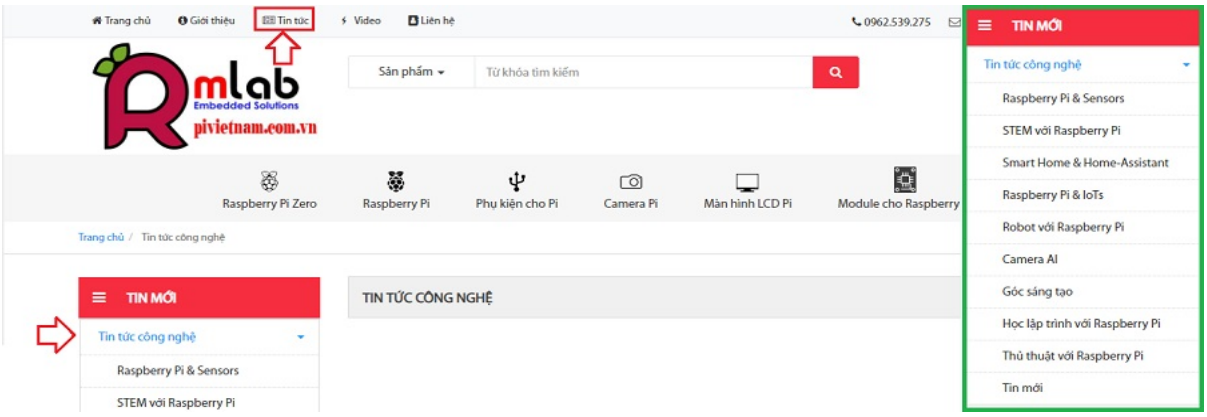
Bài 3 : Lập trình Raspberry giao tiếp SPI

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 PVIETNAM.COM.VN



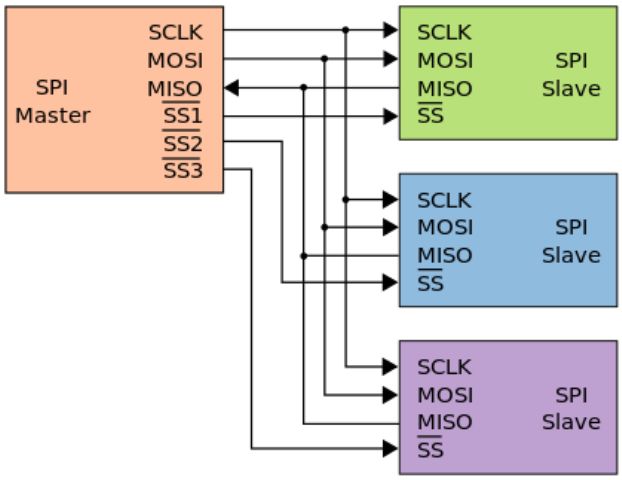
Bài viết hỗ trợ kỹ thuật tại website PIVIETNAM.COM.VN - Bài 3: Lập trình Raspberry giao tiếp SPI (Link here)

1. Giới thiệu SPI

SPI (Serial Peripheral Bus) là một chuẩn truyền thông nối tiếp tốc độ cao do hang Motorola đề xuất. Đây là kiểu truyền thông Master-Slave, trong đó có 1 chip Master điều phối quá trình tuyền thông và các chip Slaves được điều khiển bởi Master vì thế truyền thông chỉ xảy ra giữa Master và Slave. SPI là một cách truyền song công (full duplex) nghĩa là tại cùng một thời điểm quá trình truyền và nhận có thể xảy ra đồng thời.

1.1. Mô hình kết nối

Mô hình gồm có các chân kết nối :



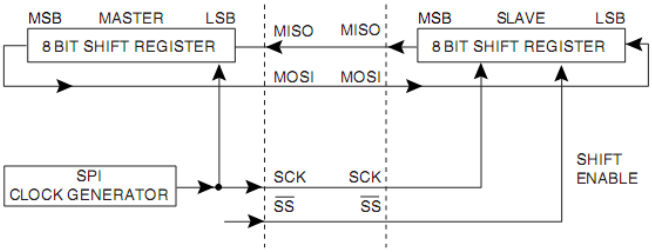
- MOSI hay SI – cổng ra của bên Master (Master Out Slave IN). Đây là chân dành cho việc truyền tín hiệu từ thiết bị chủ động đến thiết bị bị động.

- MISO hay SO – Công ra bên Slave (Master IN Slave Out). Đây là chân dành cho việc truyền dữ liệu từ Slave đến Master.

- SCLK hay SCK là tín hiệu clock đồng bộ (Serial Clock). Xung nhịp chỉ được tạo bởi Master.

- CS hay SS là tín hiệu chọn vi mạch (Chip Select hoặc Slave Select). SS sẽ ở mức cao khi không làm việc. Nếu Master kéo SS xuống thấp thì sẽ xảy ra quá trình giao tiếp. Chỉ có một đường SS trên mỗi slave nhưng có thể có nhiều đường điều khiển SS trên master, tùy thuộc vào thiết kế của người dùng.

Mỗi master và slave có một thanh ghi 8 bit. Cứ mỗi xung nhịp do Master tạo ra trên đường giữ nhịp SCK, một bit trong thanh ghi dữ liệu của Master được truyền qua Slave trên đường MOSI, đồng thời một bit trong thanh ghi dữ liệu của chip Slave cũng được truyền qua Master trên đường MISO. Do 2 gói dữ liệu trên 2 chip được gởi qua lại đồng thời nên quá trình truyền dữ liệu này được gọi là “song công”.



Việc đọc và nhận cũng chỉ trên duy nhất một thanh ghi. Khi master muốn đọc 1 byte dữ liệu thì master phải gửi 1 byte đi trước. muốn đọc n bytes thì phải gửi đúng n bytes.

1.2. So sánh với I2C

Chuẩn giao tiếp I2C chỉ cần 2 dây là có thể giao tiếp trong khi chuẩn SPI cần tới ít nhất 3-4 dây để có thể hoạt động. Lắp đặt phần cứng I2C sẽ trở nên dễ dàng thuận tiện hơn. I2C có thể kết nối với số lượng lớn (7bit/ 10bit địa chỉ) mà không cần thêm kết nối, SPI muốn thêm slave cần thêm chân nối SS, càng lớn sẽ càng trở nên phức tạp

Về tốc độ thì SPI tỏ ra vượt trội so với I2C. SPI là full-duplex, không giới hạn tốc độ tối đa thường hơn 10Mbps. Trong khi đó I2C giới hạn tốc độ thông thường 1Mbps nếu ở fasst mode và 3.4Mbps ở High speed mode

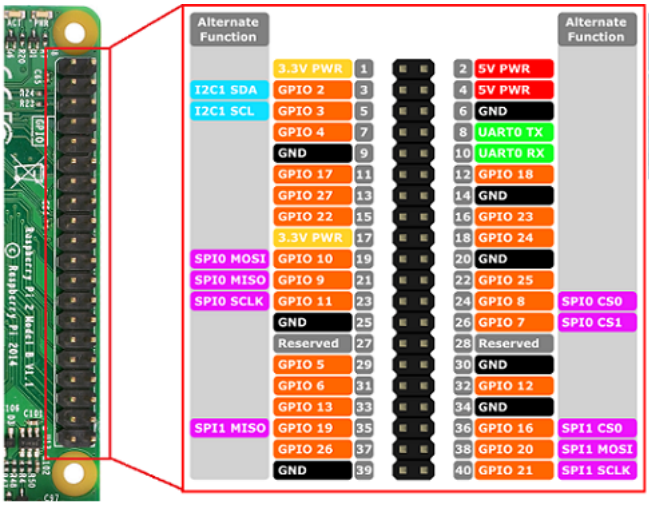
2. Giao tiếp SPI cho Raspberry Pi

Raspberry có hỗ trợ 2 cổng SPI là SPI-0 và SPI-1. Tại SPI-0 có 2 chân CS0 và CS1 có thể giao tiếp với 2 slave còn tại SPI-1 chỉ có duy nhất 1 chân CS0 nên chỉ giao tiếp được 1 slave.

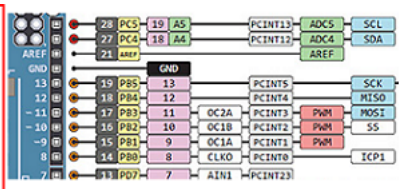
Trong phần này sẽ hướng dẫn giao tiếp Pi với Arduino. Các bạn cần chú ý rằng cả 2 vi điều khiển này đều hướng tập trung là master, nên thiết lập slave trở nên phức tạp hơn. Ở đây sẽ thiết lập Master là Pi và Slave là Arduino

2.1. Kết nối phần cứng

Các bạn kết nối phần cứng giữa Pi và Arduino



Raspberry PI



Arduino

Raspberry	Arduino
MOSI	MOSI
MISO	MISO
CSO	SS
SCK	SCK
GND	GND

2.1. Lập trình giao tiếp

Cũng giống như I2C và một số giao tiếp khác, Raspbian mặc định không mở kích hoạt cho SPI. Chúng ta sẽ mở cổng SPI trước.

- Mở terminal lên gõ lệnh Raspi-config

```
sudo raspi-config
```

Lựa chọn Advanced options → SPI → Yes → Finish. Sau đó reboot

- Kiểm tra lại thiết lập

```
ls /dev/*spi*
```

Kết quả sẽ hiện ra :

```
/dev/spidev0.0  /dev/spidev0.1
```

- Cài đặt thư viện cho SPI (thư viện spi cho python)

```
sudo apt-get install python-dev
sudo apt-get install python3-dev
sudo apt-get update
sudo apt-get install git
git clone https://github.com/doceme/py-spidev.git
cd py-spidev
sudo python setup.py install
sudo python3 setup.py install
```

Dưới đây sẽ là 2 bài tập giao tiếp SPI

Bài tập 1 : Gửi dữ liệu từ Rasp qua Arduino

Bài tập 2 : Rasp nhận dữ liệu từ qua Arduino

a) Lập trình với C

Bài tập 1

Chương trình Master **"spi-send.c"** cho Raspberry Pi

Lựa chọn Port SPI – 0 và chip select SS0 nằm trên GPIO8

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringPiSPI.h>
```

```
#define chanel 0
#define speed 500000
#define SS0     8       // GPIO 8

int main(){
        unsigned char buff[4] = "abc\n";
        pinMode(SS0, OUTPUT);
        digitalWrite(SS0, 1);
        int fd;
        if( (wiringPiSPISetup(chanel, speed)) <0){
                fprintf (stderr, "SPI Setup failed: %s\n", strerror (errno));
                return 0;
        }

        fd = wiringPiSPIGetFd (0) ;

        while(1){
                printf("%s",buff);
                fflush(stdout);
                digitalWrite(SS0, 0);
                write (fd, buff, 4) ;
                digitalWrite(SS0, 1);
                delay(1000);
        }
        return 0;
}
```

- Thiết lập SPI-0 :

  wiringPiSPISetup(chanel, speed): chanel là 0 chọn port 0, chanel 1 chọn port 1. Speed là tốc độ của SPI (500,000 - 32,000,000). Nó trả về file-descriptor với giá trị -1 nghĩa là bị lỗi.

  Vi xung nhịp chỉ được tạo bởi master nên tốc độ cũng do master quyết định. Chú ý rằng tốc độ này không được vượt quá tốc độ của arduino (Các bạn xem bên dưới)

- Gắn file-descriptor:

  fd = wiringPiSPIGetFd (chanel) : file-descriptor được dùng để đại diện cho input/output của linux.Ta sẽ dùng file này để dùng các hàm gửi/nhận thông qua nó.

- Gửi và nhận :

  write(int fd, const void *buf, size_t count);

  read(int fd, void *buf, size_t count);

  fd : là file-descriptor, buf là mảng kí tự, count là số byte gửi/ nhận.

- Trong quá trình gửi và nhận phải đưa chân SS xuống thấp, sau đó được đưa lên mức cao để kết thúc.

Các hàm được hỗ trợ : wirinpi - spi

Chương trình Slave cho Arduino

```
#include <SPI.h>

void setup (void){
    Serial.begin (9600);
    // chon che do slave
    SPCR |= bit (SPE);
    // chuyen MISO sang master in, *slave out*
    pinMode(MISO, OUTPUT);
    // bat che do interrupts
    SPI.attachInterrupt();
    Serial.println("Receive via SPI: ");

}

// SPI interrupt routine

ISR (SPI_STC_vect){
    byte c = SPDR;
    Serial.write(c);
}

void loop (void){
    delay(10);
}
```

- Chọn chế độ slave cho arduino

SPCR |= bit (SPE);

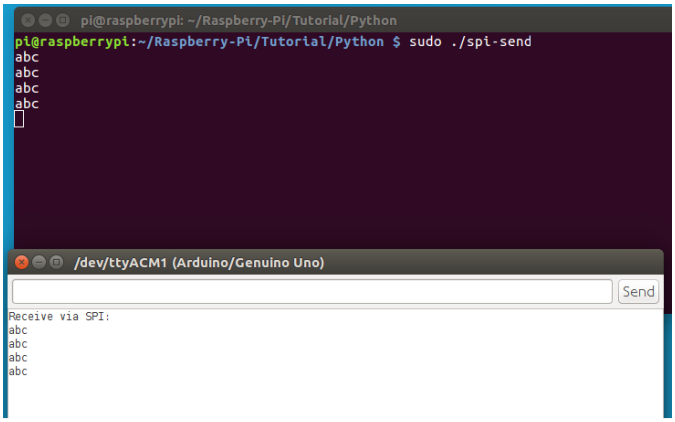
pinMode(MISO, OUTPUT);

- Gắn hàm interrupt cho SPI. Khi Master bắt đầu gửi dữ liệu, Arduino phát hiện interrupt và gọi hàm xử lý.

- Mỗi byte từ Master gửi tới sẽ được gửi lên màn hình qua serial.

Arduino uno sử dụng thạch anh 16Mhz thì SPI có thể đạt tới 16,000,000Hz

Kết quả



Bài tập 2

Thay thế chương trình Master **"spi-send.c"** của Raspberry như sau : **"spi-send-receive.c"**

```
while(1){
    printf("Send:%s",buff);
    fflush(stdout);
    digitalWrite(SS0, 0);
    write (fd, buff, 1);
    delayMicroseconds(20);
    read (fd, c, 1);
    digitalWrite(SS0, 1);
    printf(",Receive:%s\n",c);
    fflush(stdout);

    delay(1000);
}
```

- Master sẽ gửi 1 byte tới Arduino sau đó sẽ đọc 1 byte trả về. (byte trả về này nằm trên thanh ghi)

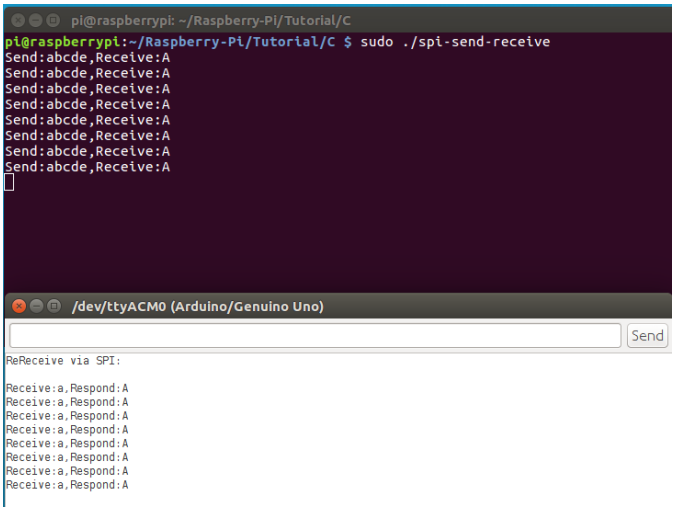
Thay thế chương trình interrupt của Slave như sau :

```
ISR (SPI_STC_vect){
    byte c = SPDR;
    if(c != NULL){
        byte C = c - 32;
        SPDR = C;
        Serial.print("\nReceive:");
        Serial.write(c);
        Serial.print(",Respond:");
        Serial.write(C);
    }
}
```

- Khi nhận 1 byte từ Master là một ký tự. Slave sẽ chuyển ký tự đó sang chữ viết hoa bằng cách trừ đi 32 trong bảng ASCII.

- Sau đó ghi ký tự đó vào thanh ghi.

Kết quả



b) Lập trình Python

Bài tập 1

Chương trình Master **"spi-send.py"** cho Raspberry Pi

```
#!/usr/bin/python3

import spidev
import time
import binascii

spi = spidev.SpiDev()        # Tạo đối tượng cho SPI
spi.open(0, 0)               # mở port 0, device (CS) 0
spi.max_speed_hz = 500000

try:
    while True:
        print("abc")
        resp = spi.xfer2([0x61,0x62,0x63,0xA]) # ['a','b','c','\n']
        time.sleep(1)                          # sleep for 1 seconds
except KeyboardInterrupt:
        spi.close()
```

-Thiết lập SPI-0 :

spi = spidev.SpiDev()

spi.open(0, 0)

spi.max_speed_hz = 5000

- Gửi / nhận dữ liệu :

xfer2([val1,val2,val3]) : gửi dữ liệu dưới dạng list.

readbytes(n) : đọc n bytes dữ liệu, kết quả trả về dạng list

Thư viện SPI sẽ tự động lựa chọn chân SS qua hàm open(). Và tự động xuống mức thấp để gửi byte và đưa lên cao để kết thúc.

- [0x61,0x62,0x63,0xA] : dữ liệu dưới dạng hex của ['a','b','c','\n']

Các hàm được hỗ trợ : python-spi

Kết quả tương tự như chương trình C.

Chú ý : nếu mọi người copy trực tiếp code python từ trang web rất có thể có khả năng lỗi sai "khoảng cách" - những câu lệnh trong cùng một khối bắt buộc phải có khoảng cách căn dòng dấu nhau (gồm dấu cách, dấu tab).

Bài tập 2

Chương trình Master **"spi-send-receive.py"** cho Raspberry Pi

```
#!/usr/bin/python3

import spidev
import time
import binascii
```

```
spi = spidev.SpiDev() # create spi object
spi.open(0, 0)# open spi port 0, device (CS) 0
spi.max_speed_hz = 500000

try:
        # ham code chinh o day
    while True:
        print("send:a,",end="")
        resp = spi.xfer2([0x61]) # transfer one byte
        c = spi.readbytes(1)
        for x in c:
                print ("receieve:",chr(x))
        time.sleep(1) # sleep for 0.1 seconds
except KeyboardInterrupt:
        spi.close()
```

Kết quả tương tự như chương trình C

tran anh pha |
11/04/2018

khá tốt tai lieu hay nên tiep tuc nhu the nay

Hiển thị từ 1 đến 1 trong tổng số 1 (1 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:

Tiếp tục