Lập Trình Nhúng và IoT

Nguyễn Thành Công

Ngày 7 tháng 2 năm 2021

Chương 1

Máy trạng thái

Máy trạng thái là một trong những những kĩ thuật lập trình mình gặp nhiều nhất khi lập trình nhúng, dường như nó xuất hiện ở khắp nơi. Nắm vững kĩ thuật này thì code các bạn viết sẽ trở nên mạch lạc, sáng sủa hơn, và việc đọc code của người khác trở nên đơn giản hơn.

Blocking vs. Non-blocking

Blocking là gì? Hãy xem qua chương trình chớp tắt led trong Arduino như sau:

```
1 void loop() {
2   digitalWrite(LED_BUILTIN, HIGH);
3   delay(1000);
4   digitalWrite(LED_BUILTIN, LOW);
5   delay(1000);
6 }
```

Có thể thấy là nó bật led, sử dụng hàm delay() trong 1 giây rồi tắt led, tiếp tục delay() và chương trình lặp lại liên tục.

Vấn đề là ở chỗ hàm delay(), chương trình đứng yên một chỗ và không làm gì cả, hay nói cách khác là nó bị **block** tại chỗ đó. Viết chương trình có chứa hàm tạm dừng một chỗ như delay() thì người ta gọi là **Blocking mode**. Trong thế giới nhúng, tài nguyên hạn chế nên chương trình đứng một chỗ như vậy là việc hết sức lãng phí. Thế nên có kiểu viết khác gọi là **Non-Blocking mode** nhằm làm cho chương trình chạy liên tục mà không bị đứng tại một điểm nào cả.

Ví dụ về viết kiểu Non-blocking cho chương trình chớp tắt led. Ý tưởng đằng sau này là lưu lại thời gian của lần bật/tắt led lần trước, rồi so sánh với thời điểm hiện tại, nếu nó vượt quá 1 giây chẳng hạn, thì tiến hành lần bật tắt tiếp theo.

```
1 void loop() {
2     static uint32_t tick=0;
3     static uint8_t led_state=LOW;
```

```
4       if(millis()-tick>1000)
5       {
6           tick=millis();
7           if(LOW==led_state){
8               led_state=HIGH;
9               digitalWrite(LED_BUILTIN, led_state);
10          }
11          else{
12              led_state=LOW;
13              digitalWrite(LED_BUILTIN, led_state);
14          }
15      }
16
17      // Other code here
18  }
```

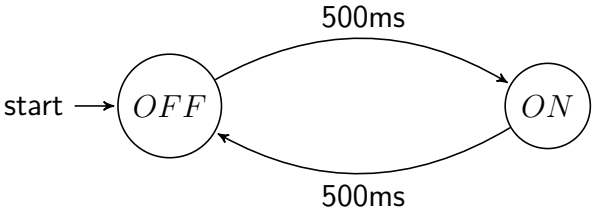
Như đoạn code trên, biến **tick** lưu lại thời gian của hoạt động bật/tắt led lần trước, lưu ý là nó phải được khai báo *static* để không khởi tạo lại khi vòng loop bắt đầu vòng lặp mới.

Với cách viết như trên thì chương trình không bị đứng ở bất cứ điểm nào. Trong hàm loop(), chương trình liên tục hỏi đã đến lúc bật/tắt led hay chưa, nếu đến lúc thì thực hiện bật/tắt và lưu lại thời điểm vừa thực hiện. Nếu chưa thì thoát xuống làm chuyện khác. Đến đây bạn có thể viết chương trình điều khiển 2 led như sau:

Rồi đến đây thì bạn vừa có thể tiết kiệm được tài nguyên CPU, bằng cách làm nhiều tác vụ tuần tự nhau, vừa có thể thêm một số tính năng mới mà không cần phải đập đi xây lại, sugoi!!! Nhớ là mỗi tính năng bạn thêm vào đều phải đảm bảo tính Non-blocking he.

Máy trạng thái

Máy trạng thái (state machine) là thứ rất thường gặp trong lập trình nhúng. Ví dụ bạn viết chương trình điều khiển một cái động cơ, thì nó sẽ có những trạng thái như là đang dừng, đang quay ngược chiều hoặc cùng chiều kim đồng hồ... Máy trạng thái có thể hiểu là sơ đồ tất cả các trạng thái của một đối tượng nào đó và những điều kiện để chuyển từ trạng thái này sang trạng thái khác. Dưới đây là một ví dụ về chớp tắt led.

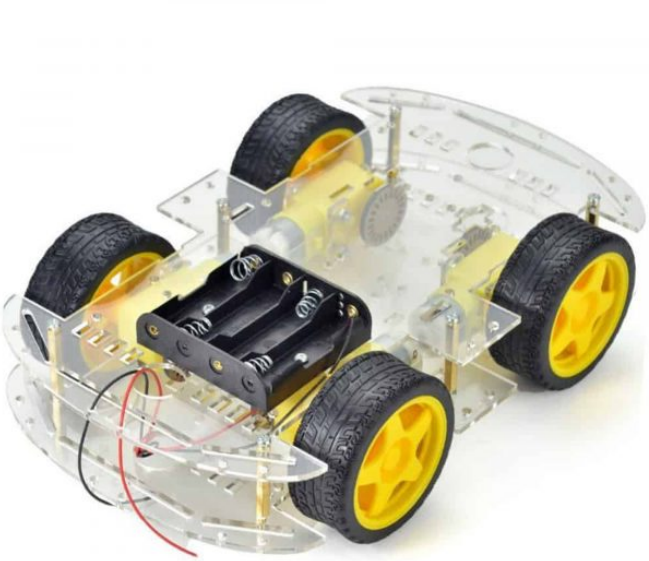


Hình 1.1: Chớp tắt led.

Đầu tiên bạn có thể thấy nó sẽ tắt led, và sau mỗi nửa giây nó lại đổi trạng thái một lần.

Đặt vấn đề

Bạn đang viết chương trình điều khiển xe mô hình, với chức năng tiến, lùi, quẹo phải, quẹo trái. Điều kiện của động cơ DC là khi đang quay sang phải thì không được quay sang trái ngay lập tức, mà phải dừng lại rồi mới được quay sang



Hình 1.2: Xe mô hình

Tài liệu tham khảo

[1] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*.

[2] Steve McConnell. *Code Complete: A Practical Handbook of Software Construction, Second Edition*.

[3] Hoàng Trang, Bùi Quốc Bảo. *Lập trình hệ thống nhúng*.

[4] Trần Đan Thư, Nguyễn Thanh Phương, Đinh Bá Tiến, Trần Minh Triết. *Nhập môn lập trình*.

[5] Trần Đan Thư, Nguyễn Thanh Phương, Đinh Bá Tiến, Trần Minh Triết, Đặng Bình Phương. *Kĩ thuật lập trình*.