Setup Guide - Raspberry Pi to Arduino Serial Communication

This guide will help you set up and run basic serial communication between a Raspberry Pi and Arduino Uno.

What You'll Need

Hardware: - Raspberry Pi 5 (or similar) - Arduino Uno R3 - USB A to B cable (to connect Arduino to Raspberry Pi)

Software: - Python 3 (pre-installed on Raspberry Pi OS) - Arduino IDE (for uploading sketches to Arduino) - PySerial library (we'll install this)

Step 1: Set Up the Virtual Environment

A virtual environment keeps your Python packages organized and prevents conflicts with other projects.

Create and activate the virtual environment:

```
# Navigate to the project directory
cd /home/dom/Serial_Comms/arduino_serial

# Create a virtual environment (if not already created)
python3 -m venv venv

# Activate the virtual environment
source venv/bin/activate
```

You'll know it's activated when you see `(venv)` at the beginning of your terminal prompt:

```
(venv) dom@raspberrypi:~/Serial_Comms/arduino_serial$
```

Tip: You need to activate the virtual environment every time you open a new terminal window to run the Python scripts.

Step 2: Install PySerial

PySerial is the Python library that enables serial communication.

```
# Make sure your virtual environment is activated first!
pip install pyserial
```

To verify installation:

```
pip list | grep pyserial
```

You should see something like: `pyserial    3.5`

Step 3: Grant Serial Port Access

Your user needs permission to access the serial port.

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

Important: After running this command, you must **log out and log back in** for the changes to take effect.

Alternative (temporary fix):

If you don't want to log out, you can temporarily grant access:

```
sudo chmod 666 /dev/ttyUSB0
# or
sudo chmod 666 /dev/ttyACM0
```

Step 4: Connect Your Arduino

1. Connect the Arduino Uno to your Raspberry Pi using the USB cable
2. Wait a few seconds for the connection to establish

Find your Arduino's port:

```
ls /dev/tty*
```

Look for either: - `/dev/ttyUSB0` (USB to serial adapter) - `/dev/ttyACM0` (Arduino's native USB)

You can also verify with:

```
lsusb
```

You should see: `Arduino SA Uno R3` in the list.

Step 5: Upload the Arduino Sketch

Option A: Basic Receiver (displays messages only)

1. Open the Arduino IDE on your Raspberry Pi
2. Go to `File` → `Open`
3. Navigate to: `/home/dom/Serial_Comms/arduino_serial/arduino_receiver/`

4. Open `arduino_receiver.ino`
5. Select `Tools` → `Board` → `Arduino Uno`
6. Select `Tools` → `Port` → `/dev/ttyUSB0` (or `/dev/ttyACM0`)
7. Click the **Upload** button (→)
8. Wait for "Done uploading" message

Option B: Receiver with LED Indicator (flashes LED when message received)

Follow the same steps as Option A, but open: `/home/dom/Serial_Comms/arduino_serial/arduino_receiver`

Important: After uploading, **close the Arduino Serial Monitor** if it's open. Only one program can access the serial port at a time.

Step 6: Update the Python Script (if needed)

Before running the script, check that the serial port matches your Arduino's port.

1. Open `test_serial.py` in a text editor
2. Find line 30: `SERIAL_PORT = '/dev/ttyUSB0'`
3. If your Arduino is on a different port (like `/dev/ttyACM0`), change this line
4. Save the file

Step 7: Run the Python Script

Make sure: - Your virtual environment is activated: `source venv/bin/activate` - The Arduino sketch is uploaded - The Arduino Serial Monitor is **closed**

Now run:

```
python3 test_serial.py
```

Expected Output:

```
Connecting to Arduino...
Sending: Hello from Raspberry Pi!

Waiting for Arduino response...
Arduino says: Received: Hello from Raspberry Pi!

Communication complete!
```

If you uploaded the LED version, you should also see the Arduino's built-in LED flash 10 times!

Troubleshooting

Error: "Could not open serial port"

Solution 1: Check if the port is correct

```
ls /dev/ttyUSB* /dev/ttyACM*
```

Update `SERIAL_PORT` in `test_serial.py` to match.

Solution 2: Close Arduino Serial Monitor Only one program can use the port at a time.

Solution 3: Grant port permissions

```
sudo chmod 666 /dev/ttyUSB0
# or
sudo chmod 666 /dev/ttyACM0
```

No response from Arduino

- Make sure you waited 2-3 seconds after opening the connection (the script already does this)
- Verify the Arduino sketch is uploaded correctly
- Try unplugging and reconnecting the Arduino
- Restart the Arduino IDE

Receiving garbage characters

- Both devices must use the same baud rate (9600)
- Check that `Serial.begin(9600)` is in the Arduino sketch
- Check that `BAUD_RATE = 9600` is in the Python script

Port is busy

Find and kill any programs using the port:

```
fuser /dev/ttyUSB0
# or
fuser /dev/ttyACM0
```

Next Steps

Once you have basic communication working:

1. Modify the message in `test_serial.py` (line 40)
2. Try sending different messages

3. Experiment with the LED flash pattern (in the Arduino sketch)
4. Explore other examples in the parent directory:
   - `/home/dom/Serial_Comms/Servo_Control/` - Control a servo motor
   - `/home/dom/Serial_Comms/two_way_comms/` - Two-way communication

Quick Reference

Activate virtual environment:

```
source venv/bin/activate
```

Deactivate virtual environment:

```
deactivate
```

Check Arduino connection:

```
ls /dev/tty*
lsusb
```

Run the script:

```
python3 test_serial.py
```

Need Help?

- Check the main project documentation: `/home/dom/Serial_Comms/CLAUDE.md`
- Arduino documentation: https://www.arduino.cc/reference/en/
- PySerial documentation: https://pyserial.readthedocs.io/

Happy coding!