Київський політехнічний інститут імені Ігоря Сікорського

Приладобудівний факультет

Кафедра автоматизації та систем неруйнівного контролю

Лабораторна робота №1

з предмету "Комп'ютерний зір"

Студент: Погорєлов Богдан
Група: ПК-51мп

2025 рік

Тема: Розробка програмного забезпечення для керування поворотною платформою та захоплення відео

Варіант 12

Розробити дві програми, які виконують такі дії з цифровим відео (ЦВ):

1. Перша програма, яка записується у контролер поворотних платформ, керує поворотною платформою з двома серво приводами. Ця програма забезпечує сканування простору зовнішньою цифровою камерою, що встановлена на поворотній платформі. Сканування здійснюється по горизонталі та по вертикалі, кількість точок сканування повинна бути не менше 5 х 5. Тип сканування та оператори циклу, які забезпечують це сканування, вказані у таблиці 2.
2. Друга програма зчитує та відображає ЦВ з зовнішньої цифрової камери.

Таблиця 2. Варіанти завдання лабораторних робіт № 1, 2.

Тип сканування	**Два цикли з параметром**	**Два цикли з передумовою**	**Зовнішній цикл з параметром, а внутрішній цикл з передумовою**	**Зовнішній цикл з передумовою, а внутрішній цикл з параметром**
По рядкам з початку рядка	1	2	3	4
По рядкам з початку рядка «змійкою»	6	7	8	9
По стовбцям з зверху вниз	11	12	13	14
По стовбцям зверху вниз «змійкою»	16	17	18	19
По рядкам з кінця рядка до початку	21	22	23	24
По стовбцям з знизу вверх	5	10	15	20

Ось структура лабораторної роботи №1 у форматі Jupyter Notebook, адаптована під твій Варіант 12:

Тип сканування: По стовбцях зверху вниз.

Тип циклів: Два цикли з передумовою (while).

Лабораторна робота №1 Тема: Розробка програмного забезпечення для керування поворотною платформою та захоплення відео. Варіант: 12

Частина 1. Програма для мікроконтролера (Arduino) Завдання: Реалізувати сканування простору (мінімум 5x5 точок). Логіка варіанту 12:

По стовбцях: Зовнішній цикл змінює горизонтальний кут (Pan), внутрішній — вертикальний (Tilt).

Зверху вниз: Внутрішній цикл йде від мінімального кута до максимального (або навпаки, залежно від монтажу серво).

Цикли з передумовою: Використовуємо конструкцію while (condition) { ... }.

Нижче наведено код для платформи Arduino (C++).

Частина 2. Програма для зчитування відео (Python) Завдання: Зчитати та відобразити відеопотік із зовнішньої камери. Інструменти: Python, бібліотека OpenCV (cv2).

Цей код виконується на комп'ютері. Він підключається до веб-камери (або камери, підключеної через USB/IP), зчитує кадри в нескінченному циклі та виводить їх у вікно

```
#include <Servo.h>

// Створення об'єктів сервоприводів
Servo servoPan;  // Горизонталь (X)
Servo servoTilt; // Вертикаль (Y)

// Налаштування пінів
const int PIN_PAN = 9;
const int PIN_TILT = 10;

// Налаштування меж сканування (градуси)
const int PAN_START = 45;
const int PAN_END = 135;
const int TILT_START = 45;
const int TILT_END = 135;

// Крок сканування та затримка
const int STEP = 18;
const int DELAY_TIME = 500;

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

  servoPan.attach(PIN_PAN);
  servoTilt.attach(PIN_TILT);

  servoPan.write(PAN_START);
  servoTilt.write(TILT_START);
  delay(1000);
}
```

```
void loop() {
  int currentPan = PAN_START;
  Serial.println("Start");
  while (currentPan <= PAN_END) {
  Serial.println("H"+String(currentPan));
  servoPan.write(currentPan);
  int currentTilt = TILT_START;
  while (currentTilt <= TILT_END) {
    servoTilt.write(currentTilt);
    delay(DELAY_TIME);
    Serial.println("W"+String(currentTilt));
    currentTilt += STEP;
  }
  currentPan += STEP;
  }
  delay(1000);
}
```

In [ ]:

```
import cv2
import serial
import time

SERIAL_PORT = 'COM3'
BAUD_RATE = 115200

def cv2_main():

  ports = serial.tools.list_ports.comports()
  # Print information about each port
  if ports:
    print("Available Serial Ports:")
    for port in ports:
      print(f"  Device: {port.device}")
      print(f"  Name: {port.name}")
      print(f"  Description: {port.description}")
      print(f"  Hardware ID: {port.hwid}")
      print("-" * 20)
  else:
    print("No serial ports found.")
    return

  cap = cv2.VideoCapture(0)
  if not cap.isOpened():
    print("Помилка: Камера не знайдена.")
    return

  # 2. Відкриття Serial порту
  try:
    ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=0.1)
    time.sleep(2) # Час на перезавантаження Arduino (DTR)
    print(f"Підключено до {SERIAL_PORT}")
  except Exception as e:
    print(f"Помилка Serial порту: {e}")
    return

  print("Очікування команд від контролера (формат Hxx/Wxx)...")
  print("Натисніть 'q' для виходу.")
```

```
try:
  while True:
    # Завжди оновлюємо кадр, щоб буфер не застарів
    ret, frame = cap.read()
    if not ret:
      break

    # 3. Перевірка даних від Arduino
    if ser.in_waiting > 0:
      try:
        # Зчитуємо рядок, декодуємо і прибираємо пробіли
        line = ser.readline().decode('utf-8').strip()

        # Логіка обробки команд
        if line == "Start":
          print("--- Початок сканування ---")

        # Обробка зміни стовпця (наприклад, "H45")
        elif line.startswith("H"):
          angle = line[1:] # Вирізаємо все після 'H'
          print(f"-> Зміна стовпця. Pan: {angle}")

        # Обробка команди знімку (наприклад, "W45")
        elif line.startswith("W"):
          angle = line[1:] # Вирізаємо все після 'W'
          print(f"   [!] Захоплення кадру. Tilt: {angle}")

          # Візуалізація моменту захоплення
          cv2.circle(frame, (50, 50), 20, (0, 0, 255), -1)
          # Виводимо текст з кутом нахилу
          text = f"REC (Tilt {angle})"
          cv2.putText(frame, text, (80, 60),
                cv2.FONT_HERSHEY_SIMPLEX,
                0.7, (0, 0, 255), 2)

          filename = f"scan/p{angle}_t{angle}_{int(time.time())}.jpg"
          cv2.imwrite(filename, frame)

      except UnicodeDecodeError:# Ігноруємо биті байти при підключенні
        pass

    # Відображення поточного відео
    cv2.imshow('Lab 1 - Scanner View', frame)

    # Вихід за 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
      break

except KeyboardInterrupt:
  print("Зупинено користувачем.")

finally:
  if 'ser' in locals() and ser.is_open:
    ser.close()
  cap.release()
  cv2.destroyAllWindows()
  print("Ресурси звільнено.")
```

In [ ]:

```python
import tkinter as tk
from PIL import Image, ImageTk
import cv2
import os

# --- КОНСТАНТИ ---
FOLDER_NAME = "scanned_photos"
THUMB_SIZE = (200, 150)
COLUMNS = 5
CHECK_INTERVAL_MS = 1000

def get_file_list(folder):
    """Повертає відсортований список файлів. Чиста функція."""
    try:
        return sorted([
            f for f in os.listdir(folder)
            if f.lower().endswith(('.png', '.jpg', '.jpeg'))
        ])
    except FileNotFoundError:
        return []

def load_image(path, size):
    """Завантажує та обробляє зображення. Повертає об'єкт ImageTk або None."""
    cv_img = cv2.imread(path)
    if cv_img is None:
        return None

    # BGR -> RGB
    cv_img = cv2.cvtColor(cv_img, cv2.COLOR_BGR2RGB)
    pil_img = Image.fromarray(cv_img)
    pil_img = pil_img.resize(size, Image.Resampling.LANCZOS)
    return ImageTk.PhotoImage(pil_img)

def clear_widgets(container):
    """Очищує контейнер."""
    for widget in container.winfo_children():
        widget.destroy()

def create_photo_card(container, image, filename, index, cols):
    """Створює та розміщує один елемент галереї."""
    frame = tk.Frame(container, bg="white", bd=2, relief="groove")

    row_idx = index // cols
    col_idx = index % cols
    frame.grid(row=row_idx, column=col_idx, padx=5, pady=5)

    tk.Label(frame, image=image, bg="white").pack()
    tk.Label(frame, text=filename, bg="white", font=("Arial", 8)).pack()

def update_ui(container, files, folder, size, cols, ref_list):
    """
    Функція з побічними ефектами: оновлює GUI.
    ref_list — мутабельний список для зберігання посилань на зображення.
    """
    clear_widgets(container)
    ref_list.clear() # Очищаємо старі посилання

    for i, filename in enumerate(files):
        path = os.path.join(folder, filename)
```

```
        tk_image = load_image(path, size)

        if tk_image:
            ref_list.append(tk_image) # Захист від Garbage Collector
            create_photo_card(container, tk_image, filename, i, cols)

def app_loop(root, container, displayed_files_set, photo_refs):
    """
    Головний цикл програми.
    Рекурсивно викликає себе через root.after, передаючи оновлений стан.
    """
    current_files = get_file_list(FOLDER_NAME)
    current_files_set = set(current_files)

    # Перевіряємо, чи змінився вміст папки
    if current_files_set != displayed_files_set:
        print("Знайдено зміни. Оновлення галереї...")
        update_ui(container, current_files, FOLDER_NAME, THUMB_SIZE,
                  COLUMNS, photo_refs)
        # Оновлюємо стан для наступної ітерації
        next_displayed_set = current_files_set
    else:
        # Стан не змінився
        next_displayed_set = displayed_files_set

    # Плануємо наступний виклик через 1 секунду
    root.after(CHECK_INTERVAL_MS, lambda: app_loop(root, container,
                                                   next_displayed_set, photo_ref

def grid_main():
    if not os.path.exists(FOLDER_NAME):
        os.makedirs(FOLDER_NAME)

    # Ініціалізація вікна
    root = tk.Tk()
    root.title("Лабораторна 1 - Галерея (Functional)")
    root.geometry("1100x600")

    container = tk.Frame(root, bg="gray90")
    container.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

    # Стан: список для посилань на картинки (щоб вони не зникали)
    # Це єдиний мутабельний об'єкт, який ми змушені зберігати довготривало
    photo_refs_storage = []

    # Запуск циклу (початковий стан порожній)
    app_loop(root, container, set(), photo_refs_storage)

    root.mainloop()
```

In [ ]:

```
import threading

# 1. Запускаємо камеру (OpenCV) у фоновому потоці.
cv_thread = threading.Thread(target=cv2_main, daemon=True)
cv_thread.start()
print("Камера запущена у фоні...")

# 2. Запускаємо інтерфейс (Tkinter) у головному потоці.
try:
    print("Запуск графічного інтерфейсу...")
```

```
    grid_main()
except KeyboardInterrupt:
    print("Зупинено.")
```

Висновок

У ході виконання лабораторної роботи №1 було розроблено комплексну систему комп'ютерного зору, яка складається з апаратної частини (поворотна платформа на базі мікроконтролера) та програмної частини (застосунок на ПК).

Основні результати роботи:

Реалізація алгоритму сканування:

Написано програму для Arduino, яка керує двома сервоприводами.

Реалізовано сканування простору по стовбцях зверху вниз.

Використано вкладені цикли з передумовою (while), що відповідає вимогам варіанту.

Забезпечено сітку сканування розміром не менше 5x5 точок з кроком 18°.

Синхронізація пристроїв:

Налаштовано протокол обміну даними через інтерфейс UART (Serial).

Мікроконтролер передає мітки станів (Start, H, W), що дозволяє комп'ютеру точно визначати моменти, коли камера стабілізована, і виконувати захоплення кадру.

Обробка відео та GUI:

Розроблено скрипт на Python з використанням бібліотеки OpenCV для зчитування відеопотоку та розпізнавання команд синхронізації.

Створено графічний інтерфейс на базі бібліотеки Tkinter для відображення галереї захоплених знімків у реальному часі.

Багатопотоковість:

Вирішено задачу одночасного запуску блокуючих процесів (відеозахват та GUI).

Функцію обробки відео винесено в окремий потік (threading.Thread), тоді як графічний інтерфейс залишено в головному потоці (Main Thread) для забезпечення стабільності роботи застосунку.

Підсумок: Розроблена система успішно виконує автоматичне сканування заданої області, синхронізує рух механіки з моментом зйомки та динамічно оновлює отримані результати на екрані користувача. Отримані навички роботи з серійним портом та потоками є критично важливими для створення складних робототехнічних систем.