Київський політехнічний інститут імені Ігоря Сікорського

Приладобудівний факультет

Кафедра автоматизації та систем неруйнівного контролю

Лабораторна робота №2

з предмету "Комп'ютерний зір"

Студент: Погорєлов Богдан
Група: ПК-51мп

2025 рік

Розробити дві програми, які виконують такі дії з ЦВ:

1. Перша програма посилає команди контролеру поворотних платформ з двома серво приводами, які забезпечують сканування простору зовнішньою цифровою камерою. Ці команди можуть бути у вигляді текстового рядка або у вигляді послідовності байтів. Сканування здійснюється по горизонталі та по вертикалі, кількість точок сканування - не менше 5 х 5. Тип сканування та оператори циклу, які забезпечують це сканування, вказані у таблиці 2. Також ця програма зчитує та відображає ЦВ з зовнішньої цифрової камери та формує мозаїку з Ц3, які отримані з зовнішньої цифрової камери у заданих точках сканування. Після закінчення кожного сканування ця мозаїка записується у файл Ц3, а ім'я цього файлу містить поточний номер сканування.
2. Друга програма, яка записується у контролер поворотних платформ, зчитує команди керування, здійснює відповідні повороти платформи з двома серво-приводами на якій встановлена зовнішня цифрова камера. Після виконання команди ця програма відсилає код підтвердження виконання команди.

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

Лабораторна робота №2 Тема: Керування платформою через послідовний інтерфейс та створення панорамних зображень. Варіант: 12

Завдання:

Програма 1 (PC): Керує скануванням, відправляючи команди. Тип сканування: По стовбцях зверху вниз. Цикли: два цикли з передумовою (while). Формує мозаїку (панораму) та зберігає її у файл.

Програма 2 (Arduino): Приймає координати, повертає сервоприводи та надсилає підтвердження виконання.

1. Програма для контролера (Arduino) Цей скетч очікує команду у форматі Pxx Tyy (Pan, Tilt), де xx та yy — кути. Після завершення руху він відправляє у Serial порт рядок DONE.

```
#include <Servo.h>

Servo servoPan;
Servo servoTilt;

// Піни підключення
const int PIN_PAN = 9;
const int PIN_TILT = 10;

// Буфер для прийому даних
String inputString = "";
bool stringComplete = false;

void setup() {
  Serial.begin(115200);
  servoPan.attach(PIN_PAN);
  servoTilt.attach(PIN_TILT);
  // Початкове положення
  servoPan.write(90);
  servoTilt.write(90);
  // Резервуємо пам'ять для рядка
  inputString.reserve(200);
}

void loop() {
  // Якщо отримано повну команду (зустрівся символ нового рядка)
  if (stringComplete) {
    parseCommand(inputString);

    // Очищення рядка для наступної команди
    inputString = "";
    stringComplete = false;
  }
}
// Функція читання даних з Serial (викликається автоматично)
void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    if (inChar == '\n') {
      stringComplete = true;
```

```
    } else {
      inputString += inChar;
    }
  }
}

void parseCommand(String command) {
  // Очікуваний формат: "P90 T45"
  int pIndex = command.indexOf('P');
  int tIndex = command.indexOf('T');
  if (pIndex != -1 && tIndex != -1) {
    // Вирізаємо значення кутів
    // substring бере текст від позиції P+1 до T
    String panVal = command.substring(pIndex + 1, tIndex);
    // substring бере текст від T+1 до кінця
    String tiltVal = command.substring(tIndex + 1);
    int panAngle = panVal.toInt();
    int tiltAngle = tiltVal.toInt();
    // Обмеження кутів (захист серво)
    panAngle = constrain(panAngle, 0, 180);
    tiltAngle = constrain(tiltAngle, 0, 180);
    // Виконання руху
    servoPan.write(panAngle);
    servoTilt.write(tiltAngle);
    // Час фізичний поворот платформи
    delay(600);
    // !!! Надсилаємо код підтвердження !!!
    Serial.println("DONE");
  }
}
```

2. Програма керування та створення мозаїки (Python) Ця програма реалізує логіку Варіанту 12:

Зовнішній цикл while: перебір стовпців (Pan).

Внутрішній цикл while: перебір рядків у стовпці (Tilt).

Після кожного кадру фото додається до списку.

В кінці стовпця фотографії склеюються вертикально (vstack).

В кінці всього сканування стовпці склеюються горизонтально (hstack).

In [ ]:
```
import cv2
import serial
import time
import numpy as np
import os

# --- НАЛАШТУВАННЯ ---
SERIAL_PORT = 'COM3'  # Змініть на свій порт
BAUD_RATE = 115200

# Параметри сканування (5x5 точок)
```

```python
PAN_START, PAN_END = 45, 135
TILT_START, TILT_END = 45, 135
STEPS_COUNT = 5

# Розрахунок кроку (цілочисельне ділення)
STEP_PAN = (PAN_END - PAN_START) // (STEPS_COUNT - 1)
STEP_TILT = (TILT_END - TILT_START) // (STEPS_COUNT - 1)

# Розмір одного кадру в мозаїці (зменшуємо, щоб не було гігантського фото)
TILE_SIZE = (160, 120)

def wait_for_confirmation(ser):
    """Чекає 'DONE' від Arduino."""
    while True:
        if ser.in_waiting > 0:
            try:
                line = ser.readline().decode('utf-8').strip()
                if line == "DONE":
                    return True
            except:
                pass

def send_command(ser, pan, tilt):
    """Відправляє команду формату 'Pxx Tyy'."""
    cmd = f"P{pan} T{tilt}\n"
    ser.write(cmd.encode())
    print(f" -> Команда: Pan {pan}, Tilt {tilt}")

def main():
    # 1. Підключення
    try:
        ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1)
        time.sleep(2) # Чекаємо перезавантаження Arduino
        print(f"Підключено до {SERIAL_PORT}")
    except Exception as e:
        print(f"Помилка порту: {e}")
        return

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

    print("Початок сканування (Варіант 12: Стовпці, While)...")

    # Списки для зберігання частин мозаїки
    mosaic_columns = []

    # --- ЛОГІКА ВАРІАНТУ 12 ---
    # Зовнішній цикл - Стовпці (Pan)
    current_pan = PAN_START

    while current_pan <= PAN_END:

        # Список для фотографій поточного стовпця
        current_col_imgs = []

        # Внутрішній цикл - Рядки у стовпці (Tilt, зверху вниз)
        current_tilt = TILT_START
```

```
        while current_tilt <= TILT_END:

            # 1. Відправка команди
            send_command(ser, current_pan, current_tilt)

            # 2. Очікування виконання (синхронізація)
            wait_for_confirmation(ser)

            # 3. Захоплення кадру
            # Читаємо кілька разів, щоб очистити буфер камери
            for _ in range(5):
                ret, frame = cap.read()

            if ret:
                # Зменшуємо кадр для мозаїки
                small_frame = cv2.resize(frame, TILE_SIZE)

                # Додаємо текст з координатами (для наочності)
                cv2.putText(small_frame, f"{current_pan},{current_tilt}",
                            (10, 20), cv2.FONT_HERSHEY_SIMPLEX,
                            0.5, (0, 255, 0), 1)

                current_col_imgs.append(small_frame)

                # Показуємо процес
                cv2.imshow("Scanner Process", frame)
                cv2.waitKey(100)

            # Крок внутрішнього циклу
            current_tilt += STEP_TILT

        # --- ФОРМУВАННЯ СТОВПЦЯ ---
        # Склеюємо всі фото стовпця вертикально (axis=0)
        if current_col_imgs:
            col_strip = np.vstack(current_col_imgs)
            mosaic_columns.append(col_strip)

        # Крок зовнішнього циклу
        current_pan += STEP_PAN

    # --- ФОРМУВАННЯ ФІНАЛЬНОЇ МОЗАЇКИ ---
    print("Сканування завершено. Генерація мозаїки...")

    if mosaic_columns:
        # Склеюємо стовпці горизонтально (axis=1)
        final_mosaic = np.hstack(mosaic_columns)

        # Генеруємо ім'я файлу
        scan_num = int(time.time())
        filename = f"scan_result_{scan_num}.jpg"

        # Зберігаємо та показуємо
        cv2.imwrite(filename, final_mosaic)
        print(f"Мозаїка збережена як: {filename}")

        cv2.imshow("Final Mosaic", final_mosaic)
        # Чекаємо клавішу для виходу
        print("Натисніть будь-яку клавішу для виходу.")
        cv2.waitKey(0)
    else:
```

```
        print("Помилка: Не отримано зображень.")

    # Закриття ресурсів
    ser.close()
    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```

Висновок

У ході виконання лабораторної роботи було реалізовано систему дистанційного керування роботизованою платформою з використанням архітектури «Master-Slave», де комп'ютер виступає керуючим пристроєм, а мікроконтролер — виконавчим.

Основні результати роботи:

Зміна парадигми керування:

На відміну від першої роботи, логіку сканування повністю перенесено на бік ПК (Python). Це дозволяє гнучко змінювати параметри сканування (межі, крок, алгоритм) без перепрошивки мікроконтролера.

Розроблено та реалізовано текстовий протокол команд (формат Pxx Tyy), що забезпечує точне позиціонування платформи.

Синхронізація процесів:

Впроваджено механізм підтвердження виконання команд (handshaking). Програма на Python очікує сигнал DONE від Arduino перед зйомкою. Це повністю усуває проблему змазаних кадрів, яка виникає при зйомці під час руху сервоприводів.

Алгоритмічна реалізація (Варіант 12):

Програмно реалізовано алгоритм сканування по стовбцях зверху вниз.

Використано вкладені цикли з передумовою (while), що дозволило організувати послідовний перебір координат відповідно до індивідуального завдання.

Обробка зображень та формування мозаїки:

Реалізовано автоматичне створення панорамного зображення (мозаїки) з окремих кадрів.

Використання бібліотеки NumPy (vstack, hstack) дозволило ефективно об'єднувати масиви пікселів у єдине зображення, що наглядно демонструє результат сканування простору розміром 5x5 точок.

Підсумок: Отримані результати демонструють ефективність використання послідовного інтерфейсу для побудови систем комп'ютерного зору, де необхідна точна координація механічних вузлів та процесів захоплення зображення. Система

дозволяє автоматизувати процес огляду територій або об'єктів та документувати результати у вигляді єдиного файлу.