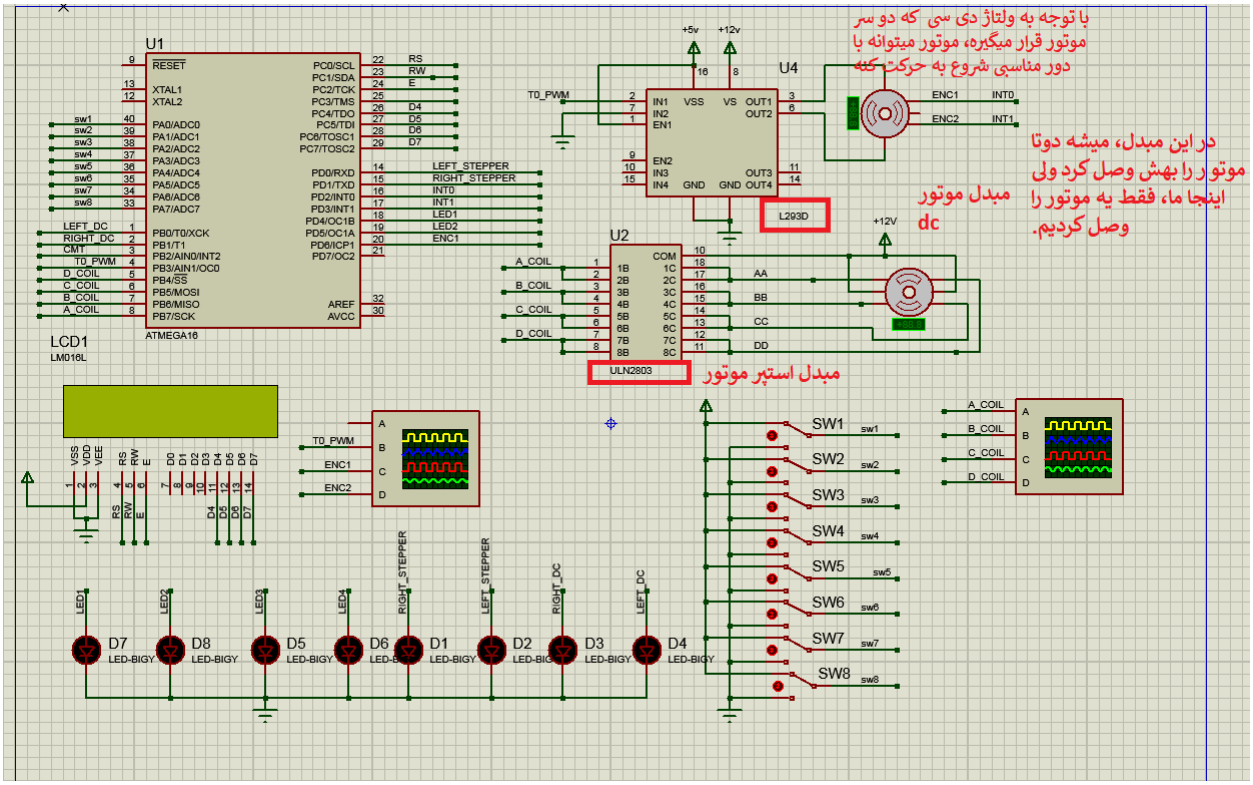
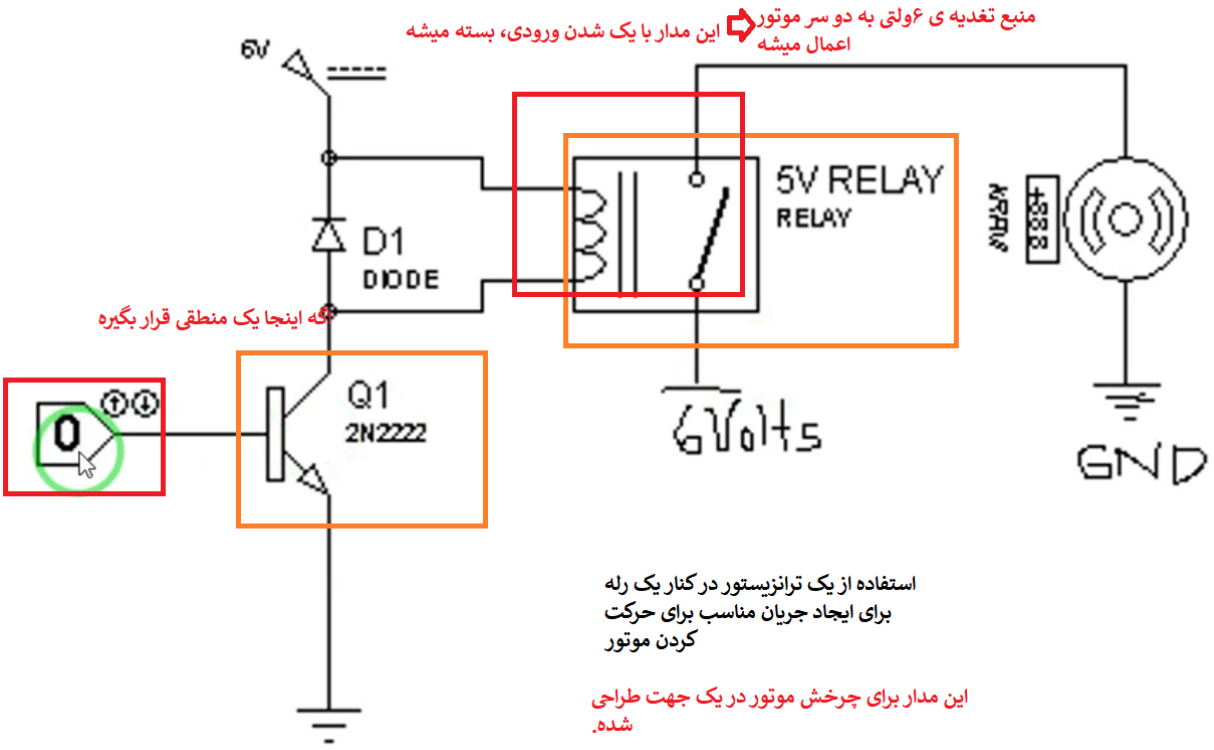
Session5: dc motors, stepper motors

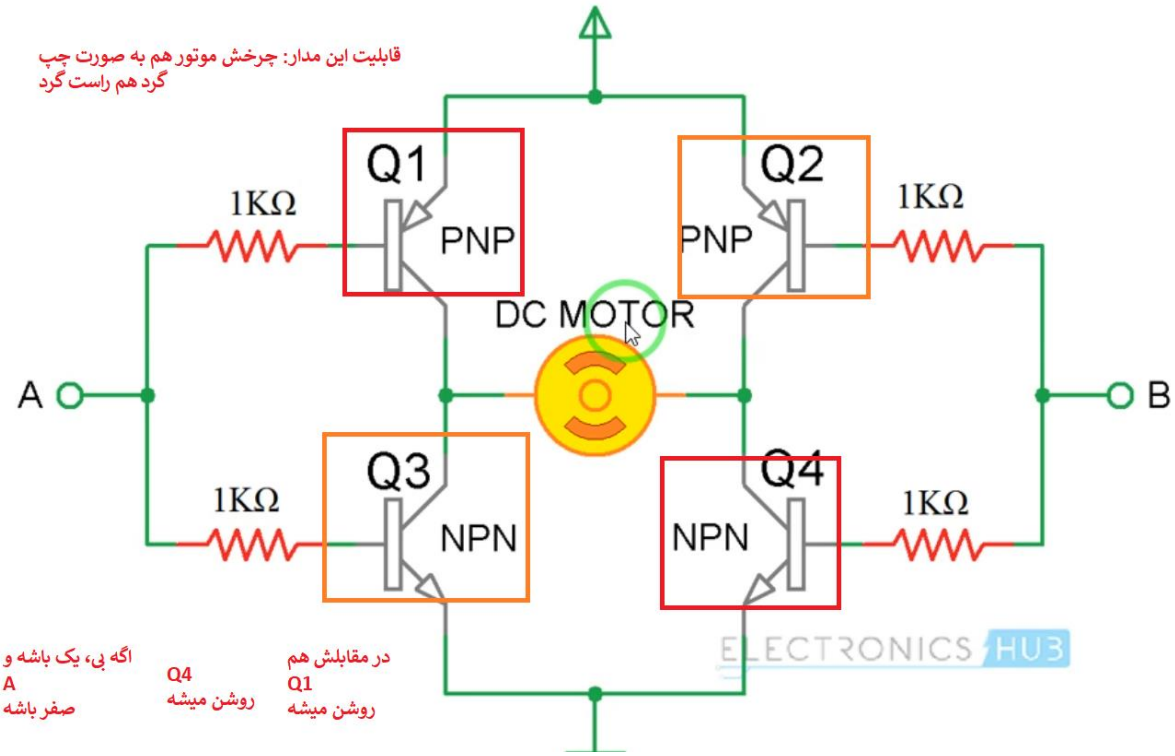
سخت افزار این جلسه:



تامیین منبع جریان یا ولتاژ برای یک موتور با یک میکروکنترلر، به تنهایی نمیتوانیم جریان برای مورد نیاز برای موتور را تامیین کنیم.

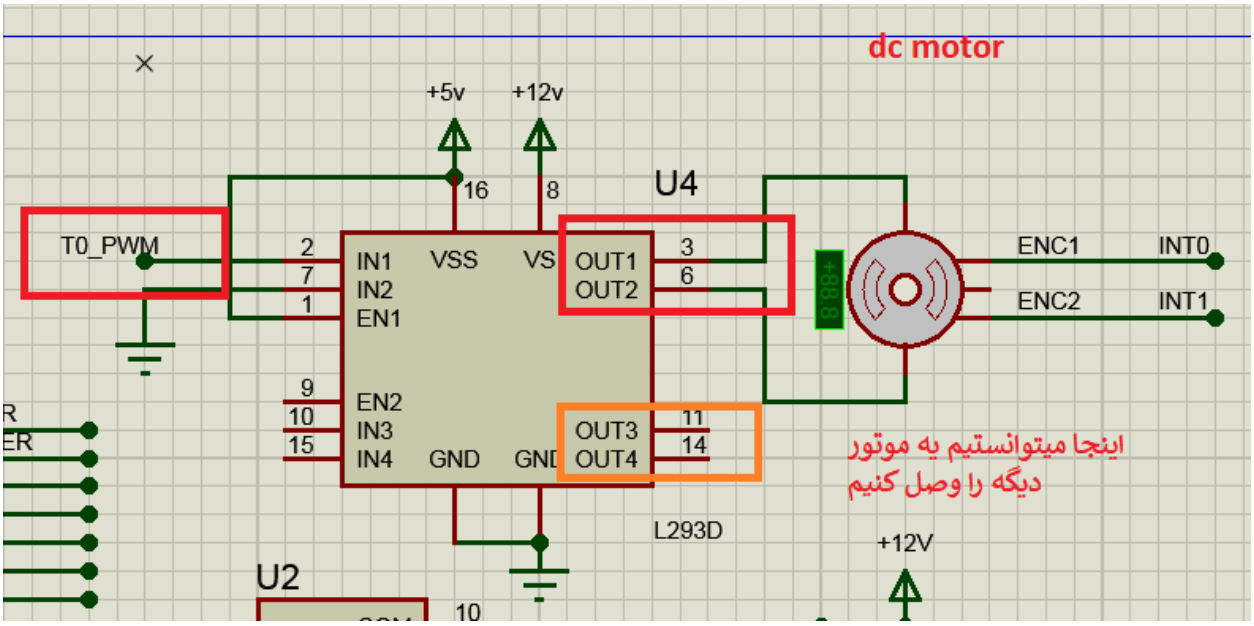
از یک سری مدارهایی کمک میگیریم که بتواند جریان را برای موتور، تامیین کنه.



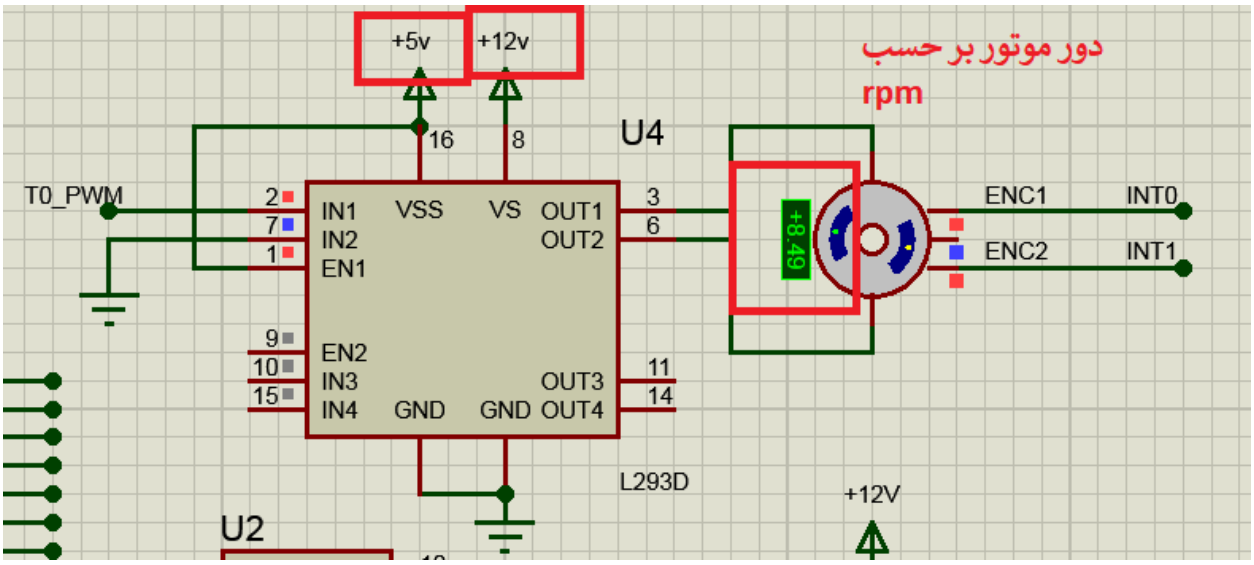


درایور موتور dc : L293D

درایور استپر موتور: ULN2803
با توجه به ولتاژ دی سی DC که دو سرموتور قرار میگیره، موتور میتوانه با دور مناسبی، حرکت کنه.
پس اگه سیگنالی که به دوسر موتور اعمال میکنیم دارای ولتاژdc متغیری باشه میتوانیم موتور را با دور های مختلفی، به حرکت دربیاریم.
به کمک یک موج PWM میتوانیم این کار را بکنیم. چون میدانیم با تغییر duty cycle موج های PWM ، سطح dc که در خروجی ظاهر میشه، تغییر خواهد کرد.
پس اگر یک موجPWM با duty cycle = 20% داشته باشیم و در پورت های OUT1,OUT2 در مبدل L293D یک موج PWM با دامنه ی صفر تا ۵ ولت در خروجی داشته باشیم => در خروجی، یک ولت دی سی خواهیم داشت.
اگر duty cycle = 80% باشد، در خروجی یک ولتاژ دی سی برابر ۴ولت، خواهیم داشت.
چون duty cycle متفاوت میشه، ولتاژ دوسر موتور تغییر میکنه => دور موتور هم تغییر میکنه.



دور موتور برحسب rpm است یعنی اگر دور موتور 157 باشه، یعنی تعداد ۱۵۷ دور در دقیقه را موتور انجام میده.



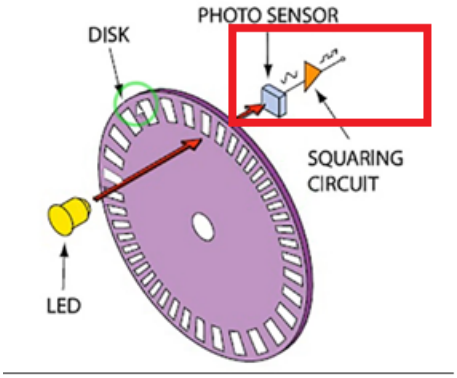
سوال مهم

۱. چطوری دور موتور dc را اندازه بگیریم؟

۲. چطوری جهت چرخش موتور dc را اندازه بگیریم؟

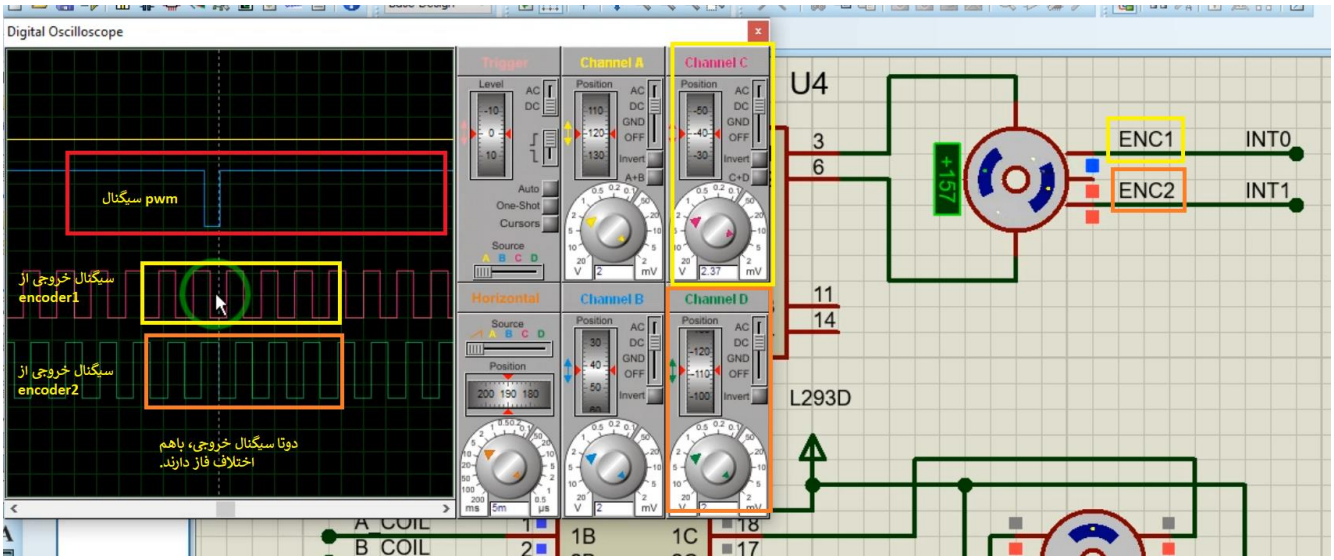
از یک صفحه ی لغزان استفاده میکنند و زمانی که موتور حرکت میکنه، صفحه ی لغزان هم میچرخه و در دو طرف این صفحه ی لغزان از یک لیزر یا LED استفاده کردند و زمانی که صفحه میچرخه و از روزنه ها، نور عبور میکنه، عبور نور توسط یک گیرنده شناسایی میشه و خروجی ای به شکل یک موج مربعی تهیه میکنه.

اگه دو سری LED روبروی صفحه ی لغزان باشه، با توجه به جهت چرخش dc ، سیگنال هایی که از این LEDها خارج میشوند، یک تقدم و تاخری باهمدیگر دارند. به کمک اختلاف فاز این دوتا سیگنال میتوانیم جهت چرخش موتور را پیداکنیم.

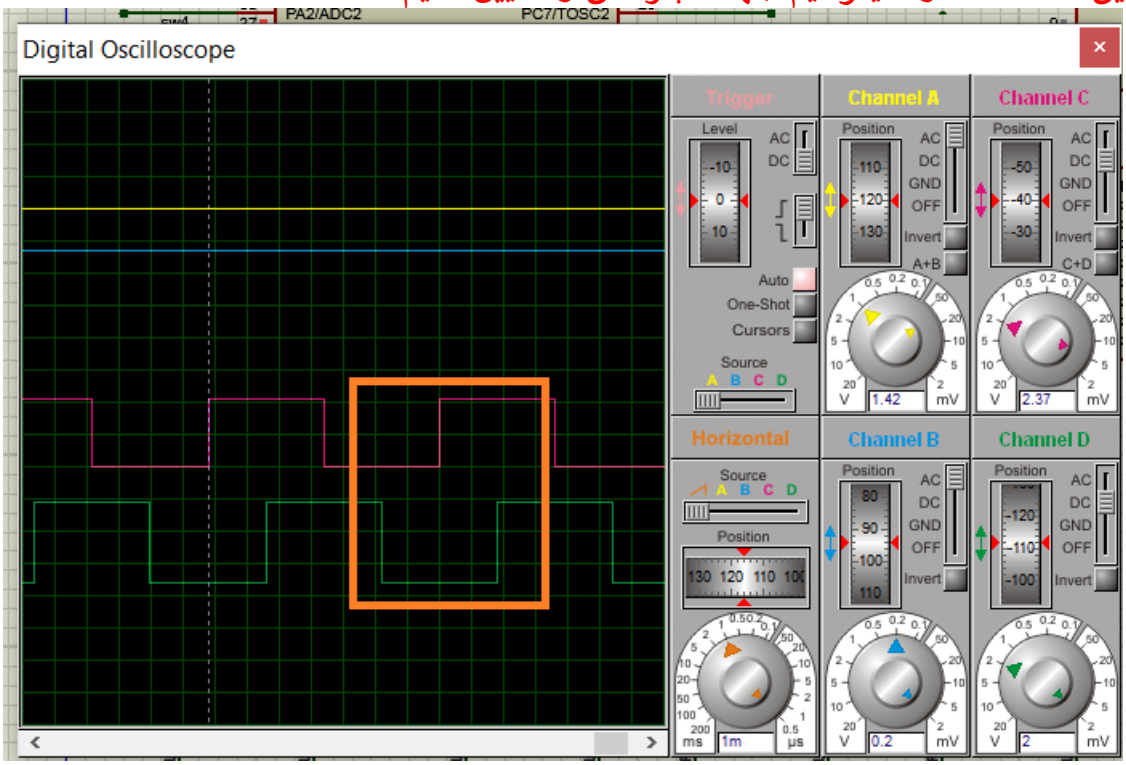


با توجه به دوره ی تناوبی که توسط پالس های مربعی خروجی ایجاد میشه، میتوانیم دور موتور دی سی را با درنظر گرفتن تعداد روزنه ها روی صفحه لغزان، محاسبه کنیم.

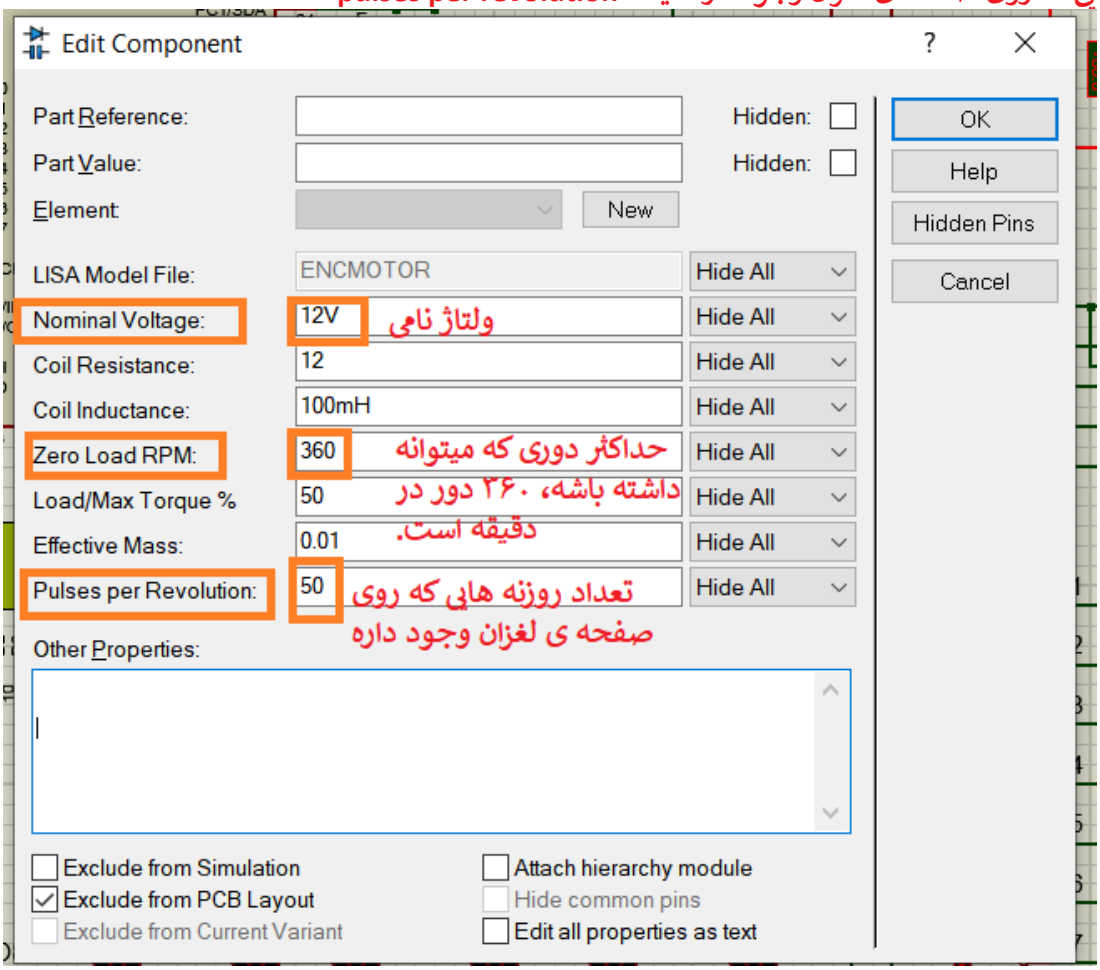
با توجه به دوره ی تناوبی که توسط پالس های مربعی خروجی ایجاد میشه، میتوانیم دور موتور دی سی را با درنظر گرفتن تعداد روزنه ها روی صفحه لغزان، محاسبه کنیم.



اختلاف فازهای سیگنال های خروجی از encoder
به کمک این اختلاف فاز، میتوانیم جهت چرخش را تعیین کنیم.

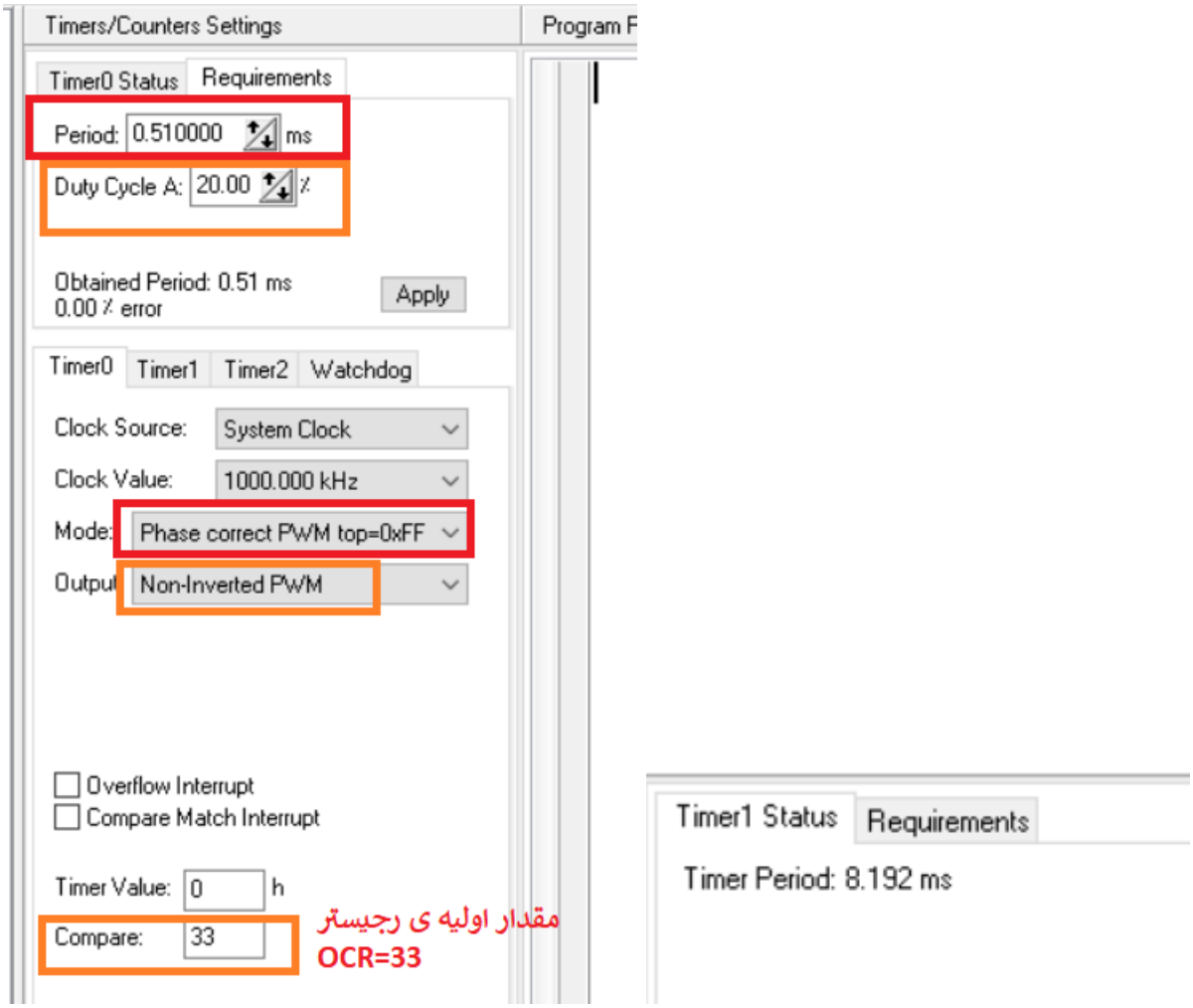


زمانی که سیگنال کانال سی (صورتی رنگ) high است و سیگنال کانال دی، low است نشان میده سیگنال صورتی، زودتر اتفاق افتاده و میتوانیم براش جهت راست گرد را درنظر بگیریم. (ساعتگرد)
اگه ابتدا سیگنال سبز برابر یک شد و سیگنال صورتی برابر صفر باشه، جهت چپ گرد میشه. (پادساعتگرد)
اگه روی موتور دی سی کلیک کنی
تعداد روزنه هایی که روی صفحه ی لغزان وجود داره میشه pulses per revolution



برای ایجاد یک پالس pwm میتوانیم از تایمر های صفر و یک و دو استفاده کنیم.
ولی چون میخاهیم دور موتور دی سی را هم اندازه بگیریم به کمک وقفه ی input capture باید این کار را کنیم => بهتره تایمر ۱ را برای اندازه گیری دور موتور، نگهداریم که وقفه ی input capture داره.
یه پروژه جدید ایجاد میکنیم که از تایمر صفر به عنوان تولید کننده ی موج pwm استفاده میکنیم.

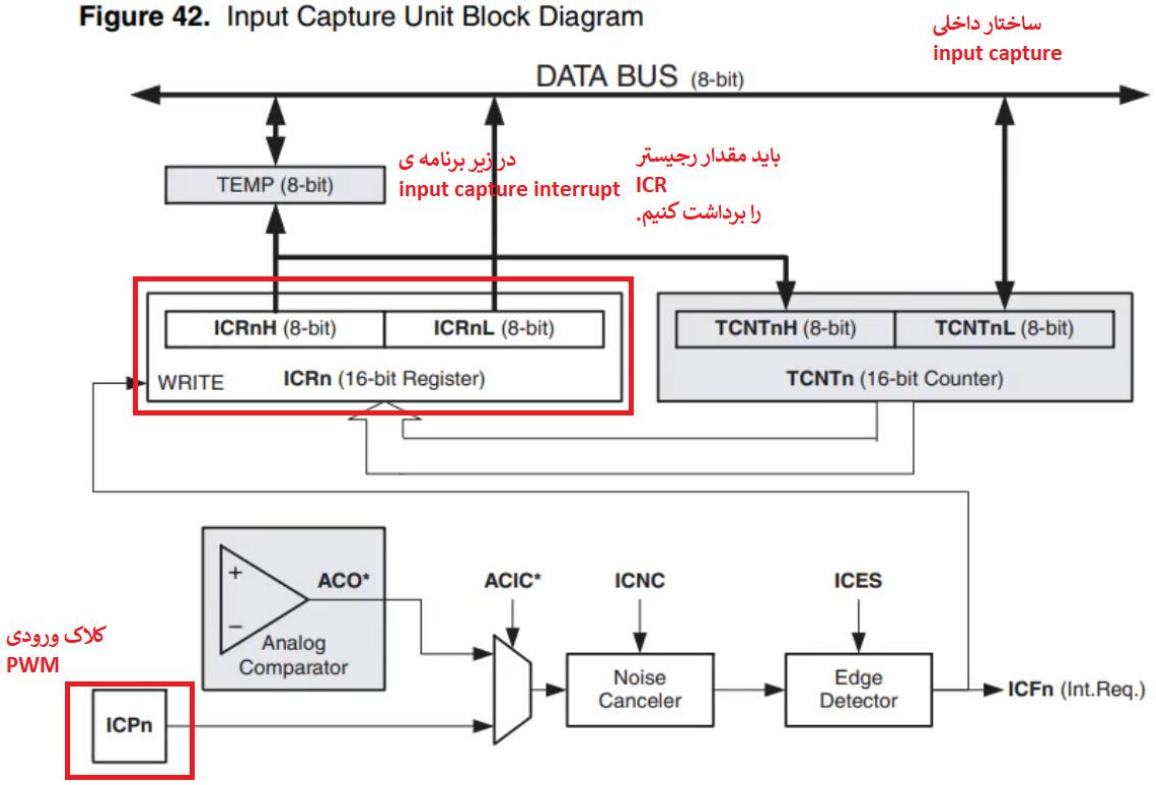
پدیده هایی که زمانشون کمتر از ۸ میلی ثانیه هستند را به کمک تایمر یک با وقفه ی INPUT CAPTURE میشه شناسایی کرد.



استفاده از وقفه ی input capture در تایمر یک
برای اندازه گیری دورموتور دی سی

به بعضی از پدیده هایی که بیرون از میکرو اتفاق میفته، برچسب زمانی اختصاص میدهیم و با مقایسه ی این برچسب های زمانی، تایمی که اون اتفاق افتاده را شناسایی میکنیم.
سوال: چرا باید مقدار رجیستر ICR را به محض رخ دادن وقفه ی INPUT CAPTURE برداشت کنیم؟

Figure 42. Input Capture Unit Block Diagram



ورودی:ICPn است که یک پالس مربعی است و لبه ی بالارونده و پایین رونده داره از طریق مدارها عبور میکنه و بعد از اینکه نویزش گرفته شد، وارد تشخیص لبه میشه و اگه ما از قبل، این را برای لبه ی بالارونده تنظیم کرده باشیم، به محض مشاهده ی لبه ی بالارونده، فلگ را فعال میکنه.
با فعال شدن فلگ، مقدار تایمر کانتر را در رجیستر ICR ذخیره میکنه.
از قبل هم این تایمر کانتر در مد نرمال، در حال شمارش است یعنی از صفر تا 0XFFFF داره میشماره.
به محض اینکه فلگ ICFn فعال شد، محتوای تایمر کانتر، هرچی که باشه داخل رجیستر ICR کپی میشه و اینتراپت input capture رخ میده.

ما باید در زیربرنامه ی اینتراپت، باید مقدار رجیستر ICR را برداشت کنیم.
چرا؟ چون اگه این کار را نکنیم، ممکن است یه لبه ی دیگه ای از پالس ورودی شناسایی بشه و مجددا، مقدار تایمر کانتر در رجیستر ICR کپی میشه و مقدار قبلی را از دست میدهیم.
پس به محض اینکه وقفه input capture اتفاق میفته باید مقدار رجیستر ICR را برداشت میکنیم.

نحوه ی برداشت از این رجیستر

Atmega16 یک میکرو ۸بیتی است و دیتاباسش، فقط میتوانه ۸ بیت را منتقل کنه پس در حالتی که ما یک رجیستر ۱۶ بیتی داریم، برداشت اطلاعات را باید در دو گام انجام بدیم.
اول مقدار ICR1L را (ICR1 LOW) برداشت میکنیم.
در مرحله ی بعدی ICR1H را میخوانیم.
در فاصله ای که ما داریم مقدار ICR1L را میخانیم و روی باس میگذاریم،ICR1H در رجیستر TEMP که ۸بیتی هست، ذخیره میشه و در گام بعدی، مقدار این رجیستر را روی باس میگذاریم.

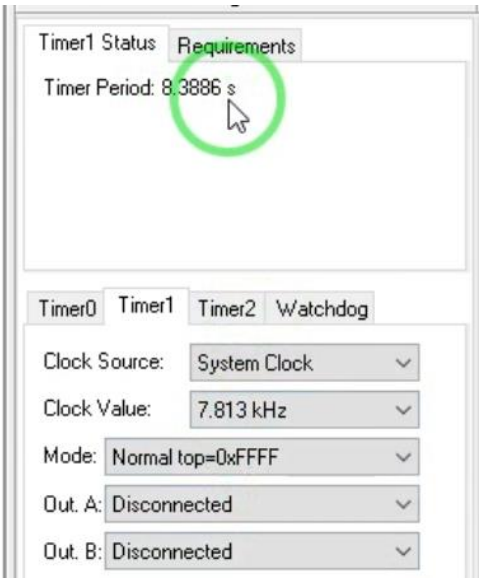
خواندن در ICRها

پس اول مقدار ICR1L را میخانیم بعد ICR1H

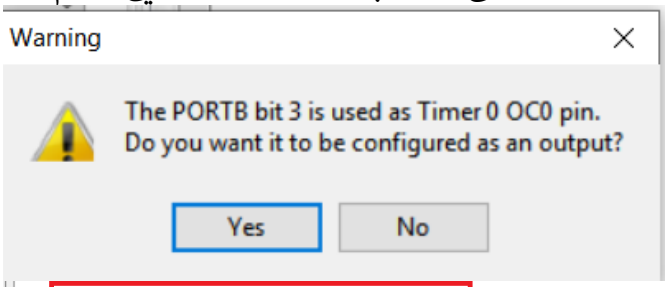
اگه بخاهیم مقدار اولیه به این ICRها بدیم برعکس است.

نوشتن در ICRها

اول ICR1H مقداردهی میشه بعد ICR1L
یعنی مقدار ICR1H که روی باس قرار داره در کلاک اول توی رجیستر TEMP ریخته میشه، در کلاک بعدی مقدار تمپ روی ICR1H قرار میگیره مقدار ICR1L هم از روی باس خوانده میشه.
اگه برای برچسب زمانی زدن، به زمان بیشتری نیاز داریم میتوانیم فرکانس میکرو را کم کنیم مثل زیرکه فرکانس را در پایین تر مقدار ممکن گذاشتیم.
که به ما دوره تناوب ۸ ثانیه را میده.اگه زمان اندازه گیری ما بیش از ۸ثانیه است،از وقفه ی INPUT CAPTURE نمیتوانیم استفاده کنیم.



از انجایی که در پروتئوس فقط فرکانس های ۸ و ۱ مگاهرتز را میشه شبیه سازی کرد => ما همان ۸مگا را انتخاب میکنیم تا بتوانیم در بازه های ۸میلی ثانیه پدیده هارا شناسایی کنیم.




```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 8000.000 kHz
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Rising Edge
// Timer Period: 8.192 ms
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: On
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
TCCR1B=(0<<ICNC1) | (1<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) | (1<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
```

مقداردهی اولیه کردن رجیستر ICR (نوشتن در رجیستر)

اول توی ICR1H نوشته

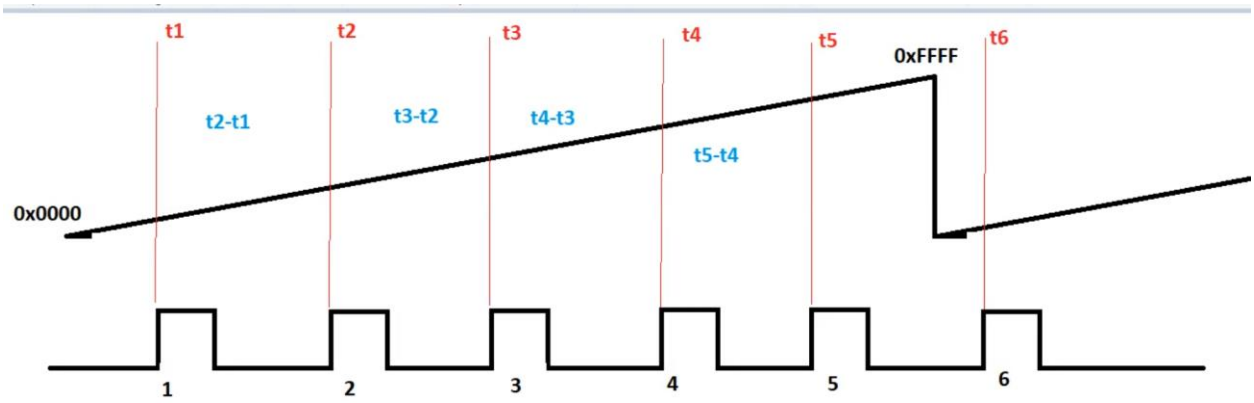
بعد توی ICR1L

```
// Timer1 input capture interrupt service routine
interrupt [TIM1_CAPT] void timer1_capt_isr(void)
{
// Place your code here

}
```

اولین کاری که در اینتراپت انجام میدهیم: خواندن ICR

```
// Timer1 input capture interrupt service routine
interrupt [TIM1_CAPT] void timer1_capt_isr(void)
{
int dataL,dataH,ICR_data;
// Place your code here
//READ ICR REGISTER
dataL = ICR1L;          //first we read Low
dataH =ICR1H;           //second, we read high
ICR_data = dataH <<8 | dataL; //get content of ICR
}
```

پیدا کردن دوره تناوب پالس زیر به کمک وقفه ی اینپوت کپچر

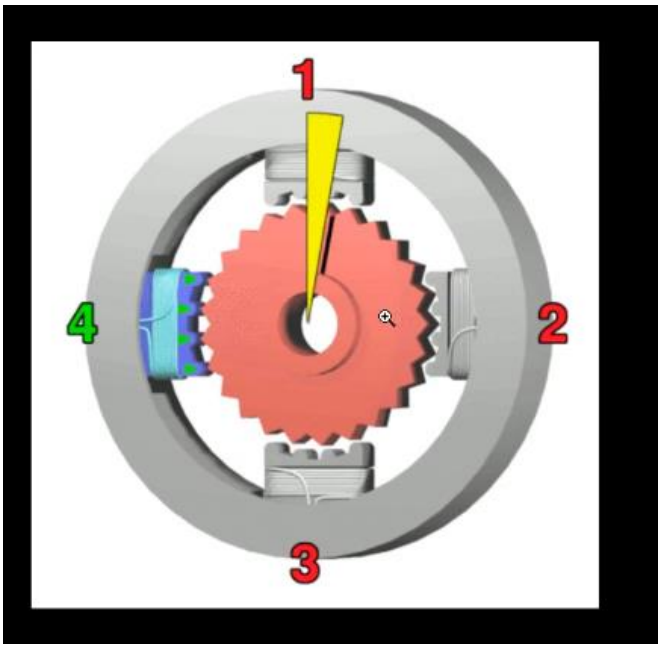


تایمر از صفر شروع به شمارش میکنه تا 0XFFFF و بعدش اورفلو رخ میده.
اگر در این بازه، ما تعدادی پالس را در پایه ی ICP دریافت کنیم، در لحظه ی ۱، مقدار t1 را از تایمر کانتر برداشت میکنه و داخل ICR قرار میده که ما در زیربرنامه ی اینپوت کپچر میتوانیم مقداری که در رجیستر ICR قرار گرفته را برداشت کنیم.
در لحظه ی T2 هم میتوانیم مقداری که دررجیستر قرار گرفته را برداشت کنیم.
اگر ما اختلاف مقدار TIMER COUNTER ها را در لحظه ای که اینتراپت اینپوت کپچر رخ داده از هم دیگر مقایسه کنیم => با توجه به تعداد پالس ها و فرکانس کاری تایمر، میتوانیم دوره تناوب پالس ورودی را بدست بیاریم. (دوره تناوب اولین بازه)
نکته:اگر در لحظه ی ۵، مقدار T5 برداشت بشه و در لحظه ی ۶، T6 و بین این دو زمان، اینتراپت مربوط به اورفلوی تایمر رخ بده در این حالت نمیتوانیم مقدار t5 را از t6 کم کنیم چون t6 کمتر از t5 است.

بهترین راه حل: زمانی که میخاهیم نمونه برداری کنیم، بیایم وضعیت اورفلوی تایمر را بررسی کنیم. اگر اینتراپت رخ داده باشه، ما از این داده نمیتوانیم برای محاسبات ریاضی استفاده کنیم و باید منتظر داده های بعدی باشیم اگر هم رخ نداده باشه که میتوانیم.

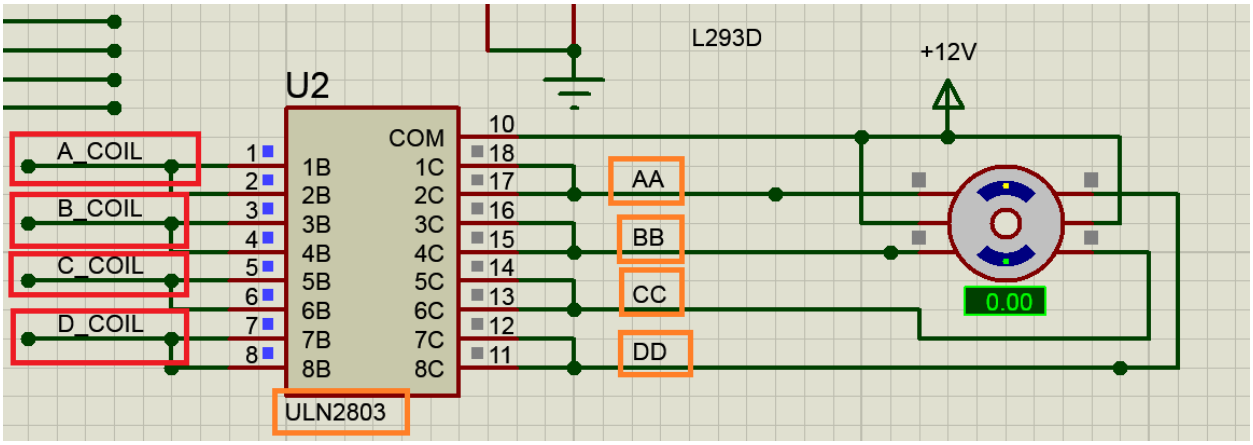
Stepper motors

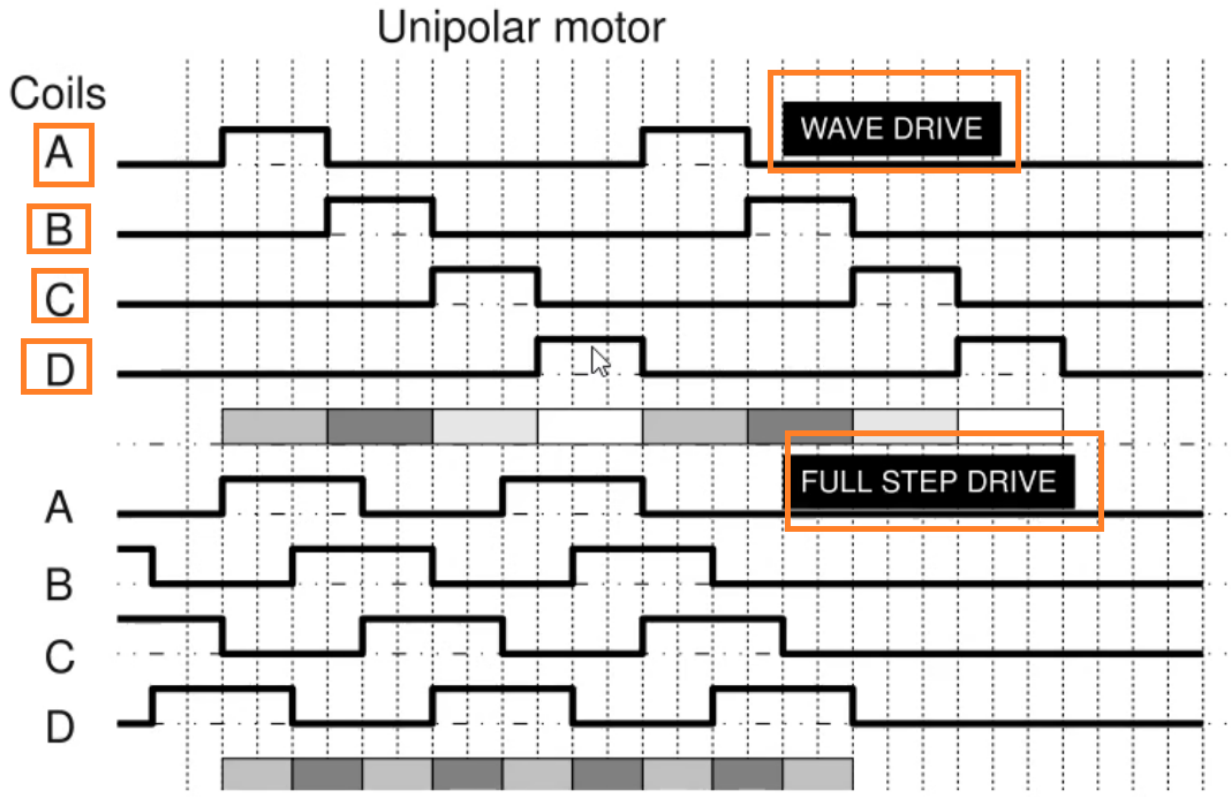
دارای تعدادی سیم پیچ هستند که سیگنالی که روی این سیم پیچ ها قرار میگیره،باعث حرکت روتور خواهد شد.
این روتور میتوانه در جهت راست گرد یا چپ گرد حرکت کنه.
هربار که یکی از این سیمپیچ ها فعال میشه، استپر موتور به اندازه ی یک گام حرکت میکنه.



منظور از گام استپر موتور؟
کمترین میزانی که استپر موتور میتوانه حرکت کنه براساس سیگنالی که بهش وارد میشه.
مثلا اگه یه استپرموتوری، با اعمال هر پالسی بتواند یک درجه تغییر وضعیت بده => برای چرخش کاملش باید ۳۶۰تا پالس بهش اعمال بشه تا بتواند یک دور کامل بزنه

برای اینکه جریان خروجی برای استپرموتور بیشتر بشه، ورودی هاوخروجی های مبدل ULN2803 را به صورت دوتایی وصل کردیم.

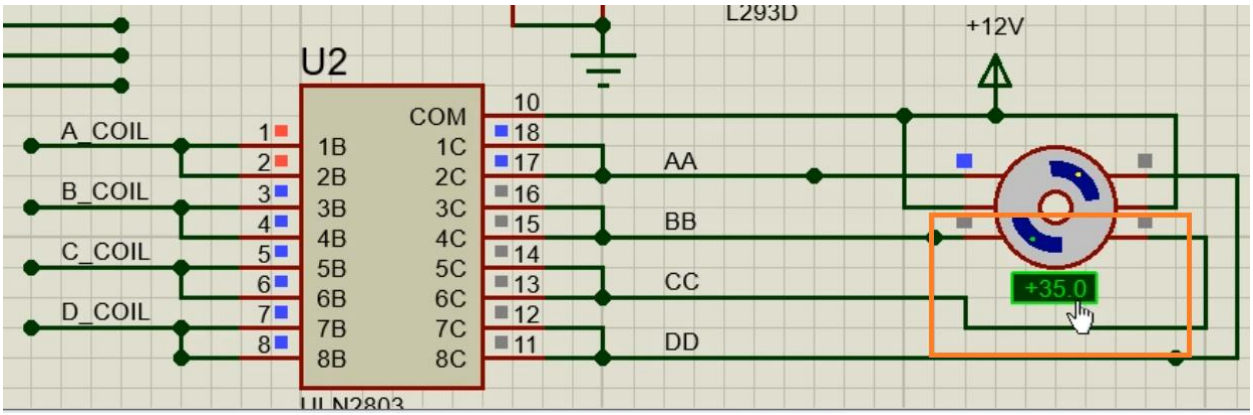




برای درایور استپر موتور:
میتوانیم به دوحالت بالا، بهش ورودی بدیم. اگر اول سیمپیچ A بعد B بعد C و بعد D را مقداردهی کنیم، استپر موتور میتوانه به صورت گام ب گام حرکت کنه. مثلا اگه بهش ۴تا پالس اعمال کردیم انتظار داریم به اندازه ی ۴گام جابه جا بشه.
در حالت اول: در هرلحظه از زمان،فقط یکی از سیمپیچ ها فعال است و تغذیه داره برای همین بهش درایور تکفازی میگن که باعث میشه نیم گام خطا داشته باشیم.

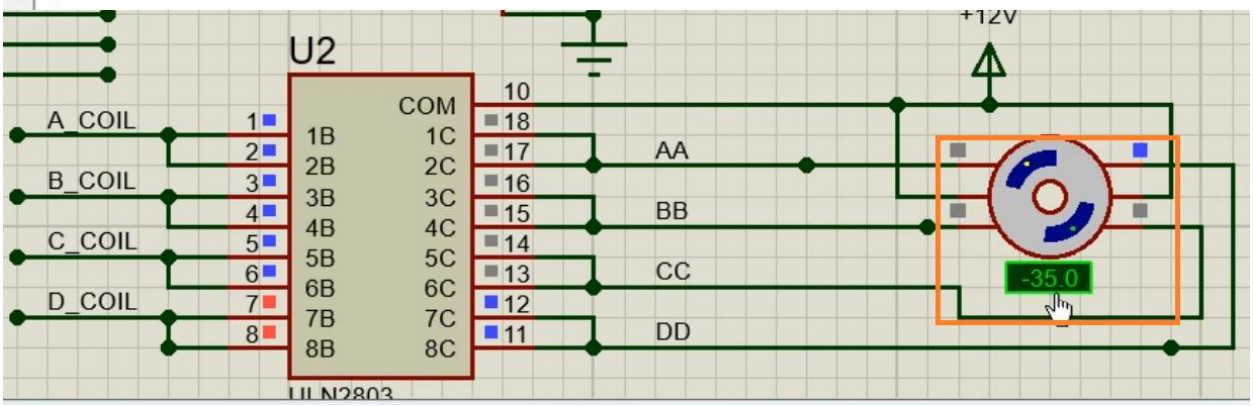
در برنامه ی زیر، به ترتیب سیمپیچ های A,B,C,D را مقداردهی کردیم. بین هرکدام یه تاخیری گذاشتیم تا موتور بتوانه جابهجا بشه.
اینجا ما ۴تا پالس بهش اعمال کردیم ولی استپر موتور در زاویه ی ۳۵ درجه قرار میگیره. (گام استپرموتور: ۱۰ درجه است) الان ما به اندازه ی نیم گام هربار خطا داریم یعنی به جای ۴۰ درجه که باید نشان بده ۳۵درجه نشان میده در نهایت.
حرکت موتور به صورت راست گرد یا ساعتگرد

```
void singlephas(void)
{
PORTB=0x10;
delay_ms(a);
PORTB=0x20;
delay_ms(a);
PORTB=0x40;
delay_ms(a);
PORTB=0x80;
delay_ms(a);
}
```

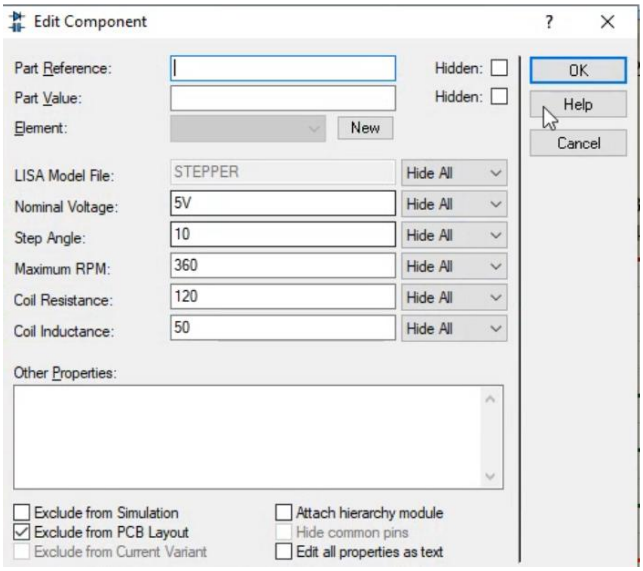


حرکت موتور به صورت چپ گرد یا پادساعتگرد

```
void singlephas_rev(void)
{
PORTB=0x80;
delay_ms(a);
PORTB=0x40;
delay_ms(a);
PORTB=0x20;
delay_ms(a);
PORTB=0x10;
delay_ms(a);
}
```



در زاویه ی 35- درجه می ایستد.
که بازهم نیم گام خطا داریم.
اعداد ۵و۱۵و۲۵و۳۵و۴۵و... را با این روش میشه پیاده سازی کرد.
برای دیدن تنظیمات استپرموتور روش کلیک کن توی پروتئوس

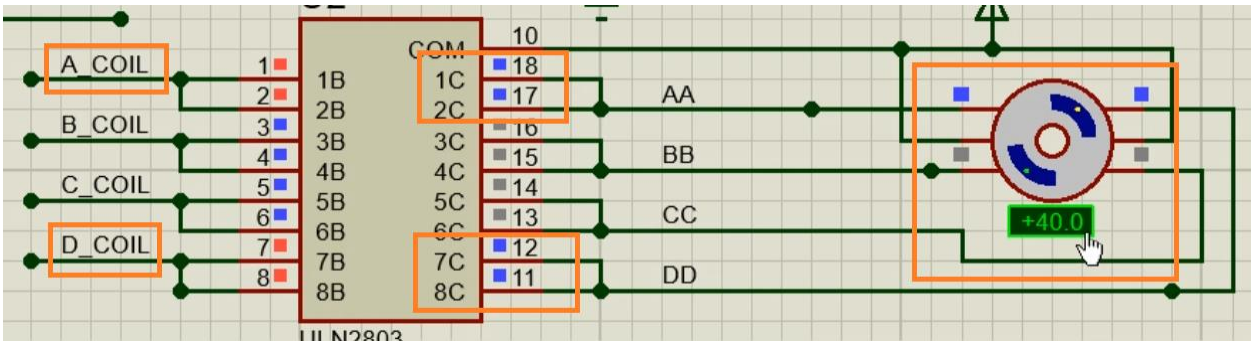


اگه بخاهیم دقیقا هربار به اندازه یک گام حرکت کنیم از شکل دوم استفاده میکنیم.
یعنی باید در هرلحظه دقیقا دوتا از سیمپیچ ها را مقداردهی کنیم.
سیگنال A,B باهم
سیگنال A,D باهم
سیگنال B,C باهم
سیگنال C,D باهم
حرکت در جهت راستگرد
اصطلاحا بهش دوفازی میگویند.

```
void twophas(void)
{
PORTB=0x30;
delay_ms(a);
PORTB=0x60;
delay_ms(a);
PORTB=0xc0;
delay_ms(a);
PORTB=0x90;
delay_ms(a);
}
```

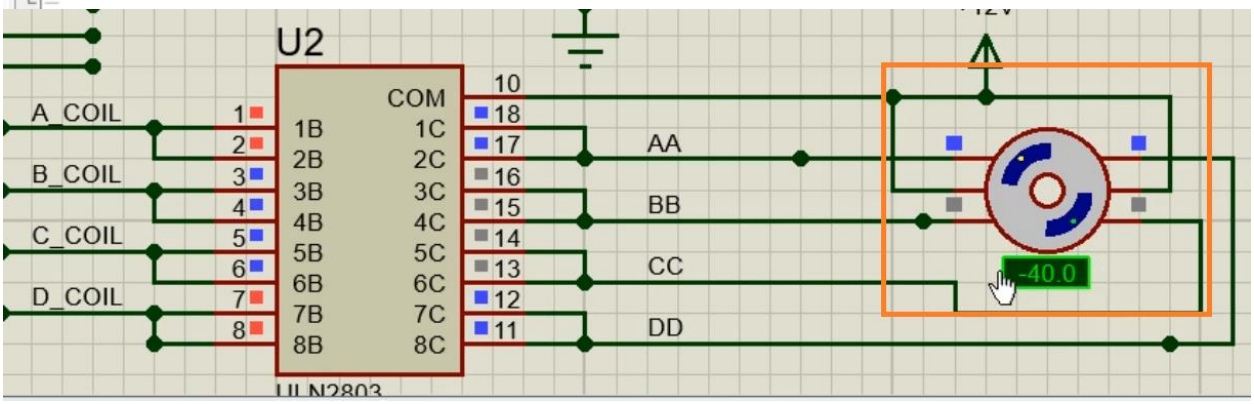
زاویه هایی که درش مکث میکنه:۱۰و۲۰و۳۰و۴۰
ما ۴ تا سری پالس اعمال کردیم و درهرسری استپرموتور، به اندازه ی یک گام حرکت کرد. => اینجا دیگه خطایی نداریم

```
void main(void)
{
DDRB=0xf0;
 twophas();
// twophas_rev();
// singlephas();
// singlephas_rev();


while (1)
    {
    }
}
```



دوفازی به صورت معکوس

```
void twophas_rev(void)
{

PORTB=0xC0;
delay_ms(a);
PORTB=0x60;
delay_ms(a);
PORTB=0x30;
delay_ms(a);
PORTB=0x90;
delay_ms(a);
}
```

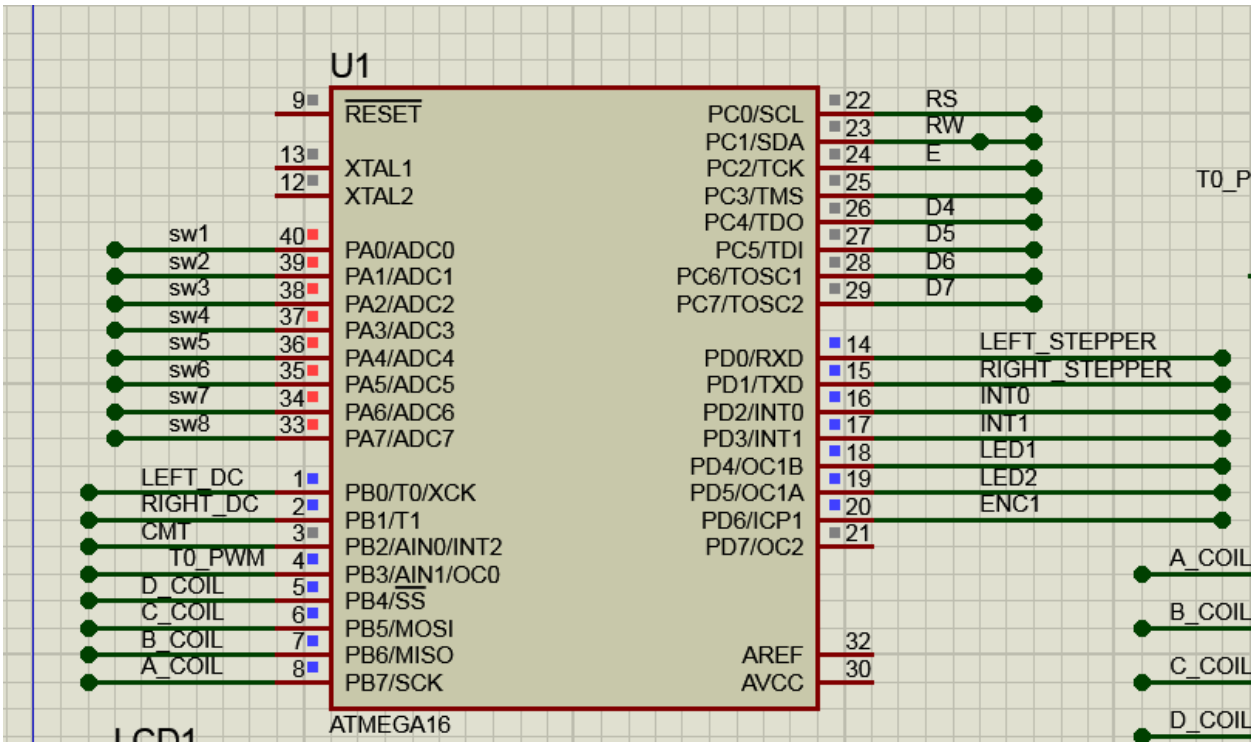


40- نشان دهنده ی درجه ای است که شافت موتور قرار میگیره

نکته:بین مقداردهی هرکدام از سیمپیچ ها باید یه زمانی را درنظر بگیریم تا استپرموتور زمان کافی برای حرکت کردن و رسیدن به اون نقطه را داشته باشه.

```
#include <io.h>
#include <delay.h>
#define a 1000
void singlephas_rev(void);
void singlephas(void);
void twophas_rev(void);
void twophas(void);
```

از تاخیر های میلی ثانیه ای استفاده کردیم و درواقع بین هر گام یک ثانیه تاخیر ایجاد کردیم.



هرکدوم از پایه های این میکرو چطوری باید در ابتدا تعریف بشوند؟ خروجی یا ورودی؟

step

speed(rpm)

T=

تعداد پالس در هر دور =360/step

تعداد پالس در سرعت مورد نظر =Speed(rpm)*(360/step)

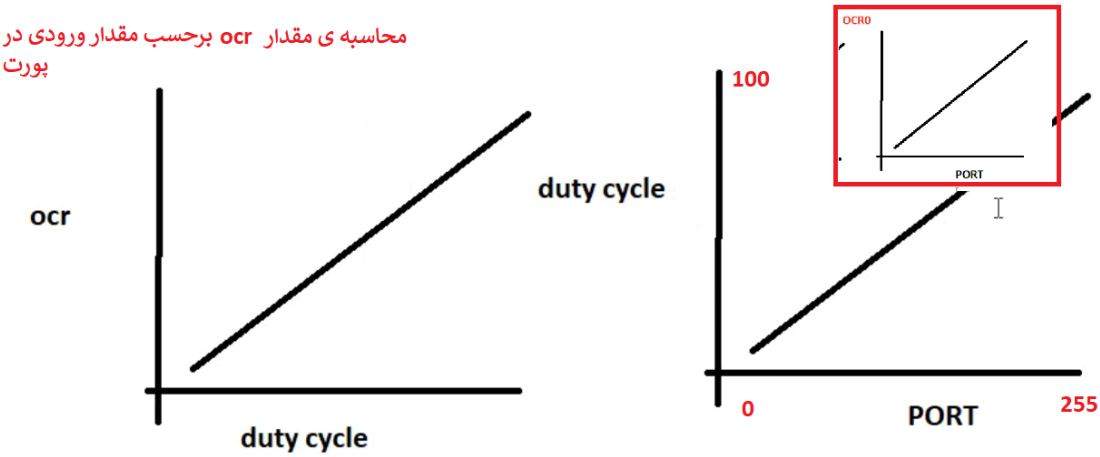
فاصله زمانی بین ارسال پالس = 60/(تعداد پالس در سرعت مورد نظر)

مثال

step=10
Speed(rpm)=360
T= 60/(360*360/10)= 4.7ms

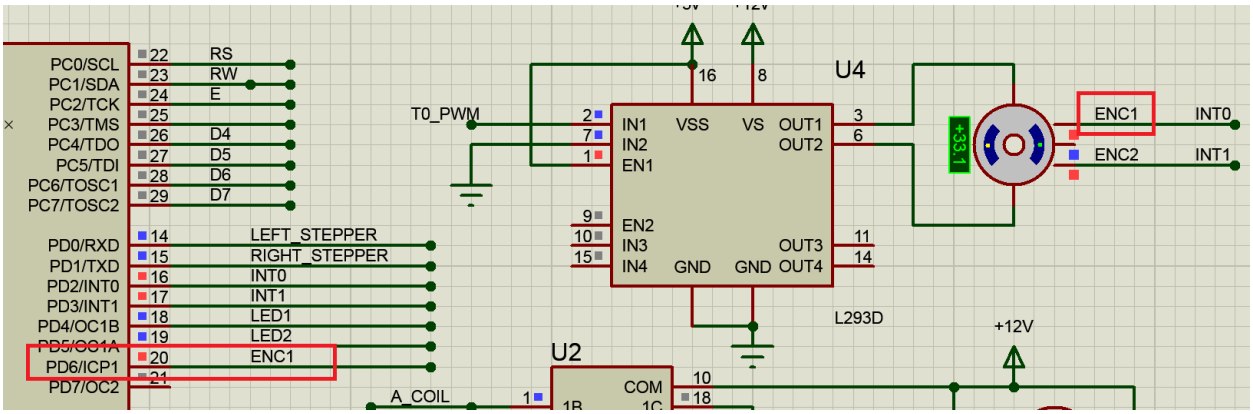
ما گام استپر موتور و سرعتی که میخاهیم به حرکت دربیاد را داریم میخاهیم فاصله ی زمانی بین اعمال سیگنال ها به موتور (بین مقداردهی به سیمپیچ ها) را بدست بیاریم.
سرعت ما برحسب دور بر دقیقه است => باید به ثانیه ببریم.

محاسبه ی مقدار ocr برحسب مقدار ورودی در پورت



تولید یک موج pwm با duty cycle های مختلف توسط تایمر صفر

اگر مقدار پورت تغییر کرد باید مقدار duty cycle هم تغییر کنه.



PD6/ICP1: input capture of timer1