Servo Control Setup Guide

This guide will help you set up and run the Servo Control project, which demonstrates serial communication between a Raspberry Pi 5 and an Arduino Uno R3 to control an SG90 servo motor.

Table of Contents

Overview

This project allows you to control an SG90 micro servo motor connected to an Arduino Uno R3 from a Raspberry Pi 5 via USB serial communication. The Python script provides an interactive menu with multiple control modes:

- Manual angle control (0-180 degrees)
- Preset positions (0°, 45°, 90°, 135°, 180°)
- Sweep demo
- Smooth movement demo
- Quick center function

Communication occurs at 9600 baud using the protocol: `SERVO:angle\n`

Hardware Requirements

Required Components

- **Raspberry Pi 5** (or other Raspberry Pi model)
- **Arduino Uno R3**
- **SG90 Micro Servo Motor**
- **USB A to USB B cable** (for Raspberry Pi to Arduino connection)
- **Jumper wires** (Male-to-Male, 3 wires minimum)

SG90 Servo Specifications

- Operating Voltage: 4.8V - 6V
- Stall Torque: 1.2kg/cm at 4.8V
- Rotation Range: 0° to 180°

- Operating Speed: 0.12s/60° at 4.8V
- Weight: 9g

Hardware Setup

Servo Motor Wiring

Connect the SG90 servo to the Arduino Uno R3:

```
SG90 Servo Wire     →  Arduino Uno R3

Orange/Yellow       →  Digital Pin 9 (PWM)
Red                 →  5V
Brown/Black         →  GND
```

Power Considerations

- **Light Load Testing**: Arduino's 5V pin can power a single SG90 servo for testing without load
- **Under Load**: Use an external 5V power supply (4.8-6V) when the servo is under load or controlling multiple servos
- **Current Draw**: SG90 can draw 100-250mA when moving under load

USB Connection

Connect the Arduino to the Raspberry Pi using a USB A to USB B cable. The Arduino will appear as `/dev/ttyUSB0` or `/dev/ttyACM0`.

Software Setup

Step 1: Verify Arduino Connection

Open a terminal on your Raspberry Pi and verify the Arduino is detected:

```
# List serial devices (should show ttyUSB0 or ttyACM0)
ls /dev/tty*

# Verify USB device
lsusb
# Should display: "Arduino SA Uno R3" or similar
```

Step 2: Grant Serial Port Permissions

Your user needs permission to access the serial port:

```
# Add user to dialout group
sudo usermod -a -G dialout $USER

# Log out and log back in for changes to take effect
# Or reboot the Raspberry Pi
```

Temporary Alternative (if you can't log out):

```
sudo chmod 666 /dev/ttyUSB0  # Replace with your port
```

Step 3: Set Up Python Virtual Environment

Create and activate a virtual environment in the Servo_Control directory:

```
# Navigate to the Servo_Control directory
cd /home/dom/Serial_Comms/Servo_Control

# Create virtual environment
python3 -m venv venv

# Activate virtual environment
source venv/bin/activate

# Your prompt should now show (venv) at the beginning
```

Step 4: Install PySerial

With the virtual environment activated, install the PySerial library:

```
# Install pyserial
pip install pyserial

# Verify installation
pip list | grep pyserial
# Should show: pyserial    X.X
```

Alternative: Use Existing Virtual Environment

If you prefer to use the parent directory's virtual environment:

```
# From Servo_Control directory
source ../venv/bin/activate

# Verify pyserial is installed
pip list | grep pyserial

# If not installed, run:
pip install pyserial
```

Arduino Setup

Step 1: Install Arduino IDE

If not already installed on your Raspberry Pi:

```
# Update package list
sudo apt update

# Install Arduino IDE
sudo apt install arduino
```

Step 2: Upload the Sketch

1. Open Arduino IDE on your Raspberry Pi
2. Open the sketch: `File > Open > /home/dom/Serial_Comms/Servo_Control/arduino_servo/arduino_`
3. Select your board: `Tools > Board > Arduino AVR Boards > Arduino Uno`
4. Select the port: `Tools > Port > /dev/ttyUSB0` (or `/dev/ttyACM0`)
5. Click the **Upload** button (right arrow icon)
6. Wait for "Done uploading" message

Step 3: Verify Upload

After uploading, the Arduino's built-in LED should flash 3 times, indicating the sketch is ready.

IMPORTANT: Close the Arduino Serial Monitor before running the Python script, as only one program can access the serial port at a time.

Running the Project

Step 1: Activate Virtual Environment

```
cd /home/dom/Serial_Comms/Servo_Control
source venv/bin/activate
```

Step 2: Configure Serial Port (if needed)

If your Arduino appears on a different port:

```
# Check which port your Arduino is on
ls /dev/tty* | grep -E "USB|ACM"
```

Edit `servo_control.py` if needed:

```
SERIAL_PORT = '/dev/ttyUSB0'  # Change to your port
```

Step 3: Run the Script

```
# Make script executable (first time only)
chmod +x servo_control.py

# Run the script
python3 servo_control.py

# Or directly:
./servo_control.py
```

Step 4: Use the Interactive Menu

You'll see a menu with options:

```
==================================================
Servo Control Menu:
==================================================
1. Manual Control (enter specific angle)
2. Preset Positions (0°, 45°, 90°, 135°, 180°)
3. Sweep Demo (0° to 180° and back)
4. Smooth Movement Demo
5. Center Servo (90°)
6. Exit
```

- **Option 1**: Enter any angle from 0-180 degrees
- **Option 2**: Choose from preset positions
- **Option 3**: Watch the servo sweep from 0° to 180° and back
- **Option 4**: See smooth movement from 0° to 180° and back
- **Option 5**: Quickly center the servo at 90°
- **Option 6**: Center servo and exit

Exiting the Program

- Press **Ctrl+C** or select **Option 6** to exit
- The servo will automatically center at 90° before the program closes

---

Troubleshooting

Port Permission Denied

Error: `Permission denied: '/dev/ttyUSB0'`

Solution:

```
# Add user to dialout group
sudo usermod -a -G dialout $USER
# Log out and log back in
```

```
# OR temporary fix:
sudo chmod 666 /dev/ttyUSB0
```

Arduino Not Found

Error: `Serial Error: could not open port /dev/ttyUSB0`

Solutions: 1. Verify Arduino is connected: `ls /dev/tty* | grep -E "USB|ACM"` 2. Check USB cable (try a different one) 3. Try unplugging and reconnecting Arduino 4. Update `SERIAL_PORT` in `servo_control.py` if Arduino is on different port

No Response from Arduino

Problem: Python script connects but servo doesn't move

Solutions: 1. Verify Arduino sketch is uploaded correctly 2. Close Arduino Serial Monitor (only one program can use port at a time) 3. Check servo connections (especially signal wire to Pin 9) 4. Restart Arduino by unplugging/replugging USB

Port Already in Use

Error: `Serial port /dev/ttyUSB0 is already in use`

Solutions:

```
# Find which program is using the port
fuser /dev/ttyUSB0

# Kill the process (replace PID with actual process ID)
kill -9 [PID]

# Or close Arduino Serial Monitor/other Python scripts
```

Servo Jitters or Doesn't Move Smoothly

Problem: Servo movement is erratic

Solutions: 1. Check power supply (use external 5V supply if servo is under load) 2. Verify all connections are secure 3. Check for loose wires 4. Ensure servo is not mechanically obstructed

Garbage Characters in Output

Problem: Random characters instead of messages

Solutions: 1. Verify both Python and Arduino use 9600 baud 2. Reset Arduino (unplug/replug USB) 3. Add buffer clear after connection in Python: `python ser.reset_input_buffer()`

Virtual Environment Issues

Problem: Can't find `pyserial` or Python modules

Solutions:

```
# Ensure virtual environment is activated
source venv/bin/activate

# Reinstall pyserial
pip install --upgrade pyserial

# Verify installation
pip list | grep pyserial
```

Check Python Version

```
python3 --version
# Should be Python 3.7 or higher
```

Additional Resources

Testing the Setup

Test Arduino connection without Python:

1. Open Arduino IDE Serial Monitor (Tools > Serial Monitor)
2. Set baud rate to 9600
3. Type: `SERVO:90` and press Enter
4. Arduino should respond: `Servo moved to 90 degrees`
5. Close Serial Monitor before running Python script

Serial Port Reference

- `/dev/ttyUSB0` - Most common for Arduino Uno on Raspberry Pi
- `/dev/ttyACM0` - Alternative port for Arduino
- `/dev/ttyUSB1` - If multiple USB serial devices connected

Communication Protocol

All commands follow this format:

```
COMMAND:parameter\n
```

Example:

```
SERVO:90\n   # Set servo to 90 degrees
```

Arduino responds with:

```
Servo moved to 90 degrees
```

Project Files

```
Servo_Control/
  servo_control.py           # Main Python control script
  arduino_servo/
     arduino_servo.ino      # Arduino sketch
  venv/                      # Virtual environment (after setup)
  README.md                  # This file
```

Support

If you encounter issues not covered in this guide:

1. Verify all hardware connections
2. Check serial port permissions
3. Ensure Arduino sketch is uploaded correctly
4. Verify virtual environment is activated
5. Check that only one program is accessing the serial port

For project-specific questions, refer to the parent directory's CLAUDE.md file for additional context about the serial communication architecture.