

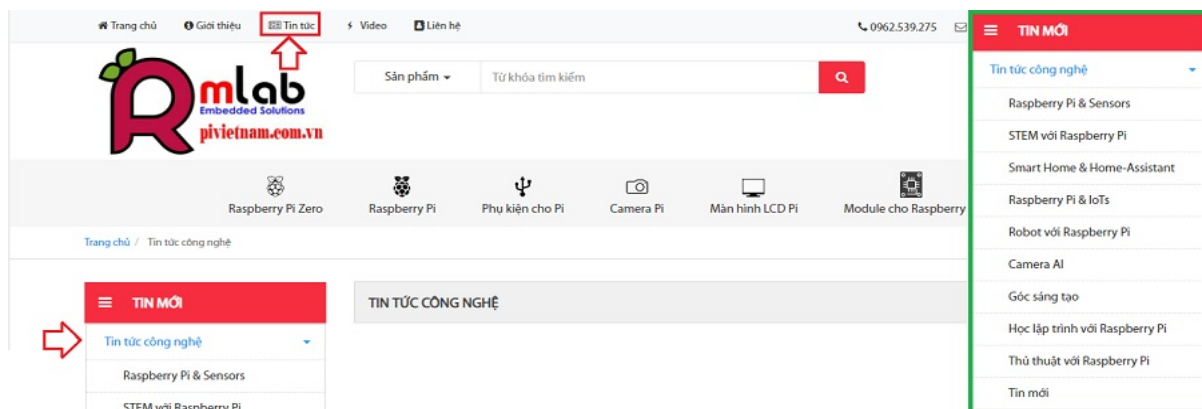
# Bài 1 : Lập trình cơ bản Raspberry Pi với GPIO

**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](http://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 1: Lập trình cơ bản Raspberry Pi với GPIO ([Link here](#))**



Khi chúng ta bắt tay làm quen với một dòng vi xử lý mới nào đó, chúng ta luôn cần chuẩn bị những thông tin và kiến thức cần thiết trước khi bắt đầu.

Đầu tiên là thông tin về phần cứng: dòng vi xử lý, tốc độ của nó, dung lượng của Ram, dung lượng bộ nhớ cứng (bộ nhớ để ghi chương trình) số lượng và vị trí của GPIO, hay các chuẩn giao tiếp được hỗ trợ.

Tiếp đó là kiến thức về lập trình cho vi xử lý. Ví như assembly – ngôn ngữ cấp thấp, ngôn ngữ C – hầu hết được hỗ trợ trên các công cụ lập trình, python – công cụ trên dòng vi xử lý chạy hệ điều hành, hay bất cứ ngôn ngữ nào mà công cụ cho vi xử lý đó hỗ trợ.

## 1. Kiến thức phần cứng

Trước hết hãy cùng xem GPIO mapping của Raspberry Pi. Đây là sơ đồ trên Raspberry 2 Model B


Trong 40 chân GPIO bao gồm :

- 26 chân GPIO. Khi thiết lập là input, GPIO có thể được sử dụng như chân interrupt, GPIO 14 & 15 được thiết lập sẵn là chân input.
- 1UART, 1 I2C, 2 SPI, 1 PWM (GPIO 4)
- 2 chân nguồn 5V, 2 chân nguồn 3.3V, 8 chân GND
- 2 chân ID EEPROM

Ví xử lý ARMv7 32bit quad core 900Mhz, dung lượng Ram 1G, và bộ nhớ kiểu micro SD dung lượng tùy chọn ( nên >=4G).

Khi một chân GPIO lên mức cao sẽ đạt điện áp 3.3V, dòng ra tối đa I<sub>max</sub> = 5mA.

Kiến thức cơ bản cho Pi như trên là đủ để có thể bắt đầu lập trình. Chúng ta cùng chuyển qua phần kiến thức tiếp theo.



Alternate Function					Alternate Function
	3.3V PWR	1		2	5V PWR
I2C1 SDA	GPIO 2	3		4	5V PWR
I2C1 SCL	GPIO 3	5		6	GND
	GPIO 4	7		8	UART TX
	GND	9		10	UART RX
	GPIO 17	11		12	GPIO 18
	GPIO 27	13		14	GND
	GPIO 22	15		16	GPIO 23
	3.3V PWR	17		18	GPIO 24
SPI0 MOSI	GPIO 10	19		20	GND
SPI0 MISO	GPIO 9	21		22	GPIO 25
SPI0 SCLK	GPIO 11	23		24	GPIO 8
	GND	25		26	GPIO 7
	Reserved	27		28	Reserved
	GPIO 5	29		30	GND
	GPIO 6	31		32	GPIO 12
	GPIO 13	33		34	GND
SPI1 MISO	GPIO 19	35		36	GPIO 16
	GPIO 26	37		38	GPIO 20
	GND	39		40	GPIO 21
					SPI1 SCLK

## 2. Kiến thức về ngôn ngữ lập trình

Lập trình trên Pi có nhiều sự lựa chọn. Có thể lập trình trực tiếp từ bash-shell của linux, hoặc lập trình với C đơn thuần, ngoài ra còn có python, perl hay Ruby (bạn có thể xem các code mẫu ở đây). Bạn nên chọn lựa một bộ thư viện thay vì chỉ lập trình với ngôn ngữ đơn thuần, vì đơn giản bạn đặt gạch xây nhà nhanh hơn là làm từng viên gạch cho ngôi nhà của mình. Thư viện sẽ giúp bạn bỏ qua lượng công việc vừa phức tạp và tốn công sức như gán địa chỉ của chân GPIO hay làm việc với thanh ghi ..v.v. Bạn có thể tập trung hơn vào xây dựng ứng dụng của mình.

Thư viện cho Pi cũng đa dạng không kém. Một thư viện tốt khi nó cung cấp nhiều hàm xử lý linh hoạt, hỗ trợ nhiều giao tiếp và tốc độ của thư viện nhanh ( tức là nó mất không quá nhiều lần gọi lệnh hay hàm trung gian để có thực hiện mong muốn của bạn ). Vấn đề tốc độ chỉ đáng quan tâm khi bạn thực sự làm việc với yêu cầu vi xử lý thực hiện lệnh nhanh chóng (giả như PWM). Bạn có thể xem qua [Benchmarking](#) cho các thư viện của Raspberry Pi.

Trong bài này, mình sẽ giới thiệu lập trình trên 2 ngôn ngữ được sử dụng rộng rãi trên Pi là C và Python. Hai bộ thư viện tương ứng là [WiringPi](#) và [RPiGPIO](#). WiringPi được viết dưới dạng framework của [wiring](#), nó cũng là framework mà Arduino sử dụng.

## 3. Bắt đầu với bài lập trình GPIO

Để bắt đầu với chuỗi bài lập trình cho Raspberry Pi. Chúng ta hãy cùng bắt đầu với những bài căn bản nhất mà có lẽ hơi buồn tẻ của lập trình – lập trình cho GPIO. Những bài dưới đây sẽ lập trình để điều khiển LED. Bạn hãy chuẩn bị phần cứng nhưng sợ đồ dưới đây.

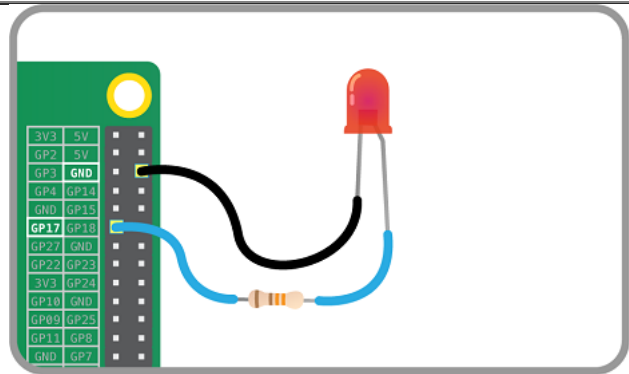
Nhiệm vụ của chúng ta sẽ điều khiển bật tắt chiếc đèn led này. Lưu ý rằng, khi GPIO của Pi được thiết lập lên mức cao thì hiệu điện thế U = 3.3V và dòng tối đa là I<sub>max</sub> = 50mA .

Giả sử bóng đèn Led thường dùng sáng ở 2V và I = 5mA thì điện trở phù hợp sẽ được tính là :

$$R = (3.3 - 2) / 0.005 = 260 \Omega$$

Bạn có thể chọn điện trở quen thuộc hơn như 220 Ω , 270 Ω hay 330 Ω.

Sau khi hoàn thành thiết kế phần cứng, chúng ta hãy bắt tay ngay vào lập trình.



### a) Lập trình với ngôn ngữ C

Các bạn cần cài đặt thư viện wiringpi trước khi lập trình. Có thể tải thư viện và xem hướng dẫn cài đặt tại [Wiringpi-project](#).

#### Bài 1: lập trình bật tắt LED

```
#include <wiringPi.h>
```

```
int main(void)
```

```
{
```

```
    wiringPiSetupGpio();
```

```
    pinMode(17, OUTPUT);
```

```
    while(1){
```

```
        digitalWrite(17, HIGH);
```

```
    }
```

```
    return 0;
```

1. Thêm thư viện Wiringpi : **#include<wiringPi.h>**

2. Thiết lập chọn kiểu đánh số chân GPIO

**wiringPiSetupGpio();**

Wiringpi có 4 kiểu chọn đánh số chân.

- wiringPiSetup() : thiết lập đánh số theo cách riêng của Pi
- wiringPiSetupGpio() : đánh số theo Broadcom GPIO – tương ứng với chân của hình 1.
- wiringPiSetupPhys() : đánh số theo chân header trên board.
- wiringPiSetupSys() : đánh số theo system class GPIO.

Để tiện sử dụng. Tất cả các chương trình chúng ta sẽ sử dụng cách đánh số thứ 2.

3. Chọn và thiết lập Output chân LED

**pinMode(pin, OUTPUT);**

**//pinMode(pin, INPUT);**

4. Bật-tắt LED (2 kiểu)

**digitalWrite(17, 1); //digitalWrite(17, HIGH);**

**digitalWrite(17, 0); //digitalWrite(17, LOW);**

5. Thực hiện build chương trình trên terminal

**gcc -Wall -o blink blink.c -lwiringPi**

**sudo ./blink**

Các bạn có thể sẽ ngạc nhiên khi kết thúc chương trình mà led vẫn sáng vì trạng thái thiết lập hiện tại của chương trình không bị thay đổi. Minh trình bày phần này kỹ hơn ở mục cuối của bài.

## Bài 2 : Nhấp nháy LED

```
#include <wiringPi.h>
```

```
int main(void)
```

```
{
```

```
    wiringPiSetupGpio();
```

```
    pinMode(17, OUTPUT);
```

```
    while(1)
```

```
    {
```

```
        digitalWrite(17, HIGH);
```

```
        delay(1000);
```

```
        digitalWrite(17, LOW);
```

```
        delay(1000);
```

```
    }
```

```
    return 0;
```

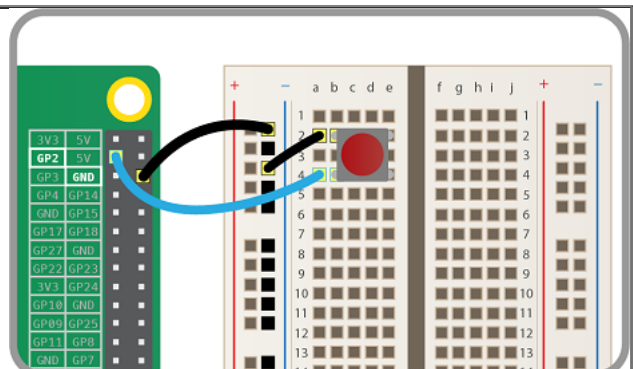
```
}
```

## Bài 3 : Điều khiển LED bằng button

Chương trình thực hiện yêu cầu đơn giản, khi bạn nhấn button thì đèn sẽ sáng. Chuẩn bị sơ đồ phần cứng như sau.

Khi button được nhấn thì trạng thái trên GPIO-2 sẽ xuống mức thấp.

Lưu ý rằng, khi set chân GPIO làm input thì trạng thái của GPIO sẽ lơ lửng (float) lúc cao, lúc thấp không xác định. Để xác định mức rõ ràng cần dùng điện trở để kéo GPIO lên mức cao hoặc kéo xuống thấp. Ở đây, sử dụng điện trở trong của GPIO để kéo lên mức cao ( $R=10k\Omega$ ).



```
#include <wiringPi.h>
```

```
int main(void)
{
    wiringPiSetupGpio();

    pinMode(17, OUTPUT);

    pinMode(2, INPUT);

    pullUpDnControl (2, PUD_UP);


    digitalWrite(17, 0);

    while(1)
    {
        if(!digitalRead(2)){

            digitalWrite(17, 1);

            // digitalWrite(17, !digitalRead(17));

            delay(300);

        }

    }

    return 0;
}
```

*Thiết lập input cho chân button*

```
pinMode(pin, INPUT);
```

*- Đọc tín hiệu từ button*

```
pullUpDnControl (pin, PUD_UP);
```

```
digitalRead(pin);
```

Kéo điện trở pin lên cao và đọc tín hiệu từ button.

- Để thực hiện ấn button để bật và tắt LED, các bạn hãy sửa dòng code này

```
//digitalWrite(17, 1);
```

```
digitalWrite(17, !digitalRead(17));
```

Nó sẽ đọc giá trị trên chân GPIO rồi thiết lập giá trị đảo ngược cho chân GPIO đó. (dù chúng ta không set chân đó là input).

## b) Lập trình với ngôn ngữ Python

Phần này chỉ dành cho những bạn có kiến thức cơ bản về Python, tuy nhiên vì Python được coi là ngôn ngữ cho người mới bắt đầu, rất dễ học nên các bạn có thể nhanh chóng xem qua Python cơ bản và bắt đầu lập trình với phần này.

Thư viện Rpi.GPIO được nhúng sẵn trên hệ điều hành Raspbian nên bạn có thể sử dụng luôn. Thư viện GPIO hỗ trợ thiết lập input/output cho GPIO và PWM software.

Lưu ý : Python tuân thủ nghiêm ngặt khoảng cách tương ứng giữa các câu lệnh. Những câu lệnh trong cùng một khối thì phải có khoảng cách bằng nhau. Nếu bạn copy đoạn code dưới đây thì có thể sẽ bị lỗi sai khoảng cách (space error), các bạn nên sửa lại cho đúng.

## Bài 1 : Bật tắt LED

```
#!/usr/bin/python3
```

```
import RPi.GPIO as GPIO
```

```
GPIO.setmode(GPIO.BCM)
```

```
GPIO.setup(17, GPIO.OUT)
```

```
GPIO.output(17, 1)
```

1. Từ giao diện màn hình chính chọn main > Programming > Python3 (IDLE).

2. Lựa chọn tạo new file và save file đó với tên "led.py"
3. Bước đầu tiên là import thư viện của GPIOzero

**import RPi.GPIO as GPIO**

4. Chọn và thiết lập output chân LED

**GPIO.setmode(GPIO.BCM)**

**GPIO.setup(pin, GPIO.OUT)**

Thư viện Rpi.GPIO hỗ trợ 2 kiểu đánh số GPPIO là **BCM** và **Board number** bạn cần chọn kiểu để thư viện có thể hiểu bạn đang dùng theo kiểu nào.

Thiết lập chân số 17 là output

5. Bật – tắt LED (có 3 kiểu thiết lập)

**GPIO.output(pin, 1)**

**# GPIO.output(pin, True)**

**# GPIO.output(pin, GPIO.HIGH)**

**GPIO.output(pin, 0)**

**# GPIO.output(pin, False)**

**# GPIO.output(pin, GPIO.LOW)**

6. Lưu file lại Ctrl+S, sau đó chạy code nhấn F5. Để kết thúc bạn nhấn Ctrl + C

Cũng giống như bài lập trình với C. Trạng thái của chương trình trên GPIO vẫn còn được lưu giữ.

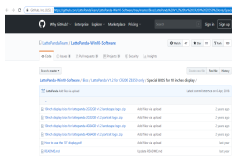
Ngoài ra bạn có thể chạy file led.py từ terminal

1. Trước khi chạy bạn cần khai báo thêm "shebang line" - vị trí của trình biên dịch . Ở đây là python3 (phiên bản 3).

**#!/usr/bin/python3**

Dòng code này được đặt ở dòng đầu tiên của file. Ở ví dụ trên bạn không cần khai báo thêm vì bạn đang sử dụng Python3 IDLE nên nó biết được luôn trình biên dịch của bạn.

2. Bạn mở terminal lên từ màn hình chính hoặc nhấn Ctrl + Shift + T
3. Gõ lệnh chuyển vị trí hiện tại của terminal tới thư mục led.py và thực hiện các lệnh để chạy chương trình



Lệnh thứ 2 để chuyển file đó thành file có thể chạy như file thực thi. Lệnh cuối để thực hiện chương trình. Để kết thúc bạn cũng ấn Ctrl + C

Tuy nhiên bạn sẽ chỉ thấy chương trình hiện lên rồi tắt đi ngay. Lí do là chương trình python chỉ đọc từ đầu tới cuối và kết thúc luôn. Muốn chương trình có thể tiếp tục bạn có thể đặt vòng lặp bên ngoài hoặc đặt thiết lập pause() ở cuối chương trình.



Hoặc

```
#!/usr/bin/python3
import RPi.GPIO as GPIO
from signal import pause

GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)

GPIO.output(17, 1)
pause()
```

## Bài 2 : Nhấp nháy LED

```
#!/usr/bin/python3
import RPi.GPIO as GPIO
from signal import pause

GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)

GPIO.output(17, 1)
pause()
```

### Bài 3: Điều khiển LED bằng button

```
#!/usr/bin/python3
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(2, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:
    if GPIO.input(2) == False:
        GPIO.output(17, 1)
        # GPIO.output(17, not GPIO.input(2))
        sleep(0.3)
```

## 4. Giải phóng GPIO

Như đã biết ở trên, để kết thúc chương trình các bạn có thể nhấn **Ctrl+C**. Đó là keyboard interrupt, chương trình sẽ bị break và dừng chương trình. Nhưng nếu bạn chú ý, trạng thái của chương trình vẫn còn được giữ nguyên sau khi break, ví dụ như trước khi bị break chân GPIO được set lên mức cao thì nó vẫn còn giữ nguyên mức cao đó sau khi bị break. Sẽ chẳng ảnh hưởng gì nếu bạn thực hiện chạy tiếp một chương trình mà không lo tới GPIO đó, nhưng sẽ có vấn đề nếu Pi vẫn đang kết nối với phần cứng bên ngoài, nghĩa là nó vẫn có thể tác động lên phần cứng đó mà bạn không mong muốn (ví dụ đèn sẽ vẫn sáng trong chương trình điều khiển led).

Nếu chạy lại file trên một lần nữa sẽ có thông báo warning hiện ra :

"RuntimeWarning: This channel is already in use, continuing anyway. Use GPIO.setwarnings(False) to disable warnings."

Thông báo này có nghĩa là chân GPIO đó đang được sử dụng. Các bạn có thể bỏ qua vì chương trình vẫn sẽ tiếp tục chạy, hoặc ấn nó đi bằng cách dùng một dòng lệnh `GPIO.setwarnings(False)`. Nhưng warning này khiến cho chương trình của bạn trở nên "out of control". Đó là điều không thể chấp nhận với lập trình viên.

Chúng ta nên khắc phục bằng cách giải phóng toàn bộ GPIO (hoặc một phần được sử dụng) ngay sau khi phát hiện interrupt. Nhưng trước hết hãy cùng nâng cấp chương trình lên mức mới để có thể lập trình chuyên nghiệp hơn.

Chương trình C bắt sự kiện **"Ctrl+C"**

```
#include<stdio.h>
#include<signal.h>
#include<unistd.h>

void sig_handler(int signo)
{
    if (signo == SIGINT){
        printf("received \"Ctrl+C\" \n");
        // cleanup port o day

        // example :
        // digitalWrite(17, 0);
        // pinMode(17, INPUT);
    }
}

int main(void)
{
    if (signal(SIGINT, sig_handler) == SIG_ERR)
        printf("\n can't catch \"Ctrl+C\" \n");

    // Chuong trinh chinh duoi day
}
```

Thư viện C chưa hỗ trợ hàm chuyên dụng để bạn có thể sử dụng, nên bạn phải xử lý bằng tay từng GPIO hay PORT. Hãy chú ý phần example trong hàm bắt interrupt trên. Hàm này sẽ chạy độc lập với hàm main. Còn những sự kiện interrupt khác các bạn có thể tham khảo thêm ở một bài viết khá hữu ích [ở đây](#)

Chương trình Python bắt sự kiện **"CTR+C"**

```
import RPi.GPIO as GPIO

try:
    pass
    # ham code chinh o day
except KeyboardInterrupt:
    pass
    # code cho keyboard interrupt
except:
    pass
    # code cho interrupt khac
finally:
    GPIO.cleanup() # clean up all port
```

Chương trình chính bây giờ sẽ được đặt dưới hàm "try". Khi bạn nhấn **Ctrl+C** tức là keyboard interrupt, chương trình sẽ ngay lập tức nhảy sang vùng code cho keyboard interrupt, nếu là interrupt loại khác thì chương trình sẽ nhảy sang vùng code cho interrupt loại khác, và sau khi thực hiện chương trình cho interrupt chương trình sẽ nhảy tới vùng code kết thúc chương trình. Hàm **GPIO.cleanup()** có thể thay bạn xử lý toàn bộ các port.

Trong vùng code kết thúc chương trình nhất định phải có câu lệnh **GPIO.cleanup()**. Nó sẽ giúp giải phóng tất cả các GPIO đang được sử dụng, như vậy trạng thái cũ của chương trình sẽ được xóa bỏ hoàn toàn.

hung nguyen | ★★★★★  
09/17/2021

anh có bài nào hướng dẫn lập trình stream video sử dụng raspberry pi không ạ, em cảm ơn

huong ho | ★★★★★  
09/17/2021

rat hay aaaaaaaaaaaaaaaaaaaaa

david | ★★★★★  
09/17/2021

Tôi rất thích nội dung trang này

knx | ★★★★★  
09/17/2021

Tôi rất thích nội dung trang này

Ngô Văn Trường | ★★★★★  
04/19/2021

Bài viết hay cho người mới bắt đầu.

12345678910>>

Hiển thị từ 1 đến 5 trong tổng số 47 (10 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:

6ed2e4

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.group@gmail.com

Email hỗ trợ kỹ thuật : mlab.services.tech@gmail.com

https://mlab.vn/index.php?\_route\_=13456-bai-1-lap-trinh-co-ban-raspberry-pi-voi-gpio.html

7/8

