# ДОДАТОК А

Лістинг коду

```
import sqlite3
from datetime import datetime
from tabulate import tabulate

# Підключення до бази
conn = sqlite3.connect('pipes.db')
cursor = conn.cursor()

# Перевірка таблиць
def check_tables():
    required_tables = ['Orders', 'OrderDetails', 'Pipes', 'StorageZones', 'Events', 'ShippedPipes']
    for table in required_tables:
        cursor.execute("SELECT name FROM sqlite_master WHERE type='table' AND name = ?", (table,))
        if not cursor.fetchone():
            print(f"Помилка: Таблиця '{table}' не існує.")
            exit()
    print("Усі таблиці є.")

# Логування події
def log_event(message):
    cursor.execute("INSERT INTO Events (event_time, event_message) VALUES (?, ?)",
                   (datetime.now().strftime('%Y-%m-%d %H:%M:%S'), message))
    conn.commit()
    print(f"Подія: {message}")
```

```
# Виведення подій
def print_events():
    cursor.execute("SELECT event_time, event_message FROM Events ORDER BY event_time")
    events = cursor.fetchall()
    if events:
        print("\nЖурнал подій:")
        print(tabulate(events, headers=["Час", "Подія"], tablefmt="grid"))
    else:
        print("\nЖурнал подій порожній.")

# Зчитування замовлень
def get_orders():
    cursor.execute("""
        SELECT DISTINCT o.order_id, o.deadline, o.status, o.order_date, o.customer
        FROM Orders o
        JOIN OrderDetails od ON o.order_id = od.order_id
        WHERE o.status = 'в обробці'
        ORDER BY CASE WHEN o.deadline IS NULL THEN 1 ELSE 0 END, o.deadline ASC, o.order_date ASC
    """)
    orders = cursor.fetchall()
    result = []
    for order in orders:
        order_id = order[0]
        cursor.execute("SELECT purpose, pipe_count FROM OrderDetails WHERE order_id = ?", (order_id,))
        details = cursor.fetchall()
        for detail in details:
```

```
        result.append((order_id, order[1], order[2], detail[0], detail[1], order[3], order[4]))
    return result

# Зчитування зон
def get_storage_zones():
    cursor.execute("SELECT zone_name, purpose, current_count, available_count FROM StorageZones")
    return cursor.fetchall()

# Оновлення статусу труби
def update_pipe_status(pipe_id, order_id, new_status, zone_name):
    cursor.execute("UPDATE Pipes SET order_id = ?, status = ?, status_time = ?, zone_name = ? WHERE pipe_id = ?",
                   (order_id, new_status, datetime.now().strftime('%Y-%m-%d %H:%M:%S'), zone_name, pipe_id))
    conn.commit()

# Оновлення зони
def update_zone_count(zone_name, count_change):
    cursor.execute("UPDATE StorageZones SET current_count = current_count + ? WHERE zone_name = ?",
                   (count_change, zone_name))
    conn.commit()

# Відправлення труб
def ship_pipes():
    cursor.execute("SELECT pipe_id, purpose, order_id, zone_name FROM Pipes WHERE status = 'готово' AND zone_name IN ('Send1', 'Send2')")
    shipped = cursor.fetchall()
```

```
    for pipe in shipped:
        pipe_id, purpose, order_id, zone_name = pipe
        cursor.execute("INSERT INTO ShippedPipes (pipe_id, purpose, order_id, zone_name, shipped_time) VALUES (?, ?, ?, ?, ?)",
                       (pipe_id, purpose, order_id, zone_name, datetime.now().strftime('%Y-%m-%d %H:%M:%S')))
        cursor.execute("DELETE FROM Pipes WHERE pipe_id = ?", (pipe_id,))
        update_zone_count(zone_name, -1)  # Очищення зони відправки
    conn.commit()
    if shipped:
        log_event(f"Відправлено {len(shipped)} труб.")

# Сортування труб
def sort_pipes(pipes, orders, zones):
    sorted_pipes = []
    defect_count = 0
    order_zones = {}
    external_order_count = 0
    overflow_logged = {'водопровідні': False, 'газові': False}
    logged_orders = set()  # Для відстеження логування початку виробництва

    # Групуємо труби за order_id і purpose
    pipes_by_order = {}
    for pipe in pipes:
        pipe_id, purpose, status, order_id, zone_name, _ = pipe
        if status != 'виробляється':
            continue
        key = (order_id, purpose)
        if key not in pipes_by_order:
            pipes_by_order[key] = []
```

```
        pipes_by_order[key].append((pipe_id, purpose, status, order_id, zone_name))

    # Обробка замовлень у порядку дедлайнів
    for order in orders:
        order_id, deadline, status, purpose, pipe_count, order_date, customer = order
        key = (order_id, purpose)
        if key not in pipes_by_order:
            continue

        # Логування початку виробництва один раз для замовлення
        if order_id not in logged_orders:
            log_event(f"Початок виробництва замовлення ID:{order_id}")
            logged_orders.add(order_id)

        pipe_counter = 0  # Скидаємо лічильник для кожного замовлення і типу

        zone_code = 'VT' if purpose == 'водопровідні' else 'GT'
        storage_zone_name = f'Storage{zone_code}' if customer == 'Склад' else None
        temp_zone_name = f'Temp{zone_code}' if customer == 'Склад' else None
        storage_zone = next((z for z in zones if z[0] == storage_zone_name), None) if storage_zone_name else None
        temp_zone = next((z for z in zones if z[0] == temp_zone_name), None) if temp_zone_name else None

        storage_free = storage_zone[3] - storage_zone[2] if storage_zone else 0
        temp_free = temp_zone[3] - temp_zone[2] if temp_zone else 0

        # Визначаємо зону відправлення для зовнішніх замовлень
        if customer != 'Склад':
            if order_id not in order_zones:
```

```
            external_order_count += 1
            zone_name = 'Send1' if external_order_count == 1 else 'Send2'
            order_zones[order_id] = zone_name
        else:
            zone_name = order_zones[order_id]
    else:
        zone_name = None

    for pipe in pipes_by_order[key]:
        pipe_id, pipe_purpose, status, pipe_order_id, pipe_zone_name = pipe
        pipe_counter += 1

        # Брак: кожна 50-та труба в межах замовлення
        if pipe_counter % 50 == 0:
            defect_count += 1
            zone = next((z for z in zones if z[0] == 'Defect'), None)
            if zone and zone[2] < zone[3]:
                sorted_pipes.append((pipe_id, pipe_purpose, 'готово', None, 'Defect'))
                update_pipe_status(pipe_id, None, 'готово', 'Defect')
                update_zone_count('Defect', 1)

                # Перевіряємо місце перед додаванням заміни
                if customer == 'Склад':
                    if storage_free > 0:
                        sorted_pipes.append((None, pipe_purpose, 'готово', order_id,
storage_zone_name))
                        cursor.execute("INSERT INTO Pipes (purpose, status, order_id,
status_time, zone_name) VALUES (?, 'готово', ?, ?, ?)",
                                       (pipe_purpose, order_id, datetime.now().strftime('%Y-%m-
%d %H:%M:%S'), storage_zone_name))
```

```
                update_zone_count(storage_zone_name, 1)
                storage_free -= 1
            elif temp_free > 0:
                sorted_pipes.append((None, pipe_purpose, 'готово', order_id, temp_zone_name))
                cursor.execute("INSERT INTO Pipes (purpose, status, order_id, status_time, zone_name) VALUES (?, 'готово', ?, ?, ?)",
                               (pipe_purpose, order_id, datetime.now().strftime('%Y-%m-%d %H:%M:%S'), temp_zone_name))
                update_zone_count(temp_zone_name, 1)
                temp_free -= 1
                if not overflow_logged[pipe_purpose]:
                    log_event(f"Зона {storage_zone_name} заповнена. Труби перенаправлено в {temp_zone_name}.")
                    overflow_logged[pipe_purpose] = True
            else:
                log_event(f"Немає місця для заміни бракованої труби ID:{pipe_id} (замовлення {order_id}). Виробництво зупинено.")
                print(f"Помилка: Немає місця для заміни бракованої труби. Виробництво зупинено.")
                return sorted_pipes
        else:
            zone = next((z for z in zones if z[0] == zone_name), None)
            if zone and zone[2] < zone[3]:
                sorted_pipes.append((None, pipe_purpose, 'готово', order_id, zone_name))
                cursor.execute("INSERT INTO Pipes (purpose, status, order_id, status_time, zone_name) VALUES (?, 'готово', ?, ?, ?)",
                               (pipe_purpose, order_id, datetime.now().strftime('%Y-%m-%d %H:%M:%S'), zone_name))
```

```
                update_zone_count(zone_name, 1)
            else:
                log_event(f"Зона {zone_name} переповнена для заміни бракованої труби ID:{pipe_id} (замовлення {order_id}). Виробництво зупинено.")
                print(f"Помилка: Зона {zone_name} переповнена. Виробництво зупинено.")
                return sorted_pipes

        conn.commit()
        log_event(f"Виявлено брак труби ID:{pipe_id}. Додано заміну.")
    else:
        log_event(f"Зона браку переповнена для труби ID:{pipe_id} (замовлення {order_id}). Виробництво зупинено.")
        print(f"Помилка: Зона браку переповнена. Виробництво зупинено.")
        return sorted_pipes
    continue

# Розподіл труб
if customer == 'Склад':
    if storage_free > 0:
        sorted_pipes.append((pipe_id, pipe_purpose, 'готово', order_id, storage_zone_name))
        update_pipe_status(pipe_id, order_id, 'готово', storage_zone_name)
        update_zone_count(storage_zone_name, 1)
        storage_free -= 1
    elif temp_free > 0:
        sorted_pipes.append((pipe_id, pipe_purpose, 'готово', order_id, temp_zone_name))
        update_pipe_status(pipe_id, order_id, 'готово', temp_zone_name)
        update_zone_count(temp_zone_name, 1)
```

```
                temp_free -= 1
                if not overflow_logged[pipe_purpose]:
                    log_event(f"Зона {storage_zone_name} заповнена. Труби перенаправлено в {temp_zone_name}.")
                    overflow_logged[pipe_purpose] = True
            else:
                log_event(f"Зони {storage_zone_name} і {temp_zone_name} заповнені для замовлення {order_id}. Виробництво зупинено.")
                print(f"Помилка: Зони {storage_zone_name} і {temp_zone_name} заповнені. Виробництво зупинено.")
                return sorted_pipes
        else:
            zone = next((z for z in zones if z[0] == zone_name), None)
            if zone and zone[2] < zone[3]:
                sorted_pipes.append((pipe_id, pipe_purpose, 'готово', order_id, zone_name))
                update_pipe_status(pipe_id, order_id, 'готово', zone_name)
                update_zone_count(zone_name, 1)
            else:
                log_event(f"Зона {zone_name} переповнена для труби ID:{pipe_id} (замовлення {order_id}). Виробництво зупинено.")
                print(f"Помилка: Зона {zone_name} переповнена. Виробництво зупинено.")
                return sorted_pipes

    return sorted_pipes

# Основна логіка
def sorting_module():
    check_tables()
```

```
    ship_pipes()

    cursor.execute("UPDATE Pipes SET status = 'виробляється', status_time = ? WHERE status = 'в обробці'",
                   (datetime.now().strftime('%Y-%m-%d %H:%M:%S'),))
    conn.commit()

    orders = get_orders()
    zones = get_storage_zones()
    cursor.execute("""
        SELECT p.pipe_id, p.purpose, p.status, p.order_id, p.zone_name, o.deadline
        FROM Pipes p
        LEFT JOIN Orders o ON p.order_id = o.order_id
        WHERE p.status = 'виробляється'
        ORDER BY CASE WHEN o.deadline IS NULL THEN 1 ELSE 0 END, o.deadline ASC, p.pipe_id
    """)
    pipes = cursor.fetchall()

    if not pipes or not zones:
        log_event("Немає труб або зон для обробки.")
        print_events()
        return

    sorted_pipes = sort_pipes(pipes, orders, zones)

    # Перевірка завершення в порядку дедлайнів
    unique_order_ids = list(dict.fromkeys([order[0] for order in orders]))  # Уникнення дублікатів
```

```
    for order_id in unique_order_ids:
        cursor.execute("SELECT purpose, pipe_count FROM OrderDetails WHERE order_id = ?", (order_id,))
        details = cursor.fetchall()
        completed = True
        for purpose, pipe_count in details:
            cursor.execute("SELECT COUNT(*) FROM Pipes WHERE order_id = ? AND purpose = ? AND status = 'готово'",
                           (order_id, purpose))
            count = cursor.fetchone()[0]
            if count < pipe_count:
                completed = False
                break
        if completed:
            cursor.execute("UPDATE Orders SET status = 'завершено' WHERE order_id = ?", (order_id,))
            conn.commit()
            log_event(f"Замовлення ID:{order_id} завершено.")

    print_events()

# Запуск
try:
    sorting_module()
finally:
    conn.close()
```

# ДОДАТОК Б

Демостраційний метріал

# Кваліфікаційна робота бакалавра

На тему: «Розроблення модуля автоматизованого управління сортуванням поліетиленових труб»

Виконав:
студент групи АКТАКІТ-21-1
Одноралов Денис

Керівник:
Чала О.О.

ХНУРЕ
2025

# ВСТУП

- **Мета роботи** – підвищення якості автоматизованого процесу сортування в технологічних операціях виробництва та зберігання поліетиленових труб шляхом розробки програмного модуля на основі бази даних. Для цього було створено прототип програмної частини модуля, який забезпечує гнучке сортування та готовий до інтеграції з апаратними компонентами.
- Для досягенння поставленої мети необхідно виконати такі **завдання**: проаналізувати предметну область, розробити архітектуру, реалізувати програмну частину, провести тестування.
- **Актуальність** зумовлена різноманіттям типів труб, великими обсягами виробництва та потребою в оптимізації сортувально-логістичних процесів. Сортування є ключовим етапом логістики, що забезпечує ефективне транспортування та зберігання, а ручне чи частково автоматизоване сортування не відповідає сучасним вимогам продуктивності.
- **Об’єкт розробки** – автоматизоване управління сортуванням, **предмет розробки** – модуль автоматизованого управління сортуванням поліетиленових труб.

# Аналіз предметної області та вимог

Сортування є невід'ємною частиною логістичного процесу, що охоплює перевірку якості труб за допомогою ультразвукового дефектоскопу та комп'ютерного зору, нанесення штрих-кодів і транспортування по конвеєрній лінії до складських зон.

Вимоги до модуля:

- взаємодія з виробничою системою;
- інтеграція з базою даних;
- взаємодія з виконавчими механізмами;
- швидка обробка інформації;
- можливість маштабування.

**Порівняння з аналогами** (конвеєрні системи з фіксованими алгоритмами) показало їх обмежену гнучкість. Прототип перевершує їх завдяки автоматизації, гнучкій маршрутизації та пріоритизації замовлень.

| Критерій | JR Automation | Xuzhou DKEC Electrical Technology Co Ltd | Розроблений модуль |
|---|---|---|---|
| Сортування | За діаметром, стоси (2, 3, 5, 7 труб) | За кількістю | За замовленням |
| Інтеграція з MES | Відсутня | Через цифрову шину | OPC UA, затримка до 100 мс, шифрування AES-256 |
| Обробка браку | Ймовірно ручна | Частково автоматизована (оповіщення) | Автоматична |
| Пріоритизація замовлень | Не підтримується | Не підтримується | Через базу даних |
| Масштабність | Обмежена (палетування) | Обмежена (ліміт лічильної зони) | Гнучка |

**Сортування в прототипі** виконується за параметрами:

– якість (брак);

– терміновість замовлення;

– заповненість складських зон.

Ці параметри обрано для відповідності тестовим умовам, але система підтримує додаткові критерії (діаметр, призначення, товщина стінки, матеріал), що потребує розширення виробничих ліній і зон зберігання.

# Архітектура та алгоритми модуля

Програмний модуль складається з **програмної** та **апаратної** частин.

**Програмна частина**:

- Бази даних SQLite – забезпечує зберігання параметрів труб і замовлень для їх обробки та сортування.
- Програмний код Python з алгоритмом сотування та взаємодії між елементами системи – реалізує групування даних труб і координує роботу між датчиками, виконавчими механізмами та базою даних.

**Алгоритм сортування**

- Перевірка якості – дані з комп'ютерного зору та ультразвукового дефектоскопу аналізуються для визначення статусу труби (брак/якісна).
- Зчитування штрих-коду – сканер зчитує штрих-код труби, передаючи дані (ідентифікатор, тип, розміри, замовлення) до бази даних через модуль.
- Визначення маршруту – алгоритм групує труби за параметрами, пріоритетом замовлення та станом складських зон, обираючи зону призначення.
- Транспортування – модуль надсилає команди через PLC на конвеєрну лінію та розподільні вузли для переміщення труби.
- Зчитування та новлення статусу – статус труби (в обробці, бракована, готова) фіксується в базі даних після кожного основного етапу переміщення.
- Обробка винятків – браковані труби спрямовуються до зони «Defect», при переповненні зон або збоях модуль зупиняє систему та сигналізує MES.

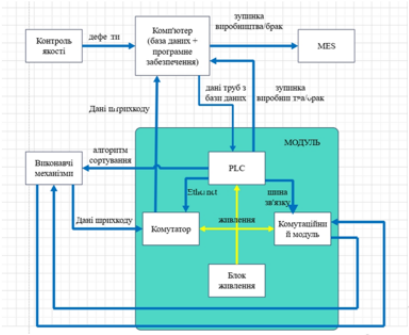

**Апаратна частина** – визначено такі компоненти для інтеграції:

- Програмований логічний контролер (PLC) Siemens S7-1200, який виконує центральну роль в обробці вхідних даних, керуванні виконавчими механізмами та передачі інформації.
- Для забезпечення стабільного живлення всіх компонентів використовується блок живлення SITOP PSU100S.
- Комутатор – модуль зв'язку SCALANCE XB005, який забезпечує надійне мережеве з'єднання.
- Комутаційний модуль – модуль цифрового вводу/виводу (I/O) SM 1223 передає та обробляє сигнали від сенсорів і виконавчих пристроїв.
- Монтажна шафа – фізична структура модуля Rittal AE (розміри 600 мм×400 мм×200 мм, клас захисту IP54), яка захищає обладнання від пилу, вологи та вібрацій, забезпечуючи довговічність і надійність у промислових умовах.
- Усі компоненти з'єднані екранованими кабелями (Ethernet Cat5e, M12 для мережевих з'єднань і сигнальні кабелі 24 В), що мінімізує електромагнітні перешкоди.

До **виконавчіх механізмів** відносяться:

- Комп'ютерний зір – використовується для розпізнавання розмірів і типу труб за допомогою камер і алгоритмів обробки зображень.
- Ультразвуковий дефектоскоп – виявляє внутрішні дефекти труб шляхом аналізу ультразвукових хвиль.
- Штампувальник штрих-кодів – наносить штрих-коди на труби для їх ідентифікації в системі.
- Сканери штрих-кодів – зчитують штрих-коди для відстеження труб у процесі сортування.
- Конвеєрна лінія та розподільний вузол «перекидний стіл» – забезпечують транспортування та розподіл труб за заданими маршрутами.
- Маніпулятори для переміщення труб – виконують точне переміщення труб між етапами сортування та зберігання.

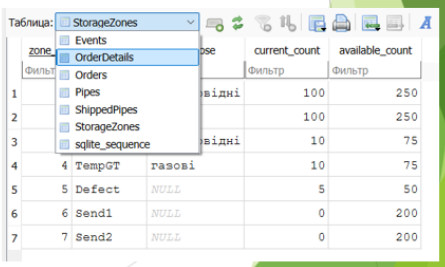
Також в роботі наведено структурно-функціональну схему, яка показує зв'язки між елементами системи.



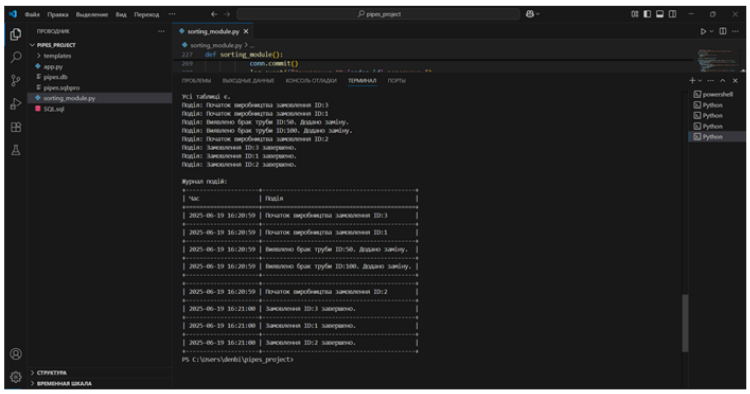

# Реалізація

Для симуляції роботи модуля розроблено **веб-ресурс**, який дозволяє створювати замовлення, що записуються до бази даних для подальшої обробки.


Також було створено **базу даних** SQLite, обрану для симуляції через її легкість і простоту. У реальних умовах та для масштабування можна використовувати інші бази даних, наприклад PostgreSQL або MySQL.



І найголовніше – **програмна реалізація модуля, виконана на Python**, який було обрано завдяки його універсальності та підтримці бібліотек для обробки даних і взаємодії з базами даних.



# Тестування

Програмна частина модуля протестованав в симуляційному середовищі за параметрами:

– брак (якість труби);

– терміновість замовлення (пріоритет за дедлайнами);

– заповненість складських зон.

З використанням початкових даних, заданих у таблиці StorageZones бази даних.

Де складські зони були ініціалізовані наступним чином:

– StorageVT: 100 водопровідних труб, вільно 150 з 250;

– StorageGT: 100 газових труб, вільно 150 з 250;

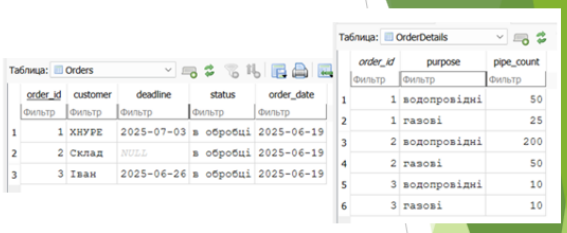
– TempVT: 10 водопровідних труб, вільно 65 з 75;

– TempGT: 10 газових труб, вільно 65 з 75;

– Defect: 5 бракованих труб, вільно 45 з 50;

– Send1 і Send2: порожні, вільно 200 з 200 для кожної.

Замовлення створювалися через веб-інтерфейс, користувач вводив дані (замовник, тип труб, кількість, дедлайн для тип замовленя), які записувалися до таблиць бази даних після підтвердження.

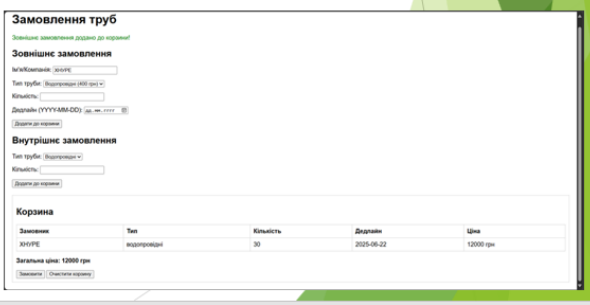


Таблиця: Orders

| order_id | customer | deadline | status | order_date |
|---|---|---|---|---|
| 1 | ХНУРЕ | 2025-07-03 | в обробці | 2025-06-19 |
| 2 | Склад | NULL | в обробці | 2025-06-19 |
| 3 | Іван | 2025-06-26 | в обробці | 2025-06-19 |

Таблиця: OrderDetails

| order_id | purpose | pipe_count |
|---|---|---|
| 1 | водопровідні | 50 |
| 1 | газові | 25 |
| 2 | водопровідні | 200 |
| 2 | газові | 50 |
| 3 | водопровідні | 10 |
| 3 | газові | 10 |



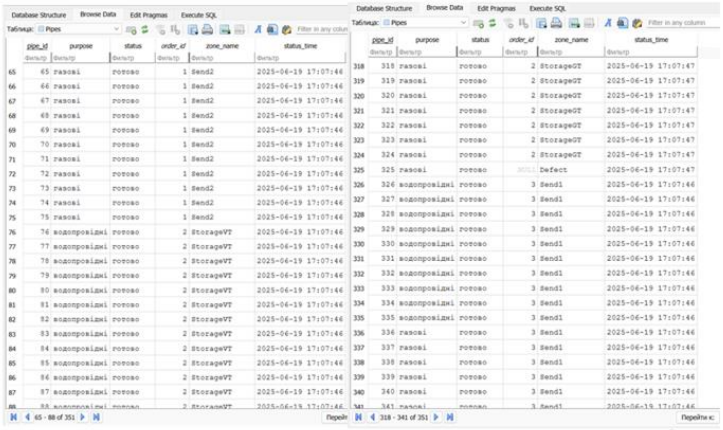
В результаті тестування модуль обробив 345 труб (260 водопровідних, 85 газових) для трьох замовлень із пріоритизацією за дедлайнами.

Замовлення «Іван» (було виконано першим за пріоритезацією виробництва) (20 труб) і «ХНУРЕ» (було виконано другим) (75 труб) розміщено в зонах Send1 і Send2.

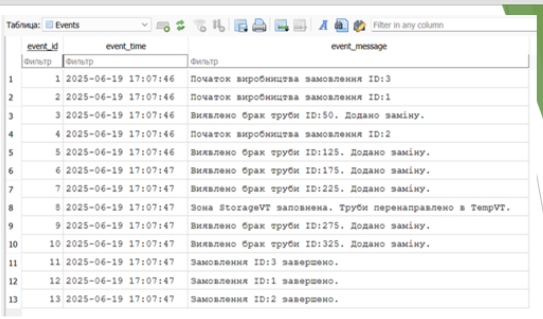
Замовлення «Склад» (200 водопровідних + 50 газових) (було виконано останнім за відсутністю пріоритету): 150 труб у StorageVT, 50 у TempVT, 50 у StorageGT.

Виявлено 5 бракованих труб, перенаправлено до Defect, заміни додано до виробництва.

Переповнення: 50 труб перенаправлено з StorageVT до TempVT через перевищення місткості.



Логування – усі події (сортування, брак, переповнення, завершення) зафіксовано в таблиці Events

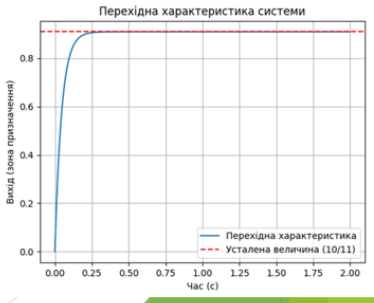


| event_id | event_time | event_message |
|---|---|---|
| 1 | 2025-06-19 17:07:46 | Початок виробництва замовлення ID:3 |
| 2 | 2025-06-19 17:07:46 | Початок виробництва замовлення ID:1 |
| 3 | 2025-06-19 17:07:46 | Виявлено брак труби ID:50. Додано заміну. |
| 4 | 2025-06-19 17:07:46 | Початок виробництва замовлення ID:2 |
| 5 | 2025-06-19 17:07:46 | Виявлено брак труби ID:125. Додано заміну. |
| 6 | 2025-06-19 17:07:47 | Виявлено брак труби ID:175. Додано заміну. |
| 7 | 2025-06-19 17:07:47 | Виявлено брак труби ID:225. Додано заміну. |
| 8 | 2025-06-19 17:07:47 | Зона StorageVT заповнена. Труби перенаправлено в TempVT. |
| 9 | 2025-06-19 17:07:47 | Виявлено брак труби ID:275. Додано заміну. |
| 10 | 2025-06-19 17:07:47 | Виявлено брак труби ID:325. Додано заміну. |
| 11 | 2025-06-19 17:07:47 | Замовлення ID:3 завершено. |
| 12 | 2025-06-19 17:07:47 | Замовлення ID:1 завершено. |
| 13 | 2025-06-19 17:07:47 | Замовлення ID:2 завершено. |

Також було перевірено модуль на стікість системи.

Модуль автоматизованого управління сортуванням труб протестовано на стійкість у симуляційному середовищі Python з бібліотекою control, моделюючи конвеєр як інерційний об’єкт із передатною функцією W(s)=1/(0,5s+1) та пропорційним регулятором (Kp=10).

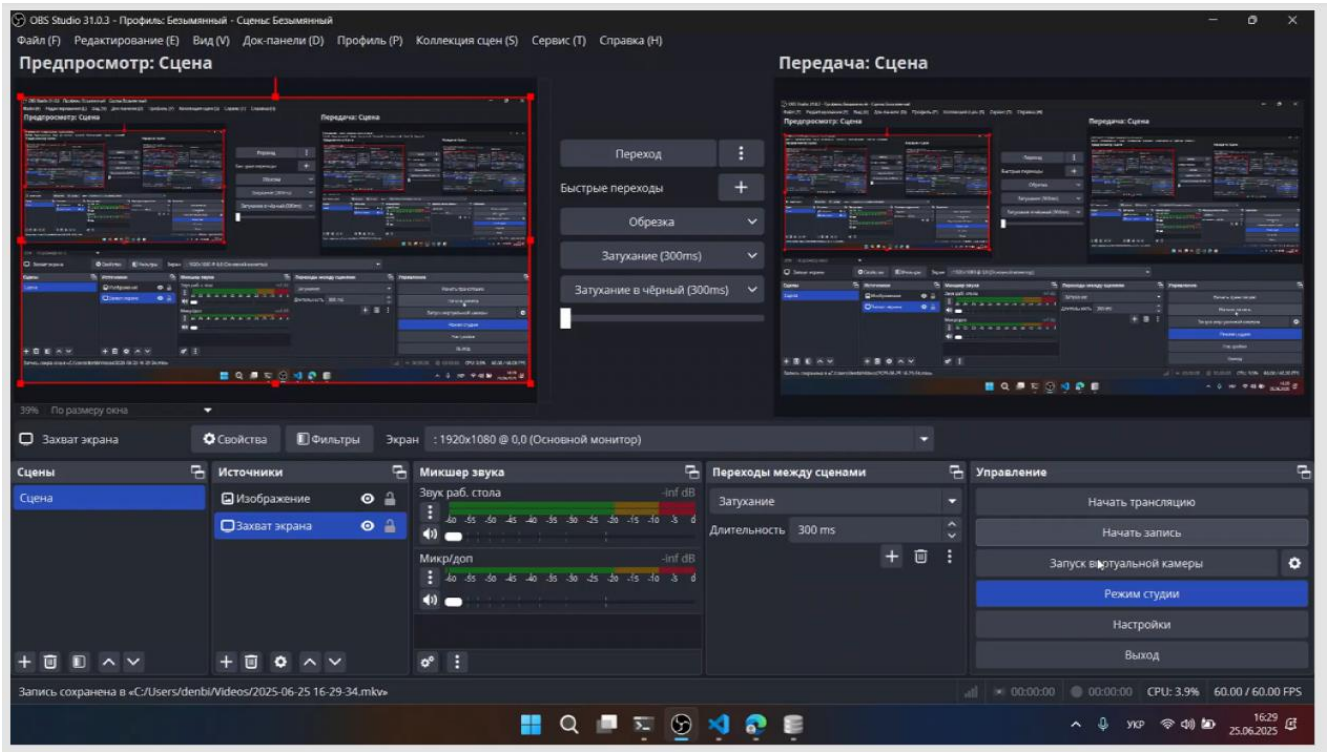
Стійкість контуру управління маршрутами підтверджено критерієм Гурвіца (полюс s=-22, коефіцієнти a1=0,5, a0=11). Симуляція перехідного процесу показала нульове перерегулювання (σ=0%) та швидкий час регулювання 0,18 с, забезпечуючи плавне досягнення цільової зони (y≈0,909). Графік перехідної характеристики демонструє надійність і швидкість модуля в управлінні маршрутами труб.




Аналіз безпеки праці доповнив відповідність модуля вимогам безпеки.

# Демонстрація

Пропоную переглянути коротке відео, яке демонструє роботу модуля: створення замовлення через веб-інтерфейс, обробку даних із бази даних для сортування.



# ВИСНОВКИ

В результаті кваліфікаційної роботи бакалавра було розроблено модуль автоматизованого управління сортуванням поліетиленових труб, який підвищує ефективність виробничих і складських процесів завдяки взаємодії з базою даних.

Аналіз предметної області виявив потребу в автоматизації через складність сортувально-логістичних процесів і обмежену гнучкість аналогів із фіксованими алгоритмами.

Архітектура модуля та його програмна реалізація забезпечують гнучке сортування, а тестування підтвердило працездатність системи.

# Дякую за увагу

| № документа | Призначення | Найменування | Додаткові відомості |
|---|---|---|---|
| | | Текстові документи | |
| | | | |
| 1 | ГЮИК. 505200.014 ПЗ | Пояснювальна записка | А4, 80 с. |
| | | | |
| | | Додаткові матеріали | |
| | | | |
| 2 | | Лістинг коду | А4, 11 с. |
| | | | |
| 3 | | Демонстраційний матеріал | А4, 8 с. |

| Змін. | Арк. | Номер докум. | Підпис | Дата | ГЮИК. 505200.014 ВД | | | |
|---|---|---|---|---|---|---|---|---|
| Розроб. | | Одноралов Д.Ю. | | | Розроблення модуля автоматизованого управління сортуванням поліетиленових труб | Літера | Аркуш | Аркушів |
| Перевір. | | Чала О.О. | | | | Н | 1 | 1 |
| | | | | | | Кафедра КІТАР ХНУРЕ | | |
| Н.контр. | | Деська Н.П. | | | | | | |
| Затв. | | Невлюдов І.Ш. | | | | | | |