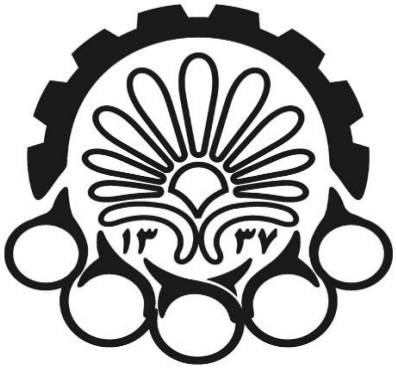


دانشکده مهندسی برق

پروژه ی کنترل صنعتی

استاد درس: دکتر احمد افشار

تدریس یار : مهندس امیرحسن آشنایی

دانشجویان :

محمد امینی 9523009

گلناز بشیریان 9523405

کسری خلفی 9523038

مقدمه :

در این پروژه طراحی کنترلر PID و پیاده سازی آن به صورت نرم افزاری در نرم افزار متلب و شبیه سازی آن و به صورت سخت افزاری جهت کنترل سرعت و موقعیت یک موتور DC انجام شده است.

سخت افزار استفاده شده برای پیاده سازی عملی به صورت زیر است:

- میکروکنترلر STM32f103C8 از خانواده ی ARM
- موتور DC انکودردار – مدل M25N-2 R-14 2241

مشخصات موتور:

Items	Specifications
Rated Voltage	31.0V
Voltage Range	28.0~34.0V
Rated Load	23.4mN·m
No Load Speed	11,000rpm
No Load Current	100mA or less
Starting Torque	68mN·m
Rotation	CW/CCW

- ماژول L298 (جهت راه اندازی موتور و تعیین جهت گردش)
- منبع ولتاژ
- Programmer – ST-LINK Ver2

بخش اول : تنظیم سرعت موتور DC طبق ساختار مرسوم

در این بخش کنترل سرعت یک موتور DC با استفاده از اعمال پالس PWM به آن کنترل شده است. حلقه ی کنترلی در محیط سمولینک متلب شبیه سازی شده است. این عمل به دو روش انجام شده است:

- **روش اول** : در این روش موتور DC مدل 5 hp 240 v 1750 rpm Field : 300 v از مدل های آماده ی سیمولینک استفاده شده است. برای طراحی کنترل PID از روش زیگلر-نیکلز استفاده شده است. به این علت که تابع تبدیل موتوردردسترس نمی باشد باید از تقریب FOPDT استفاده شود. برای این کار ، پاسخ پله موتور را بررسی کرده و با استفاده از روش تانژانت ، مقادیر t_0 و τ محاسبه شده است. مقادیر محاسبه شده به شرح زیر می باشد :

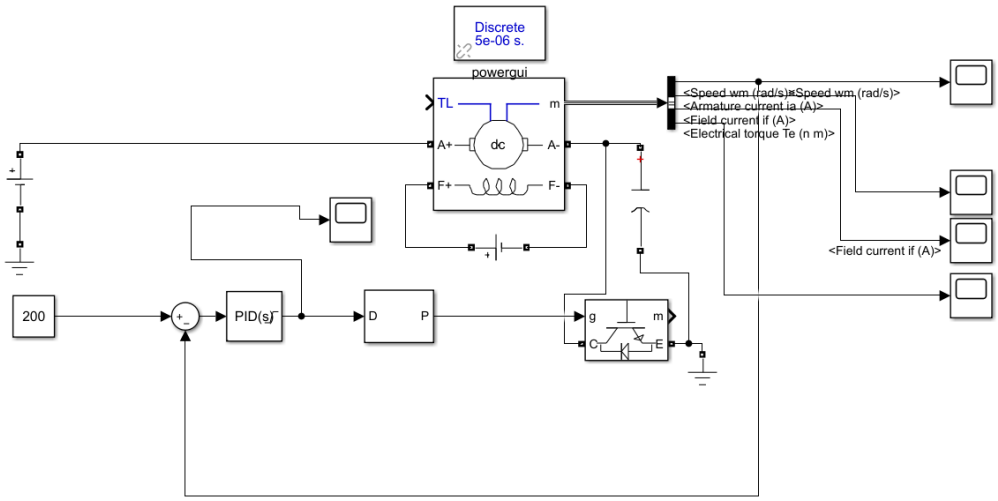
$$\tau = 0.06$$

$$t_0 = 0.05$$

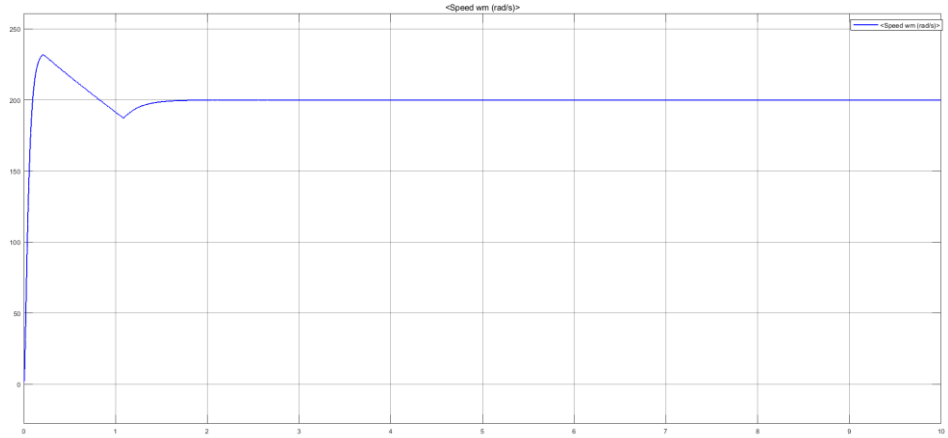
طراحی کنترلر PI با استفاده از جداول زیگلر نیکلز :

$$K_p = \frac{0.9\tau}{t_0} = 1.08$$

$$T_I = 3.33t_0 = 0.1665 \rightarrow K_I = \frac{1}{T_I} = 6.006$$



حلقه کنترل سرعت همراه با کنترلر PI

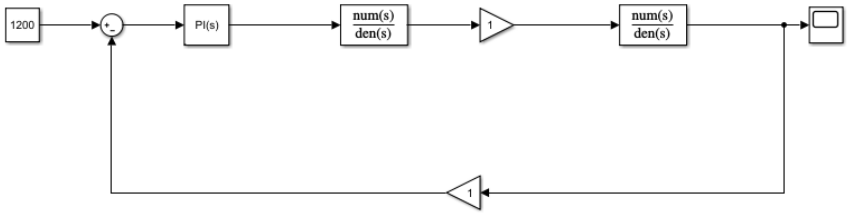


پاسخ حلقه کنترل سرعت موتور برحسب همراه با کنترلر PI طراحی شده

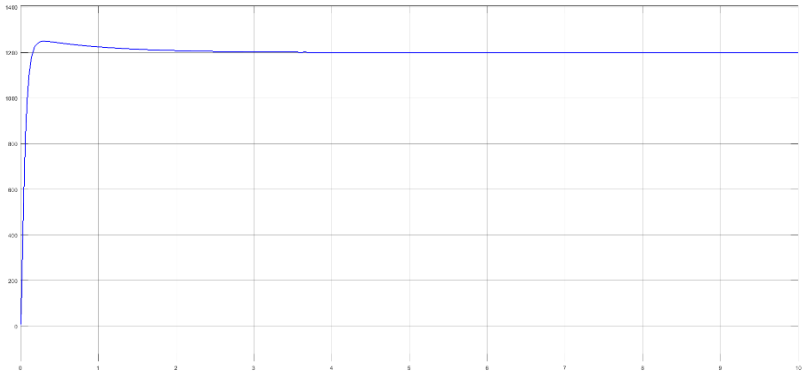
Overshoot : 16%

Settling Time : 2s

- **روش دوم:** در این روش به جای موتور ، تابع تبدیل بدست آمده براساس مشخصات موتور در حلقه کنترل سرعت جایگزین شده است. برای کنترل سرعت از کنترلر PI استفاده شده است.ضرایب کنترل کننده از روش سعی و خطا به دست آمده اند.



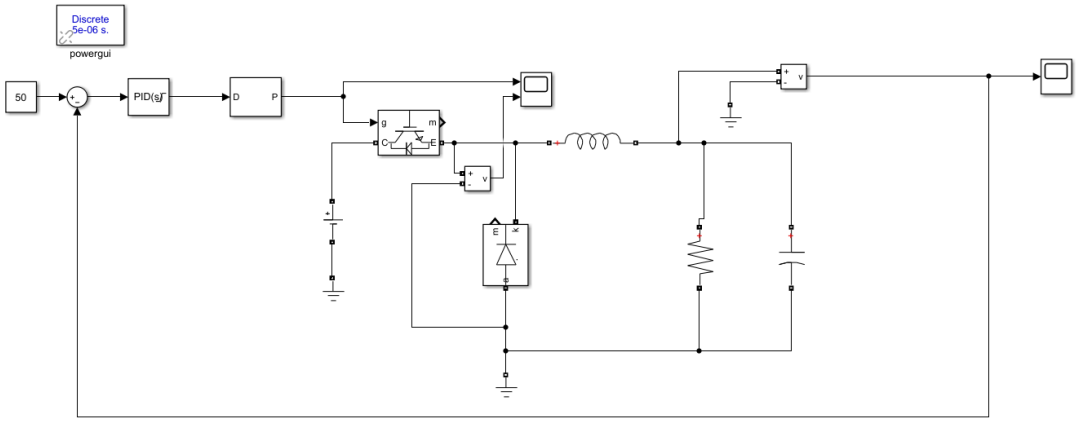
حلقه کنترل سرعت با جایگذاری تابع تبدیل موتور



پاسخ حلقه کنترل سرعت موتور برحسب همراه با کنترلر PI طراحی شده

بخش دوم : طراحی مبدل DC-DC

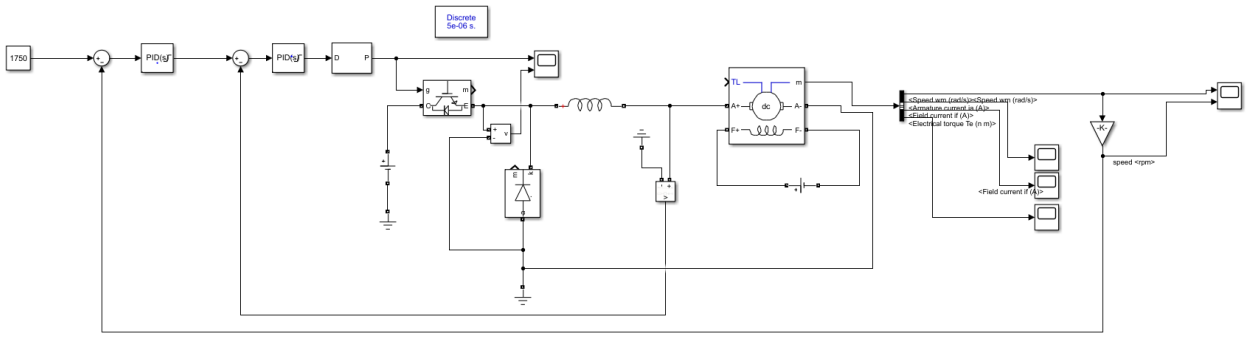
کنترل سرعت موتور از طریق تنظیم ولتاژ پایانه‌های آن انجام می‌شود. تغییر ولتاژ خط DC از طریق یک مبدل Buck ، یکی از روش‌های کنترلی می باشد. در این حالت به جای اعمال پالس PWM ، کنترل سرعت از طریق اعمال ولتاژ آنالوگ به آن انجام می شود.همچنین سبب می شود که ولتاژ های بیش از اندازه به موتور اعمال نگردد. در این بخش حلقه کنترل ولتاژ خروجی مدار BUCK در محیط سیمولینک متلب پیاده سازی و شبیه سازی شده است. منبع ولتاژ در دسترس برای کنترل موتور ، 310 ولت می باشد. برای کنترل ولتاژ از کنترلر PID استفاده شده است.



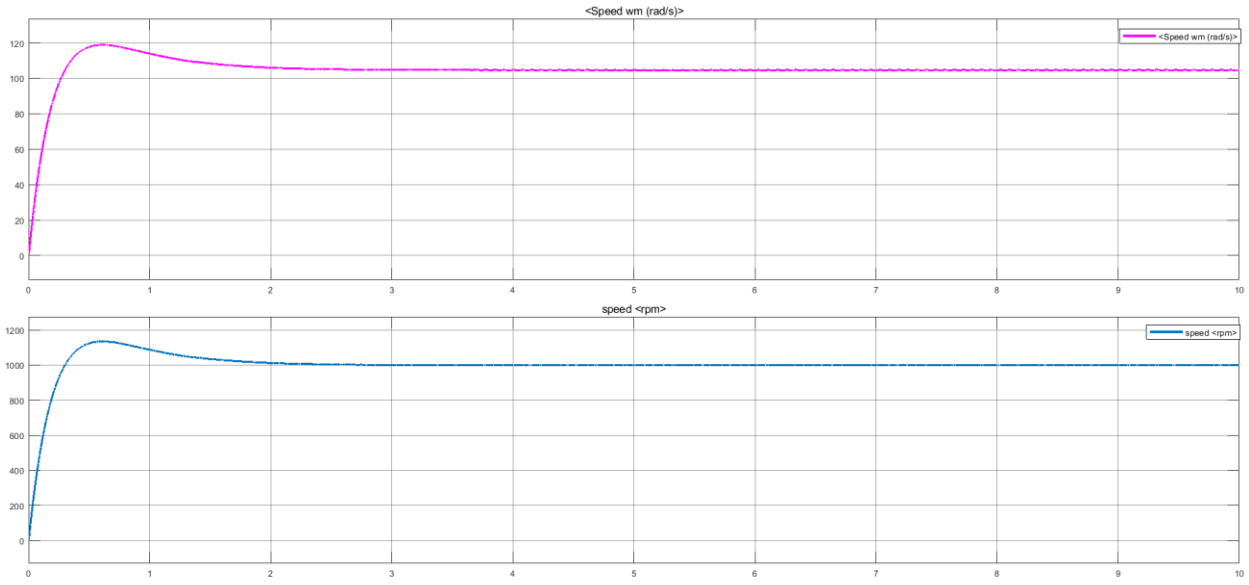
حلقه کنترل ولتاژ مدار BUCK با کنترلر PID

بخش سوم : کنترل موتور DC با استفاده از مبدل طراحی شده

در این بخش کنترل سرعت موتور DC از طریق تغییر ولتاژ خط DC و با بهره‌گیری از BUCK صورت گرفته است. بدین منظور، ابتدا با تشکیل حلقه کنترلی و از طریق اعمال خروجی کنترلر به مبدل DC-DC ، سرعت موتور را تنظیم نمایید. جهت بهبود فرآیند کنترل، ساختار کنترل آبشاری (Cascade) تشکیل شده و با استفاده از دو حلقه کنترلی، سرعت موتور کنترل شده است.حلقه داخلی حلقه کنترل ولتاژ مدار BUCK و حلقه بیرونی حلقه کنترل سرعت می باشد. حلقه کنترل ولتاژ همان حلقه ی طراحی شده در بخش دوم می باشد. ضرایب کنترلر سرعت برای دست یابی به کمترین خطای حالت دائم و بالازدگی و زمان نشست به دست آمده اند.



ساختار کنترل آبشاری جهت کنترل سرعت



پاسخ حلقه کنترل سرعت

بخش چهارم : پیاده سازی سخت افزاری کنترل سرعت و موقعیت موتور DC

1) کنترل سرعت

در این قسمت از سوال به بخش پیاده سازی کنترل سرعت و موقعیت می پردازیم. ضرایب طراحی شده در قسمت قبل با موتوری که در آزمایش عملی استفاده شد متفاوت میباشد در نتیجه از ضرایب بدست آمده در مراحل قبل استفاده نشده است و با توجه به آن که مدل موتور فرق میکند ضرایب جدید برای موتور طراحی شده است .

فرکانس سوچینگ pwm برابر 10 کیلوهرتز است برای محاسبه این مقدار ،مقدار فرکانس cpu را که برابر با 72 مگاهرتز بود تقسیم بر (prescaler+1)*7200 کردیم که 7200 براب مقدار ماکسیمم تایمر pwm است .

تایمر 1 برای pwm و تایمر 2 برای انکودر است (که انکودر دارای 2 کانال (فاز A و فاز B)است)

برای انتگرالگیر مقدار خطا های قبلی را با هم جمع میکنیم و برای آن حد اشباع قرار میدهیم (برابر نصف مقدار حداکثرPWM). همچنین برای کل pid هم یک حد اشباع قرار میدهیم.

این ضرایب انگرالگیر و تناسبی را با سعی و خطا بدست می آوریم .

برای محاسبه سرعت مقدار پالس خوانده شده در لحظه را منهای مقدار پالس قبلی میکنیم (چون مشتق موقعیت است ،البته در اینجا چک میکنیم که آیا یک دور چرخیده است یا خیر).

2) کنترل موقعیت

برای این قسمت از کنترلر casecade استفاده میکنیم به این ترتیب که حلقه درونی کنترل سرعت طراحی شده و حلقه بیرونی قسمت کنترلر موقعیت میباشد البته لازم به ذکر است که حلقه درونی که کنترلر سرعت میباشد باید 5 الی 10 برابر از حلقه بیرونی سریعتر باشد . در اینجا خروجی کنترلر موقعیت set point حلقه داخلی سرعت میباشد .

البته در این قسمت باید راستگرد یا چپ گرد بوده نیز را به وسیله کد دستوری داخل برنامه چک کرده و بر حسب آن دستورکنترلی اعمال نمود . در این قسمت کنترل کننده موقعیت فقط کنترل کننده فقط تناسبی است .

```
void PIDCalc()
{

  Pulse=__HAL_TIM_GET_COUNTER(&htim2);
  posErr= posSetpoint - Pulse;


  posControllerOut = KpPos * posErr;
  Speed=1300*Revolve1-1300*Revolve2+Pulse-PPulse;
  PPulse=Pulse;
  Revolve1=0;
  Revolve2=0;

  MyError = posControllerOut-Speed;
//    if( posControllerOut< 0)
//  {
//      HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_11);
//      HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_12);
//      posControllerOut=-posControllerOut;
//  }
  IntError+=MyError;
  if(IntError>3600)
  {
    IntError=3600;
  }
  if(IntError<-3600)
  {
      IntError=-3600;
  }
  Pterm=Kp*MyError;
  Iterm=Ki*IntError;

  ControllerOut=Iterm+Pterm;

  if(ControllerOut<-7200)
    ControllerOut=-7200;

  if(ControllerOut>7200)
    ControllerOut=7200;

    if( ControllerOut< 0)
  {
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, GPIO_PIN_RESET);
      __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_1,-ControllerOut);
  }
  else
  {
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, GPIO_PIN_SET);
    __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_1,ControllerOut);
  }
```

کنترلر سرعت

```
202
203
204  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
205 {
206   if(htim->Instance==TIM1)
207   {
208     cnt = cnt+1;
209     if(cnt == 100){
210       PIDCalc();
211       cnt = 0;
212     }
213   }
214   if(htim->Instance==TIM2)
215   {
216     Revolve=1;
217   }
218
219   /* USER CODE END 3 */
220
221 }
222
```

فرمان اعمال کنترلر در هر 10 میلی ثانیه

```
void PIDCalc()
{

  Pulse=__HAL_TIM_GET_COUNTER(&htim2);
  posErr= posSetpoint - Pulse;


  posControllerOut = KpPos * posErr;
  Speed=1300*Revolve1-1300*Revolve2+Pulse-PPulse;
  PPulse=Pulse;
  Revolve1=0;
  Revolve2=0;

  MyError = posControllerOut-Speed;
//    if( posControllerOut< 0)
//  {
//      HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_11);
//      HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_12);
//      posControllerOut=-posControllerOut;
//  }
  IntError+=MyError;
  if(IntError>3600)
  {
    IntError=3600;
  }
  if(IntError<-3600)
  {
      IntError=-3600;
  }
  Pterm=Kp*MyError;
  Iterm=Ki*IntError;

  ControllerOut=Iterm+Pterm;

  if(ControllerOut<-7200)
    ControllerOut=-7200;

  if(ControllerOut>7200)
    ControllerOut=7200;

    if( ControllerOut< 0)
  {
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, GPIO_PIN_RESET);
      __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_1,-ControllerOut);
  }
  else
  {
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, GPIO_PIN_SET);
    __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_1,ControllerOut);
  }
```

کنترل موقعیت