Bài 15: Project thời gian thực cho Raspberry Pi – module DS3231

16:03 - 15/01/2019

thời gian thực cho Raspberry Pi với module DS3231

» Hướng cài đặt Hệ điều hành và Remote Desktop cho Raspberry Pi nhanh chóng và cực kỳ đơn giản (https://pivietnam.com.vn/huong-cai-dat-he-dieu-hanh-va-remote-desktop-cho-raspberry-pi-nhanh-chong-va-cuc-ky-don-gian-pivietnam-com-vn.html)

» Remote Desktop Raspberry Pi không cần Wifi, mạng LAN và IP (https://pivietnam.com.vn/remote-desktop-raspberry-pi-without-wifi-lan-and-ip-pivietnam-com-vn.html)

» Camera nhiệt giải pháp tuyệt vời cho mùa Covid-19 (https://pivietnam.com.vn/camera-nhiet-giai-phap-tuyet-voi-cho-mua-covid-19-pivietnam-com-vn.html)

» Lập trình cơ bản với OpenPLC trên Raspberry Pi (https://pivietnam.com.vn/lap-trinh-co-ban-voi-openplc-tren-raspberry-pi-pivietnam-com-vn.html)

» Hướng dẫn cài đặt OpenPLC trên Raspberry Pi (https://pivietnam.com.vn/huong-dan-cai-dat-openplc-tren-raspberry-pi-pivietnam-com-vn.html)

Nội dung trong bài viết bao gồm :

1.Thời gian thực cho Raspberry Pi với module DS3231.

Giải thích lý do tại sao cần module thời gian thực. Các bước để hệ thống tự cập nhật thời gian từ module.

2.Bộ thư viện cho module DS3231

Giới thiệu bộ thư viện cho DSB3231. Trình bày hai ví dụ căn bản về chức năng của module, liệt kê danh sách hàm chức năng.

1. Thời gian thực cho Raspberry Pi với module DS3231

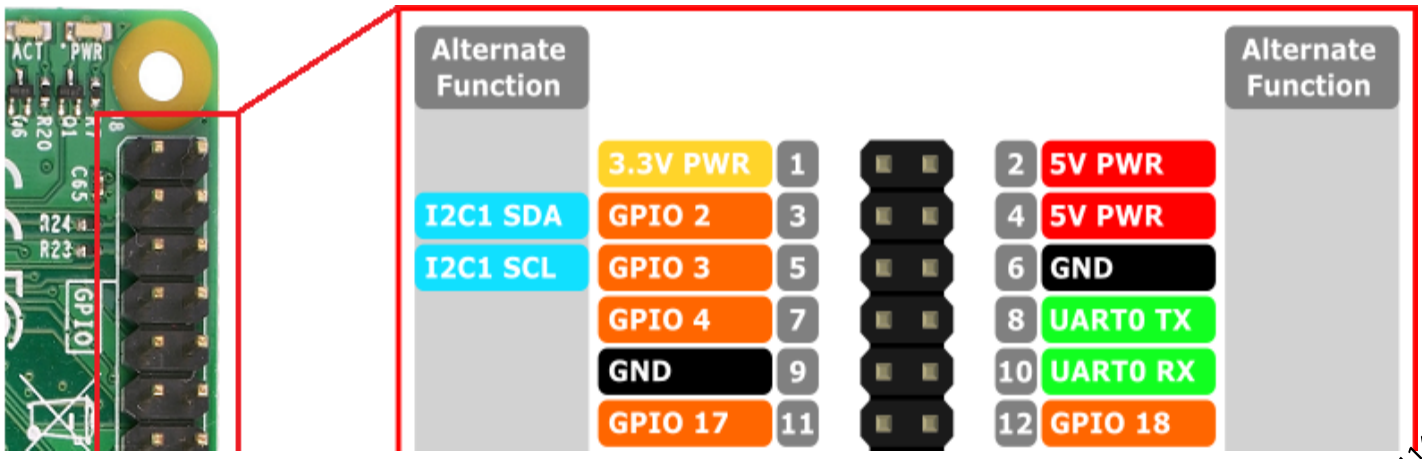
Trong tất cả các phiên bản của Pi đều không có bộ thời gian thực đi kèm. Có nghĩa là hệ thống sẽ mất ngày giờ sau khi tắt máy. Thông thường nếu Pi được kết nối internet, ngày giờ sẽ được cập nhật tự động khi Pi khởi động bằng phương thức NTP. Điểm này gây ra hạn chế nếu muốn biến Pi thành master/server thực thụ. Khi đó hệ thống luôn cần phải chạy đúng thời gian thực ngay sau khi đã mất mạng hoặc mất điện.

Module thời gian thực DS3231 có chi phí rất tiết kiệm. Khi so với 2 phiên bản DS1302 và DS1307 thì có độ chính xác cao hơn và nhiều hàm chức năng hơn.

Các bước tiến hành cài đặt module với Pi như sau :

1.1) Nối Raspberry với DS3231

Kết nối 2 thiết bị qua cổng I2C. Nếu bạn chưa biết thiết lập I2C trên Pi thế nào thì có thể xem lại bài hướng dẫn dùng I2C với Pi. VCC - VCC, GND - GND< SCL - SCL, SDA - SDA.



Hình 1 : Chân I2C trên Raspberry Pi

1.2) Mở file /boot/config.txt bằng lệnh

```
sudo nano /boot/config.txt
```

Thêm *dtoverlay=i2c-rtc,ds3231* vào cuối file. **Reboot** lại Pi. Driver cho module thời gian thực đã được tích hợp sẵn trong Raspian nên bây giờ các bạn có thể sử dụng luôn module.

Lưu ý đây là phiên bản Raspian Jessy.

1.3) Thiết lập thời gian cho module thời gian thực

```
sudo hwclock -w
sudo hwclock -r
```

Lệnh đầu tiên để cài đặt thời gian cho module. Thời gian được lấy từ thời gian của hệ thống. Lệnh thứ hai để đọc thời gian của module.

Nếu câu lệnh trên bị lỗi là do hệ thống không nhận ra module. Kiểm tra lại kết nối v à reboot lại.

1.4) Tiếp tục mở file /etc/init.d/hwclock.sh và sửa dòng "HWCLOCKACCESS=yes" thành "HWCLOCKACCESS=no". Sau đó reboot là hoàn tất việc cài đặt.

Bạn có thể kiểm tra lại bằng việc tắt nguồn cho Pi, ngắt kết nối mạng, đợi sau khoảng thời gian và bật Pi lên kiểm tra giờ. Chú ý đừng quên gắn Pin cho module trước khi kiểm tra.

---

2. Bộ thư viện cho module DS3231

Nếu mục đích của bạn với DS3231 không chỉ dừng lại ở mức cập nhật thời gian cho Pi chẳng hạn như bạn muốn dùng nó để quản lý công việc theo thời gian hay dùng làm báo thức (DS3231 có 2 alarm sử dụng cho việc này) thì thư viện cho nó là điều cần thiết.

Thư viện thời gian thực được mình (một phần từ jeelab (https://github.com/jcw/rtclib)) viết trên ngôn ngữ C. Nó được sửa lại từ thư viện DS3231 (https://github.com/DuongNguyenHai/Arduino/tree/master/Realtime-DSB3231) dành cho Arduino. Thư viện hỗ trợ là I2C cũng theo framework arduino được mình giới thiệu trong bài viết trước.

Bộ thời gian thực cung cấp 3 chức năng chính là thời gian ngày tháng theo chuẩn như lịch treo tường thông thường tới hết năm 2100, hỗ trợ cả năm nhuận, thời gian có 2 kiểu là 12h và 24h; cung cấp 2 bộ báo thức có thể dùng làm interrupt báo thức cho pi; tạo xung clock 1hz/1khz/4khz/8khz.

Bộ thư viện được public trên github (https://github.com/DuongNguyenHai/Raspberry-Pi/tree/master/libraries/Realtime-DSB3231) của mình. Trước tiên hãy cùng xem qua 2 ví dụ đơn giản để xem nó hoạt động thế nào. Bạn nên chạy chương trình ví dụ trong thư mục example để thực hành luôn, nhìn chương trình chạy sẽ thấy nó dễ dàng hơn rất nhiều.

Ví dụ 1 : time. cc

Đọc giá trị thời gian – ngày tháng và in ra màn hình.

```
// g++ -Wall -std=c++11 time.cc ../src/RTClib.cpp ../src/Wire/Wire.cpp -o time

#include <stdio.h>
#include "../src/RTClib.h"

RTC_DS3231 rtc;

int main(int argc, char const *argv[])
{
  char tm[22];

  if (! rtc.begin()) {
    printf("Couldn't find RTC\n");
    while (1);
  }

  if (rtc.lostPower()) {     // check RTC time has set or not.
    printf("RTC lost power, lets set the time!");
    // following line sets the RTC to the date & time.
    //  adjust(DateTime(year,month,day,hour,minute,second))
    rtc.adjust(DateTime(2017, 01, 1, 00, 00, 00));
  }


  while(1) {
    rtc.getDateTimeString(tm);
    printf("Date time: %s\n", tm);
    usleep(1000000);
  }

  return 0;
}
```

Đầu tiên khai báo một đối tượng thời gian thực là rtc. Thiết lập cho nó khởi động bằng hàm begin(), kiểm tra xem đã có thiết lập thời gian cho nó chưa, dùng hàm lostPower(). Nếu chưa được thiết lập thời gian thì sẽ thiết lập với hàm adjust(DateTime(2017, 01, 1, 00, 00, 00)). Hàm này thiết lập thời gian là năm 2017 ngày 1 tháng 1 lúc 0h0’0s. Để lấy kết quả thời gian sử dụng hàm getDateTimeString() và in nó ra màn hình . Cứ sau mỗi giây sẽ lấy một lần.

Để compile chương trình dùng lệnh :

```
g++ -Wall -std=c++11 time.cc ../src/RTClib.cpp ../src/Wire/Wire.cpp -o time
```

Kết quả

Ví dụ 2 : alarm.cc

Thiết lập báo thức và kiểm tra báo thức.

```
// g++ -Wall -std=c++11 alarm.cc ../src/RTClib.cpp ../src/Wire/Wire.cpp -o alarm

#include <stdio.h>
#include "../src/RTClib.h"

RTC_DS3231 rtc;

int main(int argc, char const *argv[])
{
  char tm[22];

  if (! rtc.begin()) {
    printf("Couldn't find RTC\n");
    while (1);
  }

  DateTime dt(2017, 01, 18, 05, 59, 55);

  printf("set: %d/%d/%dT%d:%s%d:%s%d\n", dt.year(), dt.month(), dt.day(), dt.hour(),
         ((dt.minute() > 9) ? "": "0"), dt.minute(), ((dt.second() > 9) ? "": "0"), dt.second());

  rtc.adjust(dt);

  alarm_t alarmTime;

  alarmTime.hh = 5;    // hour
  alarmTime.mm = 59;   // minute
  alarmTime.ss = 58;   // second

  rtc.setAlarm(ALARM_1, alarmTime, MATCH_HHMMSS_OR_HHMM);
  printf("set alarm 1 at %d:%d:%d\n", alarmTime.hh, alarmTime.mm, alarmTime.ss);
  rtc.setAlarmHour(ALARM_2, 6); // match hour
  printf("set alarm 2 at 6h\n");

  while(1) {
    rtc.getDateTimeString(tm);
    printf("Date time: %s\n", tm);
    if(rtc.isAlarmRinging(ALARM_1)) {
      printf("alarm 1 is on !\n");
      rtc.clearAlarm(ALARM_1);
    }
    if(rtc.isAlarmRinging(ALARM_2)) {
      printf("alarm 2 is on !\n");
```

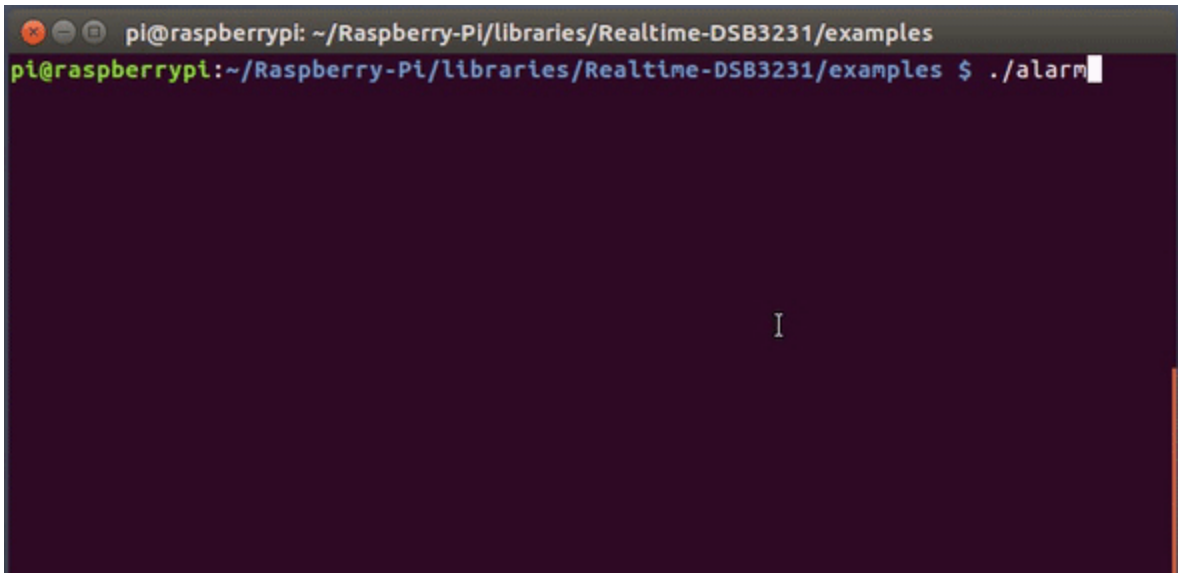
```
        rtc.clearAlarm(ALARM_2);
      }
      usleep(1000000);
    }

    return 0;
}
```

Cài đặt thời gian cho module là 5h59’55s. Thiết lập alarm 1 bật vào lúc 5h59’58s và alarm 2 bật vào lúc 6h. Kiểm tra alarm được bật bằng hàm isAlarmRinging(). Vì alarm không tự ngắt nên cần ngắt thủ công cho nó; dùng hàm clearAlarm().

Để compile chương trình dùng lệnh :

```
g++ -Wall -std=c++11 alarm.cc ../src/RTClib.cpp ../src/Wire/Wire.cpp -o alarm
```

Kết quả :



Dưới đây là danh sách những hàm thông dụng trong bộ thư viện bao hàm hầu hết các chức năng của module. Ngoài ra còn vài hàm nhỏ khác có thể xem trong file nguồn thư viện.

1. Hàm khởi tạo

```
bool begin();
```

Hàm này thực hiện việc khởi tạo I2C.

2. Hàm cài đặt thời gian - ngày tháng

```
static void adjust(const DateTime& dt);
```

Tham số đầu vào là đối tượng của class DateTime. Khởi tạo giá trị cho đối tượng này có 4 cách :

```
DateTime (uint32_t t =0);
DateTime (uint16_t year, uint8_t month, uint8_t day, uint8_t hour =0, uint8_t min =0, uint8_t sec =0);
DateTime (const DateTime& copy);
DateTime (const char* date, const char* time);
```

- Cách 1: sử dụng unix poch time để cài đặt.
- Cách 2: truyền thẳng các giá trị ngày tháng vào đối tượng
- Cách 3: Copy từ đối tượng khác
- Cách 4: Cắt giá trị từ chuỗi. Ví dụ date = "Dec 26 2009", time = "12:34:56"

3. Hàm kiểm tra xem thời gian được thiết lập chưa

```
bool lostPower(void);
```

Thời gian có thể mất đi nếu bị mất nếu nguồn VCC bị ngắt và không có pin dự phòng. Nên kiểm tra lại giờ của module trước khi làm việc với nó.

4. Hàm đọc thời gian – ngày tháng

```
bool getDateTimeString(char *s);
bool getDateString(char *s);
bool getTimeString(char *s);
```

Hàm đầu tiên sẽ gán chuỗi thời gian – ngày tháng cho chuỗi s. Chuỗi có đinh dạng theo chuẩn ISO 8601 : “yyyy:mm:ddThh:mm:ss”. Tham số ‘s’ là một mảng chứa 22 ký tự.

Hàm thứ 2 gán chuỗi ngày tháng. Định dạng : “yyyy:mm:dd”. Tham số ‘s’ chứa 10 ký tự.

Hàm thứ 3 gán chuỗi thời gian. Định dạng “hh:mm:ss” . Tham số ‘s’ chứa 10 ký tự.

Cả 3 hàm sẽ trả về false nếu không đọc được thời gian. Ngoài ra còn có thêm hàm

```
static DateTime now();
```

Hàm này trả về đội tượng của class DateTime. Ví dụ ta khai báo DateTime dt = rtc.now(). Sau đó có thể dùng đối tượng dt để truy xuất tới từng giá trị thời giant heo định dạng số nguyên. Ví dụ giờ sẽ là dt.hour(), phút là dt.minute(). Bạn hãy xem thư viện để có thể thấy rõ hơn.

5. Hàm bật/tắt chế độ Alarm

```
bool switchClock(bool state);
```

Module DS3231 chỉ cho phép chạy một trong hai chế độ là tạo xung qua chân INT/SQW hoặc sử dụng nó làm chân báo thức (interupt). Nếu thiết lập swtichClock(true) thì sẽ sử dụng là alarm và ngược lại.

6. Hàm thiết lập thời gian cho Alarm

```
bool setAlarm(uint8_t alarm);
bool setAlarmRepeat(uint8_t alarm);
bool setAlarm(uint8_t alarm, alarm_t tm, uint8_t mode = MATCH_HHMMSS_OR_HHMM);
```

Có 2 alarm có thể thiết lập thời gian. Khi thời gian thực trùng với thời gian được hẹn giờ trong alarm thì 1 bit trong thanh ghi trạng thái của DS3231 tương ứng với alarm đó sẽ được thiết lập là 1, và đồng thời chân INT/SQW sẽ được đưa xuống mức 0.

Alarm 1 có thể hẹn giờ chính xác tới giây, phút, giờ, ngày trong tháng; Alarm 2 hẹn giờ được phút, giờ, ngày trong tháng.

Hàm đầu tiên thiết lập alarm 1 hoặc alarm 2 chạy. Cả 2 có thể chạy đồng thời. Chỉ cần dùng hàm này mà không cần dùng hàm switchClock(bool state) vì bên trong hàm này đã gọi sẵn đó.

Hàm thứ 2 thiết lập kiểu chạy repeat time. Đối với alarm 1 thì cứ sau mỗi giây nó sẽ báo alarm 1 lần; với alarm 2 thì cứ sau mỗi phút sẽ báo. Tham số là chọn alarm : ALARM_1 hoặc ALARM_2.

Hàm thứ 3 thiết lập thời gian cho alarm. Tham số đầu tiên là chọn alarm, tham số thứ 2 là thời gian thiết lập, và cuối cùng là chế độ thiết lập.

```
typedef struct Alarm {
  uint8_t d, hh, mm, ss;
} alarm_t;
```

alarm_t có 4 giá trị : d là ngày trong tháng, hh là giờ, mm là phút, ss là giây.

Có 5 chế độ cả thể cho 2 alarm :

Mode	Alarm 1	Alarm 2
0	repeat time	repeat time
1	ss	
2	mm, ss	mm
3	hh, mm, ss	hh, mm
4	date, hh, mm , ss	date, hh, mm

Chế độ 0 tương ứng với hàm setAlarmRepeat(). Chế độ 1 chỉ chạy vơi alarm 1, sẽ báo thức khi giây trong thời gian thực trùng với giây được cài đặt. Ví dụ khi cài đặt giây thứ 30 làm báo thức thì cứ sau mỗi phút tại giây thứ 30 alarm 1 sẽ báo. Chế độ 2 có thể hẹn giờ với phút và giây đối với alarm 1, hẹn giờ phút với alarm 2. Tương tự chế độ càng lên cao sẽ hẹn giờ được càng nhiều giá trị thời gian hơn.

Các macro define dưới đây sẽ tương ứng với các chế độ bên trên, mục đích của nó là để người sử dụng tiện theo dõi. MATCH__MMSS_OR__MM nghĩa là báo thức khi trùng mm, ss (phút, giây) với alarm1 và trùng mm với alarm 2. Macro này tương ứng với chế độ 2.

```
#define EACH_SS_OR_MM 0
#define MATCH_SS 1
#define MATCH_MMSS_OR_MM 2
#define MATCH_HHMMSS_OR_HHMM 3
#define MATCH_DDHHSS_OR_DDHHMM 4
```

7. Hàm kiểm tra báo thức

```
bool isAlarmRinging(uint8_t alarm);
```

Tham số đầu vào lả ALARM_1 hoặc ALARM_2. Trả về giá trị true nếu alarm được bật, false nếu không.

Lưu ý là khi alarm được bật, nó sẽ không tự tắt nên sẽ phải thiết lập tắt cho nó sau mỗi lần hàm isAlarmRinging() trả về giá trị true.

```
bool clearAlarm(uint8_t alarm=0);
```

alarm là tham số chọn alarm 1 hoặc 2. Nếu alarm = 0 nghĩa là tự động xóa cả 2 alarm đồng thời.

8. Hàm thiết lập xung clock

```
static void writeSqwPinMode(Ds3231SqwPinMode mode);
```

Ds3231SqwPinMode có 5 chế độ :

- DS3231_OFF : tắt tạo xung
- DS3231_SquareWave1Hz : tạo xung 1Hz
- DS3231_SquareWave1kHz : tạo xung 1kHz
- DS3231_SquareWave4kHz : tạo xung 4kHz
- DS3231_SquareWave8kHz : tạo xung 8kHz

Có thể sử dụng hàm

```
static Ds3231SqwPinMode readSqwPinMode();
```

Để đọc xem xung clock đang ở chế độ nào.

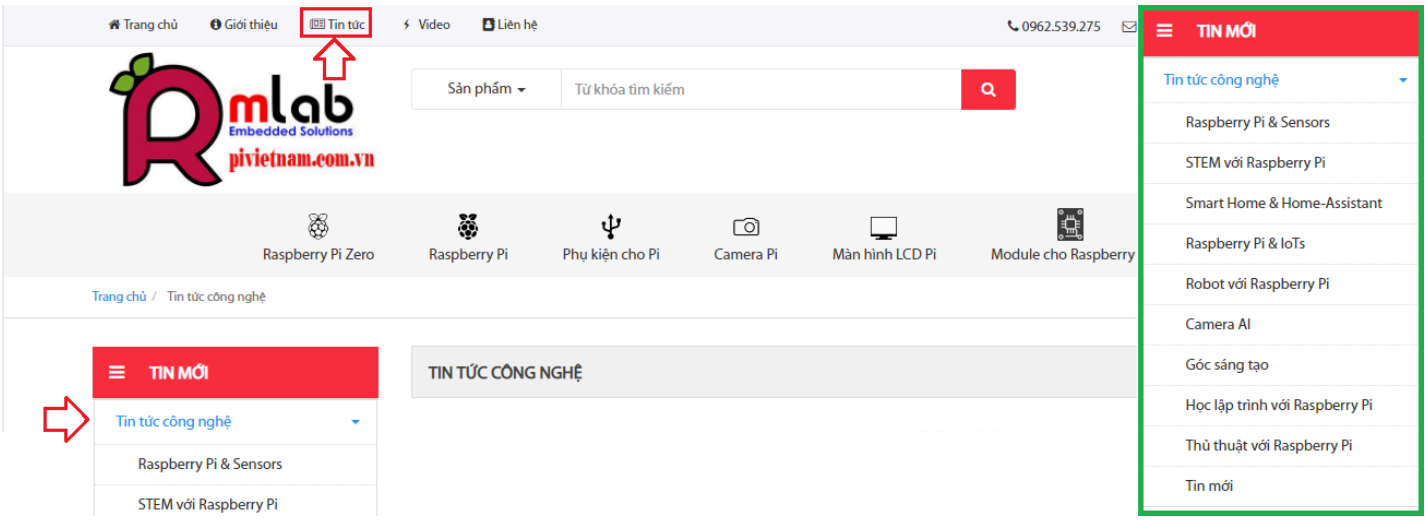
9. Đọc chân ngắt alarm

Khi alarm được bật, chân INT/SQW sẽ được đưa xuống mức thấp. Có thể dùng chân này làm chân báo tín hiệu ngắt (interupt) cho Pi. Mình không hỗ trợ hàm này vì có thể sử dụng hàm isAlarmRinging() để kiểm tra báo thức, tuy nhiên nếu các bạn muốn kiểu interupt thực thụ thì các bạn có thể tự viết thêm vào.

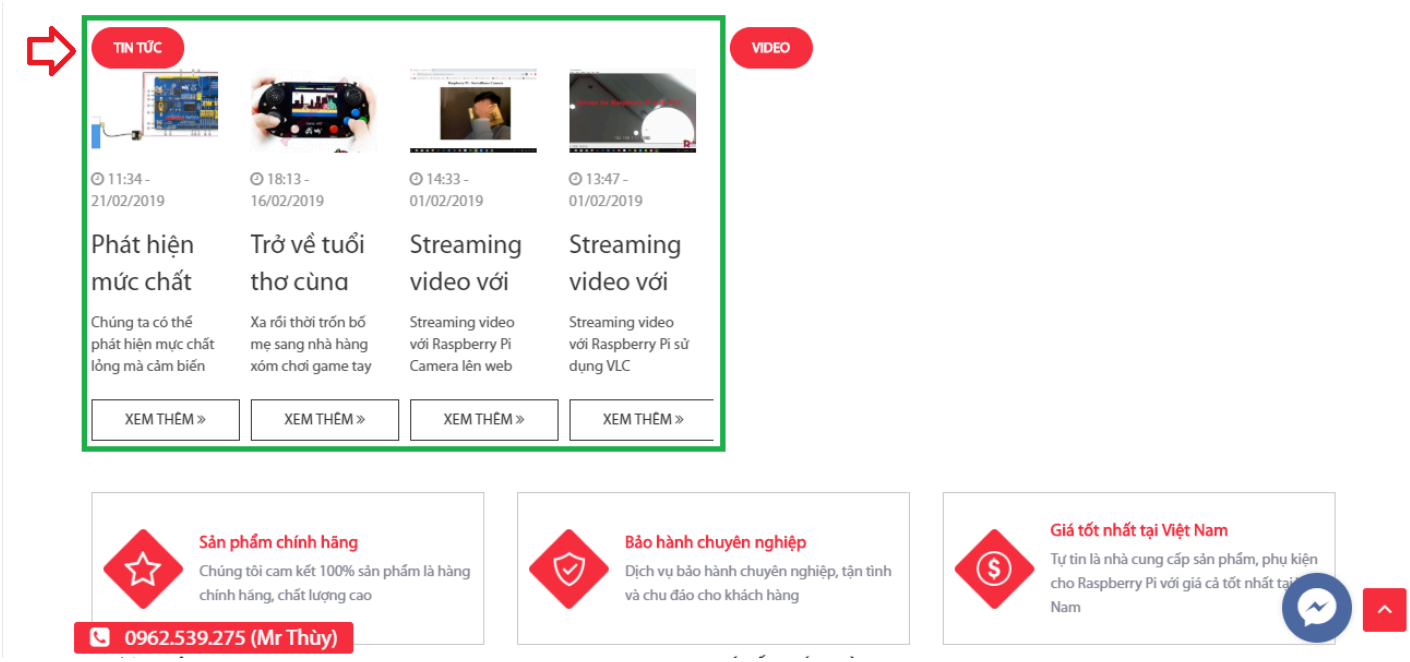
Mọi câu hỏi và đóng góp có thể bình luận trên trang web hoặc trên github (https://github.com/DuongNguyenHai/Raspberry-Pi)của mình.

Để cập nhật các tin tức công nghệ mới các bạn làm theo hướng dẫn sau đây :

Các bạn vào Trang chủ >> Tin tức. ở mục này có các bài viết kỹ thuật thuộc các lĩnh vực khác nhau các bạn có thể lựa chọn lĩnh vực mà mình quan tâm để đọc nhé !!!



Các bạn cũng có thể kéo xuống cuối trang để xem những tin tức công nghệ mới nhất.



VIDEO

Message us (https://m.me/599911456

VỀ CHÚNG TÔI

Website uy tín cung cấp Raspberry Pi chính hãng , và các phụ kiện , board mạch mở rộng cho Raspberry Pi tại Việt Nam.

Số 30F9 - Ngõ 104 Lê Thanh Nghị - Hai Bà Trưng - Hà Nội

02436.231.170

smarttechvn.group@gmail.com

HOTLINE TƯ VẤN TRỰC TIẾP

086.262.8846 (Mr Thùy) (tel:0962539275)

(Thời gian làm việc 8h - 17h30, thứ 2 tới thứ 7. Hỗ trợ Online ngoài giờ hành chính và chủ nhật.)

VỀ CHÚNG TÔI

Giới thiệu (https://pivietnam.com.vn/ve-chung-toi)

Lịch sử hình thành (https://pivietnam.com.vn/lich-su-hinh-thanh)

Đội ngũ lạnh đạo (https://pivietnam.com.vn/doi-ngu-lanh-dao)

Tuyển dụng (https://pivietnam.com.vn/tuyen-dung-quy-i)

Liên hệ (https://pivietnam.com.vn/lien-he)

ĐÃ THÔNG BÁO BỘ CÔNG THƯƠNG (http://online.gov.vn/Home/WebDetails/101224)

CHÍNH SÁCH

Hướng dẫn mua hàng online (https://pivietnam.com.vn/huong-dn-mua-hang-online-mlab-vn)

Chính sách vận chuyển và giao nhận (https://pivietnam.com.vn/chinh-sach-van-chuyen-va-giao-nhan-mlab-vn)

Chính sách kiểm hàng (https://pivietnam.com.vn/chinh-sach-kiem-hang)

Thông tin chuyển khoản (https://pivietnam.com.vn/thong-tin-chuyen-khoan-mlab-vn)

Hỗ trợ sau bán hàng (https://pivietnam.com.vn/ho-tro-sau-ban-hang-mlab-vn)

Chính sách bảo hành (https://pivietnam.com.vn/chinh-sach-bao-hanh-mlab-vn)

Chính sách đổi trả, hoàn tiền (https://pivietnam.com.vn/chinh-sach-doi-tra-hoan-tien-mlab-vn)

Chính sách bảo mật thông tin (https://pivietnam.com.vn/chinh-sach-bao-mat-thong-tin-mlab-vn)

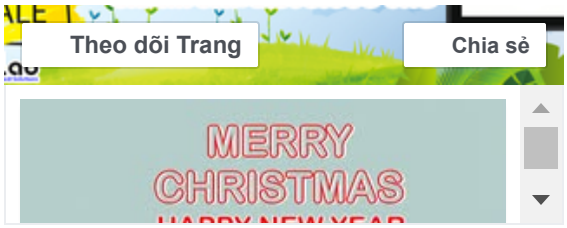
ĐĂNG KÝ NHẬN BẢN TIN

Nhập email đăng ký Đăng ký

FACEBOOK FANPAGE

Message us (https://m.me/599911456

MLAB
5.572 người theo dõi



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 : smarttechvn.group@gmail.com

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

Website : https://pivietnam.com.vn/

Số điện thoại : 02436.231.170 or 086.262.8846

(tel:0962539275)

Message us (https://m.me/599911456