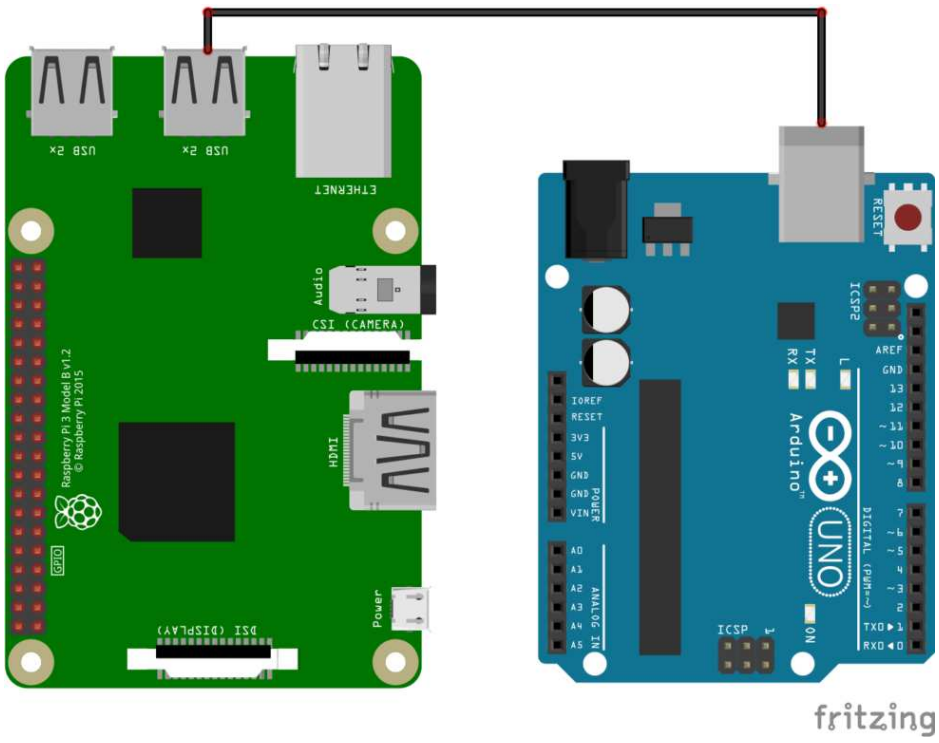
Comunicación en serie entre Raspberry Pi y Arduino

por Xukyo | 27 Sep 2020 | Tutoriales | 0 Comentarios



Etiquetas: Arduino, C/C++, Comunicación serie, Python, Raspberry PI

5 (2)

En algunos proyectos, puede ser interesante establecer una comunicación en serie entre Raspberry Pi y Arduino. Así, es posible acoplar la potencia de computación y las interfaces inalámbricas del Raspberry Pi con las entradas/salidas y la colección de módulos Arduino. El primer ejemplo que viene a la mente es el uso de este sistema para la automatización del hogar en el que el Raspberry Pi albergará la interfaz de control e inteligencia y el Arduino servirá como un PLC que actuará sobre los componentes al final de la cadena (luz, radiador, ventilador, sensores, etc.).

Veremos en este tutorial cómo configurar una comunicación en serie entre Raspberry Pi y Arduino a través del puerto USB. En este artículo utilizamos la tarjeta Arduino UNO pero puede ser adaptada a otros tipos de tarjetas con conexión en serie (Nano, Mega, Feather, EPS32, etc.).

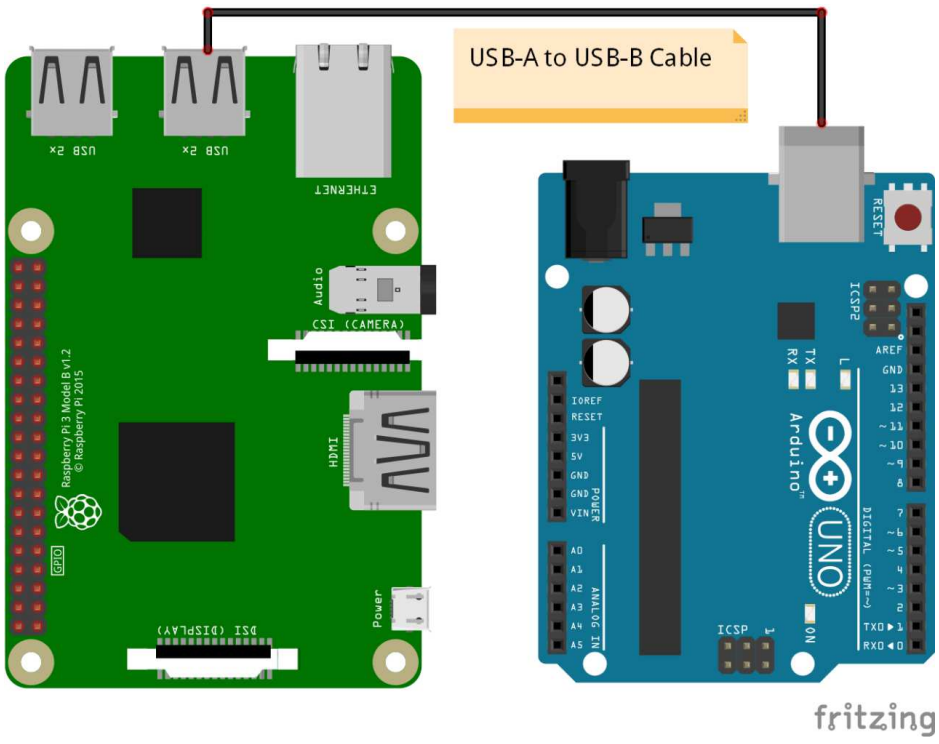
Prerrequisito: Comunicación en serie con Arduino, Acceso remoto a Raspberry Pi con VNC

Hardware

- Computadora
- Arduino UNO
- Raspberry Pi 3B+
- USB A Macho / USB B Macho

Diagrama de conexión

Para establecer una comunicación en serie entre Raspberry Pi y Arduino, simplemente conéctelos con un cable USB adecuado. En nuestro caso, usamos un Raspberry Pi 3B+ y un Arduino UNO. Así que necesitamos un cable USBA macho a USB B macho.



También es posible crear una comunicación en serie utilizando los pines Rx/Tx de ambos componentes. En este tutorial nos centraremos en la conexión USB.

- Tx GPIO14(RPI) <-> Rx 0(Arduino)
- Rx GPIO15(RPI) <-> Tx 1(Arduino)

Configuración del Raspberry Pi

Le recordamos que para poder usar su Raspberry Pi sin pantalla o teclado, la Conexión remota de VNC debe ser configurada.

Para usar la interfaz en serie del Raspberry Pi, debe estar habilitada en el menú de configuración. Para ello, introduzca el siguiente comando en una terminal:

```
sudo raspi-config
```

En el menú, seleccione «5 – Opciones de interfaz» y luego «Serie P6» y valide.

Una vez realizada la conexión, puede comprobar los dispositivos conectados al puerto serie tecleando el comando en la terminal:

```
lsusb
```

El Raspberry Pi devuelve la lista de dispositivos conectados a los puertos USB.

```
pi@raspberrypi:~ $ lsusb
Bus 001 Device 002: ID 2341:0043 Arduino SA Uno R3 (CDC ACM)
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Para encontrar el nombre del puerto en el que está conectado el Arduino, usamos el comando:

```
dmesg | grep "tty"
```

Este comando devuelve los mensajes del sistema relacionados con los puertos serie. Tenías que encontrar el nombre del puerto en los últimos mensajes. En nuestro caso el nombre del puerto es ttyACM0.

```
pi@raspberrypi:~ $ dmesg | grep "tty"
[    0.000000] Kernel command line: coherent_pool=1M 8250.nr_uarts=1
bcm2708_fb.fbwidth=1824 bcm2708_fb.fbheight=984 bcm2708_fb.fbswap=1
smsc95xx.macaddr=B8:27:EB:23:CF:07 vc_mem.mem_base=0x1ec00000
vc_mem.mem_size=0x20000000  console=ttyS0,115200 console=tty1
root=/dev/mmcblk0p7 rootfstype=ext4 elevator=deadline fsck.repair=yes
rootwait quiet splash plymouth.ignore-serial-consoles
[    0.000636] console [tty1] enabled
[    0.951553] 20201000.serial: ttyAMA0 at MMIO 0x20201000 (irq = 81,
base_baud = 0) is a PL011 rev2
[    0.954393] console [ttyS0] disabled
[    0.954466] 20215040.serial: ttyS0 at MMIO 0x0 (irq = 53, base_baud
= 31250000) is a 16550
[    0.954570] console [ttyS0] enabled
[    5.378641] systemd[1]: Created slice system-serial\x2dgetty.slice.
[ 1455.402071] cdc_acm 1-1:1.0: ttyACM0: USB ACM device
[ 1581.980257] cdc_acm 1-1:1.0: ttyACM0: USB ACM device
```

Instalando el IDE de Arduino en Raspberry Pi

Para instalar el IDE de Arduino en el Raspberry Pi, lo mejor es pasar por la terminal. Sólo tienes que introducir las siguientes líneas de código:

```
mkdir ~/Applications
cd ~/Applications
wget https://downloads.arduino.cc/arduino-1.8.9-linuxarm.tar.xz
tar xvJf arduino-1.8.9-linuxarm.tar.xz
cd arduino-1.8.9/
 ./install.sh
rm ../arduino-1.8.9-linuxarm.tar.xz
```

Esto le permitirá programar el Arduino directamente desde el Raspberry Pi.

Código

Código Arduino

La biblioteca utilizada para la comunicación en serie en el lado de Arduino es la misma que la utilizada para comunicarse con el monitor de serie, la biblioteca Serial.h que conocemos bien. Asegúrate de que la velocidad de comunicación es la misma para ambos dispositivos (baudrate=9600), de lo contrario la comunicación no funcionará.

```
String nom = "Arduino";
String msg;

void setup() {
  Serial.begin(9600);
}

void loop() {
  readSerialPort();

  if (msg != "") {
    sendData();
  }
```

```
void readSerialPort() {
  msg = "";
  if (Serial.available()) {
    delay(10);
```

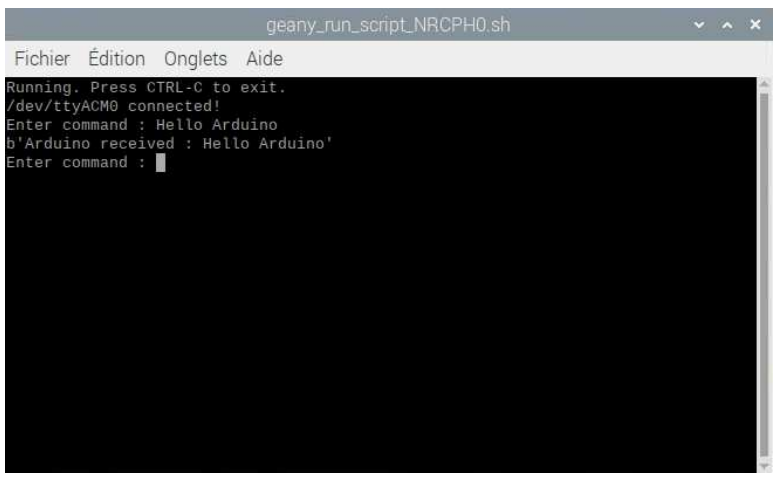
Código Python

En este tutorial, usaremos el lenguaje Python en el lado del Raspberry Pi. La biblioteca que se utiliza para gestionar la comunicación en serie es la biblioteca en serie.

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# lsusb to check device name
#dmesg | grep "tty" to find port name

import serial,time


if __name__ == '__main__':

    print('Running. Press CTRL-C to exit.')
    with serial.Serial("/dev/ttyACM0", 9600, timeout=1) as arduino:
        time.sleep(0.1) #wait for serial to open
        if arduino.isOpen():
            print("{} connected!".format(arduino.port))
            try:
                while True:
                    cmd=input("Enter command : ")
                    arduino.write(cmd.encode())
                    #time.sleep(0.1) #wait for arduino to answer
                    while arduino.inWaiting()==0: pass
                    if  arduino.inWaiting()>0:
                        answer=arduino.readline()
                        print(answer)
                        arduino.flushInput() #remove data after reading
            except KeyboardInterrupt:
                print("KeyboardInterrupt has been caught.")
```

Resultado

El Raspberry Pi envía el comando «Hello Arduino» al Arduino, y el Arduino responde con su nombre y la orden recibida.



Ejemplo práctico

Una vez establecida la comunicación entre Raspberry Pi y Arduino, lo interesante es conducir la E/S de Arduino y recuperar los valores del sensor. En este ejemplo, definiremos diferentes comandos:

- uno para recuperar los valores de los sensores
- uno para encender y apagar el LED en el pin 13

Código Arduino

El Arduino, devuelve datos cuando recibe el comando «datos» y enciende y apaga el LED conectado al pin 13 según los comandos «led0» y «led1».

```
const int ledPin=13;
String nom = "Arduino";
String msg;

void setup() {
  Serial.begin(9600);
}

void loop() {
  readSerialPort();

  if (msg == "data") {
    sendData();
  }else if(msg=="led0"){
    digitalWrite(ledPin,LOW);
    Serial.print(" Arduino set led to LOW");
  }else if(msg=="led1"){
    digitalWrite(ledPin,HIGH);
    Serial.print(" Arduino set led to HIGH");
  }
```

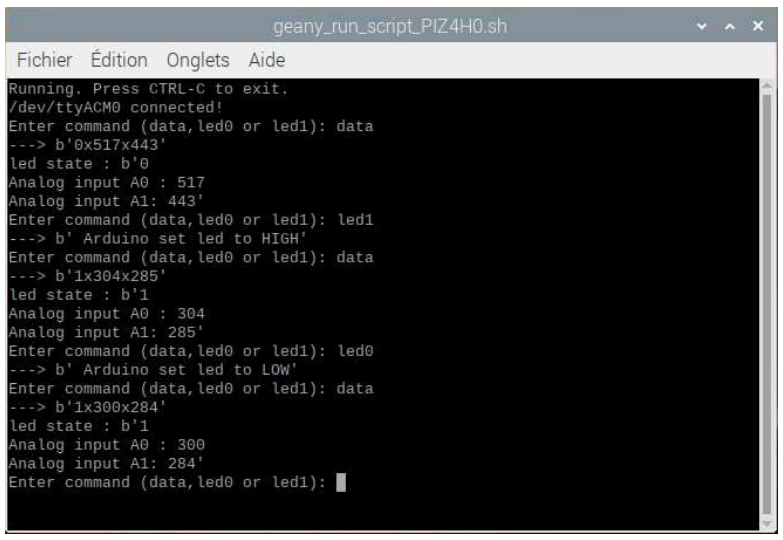
Copy

Código Python

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# lsusb to check device name
#dmesg | grep "tty" to find port name

import serial,time


if __name__ == '__main__':
    print('Running. Press CTRL-C to exit.')
    with serial.Serial("/dev/ttyACM0", 9600, timeout=1) as arduino:
        time.sleep(0.1) #wait for serial to open
        if arduino.isOpen():
```

```
                    arduino.write(cmd.encode())
                    #time.sleep(0.1) #wait for arduino to answer

                    while arduino.inWaiting()==0: pass
                    if  arduino.inWaiting()>0:
                        answer=str(arduino.readline())
                        print("---> {}".format(answer))
                        if cmd=="data":
                            dataList=answer.split("x")
                            print("led state : {}".format(dataList[0]))
                            print("Analog input A0 : {}".format(dataList[1]))
                            print("Analog input A1: {}".format(dataList[2]))
```

Una vez que los dos códigos han sido cargados y lanzados, se puede observar que cuando se introduce el comando «datos» en la terminal, Arduino devuelve muchos bytes que contienen los valores del sensor. Es posible separar esta respuesta en una lista usando la función split() y el carácter «x» y así recuperar los valores del sensor y el estado del led. Gracias a este código, podemos controlar el estado del LED del pin 13.



Modificando este código, podrás conducir y observar cualquier entrada/salida de Arduino en el Raspberry Pi.

Application

- Crear una interfaz gráfica de usuario (GUI) bajo el Raspberry Pi para manejar un Arduino y recuperar los valores del sensor

Sources

- Librería pySerial
- Comunicación en serie con Arduino
- Acceso remoto a Raspberry Pi con VNC

Enviar comentario

Tu dirección de correo electrónico no será publicada. Los campos obligatorios están marcados con *

Comentario

Nombre *

Correo electrónico *

Web

Enviar comentario

- CGV
- Fritzing