Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра автоматизації та комп'ютерно-інтегрованих технологій

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

АВТОМАТИЗОВАНА СИСТЕМА КЕРУВАННЯ ТЕХНОЛОГІЧНИМ ПРОЦЕСОМ ВИРОБНИЦТВА СОРОЧОК

AUTOMATED CONTROL SYSTEM FOR THE TECHNOLOGICAL PROCESS OF SHIRT PRODUCTION

Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка

освітня програма «Автоматизація та комп'ютерно-інтегровані технології»

Виконав: здобувач вищої освіти
групи АВмз - 21
Голіков Тимур Юрійович
_____
(підпис)

Керівник:
к. т. н., доцент
Гуменюк Лариса Олександрівна
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» _____ 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Гуменюк П. О.
_____
(підпис)

Луцьк – 2025

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій
Кафедра автоматизації та комп'ютерно-інтегрованих технологій
Ступінь вищої освіти: _*магістр*_
Галузь знань: *17 Електроніка, автоматизація та електронні комунікації*
Спеціальність: *174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка*
Освітня програма: *«Автоматизація та комп'ютерно-інтегровані технології»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О. Ю. Повстяной
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА
ДРУГОГО (МАГІСТЕРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ

Голікова Тимура Юрійовича
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *Автоматизована система керування технологічним процесом виробництва сорочок*

Керівник роботи: *к.т.н., доцент Гуменюк Лариса Олександрівна*

затверджені наказом закладу вищої освіти від « 27 » 06 2025 року N 304/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи: « 1 » 12 2025 року

3. Вихідні дані до роботи: *технологічний процес виробництва сорочок*

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

Аналіз технологічних процесів та сучасних автоматизованих систем у виробництві сорочок. Переваги C# в розробці програмного забезпечення. Застосування баз даних. Проєктування користувацького інтерфейсу

5. Перелік графічного матеріалу :

графічний матеріал виконано у вигляді презентації, яка складається з 10 слайдів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1	*Гуменюк Л. О.*		
Розділ 2	*Гуменюк Л. О.*		
Розділ 3	*Гуменюк Л. О.*		
Розділ 4	*Гуменюк Л. О.*		
Нормоконтроль	*Лапченко Ю. С.*		
Показник запозичень тексту			
Академічна доброчесність	*Федік Л. Ю.*		

7. Дата видачі завдання _____*27.06.2025 р.*_____

КАЛЕНДАРНИЙ ПЛАН

N з/п	Назва етапів кваліфікаційної роботи магістра	Строк виконання етапів роботи	Примітка
1	Аналіз проблеми за темою роботи та постановка задач	01.09.2025 р.	
2	Аналіз і вибір напрямків дослідження	10.09.2025 р.	
3	Теоретичне дослідження та практична реалізація	20.09.2025 р.	
4	Опис засобів розробки об’єкта проєктування	01.10.2025 р.	
5	Загальні висновки та рекомендації	20.10.2025 р.	
6	Оформлення роботи	10.11.2025 р.	
7	Оформлення презентації	20.11.2025 р.	
8	Здача чистового варіанту кваліфкаційної роботи на	01.12.2025 р.	
	кафедру		

Здобувач вищої освіти _____ _____Голіков Т. Ю._____
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи _____ _____Гуменюк Л. О._____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Голіков Т. Ю. Автоматизована система керування технологічним процесом виробництва сорочок. Рукопис.

Кваліфікаційна робота магістра ОП «Автоматизація та комп'ютерно-інтегровані технології» спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка. Луцький національний технічний університет, Луцьк 2025.

Кваліфікаційна робота магістра складається зі вступу, чотирьох розділів, висновків, переліку використаних джерел та додатків.

Робота присвячена дослідженню та автоматизації технологічного процесу виготовлення сорочок із застосуванням сучасних програмних та технічних засобів.

У роботі проаналізовано традиційні методи розкрою, шиття та прасування, визначено їхні недоліки та основні збурювальні фактори, що впливають на якість продукції. Проведено критичний огляд сучасних автоматизованих і роботизованих систем у швейній промисловості та окреслено обмеження їх інтеграції в єдиний виробничий цикл.

На основі аналізу розроблено концепцію автоматизації технологічних операцій пошиву сорочки та створено програмне забезпечення мовою C# для оптимізації послідовності виробничих процесів і зменшення трудомісткості операцій. Обґрунтовано вибір мови програмування, інтерфейсних технологій і бази даних для збереження параметрів тканин та налаштувань обладнання.

Обсяг пояснювальної записки становить 60 сторінок, графічна частина містить 10 слайдів презентації.

Ключові слова: автоматизація, швейне виробництво, C#, розкрій тканини, автоматизоване шиття, прасування, бази даних, технологічний процес.

ANNOTATION

Holikov T. Automated control system for the technological process of shirt production. Manuscript.

Master's qualification work of OP "Automation and computer-integrated technologies" specialty 174 Automation, computer-integrated technologies and robotics. Lutsk National Technical University, Lutsk 2025.

The master's qualification work consists of an introduction, four chapters, conclusions, a list of references, and appendices.

The work is devoted to the study and automation of the technological process of shirt manufacturing using modern software and technical tools.

The thesis analyzes traditional methods of fabric cutting, sewing, and ironing, identifies their shortcomings, and determines the main disturbance factors affecting product quality. A critical review of existing automated and robotic systems in the garment industry is conducted, along with an assessment of the limitations of their integration into a unified production cycle.

Based on this analysis, a concept for automating the technological operations of shirt production was developed, and software in C# was created to optimize the sequence of manufacturing processes and reduce labor intensity. The choice of programming language, interface technologies, and database architecture for storing fabric parameters and equipment settings is justified.

The explanatory note comprises 60 pages, and the graphic part includes 10 presentation slides.

Keywords: automation, garment manufacturing, C#, fabric cutting, automated sewing, ironing, databases, technological process.

ЗМІСТ

ВСТУП

Легка промисловість швидко розвивається під впливом автоматизації, концепції Індустрії 4.0, а також зростаючого попиту на масову індивідуалізацію та сталість виробництва. Традиційне виготовлення сорочок – засноване на ручних або напівавтоматизованих процесах розкрою, пошиття та прасування – є трудомістким, схильним до помилок і дорогим [1].

Останні досягнення в галузях робототехніки, машинного навчання та Інтернету речей трансформують процес виробництва одягу, забезпечуючи швидші виробничі цикли, зменшення кількості помилок і впровадження «розумного» виробництва. Автоматизація підтримує розвиток персоналізованої моди та виробництва «на вимогу», допомагаючи виробникам мінімізувати відходи, швидко реагувати на тенденції та задовольняти споживчий попит на одяг з індивідуальною посадкою та високою якістю.

Однак повна автоматизація процесу виготовлення сорочок залишається складним завданням через гнучкість і варіативність тканин. Сучасні системи добре справляються з окремими операціями (наприклад, лазерним розкроєм або зшиванням прямих швів), але мають труднощі з більш комплексними процесами, такими як пришивання коміра, вирівнювання тканини чи прасування складних форм. Більшість існуючих систем не мають достатньої інтеграції, адаптивності в реальному часі та безперервності робочих процесів, що призводить до неефективності та проблем із якістю.

Сучасні рішення автоматизації у виробництві одягу страждають від нестачі інтеграції. Пристрої для розкрою, пошиття та прасування функціонують як ізольовані одиниці, що потребують ручного втручання для переміщення тканини між етапами. Така фрагментація уповільнює виробничий процес, призводить до появи помилок (наприклад, зміщення або деформації тканини) та обмежує адаптивність системи. Без адаптивного керування в реальному часі

автоматизовані процеси не можуть ефективно реагувати на варіації або дефекти тканини, що створює «вузькі місця» та знижує якість продукції.

Метою роботи є програмна реалізація методів автоматизації технологічного процесу пошиву сорочки з розробкою програмного забезпечення на мові C# для автоматизації процесів, що дозволяє оптимізувати послідовність технологічних операцій та зменшити трудомісткість підготовки й виготовлення виробу.

Актуальність теми зумовлена зростаючими вимогами до швидкості, точності та стандартизації процесів у швейній промисловості. Автоматизація окремих етапів пошиву дозволяє значно зменшити трудові витрати, мінімізувати кількість помилок та підвищити якість готового виробу. У зв’язку з поширенням цифрових технологій у виробництві текстильних виробів зростає потреба у програмних засобах, які можуть підтримувати або частково виконувати операції, пов’язані з підготовкою та пошивом одягу. Швидкий розвиток програмних рішень та інструментів автоматизації відкриває нові можливості для оптимізації технологічних процесів, що є важливим фактором підвищення конкурентоспроможності сучасних виробництв.

Об’єктом дослідження є технологічний процес пошиву сорочки, що включає послідовність операцій з підготовки, обробки та виготовлення виробу.

Предметом дослідження є методи та програмні засоби автоматизації технологічних операцій пошиву сорочки, реалізовані засобами мови програмування C#.

Практична новизна роботи полягає у розробці програмного забезпечення, що дозволяє частково автоматизувати технологічний процес пошиву сорочки через оптимізацію послідовності операцій та зменшення їх трудомісткості. Запропоноване рішення підвищує ефективність роботи швейного виробництва та забезпечує можливість застосування цифрових технологій у процесах підготовки та виготовлення одягу.

У процесі виконання роботи необхідно вирішити наступні задачі.

1. Проаналізувати структуру технологічного процесу пошиву сорочки, визначивши ключові операції.

2. Оцінити сучасні методи та підходи до автоматизації технологічних процесів.

3. Вибрати та обґрунтувати мову програмування.

4. Розробити структуру бази даних для зберігання параметрів для автоматизованої системи керування операціями пошиву сорочок.

5. Розробити програмне забезпечення, яке забезпечить автоматизацію окремих етапів процесу пошиву сорочки та оптимізує послідовність виконання технологічних операцій.

Апробація результатів дослідження. Результати роботи доповідались та обговорювались на XIII Міжнародній науково-практичній інтернет-конференції молодих учених та студентів «Actual problems of automation and control». Опублікована стаття Голіков Т. Ю. Автоматизована система керування технологічним процесом пошиву сорочок. *Actual problems of automation and control:* матеріали XIII Міжнародної науково-практичної інтернет-конференції молодих учених та студентів. Випуск № 13. Луцьк: ЛНТУ, 2025. С. 42-45.

Копію статті приведено у додатку А.

РОЗДІЛ 1
АНАЛІЗ ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ ТА СУЧАСНИХ АВТОМАТИЗОВАНИХ СИСТЕМ У ВИРОБНИЦТВІ СОРОЧОК

1.1 Опис робочого процесу та його обмежень

Процес виготовлення сорочки традиційно включає кілька основних етапів:

− розкрій тканини: розкладання полотна, вирівнювання викрійок і ручне вирізання деталей;

− шиття: з’єднання частин виробу, зокрема швів, комірців, манжет і рукавів;

− прасування: відпарювання та остаточне оздоблення виробу для усунення зморшок і надання форми.

У традиційному виробництві сорочок кожен етап є трудомістким, займає багато часу та схильний до людських помилок. Перехід між операціями розкрою, шиття та прасування зазвичай є фрагментованим і передбачає ручне переміщення деталей між робочими місцями. Це підвищує ризик помилок, наприклад, перекосу або деформації тканини, а також знижує загальну швидкість виробництва.

Під час розкрою тканини робітники вручну розкладають шари тканини на розкрійних столах, розміщують зверху шаблони викрійок і використовують ручні інструменти (наприклад, кругові ножі або ножиці) для вирізання деталей. Цей процес часто супроводжується проблемами з неточним вирівнюванням, нерівними зрізами та перевитратою матеріалу, особливо у випадку складних викрійок або делікатних тканин. Кваліфіковані працівники мають ретельно вирівнювати та вирізати кожен елемент, що призводить до зниження ефективності й збільшення собівартості виробництва.

Шиття – це один із найбільш трудомістких етапів. Працівники керують швейними машинами, вручну вирівнюючи та подаючи тканину. Складні операції, як-от пришивання комірців, вшивання рукавів або обробка манжет,

вимагають високої точності та уважності. Через втому або неуважність можливі дефекти – перекоси, нерівні шви, зморшки тощо. Перехід на інший фасон або тип тканини часто потребує переналаштування обладнання, що сповільнює виробництво.

Прасування здійснюється вручну за допомогою парових прасок або пресів. Робітники ретельно пропрасовують кожен виріб, особливо зосереджуючись на складних ділянках – комірцях, манжетах, швах. Процес є тривалим і потребує досвіду, аби уникнути пошкодження делікатних тканин або нерівномірного оздоблення. Ручне прасування не гарантує стабільної якості – результат залежить від досвіду та уважності оператора.

1.2 Концептуальна основа автоматизованої системи

У традиційному виробництві сорочок кожний етап є трудомістким, витратним за часом і схильним до людських помилок. Процес є високофрагментованим: переходи між операціями розкрою, зшивання та прасування вимагають ручного перенесення деталей між робочими станціями. Така фрагментованість підвищує ризик помилок – зокрема деформації тканини, зміщення або нерівномірної якості – і суттєво знижує загальну швидкість виробництва. У середньому виготовлення однієї сорочки може займати від 1,5 до 3 годин залежно від складності, з істотною варіативністю у якості та ефективності.

Під час етапу розкрою працівники вручну розстилають шари тканини на великих столах, розміщують лекала зверху та використовують ручні інструменти (ротаційні ножі або ножиці) щоб вирізати окремі деталі. Цей процес є відомо неефективним та схильним до помилок. На розкрій компонентів однієї сорочки може йти від 20 до 45 хвилин, залежно від складності лекал і кількості шарів, що розкроюються одночасно.

CO_2-лазери не є стандартом у традиційному виробництві, деякі сучасні фабрики застосовують їх для високоточного розкрою. Лазер CO_2 споживає 1,5-3 кВт·год електроенергії на годину роботи та викидає приблизно 0,5-1 кг CO_2 за годину.

Ручний розкрій часто спричиняє 10-20 % відходів тканини через неточне вирівнювання, нерівні розрізи або неоптимальне розташування лекал. Для делікатних чи складних тканин цей показник може перевищувати 25 %.

Кваліфіковані працівники мають ретельно вирівнювати та вирізати кожен компонент, що уповільнює виробництво та підвищує витрати на працю. Невірне вирівнювання або нерівні розрізи можуть призвести до дефектів, які проявляються на пізніших етапах, ще більше затримуючи процес.

Зшивання є найбільш трудомістким етапом у виготовленні сорочок. Працівники керують швейними машинами, вручну вирівнюючи та подаючи тканину, що потребує високої точності та концентрації:

Пошиття сорочки зазвичай займає від 1 до 2 годин, а складні операції – такі як пришивання комірця, вшивання рукавів або обробка манжет – значно збільшують цей час. Налаштування машин для різних моделей чи типів тканин може додати ще 10-30 хвилин на кожен перехід.

Витрата ниток становить у середньому 50-100 метрів на одну сорочку; голки потрібно часто замінювати через зношення, що додає експлуатаційних витрат.

Втома або втрата концентрації можуть призвести до дефектів – зміщення швів, нерівних стібків чи зморщування тканини. Кожен дефект часто вимагає переробки, збільшуючи тривалість виробництва на 10-30 %. Перехід між різними моделями чи тканинами зазвичай вимагає переналаштування машин, що додатково уповільнює процес.

Прасування виконується вручну за допомогою парових прасок або пресів. Працівники ретельно пропрасовують кожен виріб, приділяючи особливу увагу комірам, манжетам та швам.

На прасування однієї сорочки може йти від 15 до 30 хвилин залежно від типу тканини та кваліфікації оператора. Праски споживають 0,5-1 літр води на годину роботи, в залежності від типу використаної тканини.

Процес значною мірою залежить від досвіду оператора. Нерівномірний тиск або неправильне застосування тепла можуть пошкодити делікатні тканини або спричинити нерівний зовнішній вигляд. Ручне прасування також не гарантує стабільної якості, оскільки результат залежить від уважності та майстерності працівника.

Традиційне виробництво сорочок є ресурсомістким, тривалим і схильним до неефективності. Залежність від ручної праці не лише підвищує собівартість, а й обмежує масштабованість і стабільність якості. Інновації, такі як автоматизовані системи розкрою, роботизоване зшивання та парові преси, можуть суттєво зменшити час, відходи та споживання енергії, трансформуючи галузь у напрямку більшої сталості та ефективності.

1.3 Критичний аналіз існуючих системи

Попри значні досягнення в галузі автоматизації та робототехніки, швейна промисловість і надалі стикається з істотними труднощами на шляху до повністю автономного виробництва сорочок. Більшість наявних автоматизованих систем обмежені виконанням окремих операцій – таких як лазерне різання, зшивання прямих швів або парове прасування – без можливості інтеграції цих процесів у єдиний безперервний виробничий цикл.

Наприклад, роботизовані швейні машини, подібні до тих, що розробляє компанія SoftWear Automation, здатні виконувати прості шви, але мають труднощі з більш складними операціями – такими як пришивання комірця, вшивання рукавів або робота з делікатними тканинами. Аналогічно, автоматизовані системи розкрою, хоча й забезпечують високу точність, часто потребують ручного втручання для вирівнювання тканини та виявлення

дефектів, що обмежує їхню ефективність і масштабованість [2].

Відсутність повністю інтегрованої системи, здатної автономно виконувати всі етапи – розкрій, пошиття та прасування готової сорочки – є критичною прогалиною у сучасному виробництві одягу. Основною причиною цього є складність маніпуляцій із тканинами: текстильні матеріали є гнучкими, деформівними та схильними до зсувів під час обробки, що ускладнює такі завдання, як вирівнювання, подавання та контроль натягу.

Сучасні рішення часто базуються на використанні попередньо стабілізованих або жорстких матеріалів, які не відображають різноманіття тканин, що застосовуються у реальному виробництві сорочок. Крім того, відсутність систем адаптивного керування в режимі реального часу призводить до того, що більшість автоматизованих процесів не можуть динамічно реагувати на зміни властивостей тканини чи на неочікувані дефекти, що викликає вузькі місця у виробництві та проблеми з якістю продукції.

Сучасні автоматизовані системи розкрою досягли значного прогресу у підвищенні точності та ефективності. Наприклад:

− Paragon від Gerber Technology використовує передові технології комп’ютерного зору та інтеграцію з CAD-системами для оптимізації процесу розкрою тканини, що дозволяє зменшити витрати матеріалу та підвищити точність. Однак система має труднощі з тканинами, які мають складні візерунки або еластичні властивості, тому часто потребує ручного коригування для вирівнювання матеріалу та виявлення дефектів [3];

− Vector від Lectra широко застосовується завдяки високій швидкості роботи та сумісності з різними видами матеріалів. Проте система стикається з обмеженнями при роботі з делікатними або дуже еластичними тканинами, що вимагає участі оператора для контролю якості [4];

− системи автоматизованого розкрою Kornit Digital поєднують цифровий друк із лазерним різанням, що дає змогу реалізовувати виробництво «на вимогу». Водночас ці системи менш ефективні при роботі з тканинами, схильними до

деформації, такими як трикотаж або рихло переплетені текстильні матеріали [5].

Роботизовані швейні машини значно вдосконалилися, проте все ще мають обмеження у виконанні складних операцій:

− Sewbots від SoftWear Automation призначені для автоматизації зшивання простих швів із використанням роботизованих маніпуляторів і систем комп'ютерного зору. Вони забезпечують високу точність при зшиванні прямих швів, але мають труднощі з більш складними операціями, такими як пришивання коміра або вшивання рукавів, особливо при роботі з еластичними чи нестабільними тканинами [6];

− роботизована система Sewbo використовує хімічний процес тимчасового ущільнення тканини, що полегшує її маніпуляцію роботами. Однак цей метод не підходить для всіх типів тканин і потребує додаткових технологічних етапів, які збільшують час і вартість виробництва [7];

− автоматизовані швейні блоки Brother оптимізовані для високошвидкісного виконання повторюваних операцій, але їм бракує гнучкості для роботи зі складними дизайнами одягу чи змінами властивостей тканин, що може спричиняти нерівномірну якість швів [8].

Автоматизовані системи прасування ефективно працюють із плоскими тканинами, однак стикаються з труднощами при обробці складних форм одягу:

− автоматизовані прасувальні машини Veit Group широко застосовуються завдяки високій швидкості та ефективності при обробці плоских виробів. Проте вони потребують ручного регулювання при прасуванні елементів із дрібними деталями, такими як коміри чи манжети, і є менш ефективними для делікатних або структурованих тканин [9];

− роботизовані системи прасування Meyer відзначаються високим технічним рівнем, однак функціонують як автономні модулі, що потребують ручного переміщення виробів між виробничими станціями. Це порушує безперервність процесу та знижує загальну ефективність, особливо у високопродуктивних виробничих середовищах [10].

Автоматизація шиття менш розвинена у порівнянні з процесами різання та прасування. Більшість операцій все ще виконуються на напівавтоматичних машинах (наприклад, прямий шов, оверлок) із залученням людини для складних завдань (наприклад, пришивання коміра, встановлення рукава).

Рівні автоматизації:

− повністю автоматизовані: виконання петель, пришивання ґудзиків та простих швів;

− напівавтоматичні: збірка коміра/манжет, пришивання кишень та встановлення рукава (потребує керівництва оператора);

− ручні: дуже варіабельні завдання, такі як обробка тканини, контроль якості та переробка.

Висновок до розділу 1

Проаналізовано основні етапи технологічного процесу виготовлення сорочок, зокрема розкрій тканини, шиття та прасування. Визначені ключові параметри технологічного процесу, які необхідно дотримуватися для забезпечення якості та ефективності виробництва. Поставлено задачі автоматизації, а саме: контроль і регулювання параметрів розкрою, шиття та прасування, автоматичне налаштування обладнання залежно від типу тканини та моделі, а також дистанційне керування роботою швейних машин і прасувальних пресів. Розглянуто взаємодію ключових параметрів для кожного етапу виробництва, що дозволяє визначити оптимальні умови для автоматизації та підвищення ефективності процесу.

Рівні автоматизації у виробництві сорочок охоплюють як повністю автоматизовані, так і напівавтоматичні та ручні операції. До повністю автоматизованих належать процеси виконання петель, пришивання ґудзиків і виконання прямих швів, які машини можуть здійснювати без участі оператора. Напівавтоматичні етапи, такі як збирання коміра чи манжет, пришивання кишень

або встановлення рукава, усе ще потребують контролю та залучення працівника. Водночас значна частина робіт залишається повністю ручною: це обробка тканини, контроль якості, переробка виробів, а також виконання операцій зі складними дизайнами або нестабільними тканинами, які вимагають високої чутливості та гнучкості, поки що недоступних автоматизованим системам.

Критичні прогалини у розвитку галузі пов’язані насамперед із відсутністю повністю інтегрованих рішень, здатних автономно виконувати весь цикл виробництва – від розкрою матеріалу до фінального прасування готової сорочки. Існуючі автоматизовані системи залишаються обмежено гнучкими, оскільки здебільшого оптимізовані під повторювані операції й погано пристосовані до різноманітності тканин, конструкцій та дизайнерських рішень. Додаткові технологічні етапи часто ускладнюють процес і збільшують як загальний час виробництва, так і його собівартість.

Перспективи розвитку цієї галузі пов’язані з упровадженням комплексних рішень, що передбачають інтеграцію окремих автоматизованих модулів у єдину систему з адаптивним керуванням у реальному часі. Подальший поступ м’якої робототехніки та штучного інтелекту створює можливості для вдосконалення маніпуляцій із тканинами та забезпечення оперативного реагування на зміни їхніх властивостей. Водночас важливо орієнтуватися на потреби малих і середніх підприємств здатні підвищити їхню ефективність без надмірних інвестицій.

Практичні рекомендації передбачають різні підходи для підприємств різного масштабу. Великим виробництвам варто інвестувати в інтеграцію вже наявних автоматизованих систем і розробку адаптивних технологій, що підвищать гнучкість і ефективність виробничих процесів. Малим підприємствам доцільно розпочати з упровадження CAD/CAM-рішень та технологій Інтернету речей, які допоможуть оптимізувати розкрій матеріалів і здійснювати постійний моніторинг стану обладнання. Важливо також приділяти увагу підготовці персоналу: фахівці мають бути навчені працювати з автоматизованими системами та грамотно поєднувати роботу ручних і автоматизованих процесів.

РОЗДІЛ 2
ПЕРЕВАГИ C# В РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Обгрунтування використання мови програмування C#

Вибір C# у поєднанні з фреймворком Windows Forms (WinForms) забезпечує значні технічні та операційні переваги для розроблення інтерфейсів користувача та програмного забезпечення керування для автоматизованих систем шиття, різання та прасування. Ці переваги зумовлені зрілістю екосистеми .NET, стабільністю середовища Windows в промислових умовах та придатністю WinForms для проєктування орієнтованих на оператора інтерфейсів [11].

Промислові ПК на базі Windows широко використовуються у виробництві завдяки тривалій підтримці, стандартизованій доступності драйверів та сумісності з широким спектром апаратних постачальників. Застосунки на C#, побудовані за допомогою WinForms, працюють надійно в цій екосистемі, використовуючи стабільну поведінку під час виконання та передбачувану взаємодію з апаратними інтерфейсами, такими як послідовні порти, USB-пристрої та Ethernet-модулі зв’язку. Така стабільність є критично важливою для систем, що вимагають безперервної роботи та мінімальних простоїв, забезпечуючи неперервність виробничих процесів.

WinForms пропонує легковаговий та зрілий фреймворк для побудови інтерфейсів, який сприяє швидкому прототипуванню та ітеративній розробці. Його засоби побудови інтерфейсу методом перетягування та подієво-орієнтована архітектура спрощують процес розроблення, що є особливо корисним на ранніх етапах тестування та під час промислових випробувань. Чіткі принципи об’єктно-орієнтованого програмування в C#, разом із вбудованим керуванням пам’яттю та великою бібліотекою багаторазових компонентів .NET, підвищують підтримуваність і спрощують довгострокові оновлення програмного забезпечення. Ці властивості дозволяють інженерним командам ефективно

розширювати або модифікувати систему відповідно до змін експлуатаційних вимог.

Критичною вимогою для автоматизованого виробничого обладнання є надійний обмін даними з датчиками, виконавчими механізмами та програмованими логічними контролерами (PLC). Екосистема .NET пропонує широкий спектр бібліотек для поширених промислових протоколів, включно з Modbus TCP/RTU, OPC UA та різними спеціалізованими API від виробників. Використовуючи C#, розробники можуть легко реалізувати ці комунікаційні рівні, що дає змогу застосунку відстежувати параметри процесу, запускати операції машини та синхронізувати роботу кількох підсистем, таких як модулі шиття, різання та прасування.

Хоча WinForms не призначений для жорстких реального часу завдань, він чудово підходить для наглядових функцій, моніторингу системи та взаємодії з оператором. C# забезпечує потужну підтримку асинхронних операцій, багатопотоковості та фонових завдань, що є необхідним для безперервного збирання даних з датчиків і відображення станів машини в реальному часі. Оператори та технічний персонал отримують своєчасні відомості щодо таких параметрів, як температура, швидкість або прогрес циклу, що дозволяє оперативно реагувати на будь-які відхилення процесу.

Порівняно з більш сучасними чи складними фреймворками інтерфейсів, WinForms зберігає низький поріг входу, що робить його доступним для інженерних команд і розробників промислового програмного забезпечення, які часто вже знайомі з екосистемою .NET. Така доступність знижує витрати на розробку та гарантує, що обслуговування системи може виконувати широке коло фахівців, навіть у випадках, коли потрібна тривала підтримка життєвого циклу.

Попри досягнутий вік, WinForms продовжує отримувати підтримку в сучасних версіях .NET і користується перевагами великої кількості документації, ресурсів спільноти й сторонніх компонентів. Його зрілість і зворотна сумісність роблять його надійним вибором для промислових HMI, де першочергово

важливими є надійність і довговічність, а не найновіші графічні можливості. Це забезпечує збереження підтримуваності та адаптивності програмного забезпечення навіть у випадку модернізації апаратної платформи чи зміни експлуатаційних потреб.

2.2 Переваги C# у розробці інтерфейсів для роботи з промисловими машинами

Під час розроблення інтерфейсів користувача та прикладного програмного забезпечення для промислових машин шиття, різання та прасування C# пропонує низку суттєвих переваг порівняно з такими мовами, як Java, Python та інші. Ці переваги зумовлені глибокою інтеграцією з екосистемою Windows, розвинутими інструментами розробки та продуктивністю, орієнтованою на потреби промислової автоматизації.

C# тісно інтегрований з операційною системою Windows, яка залишається домінуючою платформою для промислових ПК. Така інтеграція забезпечує безперебійну сумісність з апаратними інтерфейсами, такими як послідовні порти, USB-пристрої та Ethernet-модулі зв’язку – типовими компонентами керування промисловими машинами. Java та Python також можуть взаємодіяти з апаратним забезпеченням, проте вони часто вимагають додаткових бібліотек або проміжного програмного забезпечення, що може ускладнювати систему та створювати додаткові точки відмови. Підтримка P/Invoke та COM Interop у C# спрощує прямий обмін даними з обладнанням, скорочуючи час розробки та підвищуючи надійність.

C# є компільованою мовою, що працює на .NET Runtime, оптимізованому для застосунків із високими вимогами до продуктивності. На відміну від інтерпретованих мов, таких як Python, застосунки мови програмування C# використовують Just-In-Time (JIT) компіляцію, що забезпечує швидше виконання та нижчу затримку – критичні фактори в реальних промислових

умовах. Java, хоча також компілюється в байт-код, зазвичай має більші накладні витрати на пам’ять через роботу на віртуальній машині JVM. Ефективне керування пам’яттю в C#, включаючи автоматичне збирання сміття, додатково підвищує стабільність під час тривалих операцій, що є важливим для систем, які повинні працювати безперервно без деградації продуктивності [12].

Поєднання C# і Visual Studio забезпечує винятково зручне середовище розробки для створення промислових HMI. Конструктор інтерфейсів із підтримкою перетягування, інтегровані засоби налагодження та широка бібліотечна підтримка прискорюють прототипування й ітеративну розробку. Хоча Java також пропонує подібні IDE (IntelliJ IDEA, Eclipse), а Python має інструменти на кшталт PyCharm, зрілість і глибина інтеграції Visual Studio з C# і WinForms робить його особливо придатним для промислового проєктування інтерфейсів. Це зменшує криву навчання і дозволяє розробникам зосередитися на ключовій функціональності, а не на службовому коді.

Сувора типізація C# і перевірки на етапі компіляції мінімізують помилки під час виконання, що є критично важливим для промислових систем, де неочікувані збої можуть спричинити значні простої. Динамічна типізація Python, хоча й гнучка, може призвести до помилок, які складніше діагностувати в складних промислових середовищах. Java має аналогічну строгість типів, однак C# часто вважається більш продуктивним завдяки сучаснішому синтаксису та просунутим можливостям, таким як LINQ та async/await.

Екосистема .NET забезпечує широкий вибір бібліотек, створених спеціально для промислової автоматизації, зокрема для протоколів Modbus, OPC UA та різноманітних власницьких API обладнання. Хоча Java та Python теж пропонують підтримку цих протоколів, інтеграція C# із Windows та наявність сторонніх компонентів WinForms (ComponentOne, DevExpress, Syncfusion) значно спрощують створення функціонально насичених інтерфейсів. Ці компоненти оптимізовані для промислових HMI та забезпечують розширені можливості візуалізації даних, звітності та моніторингу в реальному часі.

C# і .NET тривалий час застосовуються в промисловій автоматизації, мають широку спільноту розробників і великий обсяг документації. Така зрілість гарантує довготривалу підтримку та сумісність із застарілими системами – важлива вимога для виробничого обладнання, термін служби якого може тривати десятиліттями. Java, хоча також є зрілою, частіше використовується в корпоративних і веб-застосунках, тоді як Python у промислових середовищах поки що менш поширений, особливо для HMI на базі Windows.

Застосунки C# використовують вбудовані механізми безпеки Windows, такі як контроль доступу до коду та рольова автентифікація, що є критично важливими для захисту промислових систем від несанкціонованого втручання. Програми Java вимагають JVM, що додає складності під час розгортання, тоді як Python-застосунки часто залежать від зовнішніх бібліотек, які потрібно окремо керувати [13].

Висновок до розділу 2

Для застосунків у сфері промислових машин шиття, різання та прасування C# пропонує переконливе поєднання продуктивності, надійності та зручності розробки. Його нативна інтеграція з Windows, строга типізація, багата екосистема промислових бібліотек та розвинуті інструменти роблять його кращим вибором порівняно з Java, Python та іншими мовами для створення високопродуктивного, підтримуваного й зручного програмного забезпечення для промислових систем підтримки.

Комбінація C# і Windows Forms забезпечує потужне, стабільне та ефективне рішення для розроблення інтерфейсів і програмного забезпечення нагляду в автоматизованих системах шиття, різання та прасування. Здатність цих технологій до безшовної інтеграції з обладнанням, підтримка моніторингу в реальному часі та зручність середовища розробки роблять їх оптимальним вибором для інженерних команд.

РОЗДІЛ 3
ЗАСТОСУВАННЯ БАЗ ДАНИХ

3.1 Обгрунтування використання баз даних

Використання бази даних для зберігання та керування попередньо заданими параметрами тканин – такими як густина стібка, натяг нитки, температура прасування, тиск пари, потужність лазера, швидкість різання та інші налаштування обладнання – є необхідним для забезпечення точності, ефективності та стабільності в промисловій обробці текстилю [14]. Нижче наведено причини, чому база даних є не просто корисною, а часто незамінною.

База даних забезпечує централізоване сховище всіх параметрів тканин та машин, завдяки чому кожен оператор і кожна машина отримують доступ до однакових стандартизованих налаштувань. Це усуває розбіжності, спричинені помилками ручного введення або неконсистентними конфігураціями на різних машинах. Наприклад, зберігання густини стібка (стік на сантиметр) та натягу нитки (Ньютони) в базі даних гарантує, що кожна операція шиття відповідає визначеним стандартам якості, зменшуючи кількість дефектів і переробок.

У високотоварних виробничих середовищах час є критичним ресурсом. База даних дозволяє операторам швидко отримувати та застосовувати попередні налаштування для різних типів тканин або технологічних операцій. Наприклад, під час переходу від бавовни до змішаних тканин машина може автоматично завантажити правильну температуру прасування (°C) та час прасування (с) із бази даних, мінімізуючи час налаштування та людські помилки.

Сучасна текстильна обробка охоплює широкий спектр матеріалів, кожен із яких потребує унікальних параметрів. База даних може зберігати кілька профілів для різних типів тканин, включаючи параметри, такі як тиск пари (Бар), потужність лазера (Вт) і швидкість різання (мм/с). Така гнучкість забезпечує швидку перенастройку машин для нових матеріалів, підвищує універсальність і зменшує простої.

Завдяки журналюванню та аналізу параметрів у часі база даних дозволяє впроваджувати контроль якості на основі даних. Наприклад, відстеження частоти імпульсів (Гц), діаметра променя (мм) та тиску газу (Бар) при лазерному різанні допомагає виявляти тенденції чи аномалії, що можуть впливати на якість продукції [15]. У разі відхилення система може повідомити про проблему або автоматично скоригувати параметри для підтримки стабільності.

База даних фіксує історичні відомості про використання параметрів, що є надзвичайно важливим для простежуваності та відповідності стандартам. Якщо виникає проблема якості, інженери можуть переглянути точні налаштування – наприклад, позицію фокусу (мм) або натяг нитки (Н) – щоб визначити першопричину. Історичне відстеження також підтримує ініціативи постійного вдосконалення, забезпечуючи дані про те, які параметри дають найкращі результати для конкретних тканин.

Сучасні промислові машини часто працюють як частина автоматизованої виробничої лінії. База даних забезпечує безперервну інтеграцію з ПЛК (програмованими логічними контролерами) та іншими системами автоматизації. Наприклад, після вибору типу тканини база даних може автоматично передати необхідну густину стібка, потужність лазера та швидкість різання на машину, забезпечуючи синхронізовану роботу всієї лінії.

У великомасштабному виробництві може бути необхідно забезпечити однакову роботу кількох машин. Централізована база даних гарантує використання однакових параметрів усіма машинами, забезпечуючи стабільність продукції на всьому виробничому майданчику. Така масштабованість особливо важлива для галузей, де рівномірність – наприклад, температура прасування або тиск пари – є критичною для виконання вимог замовника.

Бази даних дозволяють реалізувати розмежування доступу за ролями, забезпечуючи можливість змінювати критичні параметри лише для авторизованого персоналу. Це запобігає несанкціонованим змінам, що можуть спричинити несправності обладнання або погіршення якості продукції.

Наприклад, лише керівники можуть мати дозвіл змінювати параметри потужності лазера чи тиску газу, що знижує ризик операторських помилок.

Зберігаючи дані про параметри у часі, база даних може забезпечувати машинні навчальні моделі інформацією для прогнозування оптимальних налаштувань для нових тканин або для виявлення зношування компонентів машини. Наприклад, аналіз тенденцій у натягу нитки або швидкості різання може допомогти передбачити можливий вихід вузла з ладу, що дозволяє виконати технічне обслуговування заздалегідь і зменшити непланові простої.

Багато галузей потребують суворого дотримання стандартів якості та безпеки. База даних забезпечує надійний журнал подій, який підтверджує, що машини працювали в межах встановлених параметрів (наприклад, час прасування, потужність лазера).

3.2 Пояснення побудови бази даних

Для забезпечення автоматизованого налаштування обладнання та стабільності технологічних процесів у системі створено окрему таблицю бази даних, яка зберігає попередньо визначені параметри (пресети) для різних типів тканин. Таблиця використовується програмним забезпеченням машини для автоматичного завантаження оптимальних режимів шиття, прасування та лазерного різання під час переходу між матеріалами. Така структура даних мінімізує людський фактор, скорочує час переналаштування та забезпечує відповідність продукції встановленим стандартам якості.

Код створення бази даних наведено в лістингу 3.1.

Лістинг 3.1 – Код створення бази даних

```
CREATE TABLE `fabricdb`.`fabric` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `FabricType` VARCHAR(100) NOT NULL,
```

```
`StitchDensity` DECIMAL(5,2) NOT NULL,
`ThreadTension` DECIMAL(6,2) NOT NULL,
`PressTempBlended` DECIMAL(5,2) NULL,
`SteamPressure` DECIMAL(5,2) NULL,
`PressingTime` INT NULL,
`LaserPower` DECIMAL(6,2) NULL,
`CuttingSpeed` DECIMAL(6,2) NULL,
`FocusPosition` DECIMAL(6,2) NULL,
`GasPressure` DECIMAL(5,2) NULL,
`PulseFrequency` DECIMAL(6,2) NULL,
`BeamDiameter` DECIMAL(6,2) NULL,
PRIMARY KEY (`FabricType`),
UNIQUE INDEX `Id_UNIQUE` (`Id` ASC) VISIBLE,
UNIQUE INDEX `FabricType_UNIQUE` (`FabricType` ASC) VISIBLE);
```

кінець лістингу 3.1

Опис полів бази даних:

− Id: унікальний ідентифікатор для кожного запису, що забезпечує можливість посилання на кожен запис програмно;

− FabricType: первинний ключ, що представляє тип тканини (наприклад, «Бавовна», «Суміш поліестеру»). Це поле гарантує, що для кожного типу тканини існує єдиний стандартизований набір параметрів;

− StitchDensity: кількість стібків на сантиметр, важлива для якості шиття та довговічності тканини;

− ThreadTension: натяг нитки (вимірюється в ньютонах), який впливає на міцність стібка та його зовнішній вигляд;

− PressTemp: температури прасування для тканин, що забезпечують оптимальні результати прасування без пошкодження матеріалу;

− SteamPressure: тиск пари під час прасування (вимірюється в барах), що впливає на видалення зморшок та обробку тканини;

− PressingTime: тривалість прасування (в секундах), що впливає на ефективність та обробку тканини;

− LaserPower: потужність лазера (в ватах), що використовується для різання, впливає на точність та якість країв;

− CuttingSpeed: швидкість руху лазера або леза (в міліметрах за секунду), що забезпечує баланс між швидкістю та якістю різання;

− FocusPosition: положення фокусу лазера (в міліметрах), критично важливе для чистоти та точності різів;

− GasPressure: тиск допоміжного газу (в барах), що використовується при лазерному різанні, впливає на якість та швидкість різання;

− PulseFrequency: частота імпульсів лазера (в герцах), яка впливає на гладкість та точність різів;

− BeamDiameter: діаметр лазерного променя (в міліметрах), що визначає ширину та глибину різів.

База даних слугує єдиним джерелом правди для всіх параметрів, специфічних для тканин, усуваючи невідповідності, які можуть виникати через ручні налаштування або помилки операторів. Використовуючи FabricType як первинний ключ, кожному типу тканини гарантується унікальний та стандартизований набір параметрів, забезпечуючи однорідність у всіх виробничих партіях.

Структурований формат дозволяє безперешкодно інтегрувати базу даних з промисловими машинами та ПЛК (програмованими логічними контролерами). Коли оператор обирає тип тканини, система автоматично може отримати та застосувати відповідні параметри, зменшуючи час налаштування та мінімізуючи людські помилки.

Використання типу даних DECIMAL забезпечує точність для критичних параметрів, таких як StitchDensity, LaserPower та CuttingSpeed. Ця точність є

ключовою для підтримки якості продукції та дотримання галузевих стандартів. Обмеження NOT NULL на обов’язкових полях (наприклад, StitchDensity, ThreadTension) забезпечують повноту даних, запобігаючи неповним або некоректним конфігураціям.

Включення параметрів, специфічних для тканин, таких як PressTemp, дозволяє системі обробляти широкий спектр матеріалів, кожен з унікальними вимогами до обробки. Така гнучкість є необхідною для галузей, що працюють з кількома типами тканин, забезпечуючи оптимальну обробку для кожного матеріалу.

Зберігаючи параметри у базі даних, оператори можуть швидко перемикатися між типами тканин без ручного переналаштування машин. Це зменшує час простою та підвищує продуктивність, особливо у високовиробничих середовищах.

База даних дозволяє вести логування та аудит використання параметрів, що є критично важливим для відстежуваності та дотримання стандартів. У разі виникнення проблем з якістю, інженери можуть переглянути точні налаштування, використані для конкретного типу тканини, що полегшує аналіз причин та безперервне вдосконалення.

Дизайн підтримує легке розширення, дозволяючи додавати додаткові параметри або типи тканин за потреби. Це забезпечує адаптацію системи до змінних виробничих вимог або нових інновацій у тканинах.

Структурований формат даних сумісний із передовими системами, такими як моделі машинного навчання, які можуть аналізувати історичні дані для оптимізації параметрів або прогнозування потреб у технічному обслуговуванні. Наприклад, тенденції у LaserPower чи CuttingSpeed можуть використовуватися для визначення оптимальних налаштувань або виявлення потенційного зносу обладнання.

База даних може бути інтегрована із системами автентифікації користувачів, забезпечуючи, що лише уповноважений персонал може змінювати

критично важливі параметри. Це зменшує ризик несанкціонованих змін, які можуть вплинути на якість продукції або роботу машин.

Висновок до розділу 3

Використання бази даних для керування попередніми параметрами тканин є ключовим елементом сучасної промислової текстильної обробки. Це забезпечує стабільність, ефективність та простежуваність, а також підтримує автоматизацію, контроль якості й безперервне вдосконалення. Централізуючи та стандартизуючи критично важливі налаштування машин, система бази даних підвищує надійність виробничих процесів і якість кінцевої продукції.

Запропонована таблиця бази даних є фундаментальним компонентом промислової системи обробки тканин, що забезпечує стандартизоване, ефективне та високоякісне виробництво. Централізуючи попередньо задані параметри для тканин, дизайн підтримує автоматизацію, контроль якості та масштабованість, роблячи систему незамінним інструментом для сучасного виробництва. Такий структурований підхід гарантує, що машини працюють стабільно та оптимально незалежно від типу тканини чи обсягів виробництва.

РОЗДІЛ 4
ПРОЄКТУВАННЯ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

4.1 Пояснення ініціалізації та навігаційних елементів керування

Надійна система обробки введення є необхідною умовою для будь-якої промислової автоматизованої системи, оскільки ефективність оператора та точність керування безпосередньо впливають на продуктивність і безпеку. У цьому об’єднаному розділі розглядається базовий етап ініціалізації разом із функціональністю клавіш «ОК», «Вліво», «Вправо», «Вгору» та «Вниз», які спільно забезпечують структуровану та стійку до помилок навігацію в межах машини станів.

Процес ініціалізації встановлює передбачуваний і стабільний початковий стан застосунку, гарантуючи, що всі системні змінні, стани меню та компоненти інтерфейсу правильно налаштовані до початку взаємодії оператора з машиною. Визначення початкового стану та підготовка інтерфейсу до введення мінімізують невизначеність і підвищують надійність – ключову вимогу в промисловому середовищі. Початковий код ініціалізації наведено в лістингу 4.1.

Лістинг 4.1 – Початковий код ініціалізації

```
public Form1()
{
   InitializeComponent();
   DrawMainMenu();
}
private AppState currentState = AppState.MainMenu;
int selectedIndex = 0;
bool ignoreButt = false;
```

кінець лістингу 4.1

Конструктор Form1 ініціалізує головну форму застосунку та готує систему до взаємодії з користувачем, встановлюючи всі необхідні початкові стани та компоненти інтерфейсу:

− InitializeComponent(): стандартний метод Windows Forms, який створює та налаштовує всі елементи інтерфейсу, визначені у дизайнері форми;

− DrawMainMenu(): негайно відображає головне меню під час запуску, забезпечуючи користувача чіткою та інтуїтивною стартовою точкою;

− currentState: відображає поточний робочий стан застосунку (наприклад, MainMenu, FabricSelectMenu) та ініціалізується значенням AppState.MainMenu;

− selectedIndex: відстежує пункт меню, який наразі виділений або вибраний користувачем; початкове значення – 0 (перший пункт);

− ignoreButt: логічний прапорець, що використовується для усунення деренчання кнопок, запобігаючи ненавмисним повторним натисканням і забезпечуючи стабільну обробку введення.

Запуск застосунку у стані MainMenu забезпечує для операторів стабільне, послідовне та передбачуване робоче середовище щоразу після увімкнення системи. Прапорець «ignoreButt» є простим, але ефективним механізмом антидеренчання, який запобігає випадковим подвійним натисканням – критично важливий аспект у промисловій автоматизації, де надійність, точність та впевненість оператора визначають якість роботи та безперебійність процесу.

Навігація в системі реалізується через поєднання клавіш, які керують переміщенням оператора між різними станами меню. Клавіша «ОК» слугує основним засобом підтвердження вибору та переходу до наступних станів, тоді як клавіша «Вліво» забезпечує стабільний спосіб повернення до попередніх екранів або головного меню. Для запобігання випадковим багаторазовим спрацюванням застосовується механізм антидеренчання (прапорець «ignoreButt»), який гарантує, що кожне натискання буде зареєстроване лише один раз – важливий захід безпеки в умовах критично важливих операцій.

Код обирання кнопок наведено в лістингу 4.2.

Лістинг 4.2 – Код кнопки «Обрати» або «ОК»

```
private void SelectKey_Click(object sender, EventArgs e)
{
   ignoreButt = true;
   if ((currentState == AppState.MainMenu) && (ignoreButt))
   {  MainMenuSelect();
      ignoreButt = false;}
   if ((currentState == AppState.FabricSelectMenu) && (ignoreButt))
   {  HandleFabricSelection();
      ignoreButt = false;}
   if ((currentState == AppState.FabricSetup) && (ignoreButt))
   {     SaveFabricSettings(currentFabricSettings);
   selectedIndex = 0;
   currentState = AppState.MainMenu;
   DrawMainMenu();
   ignoreButt = false;
}
if ((currentState == AppState.ProgramSelectMenu) && (ignoreButt))
   {  SendProgram();
      ignoreButt = false;
   }
}
```

кінець лістингу 4.2

Метод SelectKey_Click обробляє натискання кнопки «Ок» і виконує відповідну дію залежно від поточного стану застосунку:

− ignoreButt = true: активує механізм антидребезгу, запобігаючи швидким повторним натисканням, які можуть спричинити небажані дії;

− логіка, прив’язана до конкретного стану.

Даний метод аналізує значення currentState і передає керування відповідному обробнику:

− MainMenuSelect(): виконує логіку, пов’язану з вибором пунктів головного меню;

− HandleFabricSelection(): обробляє введення користувача щодо вибору типу тканини або її параметрів;

− FabricSetup: виконує логіку налаштування та зміни параметрів існуючих тканин, що зберігаються у базі даних програми;

− ProgramSelection: забезпечує зчитування, відображення та вибір програм пошиву, розкрою та прасування.

Такий структурований, орієнтований на стани підхід гарантує, що кожен функціональний блок застосунку має власний, чітко відокремлений обробник. Це значно полегшує підтримку, покращує читабельність коду та забезпечує можливість подальшого масштабування системи. Строга ізоляція логіки за станами мінімізує ризик некоректної взаємодії між різними режимами роботи – критично важливий аспект для промислового програмного забезпечення, де надійність і передбачуваність безпосередньо впливають на якість виробництва та безпеку оператора.

Код керування зміною меню наведено в лістингу 4.3.

Лістинг 4.3 – Код керування зміною меню

```
private void MainMenuSelect()
{
   if (selectedIndex == 0)
   { LoadProgramList(); }

   if (selectedIndex == 1)
   { FabricSelectMenu(); }
```

```
        if (selectedIndex == 2)
        { RenderFabricSetupMenu(); }

        if (selectedIndex == 3)
        { Environment.Exit(0); }
    }
```

кінець лістингу 4.3

Метод MainMenuSelect визначає подальші дії залежно від того, який пункт меню наразі вибрано. Якщо користувач зупиняється на першому пункті, застосунок завантажує перелік доступних програм та переходить до режиму їхнього вибору. Вибір другого пункту переводить інтерфейс до меню вибору тканини. Третій пункт відкриває режим налаштування параметрів тканин, що зберігаються у базі даних програми. Якщо ж оператор обирає останній пункт меню, робота застосунку завершується й система коректно завершує свою роботу.

Такий підхід дає змогу однозначно пов’язати кожний варіант меню з відповідною дією, забезпечуючи чітку та логічно вибудовану послідовність переходів між режимами роботи. Окреме опрацювання кожного вибору робить програму структурованою, зрозумілою та простою у супроводі. Це особливо важливо для промислових систем, які можуть вимагати регулярного оновлення, адаптації чи розширення функціоналу без ризику порушення загальної логіки роботи.

Крім того, застосування механізму антидребезгу забезпечує контрольовану та послідовну обробку кожного натискання кнопки, запобігаючи випадковим повторним активаціям. Це підвищує стабільність роботи, зменшує кількість операторських помилок і підтримує безперервність робочого процесу на всіх етапах автоматизованого швейного виробництва.

Доповнюючи ці елементи керування, клавіші «Вгору» та «Вниз» забезпечують ефективне переміщення між пунктами меню в кожному стані. Дизайн передбачає кругову навігацію, що дозволяє операторам безперешкодно переходити від останнього пункту до першого і навпаки. Динамічне підсвічування підсилює зрозумілість, роблячи поточний вибір візуально помітним. У складніших станах, таких як FabricSetup, застосовується рефлексія, яка дозволяє системі динамічно адаптувати структуру меню до змін параметрів тканини, демонструючи її здатність масштабуватися та підлаштовуватися під виробничі варіації.

Метод UpKey_Click відповідає за переміщення виділення вгору по пунктах меню шляхом зменшення значення selectedIndex. Логіка роботи залежить від активного стану застосунку, що гарантує коректну поведінку навігації в кожному режимі. У головному меню, коли оператор намагається піднятися вище першого пункту, виділення автоматично переноситься на останній елемент, створюючи ефект циклічної навігації. Аналогічний принцип застосовується під час вибору програм, типів тканин або параметрів у режимі налаштування тканин: якщо курсор виходить за верхню межу списку, він переміщується на його останній елемент. Після зміни позиції метод викликає оновлення графічного виділення, щоб користувач завжди бачив актуально активний пункт.

Лістинг коду приведено у додатку Б.

Ключовою особливістю такої організації є безперервне пересування між елементами меню без упирання в межі списку. Це робить роботу оператора швидшою та інтуїтивнішою, оскільки він може безперервно переглядати пункти без потреби повертатися назад вручну. Візуальне підсвічування поточного вибору додатково знижує ризик помилок, забезпечуючи чітке розуміння того, який пункт буде активований.

Циклічна навігація значно підвищує зручність роботи, оскільки запобігає зупинкам у межах меню та дозволяє оператору зосередитися на виборі параметрів, а не на самому процесі навігації. Чітке оновлення виділення гарантує

надійність та передбачуваність системи, що є необхідною умовою для промислових середовищ, де швидкість і точність взаємодії безпосередньо впливають на продуктивність.

Метод DownKey_Click працює за тим самим принципом, що й UpKey_Click, але забезпечує переміщення виділення вниз по елементах меню через збільшення selectedIndex. У кожному стані застосунку логіка враховує кількість доступних пунктів. Якщо курсор виходить за нижню межу списку, він повертається до першого елемента, що також створює безперервний цикл навігації. Це стосується головного меню, списку програм, вибору тканин та параметрів їхнього налаштування. Після зміни індексу виконується оновлення підсвічування, щоб оператор завжди мав актуальний візуальний орієнтир.

Послідовність поведінки кнопок «Вгору» і «Вниз» забезпечує передбачувану та стабільну взаємодію з інтерфейсом. Такий підхід уніфікує навігацію незалежно від активного режиму, що суттєво полегшує роботу операторів, особливо в умовах швидкого виробничого циклу. Впевненість у тому, що система завжди реагує однаково, дозволяє мінімізувати кількість помилок та забезпечує високу ефективність роботи.

Разом ці стратегії обробки введення – ініціалізація, обробка натискань із антидребезгом, двонаправлена навігація та динамічне представлення меню – формують швидкий у реакції та зручний інтерфейс керування. Такий інтегрований підхід підвищує зручність використання, зменшує ймовірність помилок оператора та забезпечує стабільну роботу користувачів із різним рівнем технічної підготовки, що в підсумку сприяє ефективності та надійності автоматизованого швейного процесу.

4.2 Пояснення коду головного меню

Навігаційна структура головного меню застосунку організована у вигляді чотирьох окремих режимів взаємодії, кожен з яких відповідає певному етапу

роботи користувача, які були описані раніше (MainMenu; ProgramSelectMenu; FabricSelectMenu; FabricSetup).

Організація програмного забезпечення як послідовності чітко визначених режимів взаємодії забезпечує послідовний і передбачуваний досвід користувача. Такий підхід спрощує навігаційну логіку, знижує складність реалізації та підвищує зручність супроводу системи. Чіткість і надійність особливо важливі в промислових інтерфейсах «людина – машина», де ключовими є точність роботи та стабільність. Код створення змінних меню наведено в лістингу 4.4.

Лістинг 4.4 – Код створення змінних меню

```
public enum AppState
{
    MainMenu,
    ProgramSelectMenu,
    FabricSelectMenu,
    FabricSetup
}
```

кінець лістингу 4.4

Головний меню-інтерфейс надає користувачу чотири основні варіанти:

− вибір програми – дозволяє оператору перейти до інтерфейсу вибору програми;

− вибір тканини – надає доступ до інтерфейсу вибору тканини;

− налаштування тканини – відкриває інтерфейс для визначення параметрів, пов’язаних з конкретним видом тканини;

− вимкнення – завершує роботу застосунку або вимикає машину.

Початковий вибір встановлено на перший пункт, що забезпечує інтуїтивний початок навігації. Поточний вибраний пункт виділено для полегшення орієнтації оператора та швидкого визначення активного вибору.

Приклад зовнішнього вигляду програми наведено на рисунку 4.1.

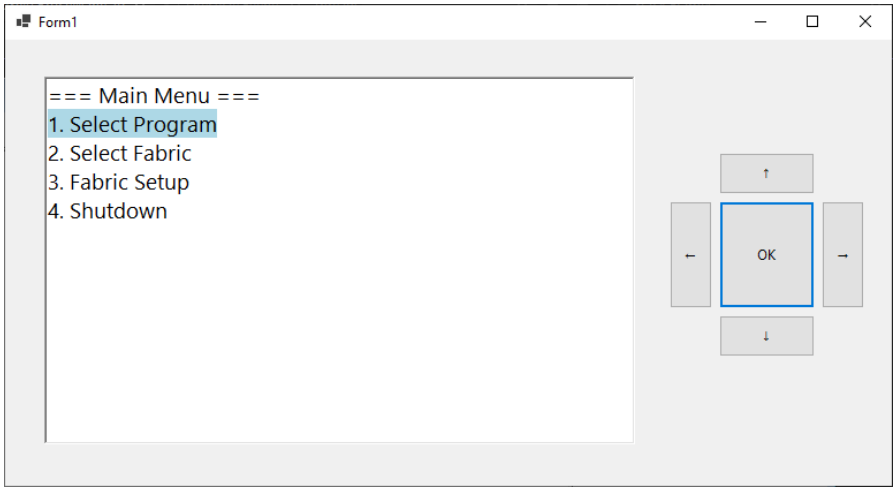

Рисунок 4.1 – Приклад зовнішнього вигляду програми

Чіткий та простий дизайн меню є критично важливим для операторів з обмеженою технічною підготовкою. Інтерфейс робить акцент на зручності використання та мінімалізмі, знижуючи когнітивне навантаження та ймовірність помилок при роботі. Гнучке форматування тексту та можливість динамічних оновлень підвищують адаптивність інтерфейсу, одночасно зберігаючи функціональну чіткість, що є особливо важливим у промислових середовищах «людина – машина».

Інтерфейс забезпечує візуальний зворотний зв’язок шляхом виділення поточного пункту меню. Ця функція підвищує обізнаність користувача щодо активного вибору та спрямовує навігацію серед доступних опцій.

Активний пункт меню виділяється контрастним кольором фону, що чітко відокремлює його від інших елементів. При цьому інтерфейс правильно розрізняє заголовки та пункти меню, гарантуючи виділення лише релевантних опцій. Механізм виділення працює динамічно при зміні вибору, забезпечуючи послідовність та оперативність взаємодії користувача.

Код підсвічування обраного рядка наведено в лістингу 4.5.

Лістинг 4.5 – Код підсвічування обраного рядка

```
private void HighlightSelectedLine(int itemCount)
{  string[] lines = richTextBox1.Lines;
   richTextBox1.Clear();
   for (int i = 0; i < itemCount + 1; i++)
   {
      if (i == selectedIndex + 1)
      {
         richTextBox1.SelectionBackColor = Color.LightBlue;
         richTextBox1.AppendText(lines[i] + "\n");
         richTextBox1.SelectionBackColor = Color.White;
      } else { richTextBox1.AppendText(lines[i] + "\n"); }
   }
}
```

кінець лістингу 4.5

Візуальні підказки є критично важливими для промислових інтерфейсів «людина – машина», оскільки вони підтримують швидку та точну роботу. Чітке позначення поточного пункту зменшує ймовірність помилок оператора та підвищує загальну зручність користування. Використання контрастних кольорів покращує видимість і доступність навіть за яскравого освітлення, характерного для виробничих приміщень.

4.3 Пояснення коду отримання данних з бази данних

Система використовує підхід, заснований на роботі з базою даних, для отримання переліку доступних типів тканин та їх відповідних технологічних параметрів. Такий підхід дає оператору змогу обирати попередньо визначені профілі тканин, що стандартизують операції шиття, розкрою та прасування.

Зберігання всіх параметрів, специфічних для певного виду тканини, у зовнішній базі даних забезпечує високу гнучкість системи та дозволяє легко адаптуватися до впровадження нових матеріалів або змін у виробничих вимогах.

Декларація змінних для отримання категорій тканин наведено в лістингу 4.6.

Лістинг 4.6 – Декларація змінних для отримання категорій тканин

```
List<string> fabricTypes = new List<string>();
private FabricSettings currentFabricSettings = null;
```

кінець лістингу 4.6

Після отримання переліку підтримуваних категорій тканин система відображає їх у інтерфейсі для вибору оператором. Після вибору конкретного виду тканини всі відповідні технологічні параметри – зокрема щільність стібків, натяг нитки або температура прасування – автоматично завантажуються в систему. Така організація забезпечує модульну та розширювану структуру керування конфігураціями тканин, завдяки якій будь-які зміни в базі даних негайно відображаються в застосунку без необхідності втручання у програмний код.

Система отримує повний перелік доступних типів тканин із зовнішньої бази даних, що гарантує актуальність інформації, з якою працює оператор. Перед виконанням запиту інтерфейс скидає попередні дані та готує оновлений список для відображення користувачу.

Код запиту даних з бази даних наведено в лістингу 4.7.

Лістинг 4.7 – Код запиту даних з бази даних

```
fabricTypes.Clear();
selectedIndex = 0;
currentState = AppState.FabricSelectMenu;
```

```
string connString = "Server=localhost; Database=fabricdb; Uid=root; Pwd=root;";
using (MySqlConnection conn = new MySqlConnection(connString))
{   conn.Open();
    string query = "SELECT FabricType FROM fabric;";
    MySqlCommand cmd = new MySqlCommand(query, conn);
    MySqlDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
      { fabricTypes.Add(reader["FabricType"].ToString()); }      }
```

кінець лістингу 4.7

Після виконання запиту з бази даних у систему надходять всі зареєстровані види тканин, які потім відображаються в інтерфейсі вибору. Такий підхід забезпечує точне відображення змін у центральному сховищі даних, включаючи додавання нових категорій тканин або коригування існуючих.

Використання централізованої реляційної бази даних забезпечує узгоджене та надійне керування інформацією у виробничому середовищі. Будь-які оновлення – наприклад, введення нових матеріалів – негайно стають доступними в застосунку без необхідності змінювати програмний код. Крім того, контрольоване управління підключеннями до бази даних сприяє стабільності системи, запобігаючи проблемам, пов’язаним із неефективним використанням ресурсів.

Інтерфейс вибору тканини автоматично формує список доступних категорій матеріалів на основі даних, отриманих із бази даних. Перед відображенням меню система завантажує актуальну інформацію, що гарантує відповідність інтерфейсу поточному набору підтримуваних тканин.

Код створення меню вибору тканини наведено в лістингу 4.8.

Лістинг 4.8 – Код створення меню вибору тканини

```
private void FabricSelectMenu()
{  ReadFabric();
   DrawFabricMenu();
   HighlightSelectedLine(fabricTypes.Count);
}

private void DrawFabricMenu()
{  richTextBox1.Clear();
   richTextBox1.AppendText("=== Fabric Type Menu ===\n");
   for (int i = 0; i < fabricTypes.Count; i++)
   { richTextBox1.AppendText($"{i + 1}. {fabricTypes[i]}\n"); }
}
```

кінець лістингу 4.8

Меню створюється динамічно. На екрані виводиться чіткий заголовок, який позначає призначення цього розділу, після чого відображається пронумерований список типів тканин. Нумерація підвищує читабельність та прискорює навігацію. Після формування списку активний пункт виділяється візуально, що сприяє зосередженню оператора та забезпечує правильність вибору.

Приклад зовнішнього вигляду програми у меню вибору тканини наведено на рисунку 4.2.

Динамічне формування меню на основі актуальних даних із централізованої бази забезпечує постійну відповідність інтерфейсу виробничим оновленням та змінам у наявних матеріалах. Структуроване та пронумероване подання інформації покращує зручність користування, а візуальне виділення активного пункту знижує ризик помилок під час роботи. Такий підхід сприяє чіткості, адаптивності та ефективній взаємодії оператора з інтерфейсом.

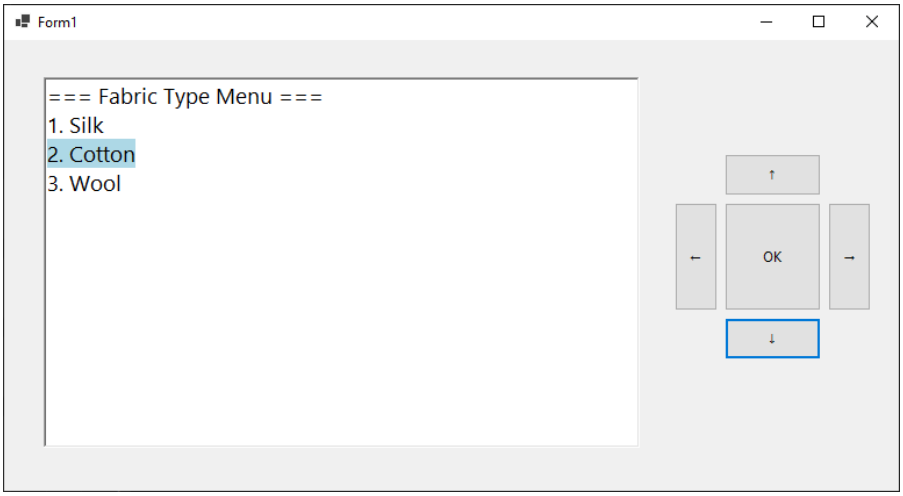

Рисунок 4.2 – Приклад зовнішнього вигляду програми у меню вибору тканини

Після того як оператор обирає тип тканини зі списку, система отримує відповідні технологічні параметри з центральної бази даних і забезпечує їх доступність для подальших етапів обробки. Після завантаження даних інтерфейс відображає підтвердження вибору, що дає змогу оператору чітко усвідомити, яка саме тканина та які параметри були застосовані.

Після цього інтерфейс повертається до основного екрана навігації, скидаючи попередній стан вибору. Такий підхід забезпечує зрозумілий і послідовний робочий процес, зменшуючи когнітивне навантаження та підтримуючи логічну структуру взаємодії.

Код завантаження характеристик тканини з бази даних наведено в лістингу 4.9.

Лістинг 4.9 – Код завантаження характеристик тканини з бази даних

```
private void HandleFabricSelection()
{
    string chosenFabric = fabricTypes[selectedIndex];
        currentFabricSettings = LoadFabricSettingsFromDB(chosenFabric);
    if (currentFabricSettings != null)
    {
```

```
                MessageBox.Show("Fabric selected:\n" +
 currentFabricSettings.FabricType);
            }
            // Return to Main Menu
            currentState = AppState.MainMenu;
            selectedIndex = 0;
            DrawMainMenu();
        }
```

кінець лістингу 4.9

Такий порядок взаємодії сприяє інтуїтивному та прозорому процесу вибору. Миттєве підтвердження підвищує впевненість оператора, а модульна структура дозволяє легко інтегрувати додаткові параметри або етапи перевірки без порушення загального алгоритму роботи. Повернення до основного екрана після кожного вибору відповідає усталеним практикам проєктування промислових інтерфейсів, забезпечуючи передбачуваність та мінімізуючи ризик помилок.

Описаний підхід забезпечує надійний та орієнтований на користувача механізм вибору типів тканин і доступу до їх попередньо визначених технологічних параметрів. Динамічне отримання даних із централізованого сховища, поєднане з чітким візуальним зворотним зв’язком та структурованою навігацією, сприяє ефективному налаштуванню виробничих режимів для різних матеріалів. Такий дизайн підвищує масштабованість, підтримуваність і узгодженість роботи системи, що робить його придатним для автоматизованих текстильних виробництв, де гнучкість і надійність є критично важливими. Оптимізуючи процес вибору профілів тканин, система робить суттєвий внесок у стандартизацію та вдосконалення промислових робочих процесів.

Приклад інтерфейсу додатку приведено у додатку В.

4.4 Пояснення коду налаштування параметрів

Клас FabricSettings є центральною моделлю даних, яка інкапсулює повний набір технологічних параметрів, необхідних для автоматизованого управління процесами пошиття, різання та прасування різноманітних типів тканин. Ця модель виступає структурованим представленням усіх ключових характеристик, що визначають режими роботи обладнання, забезпечуючи точність та відтворюваність технологічних операцій.

Клас FabricSettings об’єднує всі ключові параметри, необхідні для налаштування технологічних режимів під час роботи з різними видами тканин. Він містить унікальний ідентифікатор Id, що дозволяє однозначно визначити кожен запис у системі, а також текстову назву FabricType, яка слугує зручним засобом ідентифікації та пошуку потрібного матеріалу.

Код центральної моделі даних FabricSettings наведено в лістингу 4.10.

Лістинг 4.10 – Код центральної моделі даних FabricSettings

```
public class FabricSettings
{  public int Id { get; set; }
   public string FabricType { get; set; }
   public double StitchDensity { get; set; }
   public double ThreadTension { get; set; }
   public double PressTemp { get; set; }
   public double SteamPressure { get; set; }
   public double IroningTime { get; set; }
   public double LaserPower { get; set; }
   public double CuttingSpeed { get; set; }
   public double FocusPosition { get; set; }
   public double GasPressure { get; set; }
```

```
    public double PulseFrequency { get; set; }
    public double BeamDiameter { get; set; }
}
```

кінець лістингу 4.10

Окрім загальної інформації, клас описує детальні експлуатаційні характеристики, які впливають на режими пошиття, прасування та лазерного різання. До них належать параметри швейного процесу, зокрема StitchDensity, що визначає кількість стібків на одиницю довжини та безпосередньо впливає на міцність і якість шва, а також ThreadTension, який задає натяг нитки в ньютонах і забезпечує рівномірність формування стібка. Для прасування передбачені такі параметри, як PressTemp, що відповідає за температуру обробки та підбирається відповідно до властивостей тканини, SteamPressure, який визначає силу подачі пари, та IroningTime, що регулює тривалість термічного впливу.

Додатково клас охоплює низку характеристик лазерного різання. До них належить LaserPower, що визначає енергетичну інтенсивність променя, CuttingSpeed, яка регулює швидкість руху інструмента й впливає на якість формування краю, а також FocusPosition, що задає положення фокусної точки для максимальної точності. Параметри GasPressure і PulseFrequency забезпечують оптимальні умови обробки шляхом контролю тиску допоміжного газу та частоти імпульсів лазера, тоді як BeamDiameter визначає ширину променя й відповідно впливає на геометрію різу.

Модель легко розширюється новими характеристиками, що дозволяє адаптувати систему до змін технологічних вимог без значних модифікацій існуючого коду, а об’єднання параметрів у єдину структуру забезпечує консистентність та контроль типів, усуваючи помилки, пов’язані з невірними форматами даних. Використання класу спрощує управління даними, уникнення дублювання логіки та забезпечення єдиного джерела істини для параметрів тканин.

У промисловій автоматизації централізоване управління параметрами є критично важливим, оскільки гарантує однорідність та відтворюваність технологічних режимів. Це, у свою чергу, забезпечує стабільну якість операцій при роботі з різними типами тканин.

Наявність такої моделі дозволяє стандартизувати та оптимізувати виробничі процеси, забезпечуючи точність налаштувань для кожного типу тканини, зменшення ймовірності помилок через централізоване керування параметрами та легкість інтеграції з іншими компонентами системи, такими як бази даних або інтерфейси користувача.

Метод LoadFabricSettingsFromDB призначений для динамічного отримання технологічних параметрів конкретного типу тканини з реляційної бази даних MySQL. Цей метод є ключовим компонентом системи автоматизованого управління процесами обробки тканин, оскільки забезпечує точне та безпечне зчитування конфігураційних даних безпосередньо з централізованого сховища.

Метод починає свою роботу з відкриття з’єднання з базою даних за допомогою рядка підключення, який містить необхідні параметри (адресу сервера, назву бази даних, облікові дані) – лістинг 4.11.

Лістинг 4.11 – Код параметризованого SQL-запиту

```
public class FabricSettings
string query = "SELECT * FROM fabric WHERE FabricType = @name
LIMIT 1;";
```

кінець лістингу 4.11

Використання параметризованого запиту з параметром @name забезпечує захист від SQL-ін’єкцій, що є критично важливим для безпеки промислових систем. Запит обмежується одним записом (LIMIT 1), що підвищує ефективність та швидкість виконання.

За допомогою об’єкта MySqlDataReader метод зчитує результати запиту. Якщо запис існує, кожне поле результату перетворюється у відповідний тип даних (наприклад, Convert.ToDouble для числових параметрів) та присвоюється властивостям нового об’єкта FabricSettings – лістинг 4.12.

Лістинг 4.12 – Код для зчитування та обробки результатів

```
return new FabricSettings()
{
   Id = Convert.ToInt32(reader["Id"]),
   FabricType = reader["FabricType"].ToString(),
   StitchDensity = Convert.ToDouble(reader["StitchDensity"]),
   ThreadTension = Convert.ToDouble(reader["ThreadTension"]),
   ...
};
```

кінець лістингу 4.12

Якщо запис знайдено, метод повертає заповнений об’єкт FabricSettings, який інкапсулює усі необхідні технологічні параметри. У разі відсутності запису метод повертає null, що дозволяє програмному забезпеченню коректно обробити ситуацію відсутності даних.

Використання параметризованих запитів та конструкції using для управління ресурсами (з’єднанням та читачем) запобігає витокам пам’яті та забезпечує захист від SQL-ін’єкцій, що є критично важливим для промислових систем.

Метод дозволяє динамічно отримувати актуальні параметри безпосередньо з бази даних, що робить систему адаптивною до змін у виробничих умовах. Це особливо важливо в середовищах, де технологічні вимоги можуть змінюватися з часом.

Отримання параметрів безпосередньо з бази даних забезпечує однорідність та актуальність даних у всіх частинах системи. Це гарантує, що всі операції виконуються з одними й тими самими налаштуваннями, що критично важливо для забезпечення стабільної якості виробництва.

У контексті промислової автоматизації метод LoadFabricSettingsFromDB відіграє визначальну роль у забезпеченні точного та ефективного управління технологічними процесами. Він дозволяє:

− автоматично отримувати актуальні параметри для кожного типу тканини, що усуває необхідність ручного введення даних та зменшує ймовірність помилок;

− забезпечувати консистентність технологічних режимів у різних виробничих сценаріях;

− підтримувати довготривалу стабільність системи завдяки надійному управлінню ресурсами та захисту від потенційних вразливостей.

Метод SaveFabricSettings реалізує механізм оновлення технологічних параметрів для конкретного типу тканини в реляційній базі даних MySQL. Цей метод є критично важливим для підтримки актуальності та точності налаштувань у системах автоматизованого управління процесами обробки тканин, оскільки дозволяє динамічно коригувати параметри без необхідності зміни програмного коду.

Метод починає з відкриття з’єднання з базою даних за допомогою рядка підключення, який містить необхідні параметри (адресу сервера, назву бази даних, облікові дані) – лістинг 4.13.

Лістинг 4.13 – Формування параметризованого SQL-запиту

```
string sql = @"
    UPDATE fabric
    SET
        StitchDensity = @StitchDensity,
```

```
        ThreadTension = @ThreadTension,
        PressTemp = @PressTemp,
    SteamPressure = @SteamPressure,
        IroningTime = @IroningTime,
        LaserPower = @LaserPower,
        CuttingSpeed = @CuttingSpeed,
        FocusPosition = @FocusPosition,
        GasPressure = @GasPressure,
        PulseFrequency = @PulseFrequency,
        BeamDiameter = @BeamDiameter
      WHERE FabricType = @FabricType;
    ";
```

кінець лістингу 4.13

Використання параметризованого SQL-запиту UPDATE забезпечує безпечне та ефективне оновлення даних. Кожне поле таблиці fabric оновлюється відповідно до значень, переданих через параметри, що запобігає SQL-ін’єкціям та забезпечує коректність типів даних.

Метод прив’язує властивості об’єкта FabricSettings до параметрів SQL-запиту. Після прив’язки параметрів виконується запит за допомогою методу ExecuteNonQuery(), який оновлює відповідні записи в базі даних (лістинг 4.14).

Лістинг 4.14 – Прив’язка параметрів та виконання запиту

```
cmd.Parameters.AddWithValue("@FabricType", settings.FabricType);
cmd.Parameters.AddWithValue("@StitchDensity", settings.StitchDensity);
cmd.Parameters.AddWithValue("@ThreadTension", settings.ThreadTension)
```

кінець лістингу 4.14

Використання конструкції using для об’єктів з’єднання та команди гарантує автоматичне вивільнення ресурсів, що запобігає витокам пам’яті та підвищує надійність системи.

Використання параметризованих запитів забезпечує захист від SQL-ін’єкцій, що є критично важливим для промислових систем, де безпека даних має першорядне значення. Метод дозволяє оперативно оновлювати технологічні параметри безпосередньо в базі даних. Це робить систему адаптивною до змін у виробничих умовах, таких як зміна властивостей матеріалів або впровадження нових технологічних процесів. Оновлення параметрів безпосередньо в базі даних забезпечує консистентність та актуальність даних у всіх частинах системи. Це дозволяє підтримувати однорідність технологічних режимів та уникнути розбіжностей у налаштуваннях.

У контексті промислової автоматизації метод SaveFabricSettings відіграє визначальну роль у забезпеченні оперативного та точного управління технологічними процесами. Він дозволяє швидко адаптуватися до змін у виробничих вимогах, таких як зміна параметрів прасування, лазерного різання або швейних режимів, підтримувати високий рівень точності налаштувань обладнання, що є критично важливим для забезпечення стабільної якості продукції, зменшувати час простою завдяки можливості оперативного оновлення параметрів без зупинки виробничих процесів.

4.5 Вибірка та управління швейними програмами

Модуль вибору швейних програм є ключовим компонентом автоматизованої системи керування процесами пошиття. Він забезпечує інтерактивний інтерфейс для операторів, дозволяючи вибирати та завантажувати програми з локального сховища, а також ініціювати їх передачу до швейного обладнання. Цей модуль реалізує динамічне зчитування файлів програм, їх

візуалізацію та механізм передачі, що є критично важливим для гнучкості та адаптивності виробничого процесу.

Метод LoadProgramList відповідає за підготовку та завантаження переліку доступних швейних програм із локального сховища. Перед початком роботи він очищує наявний список програм, щоб уникнути дублювання даних, після чого встановлює початковий індекс вибору та переводить інтерфейс у режим перегляду програм. Далі метод перевіряє наявність відповідної директорії: якщо папка існує, програма зчитує всі файли, що в ній містяться, і додає їхні назви до списку; якщо ж директорію не знайдено, замість фактичних даних у перелік вноситься відповідне повідомлення.

Після завантаження інформації викликаються методи відображення та підсвітки активного елементу, що забезпечує користувачу чіткий і зручний інтерфейс навігації. Такий підхід дозволяє системі оперативно реагувати на зміни у файловій структурі та підтримує інтуїтивну взаємодію оператора з модулем вибору програм.

Використання динамічного зчитування файлів з локального сховища дозволяє гнучко керувати набором доступних програм без необхідності зміни програмного коду. Це особливо важливо в промислових умовах, де асортимент програм може часто оновлюватися. Перевірка існування папки та обробка виняткових ситуацій підвищують надійність системи, запобігаючи помилкам під час роботи – лістинг 4.15.

Лістинг 4.15 – Візуалізація списку програм

```
private void PrintFile()
{
   richTextBox1.Clear();
   richTextBox1.AppendText("=== Меню вибору програми ===\n");
   for (int i = 0; i < programList.Count; i++)
   {
```

```
            richTextBox1.AppendText($"{i + 1}. {programList[i]}\n");
        }
    }
```

кінець лістингу 4.15

Метод PrintFile відповідає за візуалізацію списку доступних програм у текстовому полі richTextBox1. Він виконує такі дії:

− очищення текстового поля (richTextBox1.Clear()) для видалення попереднього вмісту;

− циклічний перебір та відображення усіх програм зі списку programList з нумерацією, що спрощує вибір.

Чітка візуалізація списку програм з нумерацією елементів покращує зручність використання та зменшує ймовірність помилок при виборі програми. Використання richTextBox1 дозволяє гнучко форматувати текст та підтримувати різні стилі відображення, що важливо для індустріальних інтерфейсів.

Метод SendProgram відповідає за повний цикл передавання вибраної програми до швейного обладнання. Спочатку він ініціює з’єднання з апаратною частиною системи через MachineConnect(), створюючи необхідний канал комунікації. Після встановлення зв’язку програма надсилає параметри вибраної тканини за допомогою методу SendFabricSettings(), а потім передає інструкції для виконання швейного процесу через SendProg(). Завершується процедура закриттям з’єднання (CloseConnnect()), що забезпечує коректне вивільнення ресурсів і стабільність подальшої роботи системи.

Такий послідовний підхід гарантує цілісність і надійність передавання даних до обладнання, мінімізує ризик помилок та конфліктів під час виконання операцій. Розподіл процесу на окремі методи підвищує модульність і спрощує підтримку програмного коду. Це особливо важливо в промислових системах, де вимоги до надійності, гнучкості та передбачуваності роботи є критично високими. Весь лістинг коду приведено у додатку Г.

Висновок до розділу 4

Проєктування користувацького інтерфейсу для автоматизованої системи керування промисловим обладнанням вимагало комплексного підходу, спрямованого на забезпечення надійності, зручності та безпеки роботи оператора. У цьому розділі було реалізовано низку ключових рішень, які суттєво впливають на ефективність і стабільність виробничих процесів.

Використання станів (AppState) та механізму антидеренчання (ignoreButt) гарантує стабільну обробку введення, що критично важливо для промислових систем. Циклічна навігація та візуальне підсвічування активного пункту знижують когнітивне навантаження на оператора та мінімізують ризик помилок.

Застосування орієнтованого на стани підходу (MainMenu, FabricSelectMenu, FabricSetup тощо) дозволило чітко розділити логіку роботи системи, спростити підтримку та масштабування. Кожен стан має власний обробник, що забезпечує ізоляцію функціональних блоків і знижує ризик конфліктів між режимами роботи.

Динамічне формування меню на основі даних з централізованої бази даних (FabricSettings) забезпечує гнучкість та адаптивність системи до змін у виробничих умовах, таких як додавання нових типів тканин або оновлення технологічних параметрів.

Використання параметризованих SQL-запитів забезпечує захист від SQL-ін’єкцій та ефективне управління ресурсами. Централізоване зберігання параметрів у базі даних дозволяє підтримувати актуальність і консистентність даних у всіх частинах системи, що критично важливо для забезпечення стабільної якості виробництва.

Чітке візуальне підсвічування активних пунктів меню, мінімалістичний дизайн та інтуїтивна навігація знижують ймовірність помилок і підвищують ефективність роботи.

Механізми антидеренчання та контрольовані переходи між станами забезпечують стабільну роботу системи навіть у умовах високого навантаження, що особливо важливо для промислових середовищ.

Запропонована архітектура дозволяє легко інтегрувати нові функції та адаптувати систему до змін у технологічних вимогах. Наприклад, додавання нових типів тканин або оновлення параметрів обробки може здійснюватися без значних модифікацій програмного коду.

Використання централізованої бази даних відкриває можливості для впровадження аналітики та моніторингу виробничих процесів у реальному часі, що може стати основою для подальшої оптимізації та автоматизації.

Реалізований користувацький інтерфейс є надійним, зручним та адаптивним інструментом для управління промисловими процесами. Він поєднує в собі простоту використання, високу функціональність та стійкість до помилок, що робить його ефективним рішенням для автоматизації швейного виробництва. Дальший розвиток системи може включати впровадження додаткових механізмів аналітики, розширення функціоналу для роботи з новими типами матеріалів, а також інтеграцію з іншими автоматизованими системами підприємства.

ВИСНОВКИ

Проаналізовано основні етапи технологічного процесу виготовлення сорочок, зокрема розкрій тканини, шиття та прасування. Визначені ключові параметри технологічного процесу, які необхідно дотримуватися для забезпечення якості та ефективності виробництва. Поставлено задачі автоматизації, а саме: контроль і регулювання параметрів розкрою, шиття та прасування, автоматичне налаштування обладнання залежно від типу тканини, а також дистанційне керування роботою швейних машин і прасувальних пресів.

Сучасні методи автоматизації значно підвищують ефективність окремих етапів пошиву сорочок, але повна автономізація виробництва залишається недосяжною через технічні обмеження та складність роботи з текстильними матеріалами. Перспективи розвитку пов’язані з інтеграцією AI, м’якої робототехніки та адаптивних систем керування, що дозволить подолати існуючі виклики та забезпечити безперервний, високоякісний виробничий цикл.

Обґрунтовано вибір мови програмування C# та фреймворку Windows Forms для розробки програмного забезпечення автоматизованої системи керування. C# забезпечує високу продуктивність, надійність та зручність розробки завдяки глибокій інтеграції з екосистемою Windows, строгій типізації та багатій бібліотечній підтримці. Windows Forms дозволяє створювати інтуїтивні інтерфейси користувача, що є критично важливим для систем, де зручність та швидкість взаємодії безпосередньо впливають на ефективність виробництва. Комбінація C# і Windows Forms забезпечує потужне, стабільне та ефективне рішення для розроблення інтерфейсів і програмного забезпечення нагляду в автоматизованих системах. Здатність цих технологій до безшовної інтеграції з обладнанням, підтримка моніторингу в реальному часі та зручність середовища розробки роблять їх оптимальним вибором для інженерних команд.

Розроблено структуру бази даних для зберігання параметрів тканин, таких як густина стібка, натяг нитки, температура прасування, тиск пари, потужність

лазера, швидкість різання та інші. Запропонована таблиця бази даних є фундаментальним компонентом промислової системи обробки тканин, що забезпечує стандартизоване, ефективне та високоякісне виробництво. Централізоване керування параметрами забезпечує стандартизацію технологічних режимів, зменшення помилок та підвищення якості продукції. Такий структурований підхід гарантує, що машини працюють стабільно та оптимально незалежно від типу тканини чи обсягів виробництва. Реалізовано механізми динамічного зчитування та оновлення параметрів, що дозволяє оперативно адаптувати систему до змін у виробничих умовах.

Створено програмне забезпечення з інтуїтивно зрозумілим інтерфейсом, який забезпечує:

− вибір та завантаження програм керування швейним обладнанням;

− вибір типу тканини з централізованої бази даних;

− налаштування параметрів тканин для оптимізації технологічних режимів;

− передачу параметрів та програм до швейного обладнання.

Програмна реалізація передбачає використання механізмів антидеренчання для запобігання випадковим повторним натисканням, а також циклічну навігацію між пунктами меню, що підвищує зручність роботи.

Запропонована архітектура дозволяє легко інтегрувати нові функції та адаптувати систему до змін у технологічних вимогах. Наприклад, додавання нових типів тканин або оновлення параметрів обробки може здійснюватися без значних модифікацій програмного коду.

Використання централізованої бази даних відкриває можливості для впровадження аналітики та моніторингу виробничих процесів у реальному часі, що може стати основою для подальшої оптимізації та автоматизації.

Реалізований користувацький інтерфейс є надійним, зручним та адаптивним інструментом для управління промисловими процесами. Він поєднує в собі простоту використання, високу функціональність та стійкість до

помилок, що робить його ефективним рішенням для автоматизації швейного виробництва.

Практична реалізація системи автоматизації дозволить оптимізувати послідовність технологічних операцій, зменшити трудомісткість підготовки та виготовлення виробу, а також підвищити якість готової продукції. Запропоноване рішення може бути впроваджене у виробничі процеси швейних підприємств для підвищення їхньої ефективності та конкурентоспроможності.

Таким чином, у роботі розроблено автоматизовану систему керування технологічним процесом виробництва сорочок, яка базується на використанні мови програмування C# та бази даних для оптимізації технологічних операцій. Система забезпечує гнучкість, надійність та зручність керування, що робить її ефективним інструментом для сучасного швейного виробництва.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Revolutionising Garment Manufacturing. URL: https://www.fibre2fashion.com/industry-article/9904/revolutionising-garment-manufacturing-embracing-industry-5-0-s-human-tech-collaboration/ (дата звернення 18.08.2025).

2. Future Clothing Manufacturing Trends. URL: https://makersrow.com/blog/future-clothing-manufacturing-trends/ (дата звернення 16.08.2025).

3. Gerber Paragon Cutting System. URL: https://www.gerbertechnology.com/aerospace/ply-cutting/paragon-cutting-system/ (дата звернення 22.08.2025).

4. Market brochure furniture. Brochure Vectorfurniture. URL: https://www.scribd.com/document/646481656/brochure-vectorfurniture-en (PDF). (дата звернення 23.08.2025).

5. Direct to Fabric Printers. URL: https://www.kornit.com/direct-to-fabric/ (дата звернення 25.08.2025).

6. Sewbots. URL: https://softwearautomation.com/sewbots/ (дата звернення 27.09.2025).

7. Swebo Inc. URL: https://www.sewbo.com/ (дата звернення 21.08.2025).

8. Brother Corporate Vision. URL: https://global.brother/pub/com/en/corporate/pdf/2025/brother-profile-en.pdf (дата звернення 21.08.2025).

9. Veit-Group Catalog. URL: http://www.v-s-i.it/Depliants/PDF/IMBNews.pdf (PDF). (дата звернення 21.08.2025).

10. MEYER Maschinenbau. URL: https://www.meyer-maschinenbau.de/ (дата звернення 24.08.2025).

11. Troelsen A., Japikse P. Pro C# 10 with .NET 6. Apress, URL: https://dl.ebooksworld.ir/books/Pro.CSharp.10.with.NET.6.Andrew.Troelsen.Phil.Japikse.Apress.9781484278680.EBooksWorld.ir.pdf/ (дата звернення 25.10.2025).

12. Albahari J. C# 10 in a Nutshell. O'Reilly. URL: https://dl.ebooksworld.ir/books/CSharp.10.in.a.Nutshell.Joseph.Albahari.OReilly.9781098121952.EBooksWorld.ir.pdf/ (PDF). (дата звернення 08.10.2025).

13. Windows Security Documentation. URL: https://learn.microsoft.com/en-us/windows/security/ (дата звернення 08.11.2025).

14. Gilchrist A. Industry 4.0: The Industrial Internet of Things. Apress. URL: https://download.e-bookshelf.de/download/0007/6832/86/L-G-0007683286-0014731014.pdf/ (PDF). (дата звернення 10.11.2025).

15. Montgomery D. C. Introduction to Statistical Quality Control. Wiley. URL: https://sirangcoop.com/wp-content/uploads/2023/10/Douglas_C_Montgomery_Introduction_to_statistical_quality_control-2.pdf/ (PDF). (дата звернення 21.10.2025).

16. Голіков Т. Ю. Автоматизована система керування технологічним процесом пошиву сорочок. *Actual problems of automation and control:* матеріали XIII Міжнародної науково-практичної інтернет-конференції молодих учених та студентів. Випуск № 13. Луцьк: ЛНТУ, 2024. С. 42-45.