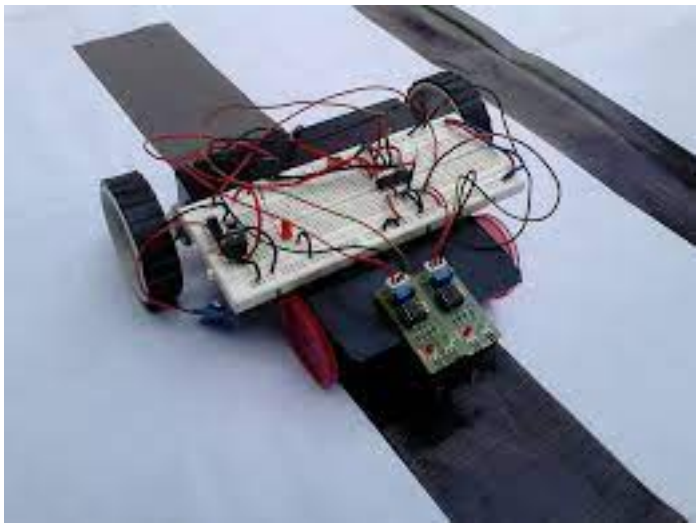
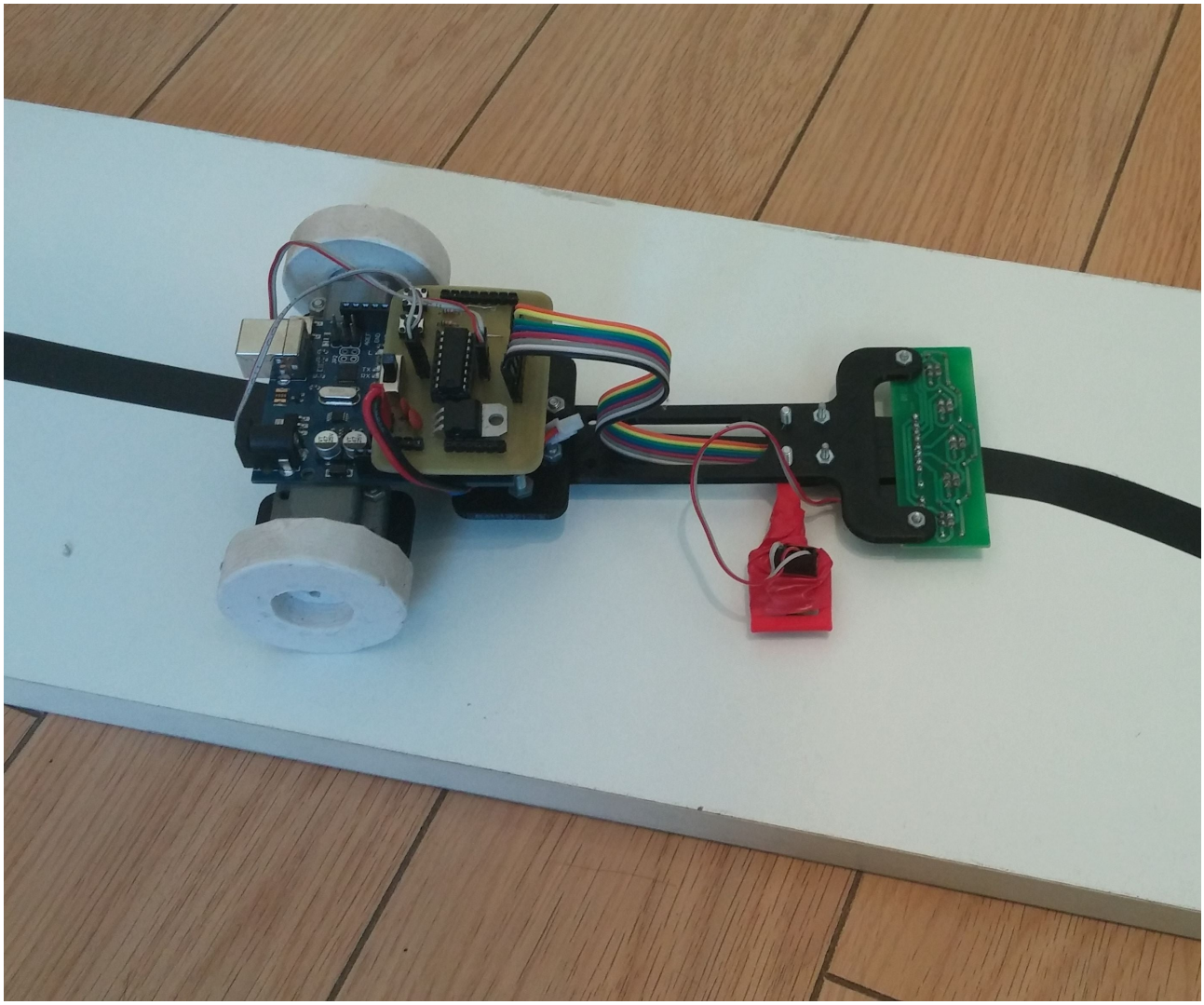


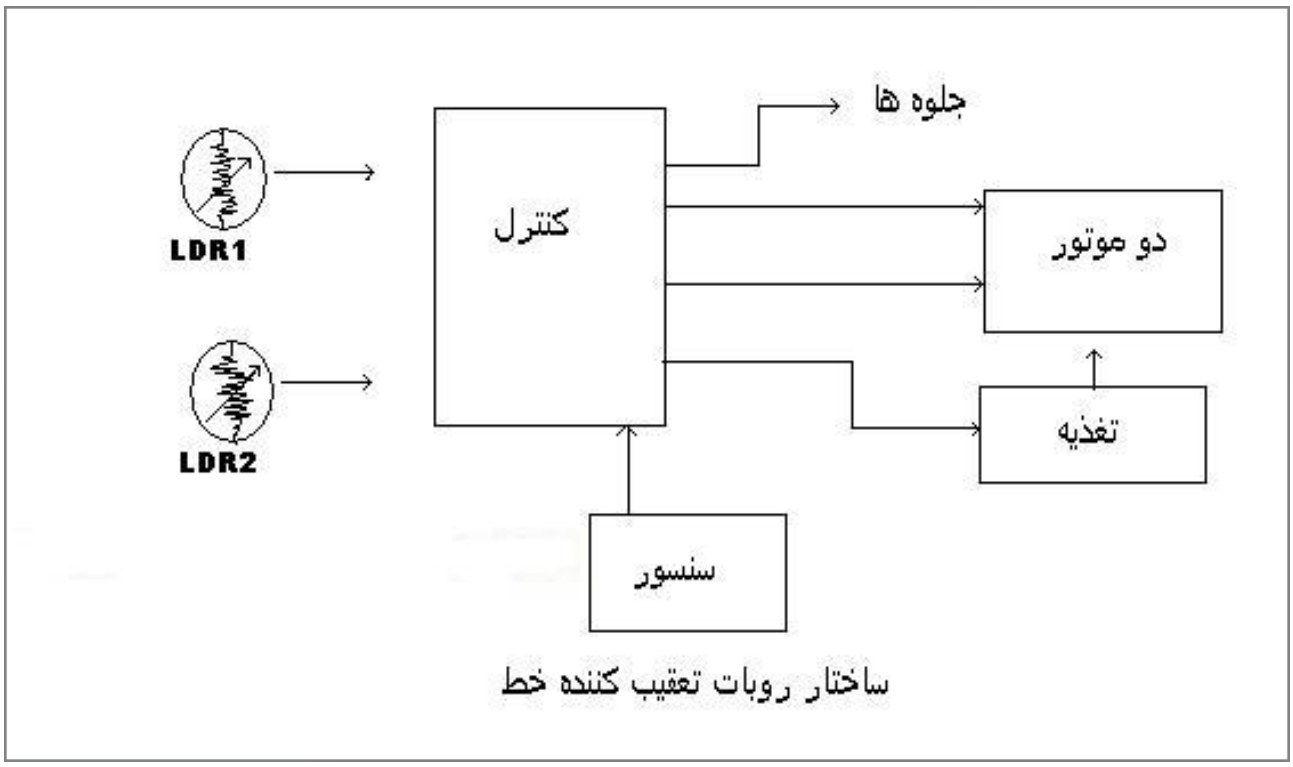
ربات تعقیب خط



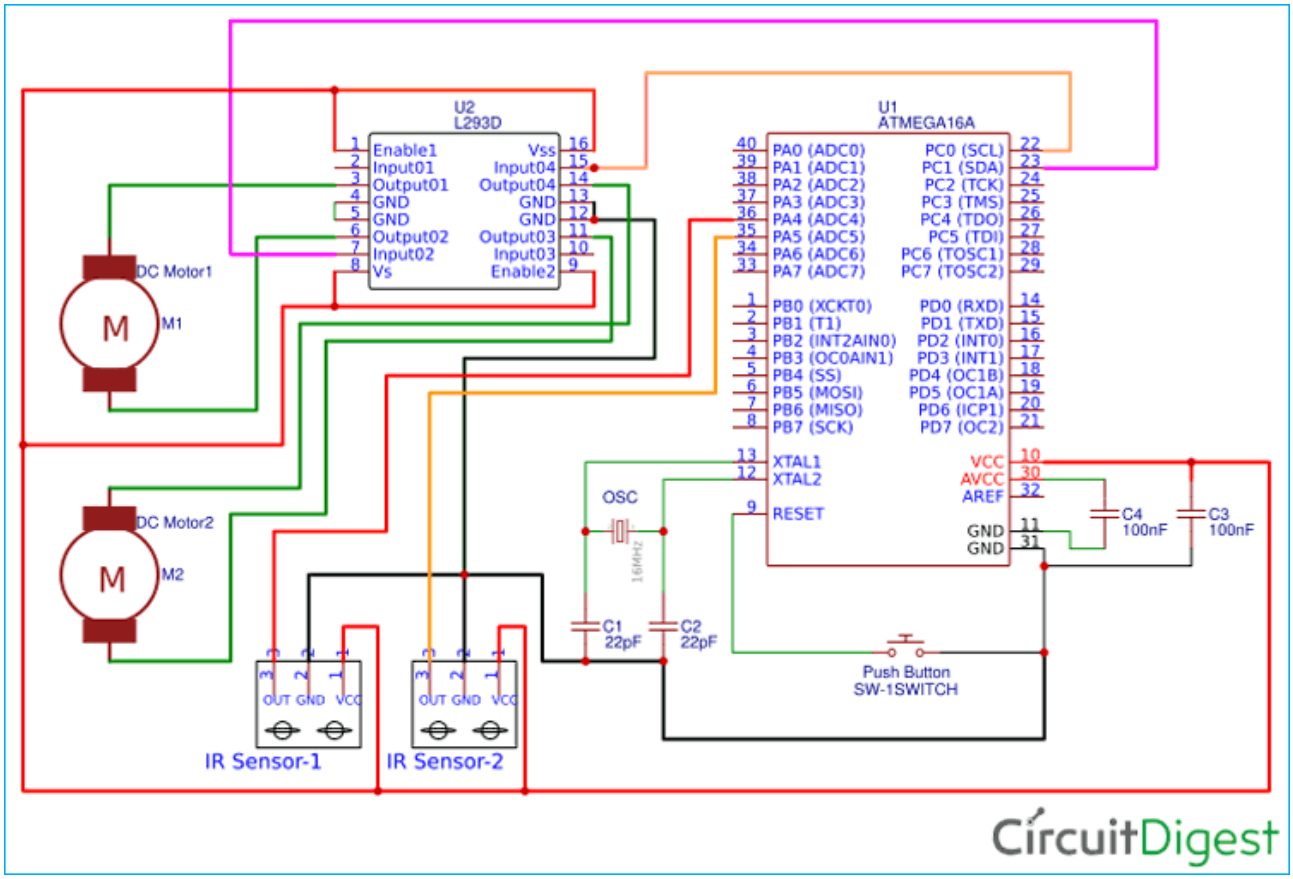
وظیفه ربات تعقیب خط دنبال کردن خط سیاه در زمینه سفید است.
این ربات از سه قسمت اصلی تشکیل شده است که عبارتند از:
1-سنسورها
2-درایور کنترل کننده
3-موتورها

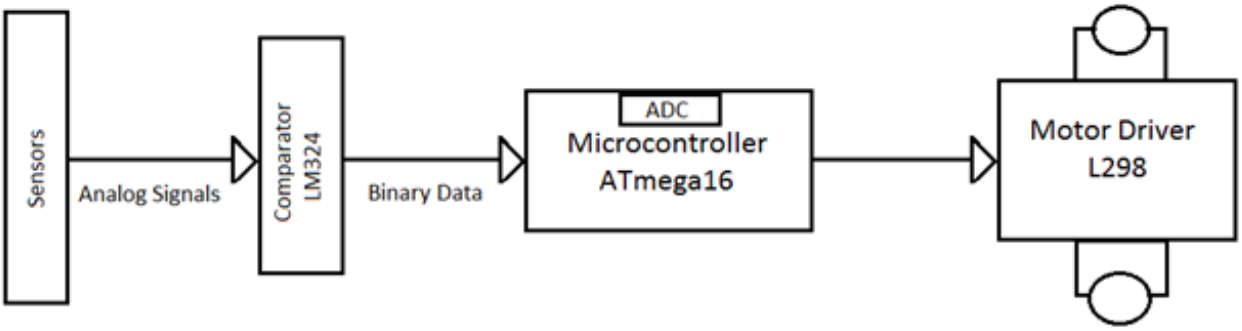
هر قسمت با توجه به اجزای اصلی دارای قسمتهای دیگری هستند برای مثال از ic درایور برای اتصال موتورها به درایور کنترل کننده و از میکروکنترلر ATmega16 به عنوان قلب ربات در درایور کنترل کننده استفاده میکنیم.

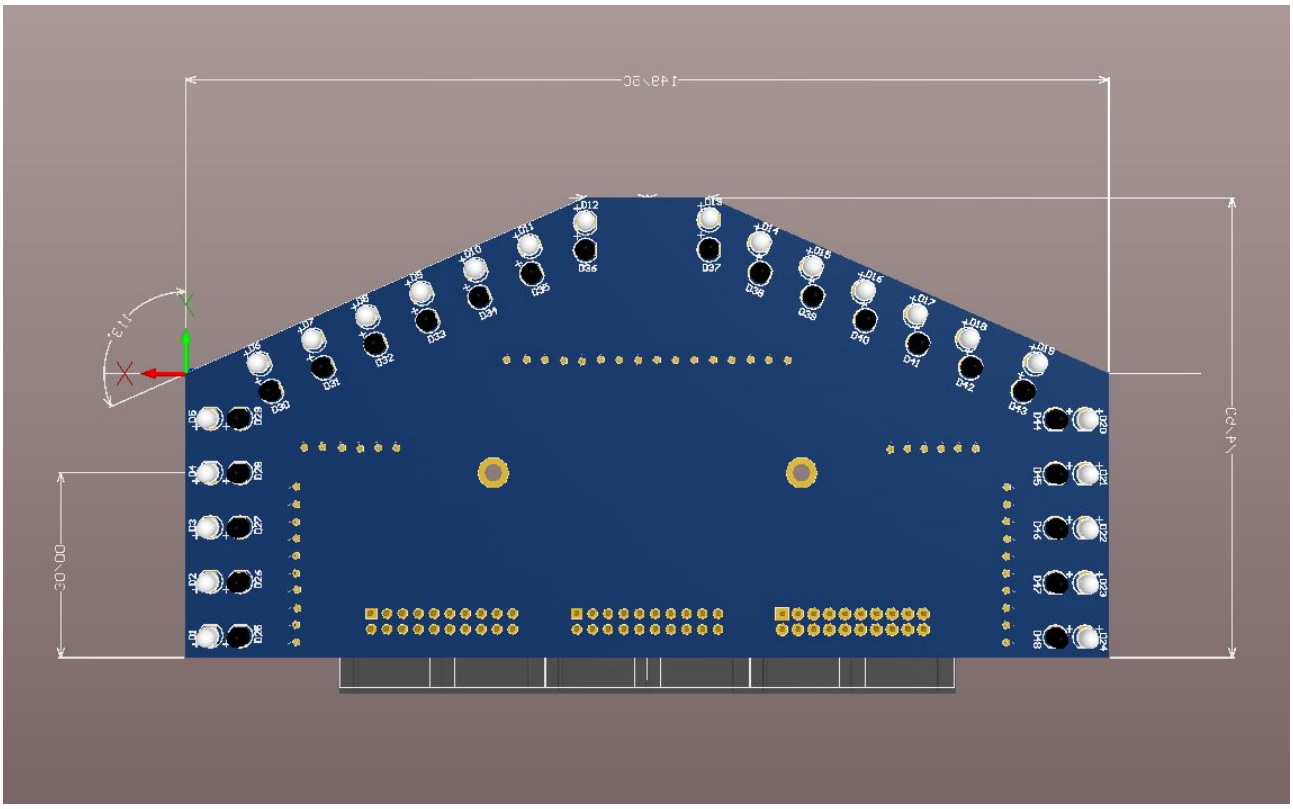
برای ساخت این ربات هر قسمت را به همراه نحوه کارکرد و نحوه بکار بردن آن توضیح خواهیم داد.
يك روبات تعقيب كننده خط كه دو موتور وهفت سنسور دارد فقط به قسمت هاي زير احتياج دارد:



ساختار روبات تعقیب کننده خط



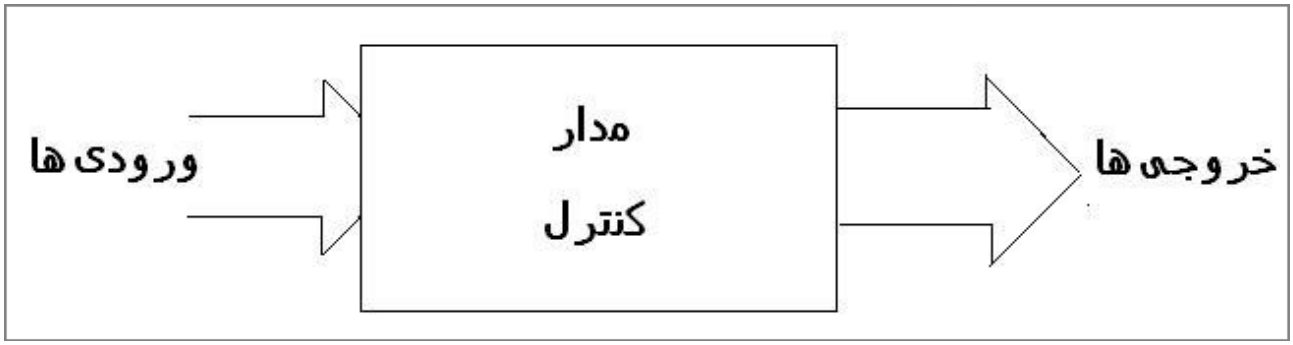




بخش اول:

در این بخش ابتدا با یکسری قطعات را توضیح می دهیم و از توضیحات درباره مقاومت خازن دیود و ... اجتناب میکنیم چرا که در درس مدار الکتریکی گفته شده و تکرار مکررات خواهد شد.

روبات تعقیب باید بتواند مسیري مشكي رنگ به عرض 18 میلیمتر را در زمینه سفید دنبال کند و از بردگي ها و زاویه هاي مسیر عبور کند . تشخیص رنگ سیاه از سفید با سنسور هاي مادون قرمزي است که در کف روبات کار گذاشته مي شود.



برخی قطعات بکار رفته با کاربرد های آن ها در زیر امده است:

سنسور ها(۷ عدد)

این گونه از سنسور ها از یك فرستنده و یك گیرنده مادون قرمز تشکیل شده است.

پرتوهاي مادون قرمز توسط فرستنده فرستاده شده و به سطح برخورد مي کند و بازتاب آن توسط گیرنده دریافت مي شود .

اینگونه از سنسورها به دلیل اینکه از محیط اثر نمي گیرند , درصد خطا بسیار پایني دارند بنابراین ما در مورد این سنسورها صحبت مي کنیم.

سنسور هاي مادون قرمز به دو دسته تقسیم مي شوند .

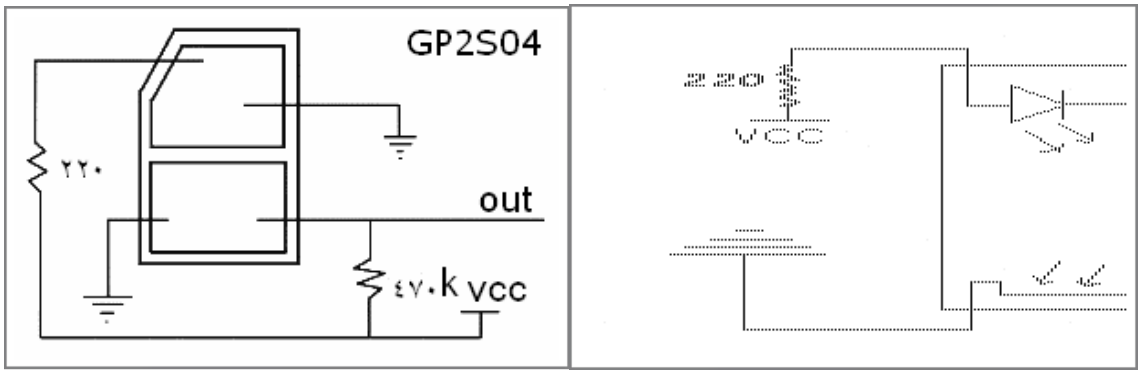
دسته اول LED فرستنده و گیرنده جدا از هم است. این گونه از سنسورها در اندازه 5mm و 3mm موجود است. نوع 3 میلیمتري به دلیل اینکه LED فرستنده و گیرنده هم رنگ هستند تشخیص آنها سخت است و باید توسط مالتي متر انجام شود.

مشكل ديگر اين سنسور ها اين است كه نوع تقلبي آن نيز بسيار زياد است و امكان دارد از بين 5 سنسور 1 يا 2 تا خراب باشد .

گونه ي ديگر سنسور هاي مادون قرمز به صورتي است كه فر ستنده و گيرنده با هم در يك قطعه قرار دارند اين گونه از سنسور ها داراي اندازه بسيار كوچك و دقت خوبي هستند . براي بایاس كردن سنسور : مقاومت R2 براي تنظيم جريان Led فرستنده قرار دارد براي فرستنده 20 تا 40 ميلي آمپر است.

Led گيرنده به صورت معكوس بایاس مي شود گيرنده به صورت يك مقاومت متغير در برابر اشعه مادون قرمز عمل مي كند با دريافت اشعه مادون قرمز مقاومت آن شدت افت پيدا مي كند ولتاژ پايه مثبت مقايسه گر را كاهش مي دهد تا اينكه خروجي آن يك (vcc)شود .

نمایی این سنسور ها در شکل زیر آمده است:



موتور هاي **DC :**

جهت چرخش موتور به جهت جريان بستگي دارد با معكوس كردن جهت جريان مي توانيم جهت حركت موتور را عوض كنيم.

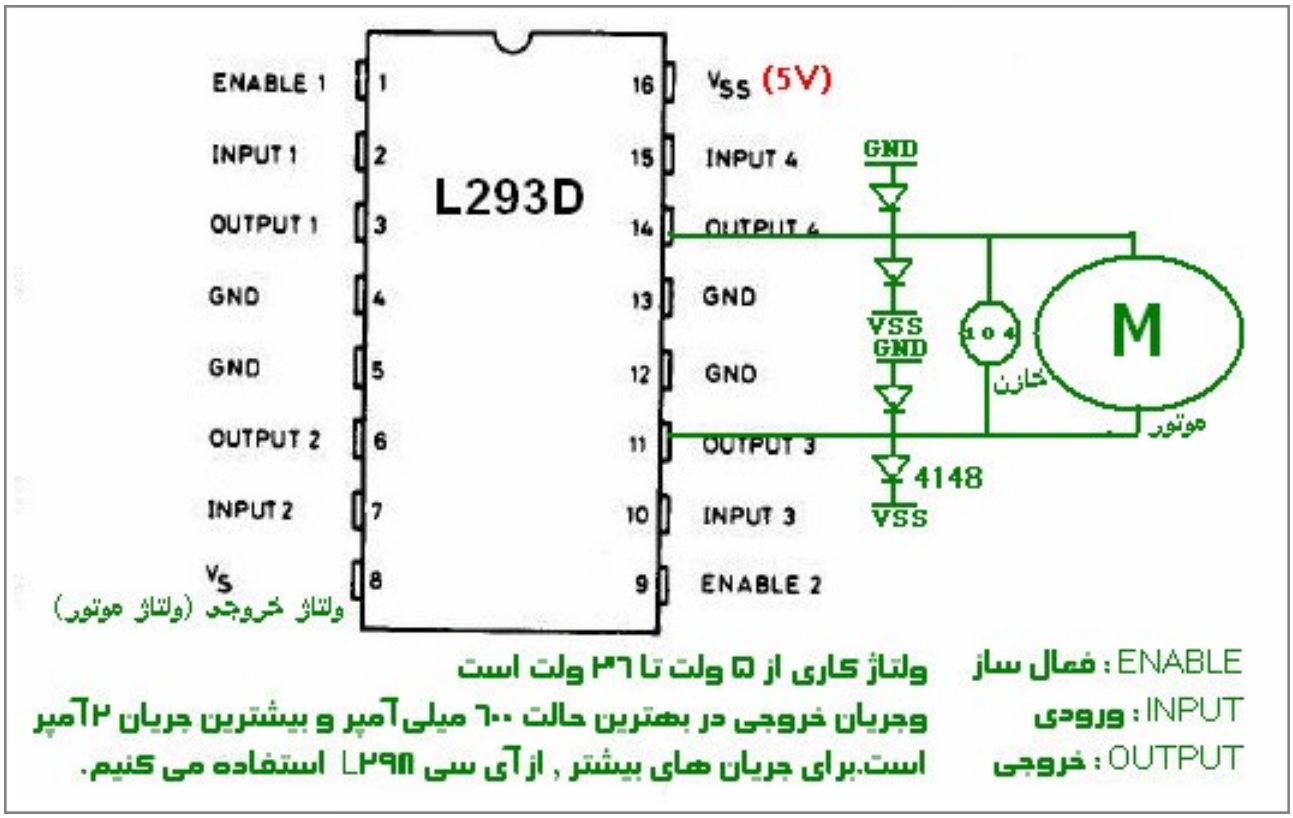
سرعت يك موتور DC بر حسب دور بر دقيقه بيان مي شود به جريان و بار موتور بستگي دارد بدون بار 500 rpm تا 1000 rpm

ولتاژ: موتور هاي DC كوچك را مي توان با ولتاژ هاي نامي در محدوده 1.5 V تا 48 V تهيه نمود ولتاژ مشخص شده بر روي موتور , ولتاژ نامي موتور را نشان مي دهد كه براي

كا ركرد معمولي موتور اعمال مي شود در عمل ولتاژ نامي بسيار مهم مي باشد , چرا كه اين ولتاژ نشان دهنده حد اكثر ولتاژ توصيه شده اي كه مي توان به موتور اعمال نمود.

جريان: مقدار جريان عبوري از موتور , زماني كه موتور تحت ولتاژ نامي كار مي كند , به مقدار بار بستگي دارد با افزايش مقدار بار مقدار جريان افزايش مي يابد . بايد از كاركرد موتور با بار بيش از حد مجاز جلوگيري كنيم باعث سوختن موتور مي شود. موتور هاي DC متداول داراي جريان هاي كار كرد در محدوده 50 ميلی آمپر تا بيش از 2 آمپر مي با شند.

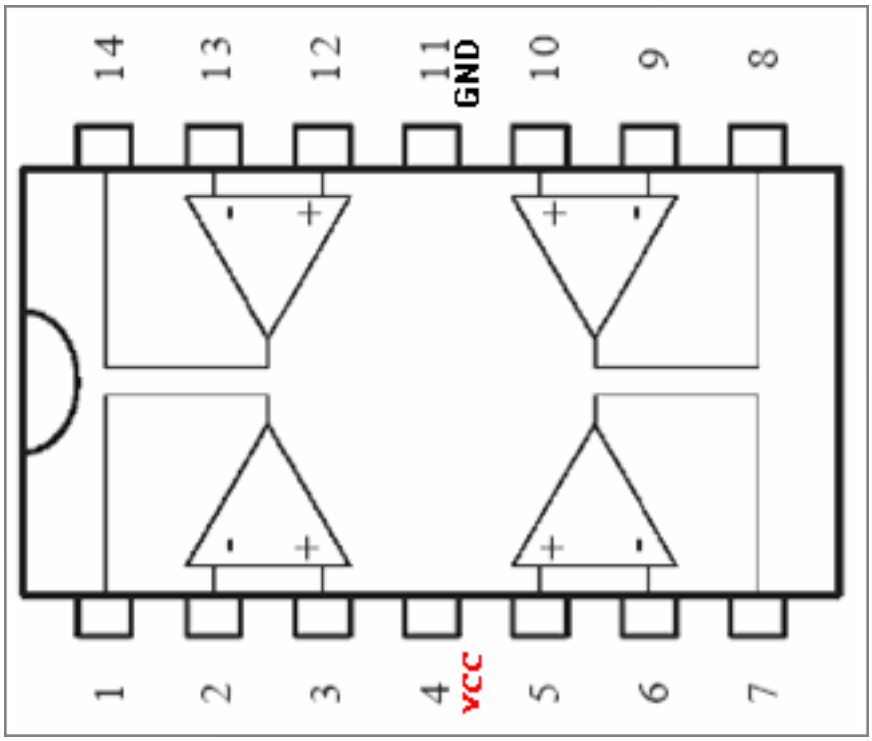
درايور موتور ها (L293 آي سي)



آيسي آپ امپ

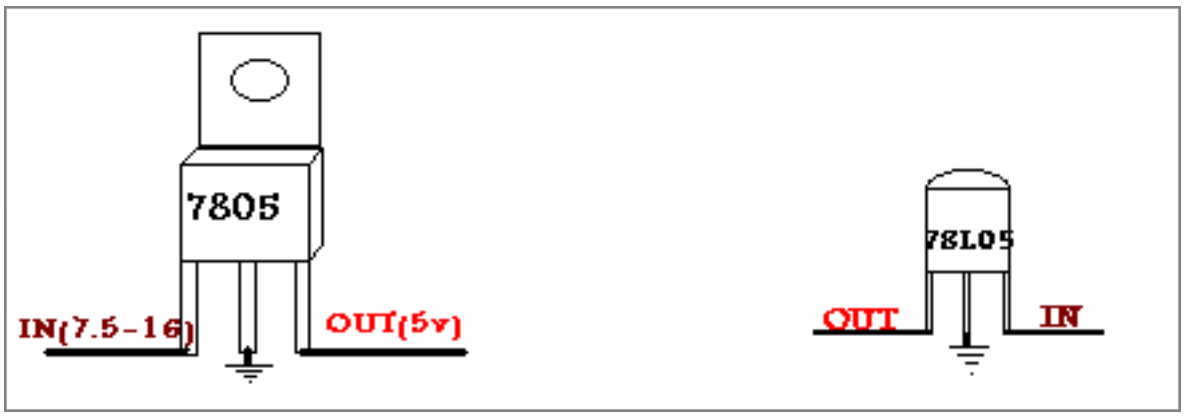
LM324

اين آي سي 4 آپ امپ دارد در يك حالت خاص: ورودي آنالوگ را از سنسور ها را با يك ولتاژ مرجع مقايسه مي كند و به صفر و يك براي ميكرو تبديل مي كند.



آي سي ريگولاتور 7805 :

آيسي7805از500 maتا1آمپرجريانميدهد آي سي 78L05 در حدود 150 ma جريان مي دهد.



ميكروكنترلر:

هر يک از ميکروکنترلر هامجموعه دستورات و مجموعه ثبات هاي خود را دارد بنابر اين با يکديگر سازگار نيستند.

برنامه اي که بر روي يکي از آنها نوشته شود بر روي ديگري قابل اجرا نيست.

تفاوت ميکرو کنترلر و ميکرو پروسسور هاي همه منظوره : ميکرو پروسسور ها فاقد RAM و ROM و پورت هاي I/O در درون خود تراشه هستند. سرعت ميکرو پروسسور ها بالا تر از ميکرو کنترلر ها است ميکرو کنترلر ها داراي يك RAM و ROM و پورت هاي I/O از پيش تعيين شده هستند ولي براي ميکرو پروسسور ها خودمان طراحي مي کنيم.

انتظاراتي که از ميکروکنترلر داريم:

1- برآورده کردن نياز محاسبات کار به صورت موثر و مقرون به صرفه .

2- در دست داشتن نرم افزار هاي کمکي مانند کامپايلر ها اسمبلر ها و عيب ياب ها.

3- منابع گسترده و قابل اعتماد براي ميکرو کنترلر ها .

4- به وسيله Programer و کامپيوتر مي توان ميکرو را پروگرام کرد.

ميکروکنترلرهاي A VR به سه دسته تقسيم مي شوند :

1-Tiny AVR (ATiny)

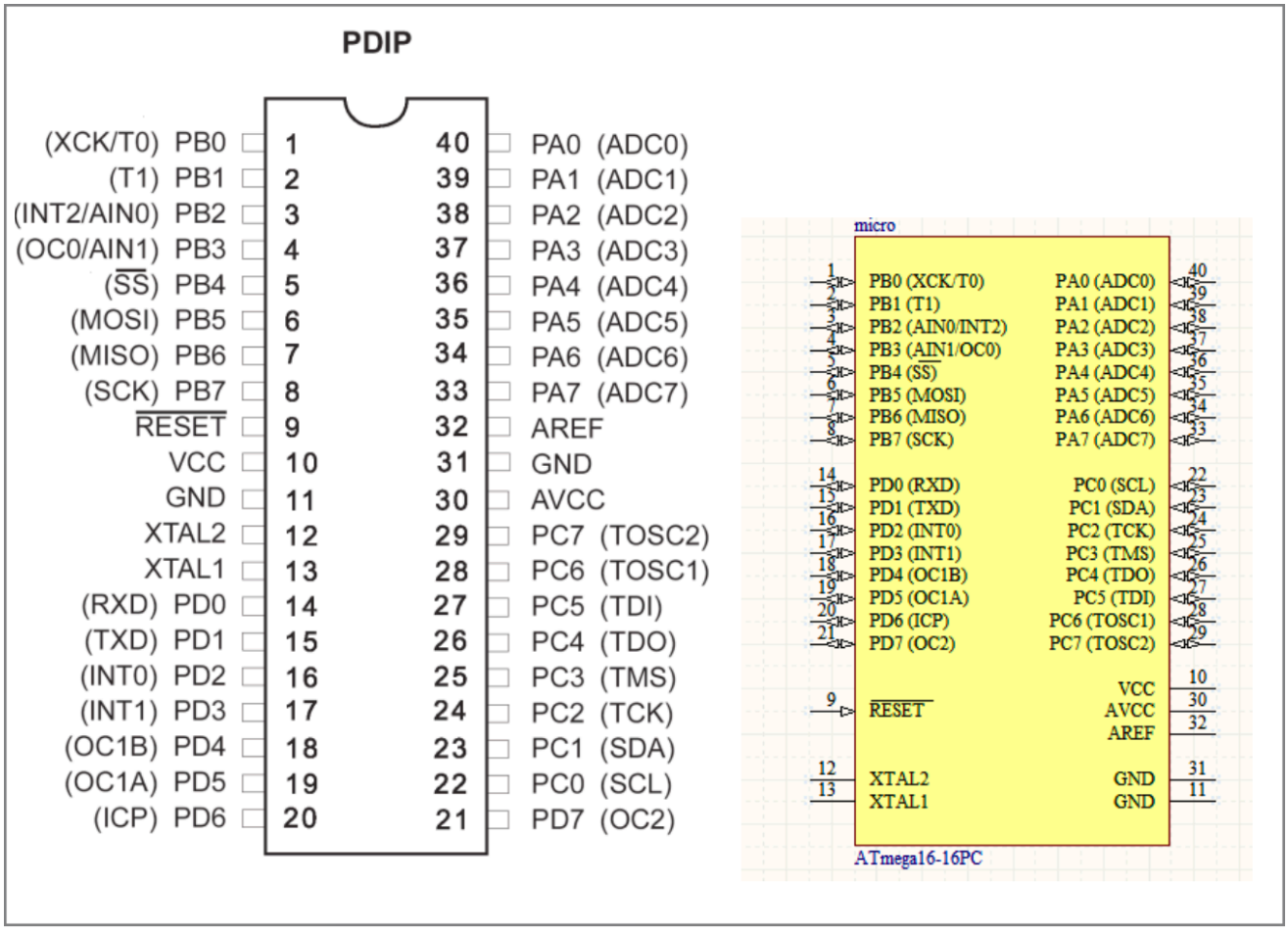
2-Classic AVR (AT90s)

(ATmega) mega AVR-3

تفاوت اين 3 نوع به امكانات موجود در آنها مربوط مي شود .

Tiny AVR ها غالبا تراشه هايي با تعداد پايه و مجموعه دستورات كمتري نسبت به Mega A VR ها مي باشند و به عبارتي از لحاظ پيچيدگي حد اقل امكانات را دارند Mega A VR ها حداكثر امكانات را دارند و Classic A VR ها جايي بين اين دو نوع قرار مي گيرند.البته از آنجايي كه از بين سه دسته ذكر شده Classic A VRها, قبل از دو گروه ديگر توليد شده اند امروزه در طرح هاي جديد كمتر از آنها استفاده مي شود و عملا هر يك از آنها با تراشه اي از گروه Mega AVR يا Tiny AVR جايگزين شده اند.

برای این ربات ما از میکرو کنترلر AT mega 16 استفاده کرده ایم که شمای آن در زیر آمده است:



ویژگی های میکروکنترلر Atmega16A :

- پایداری بالا

- مصرف توان کم
- میکروکنترلر ۸ بیتی Atmel
- معماری RISC پیشرفته ، ۱۳۱ دستورالعمل قدرتمند ، اجرای اغلب دستورالعمل ها در یک کلاک ، ۳۲ رجیستر ۸ بیتی با کاربرد عمومی ، بیش از ۱۶ میلیون دستورالعمل بر ثانیه (MIPS) با کلاک ۱۶ مگاهرتز (MHz)
- ۱۶ کیلوبایت حافظه فلش قابل برنامه ریزی
- ۵۱۲ بایت EEPROM
- ۱ کیلوبایت SRAM
- قابلیت برنامه ریزی حافظه فلش تا ۱۰,۰۰۰ بار و حافظه ی EEPROM تا ۱۰۰,۰۰۰ بار
- ماندگاری برنامه تا ۲۰ سال در دمای ۸۵ درجه و ۱۰۰ سال در دمای ۲۵ درجه سانتی گراد
- دارای قفل برنامه برای حفاظت از نرم افزار
- رابط JTAG مطابق استاندارد IEEE 1149.1
- دارای ۲ تایمر ۸ بیتی
- دارای یک تایمر ۱۶ بیتی
- دارای RTC با اسیلاتور مجزا
- ۴ کانال PWM
- ۸ کانال ADC ده بیتی
- رابط سریال TWO WIRE یا TWI
- USART
- رابط سریال SPI در حالت Master/Slave
- دارای تایمر دیده بان با اسیلاتور مجزای داخلی
- مقایسه گر آنالوگ داخلی
- دارای اسلاتور RC کالیبره شده داخلی
- ۳۲ پورت ورودی و خروجی
- ولتاژ تغذیه ۲,۷۵ تا ۵,۵ ولت
- پشتبانی از فرکانس ۰ تا ۱۶ مگاهرتز
- مصرف انرژی در فرکانس ۱ مگاهرتز ، ولتاژ ۳ ولت ودمای ۲۵ درجه : فعال : ۰,۶ میلی آمپر – حالت بیکاری: ۰,۲ میلی آمپر – حالت Power Down کمتر از ۱ میکرو آمپر.

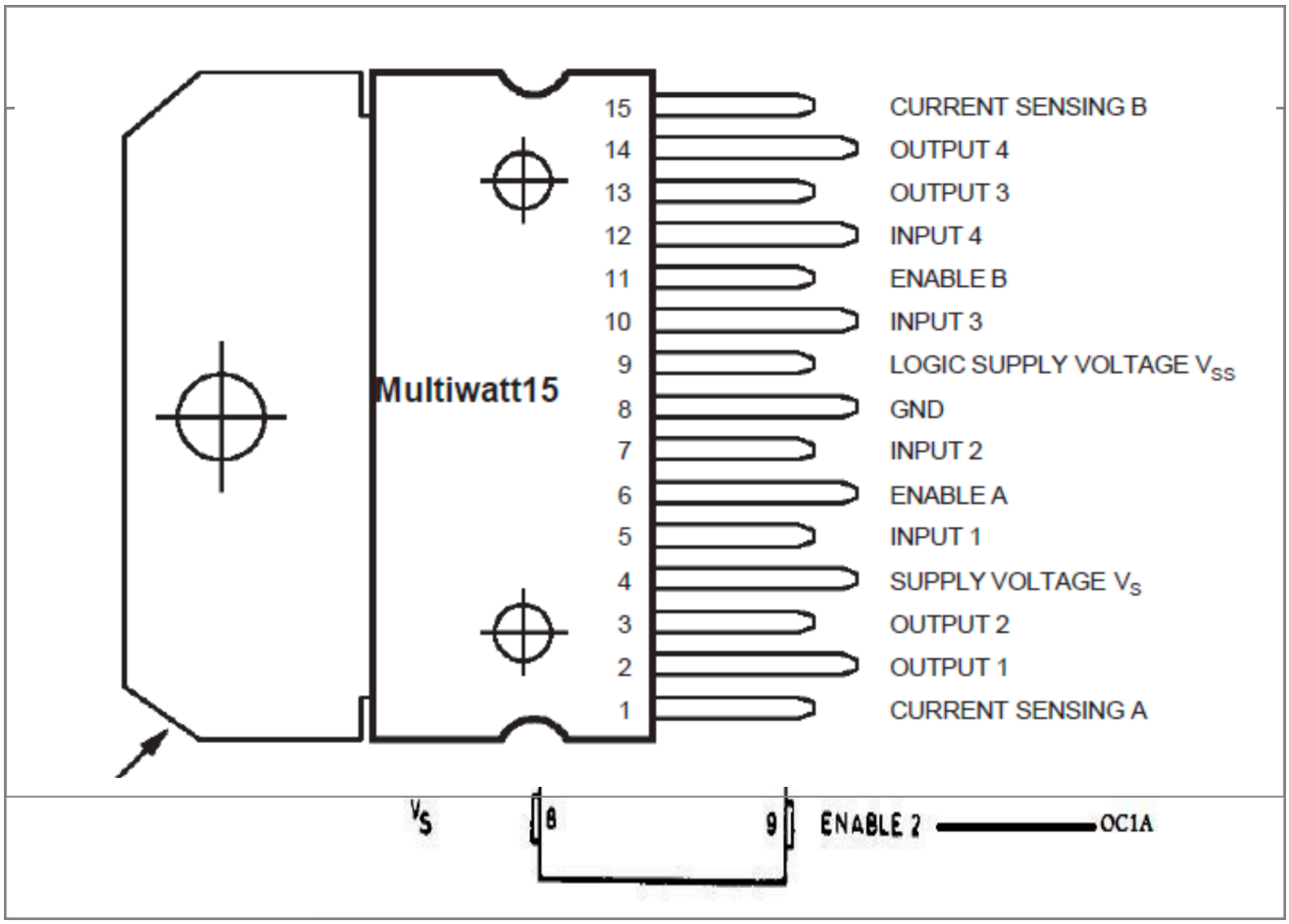
ساخت برد سنسور:

شکل چیدن سنسور ها بسیار مهم است عمومی ترین روش بصورت 8 میباشد ولی ما به صورت ذوزنقه سنسور ها را چیده ایم..برای ساخت مدار میتوان از برد سوراخ که ما از فیبر چهار هزار سوراخ استفاده کرده ایم و یا یک مدار روی pcb طراحی کرد.
ابتدا پایه های vcc فرستنده وگیرنده را به هم متصل میکنیم تا جریان را در هردو برقرار کنیم فقط در انتها gnd گیرنده را چک میکنیم پایه gndفرستنده را با یک مقاومت 222 اهم به هم وصل میکنیم(البته اندازه مقاومت به تعداد سنسور ها نیز مربوط میشود)

بعد از اتصال برد سنسور باید مقاومت متغییر را جهت تنظیم حساسیت سنسورها روی برد سوار کنیم برای اینکار پایه سوم مقاومت متغییر را به GND برد و یکی از پایه ها را به GND گیرنده سنسور متصل میکنیم.

ساخت برد مدار کنترل درایور:
ابتدا سنسور ها را به ADC یا همان تبدیل کننده آنالوگ به دیجیتال متصل میکنیم که سنسور ها به پورت های A میکرو ما متصل شده اند و نقش ورودی را دارند.
سپس از نحوه اتصال میکرو به درایور کنترل کننده و درایور به سنسور ها کد مربوطه را میزنیم.
مدار درایور جهت راه اندازی موتورها استفاده میشود در نتیجه ما میتوانیم با فرمان دادن به ای سی درایور به موتورها حرکت دهیم شماتیک مدار بصورت زیر است:

در ic درایور در پایه وجود دارد که وظیفه enable(فعال)کردن خروجی(output)را دارند این پایه ها را به oc متصل میکنیم و inputرا به دیگر پایه های میکرو متصل میکنیم هنگامی که میخواهیم موتورها را حرکت دهیم باید ابتدا به پایه های enableمیکرو فرمان دهیم.
Ic زیر درایور l298 را نشان میدهد این ای سی یک پایه کمتر از l293 دارد.



پایه های این ای سی مانند ای سی l293 است با این تفاوت که یک GND کمتر دارد.

به طور کلی در طراحی این مدار از یک هیت سینک و رگولاتور و خازن استفاده شده که ولتاژ ۱۲ ولت اداپتور به طور مستقیم به میکرو وارد نشود چون میکرو ما با ولتاژ ۶ ولت و درایود با ولتاژ ۵ ولت کار میکند .

هم چنین یک تبدیل کننده آنالوگ به دیجیتال داریم که از آن برای خواندن سنسورها استفاده میکنیم که به صورت ورودی عمل میکنند.

هم چنین دو موتور قرار داده شده است که سرعت آن ها به وسیله دو پایه تایمر یک یعنی OCR1A , OCR1Bکنترل میشود که سرعت های استفاده شده در سورس کد برنامه همگی اعدادی هستند که به صورت تجربی و با آزمون و خطا به دست آمده اند.

هم چنین در پیاده سازی این مدار بر روی زمین سفید و چسب برق مشکی جهت تشخیص سفید و مشکی برای ربات به مشکی عدد یک و به سفید عدد صفر را نسبت داده ایم هم چنین مرزی را به عنوان صفر و یک در نظر گرفته ایم که این عدد هم با توجه به اندازه گیری های به عمل آمده توسط صفر و یک ۵۰۰ می باشد

(عدد محاسبه شده باید بین ۰ و ۱۰۲۳ باشد) که عدد ۵۰۰ مرز بین صفر و یک ما می باشد که ربات ما بتواند خط سفید را از مشکی تشخیص بدهد.

سورس کد برنامه:

```
/*****************************************************
This program was created by the
CodeWizardAVR V3.12 Advanced
Automatic Program Generator
 Copyright 1998-2014 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 5/21/2018
Author  :
Company :
Comments:


Chip type               : ATmega16A
Program type            : Application
AVR Core Clock frequency: 8.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************************/

#include <mega16a.h>

#include <delay.h>

// Declare your global variables here

// Voltage Reference: AREF pin
```

```
#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) |
        (0<<ADLAR))

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | ADC_VREF_TYPE;
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=(1<<ADSC);
// Wait for the AD conversion to complete
while ((ADCSRA & (1<<ADIF))==0);
ADCSRA|=(1<<ADIF);
return ADCW;
}

void main(void)
{
// Declare your local variables here
 int S=0;
// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In
        Bit0=In
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) |
        (0<<DDA3) | (0<<DDA2) | (0<<DDA1) | (0<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) |
        (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1)
        | (0<<PORTA0);

// Port B initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In
        Bit0=In
DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) |
        (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (0<<DDB0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
```

```
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) |
      (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) |
      (0<<PORTB1) | (0<<PORTB0);

// Port C initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In
      Bit0=In
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) |
      (0<<DDC3) | (0<<DDC2) | (0<<DDC1) | (0<<DDC0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) |
      (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) |
      (0<<PORTC1) | (0<<PORTC0);

// Port D initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out
      Bit2=Out Bit1=Out Bit0=Out
DDRD=(1<<DDD7) | (1<<DDD6) | (1<<DDD5) | (1<<DDD4) |
      (1<<DDD3) | (1<<DDD2) | (1<<DDD1) | (1<<DDD0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) |
      (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) |
      (0<<PORTD1) | (0<<PORTD0);

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) |
      (0<<WGM01) | (0<<CS02) | (0<<CS01) | (0<<CS00);
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 125.000 kHz
// Mode: Fast PWM top=0x00FF
```

```
// OC1A output: Non-Inverted PWM
// OC1B output: Non-Inverted PWM
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer Period: 2.048 ms
// Output Pulse(s):
// OC1A Period: 2.048 ms Width: 0 us
// OC1B Period: 2.048 ms Width: 0 us
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=(1<<COM1A1) | (0<<COM1A0) | (1<<COM1B1) |
      (0<<COM1B0) | (0<<WGM11) | (1<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (1<<WGM12)
      | (0<<CS12) | (1<<CS11) | (1<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0<<AS2;
TCCR2=(0<<PWM2) | (0<<COM21) | (0<<COM20) | (0<<CTC2) |
      (0<<CS22) | (0<<CS21) | (0<<CS20);
TCNT2=0x00;
OCR2=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
```

```
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) |
      (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);

// USART initialization
// USART disabled
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) |
      (0<<TXEN) | (0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
// connected to the AIN0 pin
// The Analog Comparator's negative input is
// connected to the AIN1 pin
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE)
      | (0<<ACIC) | (0<<ACIS1) | (0<<ACIS0);

// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: ADC Stopped
ADMUX=ADC_VREF_TYPE;
ADCSRA=(1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) |
      (0<<ADIE) | (1<<ADPS2) | (1<<ADPS1) | (0<<ADPS0);
SFIOR=(0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);

// SPI initialization
// SPI disabled
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) |
      (0<<CPOL) | (0<<CPHA) | (0<<SPR1) | (0<<SPR0);
```

```
// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) |
      (0<<TWIE);
      PORTD.0=1;
      PORTD.1=0;
      PORTD.2=0;
      PORTD.3=1;
      OCR1AL=0;
      OCR1BL=0;

while (1)
      {
      // Place your code here


      S=0;
      if(read_adc(6)>500)S+=1;
      if(read_adc(5)>500)S+=2;
      if(read_adc(4)>500)S+=4;
      if(read_adc(3)>500)S+=8;
      if(read_adc(2)>600)S+=16;
      if(read_adc(0)>700)S+=32;
      if(read_adc(1)>500)S+=64;

      switch(S){
      case 0b0001000:
      PORTD.0=1;
      PORTD.1=0;
      PORTD.2=0;
      PORTD.3=1;
      OCR1AL=125;
      OCR1BL=125;
      break;

      case 0b0000100:
      PORTD.0=1;
      PORTD.1=0;
```

```
PORTD.2=0;
PORTD.3=1;
OCR1AL=80;
OCR1BL=125;
break;

case 0b0000010:
PORTD.0=1;
PORTD.1=0;
PORTD.2=0;
PORTD.3=1;
OCR1AL=45;
OCR1BL=125;
break;

case 0b0000001:
PORTD.0=1;
PORTD.1=0;
PORTD.2=1;
PORTD.3=0;
OCR1AL=125;
OCR1BL=125;
break;



case 0b0010000:
PORTD.0=1;
PORTD.1=0;
PORTD.2=0;
PORTD.3=1;
OCR1AL=125;
OCR1BL=80;
break;

case 0b0100000:
PORTD.0=1;
PORTD.1=0;
```

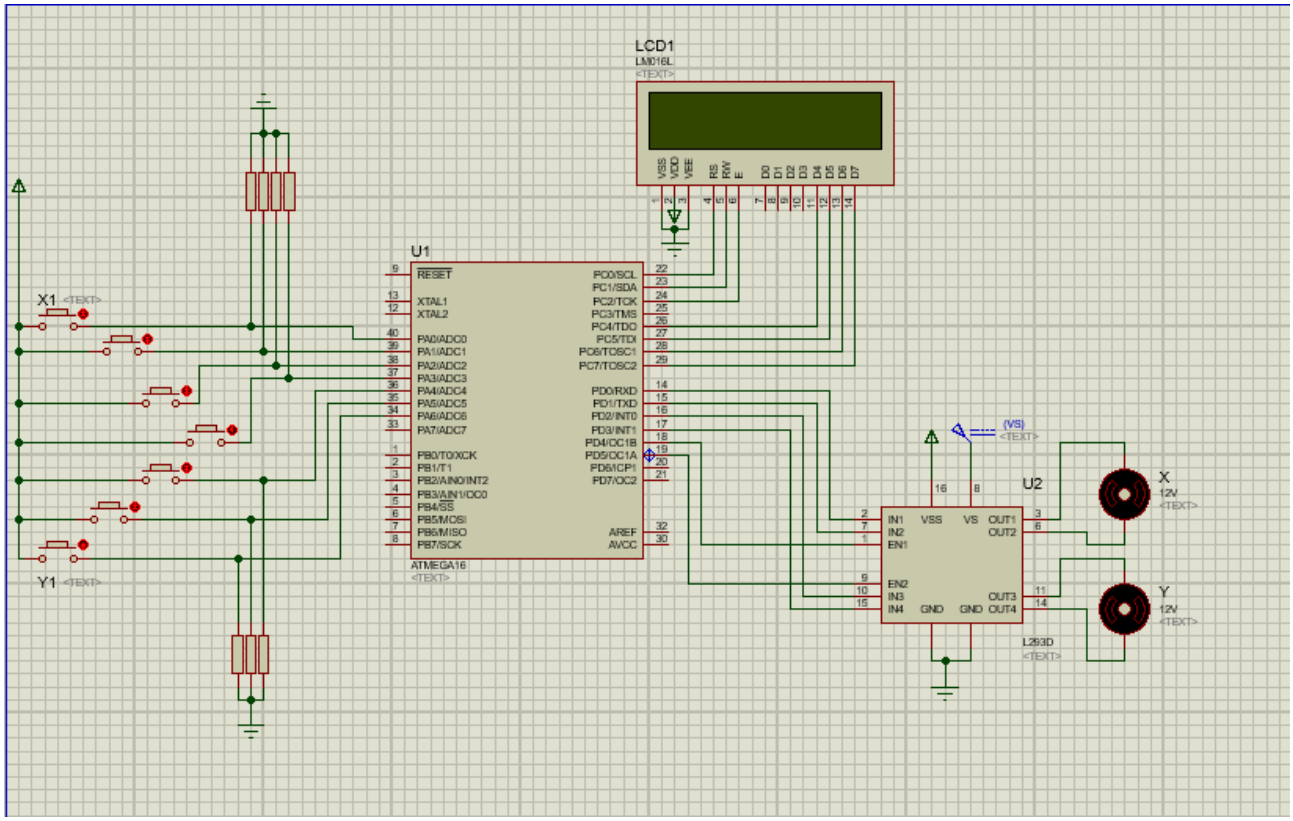
```
        PORTD.2=0;
        PORTD.3=1;
        OCR1AL=125;
        OCR1BL=45;
        break;

        case 0b1000000:
        PORTD.0=0;
        PORTD.1=1;
        PORTD.2=0;
        PORTD.3=1;
        OCR1AL=125;
        OCR1BL=125;
        break;

        default :
        break;
        }

        }
}
```

شمای ربات در پروتئوس :


LCD1
LM016L
<TEXT>
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7
U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA16
<TEXT>
X1
<TEXT>
Y1
<TEXT>
U2
IN1 VSS VS OUT1
IN2 OUT2
EN1
EN2
IN3 OUT3
IN4 GND GND OUT4
L293D
<TEXT>
(VS)
<TEXT>
X
12V
<TEXT>
Y
12V
<TEXT>