# GIÁO TRÌNH ROBOCON

Robocon đã được tổ chức từ năm 2002.Cuộc thi này đã cuốn hút rất nhiều bạn sinh viên có niềm đam mê khoa học kĩ thuật.Cuộc thi này đã đem lại cho

chúng ta rất nhiều những kiến thức về kĩ thuật ,rèn luyện đức tính kiên trì ,bền bỉ ,dám đối mặt với gian khổ .Hơn nữa ,cuộc thi đã để lại những kỉ niệm đẹp về một thời sinh viên.

Giáo trình này được biên soạn nhằm giúp các bạn có được một cái nhìn tổng quan về robot ,sự định hướng cụ thể và những kinh nghiệm quý báu khi tham gia robocon

Giáo trình gồm 8 bài :

**Bài 1: Giới thiệu về Robot Các Modul của robot.**
**Bài 2: Mạch điều khiển robot bằng tay.**
**Bài 3: Mạch cảm biến dò đường.**
**Bài 4: Mạch công suất điều khiển động cơ .**
**Bài 5: Mạch vi điều khiển.**
**Bài 6: Lập trình và kỹ thuật dò đường .**
**Bài 7: Lập trình các thao tác khác và chiến thuật.**
**Bài 8: Chương trình hoàn thiện cho một robot tự động**

Toàn bộ các bài học đều được thực hành trên mạch và robot thật.Sử dụng mạch điện và robot của đội BK-FIRE.

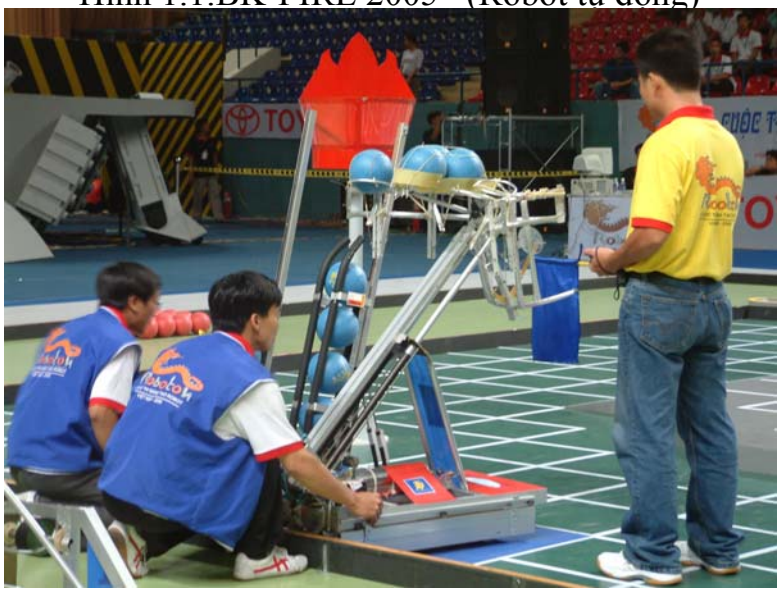
# BÀI 1.GIỚI THIỆU VỀ ROBOT VÀ CÁC MODUL CỦA ROBOT

**Bài này sẽ giới thiệu cho các bạn một cách tổng quan về robot và các modul của robot cũng như một số kinh nghiệm khi làm robocon.**

## 1.1.Giới thiệu về robot

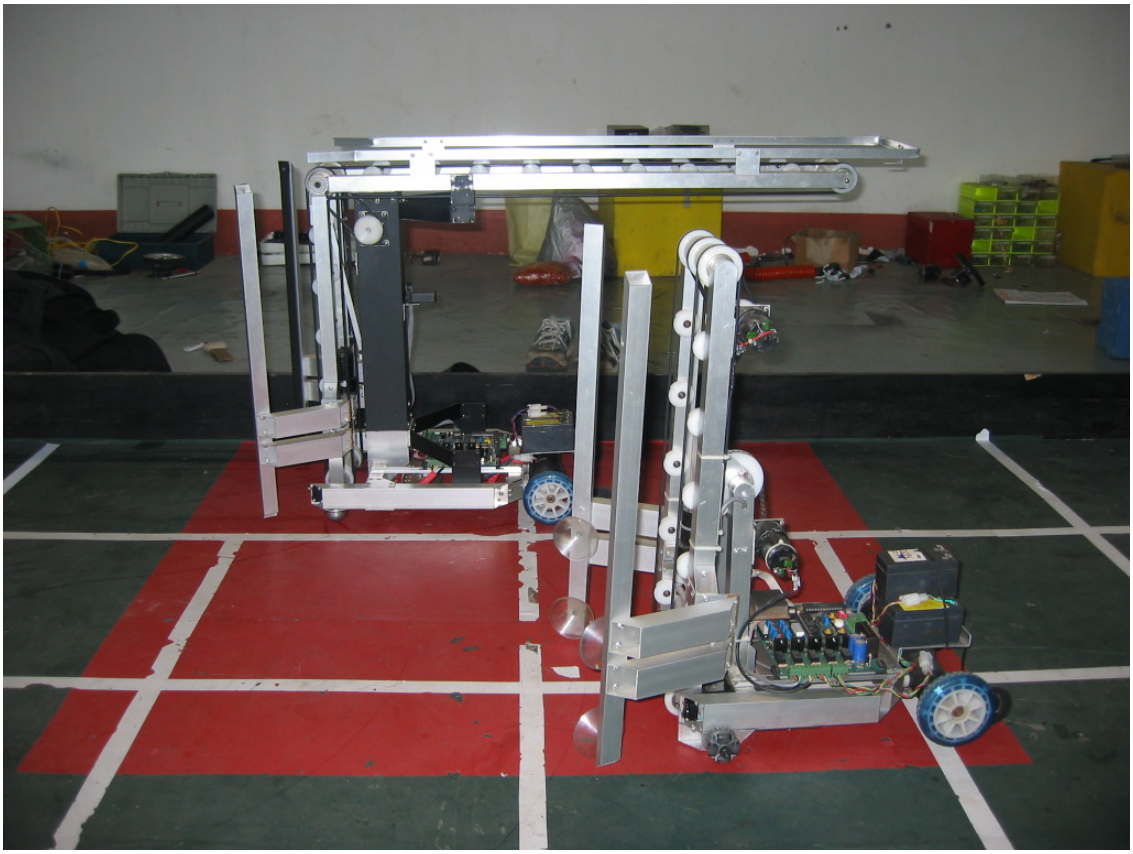
1.1.1.Những hình ảnh về robot trong các cuộc thi robocon

Hình 1.1.BK-FIRE 2005 (Robot tự động)

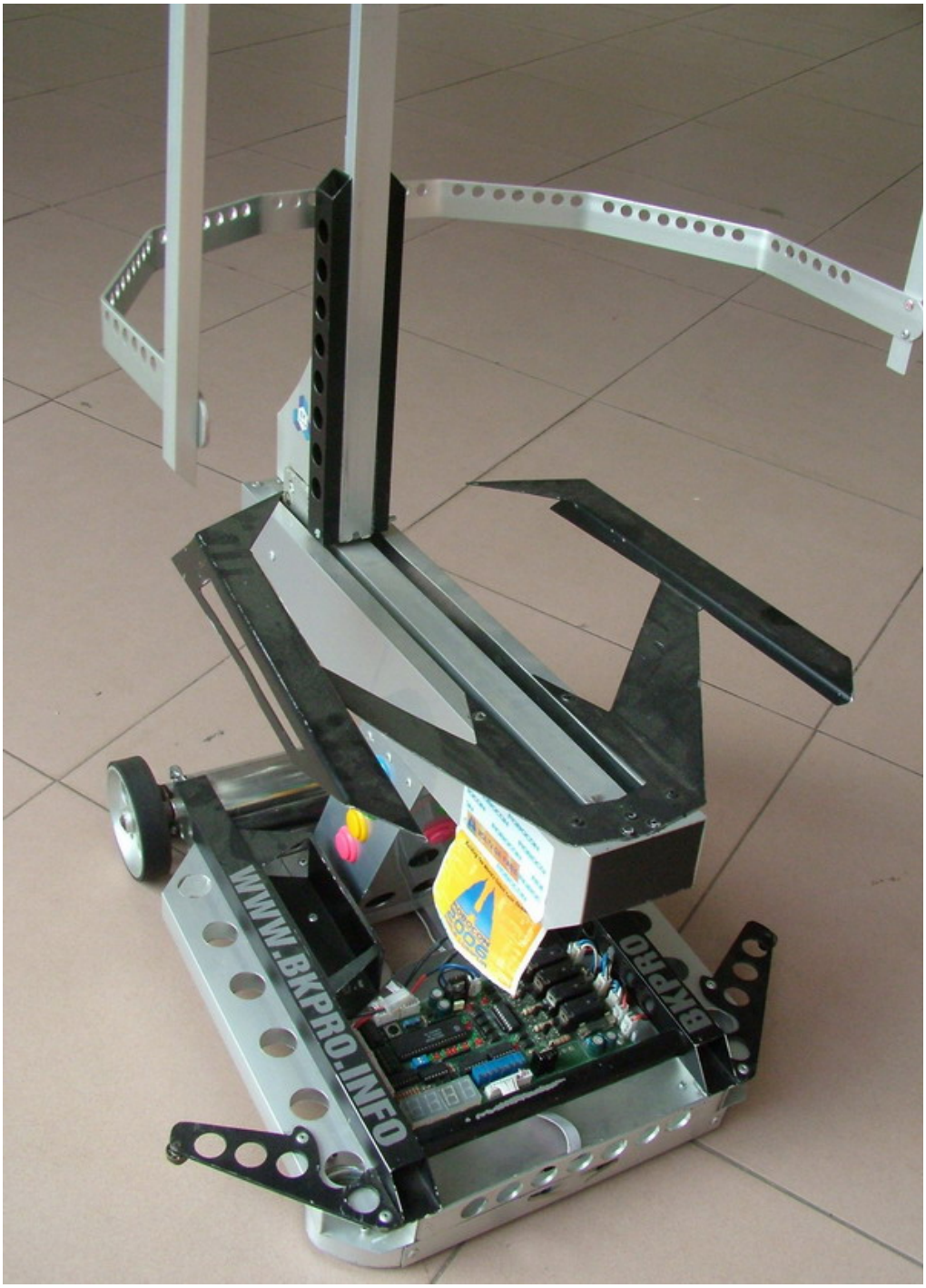


Hình 1.2.BK-FIRE 2005 (Robot bằng tay)


Hình 1.3. FXR 2004 (robot tự động)



Hình 1.4.BKPRO 2006




1.1.2.Các modul trong robot

Trong robocon ,có 2 loại robot là robot tự động và robot điều khiển bằng tay.

Robot bao gồm 3 phần chính :Cơ khí ,mạch điện tử và lập trình

**+ Cơ khí :**

Phần này bao gồm các kết câú cơ khí của robot .Phần này mỗi năm đều thay đổi tuỳ thuộc vào đề thi.Đây chính là phần thể hiện ý tưởng của bạn.Khi có một ý tưởng độc đáo ,một kết cấu cơ khí tốt ,bạn sẽ có nhiều cơ hội dành chiến thắng.Trong cuộc thi robocon ,ý tưởng đóng vai trò rất quan trọng .Do đó ,khi thiết kế cơ khí ,toàn đội cần phải họp bàn kĩ lưỡng để đưa ra một giải pháp thiết kế cơ khí tối ưu nhất.Để thiết kế cơ khí ,các bạn có thể dung các phần mềm thiết kế cơ khí chuyên dụng như Solid work ,Autocad.

Hình 1.3.Cơ cấu bánh xích


.

Hình 1.4.Cơ cấu khí nén

Cơ cấu khí nén được BK-FIRE sử dụng đầu tiên trong cuộc thi robocon 2005 .Từ đó ,cơ cấu này đã được rất nhiều đội sử dụng và đã rất thành công như FAS-01 ,BK-FIRE (2006).

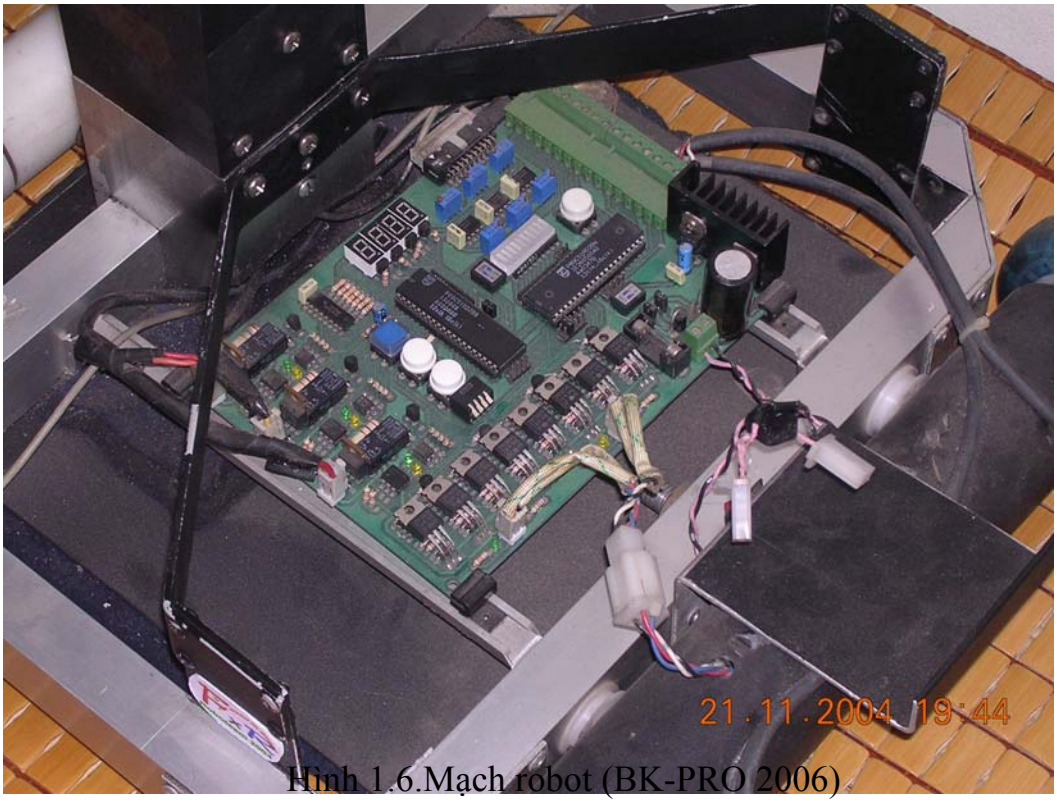
**+ Mạch điện tử**

Mạch điện tử trong robot đóng vai trò như những mạch máu trong một cơ thể người .Nó đóng vai trò kết nối toàn bộ những bộ phận khác trong robot thành một thể thống nhất.

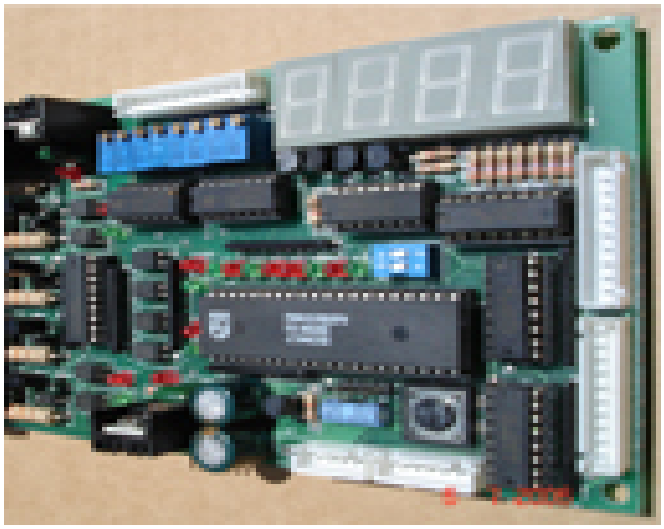
Mạch điện tử trong robot bao gồm các phần chính :

- Mạch vi điều khiển (8051 ,PIC ,AVR) đóng vai trò như bộ não của robot điều khiển toàn bộ hoạt động của robot theo chương trình lập sẵn.
- Mạch sensor đóng vai trò như các giác quan của robot để giúp robot nhận dạng môi trường xung quanh để gửi tín hiệu về vi điều khiển.
- Mạch công suất điều khiển động cơ ,dùng để điều khiển các cơ cấu của robot
- Mạch chiến thuật thi đấu :cho phép người sử dụng có thể tuỳ ý lựa chon chiến thuật khi ở trên sân thi đấu.

Hình 1.5.Mạch điện tử trong robocon (FXR)




Hình 1.6.Mạch robot (BK-PRO 2006)



**+Lập trình**

Chương trình trong robot giống như bộ não của một cơ thể người ,robot chỉ hoạt động được khi có chương trình cài đặt sẵn cho nó.Các chiến thuật thi đấu ,đường chạy của robot là do người lập trình quyết đinh .Hai ngôn ngữ được sử dụng nhiều trong robocon đó là C và ASM .Tuy nhiên ,ngôn ngữ C được sử dụng nhiều nhất vì C có cấu trúc dễ debug lỗi và đặc biệt thuận tiện

khi làm những chương trình lớn (Các chương trình trong robocon rất phức tạp).
Để một robot có thể hoạt động tốt ,cần phải có sự phối hợp chặt chẽ giữa các phần cơ khi ,mạch ,và lập trình .Nếu một bộ phận không tốt ,thì robot sẽ không hoạt động được.

**1.2.Một số kinh nghiệm khi tham gia robocon.**

1.2.1.Tổ chức đội hình.

Một đội robot bao gồm 7 đến 10 người ,và thường được chia thành 3 nhóm
-Nhóm cơ khí :Thiết kế ,gia công cơ khí (3-4 người).
-Nhóm mạch điện tử :Thiết kế ,làm mạch cho robot (2-3 người).
-Nhóm lập trình: Lập trình ,test robot (3 người)

Người đội trưởng phải có trách nhiệm phân công rõ ràng công việc cho từng người ,các thành viên phải nghiêm túc chấp hành tránh hiện tượng chê ỳ ,ỷ lại.

1.2.2.Mua sắm thiết bị linh kiện

Đội robot phải có định hướng rõ ràng ,làm robot cần có những phương tiện gì ,địa điểm mua hàng cụ thể.
Công cụ cơ khí :Khoan máy ,cưa máy ,cưa tay ,đinh rút : Địa điểm :Chợ giời (Phố Huế)
Công cụ điện tử :Mỏ hàn ,hút thiếc ,đồng hồ điện tử.v.v.v Địa điểm :Trần Cao Vân ,17 Hàn Thuyên ,70 Hàng Trống ,269 Đội Cấn.

1.2.3.Tìm kiếm tài trợ

Để làm robot ,bạn phải tốn rất nhiều tiền (trung bình từ 20 đến 30 triệu và có thể nhiều hơn nữa) .Với điều kiện sinh viên ,các bạn rất khó có thể có được số tiền như thế ,Do đó ,bạn cần phải tìm kiếm các nhà tài trợ từ các doanh nghiệp.Tuy nhiên để xin được tài trợ ,bạn cần phải có phương án kế hoạch cụ thể ,hay những mối quan hệ khác.
-Địa chỉ một số doanh nghiệp các bạn có thể xin tài trợ

a) Công ty Elcom 18 Nguyễn Chí Thanh
b) Công ty cổ phần kỹ thuật SEEN (Từ Liêm)
c) Công ty Ameco (Tầng 1 toà nhà sông đà Phương Liệt)

# BÀI 2. MẠCH ĐIỀU KHIỂN ROBOT BẰNG TAY

Robot điều khiển bằng tay là một phần không thể thiếu được trong robocon .Trong một số trận đấu ,robot bằng tay có vai trò hỗ trợ cho robot tự

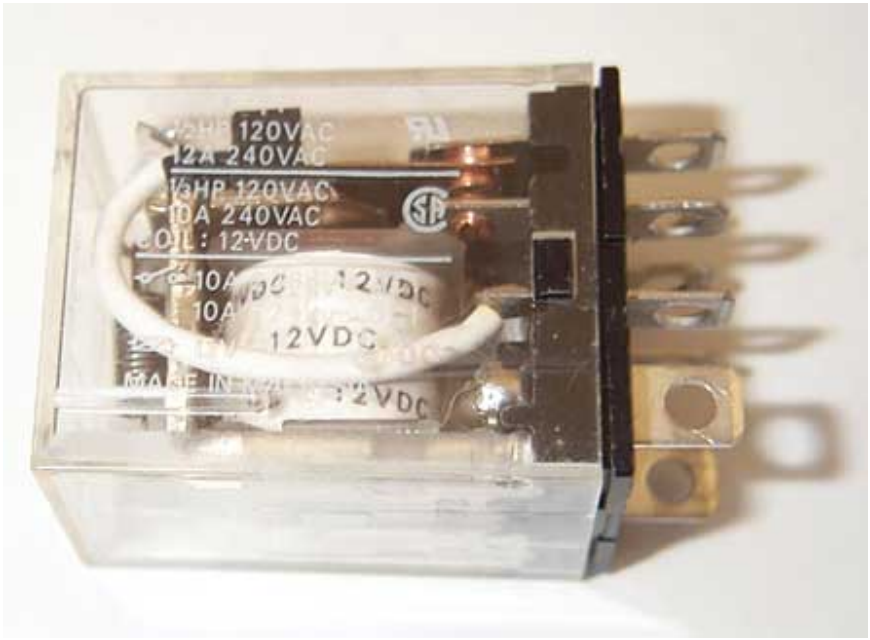
động .Đôi khi ,robot bằng tay đã giúp thay đổi cục diện trên sân.Mạch điều khiển robot bằng tay phần lớn là mạch relay điều khiển động cơ.

2.1.Chức năng của mạch điều khiển bằng tay

Mạch điều khiển bằng tay dùng để điều khiển các động cơ của robot bằng tay.Việc điều khiển động cơ bao gồm các chức năng sau

- Điều khiển động cơ quay ngược (robot lùi )
- Điều khiển động cơ quay xuôi (robot tiến )
- Dừng động cơ (dừng robot)

Relay được dùng trong robot bằng tay là loại relay OMRON ,dòng lớn

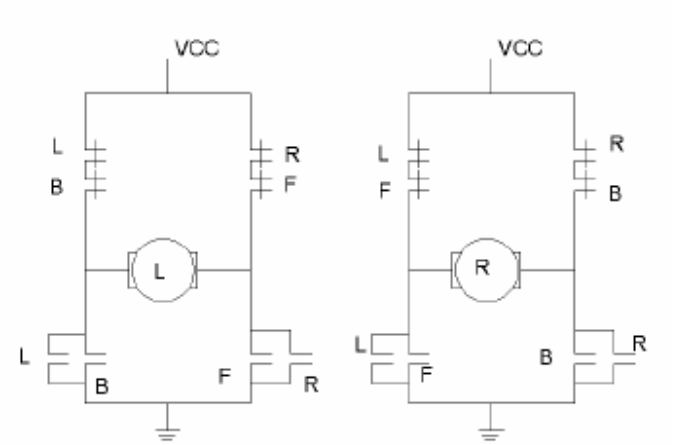



Hình 2.1.Role omron


2.2.Nguyên tắc hoạt động mạch điều khiển robot bằng tay

Mạch này sử dụng 2 relay điều khiển 1 động cơ ,cấp nguồn cho relay thứ 1 thì động cơ quay phải ,cấp nguồn cho relay thứ 2 thì động cơ đảo chiều.

Sơ đồ mạch điều khiển bằng tay




Bạn thiết kế một mạch điều khiển sử dụng các nút bấm để đóng mở các relay ,mạch rất đơn giản và hiệu quả.Tuy nhiên ,làm như vậy sẽ không thẩm mỹ và bất tiện .Do đó bạn có thể thiết kế bộ điều khiển bằng gamepad PS 2 để điều khiển .

(Phần này có tham khảo tài liệu trên mạng)

Gamepad cho Robot điều khiển bằng tay chia làm 2 loại : điều khiển từ xa bằng hồng ngoại và điều khiển qua dây cáp. Nếu dùng các mạch thu phát hồng ngoại thông thường thì tính ổn định khi thi đấu không cao (!), tốt nhất nên dùng Vi xử lí (VXL) để thu phát theo tần số riêng của mình (như POL chẳng hạn). Điều khiển qua dây cáp mang tính ổn định rất cao, ở đây chúng tôi đề cập tới cách sử dụng gamepad qua dây cáp.

**1.Sử dụng gamepad playstation của SONY.**

Kết nối trực tiếp vào Vi xử lí. Rất gọn khi thiết kế phần cứng, tuy nhiên chúng ta nên kèm theo nút Reset VXL trên gamepad để tăng tính ổn định. Đây là phương pháp sử dụng của đội FXR.

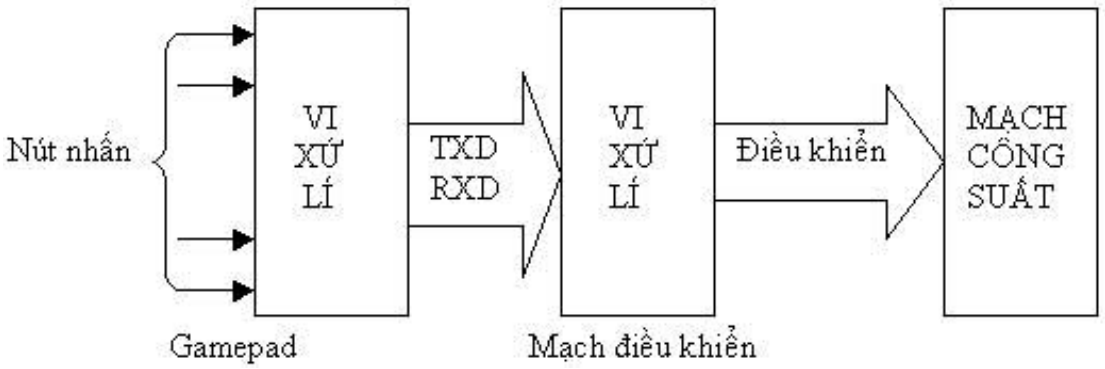
**2.Cải tạo phần mạch của 1 gamepad bất kì:**

Trong mạch của các gamepad bất kì, các nút nhấn được thiết kế như là các công tắc hành trình (tức là có 1 điểm chung nối mass).

+ Cắt bỏ đường mạch in nối các nút nhấn với phần điều khiển của gamepad.

+ Sử dụng các nút nhấn như là các công tắc hành trình độc lập.

a. Truyền nối tiếp :

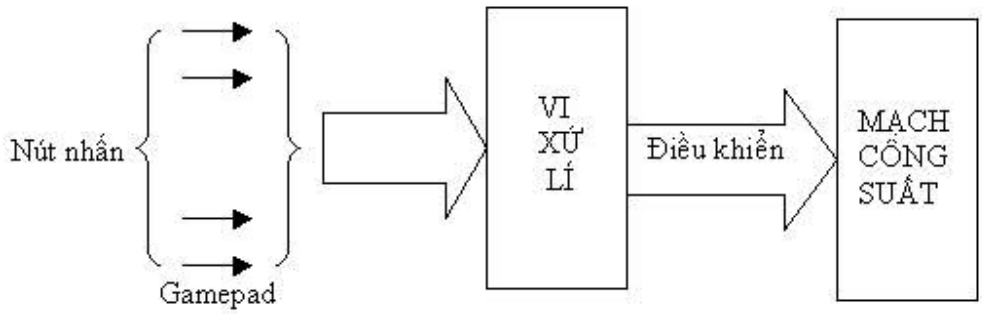



Theo cách xử lí này chúng ta chỉ sử dụng cáp 4 sợi : +5v, mass, TXD, RXD.

Cách xử lí này vẫn chưa ổn định tuyệt đối nhưng sử dụng ít Port VXL. Nên kèm theo nút Reset VXL trên gamepad để tăng tính ổn định. Đây là phương pháp Basic Bot sử dụng năm 2003 và 2004.

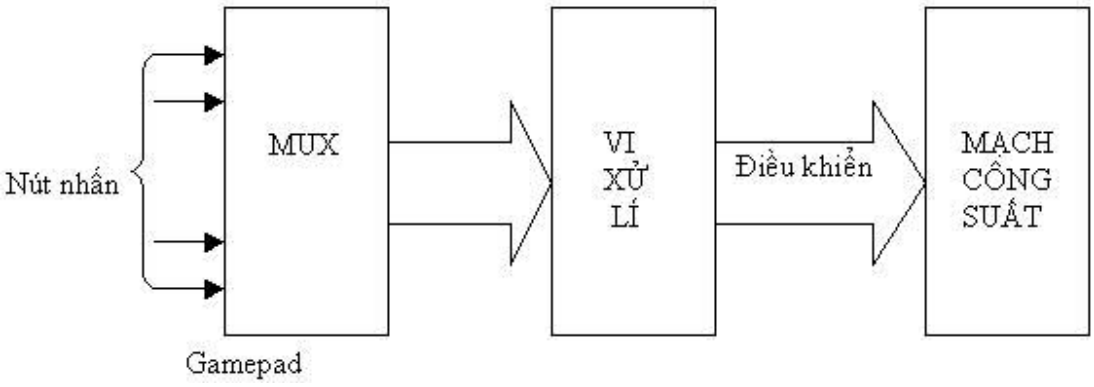
b. Truyền song song :

Thực chất ta nối tất cả các nút nhấn xuống thẳng mạch điều khiển, khi đó các nút
nhấn đóng vai trò như là những công tắc hành trình. Phương pháp này rất ổn định, tuy nhiên nhược điểm là rất tốn Port VXL (mỗi nút nhấn ứng với 1 Port).




Cách khắc phục nhược điểm trên :

Sử dụng mạch dồn kênh 16 sang 1.




Phương pháp này giảm bớt được đáng kể số lượng Port VXL, tính ổn định cao.
Thiết kế mạch robot bằng tay kiểu bán tự động đã được FXR sử dụng trong cuộc thi robocon 2004 rất thành công (đoạt chức vô địch )

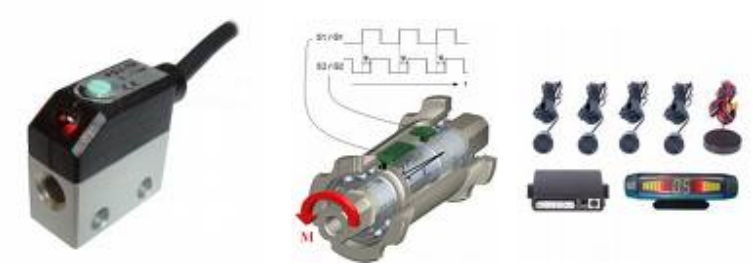
# BÀI 3:MẠCH CẢM BIẾN DÒ ĐƯỜNG

Bài học này giới thiệu về mạch cảm biến trong robocon.Mạch cảm biến đóng vai trò như "mắt " của robot giúp cho robot có khả năng nhận biết được môi trường xung quanh (vạch trắng ,chướng ngại vật ) để từ đó có biện pháp xử lý.
Trong mạch cảm biến ,các bạn có thể sử dụng các loại led thu phát hồng ngoại hoặc quang trở .Led thu phát hồng ngoại có giá thành rẻ (2500 /cặp).

Hình 2.1.Sensor hồng ngoại.

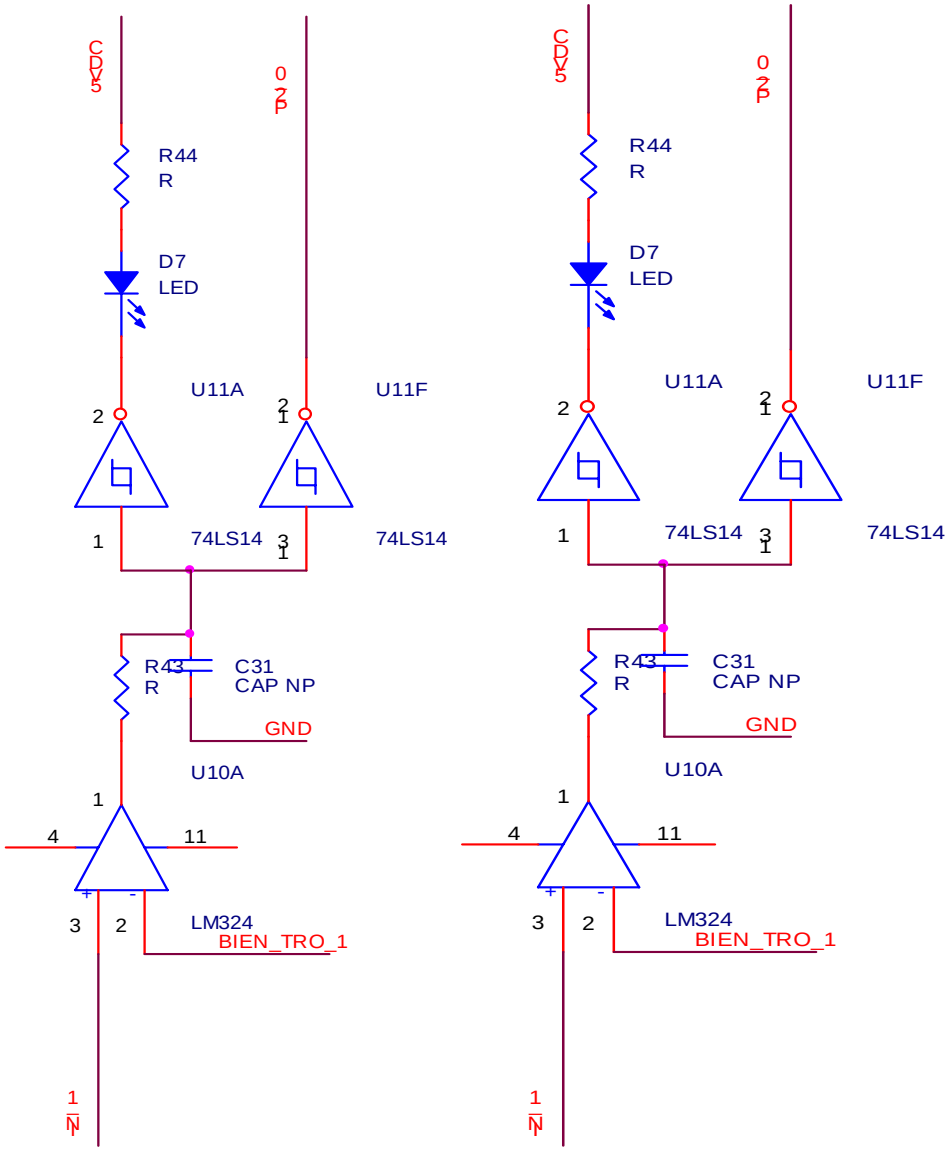
Đối với robot công nghiệp ,bạn có thể sử dụng những loại cảm biến công nghiệp của các hãng Omron ,hay Siemens.Các loại cảm biến này có độ nhạy cao và khả năng chống nhiễu tốt.Tuy nhiên giá thành rất đắt (hàng trăm đô),cho nên sử dụng chúng trong robocon thì không phù hợp .Bạn hoàn toàn có thể tự chế tạo mạch sensor cho mình.

Hình 2.2.Một số sensor dùng trong công nghiệp

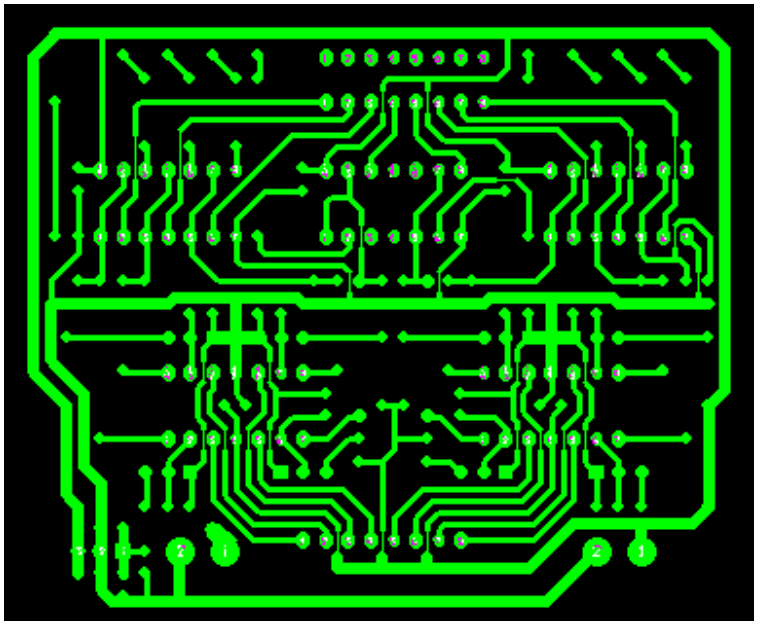


## 3.1.Mạch sensor phát thẳng.

### 3.1.1.Sơ đồ nguyên lý




### 3.1.2.Sơ đồ mạch in



3.1.3.Nguyên lý hoạt động.

Trong sơ đồ nguyên lý trên ,cặp led thu phát được đặt sát nhau ,một chân của led thu được đưa và LM324 .LM324 là IC khuếch đại so sánh ,mục đích khuyếch đại tín hiệu từ sensor .Tín hiệu ra từ LM324 được đưa và IC 7414 (Triger inverter) để đảo mức tín hiệu ,tín hiệu ra được đưa vào chân của vi điều khiển.Chiết áp đóng vai trò chỉnh độ nhạy của sensor ,đèn led có tác dụng báo hiệu khi sensor gặp vạch trắng.

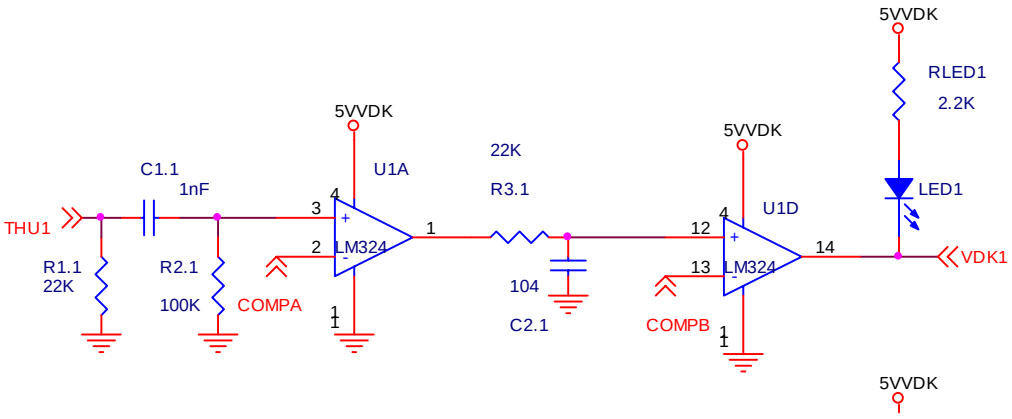
Khi hoạt động ,led phát chiếu tia hồng ngoại xuống sân thi đấu ,khi chùm tia hồng ngoại chiếu xuống nền đen , led phát không nhận được chùm phản xạ nên không dẫn ,điện trở bằng vô cùng ,tín hiệu ra LM324 là mức 1 ,qua 7414 bị đảo mức tín hiệu về 0 (led báo sang ,tín hiệu vào vi điều khiển là mức 0).Tương tự ,khi robot gặp nền trắng ,mức tín hiệu ngược lại ,led báo tắt ,tín hiệu vào vi điều khiển mức 1.

3.1.4.Ưu nhược điểm

Mạch sensor phát thẳng có ưu điểm là đơn giản ,dễ chế tạo ,tiết kiệm chi phí .Tuy nhiên ,nhược điểm của mạch sensor phát thẳng là khả năng chống nhiễu kém .Khi trên sân thi đấu ,ánh sáng đèn cao áp sẽ ảnh hưởng đến khả năng hoạt động của sensor .Do đó ,sensor phải được che chắn kĩ lưỡng để khỏi ảnh hưởng đến khả năng hoạt động của sensor (đặt trong hộp kín ,hoặc bọc băng dính đen các sensor).

## 3.2.Mạch sensor phát xung

Đặc điểm của mạch sensor phát xung đó là khả năng chống nhiễu tốt ,không bị ảnh hưởng bởi các điều kiện bên ngoài như ánh sáng.Tuy nhiên ,mạch phát xung có nhược điểm là phức tạp ,không tiện lợi




Trên thực tế ,trong các cuộc thi robocon đa số các đội sử dụng mạch sensor phát thẳng .Bởi vì mạch sensor phát thẳng tương đối đơn giản ,dễ sử dụng

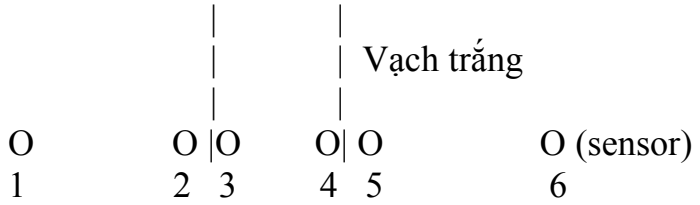
### 3.3.Bố trí sensor

Trên robot ,về nguyên tắc ,sử dụng nhiều sensor thì khả năng bám đường càng tốt .Tuy nhiên ,trên thực tế bạn chỉ cần sử dụng 6 đến 8 sensor là

đủ.Cách bố trí sensor sẽ tuỳ thuộc vào ý tưởng và chất lượng các loại động cơ bạn sử dụng trong robot.

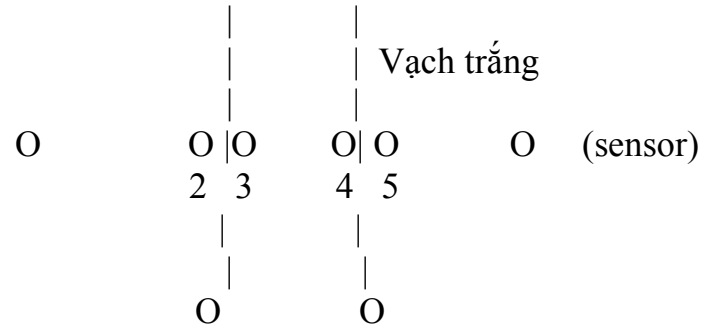
Xin giới thiệu 2 phương pháp bố trí sensor trên robot

a) Sử dụng 6 sensor xếp thành hàng ngang.




Cách này thích hợp đối với việc sử dụng các loại động cơ có độ hãm tốt ,quay ngã tư quay bằng 1 bánh (bánh quay ,bánh dừng).Trong cách bố trí này ,sensor 2,3,4,5 được sử dụng để bám đường ,hai sensor ngoài cùng 1 và 6 được dùng để đếm ngã tư và quay phải ,quay trái.Phương pháp bố trí sensor này đã được FXR sử dụng năm 2004 và đã đoạt chức vô địch.

a) Sử dụng 8 sensor

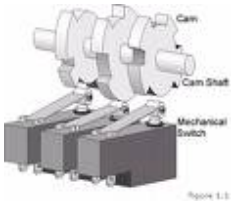



Cách này thường được áp dụng với các loại động cơ có độ hãm kém ,quay ngã tư bằng 2 bánh.( 2 bánh quay ngược chiều nhau).Ở sơ đồ trên ,2 sensor được lắp ở phía dưới đế của robot để bắt ngã tư ,còn 6 sensor phía trên dùng để bám đường.Đây là cách mà BK-FIRE sử dụng trong cuộc thi robocon năm 2005.

**3.4.Công tắc hành trình**

Công tắc hành trình được sử dụng khá nhiều trong robocon .Công tắc hành trình thường được lắp đặt ở các cơ cấu phía trên ,như cơ cấu truyền động ,gắp quả ,bỏ bóng.

Hình 3.3.Công tắc hành trình




Công tắc hành trình có 3 chân : Ở dạng thường đóng ,thường mở. Một chân tín hiệu được nối đất hoặc VCC , một chân được nối với vi điều khiển.

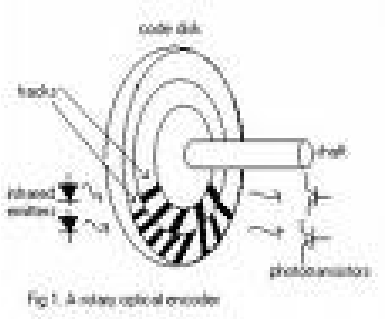
Trong robocon ,việc bố trí công tắc hành trình cần phải linh hoạt ,khéo léo để phát huy tối đa khả năng nhận biết của robot.

**3.5.Encoder**

Một công nghệ mới đã được một số đội robot áp dụng trong những năm gần đây là sử dụng các bộ encoder .

Cơ sở của phương pháp này là sử dụng đặc tính độ dài quãng đường đi được và sai số chuyển động giữa hai bánh của động cơ nhờ khai thác tính chất của bộ mã hoá xung vòng quay (Rotary encoder).

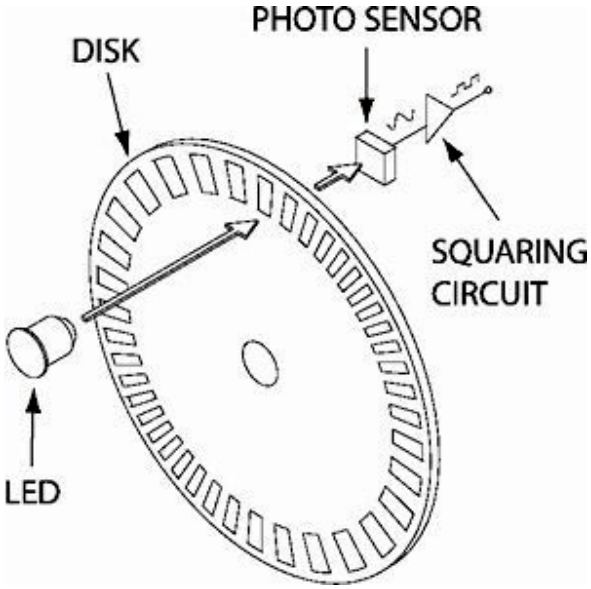
Bộ encoder thực chất là một đĩa có đục lỗ có gắn cặp sensor thu phát ở 2 bên .Bộ encoder được gắn trên bánh xe hoặc trên động cơ của robot




Nguyên tắc hoạt động của encoder

Gồm một bộ thu phát hông ngoại và một đĩa cho chia lỗ được đặt giữa hệ thống thu phát này. Đĩa được gắn trên trục của động cơ hoặc trục chuyển

động. Quá trình đĩa chuyển động làm cho phần photo sensor thay đổi trạng thái và tạo ra một chuỗi các xung vuông trên đầu ra. Đây là thông số kĩ thuật quan trọng của một encoder. Tuỳ theo số lỗ trên đĩa mà số xung tạo ra trong một vòng quay của đĩa khác nhau. Số lượng xung càng lớn nghĩa là số lỗ càng nhiều trên một vòng tròn 360. Nghĩa là ta càng có thể điều khiển chính xác. Và dĩ nhiên bộ encoder càng đắt tiền. Chúng ta có thể thấy có nhiều loại encoder dùng từ trường hoặc trên đĩa có nhiều vòng lỗ nhưng tôi giới thiệu loại phổ dụng và đơn giản nhất là sử dụng ánh sáng như trên. Trong thực tế chúng ta có thể thấy các bộ encoder trên các động cơ DC chứ không chỉ là các thành phần độc lập. Việc lựa chọn và sử dụng hai loại encoder này đều có những ưu và nhược điểm riêng mà tôi không đề cập ở đây. Hình dạng các encoder :




Khi robot muốn dịch chuyển theo một quỹ đạo xác định cần căn cứ vào hai trạng thái cơ bản đó là quãng đường đã đi được của hai bánh (tham chiếu với chương trình, và hướng chuyển động), Sai số quãng đường giữa hai động cơ.

Phương pháp này thể hiện đặc tính rất mềm dẻo khi phải chuyển hướng vì có thể xoay robot một góc bất kì ra khỏi quỹ đạo chính.

Encoder thường có 6 dây (hoặc 4 dây tùy loại ) bao gồm 2 dây nguồn ,2 dây tín hiệu A và B và dây pha Z.2 dây tín hiệu A và B cho phép bạn xác định số vòng quay của động cơ ,vận tốc và chiều quay của động cơ.Để lập trình xử l tín hiệu encoder ,bạn có thể nối 2 dây tín hiệu A và B vào 2 chân timer hoặc ngắt ngoài của vi điều khiển,thiết lập vi điều khiển ở chế độ counter ,vi điều khiển sẽ đếm xung từ vi điều khiển.

Bạn có thể tham khảo code dưới đây

```
void ngat0(void) interrupt 1 // ngat bo dinh thoi 0
 {

      if (kt0==0)  // bit nay set de dung trong luc quay va di lap ma
       {
        P1_2=1;
        P1_3=1;
        P1_0=1;
        P1_1=1;
       }

      if (kt0==1)  // bit nay xoa de bam xung dinh thoi new
        {
         TR0=0;
         if (dem_trai==tocdo_trai)
         { P1_2=~P1_2;tocdo_trai=10-tocdo_trai;dem_trai=0;}
         dem_trai++;
         TR0=1;
        }

      if (kt0==2) // dung cho bamxung quay
        {
         TR0=0;
         TH0=0xFF;
         TL0=0x47;
         d2++;
         TR0=1;
         }
```

```
    }
        // quay dung bo lap ma
void quayphai(unsigned char bytecao,unsigned char bytethap)
      {
      TMOD=0x55;// dem su kien
      P1_2=1;
      P1_3=0;
      P1_0=0;
      P1_1=1;
      delay(500);
      P1_0=1;
      kt0=0;// bit nay set de chuong trinh ngat kiem tra lua chon
      TH0=bytecao;
      TL0=bytethap; // 230 XUNG
      TR0=1;
      while(1)
        {
        if (P1_3==1) break;


        }
      }

void ngat1(void) interrupt 3      // ngat bo dinh thoi 1
  {
    if (kt1==0) // dung de di lap ma
     {
       P1_0=1;
       P1_1=1;
       P1_2=1;
       P1_3=1;
     }
    if (kt1==1) // bam xung dinh thoi kieu moi
     {
       TR1=0;
       if (dem_phai==tocdo_phai)
       { P1_0=~P1_0;tocdo_phai=10-tocdo_phai;dem_phai=0;}
       dem_phai++;
       TR1=1;
```

```
    }
    }
        // quay dung bo lap ma

void quaytrai(unsigned char bytecao,unsigned char bytethap)
  {
    TMOD=0x55;   //khoi dong bo dinh thoi dem su kien
    kt1=0;
    P1_0=1;
    P1_1=0;
    P1_2=0;
    P1_3=1;
    delay(500);
    P1_2=1;
    TH1=bytecao;
    TL1=bytethap;
    TR1=1;
    while(1)
        {
         if (P1_1==1) break;
        }
     }



void quay(unsigned char status ,unsigned char bytecao ,unsigned char
bytethap)
    {

     if (status==turn_right)
       quayphai(bytecao,bytethap);
     if (status==turn_left)
       quaytrai(bytecao,bytethap);
    }

void delay (unsigned long time)

    {
     unsigned long i;
     for (i=0;i<time ;i++)
```

```
      {}
    }

                // bam xung bang bo dinh thoi


void quayphaingatu()
  {
    motor(backward);
    delay(3500);
    motor(stop);
    delay(10000);
    motor(left_go);
    delay(4500);
    bamxung_quay(5,motor_left);
    motor(stop);
  }

void quaytraingatu()
  {
    motor(backward);
    delay(3500);
    motor(stop);
    delay(10000);
    motor(right_go);
    delay(4500);
    bamxung_quay(5,motor_right);
    motor(stop);
  }
```

# BÀI 4.MẠCH CÔNG SUẤT ĐIỀU KHIỂN ĐỘNG CƠ

**4.1. Động cơ.**

Động cơ được sử dụng trong robocon là loại động cơ 1 chiều (DC motor).

Hình 4.1.Động cơ 1 chiều


Việc lựa chọn động cơ căn cứ vào các tiêu chí sau:

-Tốc độ

-Khả năng chịu tải

-Độ hãm

-Dòng ,áp

-Đối với động cơ dùng cho cơ cấu chuyển động (phần đế của robot) yêu cầu đặt ra là phải có tốc độ nhanh ,và có độ hãm tốt .Động cơ thường được sử dụng ở phần này là loại động cơ pitman.

Ngoài động cơ pitman ,bạn có thể sử dụng các loại động cơ khác ,miễn là đạt được các tiêu chí nêu trên.Bạn có thể sử dụng động cơ vuông tháo bánh răng .Đây là các mà BK-FIRE đã sử dụng trong cuộc thi robocon2005 ,bằng cách này ,robot của BK-FIRE đạt được tốc độ khá cao.


Hình 4.3.Động cơ pitman.

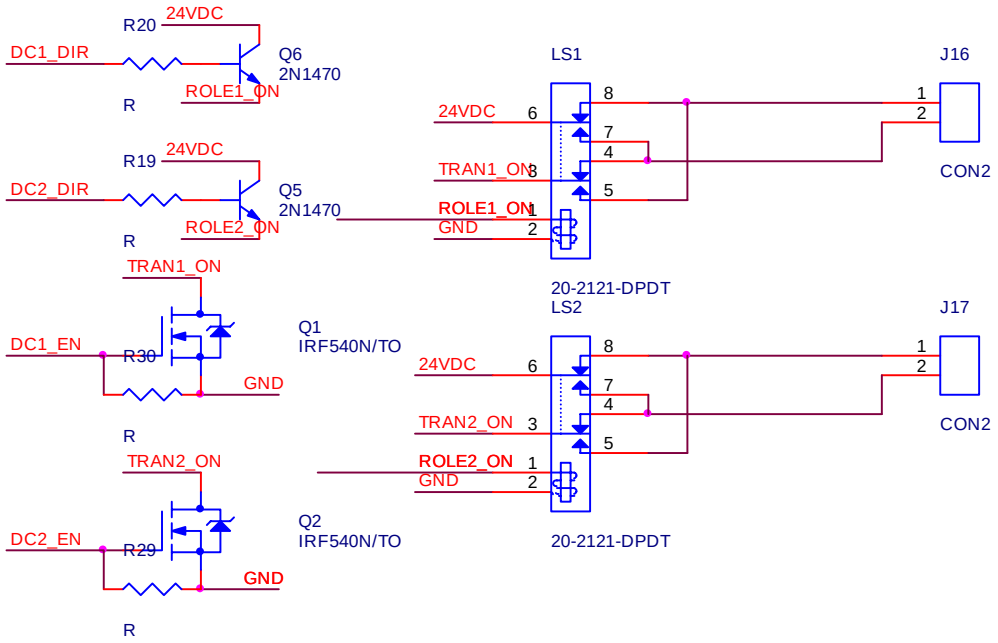
-Đối với động cơ dùng cho cơ cấu nâng hạ ,gắp quả (phần trên ) yêu cầu phải có khả năng chịu tải ,khoẻ .Loại động cơ thường được sử dụng là loại động cơ gạt nước hoặc động cơ có hộp điều tốc .Đặc điểm của loại động cơ này là tốc độ không cao nhưng lực quay rất khoẻ có thể chịu tải lớn.Trong năm 2005 ,các đội robocon BK-Đà Nẵng đã sử dụng động cơ này cho robot trung tâm và đạt hiệu quả rất cao.

-Trong một số trường hợp ,robot cần phải có tốc độ cực nhanh .VD năm 2005 và 2006 ,các robot trung tâm phải có tốc độ đặc biệt nhanh (trong vòng 1,5s-2s phải chiếm lĩnh được đuốc trung tâm).Để giải quyết vấn đề này,một số đội đã sử dụng động cơ đề xe máy cho robot trung tâm.Động cơ đề xe máy có dòng lớn ,và tốc độ rất nhanh.Tuy nhiên với loại động cơ này thì các mạch công suất không thể điều khiển được.Để gia tăng tốc độ ,bạn có thể lắp đặt cơ cấu xích cho robot của mình.

**4.2.Mạch điều khiển động cơ**

4.2.1.Mạch relay

4.2.1.1.Sơ đồ nguyên lý



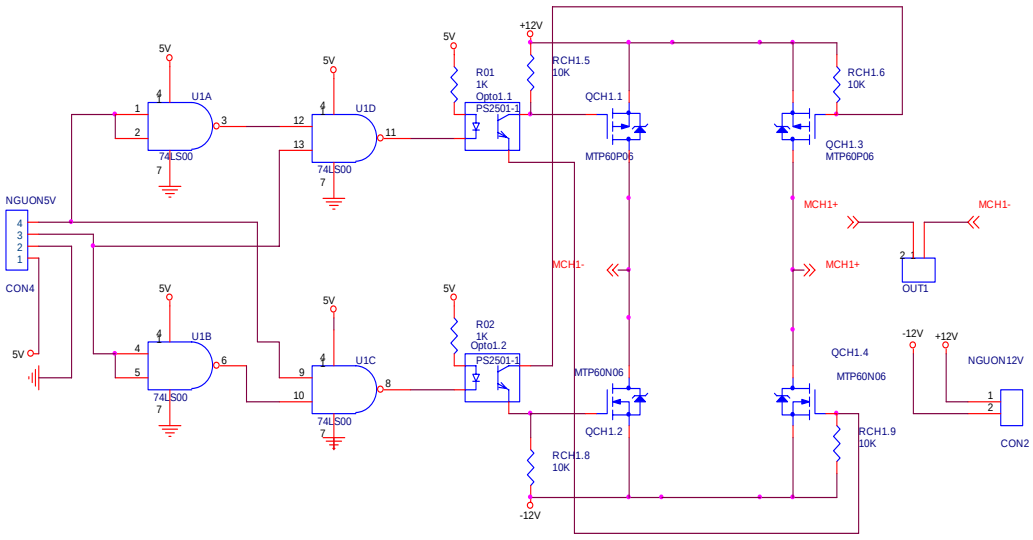

4.2.1.2.Nguyên lý hoạt động.

Loại relay được sử dụng trong mạch robot tự động là loại relay 8 chân (2 tiếp điểm )

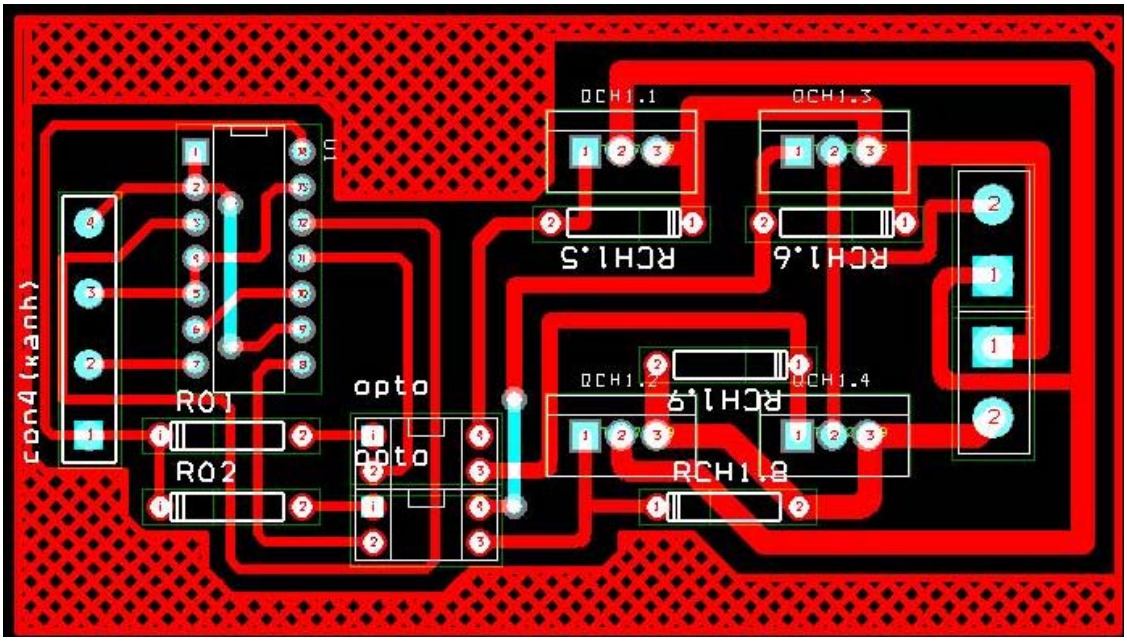
Mạch relay điều khiển động cơ sử dụng 2 tín hiệu điều khiển .Chân DC_EN và chân DC_DIR ,chân DC_EN là chân kích FET (đóng mở FET) chân DC_DIR là chân đảo chiều .Tuy nhiên ,trước khi đảo chiều động cơ bạn phải tắt FET trước ,nếu không sẽ gây chết FET.

## 4.3.Mạch cầu H điều khiển động cơ

4.3.1.Sơ đồ nguyên lý




4.3.2.Sơ đồ mạch in




4.3.3.Nguyên lý hoạt động

Mạch cầu H điều khiển động cơ sử dụng 4 MOSFET IRF540 để khuếch đại công suất và đóng mở.

Hình 4.4.IRF 540

Mosfet là loại transitor hiệu ứng trường (MOSFET = metal-oxide semiconductor field-effect transistor) MOSFET là mô hình nguồn dòng phụ thuộc áp.Như vậy để kích dẫn MOSFET cần đưa điện áp vào hai cực Vgs. Một điểm khác của MOSFET nữa là MOSFET có Vds không ổn định như Vce (BJT) mà nó phụ thuộc vào giá trị ổn định khác là Rds-on (Ở chế độ đóng ngắt). Như vậy Vds = Id*Rds. Như vậy công suất tiêu tán trên MOSFET chủ yếu phụ thuộc vào Rds.

Có hai loại MOSFET kênh P và kênh N.

Loại kênh N thường có Rds nhỏ hơn kênh P.

Loại kênh N cần Vgs > 0 để dẫn.

Loại kênh P cần Vgs < 0 để dẫn.

Đa số các MOSFET bắt đầu dẫn khi abs(Vgs) xấp xỉ 5. Thường giá trị khuyến cáo là 10V. Khi điện áp lớn hơn 15 bắt đầu có nguy cơ đánh thủng Vgs. Khá nguy hiểm là khi bị thủng MOSFET thường dẫn luôn dẫn đến dễ nổ. Trong mô hình của MOSFET có một cái tụ ký sinh từ gọi là Cgs cần quan tâm đến tụ này khi thiết kế mạch lái sử dụng PWM.Khi làm Robot thường dùng con IRF540 để lái động cơ DC hoặc dùng mạch cầu IRF540 để điều khiển hướng (Mặc dù cách điều khiển này rất khó).

Mạch cầu H điều khiển động cơ có 2 tín hiệu điều khiển .2 chân tín hiệu nay được nối với 2 chân vi điều khiển.

Giả sử 2 tín hiệu điều khiển là A,B (tương ứng 2 chân P1.0 ,P1.1)

Điều khiển động cơ DC có các trạng thái điều khiển :Động cơ quay ,đảo chiều ,dừng và điều khiển tốc độ động cơ.

Ta có bảng trạng thái

| P1.0 | P1.1 | Trạng thái |
|---|---|---|
| 1 | 0 | Động cơ quay xuôi |
| 0 | 1 | Động cơ quay ngược |
| 1 | 1 | Dừng động cơ |

| | | |
|---|---|---|
| 1 | 1 | Dừng động cơ |

4.3.4.Mã nguồn điều khiển động cơ

```
#include <at89x52.h>
#include <robot.h>
unsigned char dutycycle=0 ;
        /*
          P2_0  P2_1 : dieu khien dong co nang ha
          P2_2  P2_3 : dieu khien dong co lay bong
        */
void motor( unsigned char status)
{
  switch (status)
   {

    case nang:

     {
       P2_0=1;
       P2_1=0;
       break;
     }

    case ha:
     {

       P2_0=0;
       P2_1=1;
       break;

      }
    case dung:
     {

      P2_0=1;
       P2_1=1;
       break;
      }
    }
}
```

```
void motorkep(unsigned char state)
 {

  switch (state)
   {

    case kep:
     {
       P2_2=0;
       P2_3=1;
       break;
     }

    case nha:
     {
      P2_2=1;
      P2_3=0;
      break;
      }

     case phanh :
      {

       P2_2=1;
       P2_3=1;
       break;
      }

  }
}
void delay ( unsigned char time)
 {

  unsigned char i;
   for ( i=0; i<time ;i++)
    { }
 }

void nangnhanhdan(unsigned char time1,unsigned time2)
 {
```

```
  unsigned char  x,y;

     for (x=0;x <time1 ;x++)
      {
        for ( y=0;y<time2; y++)
        {
          P2_0=1;
          P2_1=0;
          delay(dutycycle);
          P2_1=1;
          delay(Time-dutycycle);
         }
        dutycycle++;

       if (dutycycle==255) break;;

      }

}

void nangchamdan(unsigned char Time1,unsigned Time2)
{

  unsigned char  x,y;

     for (x=0;x <Time1 ;x++)
      {
        for ( y=0;y<Time2; y++)
        {
          P2_0=1;
          P2_1=0;
          delay(dutycycle);
          P2_1=1;
          delay(Time-dutycycle);
         }
        dutycycle--;
        if (dutycycle==0) break;

      }
```

```
 }

void  nhabong(unsigned char timedelay)
 {
  motorkep(nha);
  delay(timedelay);
  motorkep(phanh);
 }

void main()
 {
  // trang thai gap bong

  motorkep(kep);
  delay(255);
  motorkep(phanh);

  // co cau nang bong hoat dong

  nangnhanhdan(100,100);    // chang 1 nang nhanh dan

   // chang 2  : nang het toc luc
  motor(nang);

  delay(100);

   // chang 3 : nang cham dan

  nangchamdan(100,100);

  motor(dung);

  delay(100);
  motorkep(phanh);
  /*  trang thai nha bong
     khi P0 cua slaver nhan tin hieu dieu khien tu master
     thi trang thai nha bong hoat dong
     */
```

```
  nhabong(100);


  while(1)
   {
    motor(dung);
    motorkep(phanh);
    }
//  P0=0xFB;
  }
            /* bo qua thu 1  */
```

4.3.5.Kĩ thuật điều chế độ rộng xung điều khiển tốc độ động cơ (PWM)

Điều khiển tốc độ động cơ là một kĩ thuật hết sức quan trọng trong việc lập trình robot .Kĩ thuật này giúp bạn có thể điều khiển robot của mình chạy nhanh chậm tuỳ ý ,giúp cho hoạt động của robot luôn linh hoạt ,thích ứng với mọi tính huống xảy ra trên sân.

a) Cách tạo xung có độ rộng thay đổi bằng VĐK.

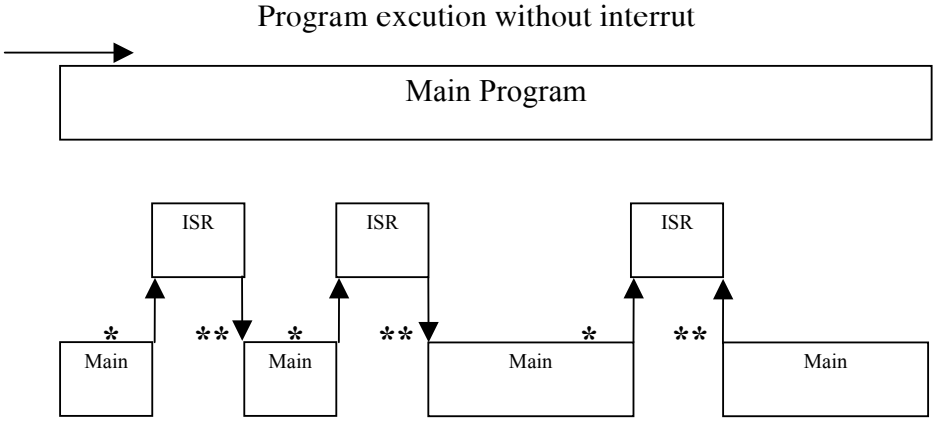
Cách 1: Như các bạn điều khiển nhấp nháy 1 con led, đó là tạo ra 1 xung ở 1 chân của vi điều khiển, nhưng xung đó có độ rộng cố định, tần số lớn, cách bạn có thể điều chỉnh lại hàm delay để tần số của nó đúng 1 Khz. Tuy nhiên vì là dùng hàm delay nên trong thời gian có xung lên 1(5V) và thời gian không có xung(0V) vi điều khiển không làm gì cả, hơn nữa tạo xung bằng việc delay mà các bạn có nhu cầu cần 2 bộ phát xung ở 2 kênh, có cùng tần số mà khác độ rộng xung thì trở nên rất khó khăn. Cho nên chúng ta dùng bộ định thời Timer của vi điều khiển trong trường hợp này rất tiện.

+ Cách 2: Dùng ngắt Timer của bộ vi điều khiển.

Trước hết nhắc lại về ngắt của vi điều khiển:

+ Ngắt là gì ? để trả lời câu hỏi này tôi xin trích đoạn về ngắt trong bài 2 ví dụ cho ngắt timer:

**Timer**




Program excution with interrupt.

Một chương trình chính không có ngắt thì chạy liên tục, còn chương trình có ngắt thì cứ khi nào điều kiện ngắt được đảm bào thì con trỏ sẽ nhảy sang hàm ngắt thực hiện xong hàm ngắt lại quay về đúng chỗ cũ thực hiện tiếp chương trình chính. Tôi có 1 ví dụ như sau: Bạn đang ăn cơm , có tiếng điện thoại , bạn đạt bát cơm ra nghe điện thoại , nghe xong lại quay về bưng bát cơm lên ăn tiếp. Thì quá trình ăn cơm của bạn là chương trình chính,có điện thoại gọi đến là điều kiện ngắt, bạn ra nghe điện thoại là thực hiện chương trình ngắt(Interrupt Service Rountine),quay về ăn cơm tiếp là tiếp tục thực hiện chương trình chính.

Ngắt đối với người mới học vi điều khiển là rất khó hiểu, vì đa số các tài liệu đều không giải thích ngắt để làm gì. Có nhiều loại ngắt khác nhau nhưng tất cả đều có chung 1 đặc điểm, ngắt dùng cho mục đích đa nhiệm. Đa tức là nhiều, nhiệm tức là nhiệm vụ. Thực hiện nhiều nhiệm vụ .

Các bạn nhìn vào tiền trình của hàm main với chương trình có ngắt :

Chương trình chính đang chạy, ngắt xảy ra, thực hiện hàm ngắt rồi quay lại chương trình chính. Chương trình trong vi điều khiển khác với ví dụ ăn cơm nghe điện thoại của tôi ở chỗ, thời gian thực hiện hàm chính là rất lớn,thời gian thực hiện hàm ngắt là rất nhỏ, cho nên thời gian thực hiện hàm ngắt không ảnh hưởng gì đến thời gian thực hiện hàm chính Như vậy trong hàm ngắt các bạn làm 1 việc, trong hàm chính của bạn làm 1 việc
như vậy coi như các bạn làm được 2 việc(đa nhiệm) trong 1 quãng thời gian tương đối ngắn cỡ mS, chứ thực ra tại 1 thời điểm vi điều khiển chỉ thực thi 1 lệnh.

Ví dụ : Bạn thử nghĩ xem làm thế nào để vừa điều chế xung PWM để điều chỉnh tốc độ động cơ , vừa đọc các cảm biến đầu vào mà tốc độ động cơ phụ thuộc đầu vào cảm biến.

Vậy ngắt là 1 điều kiện nào đó xảy ra ngẫu nhiên mà vi điều khiển có thể biết do phần cứng của vi điều khiển, rồi ta căn cứ vào đó để lập trỡnh.

* Ví dụ: Với ngắt bộ định thời timer, hay bộ đếm counter là khi tràn bộ đếm thì phần cứng của vi điều khiển sẽ bảo có ngắt xẩy ra và nhảy đến chương trình phục vụ ngắt( ISR_ Interrupt Sevice Rountine) 1 cách tự động.

Với ngắt ngoài, chân P3.2 chẳng hạn, nếu ta khai báo trước chân sử dụng chân P3.2 sử dụng cho ngắt ngoài chứ không phải sử dụng cho mục

đích IO thiừ cứ khi cú 1 xung xuất hiện từ mạch ngoại vi truyền vào chõn P3.2 thì phần cứng của vi điều khiển nhận ra và chuyển tới chương trình phục vụ ngắt.

Với ngắt nối tiếp thì cứ khi có kí tự truyền từ máy tính xuống vi điều khiển thì sẽ có hiện tượng ngắt xảy ra.

- Hàm ngắt:

Cấu trúc:

```
Void  Tênhàm(void) interrupt nguồnngắt using băngthanhghi
{
// Chuong  trinh phuc vu ngat o day
}
```

Chỳ ý về hàm ngắt:

+ Hàm ngắt không được phép trả lại giá trị hay truyền biến vào hàm.
+ Tên hàm bất kì.
+ interrupt là từ khóa phân biệt hàm ngắt với hàm thường.
+ Nguồn ngắt từ 0 tới 5 theo bảng vector ngắt.
+ Băng thanh ghi trên ram chọn từ 0 đến 3.

Tựy theo bạn viết hàm ngắt cho nguồn nào bạn chọn nguồn ngắt từ bảng sau:

| Ngắt do | Cờ | Địa chỉ vector |
|---|---|---|
| **Reset hệ thống** | **RST** | **0000H** |
| **Ngắt ngoài 0** | **IE0** | **0003H** |
| **Bộ định thời 0** | **TF0** | **000BH** |
| **Ngắt ngoài 1** | **IE1** | **0013H** |
| **Bộ định thời 1** | **TF1** | **001BH** |
| **Port nối tiếp** | **RI hoặc TI** | **0023H** |
| **Bộ định thời 2** | **TF2 hoặc EXF2** | **002BH** |

Riêng ngắt Reset không tính, bắt đầu đếm từ 0 và từ ngắt ngoài 0. Ví dụ: tôi cần viết hàm ngắt cho bộ định thời timer 1 hàm ngắt sẽ là.

```
void timer1_isr(void) interrupt 3 using 0
{
// Lenh can thuc hien.
}
```

- Về using 0: Có 4 băng thanh ghi bạn có thể chọn cho chương trình phục vụ ngắt, cái này cũng không quan trọng. Trong hàm ngắt các bạn có thể bỏ đi từ using 0, khi đó vi điều khiển sẽ tự sắp xếp là dùng băng thanh ghi nào.

- Hàm ngắt khác hàm bình thường chỗ nào. Hàm bình thường ví dụ hàm delay, cứ khi bạn gọi nó thì nó sẽ được thực hiện, có nghĩa là nó có vị trí cố định trong tiến trình hàm main, có nghĩa là bạn biết nó xảy ra khi nào. Còn hàm ngắt thì không có tiến trình cố định, điều kiện ngắt có thể xảy ra bất kì lúc nào trong tiến trình hàm main và cứ khi nào có điều kiện ngắt thì hàm ngắt sẽ được gọi tự động.
- Để sử dụng ngắt ta phải làm các công việc sau:

1) Khởi tạo ngắt: dùng ngắt nào thì cho phép ngắt đó hoạt động bằng cách gán giá trị tương ứng cho thanh ghi cho phép ngắt IE( Interrupt Enable):

| EA | ET2 | ES | ET1 | EX1 | EX0 | ET0 |
|---|---|---|---|---|---|---|

| IE | | Điều khiển các nguồn ngắt (0: không cho phép; 1: cho phép) |
|---|---|---|
| IE.7 | EA | Cho phép/ không cho phép toàn cục |
| IE.6 | --- | Không sử dụng |
| IE.5 | ET2 | Cho phép ngắt do bộ định thời 2 |
| IE.4 | ES | Cho phép ngắt do port nối tiếp |
| IE.3 | ET1 | Cho phép ngắt cho bộ định thời 1 |
| IE.2 | EX1 | Cho phép ngắt từ bên ngoài (ngắt ngoài 1) |
| IE.1 | EX0 | Cho phép ngắt từ bên ngoài (ngắt ngoài 0) |
| IE.0 | ET0 | Cho phép ngắt do bộ định thời 0 |

IE là thanh ghi có thể xử lí từng bít. Ví dụ : bạn muốn cho phép ngắt timer 1 bạn dùng lệnh: ET1=1; Không cho phép nữa bạn dùng lệnh : ET1=0; Hoặc bạn có thể dùng lệnh IE= 0x08; thì bit 3 của thanh ghi IE tức(IE) sẽ lên 1. Nhưng cách thứ nhất tiện hơn.

2) Cấu hình cho ngắt: Trong 1 ngắt nó lại có nhiều chế độ ví dụ: với ngắt timer. Bạn phải cấu hình cho nó chạy ở chế độ nào, chế độ timer hay counter, chế độ 16 bit, hay 8 bit,… bằng cách gán các giá trị tương ứng cho thanh ghi TMOD( Timer MODe).

| **TMOD** | | **Chọn model cho bộ định thời 1** | | | |
|---|---|---|---|---|---|
| **7** | **GATE** | **Chọn model cho bộ định thời 1** | | | |
| **6** | **C/T** | **Bít chọn chức năng đếm hoặc định thời:** | | | |
| **5** | **M1** | **Bit chọn chế độ thứ nhất** | | | |
| **4** | **M0** | **Bit chọn chế độ thứ 2** | | | |
| | | **M1** | **M0** | **Chế độ** | **Chức năng** |
| | | **0** | **0** | **0** | **Chế độ định thời 13 bit** |
| | | **0** | **1** | **1** | **Chế độ định thời 16 bit** |

| | | | | | |
|---|---|---|---|---|---|
| | | **1** | **0** | **2** | **Chế độ tự động nạp lại 8 bit** |
| | | **1** | **1** | **3** | **Chế độ định thời chia xẻ** |
| **3** | **GATE** | **Bit điều khiển cổng cho bộ định thời 0** | | | |
| **2** | **C/T** | **Bit chọn chức năng đếm / định thời cho bộ định thời 0** | | | |
| **1** | **M1** | **Bit chọn chế độ thứ nhất cho bộ định thời 0** | | | |
| **0** | **M0** | **Bit chọn chế độ thứ 2 cho bộ định thời 0** | | | |

Ví dụ tôi cấu hình cho bộ định thời 1 chế độ timer,với bộ đếm 8 bit tự động nạp lại(auto reload) dùng lệnh sau: TMOD=0x20.
Các bạn đừng lo vì việc phải nhớ bảng thanh ghi này, các bạn không phải nhớ nói trắng ra như vậy, chuyển sang phần lập trình các bạn sẽ được hướng dẫn làm thế nào để không phải nhớ, nhưng chỉ lập trình với C mới làm được còn lập trình Asem thì bắt buộc phải nhớ .

3)Bắt đầu chương trình có ngắt:

-Trước khi bắt đầu cho chạy chương trình ta phải cho phép ngắt toàn cục được xảy ra bằng cách gán EA(Enable All interrupt) bằng 1, thì ngắt mới xảy ra.

-Thường thì ngay vào đầu chương trình(hàm main) trước vòng while(1) chúng ta đặt công việc khởi tạo, cấu hình và cho phép kiểm tra ngắt. Ví dụ với bộ định thời timer ta gán các giá trị phù hợp cho thanh ghi TCON( Timer CONTROl).

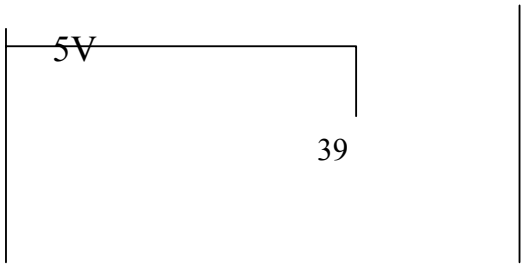
| TCON | | Điều khiển bộ đinh thời |
|---|---|---|
| TCON.7 | TF1 | Cờ tràn của bộ định thời 1. Cờ này được set bởi phần cứng khi có tràn, được xoá bởi phần mềm, hoặc bởi phần cứng khi bộ vi xử lý trỏ đến trình phục vụ ngắt |
| TCON.6 | TR1 | Bit điều khiển hoạt động của bộ định thời 1. Bit này được set hoặc xoá bởi phần mềm để điều khiển bộ định thời hoạt động hay ngưng |
| TCON.5 | TF0 | Cừ tràn của bộ định thời 0 |
| TCON.4 | TR0 | Bit điều khiển hoạt động của bộ định thời 0 |
| TCON.3 | IE1 | Cừ ngắt bên ngoài 1 (kích khởi cạnh). Cờ này được set bởi phần cứng khi có cạnh âm (cuống) xuất hiện trên chân INT1, được xoá |

| | | |
|---|---|---|
| | | bởi phần mềm, hoặc phần cứng khi CPU trỏ đến trình phục vụ ngắt |
| TCON.2 | IT1 | Cờ ngắt bên ngoài 1 (kích khởi cạnh hoặc mức). Cờ này được set hoặc xoá bởi phần mềm khi xảy ra cạnh âm hoặc mức thấp tại chân ngắt ngoài |
| TCON.1 | IE0 | Cờ ngắt bên ngoài 0 (kích khởi cạnh) |
| TCON.0 | IT0 | Cờ ngắt bên ngoài 0 ( kích khởi cạnh hoặc mức) |

Ví dụ để chạy bộ định thời timer 1 ta dùng câu lệnh: TR1=0;
TR1(Timer Run 1). Còn bạn nào thích khó thì:TCON=0xxx;
Còn các loại ngắt khác quá trình tương tự, đây là khóa học cơ bản chỉ làm việc với ngắt timer, trong khóa nâng cao sẽ có các ngắt còn lại, tuy nhiên làm việc được với ngắt timer thì các ngắt khác các bạn cũng có thể làm tương tự, các bạn làm đến ngắt nào thì dùng tài liệu tra bảng thanh ghi của ngắt đó. Tài liệu tôi sẽ gửi cùng bài này.
- Quay trở lại bài học:

Sau khi khởi tạo song và cho ngắt timer 1 chạy thì điều gì xảy ra?
Khi bắt đầu cho timer 1 chạy thì bộ đếm của timer sẽ đếm dao động của thạch anh, cứ 12 dao động của thạch anh(1 chu kì máy), bộ đếm của timer 1 TL1(Timer Low1) sẽ tăng 1,có thển nói timer 1 đếm số chu kì máy. Đối với chế độ 8 bít.
TL1 là 1 thanh ghi 8 bít, là bộ đếm của bộ định thời rõ rồi. Nó đếm được từ 0, đến 255. Nếu nó đếm đến 256 thì bộ đếm tràn, TL1 quay vòng lại bằng 0, và cờ ngắt TF1(Timer Flag 1) tự động được gán lên 1(bằng phần cứng của vi điều khiển) như 1 công tắc tự động bật, và ngắt xảy ra.

Còn với chế độ 16 bít, bộ đếm của bộ định thời còn 1 thanh ghi 8 bít nữa là TH1(Timer high 1), nếu cấu hình cho timer 1 hoạt động ở chế độ 16 bit thì khi TL1 tràn nó sẽ đếm sang TH1(TH1 sẽ tăng 1). Như vậy ta có thể đếm: $2^{16}$ chu kì máy( 2 thanh ghi 8+8=16 bít).
Chú ý là khi bộ đếm tràn ngắt sẽ xảy ra. Nếu ta cần đếm 256 chu kì máy thì khi khởi tạo ta cho TL1=0; , còn nếu không muốn đếm 256 chu kì mày mà ta chỉ cần đếm 100 thôi ngắt đã xảy ra rồi thì ta fải làm như sau: 256-100 = 156; và khi khởi tạo ta gán : TL1=155; vì đếm từ 155 đến 255 là đủ 100 lần thì ngắt xảy ra.
Với yêu cầu của bài. Tạo xung tần số 1Khz → Chu kì = $1/10^3$ = 0,001 giây= 1 mili giây=1000 uS= 1000 chu kì máy. Với 10 cấp tốc độ, tức là bạn phải tạo ra được xung 10%, 20%, 30%, 40%, …, 90%, 100%. 1 xung như sau:



0V

T : Chu kì
1000 miro giây.

Khoảng thời gian xung kéo dài 5V là T1. Xung 10% tức là T1/ T= 10%=1/10. Xung 20% T2/T=2/10…PWM(Thay đổi độ rộng xung)
Bây giờ tôi mới xin nói về phần 2.

4.3.5.Nguyên lí của PWM

- Xung PWM: Đưa ra mở transitor, xung với độ rộng lớn hơn transitor sẽ mở lâu hơn động cơ sẽ quay nhanh hơn,dĩ nhiên không tuyến tính. Không có xung động cơ sẽ không quay, có xung 100% động cơ sẽ quay max.Tuy nhiên xung phải lớn hơn 1 mức nào đó thì mới đủ khởi động cho động cơ. Các đặc tính này các bạn tham khảo trong giáo trình về máy điện, khí cụ điện, nếu các bạn cần thông số chính xác.

Để có thể thay đổi 10 cấp tốc độ với chu kì 1000uS, ta khởi tạo cho ngắt timer: 100 uS ngắt 1 lần. Trong hàm ngắt kiểm tra xem ta cần cấp xung bao nhiêu % thì ta sẽ gán giá trị cho nó. Cụ thể như sau:

* Hàm khởi tạo ngắt.

Dùng ngắt timer 0, 100 uS ngắt 1 lần, dùng chế độ 2 8 bit tự động nạp lại của timer (vì mình chỉ cần đếm đến 100).TL0 nạp bằng 156. Đối với chế độ 2 khi tràn bộ đếm TL0 sẽ quay vòng giá trị bằng 0, nhưng sau đó nó lại được nạp giá trị lưu trong TH0(giá trị nạp lại), do đó ta chỉ cần gán giá trị choTL0 và TH0 trong hàm khởi tạo, còn ở các chế độ khác 16 bit, 2 timer counter 8 bit, khi tràn bộ đếm TL0 không được nạp lại mà ta phải tự gán lại giá trị cho nó trong hàm ngắt.

```
void khoitaotimer0(void)// Ham khoi tao
{
EA=0;// Cam ngat toan cuc
TMOD=0x02;// Timer 0 che do 2 8 bit auto reload
TH0=0x9B;// Gia tri nap lai 155 doi ra so hex
TL0=0x9B;// Gia tri khoi tao 155 doi ra so hex
ET0=1;// Cho phep ngat timer 0
EA=1;// Cho phep ngat toan cuc
TR0=1;// Chay timer 0 bat dau dem so chu ki may
}
```

# BÀI 5.MẠCH VI ĐIỀU KHIỂN

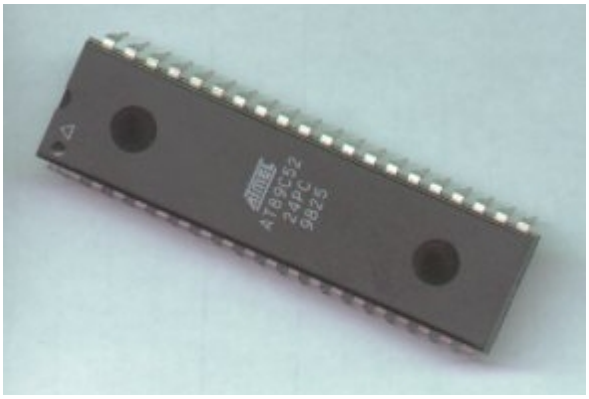
Mạch vi điều khiển đóng vai trò là phần trung tâm của robot .Vi điều khiển có thể ví như là bộ não của một cơ thể người ,điều khiển mọi hoạt động của robot.

## 5.1.Giới thiệu một số loại vi điều khiển được sử dụng trong robocon

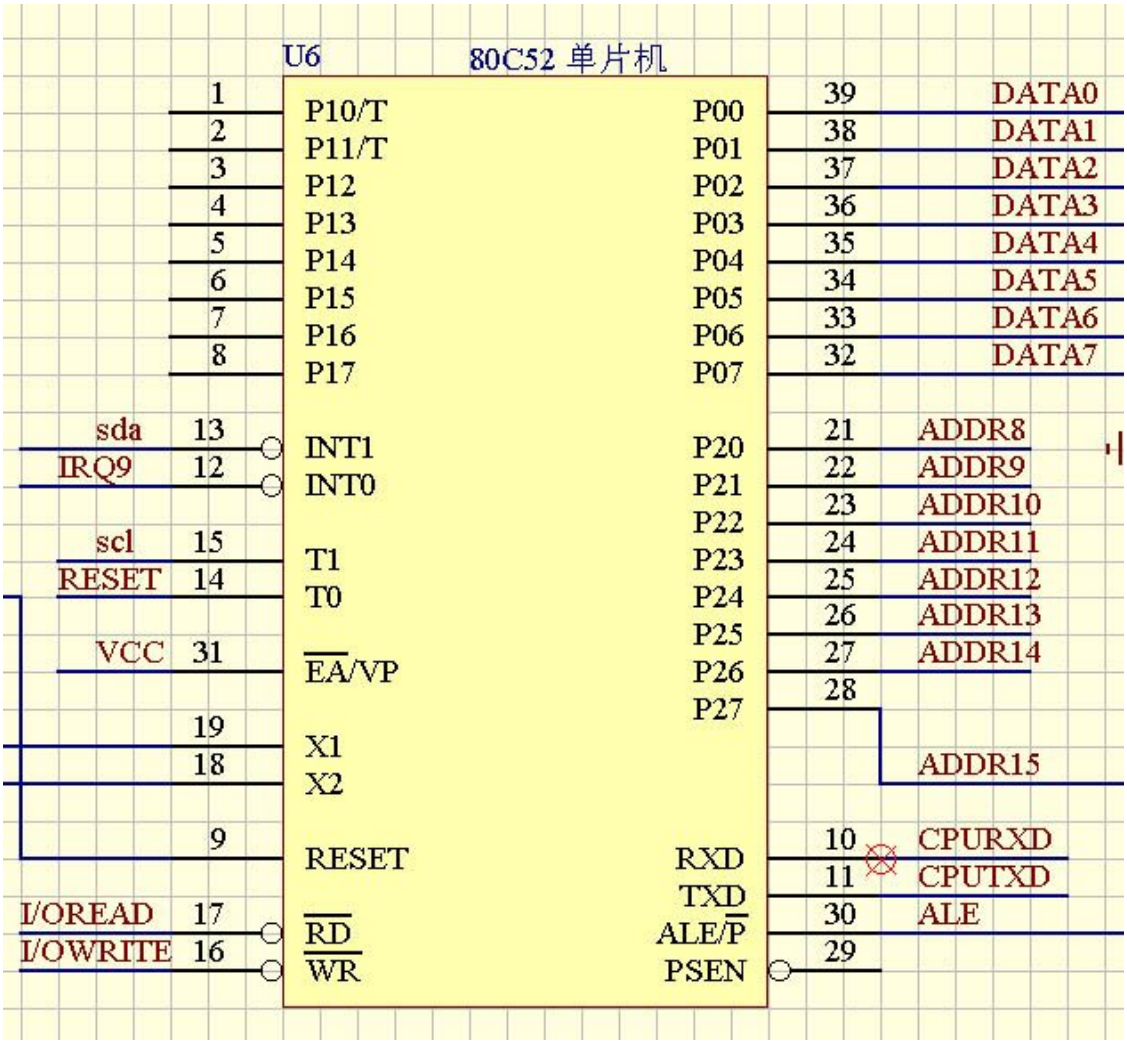
### 5.1.1.Vi điều khiển 8051

Vi điều khiển 8051 là loại vi điều khiển được sử dụng nhiều nhất trong các cuộc thi robocon từ trước tới nay.Bởi vì nó có giá thành rẻ ,dễ mua và nhiều tài liệu tiếng Việt.Chính vì thế ,nhiều đội robocon đã sử dụng 8051 là bộ não cho robot của mình.

Hình 5.1.Vi điều khiển 8051




Hình 5.2.Sơ đồ chân



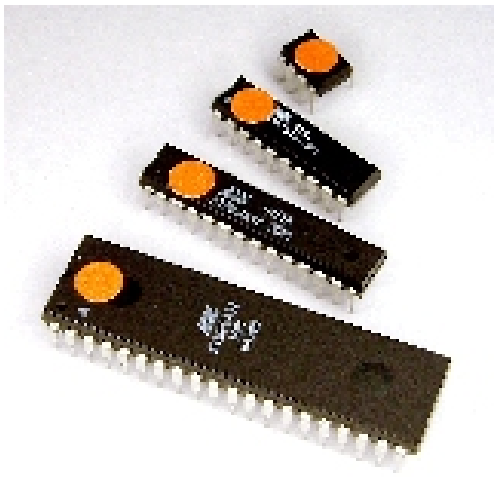

Trong các loại điều khiển họ 8051 ,vi điều khiển 89c52 và 89s52 được nhiều đội robocon sử dụng nhiều nhất.Đặc biệt là các đội đến từ Bách Khoa Hà Nội.

Những đặc điểm của vi điều khiển 8051 ,các bạn có thể tham khảo trong cuốn giáo trình 8051 của MTC.
Một loại vi điều khiển thuộc họ 8051 khá mạnh nữa do hãng Philip sản xuất là loại P89V51RD.Chip Philip được các đội miền nam ,đặc biệt là BK HCM sử dụng rất nhiều do được tài trợ miễn phí.
Nhìn chung chíp Philip có đầy đủ những tính năng của loại chíp 89c51 của atmel như timer/counter , ngắt .v ..v. Bên cạnh đó , chíp philip có những tính năng nổi bật hơn con 89c51 .
- Bộ nhớ ROM có dung lượng lớn
16/32/64 kb Flash rom và 1024 bytes Ram ( so với 8 k rom ,128 bytes ram của 89x) bộ nhớ của chíp PHILIP
-Chức năng ISP ( In system -programming)
ISP là khả năng nạp trực tiếp chương trình vào chip ngay trên mạch mà không cần phải rút chíp ra khỏi mạch .đối với vdk của atmel chỉ có con AVR ,và 89s52 mới có chức năng này.Nhờ chức năng này ,mạch nạp cho chip philip rất đơn giản dễ chế tạo
-Chức năng IAP (in-application programmable),chức năng IAP cho phép bộ nhớ flash có khả năng cấu hình lại trong khi các ứng dụng đang chạy.
-3 bộ định thời 16 bít chức năng của philip giống hệt chíp 89c51
-SPI (serial peripheral interface)
Chức năng này cho phép truyền dữ liệu đồng bộ với tốc độ cao giữa chíp philip và các thiết bị ngoại vi khác ,hay giữa các chíp với nhau.
- PWM (pulse width modulation)
Đây là một chức năng rất đặc biệt của philip ,khi chíp hoạt động ở chế độ này một chân của chíp sẽ ra phát xung với tần số khác nhau (tần số này do người lập trình xác định) . Philip có chế độ băm xung 8 bit (con AVR còn hỗ trợ băm xung 10 bít ADC) với 5 chân băm xung.
-Chức năng ngắt
Philip 8 ngắt với 4 mức ưu tiên ngắt ,nó cũng có ngắt ngoài ,ngắt timer , nối tiếp như 89c ,ngoài ra còn có chức năng ngắt PCA ,UART/SPI.Có thể nói chip philip là loại chíp rất ưu việt ,có thể thay thế rất tốt cho loại chip 89c52 thường được dùng trong các kì ROBOCON.các bạn có thể tham khảo

thêm datasheet của lọai chip này ở trang
http//www.alldatasheet.com (p89v51rd )
Trong năm 2004 ,2006 ,chip Philip được FXR và BKPRO sử dụng trong cuộc thi robocon và đã giành chức vô địch.
5.1.2.Vi điều khiển AVR

AVR là loại vi điều khiển do hãng ATMEL sản xuất .AVR có những tính năng rất mạnh so với 8051 như PWM ,ADC .AVR đã được BK-FIRE sử dụng đầu tiên vào năm 2006 và đã lọt được vào vòng chung kết toàn quốc.

Hinh 5.1.Vi điều khiển AVR



Việc sử dụng AVR trong robocon có thể nói là một cải tiến mới trong công nghệ chế tạo robot theo truyền thống của các đội đến từ BKHN.Thông tin về chip AVR các bạn có thể tham khảo giáo trình Vi điều khiển nâng cao của trung tâm MTC.

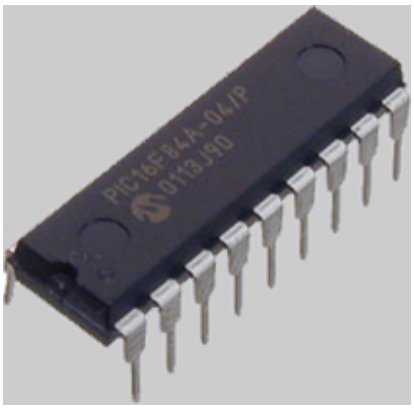
5.1.3.Vi điều khiển PIC.

Bên cạnh AVR ,PIC cũng là một sự lựa chọn khá tốt cho các đội robocon do những tính năng vượt trội của nó so với các loại vi điều khiển khác như khả năng chống nhiễu ,PWM ,ADC, ngắt.Hiện nay trên thtrường có khá nhiều dòng vi điều khiển PIC.VD:pic16f877,pic16f84 ,pic16f828..v.v.v

**Đánh giá các dòng PIC**

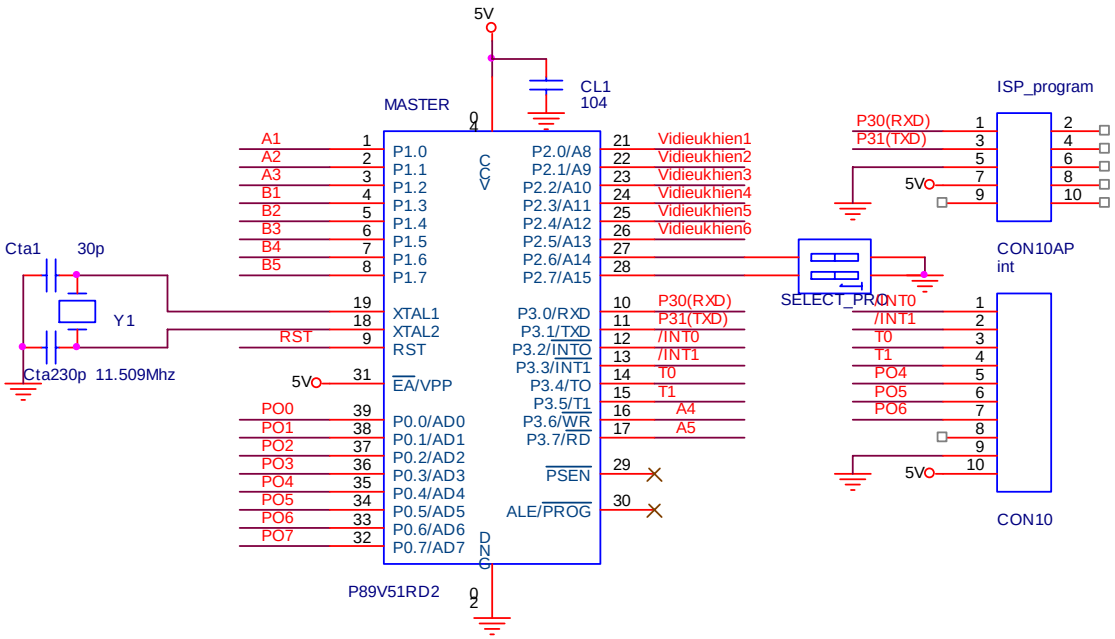
- Dòng PIC nhiều chân nhất là dòng PIC18Fxxxx, có những con số chân lên đến 80 chân
- Dòng PIC ít chân nhất là dòng PIC10Fxxx, chỉ có 6 chân
- Dòng PIC phổ biến nhất là dòng PIC16F877A (đủ mạnh về tính năng, 40 chân, bộ nhớ đủ cho hầu hết các ứng dụng thông thường)
- Dòng PIC mà chúng tôi đánh giá cao nhất là dòng PIC16F876A (28 chân, chức năng không khác gì so với PIC16F877A, nhưng nhỏ gọn hơn nhiều, và số chân cũng không quá ít như PIC16F88).
- Dòng PIC hỗ trợ giao tiếp USB là dòng PIC18F2550 và PIC18F4550
- Dòng PIC điều khiển động cơ mạnh nhất là dòng PIC18F4x31
- Khi cho rằng mình chuyên nghiệp hơn, các bạn nên dùng PIC18F458
- Dòng PIC tàng hình là dòng PIC17xxxxx, hiện nay đã không còn được sản xuất

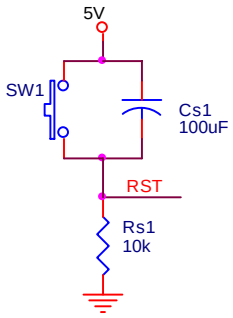
Hình 5.2.Vi điều khiển PIC



## 5.2.Sơ đồ nguyên lý

Trong giáo trình này ,chúng tôi chủ yếu tập trung vào mạch vi điều khiển dùng 89s52.Bởi vì vi điều khiển 89s52 được sử dụng khá thông dụng trong các kì robocon.Sơ đồ nguyên lý của vi điều khiển được nói nhiều trong giáo trình vi điều khiển cơ bản (8051) do MTC soạn thảo.Các bạn có thể tham khảo kĩ hơn trong tập giáo trình này.



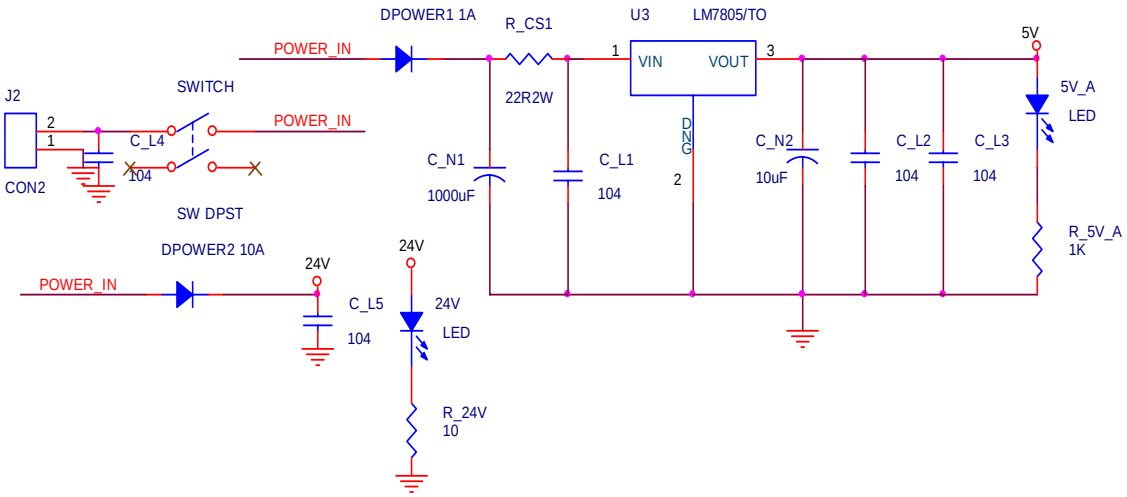



Hình 5.2.Sơ đồ nguyên lý mạch vi điều khiển.

Mạch vi điều khiển sử dụng nguồn 5 vôn.Trong robocon ,mạch này đóng vai trò là khối trung tâm của robot , kết nối với tất cả các modul khác.Từ mạch vi điều khiển này ,bạn có thể mở rộng ra các modul khác.

Mạch nguồn.

Mạch vi điều khiển sử dụng nguồn 5V ,Tuy nhiên các acquy dùng trong robot đều là nguồn 12V hoặc 24V .Do đó ,bạn không thể đấu trực tiếp nguồn ắc quy vào vi điều khiển được ,cần phải thiết kế một mạch nguồn để cung cấp một nguồn điện ổn định 5V cho vi điều khiển .




Sơ đồ nguyên lý mạch nguồn dùng trong robocon.

Mạch nguồn này sử dụng IC ổn áp 7805.Đây là loại IC ổn áp : Đầu vào > 7V đầu ra 5V 500mA. Mạch ổn áp: cần cho VĐK vì nếu nguồn cho VĐK không ổn định thì sẽ treo VĐK, không chạy đúng, hoặc reset liên tục, thậm

chí là chết chíp.Trong sơ đồ trên ,các tụ điện đóng vai trò lọc nhiễu ,ổn áp ,diode để chống ngược dòng.Ngoài ra ,trong mạch nguồn ,các bạn chú ý nên lắp thêm đèn báo nguồn và cầu chì để bảo vệ quá áp .

# BÀI 6:LẬP TRÌNH VÀ KĨ THUẬT DÒ ĐƯỜNG

Trong robocon ,lập trình là khâu cuối cùng trong 3 bộ phận cấu thành robot. Lập trình sẽ đóng vai trò quyết định cuối cùng để đánh giá robot của bạn sẽ hoạt động tốt hay không.Robot sẽ không thể chạy tốt nếu như chương trình xử lý không hoàn hảo.

6.1.Các ngôn ngữ lập trình sử dụng trong robocon.

Trong robocon có 2 ngôn ngữ lập trình được sử dụng nhiều nhất là C và ASM .2 ngôn ngữ này có những điểm mạnh và điểm yếu riêng.

Ngôn ngữ ASM có ưu điểm là gọn nhẹ ,giúp người lập trình hiểu sâu về cấu trúc phần cứng của vi điều khiển .Các chương trình viết bằng ASM thường chạy nhanh và tốc độ xử lý cao.ASM đã được BK-FIRE sử dụng trong cuộc thi 2005 và hoạt động khá hiệu qủa.Tuy nhiên ,ASM có nhược điểm là khó học và tập lệnh nghèo nàn ,không thuận tiện để lập trình các chương trình lớn.

Ngôn ngữ lập trình C có thế mạnh là dễ học ,tập lệnh phong phú ,và đặc điểm là ngôn ngữ lập trình có cấu trúc nên rất thuận lợi để xây dựng các chương trình lớn.Nhược điểm của C là không giúp người đọc hiểu sâu về cấu trúc phần cứng.

Nhìn chung , trong các cuộc thi robocon ,ngôn ngữ C được sử dụng nhiều nhất do những ưu điểm của nó.Giáo trình này cũng hướng dẫn bạn lập trình bằng ngôn ngữ C.

6.2.Mã nguồn của robocon

Xin giới thiệu một đoạn mã nguồn của đội CIRTECH-45 của BKHN năm 2004

```
/*
  Night Lamp Saver V5.0
  89C2051(ext.oscillator 680kHz) + MAC97A6 + no battery backup
  demonstration of using Micro-C and ATMEL89C2051 to build a device
  used for controlling night lamp that turn on and off night lamp
  with preset time on/off.
  After reset or power failure occured, high blink rate of led will show,
  user should press P3.0 to reset time to 18:00, low blink rate will show
  indicating normal operation.
```

The Saver V4.0 using Xtal 11.0592MHz produces EMI that interfere TV reception

This version the Xtal oscillator has changed to RC oscillator 680kHz.

cputick incremental was derived from 50Hz or 20ms main frequency.

*Copyright (c) 1999 W.SIRICHOTE*
*/

```
#include c:\mc51\8051io.h
#include c:\mc51\8051reg.h

/*-------- turn lamp on/off after reset time to 18:00 ---------*/

#define onHour1  18   /* 18:00 turn lamp on */
#define onMin1   00
#define offHour1 18   /* 18:01 turn off */
#define offMin1  01

/* every day turn on at 19:00 and and off at 22:00 */

#define onHour2  19
#define onMin2   00
#define offHour2 22
#define offMin2  00

/* set clock to 18:00 when press P3.0 */

#define setHour  18
#define setMin   00

/*-------------------------------------------------------------*/

extern register char cputick;
unsigned register char
sec25,sec50,sec,sec5,min,hour,flag1,temp,led,blink_rate;
/* above must be defined as register for tiny model */

/* variables description
cputick increments by one every 20ms
sec25   half second counter
```

```
sec50   2*25Hz counter
sec     current second
sec5    5 second counter
min     current min
hour    current hour
temp    temp register
led     counter for led on duration (times cputick)
blink_rate 0 = high blink rate, 10 low blink rate

flag1   intertask signaling                 mask byte

flag1.0 set every 1 second                  0x01
flag1.1 set every 1 min                     0x02
flag1.2 not use                             0x04
flag1.3 set every 0.5 second                0x08
flag1.4 set after P3.2 has been pressed     0x10
flag1.5 disable turn on/off 18:00-18:01 if set  0x20
flag1.6-7 not use
*/

main()
{
  cputick = 0;
  hour = 18;
  min = 0;
  sec = 0;
  sec25 = 0;
  sec50 = 0;
  flag1 = 0;
  blink_rate = 0;    /* indicate reset time to 18:00 is needed */

  asm "LAMP EQU $97"; /* P1.7 */
  asm{
        SETB $AF  /* setb EA */
        SETB $A8  /* enable external interrupt */
        SETB $88  /* negative edge triggering */
    }
  while(1)
  {
    while ( cputick < 1);
```

```
        cputick = 0;     /* 20ms has elapsed */

/*------------- the following tasks execute every 10ms ------*/

        time();
        comparetime();
        cpubeat();
        settime();
  /*      waithigh();    */
    }

/*-------------------------------------------------------------*/
}

time ()
/* update real-time clock, date */
{
    sec25++;
    if (sec25 >= 25) /* now 25 times means half second */
    {sec25 = 0;
      flag1 |= 0x08; /* set bit 3 every 0.5 s */
   sec50++;
   if (sec50 >= 2)   /* 2 * 25 * 20 ms = 1 s */
   {sec50 = 0;
    flag1 |= 0x01;   /* set bit 0 */
    sec++;
    if (sec >= 60)
    {sec = 0;
     flag1 |= 0x02; /* set bit 1 */
     min++;
     if (min >= 60)
     {min = 0;
     hour++;
     if (hour >= 24)
     {hour = 0;
     }

    }
   }
  }
```

```
}
}

comparetime()
{
   if ((flag1 & 0x10) != 0) /* enabled only after P3.2 has been pressed */
   {
      compareTimeOn_Off();
   }
}

compareTimeOn_Off()
{
  if ((flag1 & 0x01)!=0)
  {
   testOnOff();
   if(hour == onHour2 && min == onMin2)
         asm" CLR LAMP";
   if(hour == offHour2 && min == offMin2)
         asm" SETB LAMP";
   }
}

testOnOff()
{
   if ((flag1 & 0x20) == 0)
   {
      if(hour == onHour1 && min == onMin1)
         asm" CLR LAMP";
      if(hour == offHour1 && min == offMin1)
         {
          asm" SETB LAMP";
          flag1 |= 0x20; /* disable further test on off */
         }
   }
}



cpubeat()
```

```
{
   beat5sec();
   livecpu();

}

beat5sec()  /* clear P3.7 every blink rate */

{
   if ((flag1 & 0x08)!=0)
      {
       flag1 &= ~0x08;    /* clear bit 3 of flag1 */
       sec5++;
      if (sec5 > blink_rate)
       {sec5 = 0;
         flag1 |= 0x40; /* set bit 6 of flag1 to signal livecpu task */
        asm " clr P3.7"; /* make led on */
        led = 2;        /* load time on duration times cputick  */
       }
      }

}

livecpu()

{
   if ((flag1 & 0x40) != 0)
    {
      led--;
      if (led == 0)
        {
        asm " setb P3.7";
        flag1 &= ~0x40;
        }
   }
}

settime()
{
   if ((P3 & 0x01) == 0) /* reset time to 18:00 if P3.1 low */
```

```
   {
      hour = setHour;
      min = setMin;
      sec = 0;
      sec50 = 0;
      flag1 |= 0x10; /* enable compare time on/off */
      flag1 &= ~0x20; /* reenable testOnOff after pressing set clock to 18:00
*/
      blink_rate = 10;
   }
}

/*
waithigh()
{
      asm" jnb P3.2,*";
      pause(2);
      asm" jnb P3.2,*";
      pause(2);
}

pause(j)
int j;
{
   int i;
   for (i=0;i<j;i++)
   ;
}
*/
```

Mã nguồn ASM

```
;CHUONG TRINH CHO IC MASTER
;    P1: CAM BIEN NHAN VAO
;    P0: TRUYEN SANG SLAVER

STARTB EQU P3.7
CT_NANG EQU P3.2
CT_HATAY EQU P3.0
CT_QUA EQU P3.1
```

```
CB_L EQU P1.2
CB_R EQU P1.1
MKEP_DIR EQU P2.0
MKEP_EN  EQU P2.1
MNANG_DIR EQU P2.2
MNANG_EN  EQU P2.3

STOP_C EQU  11111111B
QPHAI_C EQU 00011111B
QTRAI_C EQU 00101111B
LUI_C EQU   00111111B
THANG_C EQU 01001111B
TINHCHINH_C EQU 01011111B
ORG 0000H
LJMP INIT
ORG 001BH ;DIEM NHAP VECTOR NGAT CUA T1
   LJMP NGATT1
ORG 0030H
INIT:;KHOI TAO CAC GIA TRI SAU RESET
   CLR TR0
   MOV SP,#5FH
   SETB STARTB          ;BIT KHOI DONG
   MOV P0,#0FFH
   MOV P1,#0FFH
   MOV R0,#0H     ;BIT Dem SO NGA TU
   MOV R1,#0H
   MOV P2,#0H    ;TAT CAC DONG CO
   MOV IE,#0H ;CHO PHEP NGAT BO DINH THOI 1
   MOV P3,#0FFH
   MOV TMOD,#00010001B
   CLR TR1
   CLR TR0
   CLR TF0
   CLR TF1



;=============================================
=============================================
==================================
```

```
MAIN:
   ;MOV P0,QPHAI_C
   JB STARTB,MAIN
    LAP_CHINH:
     MOV A,P1
     ANL A,#00001111B
     MOV P0,A
    LJMP LAP_CHINH
    EXIT:
    NODO:  SJMP NODO

;=========================================================
=========================================================
==================================
DEMNGATU:
    INC R0
    ;LCALL STOP_ALL
   ; LCALL TRE_2S
    CJNE R0,#2,TIEP1
     LCALL DUNG_NGATU
     LCALL TRE_QUAY
     LCALL QUAYPHAI
    TIEP1:
    CJNE R0,#10,TIEP2
      LCALL DUNG_NGATU
     LCALL TRE_QUAY
     LCALL QUAYphai
     TIEP2:
    CJNE R0,#15,DITIEP
      LCALL DUNG_NGATU
     LCALL TRE_QUAY
     LCALL QUAYTRAI
      HERE20: SJMP HERE20

DITIEP:
    TREQUAVACH: LCALL TRE0  ;CHO THOI GIAN TRE DI DI QUA
VACH NGANG
TROVE:
RET ;TRO VE CHUONG TRINH
;==================================
```

```
LAYQUA:
   LCALL STOP_ALL
   ;KEPQUA
    SETB MKEP_EN
    LCALL TRE_quay
    CLR MKEP_EN
    ;NANG QUA
    SETB MNANG_EN
    DOINANG: JB CT_NANG,DOINANG
    CLR MNANG_EN
    ;NHA QUA
    SETB MKEP_DIR
    SETB MKEP_EN
    LCALL TRE_quay

    CLR MKEP_EN
    CLR MKEP_DIR
    ;HA TAY
  ;   SETB MNANG_DIR
   ;  SETB MNANG_EN
   ;  DOIHA: JB CT_HATAY,DOIHA
   ;  CLR MNANG_EN
   ;  CLR MNANG_DIR
 RET
;==================================
DUNG_NGATU:
        LAPNT:
       JB P1.4,THOATNT
        MOV A,P1
        ANL A,#00001111B
        MOV P0,A
    LJMP LAPNT
    THOATNT:
     LCALL STOP_ALL
     LCALL TRE_QUAY
     MOV P0,#00111111B
     DOINT: JNB P1.4, DOINT
     LCALL STOP_ALL
     MOV P0,#01011111B ;TINH CHINH
     LCALL STOP_ALL
```

```
RET
;==================================
QUAYTRAI:
   MOV P0,#00101111B ;QUAYTRAI
   LCALL TRE_QUAY
   LAPQT:
   MOV A,p1
   ORL A,#11111001B
   CJNE A,#11111111B,LAPQT
   LCALL STOP_ALL
   LCALL TRE_100
   MOV P0,#00011111B ;QUAYPHAI

  LAPQT2:
   MOV A,p1
   ORL A,#11111001B
   CJNE A,#11111111B,LAPQT2
   LCALL STOP_ALL
   LCALL TRE_QUAY
      LCALL TRE_QUAY
 RET
;==============================================
============
QUAYPHAI:
   MOV P0,#00011111B ;QUAYTRAI
   LCALL TRE_QUAY
   LAPQP:
   MOV A,p1
   ORL A,#11111001B
   CJNE A,#11111111B,LAPQP
   LCALL STOP_ALL
   LCALL TRE_100
   MOV P0,#00101111B ;QUAYPHAI
  LAPQP2:
   MOV A,p1
   ORL A,#11111001B
   CJNE A,#11111111B,LAPQP2
   LCALL STOP_ALL
   LCALL TRE_QUAY
      LCALL TRE_QUAY
```

```
 RET
;=============================================
====
NGATT1:

RETI

STOP_ALL:
   MOV P0,#0FFH
RET
;====================CAC THU TUC
TRE=================
TRE0:  ;THU TUC DE TAO TRE QUA VACH NGANG NGA TU
   mov r7,#50
        again0: mov TH0,#HIGH(-10000)
              mov TL0,#LOW(-10000)
        setb tr0
        wait0:
       MOV A,P1
       ANL A,#00001111B
       MOV P0,A
   jnb tf0,wait0
        clr tr0
        clr tf0
        djnz r7,again0
RET
TRE_500:  ;    1s
   mov r7,#50
        again_1S: mov TH0,#HIGH(-10000)
              mov TL0,#LOW(-10000)
        setb tr0
        wait_1S: jnb tf0,wait_1S
        clr tr0
        clr tf0
        djnz r7,again_1S
RET
TRE_QUAY:  ;
   MOV R7,#100
        again: mov TH0,#HIGH(-10000)
              mov TL0,#LOW(-10000)
```

```
        setb tr0
        wait: jnb tf0,wait
        clr tr0
        clr tf0
        djnz r7,again
RET
TRE_100:  ;
   MOV R7,#10
        again: mov TH0,#HIGH(-10000)
               mov TL0,#LOW(-10000)
        setb tr0
        wait: jnb tf0,wait
        clr tr0
        clr tf0
        djnz r7,again
RET

TRE_2S:  ;    1s
   mov r7,#200

        again_2S: mov TH0,#HIGH(-10000)
               mov TL0,#LOW(-10000)
        setb tr0
        wait_2S: jnb tf0,wait_2S
        clr tr0
        clr tf0
        djnz r7,again_2S
RET
END
```

**6.3.Kĩ thuật dò đường.**

Bài học này sẽ nêu lên các phương pháp lập trình dò đường trong robocon. Việc lập trình dò đường trong robot thực chất là việc xử lý tín hiệu từ sensor và đưa ra các lệnh điều khiển động cơ để đảm bảo cho robot có thể bám theo vạch trắng trên sân.Khi lập trình ,các bạn phải xác định được tất cả các trường hợp có thể xảy ra của sensor ,từ đó đưa ra các tín hiệu điều khiển động cơ hợp lý.

Quay về bài sensor ,ta có sơ đồ bố trí sensor


4 sensor 2,3,4,5 dùng để bám đường.Khi trên sân thi đấu ,giả sử khi robot lệch sang phải sensor 2 chạm vạch trắng ,sensor 4 chạm nền ,bạn phải điều chỉnh cho động cơ phải quay nhanh hơn ,động cơ trái chậm lại hay dừng để robot quay lại vị trí.Tương tự ,khi robot lệch sang trái ,bạn cũng điều khiển tương tự như vậy.Bằng cách xử lý như thế ,bạn sẽ giúp cho robot có thể bám đường một cách hiệu quả.

Khi gặp ngã tư ,hai sensor ngoài cùng 1,6 sẽ chạm vạch trắng ,có mức tín hiệu 0 đưa vào vi điều khiển .Bạn cần lập trình đếm số lần mức tín hiệu 0 vào vi điều khiển để từ đó suy ra ngã tư robot đã đi.

**6.4.Kĩ thuật chống nhiễu**

Như đã học ở bài sensor ,khi thi đấu ảnh hưởng của đèn cao áp trên sân có thể gây ra nhiễu lên sensor dẫn tới việc nhận biết tín hiệu sai.Để chống nhiễu ,bạn có thể sử dụng chương trình để xử lý.

Có nhiều trường hợp do nhiễu nên sensor có thể nhận nhầm vạch trắng.Do đó ,bạn phải để vi điều khiển kiểm tra thành nhiều lần.

Mã nguồn .

```
void golong(unsigned char songatu)
{
  unsigned char d=0;
  unsigned char dem=0;
  motor(forward);
  while (1)
        {

        if ((out_left==nen)&&(out_right==nen)) motor(forward);
        if ((out_left==nen)&&(out_right==vach))
          {
           motor(right_stop);
           motor(left_go);
          }
        if ((out_right==nen)&&(out_left==vach))
          {
          motor(left_stop);
          motor(right_go);

          }
```

```
        // bat nga tu

      if ((out_left==vach)&&(out_right==vach))
        {

         while ((LEFT==vach)&&(RIGHT==vach))
         {
           d++;
          if (d==100)
           {
             dem++;
             d=0;
            if (dem<songatu)
              { h_thi(dem); motor(forward); delay(1200);}
            break;//thoat khoi while
           }//if
          }// while
        if (dem==songatu) {h_thi(dem);motor(stop);dem=0;break;}
         }
           // chinh lech nhieu

    }//end of while (1)
}//end of golong_ngatmo
```

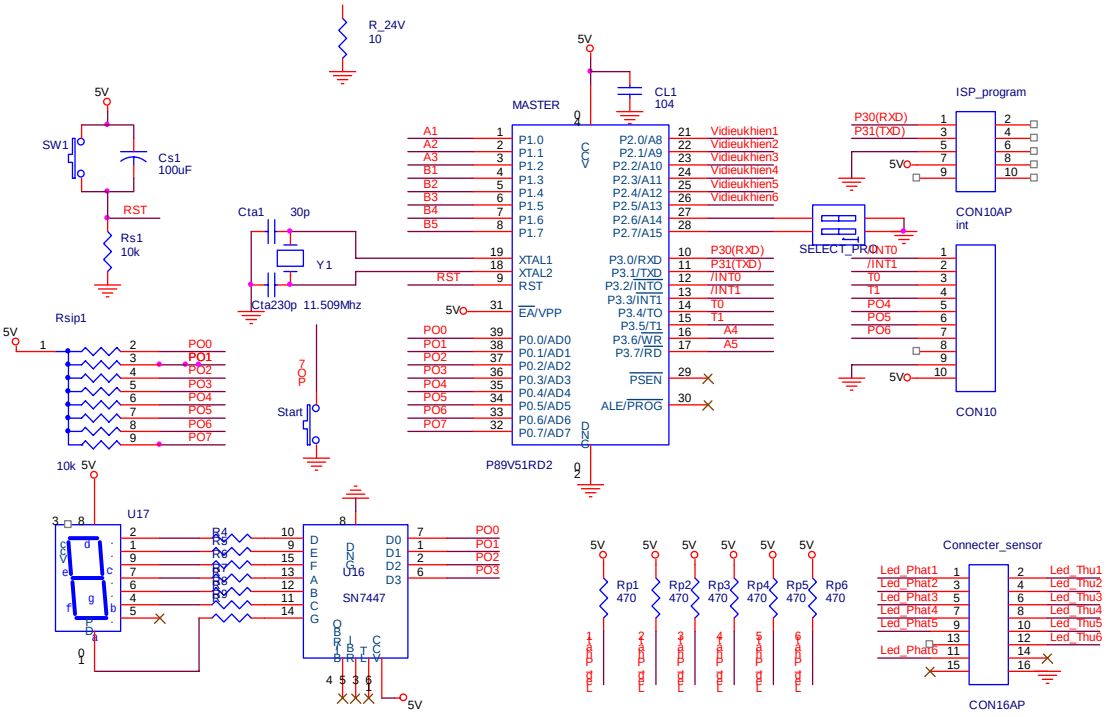
# BÀI 7:CHIẾN THUẬT THI ĐẤU

Trong khi thi đấu ,có rất nhiều các tính huống xảy ra .Do đó người lập trình cần phải xây dựng nhiều chiến thuật thi đấu khác nhau để đảm bảo có thể xử lý kịp thời mọi tình huống có thể có trên sân.

đẻ hiển thị chiến thuật thi đấu ,bạn có thể dung led 7 thanh hay màn hình LCD ,một hệ thống các phím bấm để điều khiển ,để lựa chọn các giải pháp thi đấu.Thực chất ,ở phần này chúng ta đã quy về bài toán lập trình giao tiếp vi điều khiển với bàn phím và màn hình LCD ,led 7 thanh

## 7.1.Sơ đồ nguyên lý

a)Giao tiếp với led 7 thanh




Ở sơ đồ trên ,ta sử dụng 1 led 7 thanh để hiển thị các chiến thuật thi đấu 1 ,2 ,3.v.v.2 phím bấm choice và start để lựa chọn chiến thuật.Một led 7 thanh có thể hiển thị được 10 chiến thuật thi đấu tương ứng với các số từ 0 đến 9.Ngoài ra ,led 7 thanh còn được dùng để hiển thị số ngã tư.

## 7.2.Mã nguồn

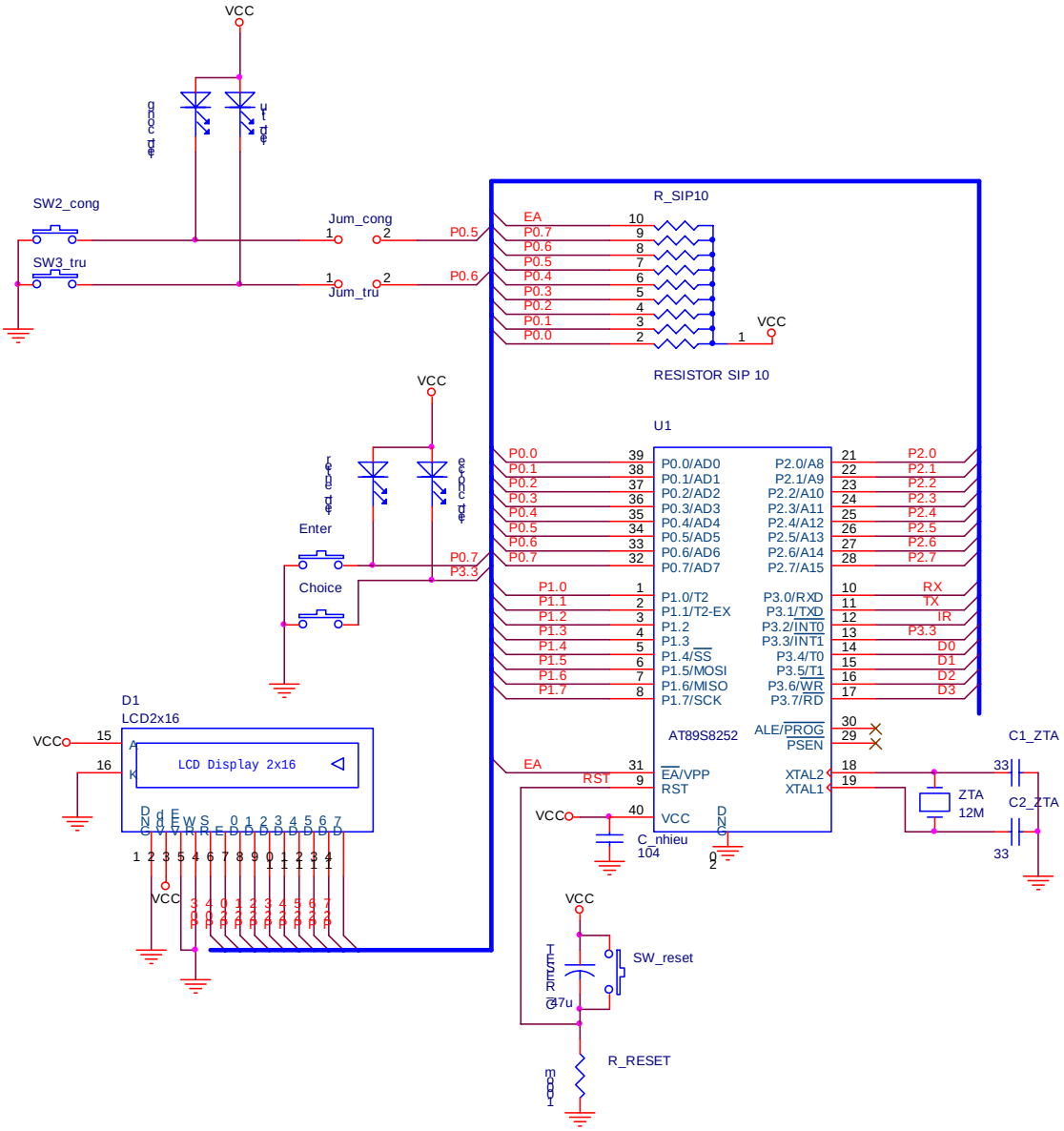
```
void main()
 {
  unsigned char tam=0;
```

```
init();
P2_6=0;
P2_7=0;
h_thi(tam);
selection();

   switch (select)
    {
     case 0:
           {
             hanhtrinh0();
           }
           break;

     case 1:{
            hanhtrinh1();

          }
           break;
     case 2:{
            hanhtrinh2();
          }
           break;
     case 3:{
             hanhtrinh3();
           }
          break;
     case 4:{
            hanhtrinh4();
          }
           break;
     case 5:{
            hanhtrinh5();
           }
           break;
     case 6:{
             hanhtrinh6();
          }
           break;
     case 7:{
```

```
                hanhtrinh7();
               }
               break;
        case 8: {
                hanhtrinh8();
               }
                break;
        case 9:{
               hanhtrinh9();
               }
                break;
       }//end of swith
        delay(50000);
        h_thi(tam);

    }
                    /* cac ham chuc nang */

void selection()
   {
     P2_6=0;
     P2_7=0;
     while (P2_7==0)
      {

        while (P2_6==1)
           {
            select++;
            if (select>9) select=0;
            h_thi(select);
            delay(12000);
           }
      }
}//end
  void h_thi(unsigned char so)
      {
       if (so==0) {P0_3=0;P0_2=0;P0_1=0;P0_0=0;}
       else if (so==1) {P0_3=0;P0_2=0;P0_1=0;P0_0=1;}
       else if (so==2) {P0_3=0;P0_2=0;P0_1=1;P0_0=0;}
       else if (so==3) {P0_3=0;P0_2=0;P0_1=1;P0_0=1;}
```

```
else if (so==4) {P0_3=0;P0_2=1;P0_1=0;P0_0=0;}
else if (so==5) {P0_3=0;P0_2=1;P0_1=0;P0_0=1;}
else if (so==6) {P0_3=0;P0_2=1;P0_1=1;P0_0=0;}
else if (so==7) {P0_3=0;P0_2=1;P0_1=1;P0_0=1;}
else if (so==8) {P0_3=1;P0_2=0;P0_1=0;P0_0=0;}
else if (so==9) {P0_3=1;P0_2=0;P0_1=0;P0_0=1;}
 }
```

b) Sơ đồ dùng LCD



Sơ đồ trên thay thế led 7 thanh bằng LCD .Nhìn chung dung LCD để hiển thị chiến thuật thi đấu chuyên nghiệp hơn (vi hiển thị được cả chữ ) tuy nhiên , dung LCD khá tốn kém và không thật sự hiệu quả .Vì thế ,tốt nhất là các bạn nên sử dụng led 7 thanh để hiển thị chiến thuật thi đấu.

```
/*****************************/
            // LCD
void lenh ()
{
RS=0;           EN=1;       delay (50);  EN=0;       delay (100);
}

void ghi ()
{
RS=1;           EN=1;       delay (50);  EN=0;       delay (100);
}


void LCDwrite(unsigned char c)
 {

   P2=c;
   ghi();
 }

void LCDputs(unsigned char *s,unsigned char row)
{
unsigned char len;
if(row==1)
{ P2=0x80;lenh (); }
else { P2=0xC0;    lenh ();}
              len=strlen(s);
              while(len!=0)
              {
                 LCDwrite(*s);
                     s++;
                     len--;
              }
}

void LCDcontrol(unsigned char dk)
```

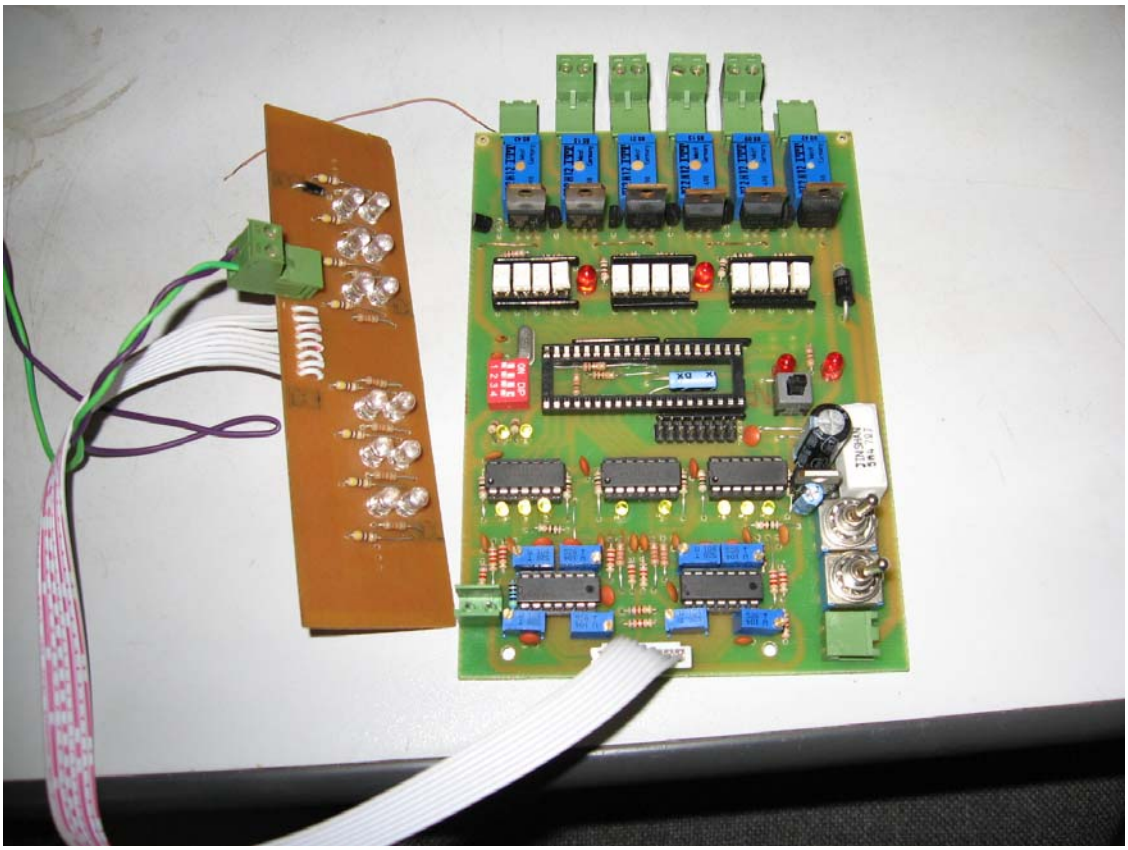
```
{
  P2=dk;      lenh ();
}

void init_LCD ()
{
delay(400);
LCDcontrol(0x38);   //LCD 2 dong _ 5x7
LCDcontrol(0x0C);         //bat hien thi, tat con tro
LCDcontrol(0x01);         //xoa man hinh
}
              /*****************************/

void keyboard()
{
 unsigned char key=0;
 unsigned char test=0;
 LCDcontrol(0x01);
 choice=enter=1;
 LCDputs("Robot Ready",1);
 delay(100000);
 LCDcontrol(0x01);
 LCDputs("1:Golong  2:Around ",1);
 LCDputs("3:Turn left  4:Turn right",2);
 delay(100000);
 LCDcontrol(0x01);
 while(1)
 {
  LCDputs("1: 2: 3: 4:”,2);
  while(choice==0)
     {
       delay(10000);
       key++;
       LCDcontrol(0x01);
       if (key==1)LCDputs("1  ",1);
       if (key==2)LCDputs("2  ",1);
       if (key==3)LCDputs("3  ",1);
       if (key==4)LCDputs("4  ",1);
       LCDputs("1:C 2:S 3:D 4:E",2);
       if (key>4) { LCDputs("0  ",1);key=0;}
```

```
    }
  if (key==1)while(enter==0)  golongr();
  if (key==2)while(enter==0)  Around();
  if (key==3)while(enter==0)  left();
  if (key==4)while(enter==0)  test=1;
  if (test==1) break;

 }//while(1)

}     // keyboard
```

# Bài 8.Hoàn thiện robot thi đấu

Ở bài này ,chúng ta sẽ hoàn thiện một robot hoàn chỉnh để thi đấu .
Chúng ta sử dụng  robot như hình dưới đây

Robot ở hình trên là robot của một đội tham gia trong cuộc thi robocon 2007.Chúng ta sẽ sử dụng robot này để thực hành .
Phần mạch của robot ,chúng ta sử dụng mạch điều khiển động cơ bằng relay Như hình dưới



Đây là mạch điều khiển của đội BK-FIRE năm 2005 ,robot hoạt động khá hiệu quả .Đấu nối sensor như hình trên ,lắp đặt mạch sensor vào phần đế của robot .Robot này sử dụng bộ ắc quy 24V DC để cấp nguồn.Mạch này sử dụng 6 sensor xếp thành hàng ngang để bám đường ,6 động cơ điều khiển các cơ cấu.
Sau khi lắp đặt mạch và sensor ,đấu nối các động cơ .Ta sẽ lập trình cho để robot có thể bám theo vạch trắng có sẵn.
Trong bài thực hành này ,chúng ta sử dụng vi điều khiển 89c52 để điều khiển robot.
Đây là toàn bộ code chương trình bám đường của robot.(viết bằng ASM)