НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет робототехніки та приладобудування

Кафедра комп'ютерно-інтегрованих технологій виробництва приладів

До захисту допущено:

Завідувач кафедри

_____ Наталія СТЕЛЬМАХ

«12» 06 2026 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерно-інтегровані системи та технології в приладобудуванні»

спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»

на тему: «Автоматизована система сортування посилок»

Виконав:
студент IV курсу, групи ПБ-21
Таран Єгор Олександрович _____

Керівник:
Доктор технічних наук, професор,
Безуглий Михайло Олександрович _____

Рецензент: Ігор Кравченко
Ст. викладач
кафедри КІОНС _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2026 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет робототехніки та приладобудування
Кафедра комп'ютерно-інтегрованих технологій виробництва приладів

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютерно-інтегровані системи та технології в приладобудуванні»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Наталія СТЕЛЬМАХ

«12» 06 2026 р.

ЗАВДАННЯ

на дипломну роботу студенту

Таран Єгор Олександрович

1. Тема роботи «Автоматизована система сортування посилок», керівник роботи Безуглий Михайло Олександрович, доктор технічних наук, професор, затверджені наказом по університету від «26» 05 2026 р. № 1905-С

2. Термін подання студентом роботи 01.06.2026

3. Вихідні дані до роботи : сумісність з операційними системами Windows, автоматизоване сортування посилок.

4. Зміст роботи Зміст пояснювальної записки : Перелік умовних позначень. Вступ. Розділ І. Основні принципи розробки автоматизованих майданчиків для паркування автомобілів: 1.1.Вимоги до організації місць паркування автомобілів; 1.2.Класифікація майданчиків для паркування автомобілів; 1.3.Аналіз сучасних автоматизованих систем контролю паркування; 1.4.Постановка задачі; Розділ ІІ. Проектування автоматичного в'їзного (виїзного) терміналу: 2.1.Візуалізація майданчику для паркування автомобілів; 2.2.Розробка структурно-функціональної схеми автоматизованого контролю перебування та оплати на автомобільних майданчиках для паркування; 2.3.Вибір основних елементів; 2.4.Проектування конструкційних модулів та елементів в'їзного (виїзного) терміналу; Розділ ІІІ.

Розробка програмного забезпечення системи контролю та оплати: 3.1. Вибір та обґрунтування мови програмування та основних бібліотек; 3.2. Розробка архітектури програмного забезпечення; 3.3. Розробка програмного забезпечення для персоналу; 3.4. Розробка програмного забезпечення для в'їзного терміналу; 3.5. Розробка програмного забезпечення для виїзного терміналу; 3.6. Розробка програмного забезпечення лічильника автомобілів; 3.7. Розробка рекомендацій для користувача. Висновки. Перелік використаних літературних джерел

5. Перелік ілюстративного матеріалу : *Структурно-функціональна схема автоматизованої системи сортування посилок (А3), Електрична принципова схема (А3), Складальне креслення (А1), Алгоритм та інтерфейс програмного забезпечення (А3). Алгоритм керування контролером (А4), Алгоритм роботи автомат і сортування посилок (А3).*

6. Дата видачі завдання 17.04.2026 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд та аналіз літературних джерел. Постановка задачі.	17.04.2026	Виконано
2	Розробка електричної схеми, структурно-функціональної схеми	24.04.2026	Виконано
3	Проектування модулів та елементів.	01.05.2026	Виконано
4	Розробка програмного забезпечення.	22.05.2026	Виконано
5	Розробка рекомендацій для користувача	26.05.2026	Виконано
6	Оформлення пояснювальної записки та графічного матеріалу	01.06.2026	Виконано

Студент Таран Єгор Олександрович

Керівн Безуглий Михайло Олександрович

Пояснювальна записка

до дипломного проєкту

на тему: **Автоматизована система сортування посилок**

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 6

Вступ 7

Анотація 8

Anotation 9

РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ АВТОМАТИЗОВАНИХ СИСТЕМ СОРТУВАННЯ ПОСИЛОК 10

1.1. Актуальність сортування посилок 10

1.2. Класифікація систем сортування посилок 10

1.3. Огляд сучасних системи сортування 12

1.4 Особливості застосування маніпулятора в автоматизованих системах сортування посилок 17

1.5. Постанова задачі 20

РОЗДІЛ 2. ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ СОРТУВАННЯ ПОСИЛОК 21

2.1. Розробка структурно-функціональної схеми автоматизованої системи сортування посилок 21

2.1.1. Принцип роботи відповідно до ФСА 21

2.1.2. Вибір компонентів АССП 22

2.2. Розробка електричної схеми конвеєру 25

2.2.1. Принцип роботи відповідно до електричної схеми 26

2.2.2. Вибір компонентів конвеєру відповідно до розрахунків 26

2.2.3. Розробка програми керування контролером 30

2.3.1. Проєктування загального вигляду конструкції 32

2.3.2. Проектування основних вузлів 33

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КЕРУВАННЯМ ПРОЦЕСОМ СОРТУВАННЯ 36

3.1 Вибір та обґрунтування мови програмування та основних бібліотек 36

3.2. Розробка алгоритму програмного забезпечення 38

3.4. Розробка Backend програмного забезпечення 42

3.5. Розробка рекомендацій для користувача 48
Висновки 50
Список джерел 51
Додаток А 54
Додаток Б 64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АССП – Автоматизована система сортування посилок

CAD – Computer-Aided Design

GUI – Graphical User Interface

ПК – Персональний комп’ютер

ФСС – функціонально-структурна схема

Вступ

Сортування посилок на сьогоднішній день є однією з найважливіших проблем на ринку. Швидкість та точність доставки є основними чинниками конкурентоспроможності будь-якої логістичної компанії.

Однією з головних переваг автоматизованих систем є їхня ефективність. Завдяки автоматизації процесів сортування, посилки обробляються автоматично, що зменшує людський фактор. Це дозволяє зменшити час очікування посилок. Традиційні методи ручного оброблення та сортування вантажів уже не спроможні забезпечити необхідну пропускну здатність, оскільки обмежені людським фактором, що неминуче призводить до критичних затримок у ланцюгах постачання, збільшення кількості помилок під час адресації та зростання собівартості оброблення кожної одиниці товару.

Впровадження автоматизованих конвеєрних систем сортування є ключовим інструментом для подолання цих викликів. Автоматизація дозволяє перевести логістичні процеси у формат безперервного циклу, мінімізуючи вплив суб’єктивних помилок операторів та оптимізуючи корисну площу складських приміщень.

Анотація

Дипломний проєкт на тему “Автоматизована система сортування посилок” розроблений студентом Тараном Єгором Олександровичем під керівництвом Безуглого Михайла Олександровича.

Метою проєкту є створення ефективної ланки сортування посилок, яка забезпечить розподіл посилок відповідно до параметрів сортування. Запропонована система складається з комплексу сортування, що обробляє посилки, а також програмного забезпечення для контролю процесу сортування.

Дипломний проєкт складається з пояснювальної записки та двох додатків. Пояснювальна записка викладена на 51 сторінках, містить 29 рисунків, 29 посилань на літературу. В додатку А наведено графічний матеріал (класифікацію, функціонально-структурну схему, складальний кресленик конвеєра сортування, алгоритми роботи та інтерфейс розробленого програмного забезпечення). В додатку Б наведено програмний код.

У пояснювальній записці розглянуто три розділи. У першому розділі викладено основні принципи розробки автоматизованих систем сортування, зокрема вимоги до таких систем, методи сортування та методи обробки QR-кодів. Другий розділ присвячений проєктуванню всього комплексу столу конвеєра з кріпленням камер та освітлення, зокрема функціонально-структурної схеми, вибору основних елементів, а також проєктуванню конструкційних модулів та елементів конвеєра. Третій розділ охоплює розробку програмного забезпечення системи контролю сортування, включаючи алгоритми та програмне забезпечення для різних компонентів системи. Проєкт спрямований на підвищення ефективності управління сортуванням, зниження витрат на їх обслуговування та забезпечення автономності, але й зручності для оператора. Наприкінці роботи наведено висновки та рекомендації щодо впровадження і подальшого розвитку системи.

Ключові слова: автоматизована система сортування посилок, програмне забезпечення, python.

Anotation

The thesis on "Automated Parcel Sorting System" was developed by student Yehor Oleksandrovych Taran under the supervision of Mykhailo Oleksandrovych Bezuhlyi.

The aim of the project is to create an efficient parcel sorting unit that ensures the organization of parcels according to sorting parameters. The proposed system includes a sorting complex that processes parcels, as well as software to control the sorting process.

The thesis consists of an explanatory note and two appendices. The explanatory note is presented on 51 pages, contains 29 figures, and 29 references. Appendix A contains graphical material (classification, functional and structural diagram, assembly drawing of the sorting conveyor, operating algorithms, and the interface of the developed software). Appendix B contains the source code.

The explanatory note is divided into three chapters. Chapter One outlines the fundamental principles of developing automated sorting systems, including the core requirements for such systems, sorting methods, and QR code processing techniques. Chapter Two is dedicated to the engineering and design of the entire conveyor table assembly with camera and lighting mounts, focusing on the functional and structural diagram, selection of main components, as well as the design of structural modules and conveyor elements. Chapter Three covers the software development for the sorting control system, including algorithms and software components for various parts of the system.

The project is aimed at increasing sorting management efficiency, reducing maintenance costs, and ensuring both system autonomy and operator convenience. Conclusions and recommendations for the implementation and further development of the system are provided at the end of the work.

Keywords: automated parcel sorting system, software, Python.

РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ АВТОМАТИЗОВАНИХ СИСТЕМ СОРТУВАННЯ ПОСИЛОК

1.1. Актуальність сортування посилок

У сучасному світі спостерігається стрімкий розвиток логістики, що призводить до значного зростання обсягів обробки посилок і потребує впровадження ефективних методів автоматизації.

Автоматизовані системи сортування посилок дозволяють підвищити продуктивність логістичних центрів і скоротити час обробки вантажів. Використання сучасних технологій, таких як роботизовані конвеєри та сенсорні системи, робить процес сортування швидким, надійним та економічно вигідним.

1.2. Класифікація систем сортування посилок

У цьому пункті розглянемо сучасні системи сортування посилок, їхні недоліки та переваги, принципи роботи, класифікацію та характеристики.

Системи сортування посилок класифікуються за кількома основними ознаками, що дозволяє систематизувати їх залежно від принципу роботи, конструкції та рівня автоматизації (креслення ДПБ.21.1720.001)

За рівнем автоматизації вони поділяються на ручні, напівавтоматичні та автоматизовані. Ручні системи передбачають виконання всіх операцій оператором, напівавтоматичні поєднують роботу людини з використанням технічних засобів, а автоматизовані функціонують без безпосереднього втручання оператора, використовуючи програмовані контролери, сенсори та виконавчі механізми.

За типом транспортної системи розрізняють стрічкові [1], роликові [2], cross-belt сортувальники [3], tilt-tray системи [4], shoe sorter [5] та pop-up сортувальники [6]. Стрічкові системи переміщують посилки по конвеєрній стрічці з подальшим відведенням у потрібному напрямку. Роликові системи транспортують посилки за допомогою роликів, що обертаються або працюють під дією сили тяжіння. Cross-belt сортувальники використовують окремі малі

стрічки для кожної посилки, що забезпечує високу точність і швидкість сортування. Tilt-tray системи здійснюють сортування шляхом нахилу лотків, з яких посилки скидаються у визначене місце. Shoe sorter переміщує посилки за допомогою рухомих штовхачів, які зсувають їх з основної транспортної лінії, а pop-up сортувальники змінюють напрям руху посилки за допомогою підйомних роликів або коліс.

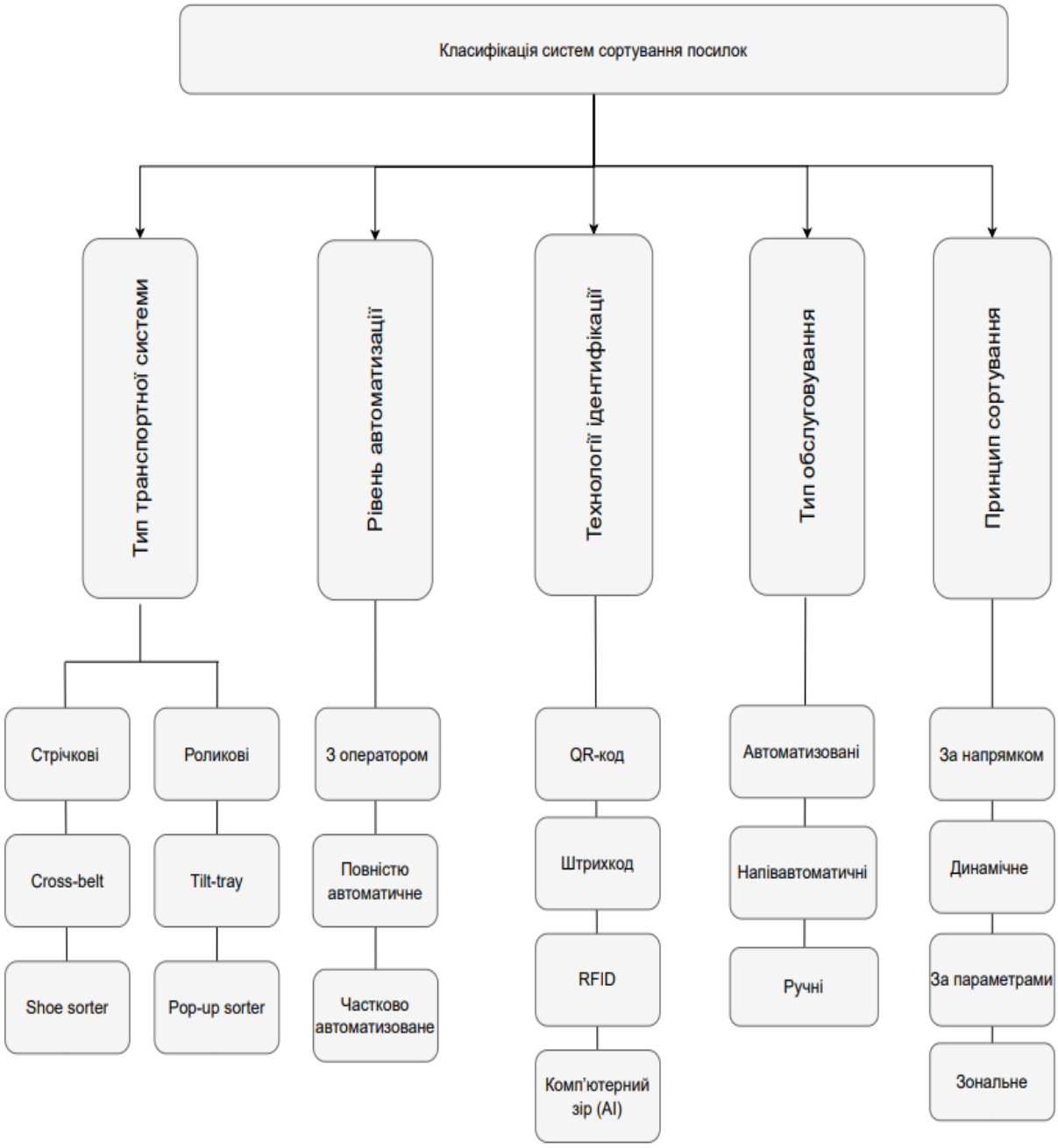


Рисунок 1.1. Класифікація систем сортування

За типом обслуговування системи можуть бути з оператором, частково автоматизованими або повністю автоматичними. За технологіями ідентифікації посилки системи поділяються на такі, що використовують штрихкоди, QR-коди, RFID-мітки або технології комп'ютерного зору. Штрихкод є одновимірним графічним кодом для зчитування оптичним сканером, QR-код дозволяє зберігати більший обсяг інформації, RFID-мітки забезпечують безконтактну ідентифікацію за допомогою радіосигналів, а комп'ютерний зір використовує камери та алгоритми обробки зображень для розпізнавання посилок і їх характеристик.

За принципом сортування розрізняють системи, що працюють за напрямком доставки, фізичними параметрами посилки, зональним принципом або динамічним маршрутизуванням у реальному часі. Така класифікація дозволяє обґрунтовано обирати тип системи сортування залежно від вимог логістичного об'єкта, необхідної продуктивності та рівня автоматизації.

1.3. Огляд сучасних системи сортування

Системи **Smart Logitex** [7] використовуються для великої зони розвантаження посилок. Він складається з телескопічних стрічкових конвеєрів, високошвидкісних стрічкових конвеєрів, двовимірних поділювачів насипного потоку, обладнання для виявлення DWS, обладнання для ідентифікації RFID та модулів сортування. Завдяки вдосконаленій системі керування рішення забезпечує автоматизоване сортування посилок від розвантаження до завантаження.

Загальний вигляд системи наведено на рис.1.2. Розглянемо переваги та недоліки системи Smart Logitex.

Переваги:

- Співпраця з кількома перевізниками для гнучкого покриття пікових навантажень (до 6000 посилок/год)
- Модульна конструкція, що дозволяє легко адаптувати систему під різні умови та типи вантажів

- Висока точність сортування (понад 99%) завдяки 6-сторонньому скануванню та відстеженню в реальному часі
- Можливість обробки великих і малих посилок у межах єдиної системи
- Висока пропускна здатність і ефективність сортування
- Дбайливе сортування без механічних ударів (зменшення пошкоджень)
- Можливість організації безпілотного (автоматизованого) сортувального центру
- Гнучке налаштування під різні швидкості та конфігурації лінії
- Багатошарова структура, що економить простір і підвищує продуктивність
- Точне розпізнавання складних об’єктів (конверти, чорні пакети тощо) за допомогою систем комп’ютерного зору

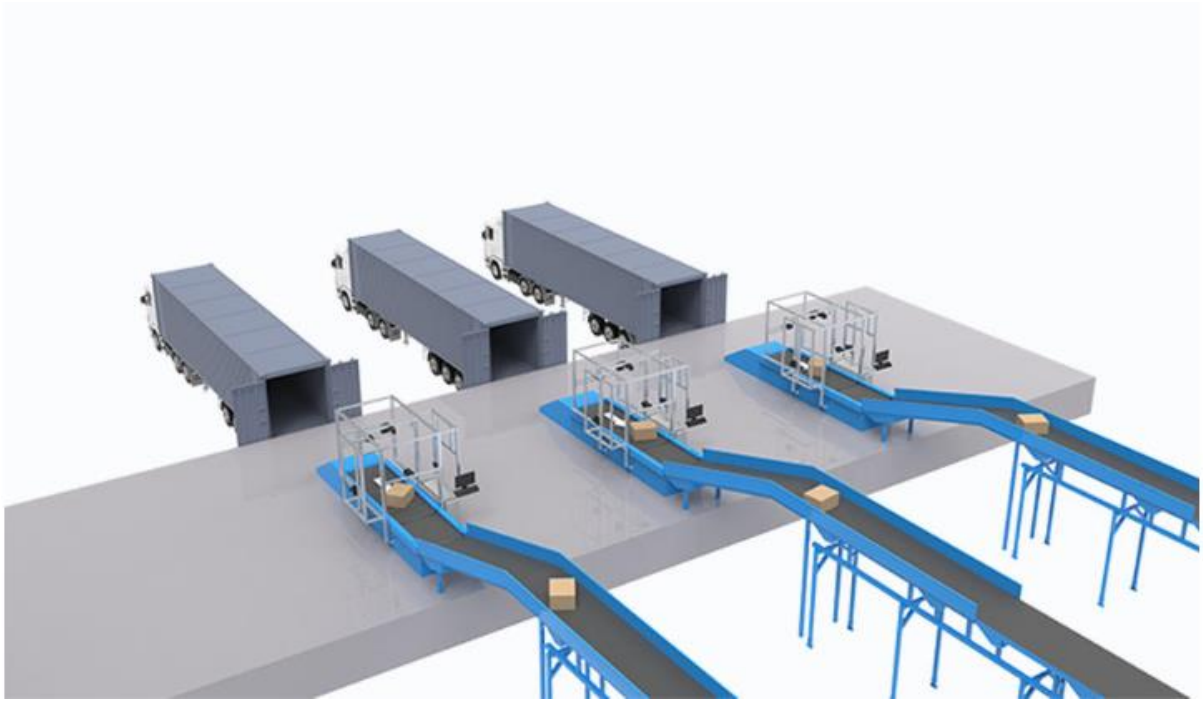


Рисунок 1.2. Система сортування Smart Logitex [7]

Недоліки:

- Висока складність системи (велика кількість модулів і підсистем);
- Значні початкові витрати на впровадження (камери, RFID, конвеєри, автоматика);
- Залежність від коректної роботи датчиків і камер (можливі збої);

- Потреба у кваліфікованому обслуговуванні та налаштуванні;
- Складність інтеграції в існуючі логістичні системи;
- Обмеження ефективності при нестандартних або дуже великих вантажах;
- У разі відмови одного модуля може зупинитися частина або вся система.

Системи **Basler** [8] забезпечують 360°-сканування посилок (рис.1.3), а також автоматичне розпізнавання кодів і тексту в будь-якому положенні.

Коди ідентифікують відправлення, матеріали, товари, контейнери та місця зберігання: автоматичне розпізнавання та зчитування кодів і тексту відіграють вирішальну роль в обробці посилок і логістиці. Завдяки 6-сторонньому скануванню посилки можна розміщувати несортованими на конвеєрних стрічках. На відміну від лазерних сканерів, система сканування з обробкою зображень є більш надійною та багатофункціональною: різні 1D та 2D коди, а також розпізнавання тексту (OCR) можуть зчитуватися одночасно.

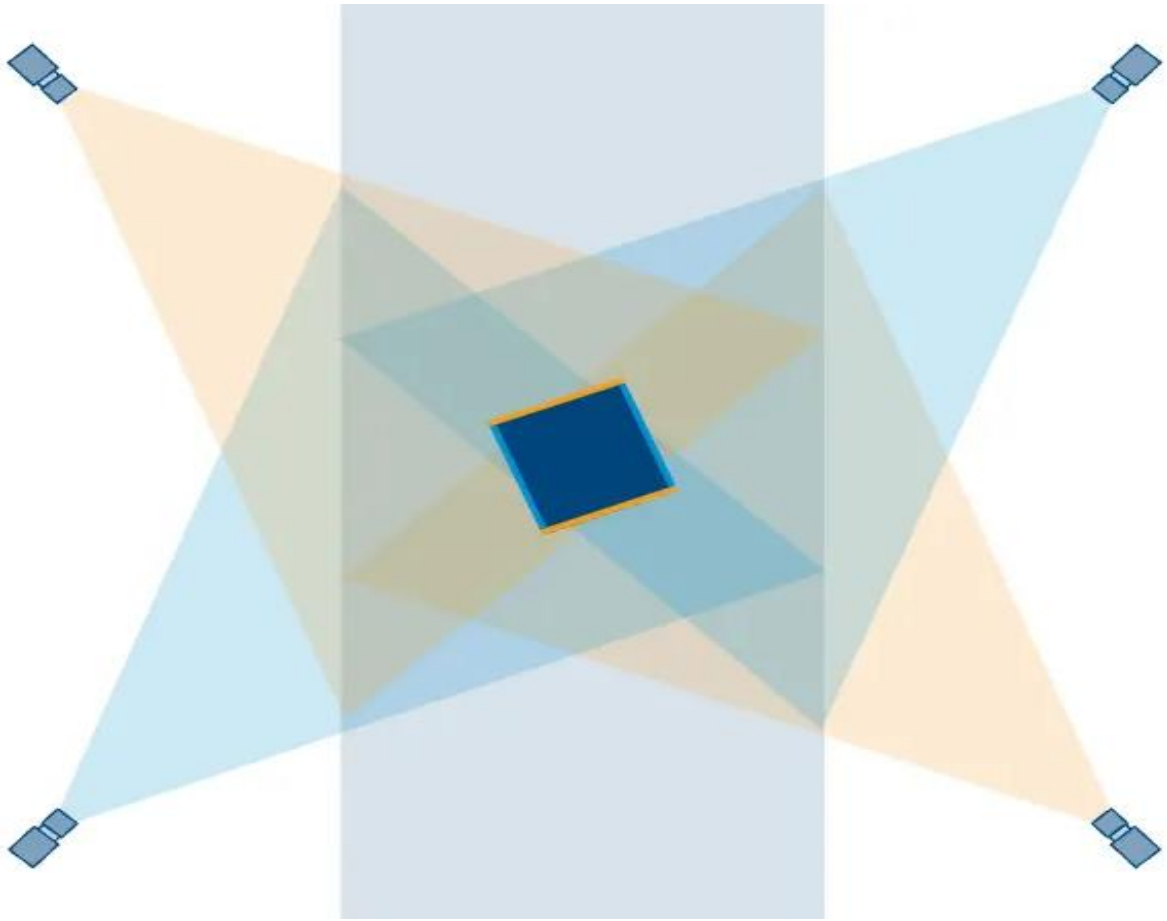


Рисунок 1.3. 360° сканування продукту Basler [8]

Переваги:

- Одночасне 360° сканування всіх граней посилки;
- Одночасне зчитування 1D, 2D кодів і тексту (OCR);
- Стійкість до складних умов (поганий контраст, відблиски, нерівні поверхні) завдяки AI OCR;
- Висока точність розпізнавання навіть пошкоджених або неправильно розташованих етикеток.

Недоліки:

- Висока вартість впровадження (камери, освітлення, ПЗ, інтеграція);
- Складність налаштування (оптика, освітлення, кути, синхронізація);
- Залежність від умов зйомки (контраст, освітлення, матеріал поверхні);
- Потреба у точному калібруванні системи для стабільної роботи;
- Можливі труднощі з дуже дрібним або сильно пошкодженим текстом;
- Залежність від AI (система є складнішою в налаштуванні).

Система **CubiQ** [9] — це автоматизована система сортування посилок, яка використовує AI та комп'ютерний зір для зчитування QR-кодів, штрих-кодів та тексту (OCR) і автоматично направляє посилки по конвеєру відповідно до заданих правил (WMS/ERP).

Переваги:

- Висока швидкість роботи — до 4000 посилок/год;
- Висока точність — мінімізація помилок завдяки AI та машинному зору;
- Компактність (вертикальне сортування) — економія площі складу.

Недоліки:

- Висока вартість впровадження (обладнання + інтеграція);
- Залежність від AI (система є складнішою в налаштуванні);
- Потреба інтеграції з WMS/ERP — складність налаштування системи;
- Обмеження за розмірами посилок (краще працює з малими/середніми об'єктами).

Рисунок 1.4. Продукт CubiQ з використанням AI [9]

BeeVision BeeSort [10] — це автоматизована система сортування посилок, яка використовує AI, комп’ютерний зір, сканування QR/штрихкодів та OCR для ідентифікації і розподілу вантажів по конвеєру. Система включає модулі зважування, вимірювання та сканування (DWS), працює з різними типами сортувальників (cross-belt, tilt-tray).

Переваги:

- Модульність і масштабованість — систему можна адаптувати під будь-який склад і легко розширювати
- Висока продуктивність — до 5000+ посилок/год

- Комплексна обробка (DWS) — одночасно вимірює, зважує та зчитує QR/штрих-код

Недоліки:

- Складність системи — потребує налаштування та інтеграції з WMS/API
- Висока вартість для малого бізнесу
- Залежність від AI (система є складнішою в налаштуванні)
- Габарити системи — потребує більше місця, ніж компактні сортери

Рисунок 1.5. Продукт BeeVision [10]

1.4 Особливості застосування маніпулятора в автоматизованих системах сортування посилок

Математичне забезпечення маніпулятора базується на кінематичних розрахунках, що дозволяють визначати положення та орієнтацію маніпулятора. Основою таких розрахунків є пряма та зворотна задача кінематики [11].

Пряма задача щодо кінематики роботів формується так: визначити кінематичні параметри вантажу при відомих кінематичних параметрах окремих ланок.

Зворотна задача кінематики полягає у визначенні потрібних параметрів руху окремих ланок, які забезпечують переміщення схопу за заданою траєкторією із заданими обмеженнями за вищими похідними в певній системі координат. Аналогічно формується прямі та зворотні задачі динаміки. Пряма задача — визначити сили та моменти на схопі за відомими силами та моментами, що діють на осях окремих ланок, а також які сили та моменти треба прикласти до осей окремих ланок, щоб на схопі сформувалися потрібні статичні та динамічні навантаження. В більш широкому сенсі можна сказати, що прямі задачі відповідають задачам аналізу, а зворотні — синтезу.

Для знаходження розв'язку оберненої задачі кінематики по заданій матриці маніпулятора більш придатним є геометричний підхід, який дає також і спосіб вибору єдиного рішення для конкретної конфігурації маніпулятора. Він включає в себе розділення завдання на декілька 2D рішень та вирішенні кожного окремо (рисунок 1.6).

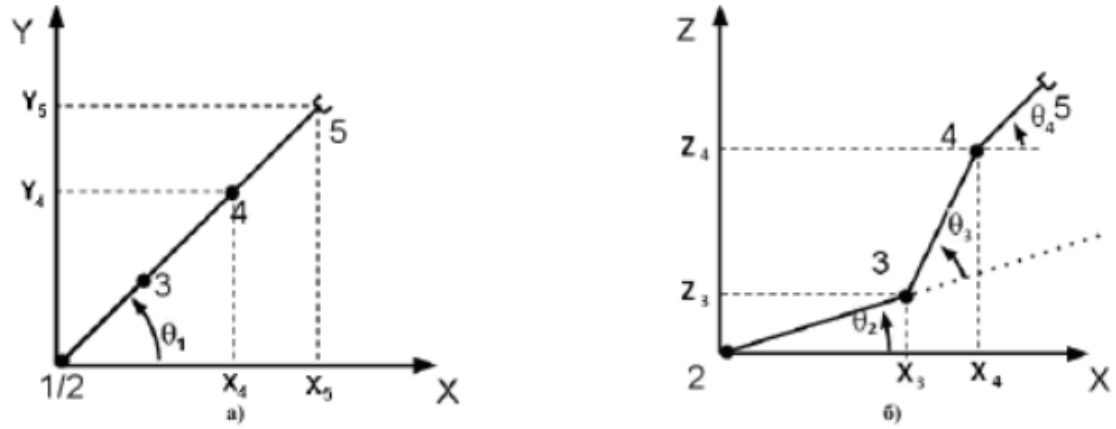


Рисунок 1.5. Маніпулятор: а) в площині XY; б) в площині XZ [11]

Перше завдання буде знаходженням θ з вигляду зверху. Друге для знаходження θ_1, θ_2, θ_3. Це робиться, враховуючи ланку як 2D-завдання в новій площині, де вона являє собою гіпотенузу, утворену кінематичним положенням в поперечній площині.

Для знаходження θ використаємо вираз (1.1):

$$\theta = arctan2(y, x). \quad (1.1)$$

Для знаходження θ_1, θ_2, θ_3 спочатку знайдемо r (2), що буде підставлятись замість p_x , де r відстань від маніпулятора до об'єкту та p_y - висота від маніпулятора (1.2):

$$r = \sqrt{x^2 + y^2}\ . \quad (1.2)$$

Для 3-ланкового 2D маніпулятору: $a1, a2, a3 -$ довжина ланок, $p_x\ , p_y -$ координати кінця маніпулятора, φ — кут орієнтації останньої ланки, θ_1, θ_2, θ_3 — кути суглобів

Спочатку знаходять точку з'єднання другої і третьої ланки (1.3):

$$w_x = p_x - a_3 cos(\varphi), \quad (1.3)$$
$$w_y = p_y - a_3 cos(\varphi),$$
$$\Delta = {w_x}^2 + {w_y}^2.$$

Кут другого суглоба знаходиться із закону косинусів (1.4) та (1.5):

$$c_2 = cos(\theta_2) = \frac{\Delta - {a_1}^2 - {a_2}^2}{2a_1 a_2} \quad (1.4)$$

$$s_2 = sin(\theta_2) = +\text{-}\sqrt{1 - {c_2}^2} \quad (1.5)$$

Тоді

$$\theta_2 = arctan2(s_2, c_2).$$

Обчислення кута першого суглоба маніпулятора (1.6) та (1.7):

$$s_1 = \frac{(a_1+a_2c_2)w_y - a_2s_2w_x}{\Delta} \quad (1.6$$

$$c_1 = \frac{(a_1+a_2c_2)w_x - a_2s_2w_y}{\Delta} \quad (1.7)$$

$$\theta_1 = arctan2(s_1, c_1)$$

Знаходження кута θ₃:

Останній кут визначається з орієнтації ефектора (1.8):

$$\theta_3 = \varphi - \theta_1 - \theta_2 \quad (1.8)$$

1.5. Постанова задачі

Провівши огляд та аналіз сучасних автоматизованих в даному проєкті запропоновано систему, що характеризується за наступними параметрами:

Ефективність: система дозволяє швидко сканувати та ефективно керувати сортуванням посилок, але й дозволяти людині спостерігати за процесом.

Зручність: система легко збирається, швидко налаштовується, не використовує дорогих і складних технологій.

Функціональність: система автоматично обробляє QR-коди, сортуючи посилки відповідно до заданого параметра. Система має додатковий функціонал, що дозволяє переглядати список оброблених посилок.

РОЗДІЛ 2. ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ СОРТУВАННЯ ПОСИЛОК

2.1. Розробка структурно-функціональної схеми автоматизованої системи сортування посилок

Функціональні компоненти автоматизованої системи сортування посилок (АССП) та їхній взаємозв'язок наведено на рисунку 2.1.

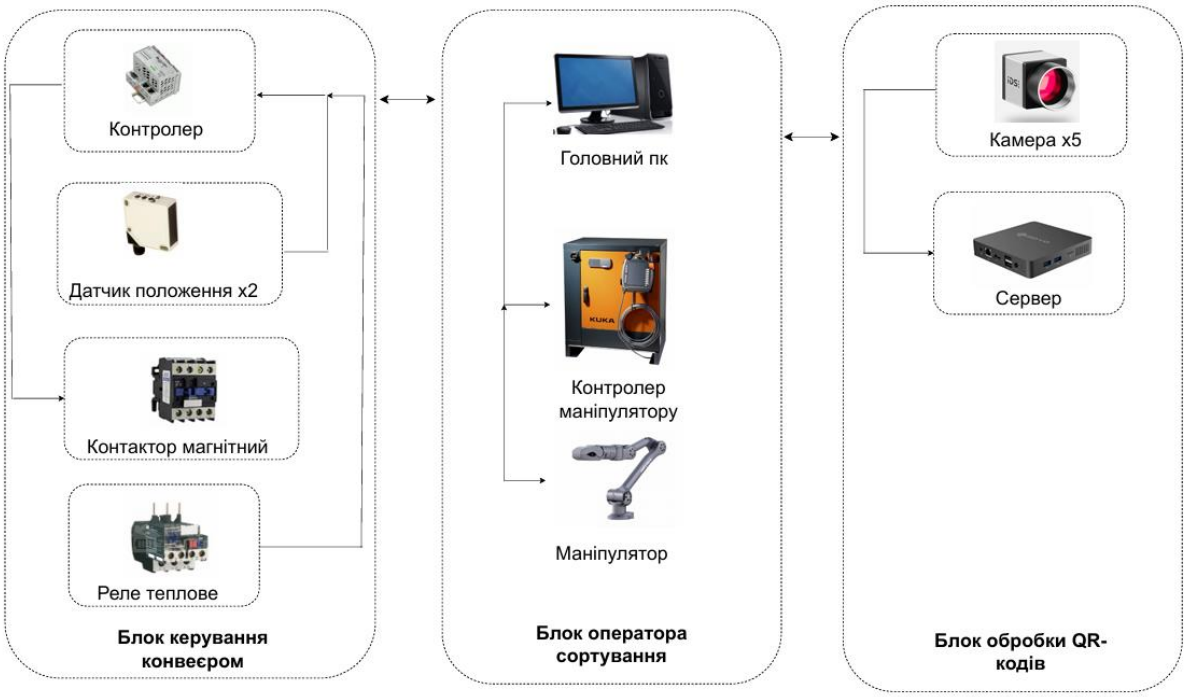


Рисунок 2.1. Структурно-функціональна схема АССП: блок 1 — блок керування конвеєром, блок 2 — блок керування застосунком сортування, блок 3 — блок обробки зображень

2.1.1. Принцип роботи відповідно до ФСА

У цьому розділі запропоновано структурно-функціональну схему АССП. Розглянемо основні елементи АССП та їх призначення:

Головний ПК у системі відповідає за керування сортуванням: спостереження за ним, керування ним, а також за мануальне втручання. Система є автономною, але дозволяє керування оператором. До головного ПК також підключене керування **маніпулятором**.

Цей комп’ютер є ключем керування системою, яка розташована на окремому **сервері**, де відбувається обробка та зберігання інформації.

Сканування відбувається за допомогою **5 камер**, які одночасно сканують всі грані посилки.

Для керування конвеєром і датчиками на ньому було розроблено окремий блок, який контролюється **контролером.** До контролера підключені елементи керування двигуна (**теплове реле, контактор-магнітний, кнопки керування**) та датчики положення для посилок.

2.1.2. Вибір компонентів АССП

Для керування модулем сортування посилок було обрано програмований логічний контролер WAGO серії PFC200 [12]. Даний контролер належить до класу промислових PLC середнього рівня та призначений для автоматизації технологічних процесів у транспортних, пакувальних та сортувальних системах (рисунок 2.2).

Рисунок 2.2. WAGO PFC200 [12]

WAGO PFC200 підтримує програмування відповідно до стандарту IEC 61131-3 (Ladder Diagram, Structured Text, Function Block Diagram тощо), що

забезпечує гнучкість розробки алгоритмів керування. Контролер має модульну архітектуру та працює з системою вводу/виводу серії WAGO 750, що дозволяє легко масштабувати систему відповідно до кількості датчиків та виконавчих механізмів.

Основні технічні характеристики контролера WAGO PFC200 наведено в таблиці**.**

Таблиця 2.1. Параметри WAGO PFC200 [12]

Параметр	**Значення**
Модель	WAGO PFC200
Процесор	32-bit ARM Cortex
Робоча напруга	24 V DC
Оперативна пам’ять	256 МБ RAM
Flash-пам’ять	512 МБ
Підтримка протоколів	Modbus TCP, OPC UA, HTTP
Тип системи вводу/виводу	Модульна, серія WAGO 750

Для передачі зображення було обрано промислові камери серії uEye [13] від компанії IDS Imaging Development Systems обрано як оптимальні компоненти для реалізації модуля 6-стороннього сканування на основі таких техніко-експлуатаційних міркувань (рисунок 2.3).

Рисунок 2.3. Камера uEye CMOS [13]

Таблиця 2.2. Параметри камери uEye CMOS [13]

Параметр	Значення
Тип сенсора	CMOS
Тип затвора	Global Shutter
Роздільна здатність	1920×1200 (≈2.3 Мп)
Розмір пікселя	~3.45 µm
Частота кадрів	до 60 fps
Інтерфейс передачі даних	GigE або USB 3.0
Підтримка тригера	Апаратний (Hardware Trigger)
Формат зображення	Mono8 / RGB8
Живлення	5–12 В (залежно від інтерфейсу)

З датчиків положення було обрано промислові датчики Omron E2B-M12KS04-M1-B1 [14]. Вони є гарним вибором для автоматизації, характеристики яких показані в таблиці 2.4 (рисунок 2.4).

Рисунок 2.4. Omron E2B-M12KS04-M1-B1 [14]

Таблиця 2.4. Характеристики Omron E2B-M12KS04-M1-B1 [14]

Параметр	**Значення**
Тип датчика	Індуктивний
Серія	E2B
Діаметр корпусу	M12
Дальність спрацювання	4 мм
Тип монтажу	Квазі-екранований (quasi-flush)
Вихід	PNP
Режим роботи	NO (нормально відкритий)
Живлення	10–30 В DC
Підключення	Роз'єм M12

2.2. Розробка електричної схеми конвеєру

Для керування двигуном конвеєра та датчиками положення було розроблено електричну принципову схему, що показана на рисунку 2.5.

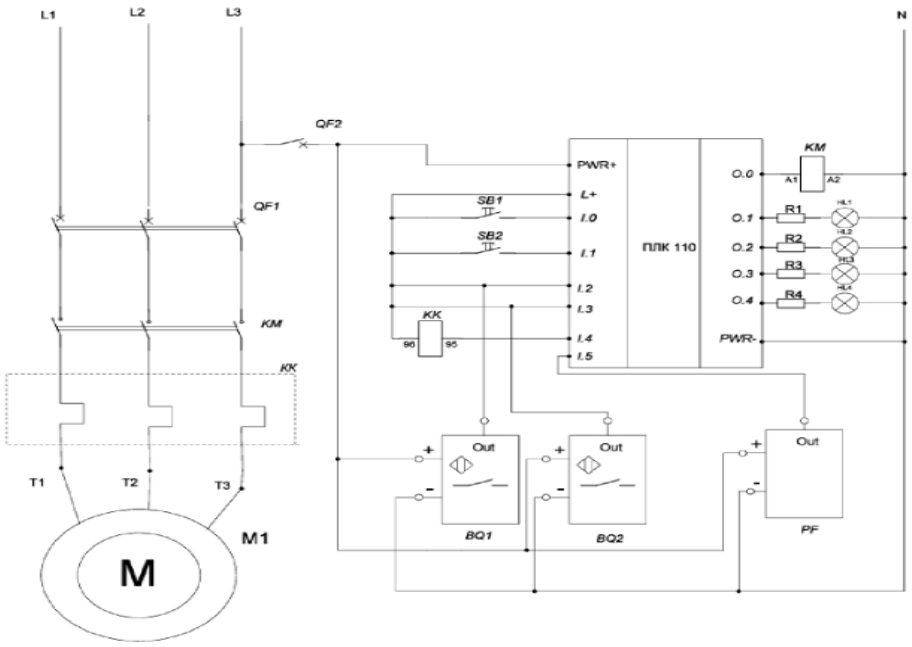


Рисунок 2.5. Електрична схема принципова підключення системи

2.2.1. Принцип роботи відповідно до електричної схеми

На електричній схемі показано підключення двох датчиків положення: один для перемикання двигуна, другий — для подачі сигналу маніпулятору про перевезення посилки. Також показано підключення двигуна конвеєра, який підключено через магнітний контактор, а також теплового реле для керування двигуном через контролер. Також було додано чотири світлодіоди та резистори для сигналізації датчиків і роботи двигуна.

2.2.2. Вибір компонентів конвеєру відповідно до розрахунків

Вхідні параметри для розрахунку:

Довжина: 200 см;

Загальна маса ременя та вантажу: m <= 30 кг

Зовнішня сила: $F_a = 0$

Коефіцієнт тертя ковзної поверхні: μ = 0.3;

Діаметр ролика: D = 50 мм;

Роликова маса: $m_2 = 1$ кг;

Ефективність стрічки та ролика: η = 0.9;

Швидкість стрічки: $V = 180\ [mm/s]$

Швидкість вихідного валу редуктора визначається як (2.1):

$$N = \frac{V*60}{\pi*D}, \tag{2.1}$$

$$N = \frac{180 * 60}{\pi*50} = 68.7 \text{ об/хв.}$$

Оскільки номінальна швидкість асинхронного двигуна (4-полюсного) при 50 Гц становить 1200–1300 [об/хв], виберемо передавальне число редуктора в цьому діапазоні:

$$i = \frac{1300}{68.7} = 18.$$

Коефіцієнт тертя ковзної поверхні (2.2):

$$F = FA + m \cdot g\ (sin\theta + \mu \cdot cos\theta), \quad (2.2)$$
$$F\ =\ 0\ +\ 30\ *\ 9.8\ (sin\ 0\circ\ +\ 0.3cos0\circ) = 88.2\ N.$$

Крутний момент навантаження (2.3):

$$T'_L : \frac{F*D}{2*\eta}\ , \quad (2.3)$$
$$T'_L\ =\ \frac{88.2*50}{2*0.9} = 2{,}4\ N \cdot m.$$

Розглянемо коефіцієнт безпеки *Sf* = 2:

$$T_L\ =\ T'_L\ *\ Sf\ = 4{,}8\ N \cdot m.$$

Для передаточного числа 18 та крутного коефіцієнту 4.8 було обрано двигун та редуктор: 5IK60GU-AWU [9] + 5GN18KA [10] (рисунок 2.6 та 2.7) з відповідними параметрами:

потужність двигуна 5IK60GU-AWU: 60 W,

тип: AC induction gear motor

gear ratio: 18:1.

Перетворимо цей крутний момент навантаження на значення на вихідному валу двигуна, щоб отримати необхідний крутний момент ТМ (2.4) . (Для 5GN18KA n_G = 0.68):

$$T_M\ =\ \frac{T_L}{i * n_G} \quad (2.4)$$
$$T_M\ =\ \frac{4.8}{18 * 0.68}\ = 0{,}37\ N \cdot m$$

Пусковий крутний момент 5IK60GU-AWU 0.38 $N \cdot m$, що задовольняє потрібний крутний момент в 0.37 $N \cdot m$.

Рисунок 2.6. Електродвигун 5IK60GU-AWU [15]

Рисунок 2.7. Редуктор 5GN18KA [16]

Для керування електродвигуном було обрано малогабаритний контактор IEK KKM21-025-230-10-U [17], що показаний на рисунку 2.8. Контактор — це електромагнітний комутаційний апарат для дистанційного вмикання та вимикання електродвигунів і інших навантажень у мережах змінного струму. Його характеристики показані в таблиці 2.5.

Рисунок 2.8. Магнітний контактор КМВ-22510 25А 220В [17]

Таблиця 2.5. Характеристики КМВ-22510 [17]

Параметр	Значення
Номінальний струм (AC-3)	25 А
Напруга котушки керування	220–230 В AC
Кількість допоміжних контактів	1 NO (нормально відкритий)
Потужність двигуна (AC-3, 400 В)	до 11 кВт

Теплове реле РТІ-1321 [18], що показано на рисунку 2.9 — це захисний електромеханічний пристрій, який встановлюється на контактор і призначений для захисту електродвигунів від перевантаження, перекосу фаз та заклинювання ротора, характеристики показані в таблиці 2.6.

Рисунок 2.9 Теплове реле РПІ-1321 [18]

Таблиця 2.6. Характеристики РТІ-1321 [18]

Параметр	Значення
Діапазон струму	12–18 А
Клас спрацювання	10
Кількість контактів	1 NO + 1 NC
Номінальна напруга ізоляції	до 660 В

2.2.3. Розробка програми керування контролером

В цьому підрозділі описаний алгоритм роботи контролеру конвеєру та складена ladder-схема керування. На рисунку 2.10. показаний алгоритм роботи електричних компонентів АССП відповідно до електричної принципової схеми.

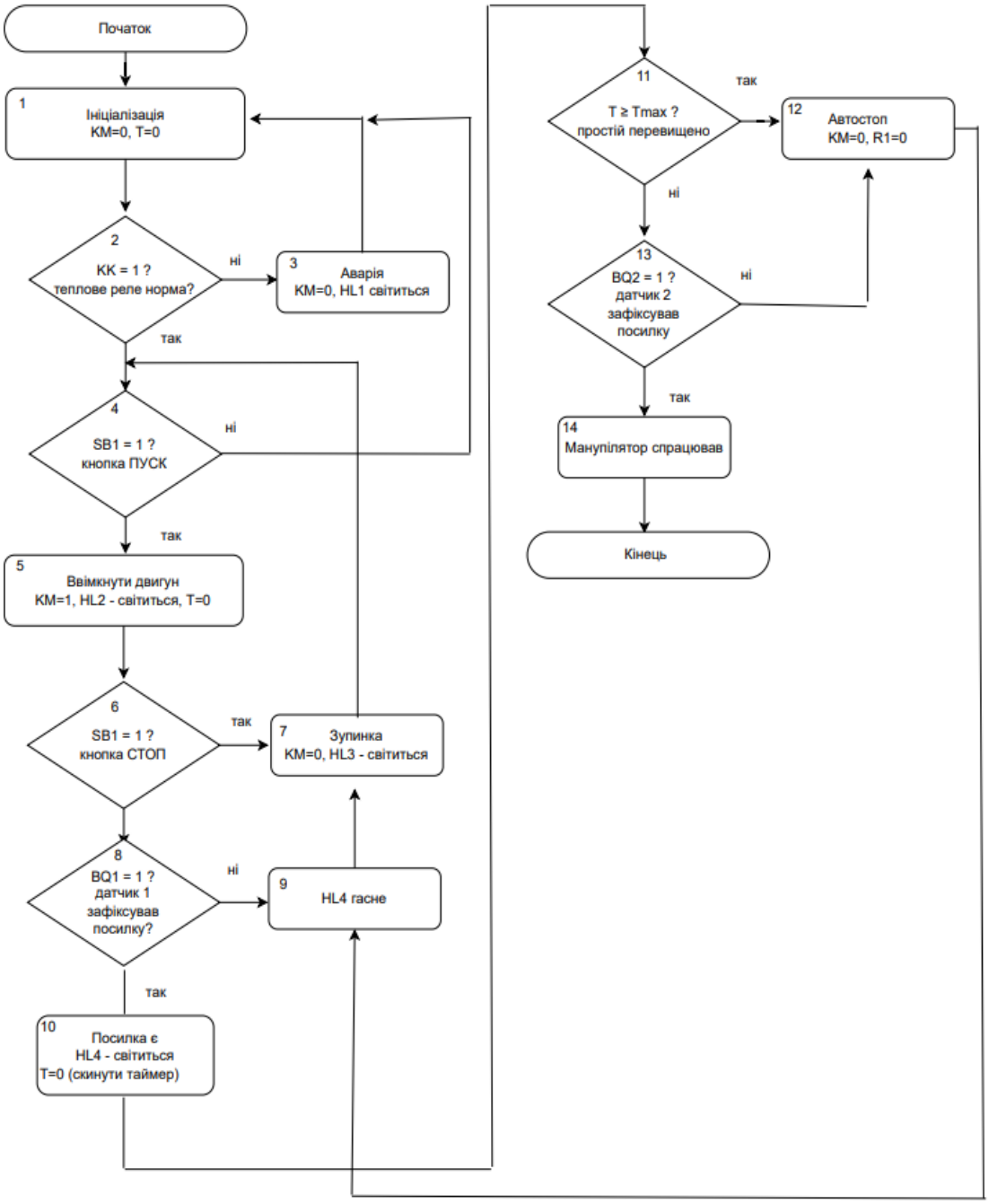


Рисунок 2.10. Блок-схема роботи конвеєру відповідно до електричної принципової схеми

Блок 1: Ініціалізація всіх електричних компонентів системи конвеєру.

Блок 2: Перевірка теплового реле

Блок 3: Подається сигнал на магнітний контактор, двигун вимикається, засвічується світлодіод HL1.

Блок 4: Перевірка чи оператор запустив двигун конвеєру

Блок 5: Відправляється сигнал на магнітний контактор KM, світиться HL2.

Блок 6: Перевірка чи оператор вимкнув двигун конвеєру

Блок 7: Відправляється сигнал на магнітний контактор KM, двигун вимикається, світиться HL3.

Блок 8: Перевірка чи зафіксував положення датчик положення

Блок 9: HL4 гасне, двигун вимикається через зазначений час.

Блок 10: Посилка присутня, HL4 світиться, таймер перезапускається.

Блок 11: Перевірка таймеру

Блок 12: Зупинка конвеєру, якщо таймер очікування перевищено

Блок 13: Перевірка чи зафіксував другий положення датчик положення

Блок 14: Маніпулятор спрацював

Для реалізації алгоритму керування двигуном та датчиками було використано середовище Codesys [19] через його широкий функціонал для проєктування контролерів. В ньому було розроблено Ladder-схему, що забезпечує ручний запуск конвеєра, його зупинку та активацію маніпулятора при спрацюванні датчика позиціонування посилки (рисунок 2.11).

Перший ранг схеми реалізує логіку самопідхвату. При натисканні кнопки Start і за умови, що кнопка Stop не активна, формується сигнал Master_Coil. Даний сигнал зберігається завдяки використанню власного контакту Master_Coil у паралельній гілці, що створює ефект самопідхвату. Це дозволяє системі залишатися в активному стані після відпускання кнопки запуску.

Другий ранг забезпечує керування двигуном конвеєра. Якщо сигнал Master_Coil активний, на виході Motor формується команда на запуск виконавчого механізму (через контактор або частотний перетворювач).

Третій ранг відповідає за керування маніпулятором. За умови активного стану Master_Coil та спрацювання датчика Sensor формується сигнал Manipulator, що забезпечує виконання операції сортування.

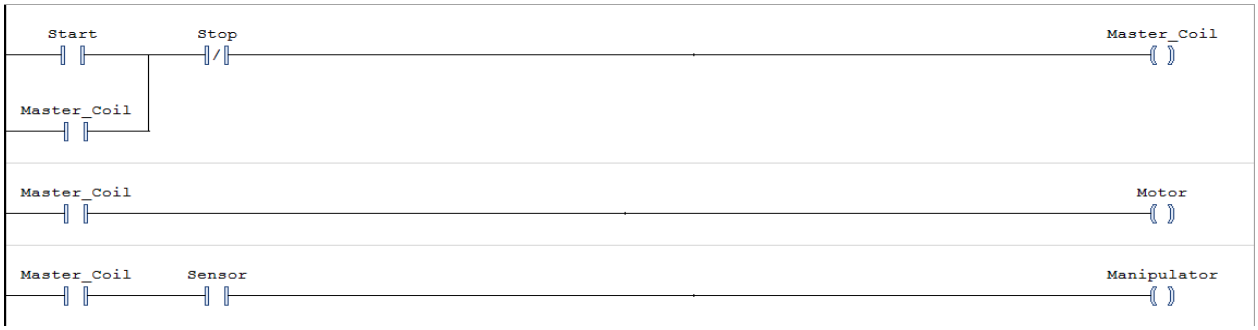


Рисунок 2.11. Логіка керування конвеєром

2.3.1. Проєктування загального вигляду конструкції

Посилка в системі автоматизованого сортування спочатку подається на стіл конвеєра, який виконує роль транспортної основи. Стіл конвеєра приводиться в рух за допомогою асинхронного двигуна, що через вал і коробку передач передає обертальний момент на стрічку конвеєра.

Конструктивно, стіл оснащений кріпленнями для забезпечення стабільності та рівномірного руху всієї системи. Під час руху посилка переміщується по конвеєру до зони сканування, де її ідентифікація здійснюється за допомогою комплексу одночасного сканування, що складається з 5 камер, що обробляють зображення всіх граней посилки.

Для проєктування було використано CAD-застосунок Solidworks [20] через його широкий функціонал та простоту в роботі. На рисунку 2.12 показано основні елементи системи:

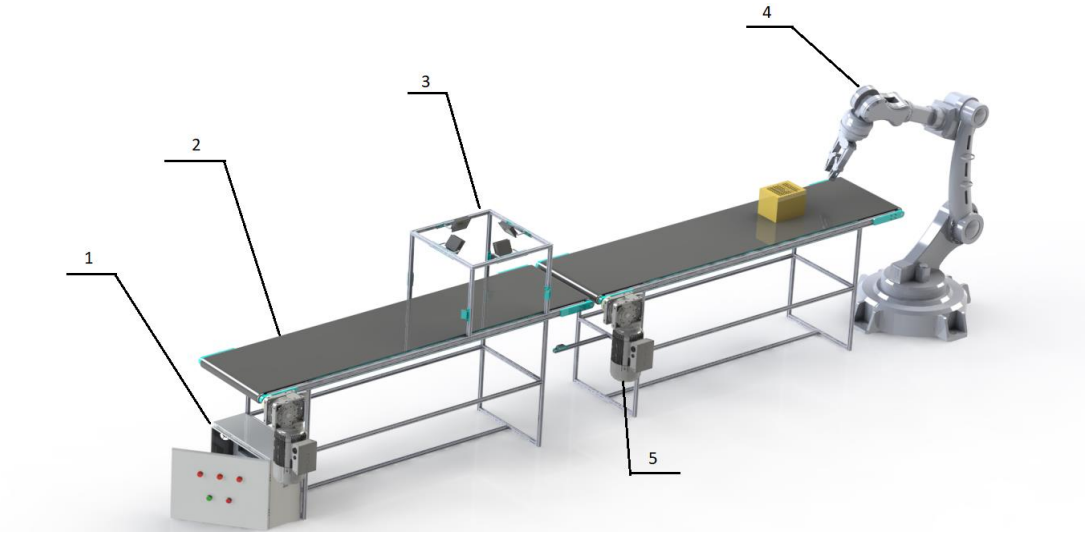


Рисунок 2.12. АССП, де 1 — електрична шафа керування, 2 — конвеєр, 3 — місце розташування камер та освітлення, 4 — маніпулятор, 5 — двигун.

2.3.2. Проектування основних вузлів

Вузол електричної шафи (1) — частина конвеєра, де знаходяться елементи контролю електродвигуна, як-от теплове реле, магнітний контактор та контролер ПЛК, що показаний на рисунку 2.13.

Рисунок 2.13. Електрична шафа конвеєра

Вузол конвеєр (2) — частина автоматизованої системи, призначена для транспортування деталей між технологічними позиціями, що забезпечує безперервність виробничого процесу та взаємодію з іншими вузлами системи, як показано на рисунку 2.14.

Рисунок 2.14. Стіл конвеєру

Вузол камери та освітлення (3) — частина автоматизованої системи, яка служить для фіксації QR-кодів посилок. Частина складається з камер та освітлення для кращої фіксації кодів, що складається з 5 камер для фіксування всіх граней посилки, показано на рисунку 2.15.

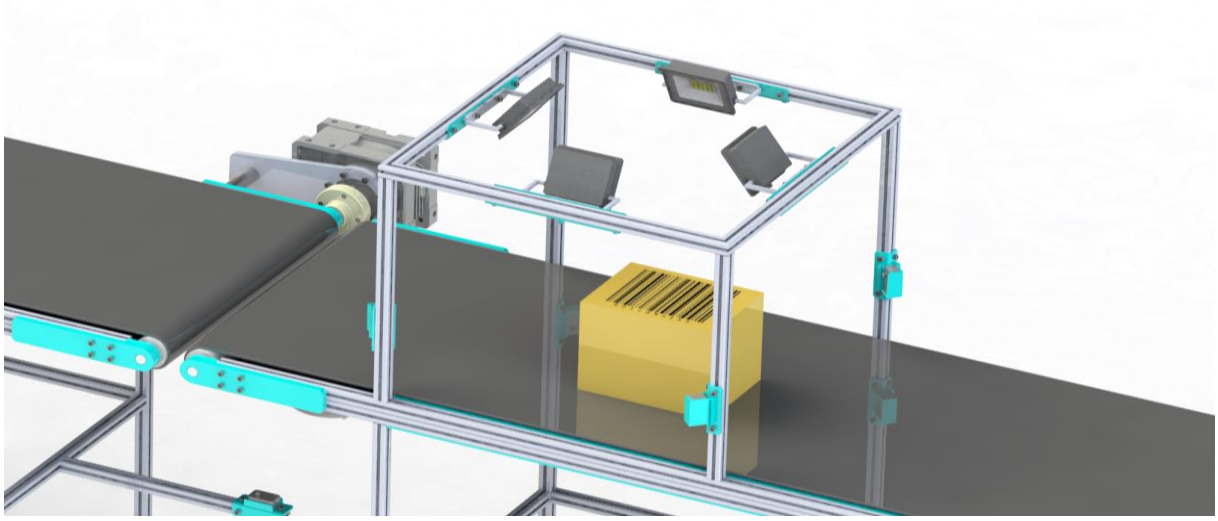


Рисунок 2.15. Вузол сканування посилок

Вузол лазерного вимірювання – частина системи, яка служить для отримання габаритних розмірів посилки. Вузол оснащений трьома лазерами для вимірювання кожної координатної осі (рисунок 2.16).

Висота (Z) - робота системи базується на порівнянні базової відстані до конвеєра з поточною точкою відбиття променю. Наприклад, якщо до порожньої стрічки зафіксовано 500 мм, а після появи вантажу камера зчитує

300 мм, то шляхом простого віднімання (500 - 300) отримуємо висоту об'єкта — 200 мм. Важливо, що лазер сканує поверхню безперервно: якщо коробка має дефекти чи деформації, комплекс зафіксує це як масив точок із різними висотними координатами.

Ширина (X) Визначення ширини об'єкта відбувається шляхом горизонтального аналізу сканованого профілю в точках різкої зміни рельєфу. Коли лазерна лінія перетинає конвеєр, програма фіксує дві ключові координати: місце, де світловий промінь різко зміщується вгору (що свідчить про виявлення лівого краю коробки), та точку його повернення на базовий рівень стрічки (яка визначає правий край). Обчислена математична відстань між цими двома крайніми точками зламу по горизонтальній осі й становить чисту ширину коробки, що дозволяє системі точно фіксувати габарити навіть у разі зміщення тари відносно центру конвеєра.

Довжина (Y) Для розрахунку довжини використовується метод послідовної збірки двовимірних зрізів, оскільки нерухомий лазер фіксує лише тонку лінію на поверхні об'єкта. Під час руху вантажу сканер безперервно фіксує профіль із частотою близько 200 кадрів на секунду, паралельно синхронізуючи кожен знімок із показниками механічного енкодера. Оскільки кожен імпульс цього датчика відповідає фіксованому зміщенню стрічки (наприклад, рівно 0.5 мм), програмний комплекс чітко визначає відстань, яку коробка подолала між сусідніми кадрами, і послідовно об'єднує ці плоскі профілі в єдину тривимірну модель для точного обчислення фінальної довжини від передньої до задньої стінки.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КЕРУВАННЯМ ПРОЦЕСОМ СОРТУВАННЯ

3.1 Вибір та обґрунтування мови програмування та основних бібліотек

У рамках розробки програмного забезпечення для керування автоматизованою системою сортування посилок було прийнято рішення використовувати мову програмування **Python** [21]. Такий вибір обумовлений рядом технічних та практичних переваг, зокрема простотою інтеграції з системами машинного зору.

Для реалізації користувацького інтерфейсу використано стандартну бібліотеку **Tkinter** [22], яка дозволяє створювати вікна, вкладки, кнопки, форми введення та прокрутку без додаткових компонентів.

Для обробки зображень та машинного зору застосовано бібліотеку **OpenCV** [23], що забезпечує підключення камери, зчитування відео-потоку та обробку кадрів у реальному часі. За допомогою даної бібліотеки виконується попередня обробка зображення, створення масок для виділення об’єктів, визначення їх контурів та координат, а також аналіз положення об’єктів на конвеєрі. OpenCV дозволяє працювати з різними колірними просторами, виконувати фільтрацію зображень, знаходити межі об’єктів та визначати їх центр, що є необхідним для подальшого керування маніпулятором у системі сортування.

NumPy [24] використовується для швидких числових обчислень та роботи з масивами даних. PIL (Pillow) [25] дозволяє відкривати, редагувати та конвертувати графічні файли. Бібліотека **Time** [26] застосовується для організації затримок і відліку часу у циклах роботи системи. Бібліотека **Serial** [28] забезпечує обмін даними між Python та мікроконтролерами через послідовний порт. Бібліотека **OS** [27] використовується для роботи з файловою системою та керування шляхами до файлів і папок.

Для зчитування QR-кодів у системі використано бібліотеку pyzbar [29], яка дозволяє розпізнавати штрихкоди та QR-коди безпосередньо із зображень або відеопотоку камери. Дана бібліотека працює спільно з OpenCV: спочатку кадр отримується з камери, після чого передається у **pyzbar** для аналізу та декодування інформації, що міститься у QR-коді. Це дає можливість автоматично визначати дані про посилку та використовувати їх для подальшого сортування.

Вибір мови **Python** для реалізації проєкту обумовлений її зручністю, гнучкістю та широким набором готових інструментів, що спрощують розробку графічного інтерфейсу та обробку даних. Завдяки цьому вдалося створити програмне забезпечення, яке є ефективним, зручним у використанні, повністю задовольняючи поставлені вимоги.

Обробка зображень системи сортування посилок відбувається за допомогою операцій бібліотеки Python OpenCV.

OpenCV (Open Source Computer Vision Library) — це потужний інструмент для комп'ютерного зору та обробки зображень, який широко застосовується у промисловості, робототехніці, безпеці та багатьох інших сферах. Цей функціонал дозволяє ефективно виявляти та відстежувати посилки на конвеєрі, розпізнавати форми, коректно визначати координати об'єктів для подальшого управління маніпуляторами та іншими виконавчими механізмами системи (Таблиця 3.1).

Завдяки OpenCV процес обробки зображень стає швидким, надійним і зручним для інтеграції в автоматизовані системи сортування.

Таблиця 3.1.Функціонал OpenCV

#	**Функція**	**Призначення**
1	cv2.cvtColor()	Конвертує колірні простори (BGR→RGB, BGR→HSV, BGR→GRAY)
2	cv2.erode()	Ерозія маски для видалення шуму
3	cv2.dilate()	Дилатація маски для заповнення прогалин
4	cv2.inRange()	Створює бінарну маску за діапазоном кольорів HSV

Продовження таблиці 3.1.

5	cv2.GaussianBlur()	Розмиває кадр перед виявленням країв
6	cv2.Canny()	Виявляє краї на розмитому зображенні у відтінках сірого
7	cv2.findContours()	Знаходить контури на бінарній масці або карті країв
8	cv2.contourArea()	Обчислює площу контуру
9	cv2.boundingRect()	Отримує обмежувальний прямокутник контуру
10	cv2.circle()	Малює коло на кадрі
11	cv2.putText()	Малює текст на кадрі
12	cv2.getTextSize()	Вимірює розмір тексту для його центрування
13	cv2.absdiff()	Обчислює абсолютну різницю між двома кадрами для виявлення руху
14	cv2.perspectiveTransform()	Перетворює піксельні координати у світові координати через гомографію
15	cv2.VideoCapture()	Відкриває камеру або відеопотік
16	cap.read()	Зчитує кадр з відеопотоку
17	cap.set()	Встановлює параметри захоплення (наприклад, розмір буфера)
18	cap.release()	Звільняє ресурс камери
19	cv2.destroyAllWindows()	Закриває всі відкриті вікна OpenCV

3.2. Розробка алгоритму програмного забезпечення

Для детального опису логіки роботи було розроблено три алгоритми: алгоритм сортування посилки, алгоритм роботи електричної схеми та алгоритм роботи програмного забезпечення.

На рисунку 3.1. показано алгоритм процесу сортування посилок, від початку посилки на конвеєру до її сортування, де:

Блок 1: Ініціалізація всіх електричних компонентів системи: камер, пк, маніпулятор.

Блок 2: Оператор включає конвеєр – початок роботи системи.

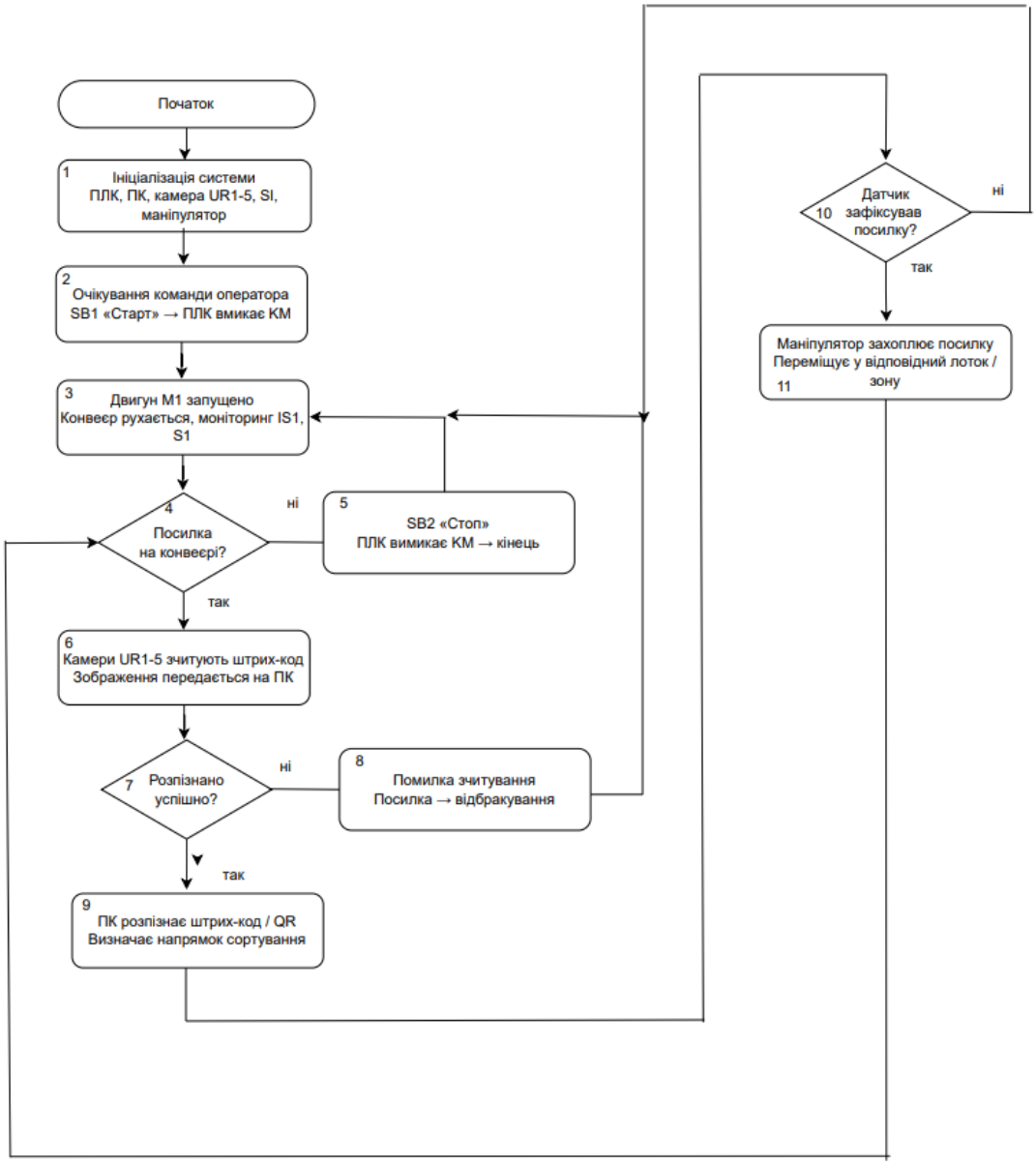


Рисунок 3.1. Блок-схема роботи АССП

Блок 3: Система моніторить датчики на наявність посилок.

Блок 4: Перевірка чи посилка на конвеєрі

Блок 5: Двигун вимикається

Блок 6: посилка доходить до камер, що сканують QR-код, передає дані на сервер

Блок 7: Перевірка чи розпізнано QR-код

Блок 8: Посилка йде на брак

Блок 9: Логіка програми на сервері вирішує шлях сортування посилки та визначає координати посилки та відправляє сигнал на маніпулятор

Блок 10: Перевірка чи посилка на конвеєрі

Блок 11: Маніпулятор відправляє посилку

На рисунку 3.2 показана блок-схема алгоритму роботи програмного забезпечення.

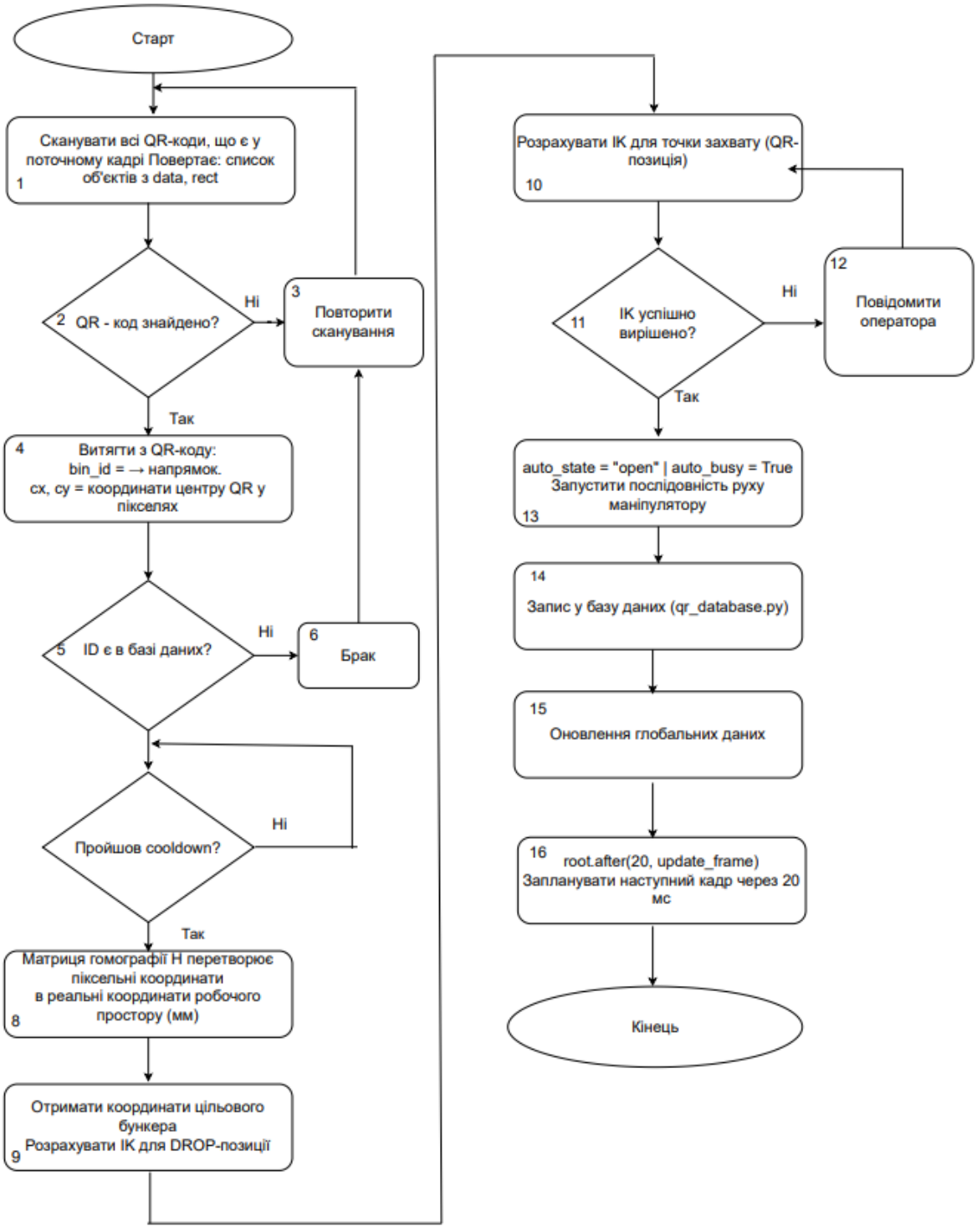


Рисунок 3.2. Блок-схема алгоритму програмного забезпечення сортування посилок

Блок 1: Сканування QR-кодів

Блок 2: Перевірка чи знайдений QR-код

Блок 3: Повторити сканування

Блок 4: Отримати інформації з QR-коду

Блок 5: Чи є id з QR-коду в локальній базі даних

Блок 6: Якщо id невідоме, то посилка йде на брак

Блок 7: Перевірка таймеру

Блок 8: Перетворення координат з камер в реальні координати

Блок 9: Отримання координат з id посилки

Блок 10: Розрахунок оберненої задачі кінематики

Блок 11: Перевірка чи розрахувались кути

Блок 12: Сигналізація оператору

Блок 13: Передача кутів на маніпулятор

Блок 14: Після успішного сортування посилки, запис у базу даних

Блок 15: Оновлення глобальних змінних

Блок 16: Запланувати наступний кадр

3.3. Розробка інтерфейсу програмного забезпечення

У графічному інтерфейсі користувача програми реалізовано головне вікно, яке містить набір вкладок для зручного сортування що показано на рисунку 3.3

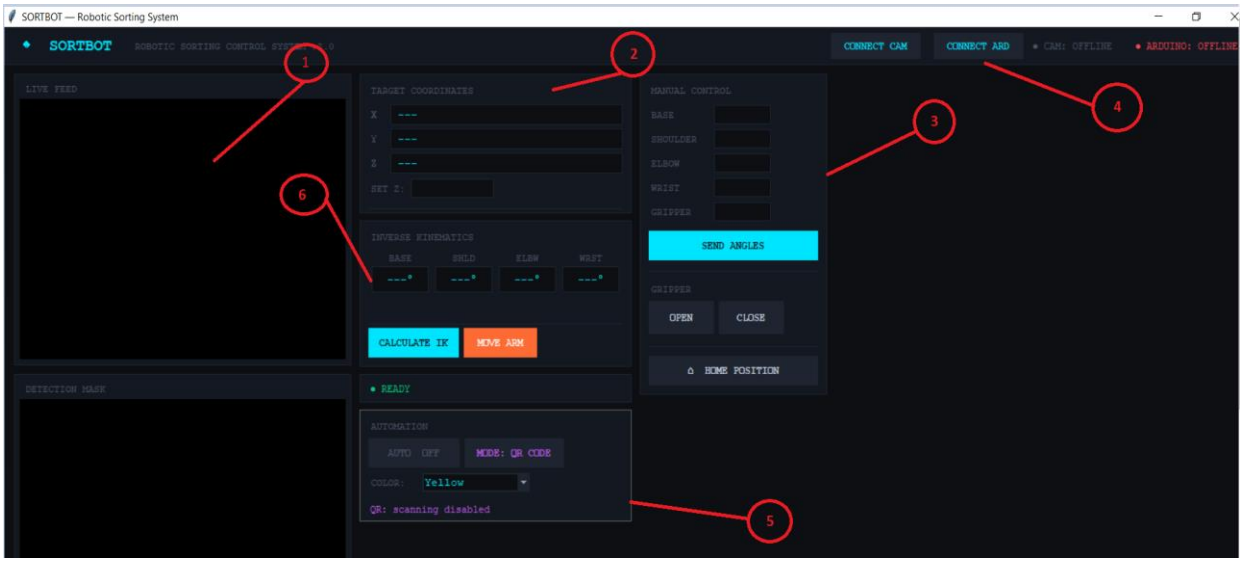


Рисунок 3.3. Інтерфейс програми

1. Вікно камери — виведення зображення камери, в цьому застосунку показано лише виведення однієї камери;
2. Вікно координат - координати об’єкта, що є точками X,Y центру посилки;
3. Вікно мануального керування, що дозволяє відправляти кути на маніпулятором;
4. Набір кнопок, що дають можливість підключення камер та контролера;
5. Набір кнопок — вибір типу сортування, мануальний розрахунок відкрити/закрити тримач, перемикач авто сортування;
6. Вікно розрахованих кутів – показує користувачу відправлені кутів на маніпулятор.

3.4. Розробка Backend програмного забезпечення

Програмне забезпечення системи сортування посилок складається з набору функцій, які забезпечують роботу камери, обробку зображення, визначення координат об’єктів, керування маніпулятором та реалізацію автоматичного режиму роботи. Основною задачею функціоналу є отримання відео-потоку з камери, аналіз зображення, знаходження об’єктів на конвеєрі та передача відповідних команд виконавчим механізмам.

Частина функцій відповідає за роботу з камерою та стабільність відео-потоку. Програма може отримувати адресу активної камери, перемикатися між кількома камерами та виконувати спроби повторного підключення у випадку втрати з’єднання. Якщо відео-потік тимчасово недоступний, система автоматично планує повторну спробу підключення та відображає інформаційне повідомлення в інтерфейсі користувача. Такий підхід забезпечує безперервність роботи системи навіть при нестабільному мережевому з’єднанні або перезавантаженні камери.

```
def update_frame():
    global current_frame, last_auto_time
    global auto_busy, auto_state, camera_available, cap

    if not camera_available:
        blank = np.zeros((300, 400, 3), dtype=np.uint8)
        cv2.putText(blank, "NO SIGNAL", (110, 155),
                    cv2.FONT_HERSHEY_SIMPLEX, 1.0, (40, 60, 80), 2)
        img = ImageTk.PhotoImage(Image.fromarray(blank))
        camera_label.imgtk = img
        camera_label.configure(image=img)
        return

    ret, frame = cap.read()
    if not ret:
        camera_available = False
        cap.release()
        cap = None
        cam_dot.config(text="● CAM: LOST", fg=C_ERROR)
        connect_btn.config(text="CONNECT CAM", fg=C_ACCENT, bg=C_BORDER)
        set_status("Camera connection lost — press CONNECT CAM to retry", "error")
        blank = np.zeros((300, 400, 3), dtype=np.uint8)
        cv2.putText(blank, "SIGNAL LOST", (90, 155),
                    cv2.FONT_HERSHEY_SIMPLEX, 1.0, (60, 30, 30), 2)
        img = ImageTk.PhotoImage(Image.fromarray(blank))
        camera_label.imgtk = img
        camera_label.configure(image=img)
        return
```

Рисунок 3.4. Функція оновлення кадру

Окремий блок функцій призначений для обробки зображень та визначення положення об’єктів у просторі. Після отримання кадру виконується аналіз зображення, що дозволяє знаходити центр посилки або іншого об’єкта на конвеєрі. Для цього застосовуються алгоритми

комп’ютерного зору, зокрема робота з кольоровими масками, пошук контурів та визначення геометричних параметрів об’єкта. Також реалізовано можливість зчитування QR-кодів, які містять інформацію про призначення посилки, наприклад, номер бункера для сортування. Після визначення координат у піксельному просторі виконується їх перетворення у реальні координати робочої зони за допомогою матриці гомографії. Це дозволяє узгодити координатну систему камери з координатною системою маніпулятора.

```
def find_object_center(mask):
    global last_center
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if not contours:
        last_center = None; return None
    largest = max(contours, key=cv2.contourArea)
    if cv2.contourArea(largest) < MIN_CONTOUR_AREA:
        last_center = None; return None
    x, y, w, h = cv2.boundingRect(largest)
    cx, cy = x + w//2, y + h//2
    new_center = (cx, cy)
    if last_center is None:
        stable = new_center
    else:
        stable = (
            CENTER_ALPHA*new_center[0] + (1-CENTER_ALPHA)*last_center[0],
            CENTER_ALPHA*new_center[1] + (1-CENTER_ALPHA)*last_center[1]
        )
    last_center = stable
    return int(stable[0]), int(stable[1])
```

Рисунок 3.5. Знаходження центру об’єкту

Для підвищення ефективності роботи системи реалізовано перевірку зміни кадру, що дозволяє уникнути зайвих обчислень, якщо зображення залишилося незмінним. Це зменшує навантаження на процесор та покращує швидкодію системи, що є важливим для роботи в режимі реального часу.

Важливим елементом програми є головний цикл оновлення кадру, який відповідає за отримання нового зображення з камери, виконання детекції об’єктів, відображення результатів у графічному інтерфейсі та передачу даних у систему керування. У цьому циклі відбувається інтеграція всіх основних

компонентів системи: комп’ютерного зору, інтерфейсу користувача та керування маніпулятором.

Функціонал програми також включає реалізацію автоматичного режиму сортування. У цьому режимі система працює за певною послідовністю станів, що відповідають етапам обробки посилки: виявлення об’єкта, розрахунок координат, переміщення маніпулятора до об’єкта, захоплення, переміщення до відповідного бункера та скидання посилки. Перехід між станами здійснюється автоматично на основі отриманих даних та поточного стану системи.

```
def auto_step():
    global auto_state, auto_state_time, auto_busy
    global last_pick_time, last_center, qr_drop_pending
    if arduino is None:
        return
    now = time.time()
    if auto_state == "open":
        open_gripper()
        auto_state = "move"
        auto_state_time = now
    elif auto_state == "move" and now - auto_state_time > 1.0:
        arduino.write(f"B {base_angle_val:.2f}\n".encode())
        arduino.write(f"S {theta1_val:.2f}\n".encode())
        arduino.write(f"E {theta2_val:.2f}\n".encode())
        arduino.write(f"W {theta3_val:.2f}\n".encode())
        auto_state = "close"
        auto_state_time = now
    elif auto_state == "close" and now - auto_state_time > 2.0:
        close_gripper()
        auto_state = "drop" if qr_drop_pending else "home"
        auto_state_time = now
    elif auto_state == "drop" and now - auto_state_time > 2.0:
        arduino.write(f"B {drop_base_val:.2f}\n".encode())
        arduino.write(f"S {drop_theta1_val:.2f}\n".encode())
        arduino.write(f"E {drop_theta2_val:.2f}\n".encode())
        arduino.write(f"W {drop_theta3_val:.2f}\n".encode())
        set_status("Moving to bin…", "info")
        auto_state = "release"
        auto_state_time = now
    elif auto_state == "home" and now - auto_state_time > 2.0:
        send_home()
        auto_state = "release"
        auto_state_time = now
    elif auto_state == "release" and now - auto_state_time > 1.0:
        open_gripper()
        if qr_drop_pending:
            send_home()
        auto_state = "idle"
        auto_busy = False
        last_pick_time = time.time()
        last_center = None
        qr_drop_pending = False
```

Рисунок 3.6. Функція автоматичного сортування

Окрім автоматичного режиму, передбачено можливість ручного керування маніпулятором. Користувач може виконувати розрахунок кутів повороту ланок маніпулятора на основі введених координат, а також надсилати ці кути безпосередньо на контролер. Також реалізовано функції для переміщення маніпулятора у початкову позицію, відкриття та закриття захвату, а також введення координат за допомогою інтерфейсу програми або кліку по зображенню з камери.

```
def send_manual_angles():
    try:
        if arduino is None:
            set_status("Arduino not connected", "error")
            return
        sent = False
        for name in ["B", "S", "E", "W", "G"]:
            val = entries[name].get().strip()
            if val:
                if name == "G":
                    if val not in ("0", "1"):
                        set_status("G must be 0 or 1", "error")
                        return
                    arduino.write(f"{name} {val}\n".encode())
                else:
                    arduino.write(f"{name} {float(val):.2f}\n".encode())
                sent = True
        if sent:
            set_status("Manual angles sent", "ok")
        else:
            set_status("No angles to send", "warn")
    except ValueError:
        set_status("Enter valid numbers", "error")
    except Exception as e:
        set_status(f"Serial error: {e}", "error")
```

Рисунок 3.7. Функція мануальної відправки координат

Таким чином, реалізований функціонал програмного забезпечення забезпечує повний цикл роботи автоматизованої системи сортування — від отримання відеоданих та аналізу зображення до керування маніпулятором і

виконання операцій переміщення об’єктів. Всі функції цього програмного забезпечення показані в таблиці 3.2.

Таблиця 3.2. Функції системи сортування посилок

#	Функція	Призначення
1	get_camera_url()	Повертає URL камери за поточним вибраним ID
2	switch_camera()	Перемикає активну камеру та оновлює інтерфейс
3	try_connect_camera()	Намагається підключитись до камери з повторними спробами

Продовження таблиця 3.2.

4	_schedule_camera_reconnect()	Планує повторне підключення камери якщо вона недоступна
5	pixel_to_world()	Перетворює піксельні координати у світові через матрицю гомографії
6	detect_qr()	Зчитує QR-код з кадру та повертає ID бункера і координати
7	find_object_center()	Знаходить центр об'єкта за кольоровою маскою з фільтрацією
8	find_box_center()	Знаходить центр коробки через виявлення контурів та країв
9	is_new_frame()	Перевіряє чи змінився кадр для уникнення зайвих обчислень
10	_show_no_feed()	Відображає заглушку з текстом коли камера недоступна
11	update_frame()	Головний цикл оновлення кадру, детекції та відображення
12	auto_step()	Керує станами автоматичного сортування (відкрити→рухатись→захопити→скинути)
13	auto_calculate_only()	Автоматично обчислює кути маніпулятора за поточними координатами
14	on_calculate()	Обробляє натискання кнопки "Розрахувати" та відображає кути

15	on_move()	Надсилає розраховані кути на Arduino
16	on_click()	Обробляє клік по кадру камери та конвертує координати
17	on_z_enter()	Обробляє введення Z-координати з клавіатури
18	send_home()	Надсилає маніпулятор у домашню позицію
19	open_gripper()	Відкриває захват та надсилає команду на Arduino
20	close_gripper()	Закриває захват та надсилає команду на Arduino
21	toggle_auto()	Вмикає або вимикає режим автоматичної детекції

Продовження таблиця 3.2.

22	toggle_sorting_mode()	Перемикає між режимами сортування за кольором та QR-кодом
23	send_manual_angles()	Зчитує кути з полів вводу та надсилає їх на Arduino вручну

3.5. Розробка рекомендацій для користувача

Для коректного використання програмного забезпечення необхідно дотримуватися визначеної послідовності дій.

Після запуску програми для автоматичного керування користувач має перевірити правильність налаштувань.

Для коректної роботи програми камери мають бути правильно відкалібровані. Для цього користувач має відвідати вкладку “Калібрування камери” та створити нову гомографію для кожної з шести камер. Для перевірки користувач має натиснути на вікні камери відому йому точку. Програма надасть координати, які були виявлені на вікні камери. Користувач має досягти точності (+-10 мм)

Далі користувач має впевнитися, що мікроконтролер маніпулятора був підключений до комп’ютера. Для перевірки користувач може мануально відправити набір кутів на маніпулятор.

Після успішного налаштування користувач може приступати до роботи. Натиснувши “Авто сортування” та обравши режим “QR-Code”, програма готова до сортування, приклад успішної роботи показаний на рисунку 3.7.

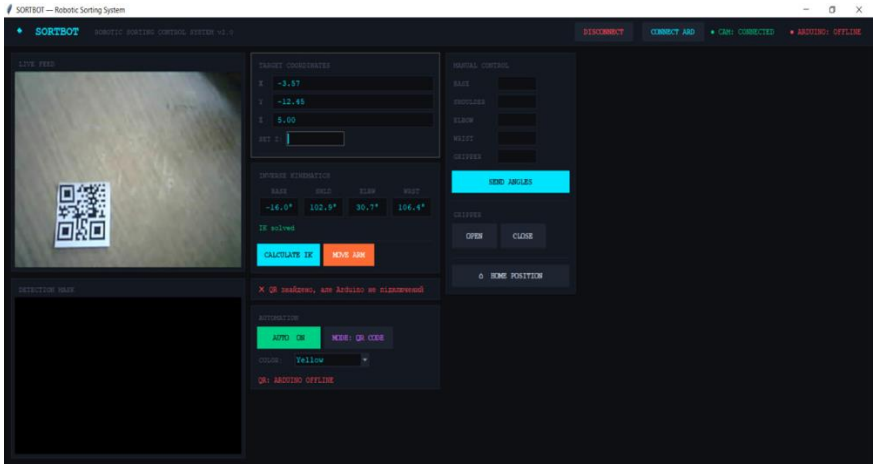


Рисунок 3.7. Робота програмного забезпечення

3.6. Тестування програмного забезпечення

В цьому розділі проведемо тестування програмного забезпечення. Спочатку перевіримо поля введення (рисунок 3.13). Поле має отримувати лише числа, але програма дозволяє передавати й літери. Кути маніпулятора не будуть розраховані, тому програма видає відповідну помилку.

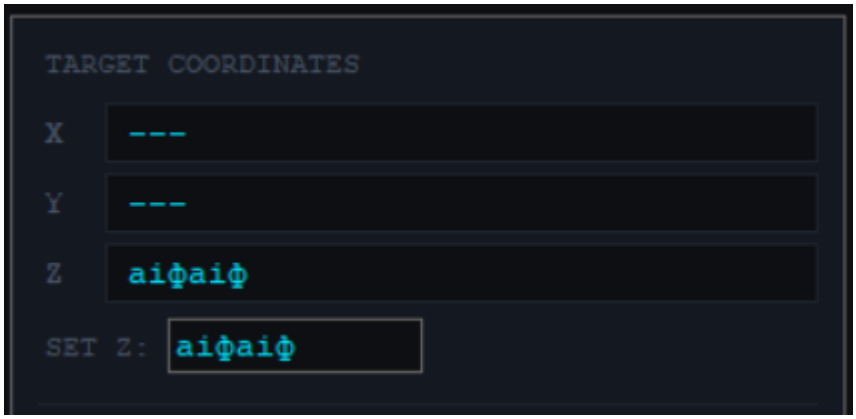


Рисунок 3.9. Поле введення висоти

Висновки

У рамках дипломного проєкту було створено сучасну та ефективну систему автоматизованого сортування посилок, яка включає розробку конструкційного модуля столу для сортування, програмного забезпечення та системи розпізнавання QR-кодів посилок.

Розробка контрукційних модулів столу сортування

Під час проектування столу сортування, що включає в собі конвеєр та кріплення на камер та освітлення, було проведено деталний аналіз існуючих аналогів. В результаті цього аналізу були визначені основні вимоги до нової моделі столу, що дозволило розробити оптимальний варіант, який поєднує в собі найкращий метод сканування всіх граней посилки.

Розробка програмного забезпечення

Під час розробки програмного забезпечення було особливу увагу приділено інтерфейсу програми. Було створено зручний інтерфейс, що дозволяє користувачам легко здійснювати сортування, мати контроль над процесом та можливість вручну контролювати розрахунок кутів і надсилати їх на контролер маніпулятора.

Важливим компонентом програмного забезпечення є розрахунок оберненої задачі кінематики та розпізнавання QR-кодів. Алгоритми обробки зображень дозволяють автоматично розпізнавати інформацію, записану в коді та інтегрувати її з базою даних.

Таким чином, у рамках дипломної роботи було створено ефективну автоматизовану систему сортування посилок, що включає розробку конструкційних модулів столів сортування, програмного забезпечення, системи обрахунку оберненої задачі кінематики та сканування QR-кодів. Завдяки проведеній роботі, вдалося досягти надійної роботи системи.

Список джерел

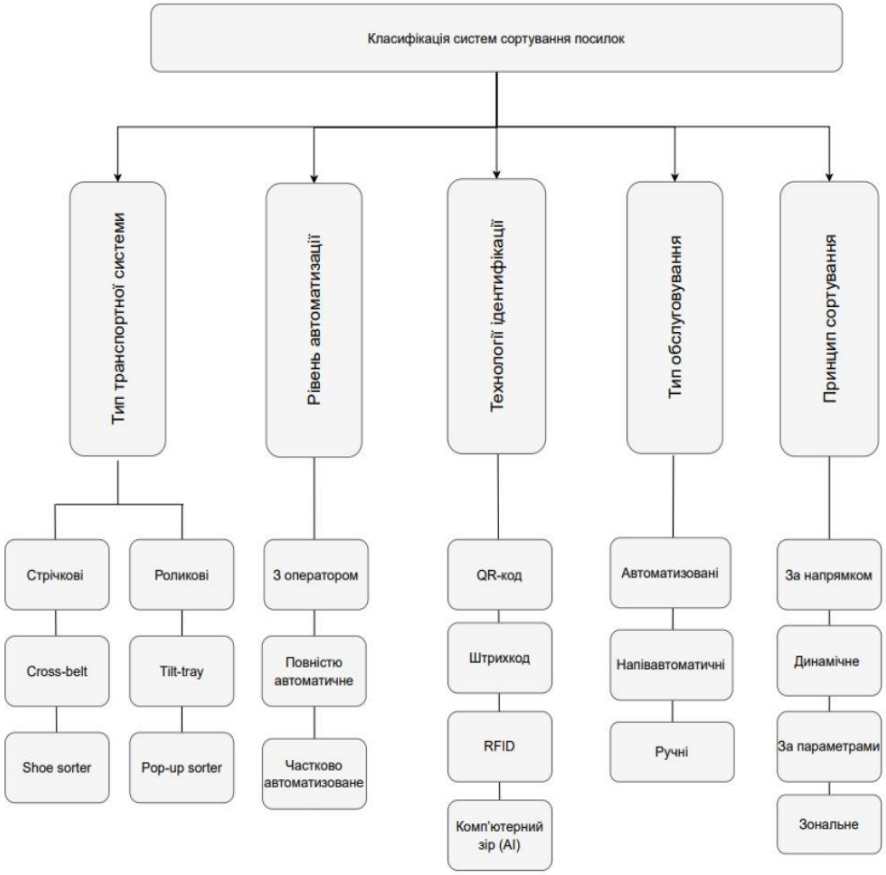
1. Dorner Conveyors — What is a Belt Conveyor? : офіційний сайт. URL: https://www.dornerconveyors.com/what-is-a-belt-conveyor
2. Ultimation — What is a Roller Conveyor? : офіційний сайт. URL: https://www.ultimationinc.com/what-is-a-roller-conveyor/
3. BEUMER Group — Cross-Belt Sorter : офіційний сайт. URL: https://www.beumergroup.com/knowledge/sortation-and-distribution-systems/cross-belt-sorter/
4. Mecalux — Tilt Tray Sorter Systems : офіційний сайт. URL: https://www.mecalux.com/warehouse-manual/automated-systems/tilt-tray-sorter
5. Honeywell — Sliding Shoe Sorter : офіційний сайт. URL: https://automation.honeywell.com/us/en/products/warehouse-automation/conveyor-systems/sortation/sliding-shoe-sorter
6. Bastian Solutions — Pop-Up Wheel Sorters : official website. URL: https://www.bastiansolutions.com/solutions/technology/conveyor-systems/sortation-conveyors/popup-wheel-sorter/
7. Smart Logitex — Інтелектуальні логістичні рішення : офіційний сайт. URL: https://smartlogitex.com/
8. Basler AG : official website. URL: https://www.baslerweb.com/
9. CubiQ — система вимірювання габаритів та ваги : офіційний сайт / Dimas-V. URL: https://cubiq.com.ua/
10. BeeSort Parcel Sorter System [Electronic resource] / BeeVision. URL: https://www.beevision.ai/BeeSort-Parcel-Sorter-System
11. ОБЕРНЕНА ЗАДАЧА КІНЕМАТИКИ НА ПРИКЛАДІ МАНІПУЛЯЦІЙНОГО РОБОТА З П’ЯТЬМА СТЕПЕНЯМИ СВОБОДИ І.С. ДМИТРІЄВА, Д.О. ЛЕВЧЕНКО *Національна металургійна академія України.*

12. Controller PFC200. WAGO I/O System 750 : Manual / WAGO GmbH & Co. KG. URL: https://www.wago.com/
13. uEye industrial cameras [Electronic resource] / IDS Imaging Development Systems GmbH. URL: https://en.ids-imaging.com/ueye-industrial-cameras.html
14. Omron Industrial Automation : official website. URL: https://ia.omron.com/
15. World K Series Induction Motor 5IK60GU-AWU : Technical Specifications / Oriental Motor. URL: https://www.orientalmotor.com/
16. Parallel Shaft Gearhead 5GN18KA : User Manual / Oriental Motor. URL: https://www.orientalmotor.com/
17. Контактор малогабаритний KMI-22510 (ККМ21-025-230-10-U) [Електронний ресурс] / IEK GROUP. URL: https://www.iek.ua/products/catalog/
18. Реле електротеплове серії PTI-1321 : техн. характеристики / IEK GROUP. URL: https://www.iek.ua/products/catalog/
19. CODESYS Group : official website. URL: https://www.codesys.com/
20. SOLIDWORKS 3D CAD : official website / Dassault Systèmes. URL: https://www.solidworks.com/
21. Python Language Reference, version 3.x [Electronic resource] / Python Software Foundation. URL: https://docs.python.org/3/
22. Graphical User Interfaces with Tk [Electronic resource] / Python Software Foundation. URL: https://docs.python.org/3/library/tk.html
23. OpenCV (Open Source Computer Vision Library) : documentation. URL: https://docs.opencv.org/
24. NumPy Documentation [Electronic resource] / NumPy Developers. URL: https://numpy.org/doc/
25. Pillow (PIL Fork) Documentation [Electronic resource]. URL: https://pillow.readthedocs.io/
26. time — Time access and conversions [Electronic resource] / Python Software Foundation. URL: https://docs.python.org/3/library/time.html

27. Os — Miscellaneous operating system interfaces [Electronic resource] / Python Software Foundation. URL: https://docs.python.org/3/library/os.html
28. PySerial’s documentation [Electronic resource] / Chris Liechti. URL: https://pythonhosted.org/pyserial/
29. Pyzbar 0.1.9 : library for reading barcodes [Electronic resource]. URL: https://pypi.org/project/pyzbar/

Додаток А

ДПБ.ПБ-21.1720.001.СХ

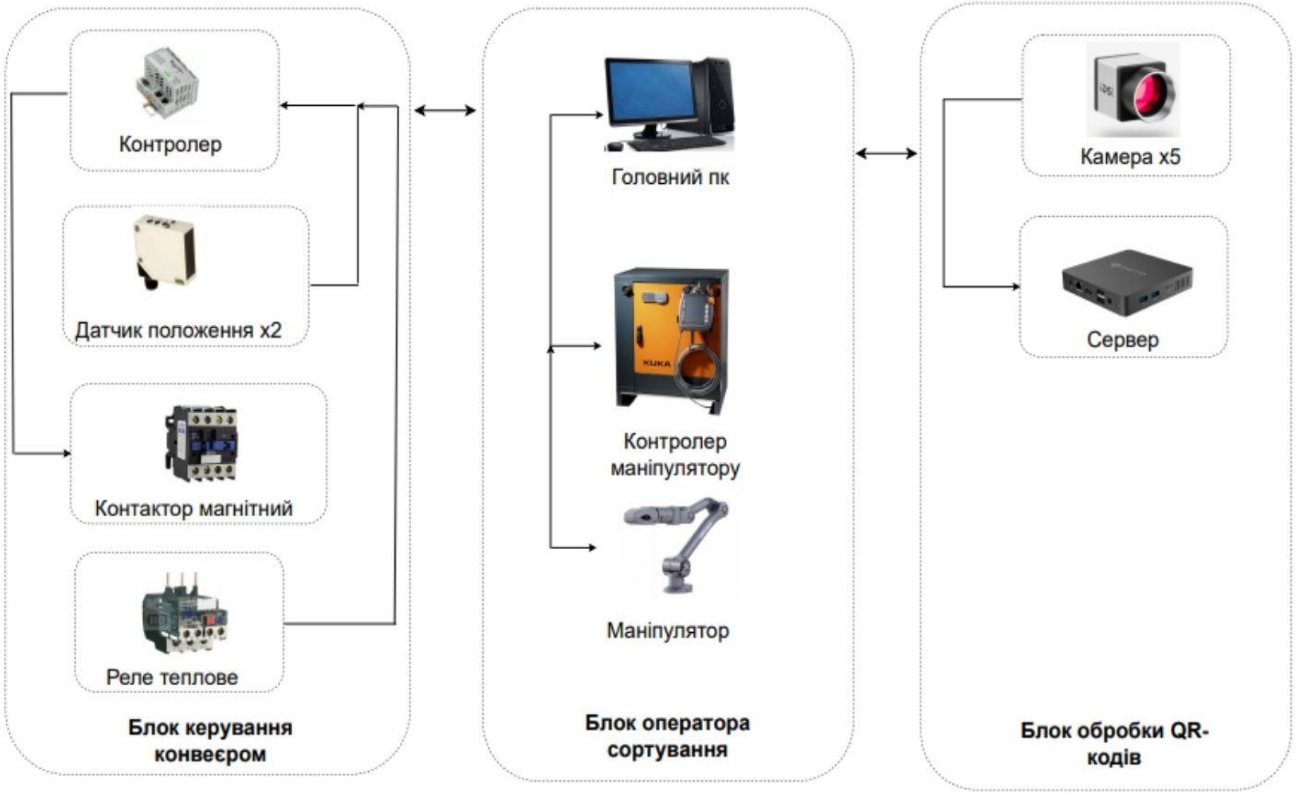


Изм.	Лист	№ докум.	Подп.	Дата	ДПБ.ПБ-21.1720.001.СХ		
Разраб.		Таран			Класифікація систем сортування	Масса	Масштаб
Пров.		Безуглий				Лист	Листов
Т. контр.							
Н. контр.					КПІ ім. Сікорського		
Утв.							

Копировал

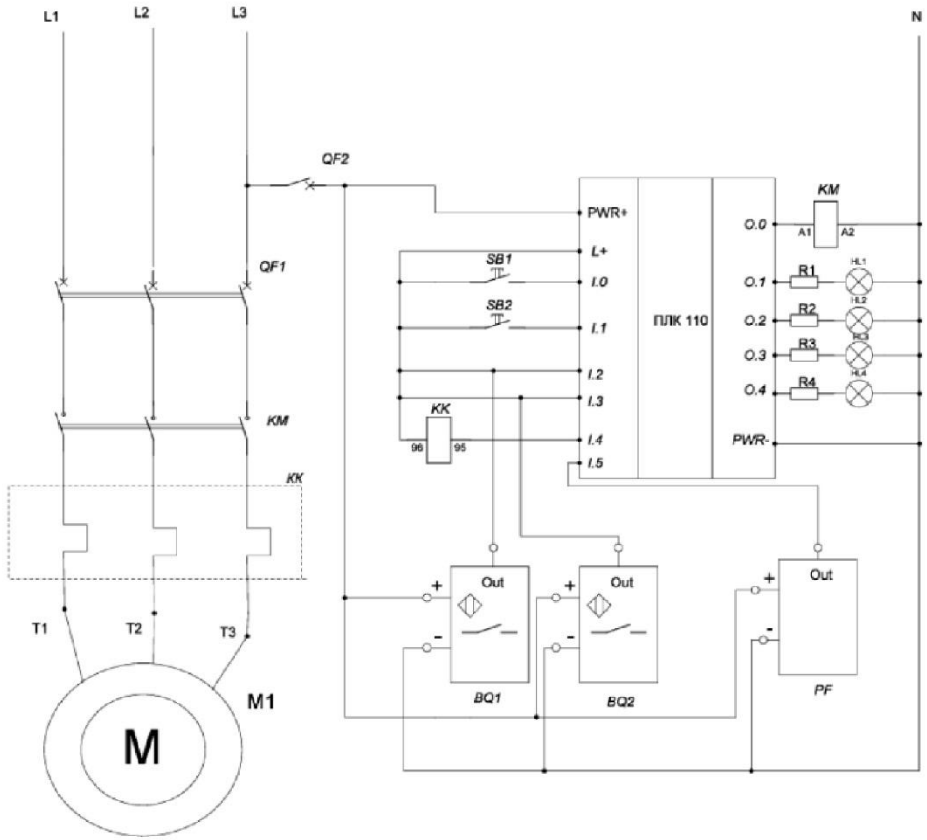
Формат А3

ДПБ.ПБ-21.1720.002.СХ



					ДПБ.ПБ-21.1720.002.СХ	
Изм	Лист	№ докум.	Подп.	Дата	Функціонально-структурна схема	Масса / Масштаб
Разраб.		Таран				
Пров.		Безуглий				Лист / Листов
Т. контр.						
Н. контр.						КПІ ім. Сікорського
Утв.						

Формат А3



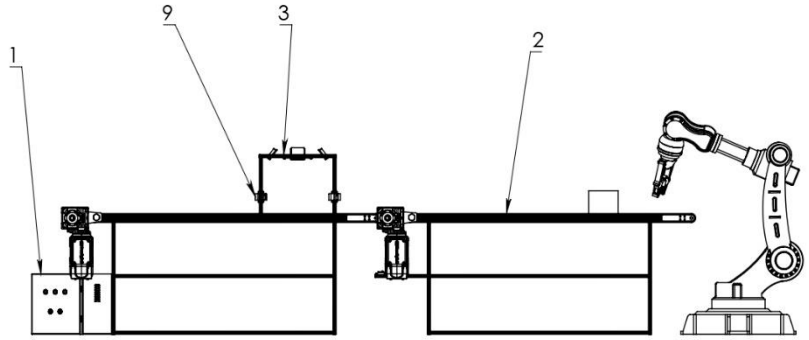
Позначення	Найменування	Кіль.	Примітка
QF1	Автоматичний вимикач ВА47-63, 3P, 25A	1	
QF2	Автоматичний вимикач ВА47-63, 1P, 6A	1	
KM	Контактор магнітний КМИ-32510, 25А, 220В	1	
KK	Реле теплове РТІ-1321, 12–18А	1	
SB1	Кнопка «Старт»	1	
SB2	Кнопка «Стоп»	1	
ПЛК	Програмований контролер WAGO PFC200	1	
BQ1	Датчик індуктивний 3-провідний, PNP	1	
BQ2	Датчик індуктивний 3-провідний, PNP	1	
PF	Цифровий індикатор швидкості 24V	1	
M1	Електродвигун асинхронний 5IK60GU-AWU,380В,	1	
HL1-4	Лампа сигнальна 24V	4	
R1-4	Резистор 1kΩ, 0.5W	4	

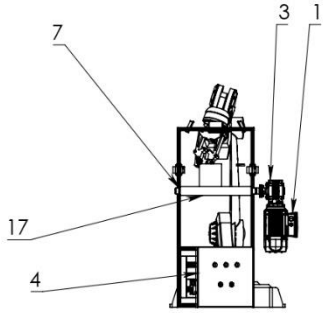
Изм.	Лист	№ докум.	Подп.	Дата	ДПБ.ПБ-21.1720.003.СХе
Разраб.		Таран			Електрична-принципова схема АССП
Пров.		Безуглий			Лит. / Масса / Масштаб
Т. контр.					Лист / Листов
Н. контр.					ПБ-21,КПІ
Утв.		Безуглий			

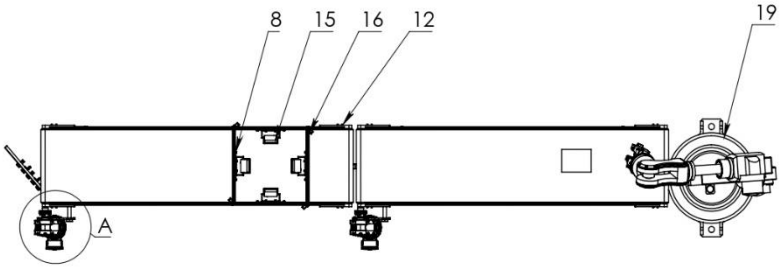
Копировал

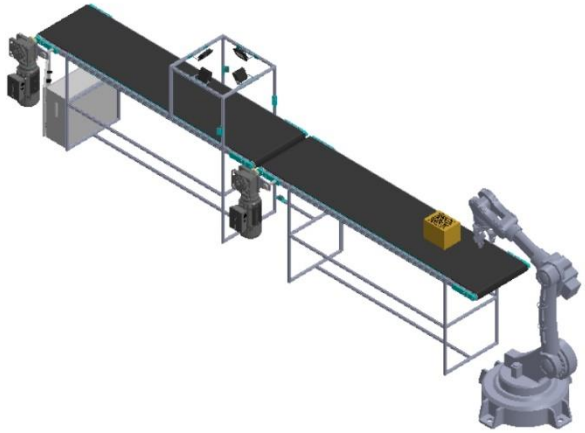
Формат А2

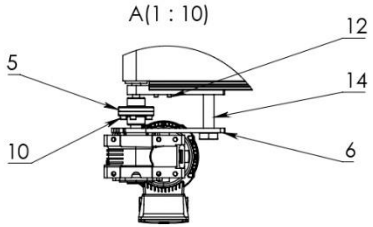
ДПБ.ПБ-21.1720.004.СК











					ДПБ.ПБ-21.1720.004.СК		
Изм.	Лист	№ докум.	Подп.	Дата	Автоматизована система сорутвання посиллок	Лит.	Масса	Масштаб
Разраб.		Таран						
Пров.		Безуглий				Лист	Листов	
Т. контр.								
Н. контр.						КПІ, ім Сікорського		
Утв.								

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Копировал | Формат А3

Файл: sorting_complex

Формат	Зона	Поз.	Позначення	Найменування	Кол.	Примітка
				Складальні одиниці		
		1		Двигун 5IK60GU-AWU	2	
		2		Стіл	2	
		3		Редуктор 5GN18KA	2	
		4		Блок електрощитка		
				Деталі		
		5		Муфта	2	
		6		Кронштейн валу	8	
		7		Вал	4	
		8		Кронштейн прожектору	4	
		9		Кронштейн камери	5	
				Стандартні вироби		
		10		ДСТУ ISO 4762 M8x20 - 20N	4	
		11		ДСТУ ISO 4762 M6x25 - 25N	8	
		12		ДСТУ ISO 4762 M5x12 - 12N	16	
		13		ДСТУ ISO 4762 M5x8 - 8N	44	
		14		ДСТУ ISO 4762 M24x90 -41WIN	2	
				Інші вироби		
		15		Прожектор 50 Вт IP65	4	
		16		Камера Basler Racer 2	5	

Перв. примен.

Справ. №

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Таран		
Пров.		Безуглий		
Т. контр.				
Н. контр.				
Утв.				

ДПБ.ПБ-21.1720.005.1

Специфікація АССП

Лит.	Масса	Масштаб
Лист	Листов	

КПІ, ім Сікорського

Копировал

Формат А4

Формат	Зона	Поз.	Позначення	Найменування	Кільк	Примітка
		17		Ремінь гумовий	1	
		18		Підшипники 16277	8	
		19		Маніпулятор DOBOT CR5	1	

Перв. примен.

Справ. №

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Таран		
Пров.		Безуглий		
Т. контр.				
Н. контр.				
Утв.				

ДПБ.ПБ-21.1720.005.2

Специфікація

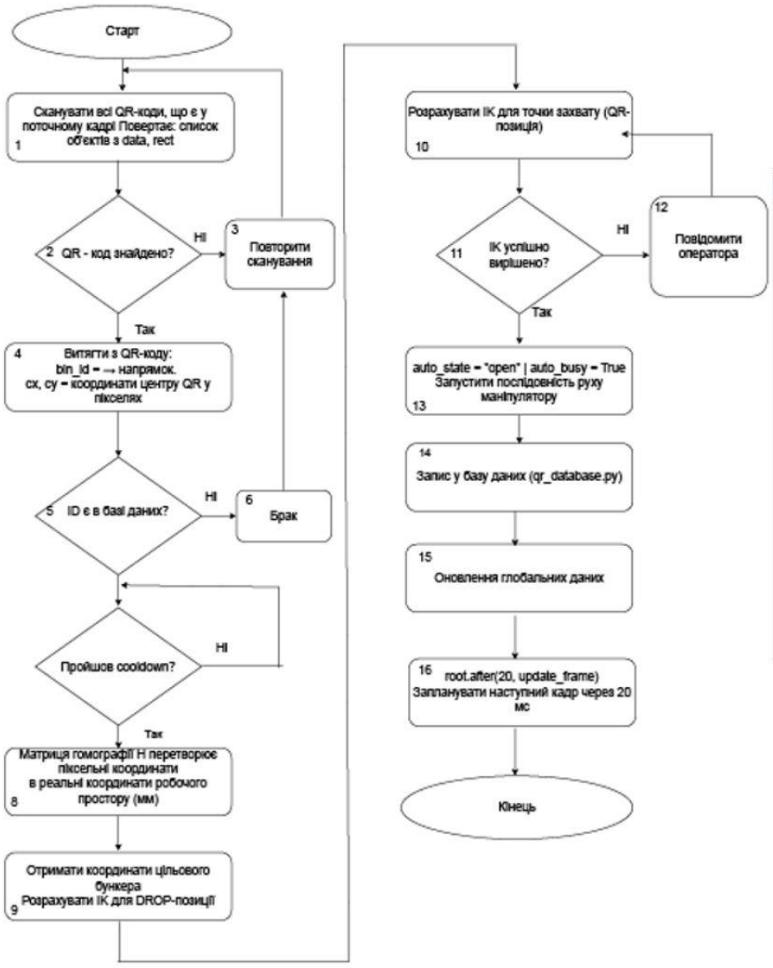
Лит.	Масса	Масштаб

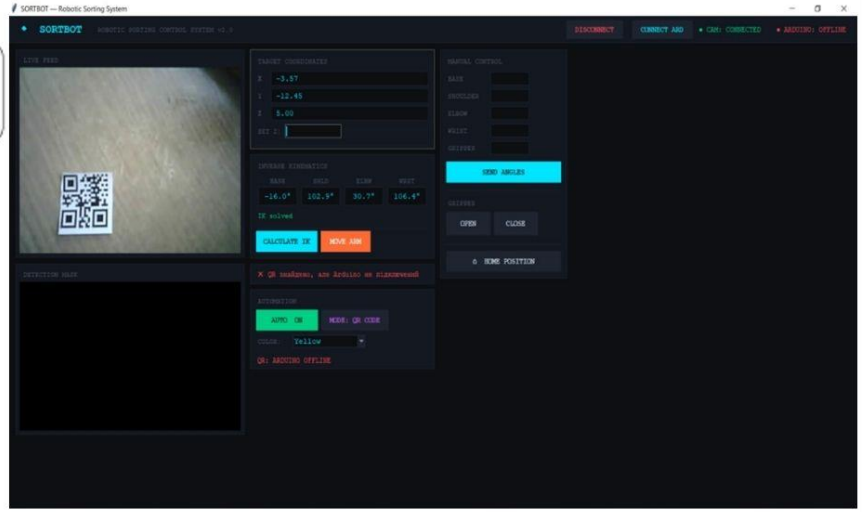
Лист	Листов

КПІ ім. Сікорського

Копировал

Формат А4

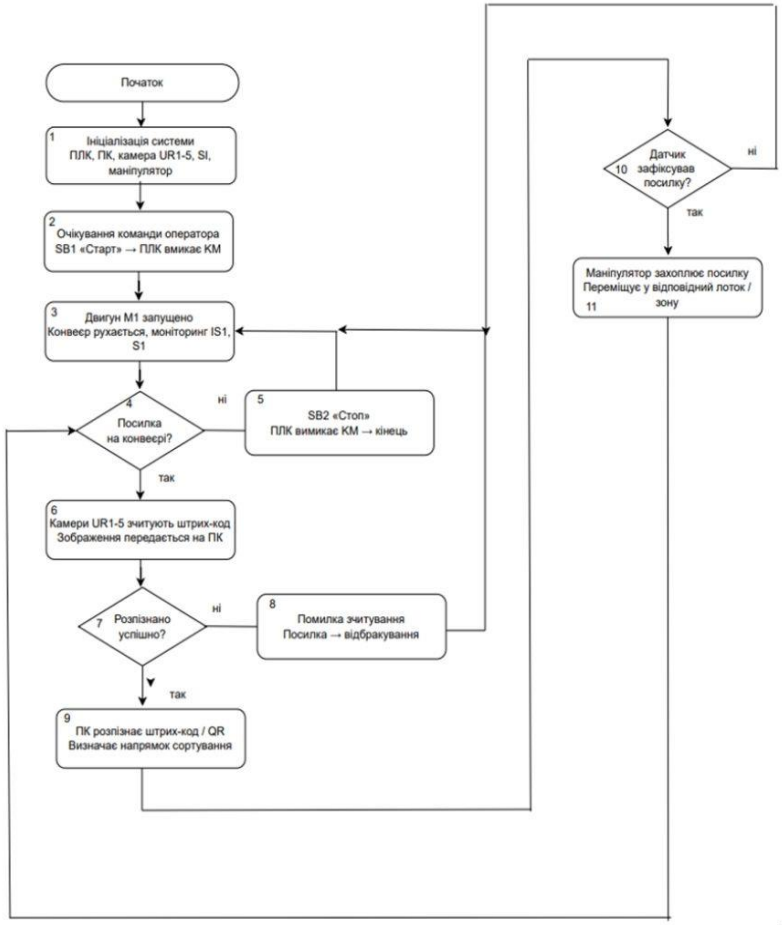




					ДПБ.ПБ-21.1720.006.СХа	
Изм.	Лист	№ докум.	Подп.	Дата		
Разраб.		Таран			Алгоритм програмного забезпечення	Масса / Масштаб
Пров.		Безуглий				Лист / Листов
Т. контр.						
Н. контр.					КПІ ім. Сікорського	
Утв.						

Формат А3

61

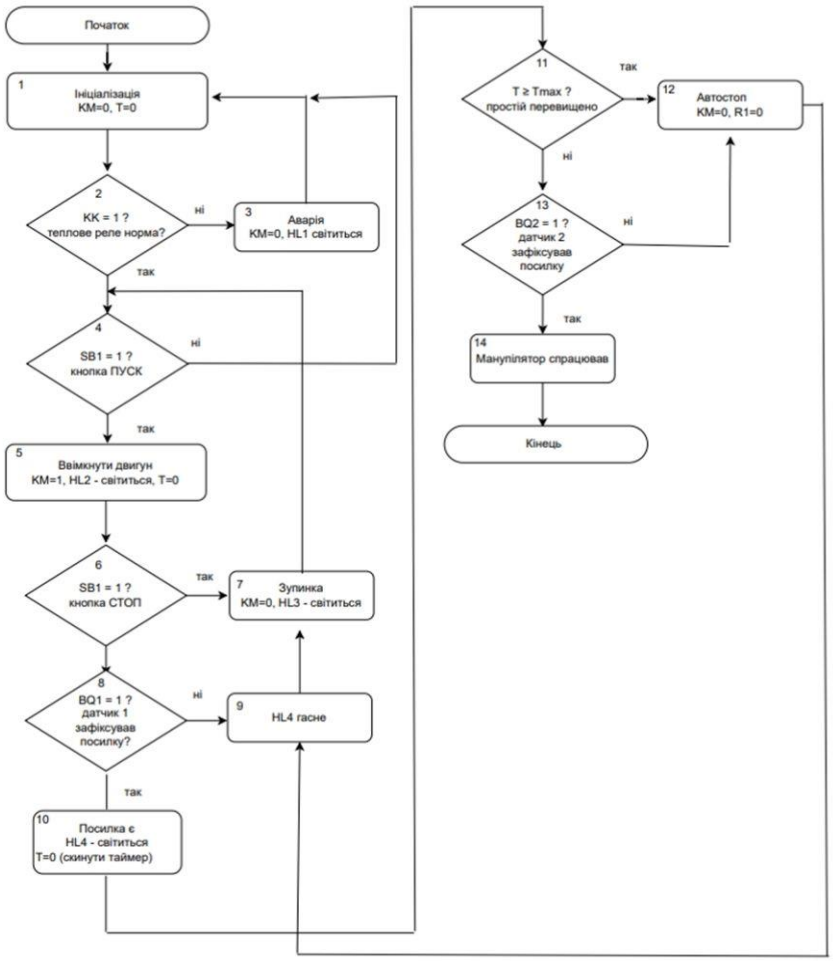


					ДПБ.ПБ-21.1720.007.СХа	
Изм.	Лист	№ докум.	Подп.	Дата	Блок схема роботи АССП	Масса / Масштаб
Разраб.		Таран				
Пров.		Безуглий				Лист / Листов
Т. контр.						
Н. контр.						КПІ ім. Сікорського
Утв.						

Копировал

Формат А3

ДПБ.ПБ-21.1720.008.СХа



					ДПБ.ПБ-21.1720.008.СХа	
Изм.	Лист	№ докум.	Подп.	Дата	Блок-схема роботи відповідно до електричної-принципової схеми	Масса / Масштаб
Разраб.		Таран				
Пров.		Безуглий				Лист / Листов
Т. контр.						
Н. контр.					КПІ ім. Сікорського	
Утв.		Безуглий				

Формат А4

Додаток Б

Програмний код gui

```
import tkinter as tk
from PIL import Image, ImageTk
import cv2
import numpy as np
from logic import calculate
import os
import serial
import time
from tkinter import ttk
import threading
from pyzbar import pyzbar
from qr_database import log_qr_scan

CAMERA_URL = url/stream'
H_FILE = 'homography.npy'

C_BG       = "#0d0f12"
C_PANEL    = "#141820"
C_BORDER   = "#1e2530"
C_ACCENT   = "#00e5ff"
C_ACCENT2  = "#ff6b35"
C_TEXT     = "#c8d6e5"
C_MUTED    = "#4a5568"
C_SUCCESS  = "#00d084"
C_ERROR    = "#ff4757"
C_WARNING  = "#ffa502"
C_QR       = "#bf5af2"

FONT_MONO   = ("Courier New", 10)
FONT_MONO_S = ("Courier New", 9)
FONT_LABEL  = ("Courier New", 8)
```

```
FONT_TITLE  = ("Courier New", 11, "bold")
FONT_HEADER = ("Courier New", 9, "bold")

auto_state = "idle"
auto_state_time = 0
auto_busy = False
sorting_mode = "color"
kernel = np.ones((5, 5), np.uint8)

base_angle_val = None
theta1_val = None
theta2_val = None
theta3_val = None
gripper_state = 1

drop_base_val = None
drop_theta1_val = None
drop_theta2_val = None
drop_theta3_val = None
qr_drop_pending = False

cap = None
camera_available = False
_camera_retry_count = 0
_CAMERA_RETRY_DELAYS = [1, 2, 5, 10, 30]

auto_detect = False
last_center = None
CENTER_ALPHA = 0.3
MIN_CONTOUR_AREA = 500
AUTO_INTERVAL = 0.5
last_auto_time = 0
```

```
last_gray_frame = None
FRAME_DIFF_THRESHOLD = 5_000
AUTO_COOLDOWN = 5.0
last_pick_time = 0

COLOR_RANGES = {
   "Yellow": ((20, 100, 100), (30, 255, 255)),
   "Red":    ((0, 120, 70),   (10, 255, 255)),
   "Green":  ((35, 80, 80),   (85, 255, 255)),
   "Blue":   ((100, 150, 50), (140, 255, 255)),
   "Black":  ((0, 0, 0),      (180, 255, 50)),
}

BIN_POSITIONS = {
   "1": (100.0,  50.0, 10.0),
   "2": (  0.0, 130.0, 10.0),
   "3": (-100.0, 50.0, 10.0),
}

qr_detected_data = None
qr_target_bin = None
last_qr_time = 0.0
QR_COOLDOWN = 5.0

H = np.load(H_FILE) if os.path.exists(H_FILE) else None

def pixel_to_world(x, y):
   if H is None:
      raise RuntimeError("Homography matrix not loaded — run cam.py first")
   pt = np.array([[[x, y]]], dtype=np.float32)
   world = cv2.perspectiveTransform(pt, H)
   return world[0][0]
```

```
try:
    arduino = serial.Serial('COM3', 9600, timeout=1)
    time.sleep(2)
except Exception as e:
    print(f"Unable to connect to Arduino: {e}")
    arduino = None

root = tk.Tk()
root.title("SORTBOT — Robotic Sorting System")
root.configure(bg=C_BG)
root.resizable(True, True)
try:
    root.state("zoomed")
except Exception:
    root.attributes("-zoomed", True)
root.minsize(960, 680)

selected_color = tk.StringVar(value="Yellow")

def make_panel(parent, **kwargs):
    return tk.Frame(parent, bg=C_PANEL, highlightbackground=C_BORDER,
                    highlightthickness=1, **kwargs)

def section_label(parent, text):
    tk.Label(parent, text=text, bg=C_PANEL, fg=C_MUTED,
             font=FONT_LABEL).pack(anchor="w", padx=10, pady=(8, 2))

def divider(parent):
    tk.Frame(parent, bg=C_BORDER, height=1).pack(fill="x", padx=10, pady=4)

def value_label(parent, width=14):
```

```
    lbl = tk.Label(parent, text="---", bg=C_PANEL, fg=C_ACCENT,
                   font=FONT_MONO, width=width, anchor="w")
    return lbl

def flat_btn(parent, text, command, color=C_ACCENT, text_color=C_BG, **kwargs):
    return tk.Button(
        parent, text=text, command=command,
        bg=color, fg=text_color, activebackground=color,
        font=FONT_HEADER, relief="flat", cursor="hand2",
        bd=0, padx=8, pady=6, **kwargs
    )

def ghost_btn(parent, text, command, color=C_BORDER, text_color=C_TEXT, **kwargs):
    return tk.Button(
        parent, text=text, command=command,
        bg=color, fg=text_color, activebackground=C_ACCENT,
        font=FONT_HEADER, relief="flat", cursor="hand2",
        bd=0, padx=8, pady=5, **kwargs
    )

header = tk.Frame(root, bg=C_PANEL, height=44, highlightbackground=C_BORDER,
                  highlightthickness=1)
header.pack(fill="x", side="top")
header.pack_propagate(False)

tk.Label(header, text="◈  SORTBOT", bg=C_PANEL, fg=C_ACCENT,
         font=("Courier New", 14, "bold")).pack(side="left", padx=18, pady=8)
tk.Label(header, text="ROBOTIC SORTING CONTROL SYSTEM v2.0",
         bg=C_PANEL, fg=C_MUTED, font=FONT_LABEL).pack(side="left", padx=4, pady=8)

conn_dot = tk.Label(header,
                    text="● ARDUINO: " + ("CONNECTED" if arduino else "OFFLINE"),
```

```
            bg=C_PANEL,
            fg=C_SUCCESS if arduino else C_ERROR,
            font=FONT_MONO_S)
conn_dot.pack(side="right", padx=18)

cam_dot = tk.Label(header, text="● CAM: OFFLINE", bg=C_PANEL,
           fg=C_MUTED, font=FONT_MONO_S)
cam_dot.pack(side="right", padx=4)

def _do_connect_camera():
  global cap, camera_available, _camera_retry_count
  root.after(0, lambda: cam_dot.config(text="● CAM: CONNECTING…", fg=C_WARNING))
  root.after(0, lambda: connect_btn.config(text="CONNECTING…", state="disabled",
fg=C_MUTED))

  tmp = cv2.VideoCapture(CAMERA_URL)
  tmp.set(cv2.CAP_PROP_BUFFERSIZE, 1)

  if tmp.isOpened():
    cap = tmp
    camera_available = True
    _camera_retry_count = 0
    root.after(0, lambda: cam_dot.config(text="● CAM: CONNECTED", fg=C_SUCCESS))
    root.after(0, lambda: connect_btn.config(text="DISCONNECT", state="normal",
                          fg=C_ERROR, bg=C_BORDER))
    root.after(0, update_frame)
  else:
    tmp.release()
    root.after(0, lambda: cam_dot.config(text="● CAM: FAILED", fg=C_ERROR))
    root.after(0, lambda: connect_btn.config(text="CONNECT CAM", state="normal",
                          fg=C_ACCENT, bg=C_BORDER))
```

```
def toggle_camera_connection():
    global cap, camera_available
    if camera_available:
        camera_available = False
        if cap is not None:
            cap.release()
            cap = None
        cam_dot.config(text="● CAM: OFFLINE", fg=C_MUTED)
        connect_btn.config(text="CONNECT CAM", fg=C_ACCENT, bg=C_BORDER)
    else:
        t = threading.Thread(target=_do_connect_camera, daemon=True)
        t.start()

def _do_connect_arduino():
    global arduino
    root.after(0, lambda: arduino_btn.config(text="CONNECTING…", state="disabled",
fg=C_MUTED))
    root.after(0, lambda: conn_dot.config(text="● ARDUINO: CONNECTING…", fg=C_WARNING))
    try:
        ser = serial.Serial('COM3', 9600, timeout=1)
        time.sleep(2)
        arduino = ser
        root.after(0, lambda: conn_dot.config(text="● ARDUINO: CONNECTED", fg=C_SUCCESS))
        root.after(0, lambda: arduino_btn.config(
            text="DISCONNECT ARD", state="normal", fg=C_ERROR, bg=C_BORDER))
    except Exception as e:
        print(f"Arduino connect failed: {e}")
        root.after(0, lambda: conn_dot.config(text="● ARDUINO: FAILED", fg=C_ERROR))
        root.after(0, lambda: arduino_btn.config(
            text="CONNECT ARD", state="normal", fg=C_ACCENT, bg=C_BORDER))

def toggle_arduino_connection():
```

```
    global arduino
    if arduino is not None:
        try:
            arduino.close()
        except Exception:
            pass
        arduino = None
        conn_dot.config(text="● ARDUINO: OFFLINE", fg=C_MUTED)
        arduino_btn.config(text="CONNECT ARD", fg=C_ACCENT, bg=C_BORDER)
    else:
        t = threading.Thread(target=_do_connect_arduino, daemon=True)
        t.start()

arduino_btn = tk.Button(
    header, text="CONNECT ARD" if arduino is None else "DISCONNECT ARD",
    command=toggle_arduino_connection,
    bg=C_BORDER, fg=C_ACCENT if arduino is None else C_ERROR,
    activebackground=C_BORDER,
    font=FONT_HEADER, relief="flat", cursor="hand2",
    bd=0, padx=14, pady=6
)
arduino_btn.pack(side="right", padx=4, pady=6)

connect_btn = tk.Button(
    header, text="CONNECT CAM",
    command=toggle_camera_connection,
    bg=C_BORDER, fg=C_ACCENT, activebackground=C_BORDER,
    font=FONT_HEADER, relief="flat", cursor="hand2",
    bd=0, padx=14, pady=6
)
connect_btn.pack(side="right", padx=12, pady=6)
```

```
body = tk.Frame(root, bg=C_BG)
body.pack(fill="both", expand=True, padx=12, pady=10)

body.columnconfigure(0, minsize=420, weight=0)
body.columnconfigure(1, minsize=260, weight=0)
body.columnconfigure(2, minsize=230, weight=0)
body.columnconfigure(3, weight=1)
body.rowconfigure(0, weight=1)

left_col = tk.Frame(body, bg=C_BG)
left_col.grid(row=0, column=0, sticky="nsew", padx=(0, 10))

cam_panel = make_panel(left_col)
cam_panel.pack(fill="x")

tk.Label(cam_panel, text="LIVE FEED", bg=C_PANEL, fg=C_MUTED,
        font=FONT_LABEL).pack(anchor="w", padx=10, pady=(6, 2))

cam_canvas = tk.Frame(cam_panel, bg="#000000", width=400, height=300)
cam_canvas.pack(padx=6, pady=(0, 6))
cam_canvas.pack_propagate(False)

camera_label = tk.Label(cam_canvas, bg="#000000")
camera_label.place(x=0, y=0, width=400, height=300)

mask_panel = make_panel(left_col)
mask_panel.pack(fill="x", pady=(10, 0))

tk.Label(mask_panel, text="DETECTION MASK", bg=C_PANEL, fg=C_MUTED,
        font=FONT_LABEL).pack(anchor="w", padx=10, pady=(6, 2))

mask_canvas = tk.Frame(mask_panel, bg="#000000", width=400, height=240)
```

```
mask_canvas.pack(padx=6, pady=(0, 6))
mask_canvas.pack_propagate(False)

mask_label = tk.Label(mask_canvas, bg="#000000")
mask_label.place(x=0, y=0, width=400, height=240)

mid_col = tk.Frame(body, bg=C_BG)
mid_col.grid(row=0, column=1, sticky="nsew", padx=(0, 10))

coord_panel = make_panel(mid_col)
coord_panel.pack(fill="x")

tk.Label(coord_panel, text="TARGET COORDINATES", bg=C_PANEL, fg=C_MUTED,
        font=FONT_LABEL).pack(anchor="w", padx=10, pady=(8, 4))

for axis, var_name in [("X", "x_label"), ("Y", "y_label"), ("Z", "z_label")]:
    row = tk.Frame(coord_panel, bg=C_PANEL)
    row.pack(fill="x", padx=10, pady=2)
    tk.Label(row, text=axis, bg=C_PANEL, fg=C_MUTED,
             font=FONT_HEADER, width=3, anchor="w").pack(side="left")
    lbl = tk.Label(row, text="---", bg=C_BG, fg=C_ACCENT,
                   font=FONT_MONO, width=14, anchor="w",
                   highlightbackground=C_BORDER, highlightthickness=1, padx=6)
    lbl.pack(side="left", fill="x", expand=True)
    if var_name == "x_label":
        x_label = lbl
    elif var_name == "y_label":
        y_label = lbl
    else:
        z_label = lbl

z_row = tk.Frame(coord_panel, bg=C_PANEL)
```

```
z_row.pack(fill="x", padx=10, pady=(4, 8))
tk.Label(z_row, text="SET Z:", bg=C_PANEL, fg=C_MUTED,
        font=FONT_LABEL, width=6, anchor="w").pack(side="left")
entry_z = tk.Entry(z_row, bg=C_BG, fg=C_ACCENT, insertbackground=C_ACCENT,
                font=FONT_MONO, relief="flat", highlightbackground=C_BORDER,
                highlightthickness=1, width=12)
entry_z.pack(side="left", padx=4)

def on_z_enter(event):
    v = entry_z.get().strip()
    if v:
        entry_z.delete(0, tk.END)
        z_label.config(text=v)

entry_z.bind("<Return>", on_z_enter)

divider(coord_panel)

ik_panel = make_panel(mid_col)
ik_panel.pack(fill="x", pady=(8, 0))

tk.Label(ik_panel, text="INVERSE KINEMATICS", bg=C_PANEL, fg=C_MUTED,
        font=FONT_LABEL).pack(anchor="w", padx=10, pady=(8, 4))

ik_grid = tk.Frame(ik_panel, bg=C_PANEL)
ik_grid.pack(fill="x", padx=10, pady=(0, 6))

ik_labels = {}
for i, name in enumerate(["BASE", "SHLD", "ELBW", "WRST"]):
    col_f = tk.Frame(ik_grid, bg=C_PANEL)
    col_f.grid(row=0, column=i, padx=4)
    tk.Label(col_f, text=name, bg=C_PANEL, fg=C_MUTED,
```

```
            font=FONT_LABEL).pack()
    lbl = tk.Label(col_f, text="---°", bg=C_BG, fg=C_ACCENT,
                   font=FONT_MONO, width=7, anchor="center",
                   highlightbackground=C_BORDER, highlightthickness=1, padx=4, pady=4)
    lbl.pack()
    ik_labels[name] = lbl

result_label = tk.Label(ik_panel, text="", bg=C_PANEL, fg=C_MUTED,
                        font=FONT_LABEL, wraplength=230, justify="left")
result_label.pack(anchor="w", padx=10, pady=(0, 6))

divider(ik_panel)

btn_row = tk.Frame(ik_panel, bg=C_PANEL)
btn_row.pack(fill="x", padx=10, pady=(0, 10))

def on_calculate():
    global base_angle_val, theta1_val, theta2_val, theta3_val
    try:
        x = float(x_label["text"])
        y = float(y_label["text"])
        z = float(z_label["text"])
        result = calculate(x, y, z)
        base_angle_val, theta1_val, theta2_val, theta3_val = result
        for name, val in zip(["BASE", "SHLD", "ELBW", "WRST"], result):
            ik_labels[name].config(text=f"{val:.1f}°")
        result_label.config(text="IK solved", fg=C_SUCCESS)
    except ValueError:
        result_label.config(text="Enter valid coordinates", fg=C_ERROR)
    except Exception as e:
        result_label.config(text=f"Error: {e}", fg=C_ERROR)
```

```
def on_move():
    try:
        if None in (base_angle_val, theta1_val, theta2_val, theta3_val):
            set_status("Calculate IK first", "error")
            return
        if arduino is None:
            set_status("Arduino not connected", "error")
            return
        arduino.write(f"B {base_angle_val:.2f}\n".encode())
        arduino.write(f"S {theta1_val:.2f}\n".encode())
        arduino.write(f"E {theta2_val:.2f}\n".encode())
        arduino.write(f"W {theta3_val:.2f}\n".encode())
        arduino.write(f"G {gripper_state}\n".encode())
        set_status("Angles sent", "ok")
    except Exception as e:
        set_status(f"Serial error: {e}", "error")

flat_btn(btn_row, "CALCULATE IK", on_calculate, color=C_ACCENT, width=13).pack(side="left",
padx=(0, 6))
flat_btn(btn_row, "MOVE ARM", on_move, color=C_ACCENT2, text_color="#fff",
width=10).pack(side="left")

status_panel = make_panel(mid_col)
status_panel.pack(fill="x", pady=(8, 0))

status_label = tk.Label(status_panel, text="● READY", bg=C_PANEL, fg=C_SUCCESS,
                        font=FONT_MONO_S, anchor="w", padx=10, pady=6)
status_label.pack(fill="x")

def set_status(msg, level="ok"):
    colors = {"ok": C_SUCCESS, "error": C_ERROR, "warn": C_WARNING, "info": C_ACCENT}
    prefix = {"ok": "●", "error": "✕", "warn": "!", "info": "◈"}
```

```
    color = colors.get(level, C_TEXT)
    status_label.config(text=f"{prefix.get(level,'●')} {msg}", fg=color)

auto_panel = make_panel(mid_col)
auto_panel.pack(fill="x", pady=(8, 0))

tk.Label(auto_panel, text="AUTOMATION", bg=C_PANEL, fg=C_MUTED,
         font=FONT_LABEL).pack(anchor="w", padx=10, pady=(8, 4))

auto_row = tk.Frame(auto_panel, bg=C_PANEL)
auto_row.pack(fill="x", padx=10, pady=(0, 6))

toggle_btn = tk.Button(
    auto_row, text="AUTO  OFF",
    bg=C_BORDER, fg=C_MUTED,
    activebackground=C_SUCCESS, font=FONT_HEADER,
    relief="flat", cursor="hand2", bd=0, padx=12, pady=6, width=12
)
toggle_btn.pack(side="left", padx=(0, 6))

sorting_btn = tk.Button(
    auto_row, text="MODE: COLOR",
    bg=C_BORDER, fg=C_ACCENT,
    activebackground=C_BORDER, font=FONT_HEADER,
    relief="flat", cursor="hand2", bd=0, padx=10, pady=6, width=14
)
sorting_btn.pack(side="left")

def toggle_auto():
    global auto_detect
    auto_detect = not auto_detect
    if auto_detect:
```

```
        toggle_btn.config(text="AUTO  ON", bg=C_SUCCESS, fg=C_BG)
    else:
        toggle_btn.config(text="AUTO  OFF", bg=C_BORDER, fg=C_MUTED)

toggle_btn.config(command=toggle_auto)

def toggle_sorting_mode():
    global sorting_mode
    if sorting_mode == "color":
        sorting_mode = "second"
        sorting_btn.config(text="MODE: QR CODE", fg=C_QR)
        color_combo.config(state="disabled")
        qr_info_label.config(text="QR: scanning disabled")
    else:
        sorting_mode = "color"
        sorting_btn.config(text="MODE: COLOR", fg=C_ACCENT)
        color_combo.config(state="readonly")
        qr_info_label.config(text="QR: ---")

sorting_btn.config(command=toggle_sorting_mode)

color_row = tk.Frame(auto_panel, bg=C_PANEL)
color_row.pack(fill="x", padx=10, pady=(0, 6))
tk.Label(color_row, text="COLOR:", bg=C_PANEL, fg=C_MUTED,
         font=FONT_LABEL, width=8, anchor="w").pack(side="left")

style = ttk.Style()
style.theme_use("clam")
style.configure("Dark.TCombobox",
                fieldbackground=C_BG, background=C_BORDER,
                foreground=C_ACCENT, selectbackground=C_ACCENT,
                selectforeground=C_BG, arrowcolor=C_ACCENT,
```

```
            bordercolor=C_BORDER, lightcolor=C_BORDER, darkcolor=C_BORDER)

color_combo = ttk.Combobox(
    color_row, textvariable=selected_color,
    values=list(COLOR_RANGES.keys()),
    state="readonly", width=14,
    style="Dark.TCombobox", font=FONT_MONO
)
color_combo.pack(side="left", padx=4)

qr_info_label = tk.Label(auto_panel, text="QR: ---", bg=C_PANEL, fg=C_QR,
                         font=FONT_MONO_S, anchor="w", padx=10, pady=4)
qr_info_label.pack(fill="x")

right_col = tk.Frame(body, bg=C_BG)
right_col.grid(row=0, column=2, sticky="nsew")

manual_panel = make_panel(right_col)
manual_panel.pack(fill="x")

tk.Label(manual_panel, text="MANUAL CONTROL", bg=C_PANEL, fg=C_MUTED,
         font=FONT_LABEL).pack(anchor="w", padx=10, pady=(8, 4))

entries = {}
for name, label in [("B","BASE"), ("S","SHOULDER"), ("E","ELBOW"),
                    ("W","WRIST"), ("G","GRIPPER")]:
    row = tk.Frame(manual_panel, bg=C_PANEL)
    row.pack(fill="x", padx=10, pady=3)
    tk.Label(row, text=label, bg=C_PANEL, fg=C_MUTED,
             font=FONT_LABEL, width=10, anchor="w").pack(side="left")
    ent = tk.Entry(row, bg=C_BG, fg=C_ACCENT, insertbackground=C_ACCENT,
                   font=FONT_MONO, relief="flat",
```

```
            highlightbackground=C_BORDER, highlightthickness=1,
            width=8)
    ent.pack(side="left", padx=4)
    entries[name] = ent

def send_manual_angles():
    try:
        if arduino is None:
            set_status("Arduino not connected", "error")
            return
        sent = False
        for name in ["B", "S", "E", "W", "G"]:
            val = entries[name].get().strip()
            if val:
                if name == "G":
                    if val not in ("0", "1"):
                        set_status("G must be 0 or 1", "error")
                        return
                    arduino.write(f"{name} {val}\n".encode())
                else:
                    arduino.write(f"{name} {float(val):.2f}\n".encode())
                sent = True
        if sent:
            set_status("Manual angles sent", "ok")
        else:
            set_status("No angles to send", "warn")
    except ValueError:
        set_status("Enter valid numbers", "error")
    except Exception as e:
        set_status(f"Serial error: {e}", "error")

flat_btn(manual_panel, "SEND ANGLES", send_manual_angles,
```

```
        color=C_ACCENT, width=18).pack(padx=10, pady=8, fill="x")

divider(manual_panel)

tk.Label(manual_panel, text="GRIPPER", bg=C_PANEL, fg=C_MUTED,
        font=FONT_LABEL).pack(anchor="w", padx=10, pady=(6, 4))

grip_row = tk.Frame(manual_panel, bg=C_PANEL)
grip_row.pack(fill="x", padx=10, pady=(0, 10))

def open_gripper():
    global gripper_state
    gripper_state = 1
    try:
        if arduino is None:
            set_status("Arduino not connected", "error"); return
        arduino.write(b"G 1\n")
        set_status("Gripper open", "ok")
        btn_open.config(bg=C_SUCCESS, fg=C_BG)
        btn_close.config(bg=C_BORDER, fg=C_TEXT)
    except Exception as e:
        set_status(f"Serial error: {e}", "error")

def close_gripper():
    global gripper_state
    gripper_state = 0
    try:
        if arduino is None:
            set_status("Arduino not connected", "error"); return
        arduino.write(b"G 0\n")
        set_status("Gripper closed", "ok")
        btn_close.config(bg=C_ACCENT2, fg="#fff")
```

```
        btn_open.config(bg=C_BORDER, fg=C_TEXT)
    except Exception as e:
        set_status(f"Serial error: {e}", "error")

btn_open = tk.Button(grip_row, text="OPEN", command=open_gripper,
                     bg=C_BORDER, fg=C_TEXT, activebackground=C_SUCCESS,
                     font=FONT_HEADER, relief="flat", cursor="hand2",
                     bd=0, padx=10, pady=8, width=8)
btn_open.pack(side="left", padx=(0, 6))

btn_close = tk.Button(grip_row, text="CLOSE", command=close_gripper,
                      bg=C_BORDER, fg=C_TEXT, activebackground=C_ACCENT2,
                      font=FONT_HEADER, relief="flat", cursor="hand2",
                      bd=0, padx=10, pady=8, width=8)
btn_close.pack(side="left")

divider(manual_panel)

def send_home():
    try:
        if arduino is None:
            set_status("Arduino not connected", "error"); return
        for cmd in [b"B 0\n", b"S 0\n", b"E 0\n", b"W 0\n"]:
            arduino.write(cmd)
        set_status("Arm returned home", "ok")
    except Exception as e:
        set_status(f"Serial error: {e}", "error")

flat_btn(manual_panel, "⌂  HOME POSITION", send_home,
         color=C_BORDER, text_color=C_TEXT, width=18).pack(padx=10, pady=(6, 10), fill="x")

def _reset_qr_state():
```

```
    global qr_drop_pending, qr_target_bin, qr_detected_data
    global drop_base_val, drop_theta1_val, drop_theta2_val, drop_theta3_val
    qr_drop_pending  = False
    qr_target_bin    = None
    qr_detected_data = None
    drop_base_val = drop_theta1_val = drop_theta2_val = drop_theta3_val = None

def auto_calculate_only():
    global base_angle_val, theta1_val, theta2_val, theta3_val
    try:
        x = float(x_label["text"])
        y = float(y_label["text"])
        z = float(z_label["text"])
        base_angle_val, theta1_val, theta2_val, theta3_val = calculate(x, y, z)
        for name, val in zip(["BASE","SHLD","ELBW","WRST"],
                             [base_angle_val, theta1_val, theta2_val, theta3_val]):
            ik_labels[name].config(text=f"{val:.1f}°")
        return True
    except Exception as e:
        result_label.config(text=f"Auto IK error: {e}", fg=C_ERROR)
        return False

def auto_step():
    global auto_state, auto_state_time, auto_busy
    global last_pick_time, last_center, qr_drop_pending
    if arduino is None:
        set_status("Arduino не підключений — цикл скинуто", "error")
        auto_busy  = False
        auto_state = "idle"
        _reset_qr_state()
        return
    now = time.time()
```

```
if auto_state == "open":
    open_gripper()
    auto_state = "move"
    auto_state_time = now
elif auto_state == "move" and now - auto_state_time > 1.0:
    arduino.write(f"B {base_angle_val:.2f}\n".encode())
    arduino.write(f"S {theta1_val:.2f}\n".encode())
    arduino.write(f"E {theta2_val:.2f}\n".encode())
    arduino.write(f"W {theta3_val:.2f}\n".encode())
    auto_state = "close"
    auto_state_time = now
elif auto_state == "close" and now - auto_state_time > 2.0:
    close_gripper()
    auto_state = "drop" if qr_drop_pending else "home"
    auto_state_time = now
elif auto_state == "drop" and now - auto_state_time > 2.0:
    arduino.write(f"B {drop_base_val:.2f}\n".encode())
    arduino.write(f"S {drop_theta1_val:.2f}\n".encode())
    arduino.write(f"E {drop_theta2_val:.2f}\n".encode())
    arduino.write(f"W {drop_theta3_val:.2f}\n".encode())
    set_status("Moving to bin…", "info")
    auto_state = "release"
    auto_state_time = now
elif auto_state == "home" and now - auto_state_time > 2.0:
    send_home()
    auto_state = "release"
    auto_state_time = now
elif auto_state == "release" and now - auto_state_time > 1.0:
    open_gripper()
    if qr_drop_pending:
        send_home()
    auto_state = "idle"
```

```
        auto_busy = False
        last_pick_time = time.time()
        last_center = None
        _reset_qr_state()


def detect_qr(frame):
    decoded = pyzbar.decode(frame)
    for obj in decoded:
        data = obj.data.decode("utf-8").strip()
        if data in BIN_POSITIONS:
            rect = obj.rect
            cx = rect.left + rect.width  // 2
            cy = rect.top  + rect.height // 2
            return data, cx, cy
    return None, None, None

def find_object_center(mask):
    global last_center
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if not contours:
        last_center = None; return None
    largest = max(contours, key=cv2.contourArea)
    if cv2.contourArea(largest) < MIN_CONTOUR_AREA:
        last_center = None; return None
    x, y, w, h = cv2.boundingRect(largest)
    cx, cy = x + w//2, y + h//2
    new_center = (cx, cy)
    if last_center is None:
        stable = new_center
    else:
        stable = (
```

```
            CENTER_ALPHA*new_center[0] + (1-CENTER_ALPHA)*last_center[0],
            CENTER_ALPHA*new_center[1] + (1-CENTER_ALPHA)*last_center[1]
        )
    last_center = stable
    return int(stable[0]), int(stable[1])

def is_new_frame(frame):
    global last_gray_frame
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    if last_gray_frame is None:
        last_gray_frame = gray; return True
    diff = cv2.absdiff(gray, last_gray_frame)
    score = np.sum(diff)
    last_gray_frame = gray
    return score > FRAME_DIFF_THRESHOLD

clicked_px, clicked_py = None, None
current_frame = None

def on_click(event):
    global clicked_px, clicked_py, current_frame
    if current_frame is None:
        set_status("No camera frame", "warn"); return
    clicked_px, clicked_py = event.x, event.y
    frame_h, frame_w = current_frame.shape[:2]
    label_w = camera_label.winfo_width()
    label_h = camera_label.winfo_height()
    scale_x = frame_w / max(label_w, 1)
    scale_y = frame_h / max(label_h, 1)
    px = int(clicked_px * scale_x)
    py = int(clicked_py * scale_y)
    try:
```

```
            world_x, world_y = pixel_to_world(px, py)
            x_label.config(text=f"{world_x:.2f}")
            y_label.config(text=f"{world_y:.2f}")
        except RuntimeError as e:
            set_status(str(e), "error")

camera_label.bind("<Button-1>", on_click)

def update_frame():
    global current_frame, last_auto_time
    global auto_busy, auto_state, camera_available, cap

    if not camera_available:
        blank = np.zeros((300, 400, 3), dtype=np.uint8)
        cv2.putText(blank, "NO SIGNAL", (110, 155),
                    cv2.FONT_HERSHEY_SIMPLEX, 1.0, (40, 60, 80), 2)
        img = ImageTk.PhotoImage(Image.fromarray(blank))
        camera_label.imgtk = img
        camera_label.configure(image=img)
        return

    ret, frame = cap.read()
    if not ret:
        camera_available = False
        cap.release()
        cap = None
        cam_dot.config(text="● CAM: LOST", fg=C_ERROR)
        connect_btn.config(text="CONNECT CAM", fg=C_ACCENT, bg=C_BORDER)
        set_status("Camera connection lost — press CONNECT CAM to retry", "error")
        blank = np.zeros((300, 400, 3), dtype=np.uint8)
        cv2.putText(blank, "SIGNAL LOST", (90, 155),
                    cv2.FONT_HERSHEY_SIMPLEX, 1.0, (60, 30, 30), 2)
```

```
    img = ImageTk.PhotoImage(Image.fromarray(blank))
    camera_label.imgtk = img
    camera_label.configure(image=img)
    return

current_frame = frame.copy()

frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
frame_pil = Image.fromarray(frame_rgb).resize((400, 300))
imgtk = ImageTk.PhotoImage(frame_pil)
camera_label.imgtk = imgtk
camera_label.configure(image=imgtk)

mask = np.zeros((frame.shape[0], frame.shape[1]), dtype=np.uint8)

if auto_detect and is_new_frame(frame):
    if sorting_mode == "color":
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        color_name = selected_color.get()
        lower, upper = COLOR_RANGES[color_name]
        mask = cv2.inRange(hsv, np.array(lower), np.array(upper))
        mask = cv2.erode(mask, kernel, iterations=1)
        mask = cv2.dilate(mask, kernel, iterations=2)
        center = find_object_center(mask)
        if center is not None:
            cx, cy = center
            try:
                world_x, world_y = pixel_to_world(cx, cy)
                x_label.config(text=f"{world_x:.2f}")
                y_label.config(text=f"{world_y:.2f}")
            except RuntimeError as e:
                set_status(str(e), "error")
```

```
            root.after(20, update_frame)
            return
        now = time.time()
        if (not auto_busy
            and now - last_auto_time > AUTO_INTERVAL
            and now - last_pick_time > AUTO_COOLDOWN):
            last_auto_time = now
            if auto_calculate_only():
                auto_state = "open"
                auto_busy = True

    elif sorting_mode == "second":
        global qr_detected_data, qr_target_bin, last_qr_time

        bin_id, qr_cx, qr_cy = detect_qr(frame)

        if bin_id is not None and bin_id not in BIN_POSITIONS:
            set_status(f"QR '{bin_id}': невідомий бункер — ігнорую", "warn")
            qr_info_label.config(text=f"QR: '{bin_id}' — UNKNOWN BIN", fg=C_WARNING)
            root.after(20, update_frame)
            return

        if bin_id is not None:
            qr_info_label.config(text=f"QR: '{bin_id}'  →  Bin {bin_id}", fg=C_QR)
            now = time.time()

            if not auto_busy and (bin_id != qr_target_bin or now - last_qr_time >
QR_COOLDOWN):

                try:
                    pickup_x, pickup_y = pixel_to_world(qr_cx, qr_cy)
                except RuntimeError as e:
```

```
    set_status(f"Гомографія недоступна: {e}", "error")
    qr_info_label.config(text="QR: homography ERROR", fg=C_ERROR)
    log_qr_scan(
        qr_data=bin_id,
        target_bin=bin_id,
        pickup_xyz=(qr_cx, qr_cy, 0),
        drop_xyz=BIN_POSITIONS.get(bin_id, (0, 0, 0)),
        status="FAILED_HOMOGRAPHY"
    )
    _reset_qr_state()
    root.after(20, update_frame)
    return

pickup_z = float(z_label["text"]) if z_label["text"] not in ("---", "") else 10.0
x_label.config(text=f"{pickup_x:.2f}")
y_label.config(text=f"{pickup_y:.2f}")
z_label.config(text=f"{pickup_z:.2f}")

bx, by, bz = BIN_POSITIONS[bin_id]

try:
    drop_angles = calculate(bx, by, bz)
except Exception as e:
    set_status(f"Drop IK помилка: {e}", "error")
    qr_info_label.config(text=f"QR: drop IK ERROR bin {bin_id}", fg=C_ERROR)
    log_qr_scan(
        qr_data=bin_id,
        target_bin=bin_id,
        pickup_xyz=(pickup_x, pickup_y, pickup_z),
        drop_xyz=(bx, by, bz),
        status="FAILED_DROP_IK"
    )
```

```
    _reset_qr_state()
    root.after(20, update_frame)
    return

global drop_base_val, drop_theta1_val, drop_theta2_val, drop_theta3_val
drop_base_val, drop_theta1_val, drop_theta2_val, drop_theta3_val = drop_angles

qr_target_bin    = bin_id
qr_detected_data = bin_id
last_qr_time     = now

cv2.circle(frame, (qr_cx, qr_cy), 8, (0, 200, 255), -1)

if auto_calculate_only():
    if arduino is None:
        set_status("QR знайдено, але Arduino не підключений", "error")
        qr_info_label.config(text="QR: ARDUINO OFFLINE", fg=C_ERROR)
        _reset_qr_state()
    else:
        qr_drop_pending = True
        auto_state = "open"
        auto_busy  = True
        set_status(
            f"QR bin {bin_id}: ({pickup_x:.1f},{pickup_y:.1f})→({bx},{by})",
            "info"
        )
        log_qr_scan(
            qr_data=bin_id,
            target_bin=bin_id,
            pickup_xyz=(pickup_x, pickup_y, pickup_z),
            drop_xyz=(bx, by, bz),
            status="PICKED"
```

```
                )
            else:
                set_status(f"QR bin {bin_id}: pickup IK не вирішено", "error")
                qr_info_label.config(text=f"QR: pickup IK ERROR bin {bin_id}", fg=C_ERROR)
                log_qr_scan(
                    qr_data=bin_id,
                    target_bin=bin_id,
                    pickup_xyz=(pickup_x, pickup_y, pickup_z),
                    drop_xyz=(bx, by, bz),
                    status="FAILED_PICKUP_IK"
                )
                _reset_qr_state()

        else:
            if not auto_busy:
                now = time.time()
                if qr_target_bin is not None and now - last_qr_time > QR_COOLDOWN:
                    qr_target_bin = None
                qr_info_label.config(text="QR: scanning…", fg=C_MUTED)

    if auto_busy:
        auto_step()

    mask_rgb = cv2.cvtColor(mask, cv2.COLOR_GRAY2RGB)
    colored_mask = np.zeros_like(mask_rgb)
    colored_mask[mask_rgb[:,:,0] > 0] = [0, 229, 255]
    mask_pil = Image.fromarray(colored_mask).resize((400, 240))
    mask_imgtk = ImageTk.PhotoImage(mask_pil)
    mask_label.imgtk = mask_imgtk
    mask_label.configure(image=mask_imgtk)

    root.after(20, update_frame)
```

```
root.mainloop()


if cap is not None:
    cap.release()
cv2.destroyAllWindows()
```

Програмний код logic

```
from numpy import *
import numpy as np

def get_angles(py, pz, phi_deg=240):
    #change to length of your 3 links robotic arm
    a1 = 10
    a2 = 7.2
    a3 = 11.5

    phi = deg2rad(phi_deg)

    wy = py - a3*cos(phi)
    wz = pz - a3*sin(phi)

    delta = wy**2 + wz**2

    c2 = (delta - a1**2 - a2**2) / (2 * a1 * a2)
    s2 = sqrt(1 - c2**2)
    theta_2 = arctan2(s2, c2)

    s1 = ((a1 + a2 * c2) * wz - a2 * s2 * wy) / delta
    c1 = ((a1 + a2 * c2) * wy + a2 * s2 * wz) / delta
    theta_1 = arctan2(s1, c1)

    theta_3 = phi - theta_1 - theta_2

    return rad2deg(theta_1), rad2deg(theta_2), rad2deg(theta_3)

def calculate_base_angle(x, y):
    x = x
    y = -y


    angle_rad = np.arctan2(x, y)
    angle_deg = np.degrees(angle_rad)

    return angle_deg
```

```
def signed_R(py, px):
    R = np.sqrt(py**2 + px**2)
    if py < 0:
        R = -R
    return R


def calculate(px,py,pz):

    offset = 0
    target_x = px + offset
    target_y = py

    R = signed_R(py,px)


    base_angle = calculate_base_angle(target_x, target_y)

    theta1, theta2, theta3 = get_angles(R, pz)
    return base_angle, theta1, theta2, theta3
```