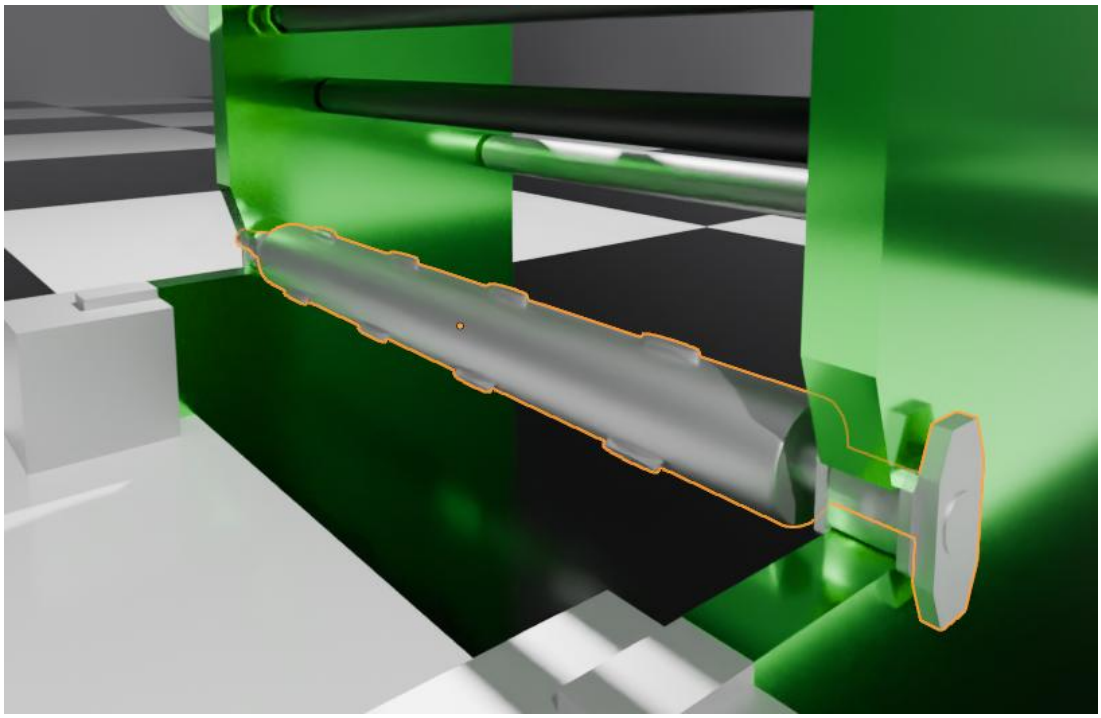

Рис. 3.5 Бобіна (без рулону)

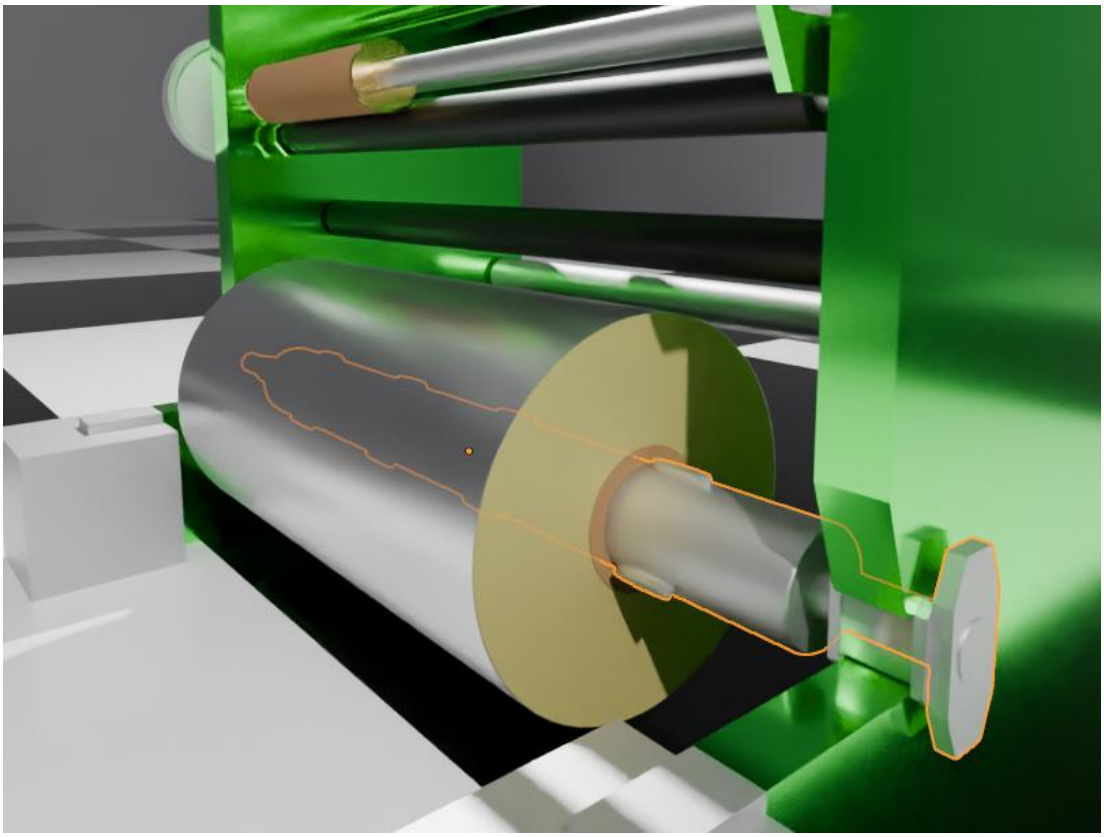

Рис. 3.6 Бобіна (з рулоном)

Намотчики – це вали, які обертає двигун за допомогою ремінів. Їх основною функцією є намотування на себе розрізаної плівки, що йде з бобіни.

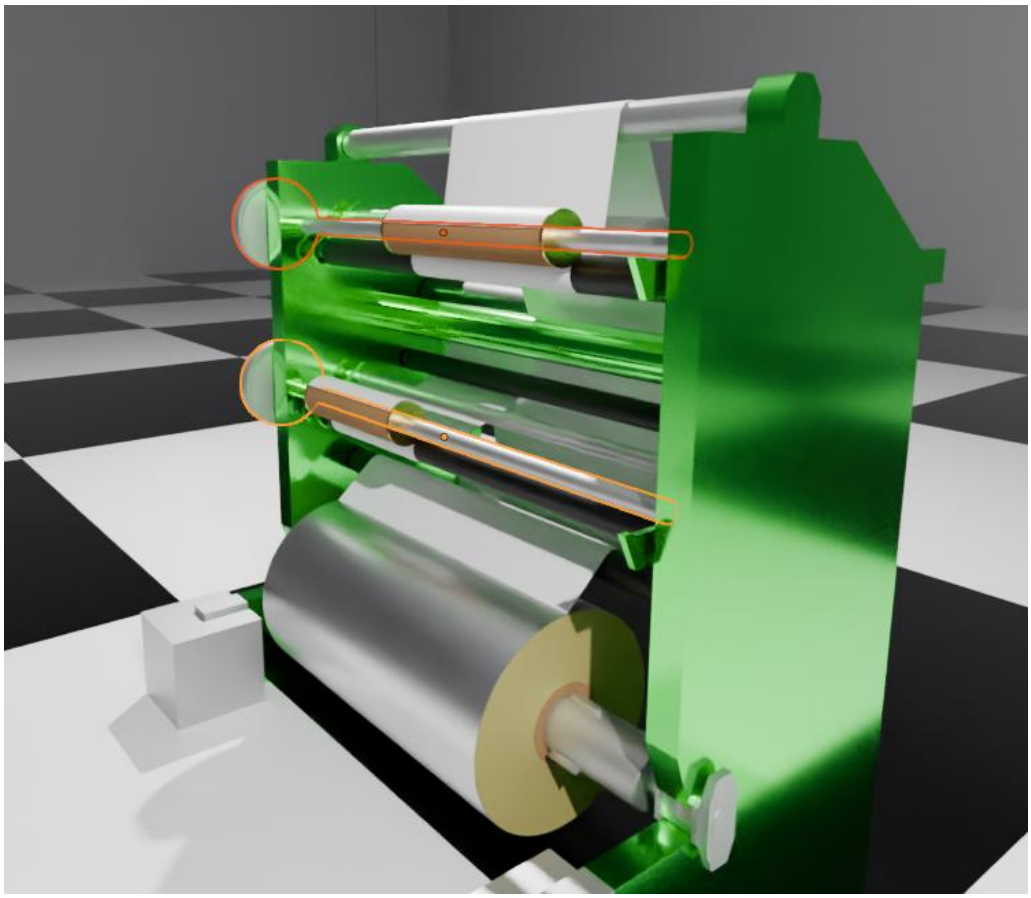

Рис. 3.7 Верхній та нижній намотчики (ракурс з правої сторони)

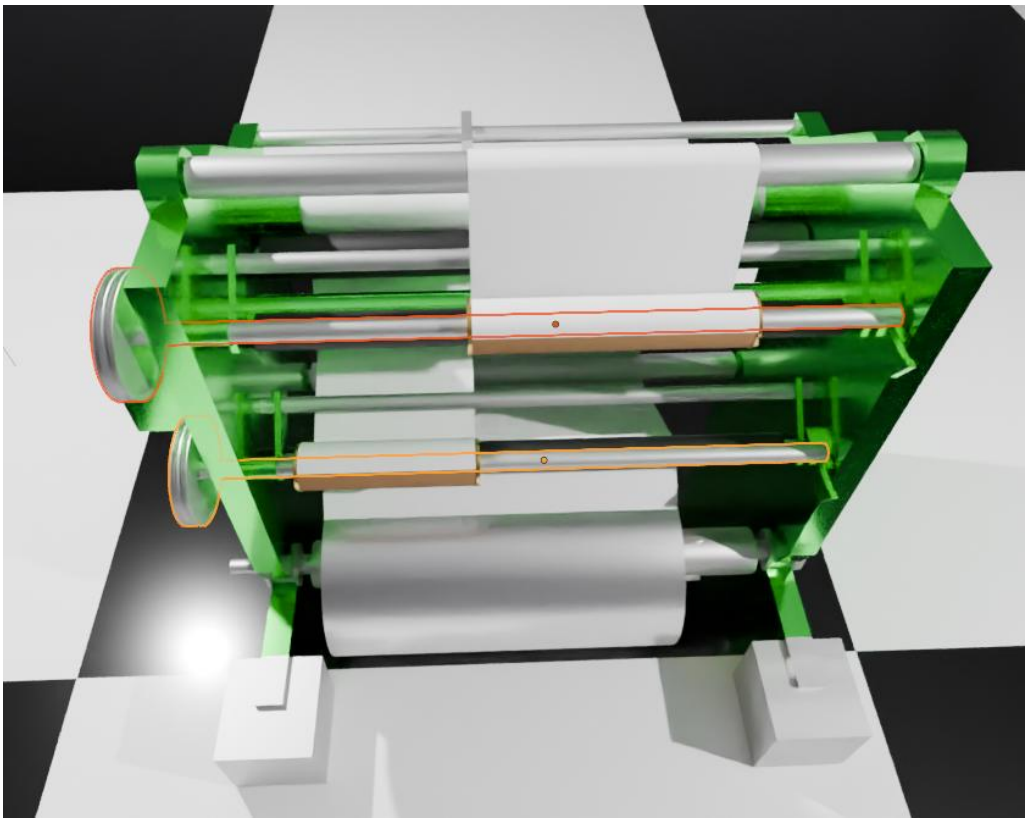

Рис. 3.8 Верхній та нижній намотчики (ракурс з верхньої сторони)

Прижимні вали – це гумові вали, які під дією пневматики, прижимають плівку до втулки, що закріплена на намотчиках для того, щоби намотування йшло під тиском та щоби плівка не висіла, намотуючись на втулку.

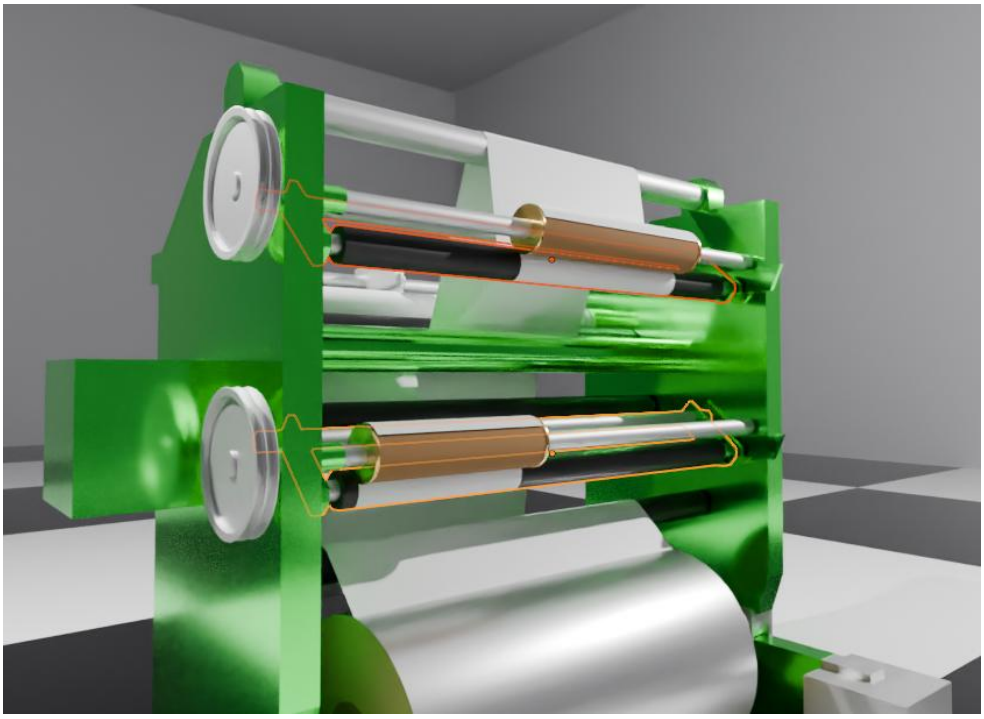

Рис. 3.9 Верхній та нижній прижимні вали (ракурс з лівої сторони)

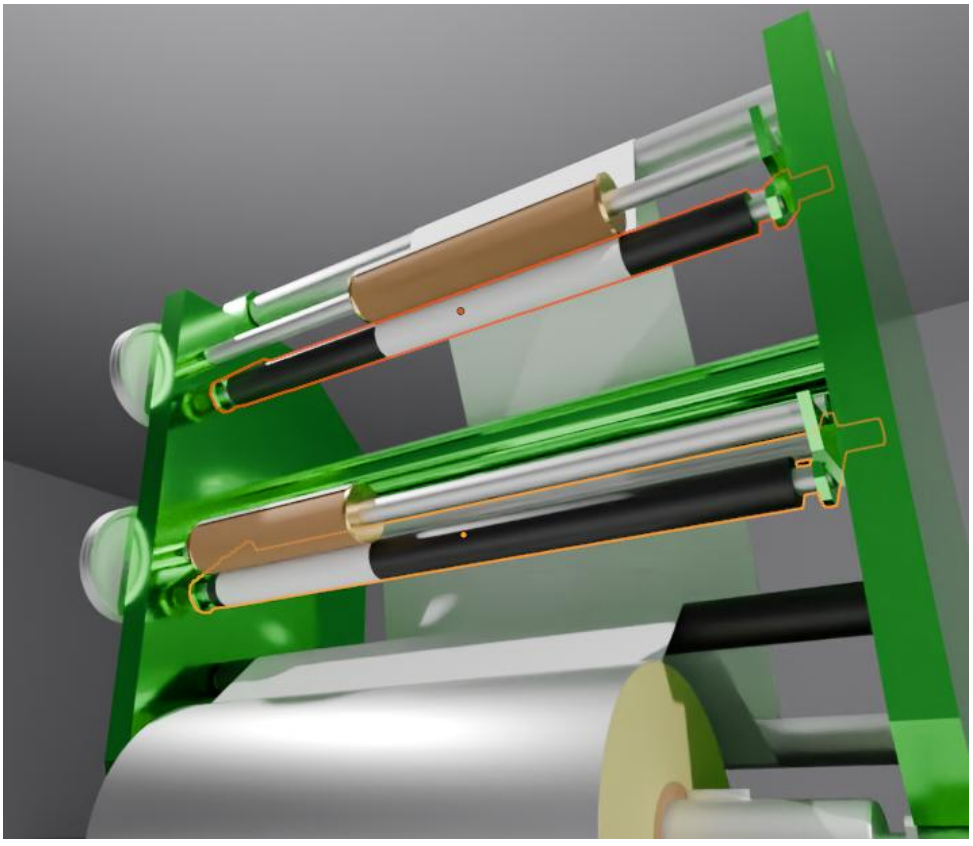

Рис. 3.10 Верхній та нижній прижимні вали (ракурс з нижньої сторони)

Рахувальний вал – це гумовий вал, головною функцією якого є рахування метрів погонних та виведення поточної кількості метрів погонних на лічильник пульта керування. Він працює за допомогою датчика Холла , який добавляє один метр погонний за шість обертів рахувального вала навколо своєї осі, бо окружність вала в шість разів менша за один метр.

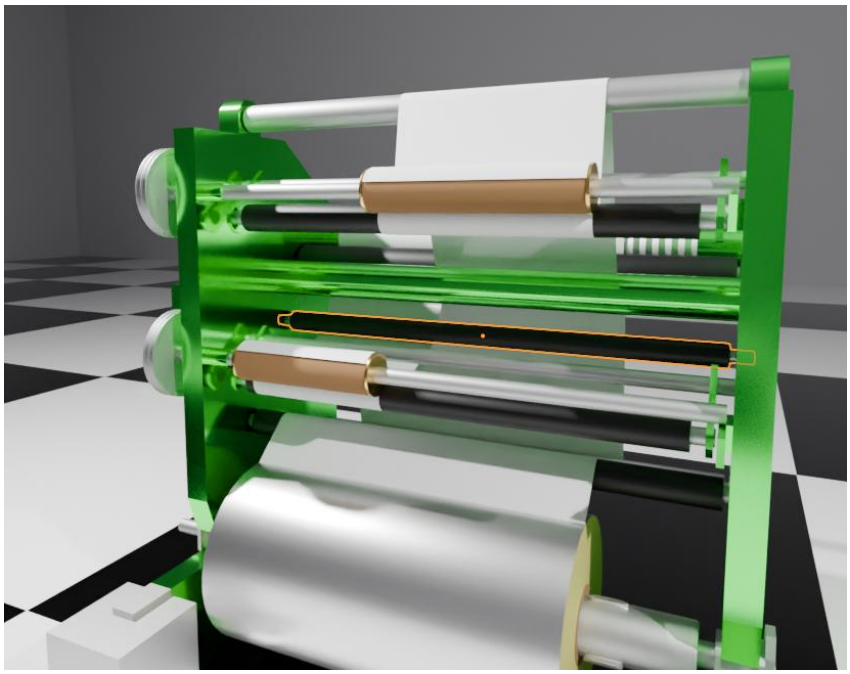

Рис. 3.11 Рахувальний вал

Тензовал – це металевий вал, в одному з країв якого встановлений тензодатчик, якій міряє натяг полотна

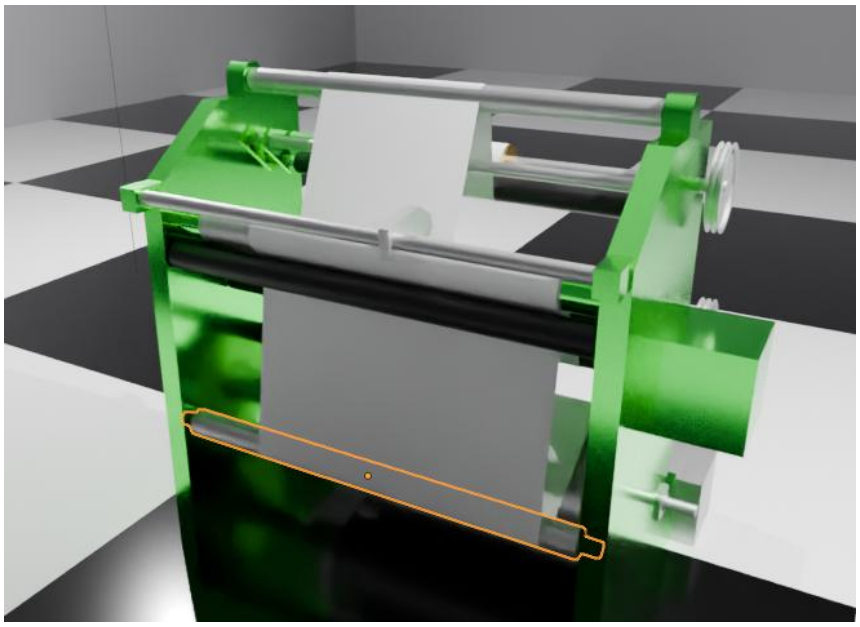

Рис. 3.12 Тензовал

Лезо – це звичайне лезо, яке кріпиться на штатив та відповідає за розрізання полотна на дві частини. Лезо може бути не одне, їх може бути декілька на всій ширині полотна. Проте, якщо лезо не одне, а їх декілька, то слід встановлювати втулки на намотчиках в шаховому порядку, щоби сусідні втулки на намотчиках не заважали одна іншій. Леза можно заміняти на нові, бо старі все одно зношуються при розрізанні полотна. Якщо леза вчасно не міняти, воно може почати рвати полотно. Також штативи з лезами можуть переміщуватись по трубі, на якій кріпляться. Це зроблено щоби можна було міняти місцеположення леза та різати плівку на інші формати.

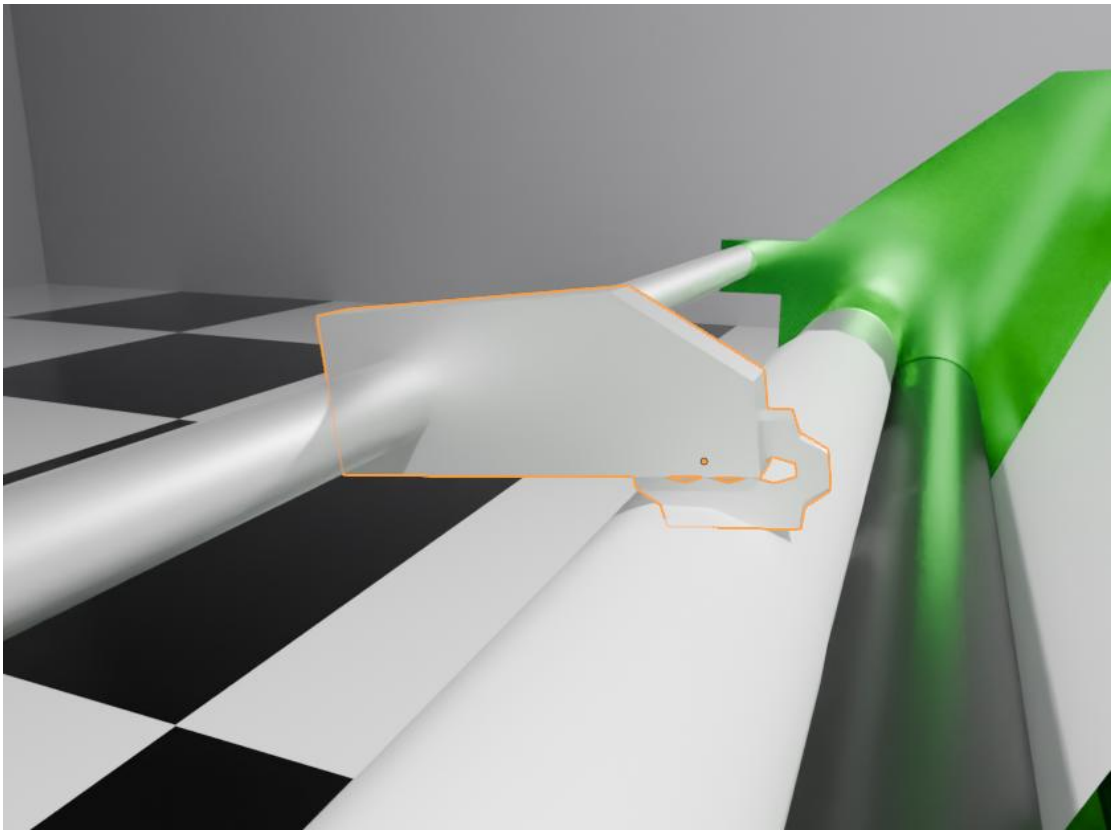

Рис. 3.13 Лезо на штативі

Протиніж – це металевий вал з багатьма борознами, які потрібні для того, щоб леза не тупилися об метал, або інший матеріал. Кожна борозна в 1 міліметрі одна від одної, а ширина кожної борозни рівняється 2 міліметрам. Лезо має бути вставлено в одну з борозен та не контактувати зі стінками борозни, бо на великій швидкості лезо може погнутися.

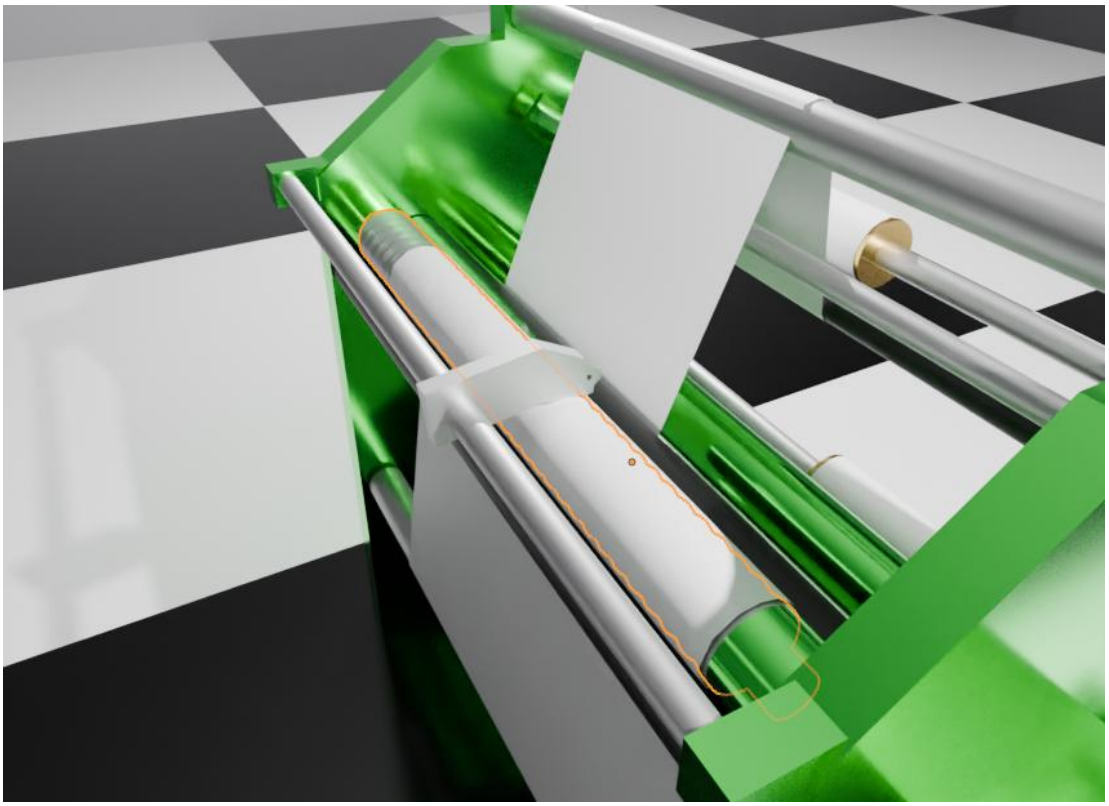

Рис. 3.14 Протиніж

3.2 Написання коду Arduino для розрахунку метрів погонних

Для розрахунку метрів погонних, можна використати рахувальний вал та принцип його роботи. Треба помістити датчик Холла, підключений до Arduino поруч з датчиком Холла, який рахує метри погонні для лічильника. Датчик Холла для лічильника на пульті керування спрацьовує на магніт, який прикріплений до однієї з сторін рахувального валу. Коли рахувальний вал обертається, магніт на одній з його сторін потрапляє в зону видимості датчика Холла та він спрацьовує.

Можна взяти попередній код Arduino та переробити його так, щоби кожні шість спрацьовувань датчика Холла рахувалися за одиницю. Також, слово «продукція» в коді треба переробити на «м.п.».

Перероблений для системи бобінорізки код Arduino для автоматизованої системи обліку наведений нижче. На цей раз спростимо систему, використовуючи бібліотеку «GyverButton.h». З цією бібліотекою буде простіше

взаємодіяти з датчиками, працюючими по принципу кнопки.

```
#include <GyverButton.h>

const int machine1Pin = 2;
const int machine2Pin = 3;
const int machine3Pin = 4;
const int machine4Pin = 5;

const int button1Pin = 6;
const int button2Pin = 7;
const int button3Pin = 8;
const int button4Pin = 9;

const int orderEndButton1 = 10;
const int orderEndButton2 = 11;
const int orderEndButton3 = 12;
const int orderEndButton4 = 13;

String currentTime = "00:00:00";

bool machineState[4] = {false, false, false, false};

int productCount[4] = {0, 0, 0, 0};

int touchCount[4] = {0, 0, 0, 0};

GButton btnProd[4] = {
  GButton(button1Pin),
  GButton(button2Pin),
  GButton(button3Pin),
```

```
  GButton(button4Pin)
};

GButton btnOrderEnd[4] = {
  GButton(orderEndButton1),
  GButton(orderEndButton2),
  GButton(orderEndButton3),
  GButton(orderEndButton4)
};

void setup() {
  Serial.begin(9600);

  pinMode(machine1Pin, INPUT);
  pinMode(machine2Pin, INPUT);
  pinMode(machine3Pin, INPUT);
  pinMode(machine4Pin, INPUT);

  Serial.println("Gotovo do roboty.");
}

void loop() {

  if (Serial.available()) {
    currentTime = Serial.readStringUntil('\n');
  }

  checkMachine(0, machine1Pin, "Mashina_1");
  checkMachine(1, machine2Pin, "Mashina_2");
  checkMachine(2, machine3Pin, "Mashina_3");
```

```
  checkMachine(3, machine4Pin, "Mashina_4");

  delay(20);
}

void checkMachine(int index, int pin, const char* name) {
  bool current = digitalRead(pin);

  if (current != machineState[index]) {
    machineState[index] = current;
    logMachineEvent(name, current);

    if (!current) {
      logProductCount(name, productCount[index]);

      productCount[index] = 0;
      touchCount[index] = 0;
    }
  }

  btnProd[index].tick();
  if (machineState[index] && btnProd[index].isClick()) {

    touchCount[index]++;

    if (touchCount[index] >= 6) {
      productCount[index]++;
      touchCount[index] = 0;
    }
  }
```

```
  btnOrderEnd[index].tick();
  if (btnOrderEnd[index].isClick()) {
    logOrderEnd(name);
  }
}

void logMachineEvent(const char* name, bool state) {
  Serial.print(name);
  Serial.print(state ? " VKL " : " VYKL ");
  Serial.println(currentTime);
}

void logProductCount(const char* name, int count) {
  Serial.print(name);
  Serial.print(" M.P.: ");
  Serial.println(count);
}

void logOrderEnd(const char* name) {
  Serial.print(name);
  Serial.println(" - -");
}
```

Також, в схемі підключення потрібно замінити інфрачервоні датчики на датчики Холлу. Суть підключення не змінилася, але на датчиках Холлу є два сигнальних піни, аналоговий та цифровий. Аналоговий пін в цій системі не потрібен, тому не підключаємо його, а просто переключаємо сигнальний пін інфрачервоного датчика на цифровий сигнальний пін датчика Холлу, він позначений як «D0».

Оскільки датчик Холлу реагує на магніт, то він не буде реагувати на все

поспіль, як це трапляється з інфрачервоним датчиком, а тому його можно ставити в більш тісні місця машини.

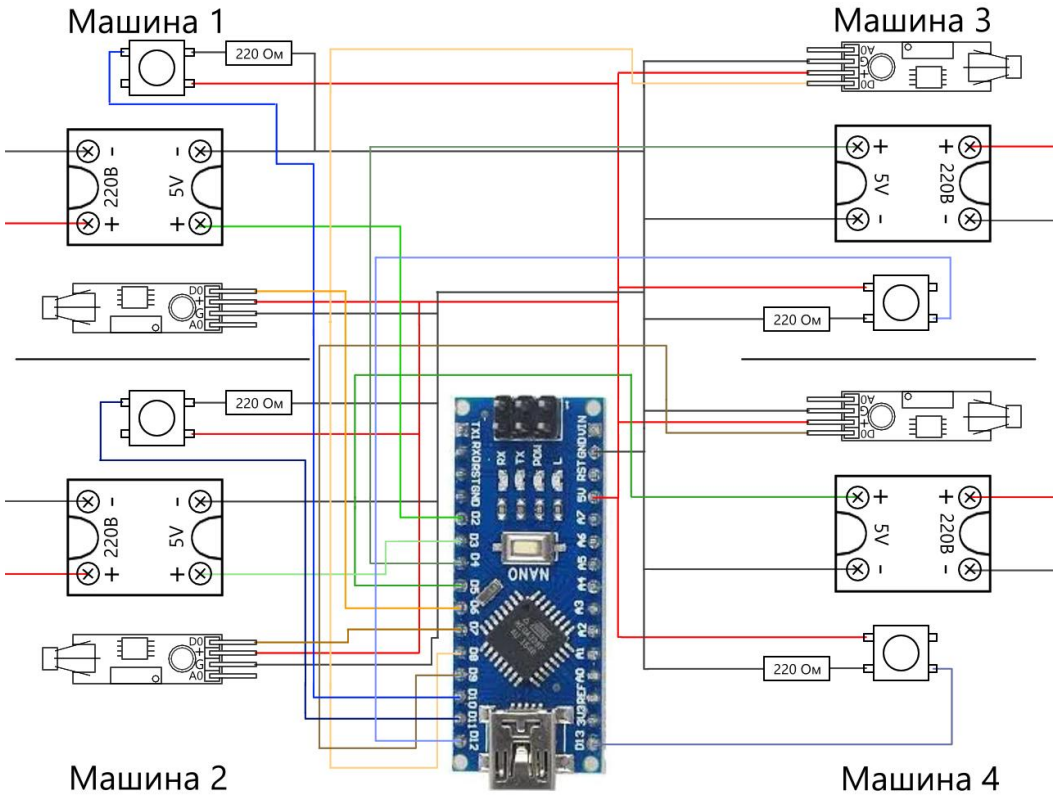

Рис. 3.15 Схема підключення системи з датчиками Холлу до машин, в яких принцип дії як у бобінорізки

3.3 Перевірка автоматизованої системи обліку, переналаштованої для бобінорізки

Підключимо Arduino NANO та «ArduinoLogger», треба перевірити робочий цикл. Перевіримо на прикладі машини під номером один та спробуємо емітувати оберти валу, підносячи магніт до датчика Холлу. Оскільки довжина кола рахувального валу в шість разів менше за один метр, то довелося редагувати код Arduino так, щоби шість спрацьовувань датчика Холлу дорівнювало одному метру погонному.

Також, лічильник виробленої продукції замінимо на кількість метрів погонних, бо на відміну від пакетоформувальної машини, бобінорізка не виробляє поштучну продукцію. Продукцією бобінорізки є рулони плівки та

вони вимірюється в метрах погонних.

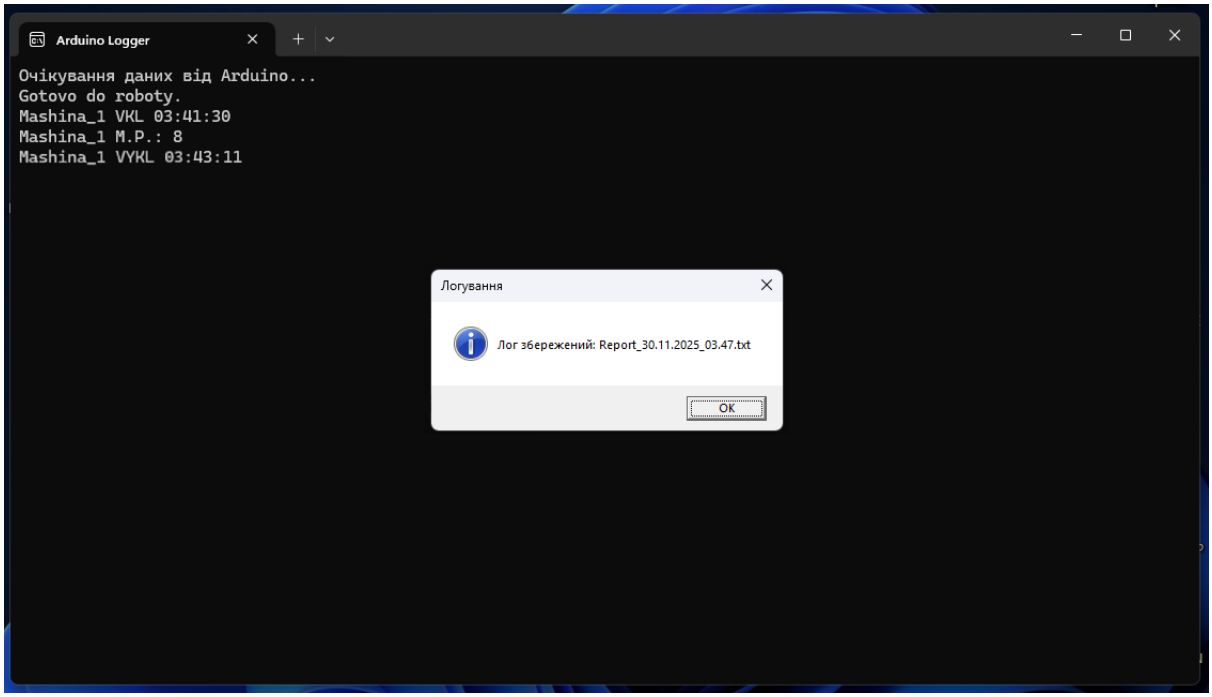

Рис. 3.16 Програма «ArduinoLogger», перевірка працездатності розробляємої системи

На робочому столі ОС Windows сформувався новий лог-файл під назвою «Report_30.11.2025_03.47».

Рис. 3.17 Сформований лог-файл на робочому столі ОС Windows

Перевіряємо зміст сформованого лог-файлу, він має містити все те, що містила консоль в час формування лог-файлу.

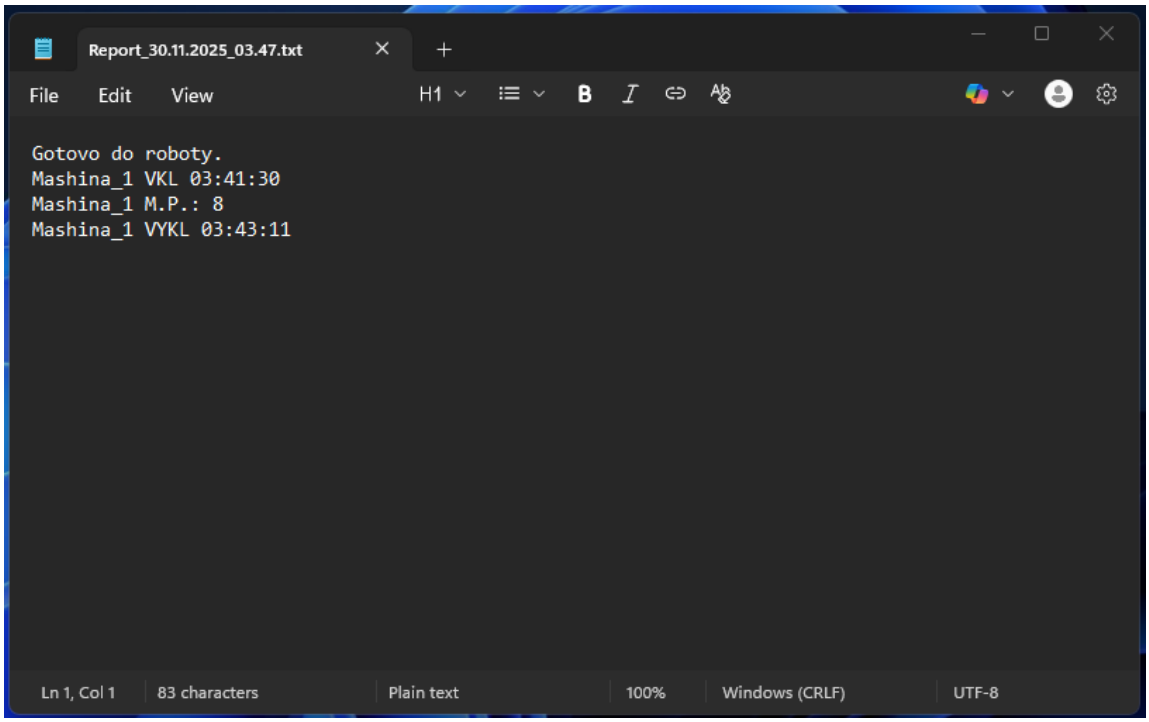

Рис. 3.18 Зміст лог-файлу.

В таблиці Excel теж необхідно змінити назву колонки з «Кількість виробленої продукції» на «Кількість м.п.»

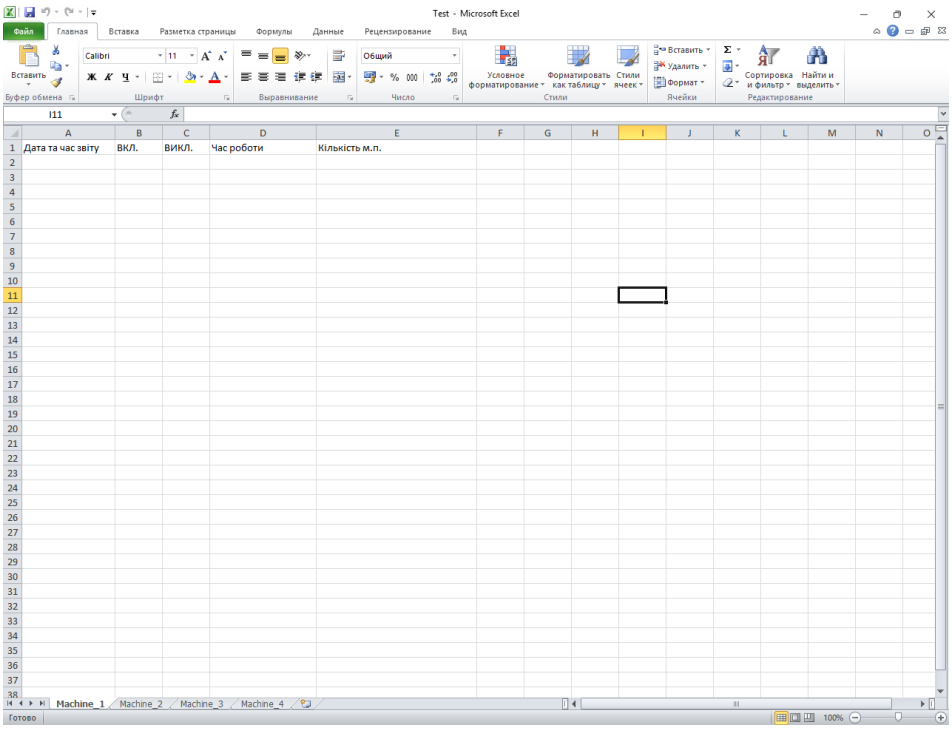

Рис. 3.19 Зміст файлу Excel зі зміненою назвою однієї з колонок.

Виконуємо процедуру перезапису файлу Excel виключно через програму «ExcelReportParser». Зараз необхідно вибрати лог-файл.

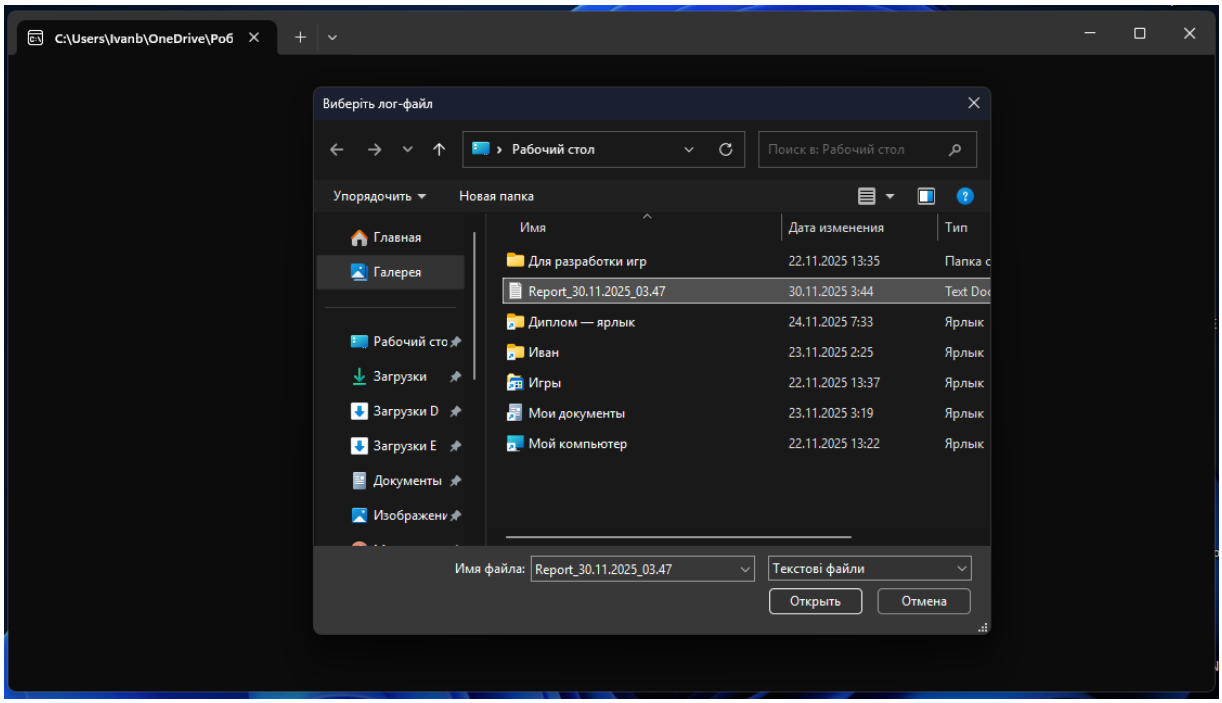

Рис. 3.20 Програма «ExcelReportParser», вікно вибору лог-файлу

Зараз необхідно вибрати файл Excel для перезапису та переконатися в тому, що система працює.

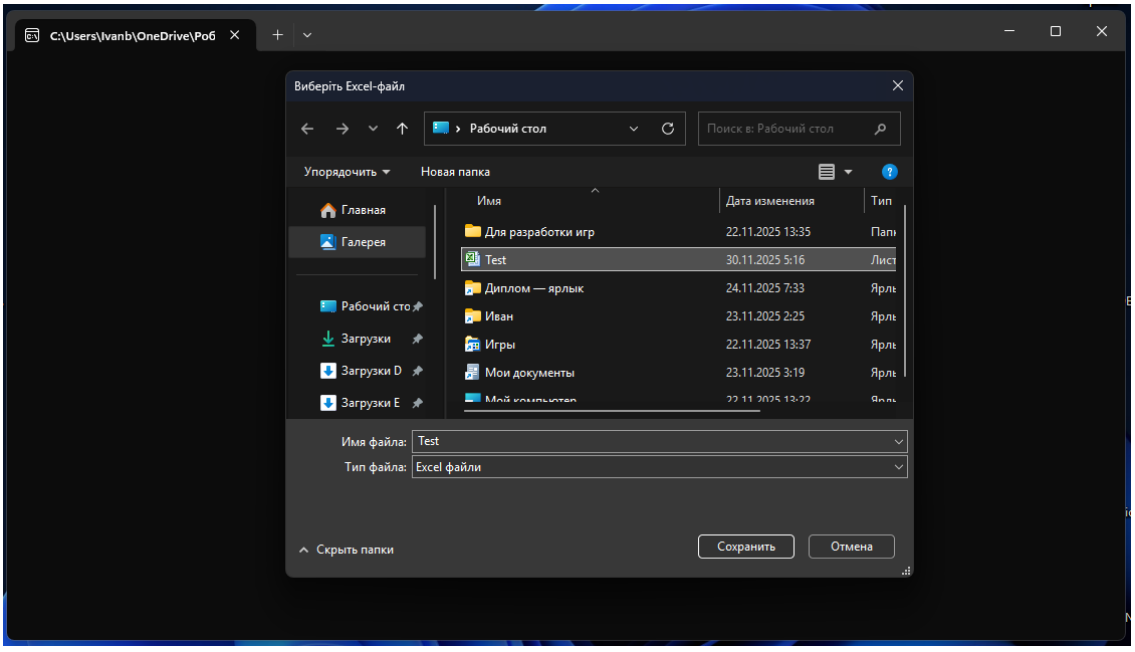

Рис. 3.21 Програма «ExcelReportParser», вікно вибору файлу Excel для перезапису

Переконуємося, що все коректно перезаписалося.

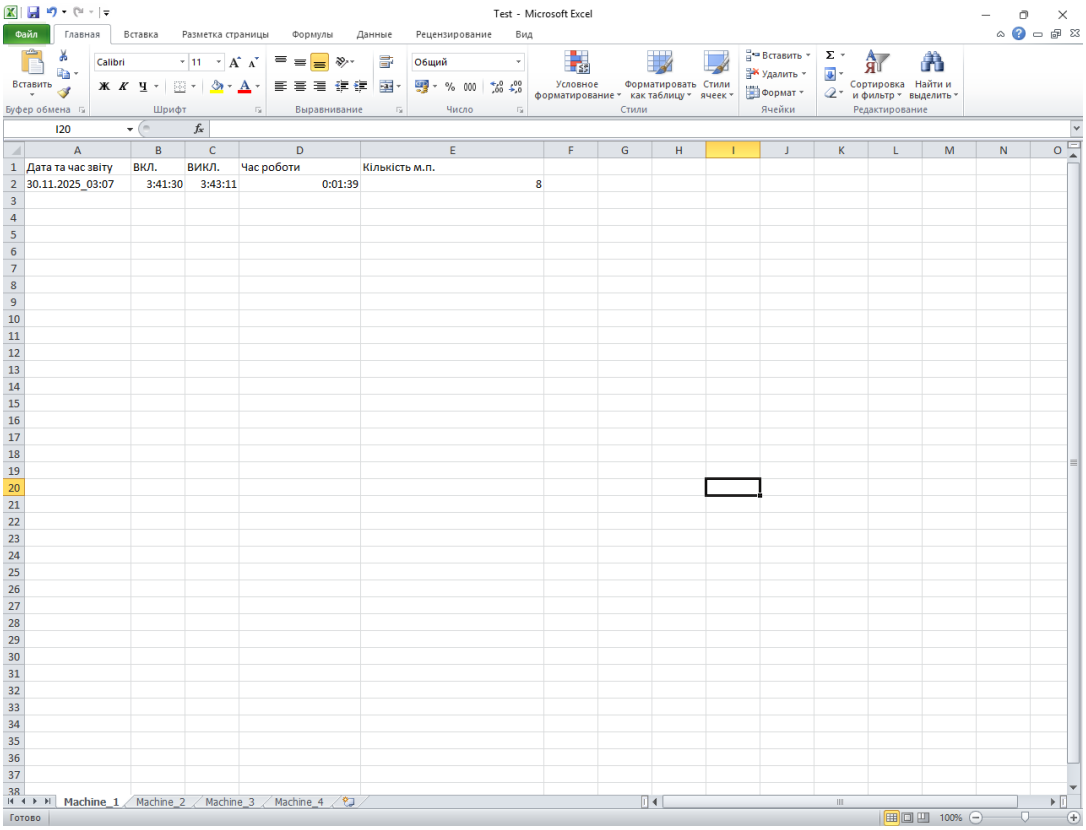

Рис. 3.22 Зміст файлу Excel після перезапису

ВИСНОВКИ

В цій кваліфікаційної роботі було наглядно розглянуто можливості використання мікроконтроллера Arduino та можливостей мови програмування C# для автоматизації обліку на підприємстві, на базі цеху з поліграфії, а саме безпосередньої передачі інформації від пакетоформуючої машини та бобінорізки до центрального комп'ютера цеху. В майбутньому можна буде використати приклад, описаний в цій кваліфікаційній роботі для створення подібних автоматизованих систем обліку на виробничих підприємствах інших направленостей. Це дасть можливість зменшити кількість помилок в роботі особи, відповідальної за розрахунок кількості матеріалу, значно зменшивши вплив людського фактору на ймовірність виникнення помилки. Також, система, що описана в даній кваліфікаційній роботі має встановлювати кількість запусків та зупинок виробничого процесу на виробничих машинах, що використовуються в цеху, щоби можна було помітити збій в роботі виробництва.

ПЕРЕЛІК ПОСИЛАНЬ

1. Wikipedia. Arduino URL: https://uk.wikipedia.org/wiki/Arduino

2. Nextpcb. The Ultimate Guide to Arduino Nano Pinout. URL: https://www.nextpcb.com/blog/arduino-nano-pinout

3. MyProject. TCRT5000 інфрачервоний датчик стеження URL: https://myproject.com.ua/tcrt5000-infrachervonij-datchik-stezhennja-ua.html?gclid=EAIaIQobChMIsbjbhf6UhgMVO1qRBR2Ifwn2EAQYBiABEgIbWfD_BwE&utm_source=google&utm_medium=cpc&utm_campaign=New_Company

4. Sxema. Датчик Холла для Arduino KY-024 магнітний лінійний, модуль швидкості Arduino URL: https://sxema.com.ua/ua/p743941707-datchik-holla-dlya.html?srsltid=AfmBOoouQ_Qzm3bHM4S2aToClrFCLyQXF3I4mlIR4LJ7rtECf0aMK0Zt

5. Prom. Твердотельне реле однофазне SSR-10 DD 10A вихід DC 12-220B URL: https://prom.ua/p12961533-tverdotelnoe-rele-odnofaznoe.html

6. Jesús López-Belmonte, José-Antonio Marín-Marín, Rebeca Soler Costa, Antonio José Moreno Guerrero. Arduino Advances in Web of Science. A Scientific Mapping of Literary Production

7. Massimo Banzi, Michael Shiloh. Getting Started with Arduino The Open Source Electronics Prototyping Platform 3rd Edition (2015)

8. John Boxall. Arduino Workshop, 2nd Edition: A Hands-on Introduction with 65 Projects (2021)

9. Jeremy Blum. Exploring Arduino: Tool and Techniques for Engineering Wizardry 2nd Edition (2019)

10.Браташов І.Р., Ткаленко О.М. «ВПРОВАДЖЕННЯ ТЕХНОЛОГІЙ IoT НА ПРОМИСЛОВОМУ ПІДПРИЄМСТВІ». Тези доповіді VIII Всеукраїнська науково-технічна конференція «Комп’ютерні технології: інновації, проблеми,

рішення» – Житомир, 3 грудня 2025 р.

11. Браташов І.Р., Ткаленко О.М. «ВПРОВАДЖЕННЯ СИСТЕМ IoT У РОБОТУ ПРОМИСЛОВОГО ОБЛАДНАННЯ». Тези доповіді III Всеукраїнська науково-технічна конференція «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу» – Київ, 11 листопада 2025 р.

12. Joseph Albahari, Ben Albahari. C# 9.0 in a Nutshell

13. Jon Skeet. C# in Depth

14. Andrew Stellman, Jennifer Greene. Head First C#

15. Bill Wagner. Effective C#: 50 Specific Ways to Improve Your C#

16. Banzi M. Getting Started with Arduino. Maker Media, 2011.

17.Banzi M., Shiloh M. Make: Getting Started with Arduino. Maker Media, 2011.

18. Blum J. Exploring Arduino: Tools and Techniques for Engineering Wizardry. Wiley, 2013.

19. Monk S. Programming Arduino: Getting Started with Sketches. McGraw-Hill Education, 2011.

20. Monk S. Arduino Cookbook. O’Reilly Media, 2016.

21. McRoberts M. Beginning Arduino. Apress, 2011.

22. McRoberts M. Arduino Cookbook, 3rd Edition. O’Reilly Media, 2014.

23. Boxall J. Arduino Workshop. No Starch Press, 2013.

24. McWhorter P. Arduino Programming and Projects. CreateSpace Independent Publishing, 2015.

25. Schwartz M. Arduino Programming for the Internet of Things. Packt Publishing, 2019.

26. Oxer J., Blemings H. Practical Arduino. Apress, 2009.

27. Allan A., Connery R. iOS Sensor Apps with Arduino. Wiley, 2011.

28. Evans B. Beginning Arduino Programming. Apress, 2011.

29. Ibrahim D. Advanced PIC Microcontroller Projects in C. Elsevier, 2008.

30. Catsoulis J. Designing Embedded Hardware. O'Reilly Media, 2005.

31. White E. Making Embedded Systems. O'Reilly Media, 2011.

32. Faludi R. Building Wireless Sensor Networks. O'Reilly Media, 2010.

33. Ganssle J. The Art of Designing Embedded Systems. Newnes, 2008.

34. Troelsen A., Japikse P. Pro C# and .NET. Apress, 2021.

35. Richter J. CLR via C#. Microsoft Press, 2012.

36. Skeet J. C# in Depth. Manning Publications, 2019.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

Державний університет інформаційно-комунікаційних технологій

Кафедра Інформаційних систем та технологій

МАГІСТЕРСЬКА РОБОТА

на тему:

"Автоматизована система обліку на виробництві з використанням Arduino та мови програмування C#"

Виконав: студент групи ІСДМ-61
Іван БРАТАШОВ

Керівник: к.т.н., доц. каф. ІСТ
Оксана ТКАЛЕНКО

Київ - 2026

- ***Актуальність теми:*** Створення автоматизованої системи обліку на підприємстві дозволяє уникнути помилок і спростити роботу особи відповідальної за облік продукції. Суть у тому, щоб створити електронну базу даних, у якій буде вся інформація про продукцію. Описана в цій дипломній роботі система дозволяє проводити точний облік продукції в реальному часі, а також контролювати час роботи і простою обладнання, що дозволить краще контролювати процес виробництва та виявляти слабкі місця виробничого ланцюга та подальшу ліквідацію недоліків виробництва.
- ***Мета роботи:*** створення автоматизованої системи обліку на виробничому підприємстві, за допомогою мікроконтролера Arduino та комп'ютерних програм, написаних мовою C#.
- ***Завдання дослідження:***
  - розробка системи та написання схем;
  - написання коду Arduino;
  - створення програм мовою C#;
  - перевірка працездатності створеної системи.
- ***Об`єкт дослідження:*** процес автоматизованої системи обліку.

Пакетоформувальна машина

Рисунок 1.1 Зовнішній вигляд пакетоформуючої машини

3D-модель пакетоформувальної машини

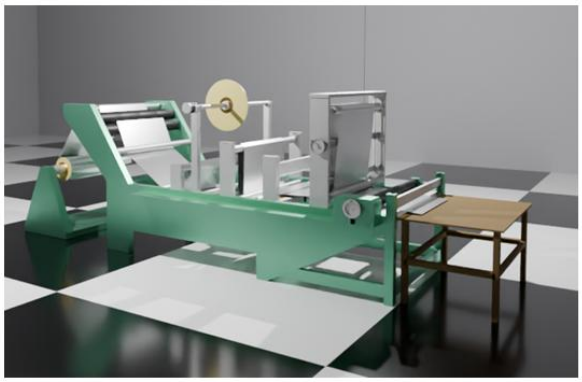

Рисунок 1.2 3D-модель пакетоформуючої машини, розроблена спеціально для цього проєкту

Схема логіки підключення автоматизованої системи обліку виробничого процесу

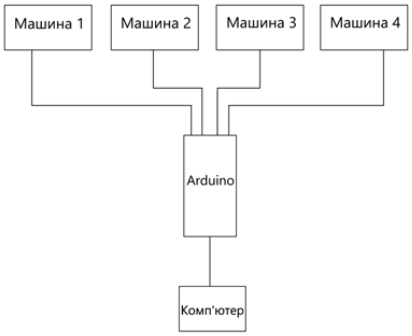

Рисунок 1.3 Схема логіки підключення автоматизованої системи обліку виробничого процесу

У виробничому цеху має знаходитися комп'ютер, який виконує роль бази даних, де зберігається інформація про вироблену продукцію. До нього має бути підключений мікроконтролер Arduino, метою якого є реєстрування сигналів з датчиків, які мають бути розташовані у вузлах виробничих машин. Також, задачею Arduino є запис інформації, переданої з датчиків, які після цього будуть зіставлені з часом, що виставлений на комп'ютері.

Схема підключення датчиків

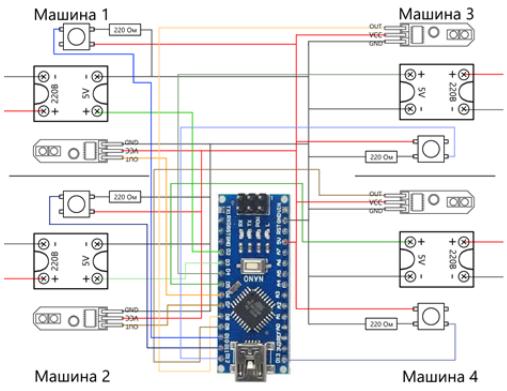

Рисунок 1.4 Схема підключення датчиків

На схемі зображена логіка підключення датчиків до мікроконтролера Arduino. Для кожної машини знадобиться один інфрачервоний датчик, що має рахувати кількість виробленої продукції, одне твердотільне реле, що має сигналізувати моменти вмикання та вимикання виробничого процесу на машині, а також знадобиться звичайна кнопка, яка має при натисканні, подавати на Arduino сигнал, щоби формувати прочерк, який означає кінець замовлення.

Паяльник та гільйотина

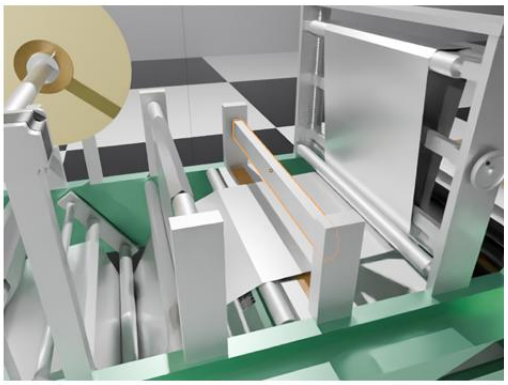

Рисунок 1.5 Паяльник

Паяльник влаштований таким чином, що у нього є дві точки дотика до полотна. Це дозволяє робити два шва одночасно.

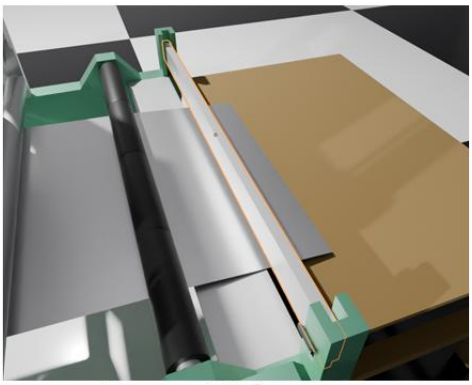

Рисунок 1.6 Гільйотина

Гільйотина, в свою чергу, відповідає за нарізання полотна на окремі пакети, суть в тому, що вона має попадати між двома швами на полотні, які робить паяльник. Ці шви – є краями пакетів.

Розташування інфрачервоного датчика

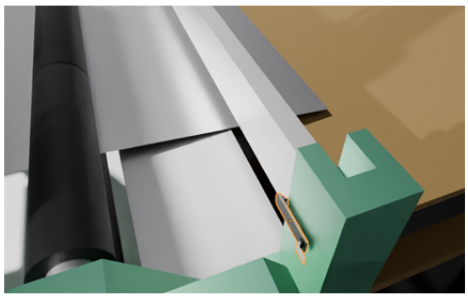

Рисунок 1.7 Розташування інфрачервоного датчика

Як було описано вище, інфрачервоний датчик відповідає за рахування кількості виробленої продукції. Краще всього його розташувати на місці, де їздить гільйотина. Оскільки гільйотина відповідає за нарізання полотна на окремі пакети, вона має рухаючись зачіпати місце, яке сприймає інфрачервоний датчик. Таким чином інфрачервоний датчик має рахувати кількість виробленої продукції.

Програма «ArduinoLogger»

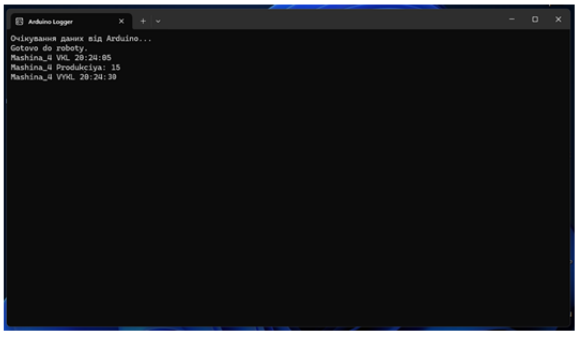

Рисунок 1.8 Вікно програми «ArduinoLogger»

Для реалізації програмної частини системи було вирішено написати дві програми на мові програмування C#. Перша програма, яку ми розглянемо це «ArduinoLogger». Ця програма представляє з себе вікно, яке нагадує командний рядок Windows. Її функція полягає в тому, що потрібно вибрати COM-порт, до якого під'єднаний мікроконтролер Arduino. Таким чином стає можливим бачити монітор порту Arduino, куди Arduino записує рядки інформації, отримані з датчиків. Кожен день о 21:00 «ArduinoLogger» автоматично створює лог-файл, куди розміщує всю інформацію, яка була записана під час сеансу програми, окрім першого рядку у якому написана налагоджувальна інформація. Також, комбінацією клавіш «Ctrl+L» можна примусово формувати лог-файл в будь-який час.

Лог-файл сформований програмою «ArduinoLogger»

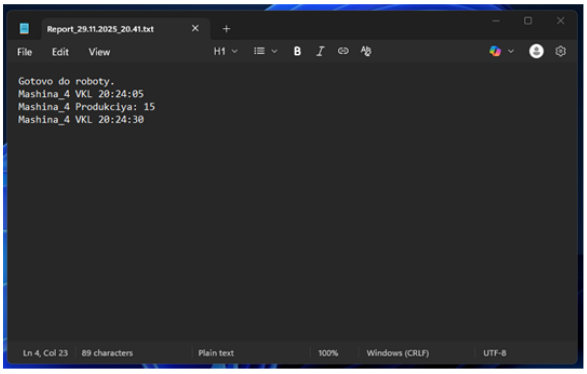

Рисунок 1.9 Лог-файл з інформацією про роботу машини

Кожен лог-файл являє собою текстовий документ, у якому пишеться вся інформація про стан роботи машин та кількість виробленої продукції за кожен сеанс запуску процесу виробництва машин. Також, позначається час, коли на машині було запущено процес виробництва продукції та коли цей процес був зупинений. При формуванні лог-файлу, йому дається ім'я «Report_поточна дата_поточний час». Наприклад, якщо лог-файл був створений о 21:00 21.12.2025, то його ім'я буде «Report_21.12.2025_21.00».

Програма «ExcelReportParser»

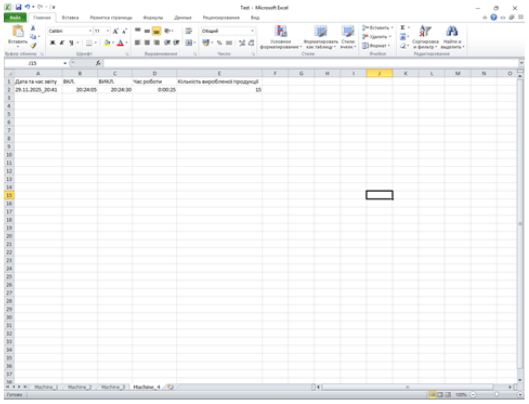

Рисунок 1.10 Результат записаного лог-файлу через «ExcelReportParser»

Друга програма, що була написана на мові програмування C#, називається «ExcelReportParser». Ця програма відповідає за коректний перезапис інформації з лог-файлу у таблицю Excel. При відкритті цієї програми, з'являється вікно з вибором лог-файлу, після вибору лог-файлу відкривається вікно з вибором Excel-файлу. Після вибору Excel-файлу, лог-файл знищується, а вся інформація записується у таблицю Excel. Кожен новий записаний лог-файл формує новий рядок в таблиці. А інформація про роботу кожної окремої машини, зберігається на окремих аркушах в Excel з аналогічними таблицями.

Висновки

- В результаті цієї кваліфікаційної роботи, були зроблені наглядні приклади, як можна використовувати мікроконтроллер Arduino для автоматизації системи обліку на виробничому підприємстві. Ця робота є прикладом, як автоматизація має спростити процес обліку на виробництві, зробивши процес виробництва ще більш автоматизованим, тим самим усуваючи можливі помилки, допущені через людський фактор.

- Апробація результатів магістерської кваліфікаційної роботи:

1. Браташов І.Р., Ткаленко О.М. «ВПРОВАДЖЕННЯ СИСТЕМ IoT У РОБОТУ ПРОМИСЛОВОГО ОБЛАДНАННЯ». Тези доповіді III Всеукраїнська науково-технічна конференція «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу» – Київ, 11 листопада 2025 р.

2. Браташов І.Р., Ткаленко О.М. «ВПРОВАДЖЕННЯ ТЕХНОЛОГІЙ IoT НА ПРОМИСЛОВОМУ ПІДПРИЄМСТВІ». Тези доповіді VIII Всеукраїнська науково-технічна конференція «Комп'ютерні технології: інновації, проблеми, рішення» – Житомир, 3 грудня 2025 р.

Дякую за увагу!