

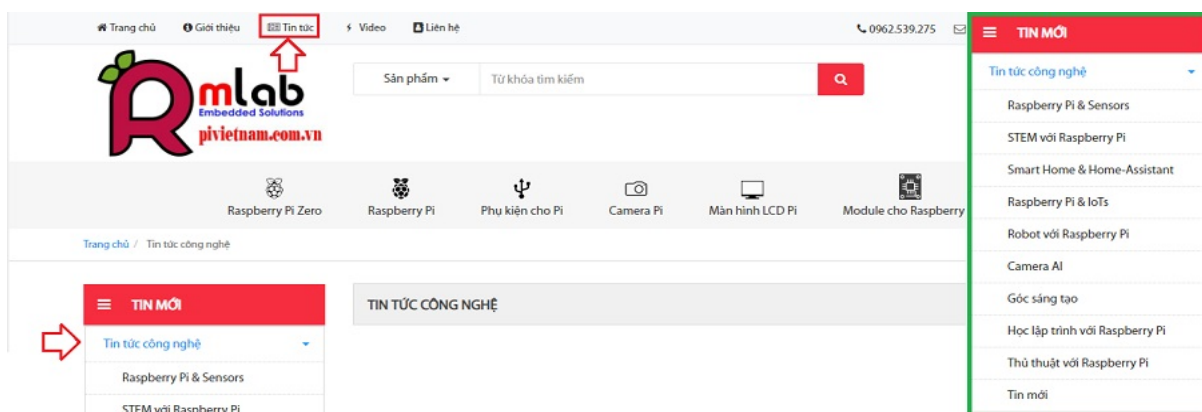
Bài 4 : Lập trình Raspberry Pi sử dụng cổng truyền thông UART

CHÚ Ý : Từ 2019 MLAB có thêm một website cho riêng Raspberry Pi và trở thành website chính về Raspberry Pi tại MLAB, các thông tin về sản phẩm - tin tức cập nhật về Raspberry Pi - Bài viết kỹ thuật hỗ trợ cho Raspberry Pi, ... MLAB cập nhật tại website : pivietnam.com.vn

MLAB trân trọng thông báo tới quý khách hàng!!!



Các bạn có thể tham khảo các bài viết hỗ trợ kỹ thuật và các tin tức mới nhất tại phần "tin tức" trên website PIVIETNAM.COM.VN



Bài viết hỗ trợ kỹ thuật tại website PIVIETNAM.COM.VN - Bài 4: Lập trình Raspberry sử dụng cổng truyền thông ([Link here](#))

Giao tiếp qua cổng UART với Raspberry Pi

1. Giới thiệu UART trên Raspberry Pi

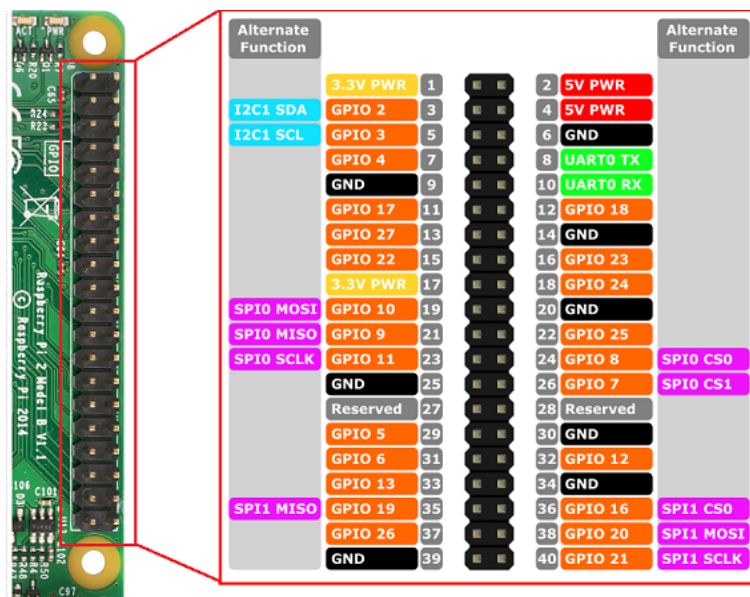
Raspberry Pi hỗ trợ chuẩn Uart trên chân BCM14 (TX) và BCM15 (RX). Mặc định uart được sử dụng làm serial console; nó sẽ gửi toàn bộ thông tin của kernel trong quá trình Pi khởi động. Bạn có thể kết nối trực tiếp uart của pi thông qua mạch USB-TTL để xem thêm. Để sử dụng Uart với mục đích riêng bạn phải giải phóng chân Uart. Lưu ý với Raspberry Pi 3 Uart còn được sử dụng để kết nối bluetooth, và chân BCM14 và BCM15 trở thành mini uart port.

Để Setup UART cho Pi dùng trong kết nối với các MCU khác, các bạn có thể tham khảo hướng dẫn trong video sau :

[Học Raspberry Pi] Video 6 : Setup UART trên Raspberry Pi, demo kết n...



2. Giao tiếp UART với Raspberry Pi



Hình 1 : chân GPIO của Raspberry Pi 2 model B

Dưới đây là hai ví dụ đơn giản giao tiếp giữa Raspberry và Arduino Vn01:

- Ví dụ 1 : Gửi chuỗi ký tự từ Raspberry tới Arduino.
- Ví dụ 2 : Gửi chuỗi ký tự từ Arduino tới Raspberry.

Thông tin : [raspberrypi 2 model B](#) và [Arduino Vn01](#)

2.1 Kết nối phần cứng

Kết nối giữa Arduino và Pi như sau : RX của arduino <-> TX của Pi và ngược lại TX của arduino <-> RX của Pi, kết nối chung 2 chân GND với nhau.

2.2.1 Lập trình giao tiếp UART

Chuẩn bị thư viện cho C là [wiring-pi](#) như trong bài 1 đã đề cập tới.

Thư viện cho python là [pySerial](#)

```
sudo apt-get install python-serial
```

Khi cần dùng thư viện chỉ cần thực hiện "import serial"

a. Lập trình bằng C

Ví dụ 1

Chương trình "uart-send.c" trên Raspberry

```
// Compile : gcc -Wall uart-send.c -o uart-send -lwiringPi

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>

int main() {

    int fd;

    printf("Raspberry's sending : \n");

    while(1) {
        if((fd = serialOpen ("/dev/ttyAMA0", 9600)) < 0 ){
            fprintf (stderr, "Unable to open serial device: %s\n", strerror (errno));
        }
        serialPuts(fd, "hello");
        serialFlush(fd);
        printf("%s\n", "hello");
        fflush(stdout);
        delay(1000);
    }
    return 0;
}
```

Chương trình trên Pi sẽ gửi chuỗi ký tự “hello” thông qua uart tới arduino. Nó sẽ gửi liên tục sau 1s.

Thư viện sử dụng : #include <wiringSerial.h>

- Mở cổng kết nối Serial.

```
int serialOpen (char *device, int baud);
```

Luôn nhớ rằng cổng uart có tên là ttyAMA0. Hàm này sẽ trả về file-descriptor. Nếu file-descriptor lỗi sẽ có giá trị là -1.

- Hàm gửi dữ liệu

```
Void serialPuchar (int fd, unsigned char c) ;
```

Hàm này gửi sẽ gửi 1 byte. Nếu muốn gửi một mảng (hoặc chuỗi ký tự) có thể sử dụng hàm sau.

```
void serialPuts (int fd, char *s) ;
```

- Hàm void serialFlush (int fd) : chờ đến khi dữ liệu được gửi xong hoặc bỏ qua tất cả dữ liệu được gửi tới. Hàm này rất cần với uart, nếu không có sẽ không thể gửi chính xác được. Bạn nên bỏ hàm này đi và xem thử. Có lẽ đây là một bug nhỏ đối với chương trình này vì đáng lý ra chương trình sau khi thực hiện lệnh gửi dữ liệu có thể thực hiện ngay lập tức lệnh tiếp theo, dữ liệu sẽ được uart tự động ngắt gửi đi. Chương trình không cần phải chờ.

Các hàm được hỗ trợ : `wiringPi`

Để compile được chương trình thực hiện lệnh

```
gcc -Wall uart-send.c -o uart-send -lwiringPi
```

Chương trình “uart-receive.ino” trên Arduino

```
void setup() {
    Serial.begin(9600);    // opens serial port, baudrate : 9600 bps
}

void loop() {

    if (Serial.available() > 0) {
        String str = Serial.readString();
        Serial.print("Received: ");
        Serial.println(str);
    }
}
```

- Khởi tạo Uart :

```
Serial.begin(9600);
```

Lưu ý rằng baudrate bằng 9600 giống như trên Pi.

- Hàm kiểm tra xem có dữ liệu tới uart không :

```
Serial.available()
```

Hàm này trả về số byte nhận được.

- Đọc dữ liệu chuỗi ký tự :

```
Serial.readString();
```

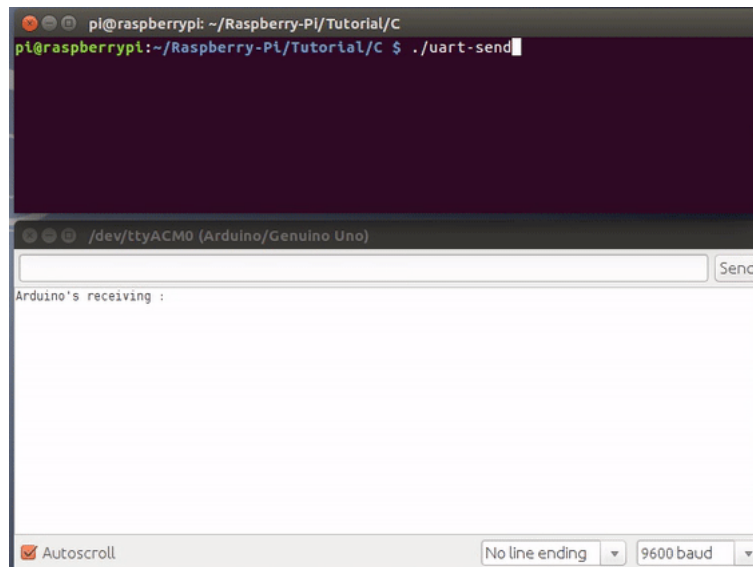
Trả về dữ liệu kiểu String.

- Lưu ý:

```
Serial.print();
```

Mục đích để hiển thị lên màn hình kết quả, nhưng nó gửi qua Uart nên cũng đồng thời nó gửi dữ liệu tới Pi.

Kết quả :



Ví dụ 2

Chương trình “uart-receive.c” trên Raspberry

```
// Compile : gcc -Wall uart-receive.c -o uart-receive -lwiringPi

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>

int main() {

    int fd;
    char c;
    printf("Raspberry's receiving : \n");

    while(1) {
        if((fd = serialOpen ("/dev/ttyAMA0", 9600)) < 0 ){
            fprintf (stderr, "Unable to open serial device: %s\n", strerror (errno)) ;
        }else{
            do{
                c = serialGetchar(fd);
                printf("%c",c);
                fflush (stdout);
            }while(serialDataAval(fd));
        }
    }
    return 0;
}
```

- Kiểm tra xem có dữ liệu tới không :

```
int serialDataAval (int fd);
```

Hàm này trả về số byte dữ liệu xuất hiện. Trả về -1 nếu bị lỗi.

- Đọc dữ liệu

```
int serialGetchar (int fd);
```

Trả về một byte dữ liệu tới. Con trỏ của nó sẽ tự tăng sau mỗi lần gọi. Hàm sẽ chờ tới tận 10s nếu không có dữ liệu. Trả về -1 nếu bị lỗi.

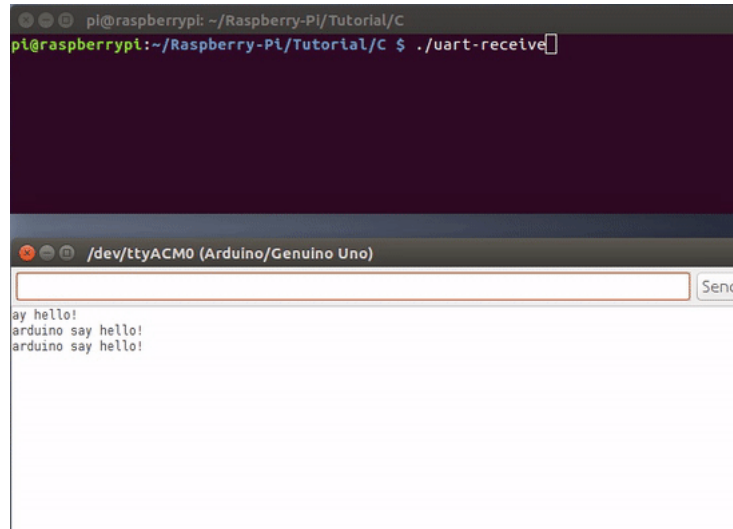
Chương trình “uart-send.ino” trên Arduino

```
void setup() {
```

```
// put your setup code here, to run once:
Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("arduino say hello!");
  delay(1000);
}
```

Kết quả :



b. Lập trình Python

Xem một số ví dụ về thư viện `serial`

Ví dụ 1

Chương trình “uart-send.py” trên Raspberry

```
#!/usr/bin/python3

import time
import serial

ser = serial.Serial(
    port = '/dev/ttyAMA0',
    baudrate = 9600,
    parity = serial.PARITY_NONE,
    stopbits = serial.STOPBITS_ONE,
    bytesize = serial.EIGHTBITS,
    timeout = 1
)

print("Raspberry's sending : ")

try:
    while True:
        ser.write(b'hehe')
        ser.flush()
        print("hehe")
        time.sleep(1)
except KeyboardInterrupt:
    ser.close()
```

- Thiết lập serial

```
serial.Serial()
```

Nếu không thể khởi tạo Serial được sau timeout thì sẽ thông báo lỗi. Ở đây đặt timeout = 1s.

- Gửi dữ liệu :

```
int write(data);
```

Hàm này có thể gửi nhiều byte dữ liệu cùng lúc. Hàm trả về là số byte đã gửi.

- Hàm flush()

Cũng như với chương trình C, cần flush nếu không serial sẽ không gửi đúng.

- Đóng Serial

```
close();
```

Nếu có interrupt xảy ra, ví dụ như CTR+C thì chương trình sẽ bắt được sự kiện đó và sẽ đóng serial. Lí do vì Serial sẽ không tự đóng lại khi chương trình bị ngắt không kiểm soát. Như vậy nó sẽ tiếp tục được mở và không cho phép chương trình nào khác sử dụng nó → chương trình chạy sẽ bị lỗi ở lần tiếp theo.

Đây là python version 3 nên khi chạy bạn thực hiện lệnh

```
python3 uart-send.py
```

Kết quả tương tự như chương trình C

Ví dụ 2

Chương trình “uart-receive.py” trên Raspberry

```
#!/usr/bin/python3

import time
import serial

ser = serial.Serial(
    port = '/dev/ttyAMA0',
    baudrate = 9600,
    parity = serial.PARITY_NONE,
    stopbits = serial.STOPBITS_ONE,
    bytesize = serial.EIGHTBITS,
    timeout = 1
)

print("Raspberry's receiving : ")

try:
    while True:
        s = ser.readline()
        data = s.decode()
        data = data.rstrip()
        print(data)
        # decode s
        # cut "\r\n" at last of string
        # print string
except KeyboardInterrupt:
    ser.close()
```

- Đọc dữ liệu

```
readline()
```

Hàm trả về là những byte đọc được. Hàm đọc kết thúc với timeout=1 được thiết lập bên trên.

- Lưu ý :

Ở trên có sử dụng 2 hàm dưới đây trước khi hiển thị kết quả

```
data=s.decode()
```

```
data=data.rstrip()
```

Mọi người nên kiểm tra kết quả khi không có 2 hàm trên để hiểu thêm về kết quả nhận được.

Kết quả tương tự với chương trình C

3. Sử dụng cổng USB để kết nối

Uart được Raspberry Pi với mục đích log cho kernel. Nếu không thiếu thốn hay bắt buộc phải sử dụng cổng uart các bạn có thể chuyển sang cổng usb để thay thế. Với hệ điều hành linux các kênh xuất nhập đều có thể kết nối thông qua file-descriptor, dù uart hay usb thì chỉ cần thay đổi file-descriptor thì có thể kết nối tới nó. Toàn bộ code bên trên có thể dùng kết nối qua cổng USB, chỉ cần thay “ttyAMA0” bằng tên cổng USB có dạng “ttyUSB”. Các bạn có thể kết nối arduino với thiết bị USB-TTL như PL2303 và sau đó cắm thiết bị vào Pi qua cổng USB là có thể kết nối. Lưu ý đây là sử dụng cổng USB để kết nối chứ không phải kết nối hoàn toàn bằng giao thức USB vì phía bên arduino sử dụng phương thức uart.

Khi PL2303 kết nối tới Raspberry, cần kiểm tra thiết bị đang nằm trên cổng USB nào

```
dmesg | grep tty
```

Kết quả sẽ trả về các Serial trên Pi. Trong đó sẽ có kết quả dạng như bên dưới:


```
pl2303 converter now attached to ttyUSB0
```

PL2303 đang kết nối tới cổng ttyUSB0.

Kết quả chương trình sẽ giống hệt như với kết nối trực tiếp với Uart.


Thông tin tham khảo :

- [1] <https://electrosome.com/uart-raspberry-pi-python/>
- [2] http://elinux.org/RPi_Serial_Connection
- [3] <http://www.briandorey.com/post/Raspberry-Pi-3-UART-Overlay-Workaround>
- [4] http://pyserial.readthedocs.io/en/latest/pyserial_api.html

NGUYỄN VĂN ĐỨC | 


10/23/2020

Mong anh sớm ra video giao tiếp không dây giữa raspberry với esp8266 qua UDP

Nguyễn Văn Đạt | 


10/23/2020

Mong anh làm video về giao tiếp Raspberry với esp8266 ạ.

MLAB | 


01/21/2019

Hì bạn Phát. Bạn ơi, cổng serial - uart chỉ được tham gia một mối liên kết thôi bạn ạ. Bạn đã mở 1 liên kết rồi, nếu muốn sử dụng thêm liên kết khác thì bạn phải đóng liên kết cũ lại. Bạn kiểm tra lại xem sao bạn nhé.

phát | 

01/02/2019

Mình chạy code từ adruino gửi lên Pi thì bình thường. Nhưng nếu để yên 1 khoảng thời gian thì nó sẽ báo lỗi trên màn hình terminal của Pi: "Unable to open serial device: Too many open files". Không biết là có cách nào để sửa không vì mình đọc giá trị cảm biến gửi liên tục

TRẦN XUÂN BÁCH | 

01/06/2018

thay đổi file file-descriptor để sử dụng usb-ttl như thế nào vậy ạ?

123>>

Hiện thị từ 1 đến 5 trong tổng số 11 (3 trang)

Viết đánh giá

Họ và tên:

Đánh giá của bạn:

Lưu ý: Không hỗ trợ HTML!

Bình chọn: Dở Hay

Nhập mã bảo vệ:

760219

Tiếp tục

TRANG CHỦ LIÊN HỆ CHÍNH SÁCH BẢO HÀNH CHÍNH SÁCH BẢO MẬT THÔNG TIN CHÍNH SÁCH VẬN CHUYỂN VÀ GIAO NHẬN CHÍNH SÁCH ĐỔI TR

Công ty TNHH MLAB
Số chứng nhận kinh doanh: 0106356768. Nơi cấp: Sở kế hoạch và đầu tư Thành Phố Hà Nội. Ngày cấp: 07/11/2013
Trụ sở : Số 30F9 - Ngõ 104 Lê Thanh Nghị - Hai Bà Trưng - Hà Nội
Email mua bán hàng: smarttechn.vn.group@gmail.com
Email hỗ trợ kỹ thuật : mlab.services.tech@gmail.com
website:https://mlab.vn
Số điện thoại: 02436231170 hoặc 0984058846 hoặc 0866828846

https://mlab.vn/index.php?_route_=bai-viet-ki-thuat/hoc-raspberry-pi/15129-bai-4-lap-trinh-raspberry-pi-su-dung-cong-truyen-thong-uart.html 7/7