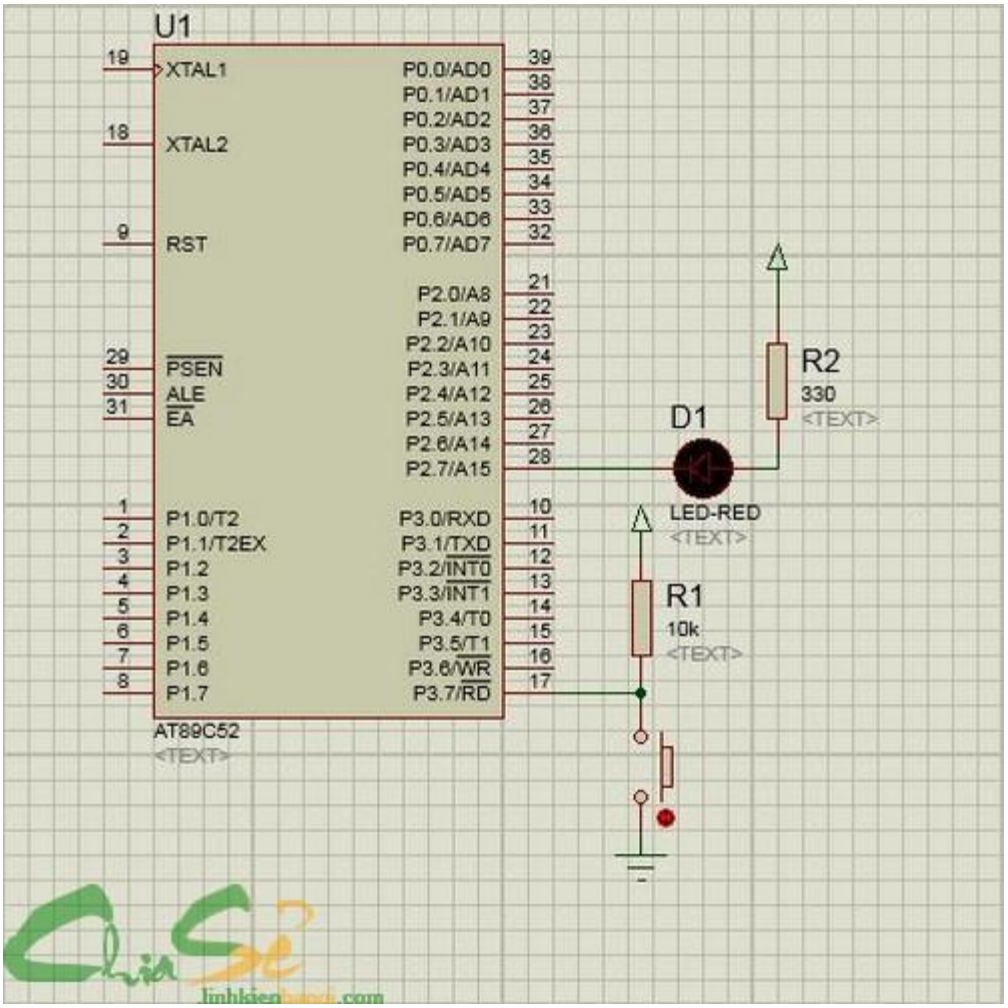
[BÀI 7]:
Lập trình nút nhấn với 8051

Hôm nay mình sẽ giới thiệu về bài toán cơ bản nhất là đọc nút nhấn cho vi điều khiển.

a: Đọc đầu vào cơ bản.
Trước tiên việc đầu tiên chúng ta cần làm là đọc được dữ liệu đang ở mức 0 hay 1. Để thực hiện bài toán này chúng ta cùng thực hiện một bài toán đơn giản như sau: Đọc dữ liệu từ 1 chân vi điều khiển và xuất dữ liệu tương ứng ra một chân khác.

Chúng ta sẽ thử bài toán này trên một mạch điện được thiết kế như sau:

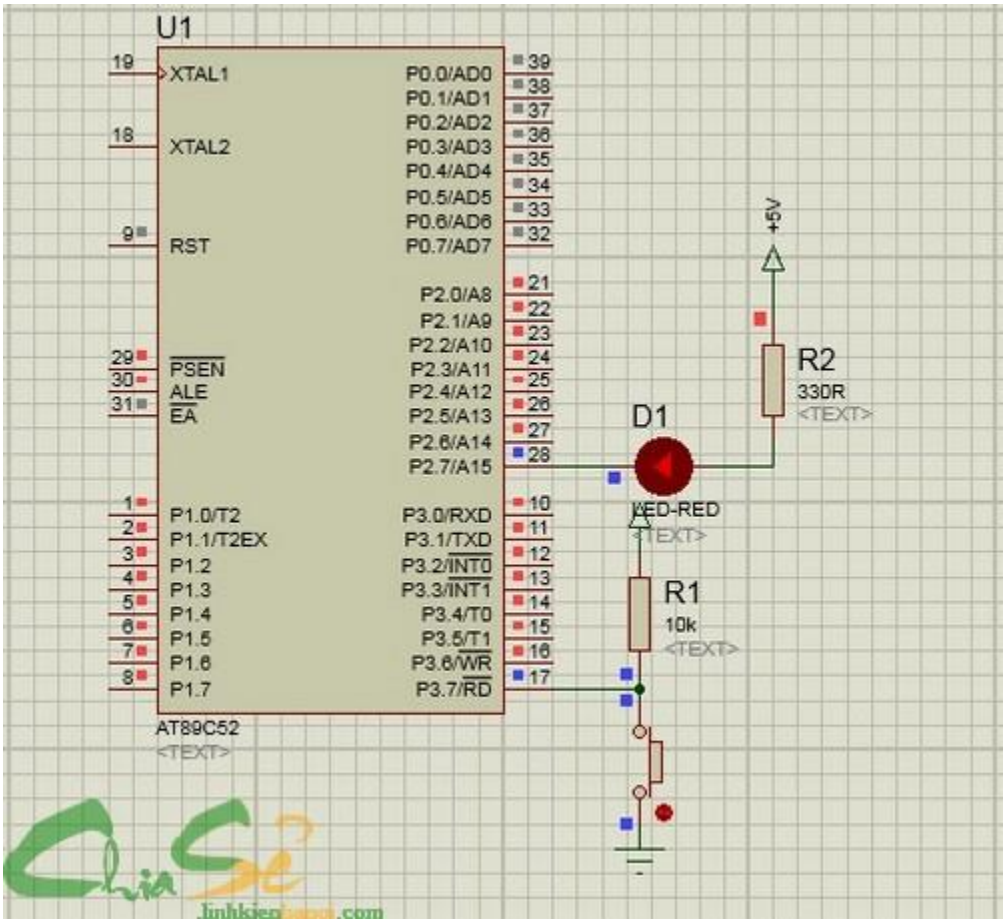


Sau đó chúng ta thực hiện việc đọc và xuất dữ liệu liên tục trong hàm while(1) như sau:

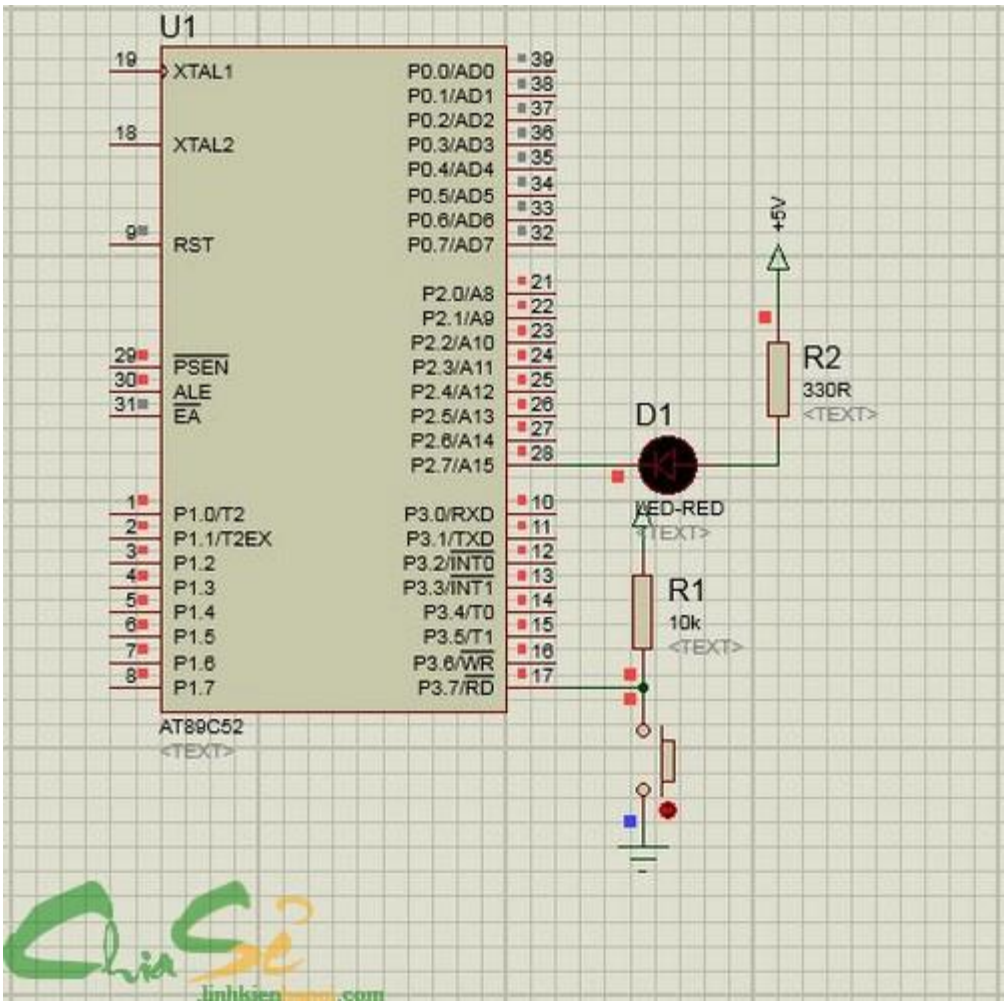
```
1  #include <REGX51.H>
2
3  /* DINH NGHIA CAC CHAN */
4  sbit BUTTON        = P3^7;
5  sbit LED    = P2^7;
6
7  /* Chuong trinh chinh */
8  void main (void)
9  {
10         while(1)
11         {
12                 /* Gan du lieu doc duoc tu nut nhan sang chan dieu khien LED */
13                 LED = BUTTON;
14         }
15 }
```

Đây là một chương trình rất đơn giản. Sau khi Build và thực hiện add file hex mọi người cho chạy mô phỏng sẽ có kết quả như sau:

- Khi nút nhấn được nhấn LED sáng



- Khi nút nhấn không được nhấn LED tắt

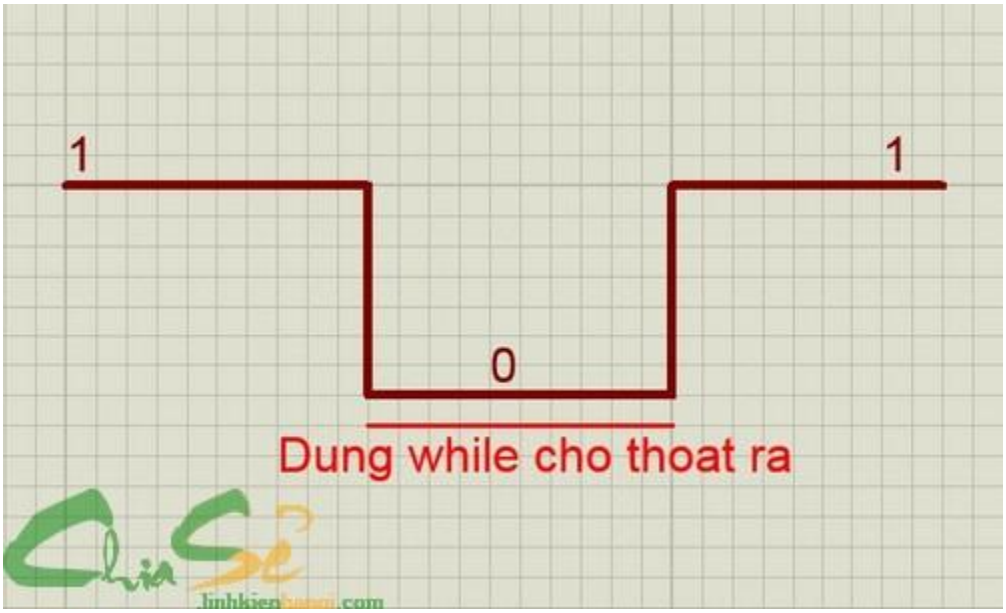


b.Đọc đầu vào và xử lý nút nhấn.

Từ phần trên các bạn đã có thể đọc được dữ liệu input của một đầu vào. Các bạn có thể thực hiện một bài toán đơn giản là khi nút giữ thì điều khiển đèn sáng và ngược lại. Tuy nhiên bài toán này khá là hiếm gặp và không cần phải dùng đến vi điều khiển :). Ở bài toán tiếp theo chúng ta sẽ nghĩ đến việc thực hiện bài toán nhấn nút thì đèn sáng nhấn thêm lần nữa thì đèn tắt.

Vẫn từ nền tảng mạch điện như trên chúng ta thực hiện chương trình theo một số hướng như sau:

Hướng giải quyết 1: Dùng hàm while để chờ chuyển mức.



Ở hướng này khi chúng ta phát hiện dữ liệu xuống mức 0 chúng ta sẽ dùng vòng lặp while chờ cho đến khi nào dữ liệu thoát khỏi mức 0 để xác nhận dữ liệu được nhấn. Hoặc áp dụng tương tự với mức 1.

Tuy nhiên, do thời gian nhấn nút thường ngắn hơn để tránh việc thời gian nằm quá nhiều trong vòng while chúng ta nên chọn mức 0 là mức chúng ta chờ thoát. Vòng lặp chờ cho đến khi dữ liệu trên chân đầu vào thoát khỏi mức 0 thì xác nhận là nút nhấn đã được nhấn xong. Lúc đó chúng ta có thể thực hiện các chức năng mà chúng ta mong muốn với việc nhấn nút này.

Ví dụ như trong đoạn code sau:

```
 1  #include <REGX51.H>
 2
 3  /* DINH NGHIA CAC CHAN */
 4  sbit BUTTON         = P3^7;
 5  sbit LED    = P2^7;
 6
 7  /* Chuong trinh chinh */
 8  void main (void)
 9  {
10          while(1)
11          {
12                  /* Kiem tra nut nhan duoc xuong muc 0 hay chua*/
13                  if(BUTTON == 0)
14                  {
```

```
15                      /* Cho den khi nut nhan duoc tha ra */
16                      while(BUTTON == 0)
17                      {
18                              ;// Trong thoi gian nay khong lam gi ca
19                      }
20                      /* Xac nhan nut nhan da thoat va thuc hien lenh dao gia tri
21 dau ra */
22                      LED = !LED;
23              }
24      }
    }
```

Đoạn code trên mô tả lại quá trình đã được nhắc ở trên. Các bạn cần chú ý cần có điều kiện if(BUTTON == 0). Nếu các bạn bỏ lệnh IF này sẽ thực sự nguy hiểm vì khi nút nhấn không được nhấn vòng WHILE không được lặp giá trị chân LED sẽ bị thực hiện lệnh đảo liên tục và đó là điều chúng ta không mong muốn. Chúng ta dùng vòng IF để kiểm tra chắc chắn nếu nút nhấn được nhấn thì chúng ta mới thực hiện lệnh chờ thoát.

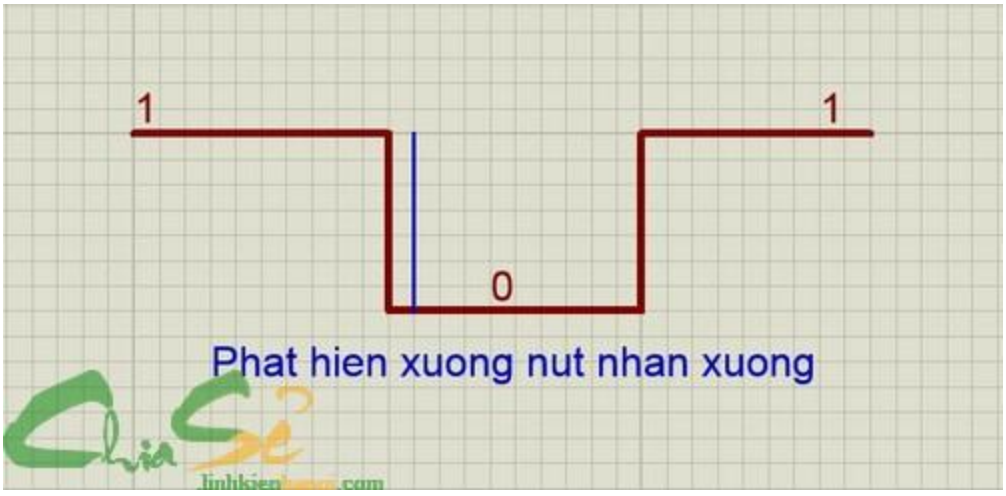
Ngoài ra còn chú ý việc chúng ta nhấn nút thường thì tiếp xúc không ngay lập tức xuống 0 mà sẽ có thể có các dao động giữa mức 1 và 0 trong một khoảng thời gian rất nhỏ trước khi dữ liệu trên chân nút nhấn về hẳn mức 0. Như thế, một lần chúng ta nhấn nút có thể vi điều khiển hiểu là rất nhiều nhấn liên tiếp.

Để tránh hiện tượng này có một thuật ngữ được dùng là *"Chống rung"* nhằm bỏ qua hiện tượng dao động được nhắc ở trên. Ý tưởng như sau: Khi phát hiện nút nhấn được nhấn chúng ta delay một khoảng thời gian nhỏ để *"vượt qua"* thời gian dao động thường gặp. Sau đó, chúng ta kiểm tra lại nút nhấn và thực hiện vòng chờ như trên.
Code ví dụ như sau:

```
 1  #include <REGX51.H>
 2
 3  /* DINH NGHIA CAC CHAN */
 4  sbit BUTTON         = P3^7;
 5  sbit LED    = P2^7;
 6
 7  void Delay (unsigned int vrui_Time)
 8  {
 9          while(vrui_Time--);
10 }
11
```

```
12 /* Chuong trinh chinh */
13 void main (void)
14 {
15          while(1)
16          {
17                    /* Kiem tra nut nhan duoc xuong muc 0 hay chua*/
18                    if(BUTTON == 0)
19                    {
20                              /* Tre mot khoang thoi gian nho */
21                              Delay(100);
22                              /* Kiem tra lai chac chan nut nhan co duoc nhan hay khong
23 */
24                              if(BUTTON == 0)
25                              {
26                                        /* Cho den khi nut nhan duoc tha ra */
27                                        while(BUTTON == 0)
28                                        {
29                                                  ;// Trong thoi gian nay khong lam gi ca
30                                        }
31                                        /* Xac nhan nut nhan da thoat va thuc hien lenh dao
32 gia tri dau ra */
33                                        LED = !LED;
34                              }
35                    }
          }
    }
```

Hướng giải quyết 2: Sử dụng so sánh dữ liệu trước và sau để phát hiện chuyển mức.



Cách ở trên chúng ta đã có thể phát hiện được sự kiện nhấn nút, tuy nhiên có một số nhược điểm mà cách giải quyết này tồn tại như: Khi chúng ta thực hiện nhấn nút quá lâu thì trong lúc đó vi điều khiển không thực hiện việc gì, nếu người lập trình vẫn muốn vi điều khiển thực hiện các lệnh thì lại lựa chọn những dọn code mong muốn chạy đưa vào vong while này. Đây là một việc bất tiện và nếu như có nhiều nút nhấn sẽ rất khó khăn cho người lập trình. Vì thế chúng ta cần lựa chọn một cách giải quyết khác.

Ý tưởng: Chúng ta thực hiện dùng dữ liệu lưu trữ lại để so sánh giữa mức ngay trước đó và mức mới. Nếu có sự sai khác giữa mức mới và mức ngay trước đó thì có nghĩa là có sự thay đổi về giá trị nút nhấn và tùy thuộc vào mục đích của người lập trình chọn sườn lên hay sườn xuống hoặc cả 2 sườn để thực hiện các lệnh lập trình theo ý muốn. Ở bài toán này mình chọn sườn xuống của nút nhấn để thực hiện lệnh điều khiển.

Ý tưởng trên được thực hiện như sau:

```
 1  #include <REGX51.H>
 2
 3  /* DINH NGHIA CAC CHAN */
 4  sbit BUTTON         = P3^7;
 5  sbit LED    = P2^7;
 6
 7  /* Khai bao bien Button_Old chua du lieu ngay truoc do cua nut nhan */
 8  bit Button_Old;
 9
10 void Delay (unsigned int vrui_Time)
11 {
12          while(vrui_Time--);
13 }
14
15 /* Chuong trinh chinh */
16 void main (void)
17 {
18          while(1)
19          {
20                  /* Kiem tra nut nhan duoc xuong muc 0 hay chua*/
21                  if(BUTTON == 0)
22                  {
23                          /* Tre mot khoang thoi gian nho */
24                          Delay(100);
```

```
25                      /* Kiem tra lai chac chan nut nhan co duoc nhan hay khong
26 */
27                      if((BUTTON == 0) && (Button_Old == 1))
28                      {
29                              /* Xac nhan nut nhan duoc nhan va thuc hien lenh dao
30 gia tri dau ra */
31                              LED = !LED;
32                      }
33              }
34              /* Cap nhat gia tri ngay truoc do cua nut nhan */
35              Button_Old = BUTTON;
36              Delay(100);
        }
    }
```

Ở đây chúng ta sử dụng một biến khai báo là biến *Button_Old* để lưu trữ dữ liệu ngay trước đó của nút nhấn tức là giá trị ngay trước thời điểm chúng ta đọc giá trị mới của nút nhấn. Giá trị *Button_Old* là giá trị của *BUTTON* được trễ lại. Dữ liệu này được cập nhật liên tục sau mỗi lần chúng ta sử dụng xong giá trị BUTTON để đảm bảo*Button_Old* luôn là giá trị gần với giá trị *ít cũ nhất* của nút nhấn. Sự kiện nút nhấn xuống tương đương với việc dữ liệu mới của nút nhấn bằng 0 và dữ liệu gần nhất của nó bằng 1. Dòng lệnh 26 thực hiện kiểm tra việc này. Nếu điều kiện đó thỏa mãn thì xác nhận là nút nhấn vừa được nhấn xuống. Và ở đây chúng ta vẫn sử dụng một hàm Delay nhỏ để chống rung.
Sau khi đưa đoạn lập trình trên vào các bạn thực hiện mô phỏng sẽ thấy ngay khi nút nhấn được nhấn dữ liệu LED đã được thay đổi mà không cần chờ nút nhấn được thoát như trước nữa.

Như vậy, chúng ta đã tìm hiểu thêm được vấn đề dữ liệu đầu vào và đọc dữ liệu nút nhấn, ngoài hai cách trên còn có một số cách khác để thực hiện bài toán này các bạn có thể thực hiện triển khai thêm. Ngoài ra, có một cách kiểm soát dữ liệu nút nhấn khác bằng ngắt ngoài, về mục này khi tìm hiểu đến ngắt ngoài mình sẽ nhắc lại và chúng ta sẽ cùng tìm hiểu thêm.

3.Bài tập:

Bài 1: Thực hiện một mạch mô phỏng gồm 04 LED 7 thanh được điều khiển theo phương pháp quét LED *(Bạn nào chưa rõ có thể tìm hiểu bài 05)*, 02 nút nhấn. Một nút có chức năng tăng số đang hiển thị, một nút có chức năng giảm số đang hiển thị.

- *Chú ý thức hiện theo hai cách thực hiện trên và đưa ra so sánh về hai kiểu đọc nút nhấn đã được nhắc đến.*

Chúc các bạn thành công !