دانشگاه صنعتی شریف

آزمایشگاه سخت‌افزار

گروه ۱

گزارش نهایی پروژه

سامانه کنترل دریچه گاز الکترونیکی

علی هاشم آبادی - ۹۷۱۰۶۳۱۳

به نام خدا

فهرست مطالب

فهرست تصاویر*

*تصاویر بخش تست و کد ذکر نشده‌اند. برای مشاهده این تصاویر به فصل تست و کد برنامه مراجعه کنید

مقدمه

واحدهای کنترلی در دنیای امروز بسیار حائز اهمیت هستند، مخصوصا در داخل خودروها. یکی از مهمترین واحدهای کنترلی درون خودرو که به طور دقیق‌تر، درون موتور قرار دارد، کنترل دریچه گاز نام دارد. دریچه گاز، جریان سوخت و هوا را از خود عبور میدهد و واحد کنترلی آن، بر روی سرعت و کنترل بر موتور موثر میباشد .طراحی چنین محصولاتی در دنیای واقعی، سخت و پرهزینه بوده و به سبب حساسیت بالای سامانه، هر اشکالی میتواند باعث ایجاد هزینه‌های فراتر از مادیات بشود. به همین دلیل، از ابزار OpenModelica برای شبیه‌سازی این سامانه استفاده خواهیم کرد. شبیه‌سازی کم‌هزینه، بی‌خطر بوده و تا حد قابل قبولی، نتایج مناسبی در اختیار ما قرار می‌دهد. ابزار OpenModelica از زبان شی‌گرای Modelica استفاده می‌کند که یک زبان مخصوص شبیه‌سازی می‌باشد.

برای کنترل دریچه گاز، میبایست سامانه را به صورت Closed-Loop طراحی نمود؛ یعنی خروجی سامانه باید به عنوان فیدبک در ورودی استفاده گردد تا کنترل سامانه راحت‌تر باشد.

در ادامه، مستندی دقیق از پروژه در اختیار شما قرار خواهد گرفت.

دیتاشیت محصول

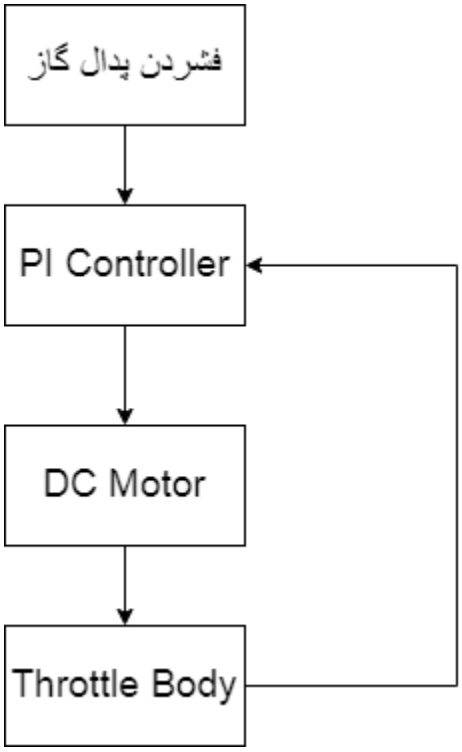
ولتاژ ورودی (پدال گاز)	حداکثر ۵ ولت
Proportional Gain	۱۶
Integral Gain	۸
فرکانس PWM	۱۰۰۰ هرتز
ضریب القاگر	۱ H
ثابت inertia	۰,۲ $Kg.m^2$
ثابت Damper	۱ N.m.s/rad
ثابت فنر	۵ N.m/rad
دمای H-bridge	۲۰ C^o ثابت فرض شده است

سایر موارد به صورت دیفالت خود ابزار OpenModelica باقی مانده است.

معماری سیستم

سیستم کنترل دریچه گاز الکترونیکی از سه بخش مختلف تشکیل شده است:

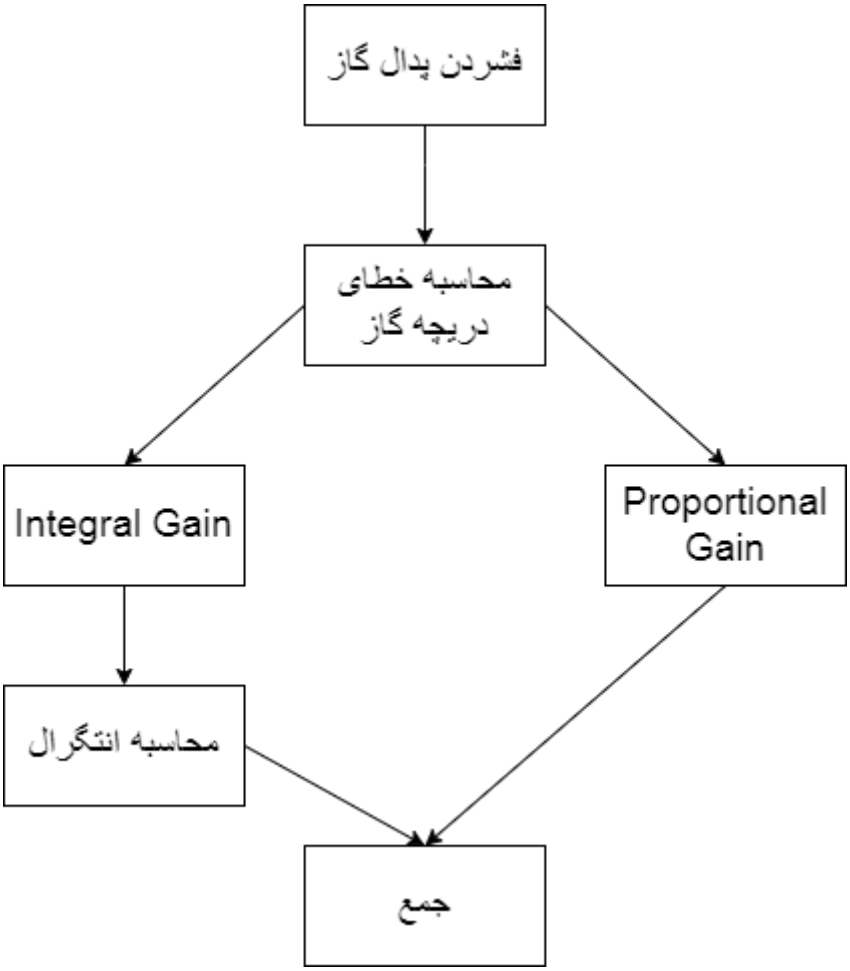
- PI Controller
- موتور DC
- بدنه دریچه گاز



پدال گاز به عنوان یکی از ورودی‌های سامانه می‌باشد. از آنجا که سامانه به صورت Closed-Loop طراحی شده است، خروجی سامانه به صورت فیدبک به عنوان ورودی دوم استفاده می‌شود.

این کار باعث کنترل بهتر سامانه می‌شود.

بخش PI Controller به صورت زیر است:



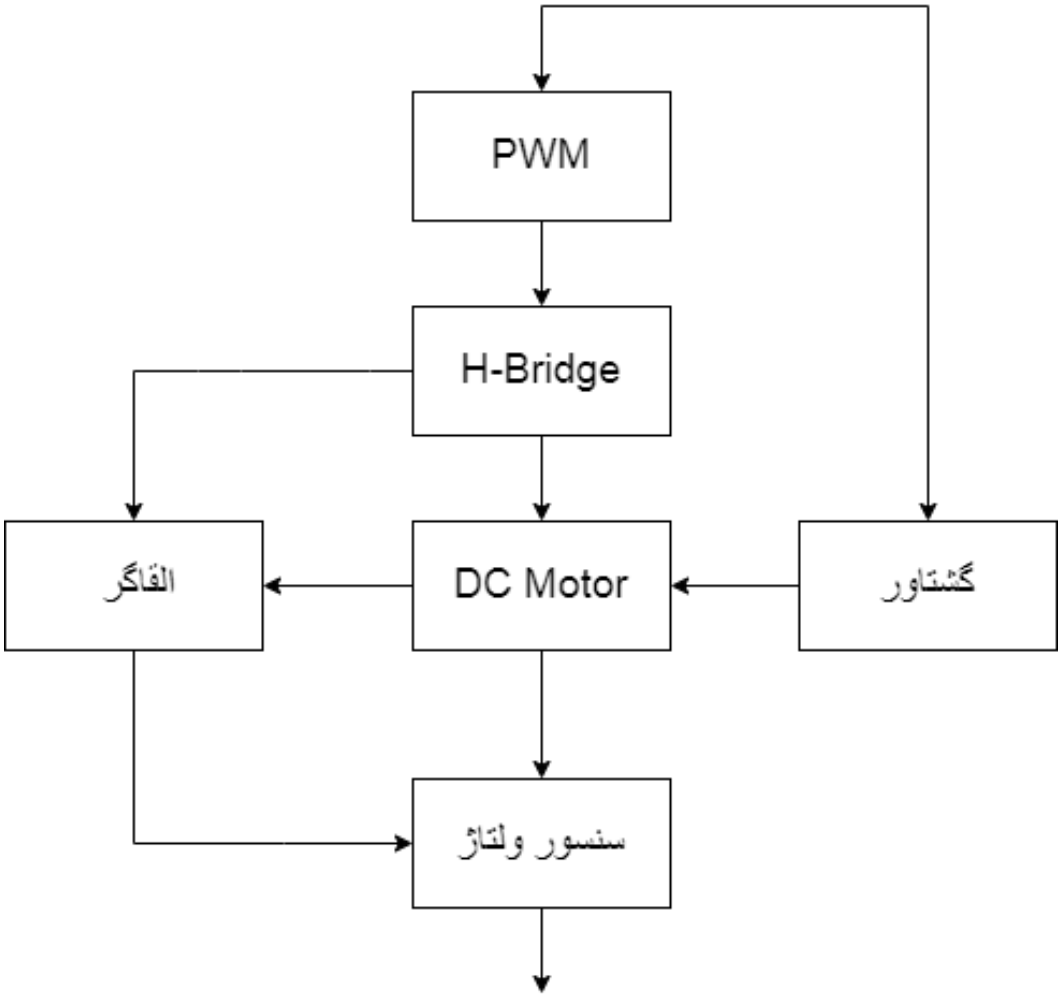
$$u(t) = Kp * e(t) + Ki \int_0^t e(x)dx$$

ورودی این بخش، ولتاژ پدال و ولتاژ دریچه گاز اســت که به عنوان فیدبک مورد اســتفاده قرار میگیرد. ابتدا نیاز داریم تا اختلاف یا ارور میان ولتاژ پدال و ولتاژ دریچه گاز را به دست آوریم که نام آن را e(t) میگذاریم.

یک K_P داریم که Gain ارور ماست و به آن Proportional Gain میگویند.

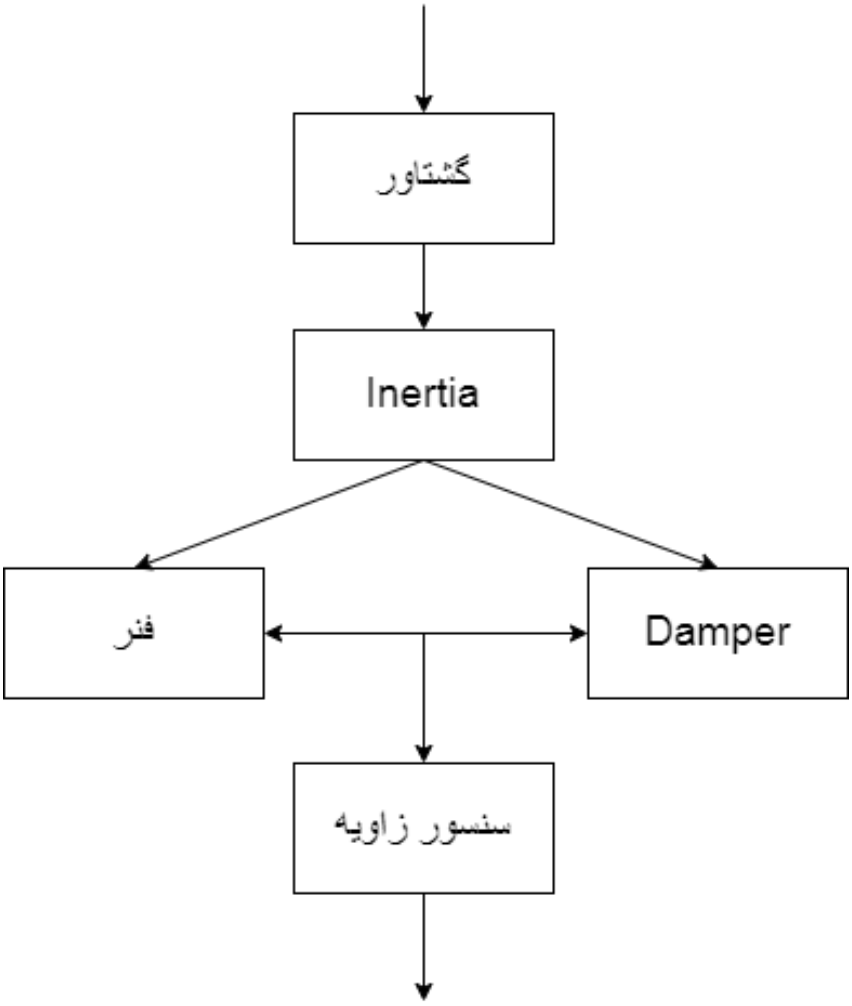
همچنین از ارور میان ولتاژ پدال و ولتاژ دریچه گاز، در واحد زمان انتگرال گرفته میشـــود و در یک K_i ضـــرب میشـــود که به آن Integral Gain میگویند.

حاصل جمع دو مقدار بالا، خروجی کنترلر PI ما خواهد شد و به عنوان ورودی به موتور DC می‌رود. معماری موتور DC به صورت زیر است:



ورودی هم به PWM داده میشود و هم به گشتاور متصل به موتور DC. خروجی PWM به سمت H-bridge رفته و یک سر H-bridge به القاگر متصل است و سر دیگر آن به موتور DC. سر دیگر موتور DC به القاگر متصل شده و سپس به کمک یک سنسور، ولتاژ تولیدی را به دست آورده و به عنوان خروجی مشخص میکنیم و آن‌را به عنوان ورودی به دریچه گاز تحویل می‌دهیم.

معماری بدنه دریچه گاز به صورت زیر است:



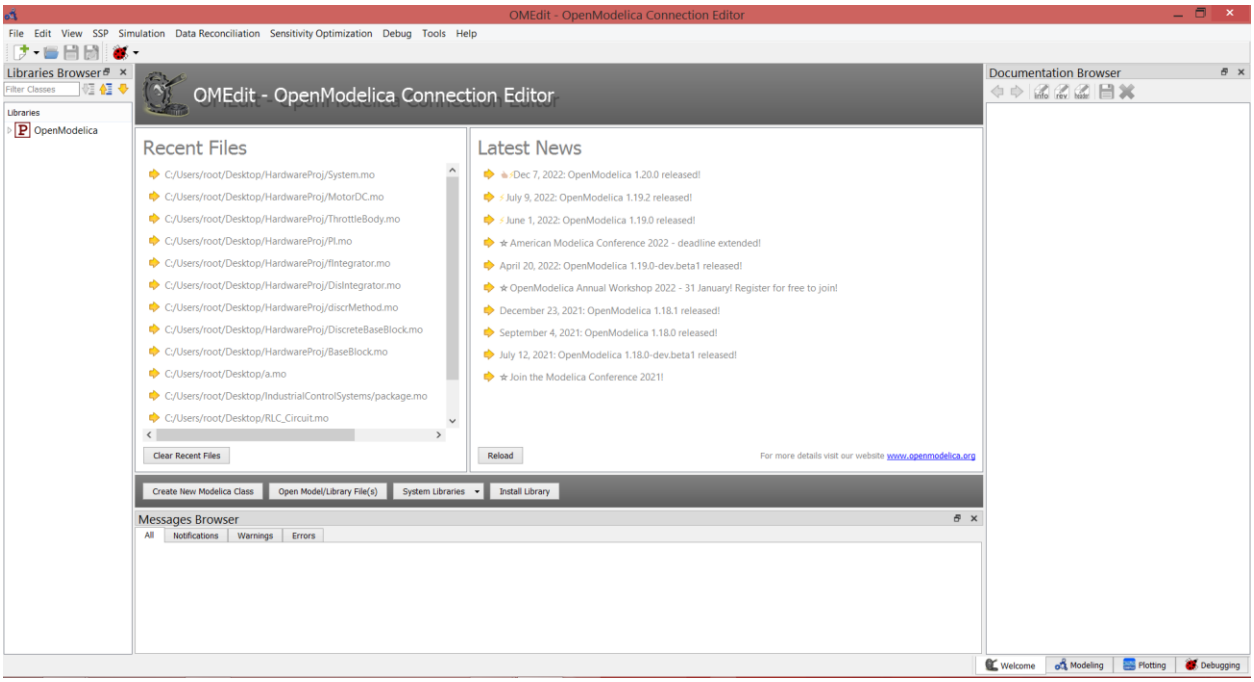
بدنه دریچه گاز متشـکل اسـت از فنر و inertia و گشـتاور و damper که به طریقه بالا متصـل شده‌اند. یک سنسور قرار دادیم تا زاویه دریچه مشـخص شـود. سـپس ولتاژ نهایی را به صـورت فیدبک به عنوان ورودی دوم PI Controller مشـخص می‌کنیم تا سامانه Closed-Loop ما تکمیل شود.

راه‌اندازی و راهنمای کاربر

ابتدا نیاز به نصب برنامه OpenModelica داریم.

این ابزار به صورت رایگان و متن‌باز در دسترس است. با مراجعه به وبسایت داده شده، امکان دانلود آن وجود دارد.

پس از نصب برنامه که بسیار آسان است، نرم‌افزار را باز کنید. با محیط زیر روبرو می‌شوید.

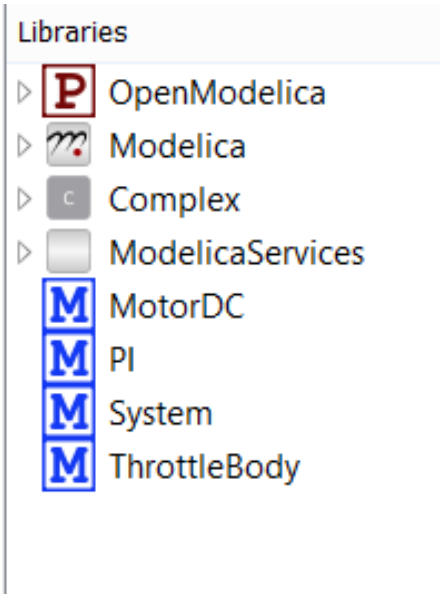


ابتدا باید کتابخانه Modelica که مخصوص خود نرم‌افزار هست را import کنیم.

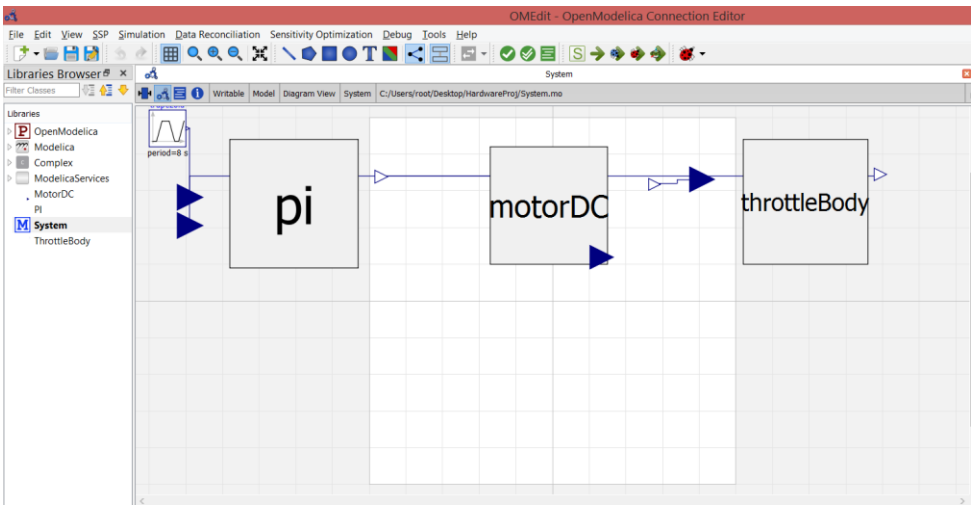
File => System Libraries => Modelica => 4.0.0

بعد از import شدن کتابخانه، فایل‌های مرتبط با پروژه را باز کنید.

File => Open Model/Library File

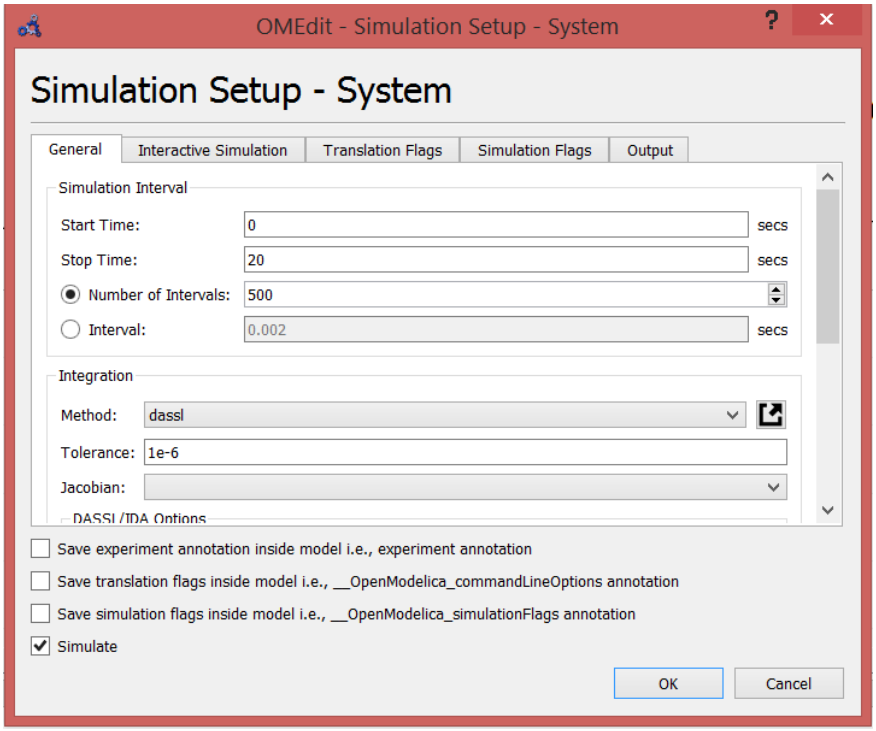


در پنجره گوشه صفحه، موارد بالا باید موجود باشند. سپس روی System کلیک راست کرده و گزینه Open Class را بزنید.



حال نحوه اجرای شبیه‌سازی را آموزش می‌دهیم.

در پنجره بالا، علامت  یا Simulation Setup را بزنید.

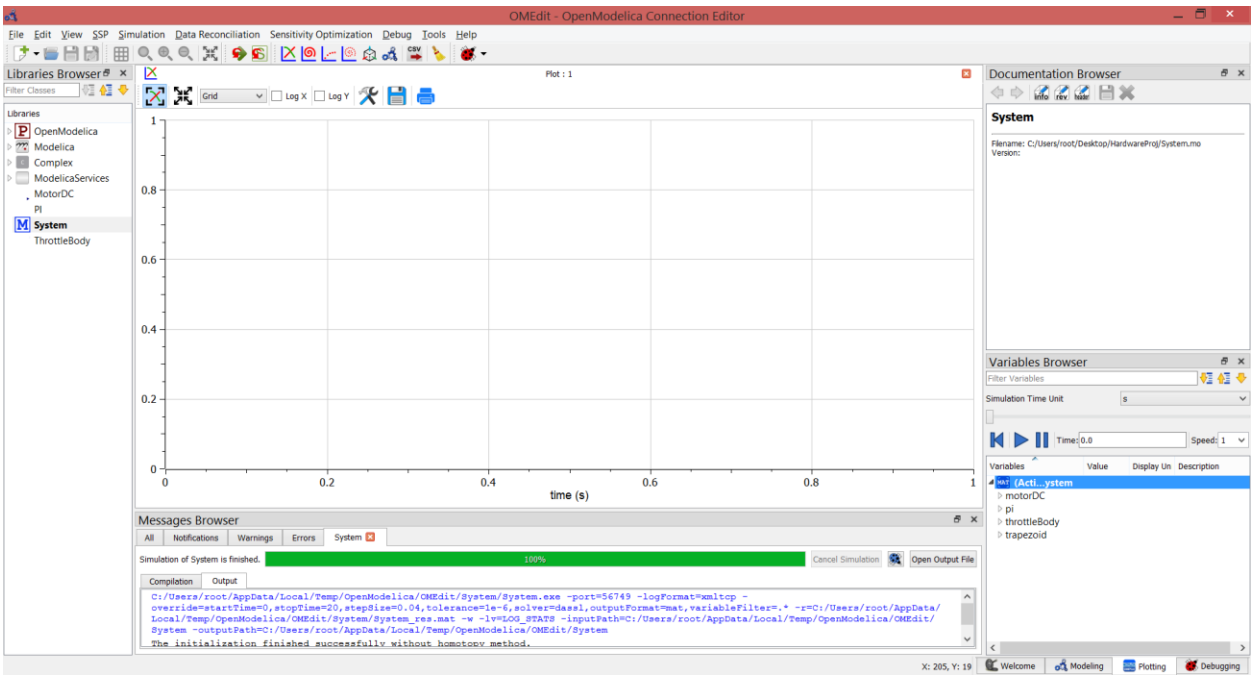


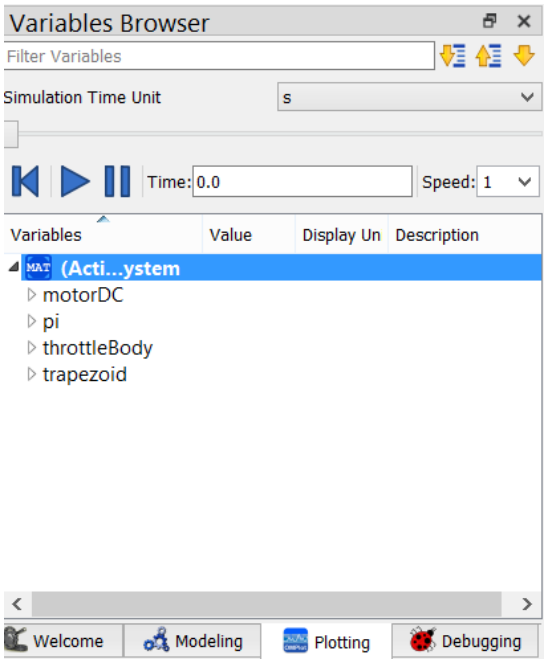
در این پنجره، می‌توانید تنظیمات شبیه‌سازی را مشخص کنید؛ از جمله زمان شروع و پایان،.....

مطمئن باشید که تیک Simulate در انتهای پنجره فعال باشد تا علاوه بر کامپایل، شبیه‌سازی نیز انجام شود. بعد از کلیک بر Ok، وارد صفحه جدیدی با نام Plotting می‌شوید و برنامه در حال کامپایل و شبیه‌سازی است.

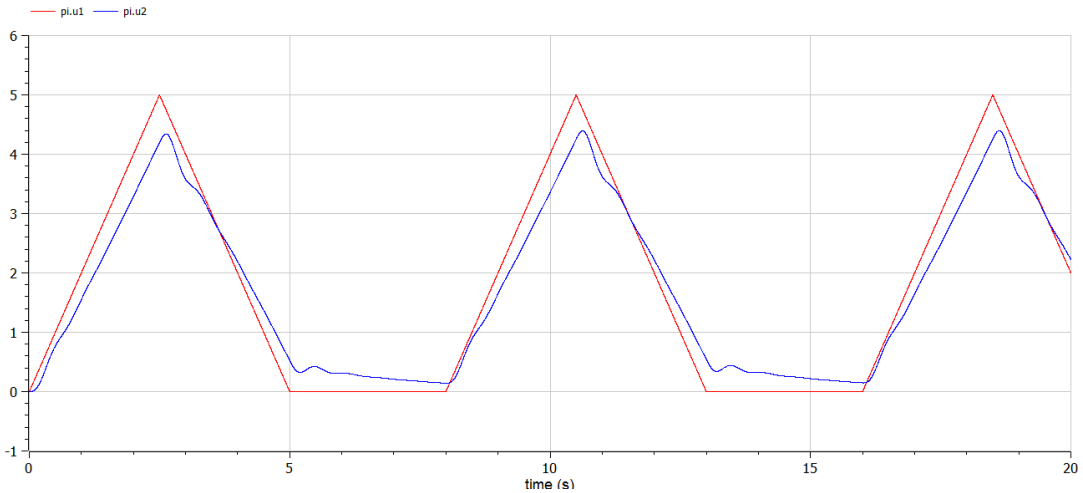
بسته به زمان شبیه‌سازی و پیچیدگی برنامه، کامپایل و شبیه‌سازی ممکن است طولانی شود.

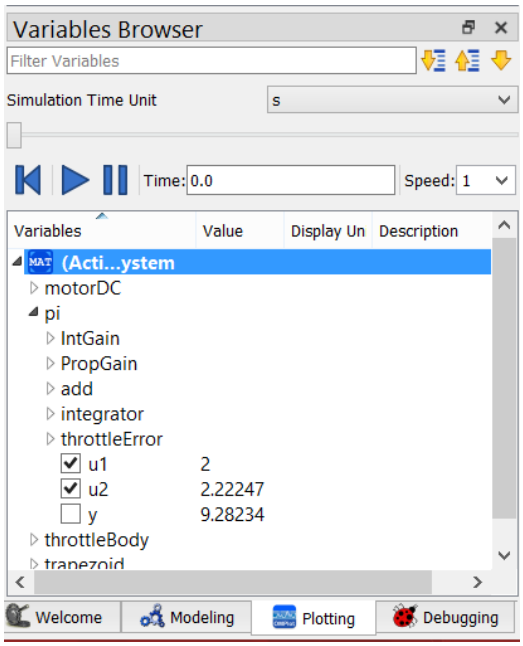
پس از اتمام کامپایل و شبیه‌سازی، در سمت راست، گوشه پایین صفحه، خروجی‌های هر بخش نمایش داده می‌شوند.



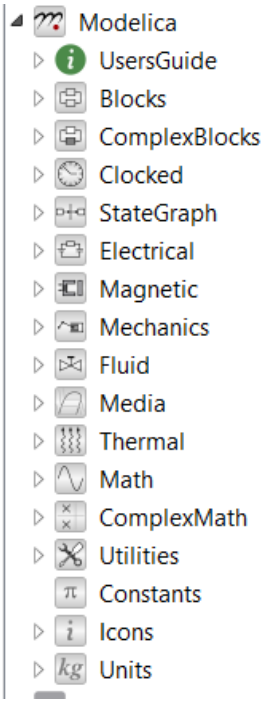


این صفحه، صفحه Plotting بوده که خروجی‌های انتخابی شما را رسم می‌کند و در نمودار روبرویتان قابل مشاهده خواهد بود. به عنوان مثال برای نمایش ولتاژ پدال و ولتاژ خروجی، بر سه‌گوش کنار PI کلیک کرده و سپس مربع کنار u1 و u2 را فعال کنید (بسته به نوع برنامه و زمان شبیه‌سازی، ممکن است تیک‌دار کردن مربعات طول بکشد)





خودتان نیز می‌توانید به بلوک‌های کتابخاه دسترسی داشته باشید. بر فلش کنار Modelica کلیک کنید و خانواده‌های مختلفی از بلوک‌ها در دسترس شما خواهد بود. هر بلوکی را می‌توانید به صورت Drag&Drop در برنامه اضافه کنید و با کلیک راست بر هرکدام و زدن گزینه Open Class میتوانییید بلوک را با جزئیات بیشتر مشاهده کنید.

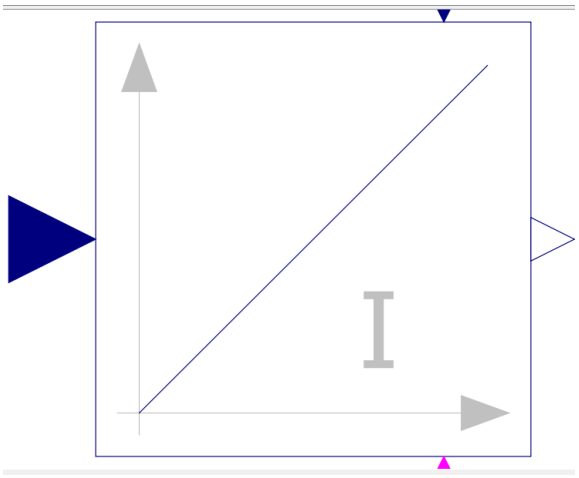


هر بلوکی کـه بـا Open Class بـاز می‌کنیـد، بـا چهـار نوع دیـد مختلف می‌توانیـد آن را مشـــاهده کنید. به عنوان مثال بلوک Integrator را در نظر بگیرید. این بلوک را میتوانید از بخش زیر پیدا کنید:

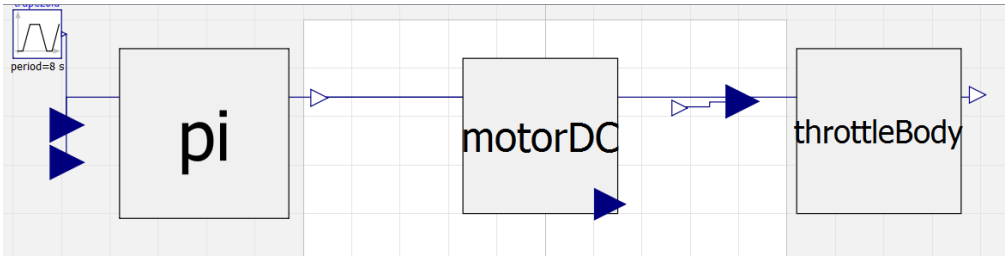
Modelica => Blocks => Continuous => Integrator

چهار نوع دید مختلف در هر مدلی وجود دارد. از سمت چپ توضیح میدهیم.

شکل اول، Icon View نام دارد و صرفا شکل بلوک را نشان می‌دهد.



شـکل دوم، Diagram View نام دارد. اگر بلوک اجزای داخلی داشـته باشـد، یعنی خودش از یکسـری بلوک دیگر تشکیل شده باشـــد، آن بلوک‌ها را نمایش می‌دهد. در Integrator بخش Diagram View نداریم اما مثلا در بلوک System که آن را ران کردیم، Diagram View به صورت زیر است.



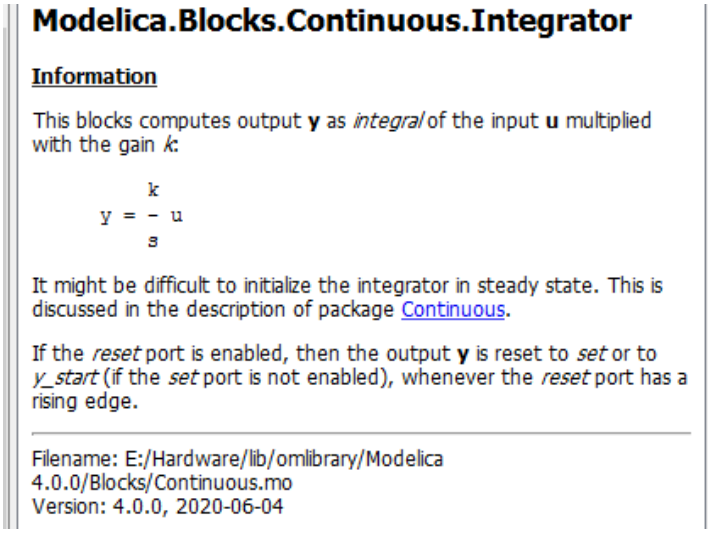
شـــکل ســـوم، Text View نام دارد و کد آن بلوک به زبان Modelica را نمایش می‌دهد. به عنوان مثال Text View بلوک Integrator به صورت زیر است:

```
block Integrator "Output the integral of the input signal with optional reset"
  import Modelica.Blocks.Types.Init;
  parameter Real k(unit="1")=1 "Integrator gain";
  parameter Boolean use_reset = false "= true, if reset port enabled"
    annotation(Evaluate=true, HideResult=true, choices(checkBox=true));
  parameter Boolean use_set = false "= true, if set port enabled and used as reinitialization value
when reset"
    annotation(Dialog(enable=use_reset), Evaluate=true, HideResult=true, choices(checkBox=true));

  /* InitialState is the default, because it was the default in Modelica 2.2
   and therefore this setting is backward compatible
*/
  parameter Init initType=Init.InitialState
    "Type of initialization (1: no init, 2: steady state, 3,4: initial output)"
annotation(Evaluate=true, ...);
  parameter Real y_start=0 "Initial or guess value of output (= state)"
    annotation (Dialog(group="Initialization"));
  extends Interfaces.SISO(y(start=y_start));
  Modelica.Blocks.Interfaces.BooleanInput reset if use_reset "Optional connector of reset signal"
annotation(Placement( ...);
  Modelica.Blocks.Interfaces.RealInput set if use_reset and use_set "Optional connector of set
signal" annotation(Placement( ...);
protected
  Modelica.Blocks.Interfaces.BooleanOutput local_reset annotation(HideResult=true);
  Modelica.Blocks.Interfaces.RealOutput local_set annotation(HideResult=true);

initial equation
  if initType == Init.SteadyState then
```

و شکل آخر، Documentation View نام دارد که مستندی از بلوک مورد نظر است. به عنوان مثال مستند بلوک Integrator به صورت زیر است:



Modelica.Blocks.Continuous.Integrator

Information

This blocks computes output **y** as *integral* of the input **u** multiplied with the gain *k*:

$$y = \frac{k}{s} u$$

It might be difficult to initialize the integrator in steady state. This is discussed in the description of package Continuous.

If the *reset* port is enabled, then the output **y** is reset to *set* or to *y_start* (if the *set* port is not enabled), whenever the *reset* port has a rising edge.

Filename: E:/Hardware/lib/omlibrary/Modelica 4.0.0/Blocks/Continuous.mo
Version: 4.0.0, 2020-06-04

در ادامه به توضیح بلوک‌های مورد استفاده از کتابخانه Modelica در پروژه می‌پردازیم تا با نوع کار سامانه آشنا شویم.

PWM

مدولاسیون پهنای پالس یا PWM روشی برای تنظیم توان الکتریکیِ بار، با تغییردادن زمان قطع و وصل شدن منبع توان به بار در هر سیکل است.

مدولاسیون پهنای پالس، در مهندسی الکترونیک و کنترل، کاربردهای گوناگونی دارد.

بخش اصلی PWM ، یک سیگنال کنترلی به شکل موج مربعی (پالس) است، به‌طوری‌که دوره کاری پالس‌ها، در هر دوره تناوب موج (هر سیکل)، قابل تنظیم است. دوره کاری، نسبت مدت High بودن موج مربعی به دوره تناوب آن است، و بر حسب درصد بیان می‌شود. در واقع این سیگنال، قطع و وصل شدن منبع توان به بار را تعیین می‌کند (با کنترل باز و بسته شدن یک سوییچ الکترونیکی).

مثلاً اگر دوره کاری موج مربعی، ۴۰٪ باشد، در ۴۰٪ هر دوره تناوب، بار به منبع توان وصل است و در باقی آن، قطع می‌شود. در این حالت، متوسط توان بار، ۴۰٪ توان منبع خواهد بود.

اگر یک میکروکنترولر با تغذیه ۵ ولت (V_{CC}) سیگنال PWM با دوره کاری ۵۰٪ تولید کند، متوسط موج مربعی تولیدشده، برابر نصف V_{CC} یا ۲,۵ ولت خواهد بود. به‌طور کلی، اگر دوره کاری با D نشان داده شود، متوسط ولتاژ، برابر با $V_{CC}D$، و RMS آن، برابر با $V_{CC}(D)^{0.5}$می‌شود.

H-Bridge

پل اچ یا H-bridge، نوعی مبدل اســت که در مدارات الکترونیکی یا الکترونیک قدرتی مورد اســتفاده قرار می‌گیرد. از خصوصیات مبدل تمام پل این اســت که ولتاژ و جریان خروجی آن، می‌تواند دارای هر جهت جریان یا پُلاریته ولتاژی باشــد. این مدارها عموماً در رباتیک و دســتگاه‌های دیگر برای راه‌اندازی موتور در جهت مســتقیم و معکوس استفاده می‌شوند. پل‌های اچ به صورت تراشه در دسترس هستند. همین‌طور می‌توان آن‌ها را با قطعات جدا از هم نیز ساخت.

از کاربردهای این مبدل می‌توان به موارد زیر اشاره کرد:

- تغذیه موتورهای جریان مستقیم و کنترل‌کننده‌های موتور
- تبدیل جریان مستقیم به جریان متناوب تک‌فاز (Inverter)
- تبدیل جریان مستقیم به جریان متناوب فرکانس‌بالا

القاگر

القاگر یک قطعه الکترونیکی، غیرفعال و معمولاً دو-پایانه‌ای است که به آن پیچه یا سیم‌پیچ یا سلف نیز می‌گویند. القاگر، در برابر تغییرهای جریان الکتریکی مقاومت می‌کند. این افزاره معمولاً از رسانایی مانند یک سیم که به صورت سیم‌پیچ درآمده است و به دور هسته‌ای از جنس آهن یا کربن خاص تشکیل می‌شود.

هنگامی که جریان الکتریکی از القاگر بگذرد، میدان مغناطیسی درونش ایجاد می‌شود و انرژی در این میدان مغناطیسی موقت ذخیره می‌شود. وقتی شدت جریان الکتریکی تغییر کند، میدان مغناطیسی متغیّر با زمان، ولتاژی را در رسانا القا می‌کند و بر اساس قانون القای الکترومغناطیسی فارادِی، این ولتاژ مانع از تغییر جریانی می‌شود که در القاگر قرار داشت؛ بنا بر قانون لِنْز، مسیر یک نیروی محرّکه الکتریکی (emf) مخالف مسیر جریانی است که آن را سبب شده است.

مشخصه اصلی القاگر، القاوَری است که یکایش در دستگاه بین‌المللی یکاها(SI) ، هِنری است و با (H) نمایانده می‌شود؛ که به نام جوزِف هِنری، دانشمند و مهندس آمریکایی سده نوزدهم میلادی، نامگذاری شده است.

بیشتر القاگرها هسته‌ای مغناطیسی، ساخته‌شده از آهن دارند که سیم‌پیچ به دور آن‌ها بسته می‌شود و باعث افزایش میدان مغناطیسی و القاوَری می‌شود.

مقاومت‌ها، خازن‌ها و القاگرها از عناصـر خطی و غیرفعال تشـکیل دهنده مدارهای الکترونیکی هسـتند. از القاگرها به‌طور گسـترده در سامانه‌هایی که با جریان متناوب کار می‌کنند، استفاده می‌شود.

از القاگرها برای جلوگیری از عبور سـیگنال با فرکانس زیاد نیز استفاده می‌شود؛ زیرا القاگر، جریان مستقیم را می‌گذرانَد، امّا مانع از گذر جریان متناوب با فرکانس زیاد می‌شود. القاگرهایی که به این منظور طرّاحی شده‌اند، چُوک نامیده می‌شوند. از دیگر کاربردهای القاگر می‌توان به اسـتفاده از آن‌ها در فیلترهای الکترونیکی برای جداسـازی سـیگنال‌ها با بسـامدهای گوناگون و در مدارهای تیونر (تنظیم کننده) گیرنده‌های رادیو و تلویزیون نام برد.

موتور DC

موتور DC یا جریان مستقیم، موتور الکتریکی است که با جریان مستقیم کار می‌کند. این موتور، انرژی الکتریکیِ جریان مستقیم را به انرژی مکانیکی تبدیل می‌کند.

همه موتورهای جریان مستقیم به یک مکانیسم داخلی (مکانیکی یا الکترونیکی) برای تغییر مداوم جهت جریان در آرمِیچر موتور نیاز دارند. به این مکانیسم، کوموتاسیون می‌گویند.

اکثر موتورهای DC حرکت چرخشی دارند، به جز موتورهای خطی که حرکت خطی دارند (نمی‌چرخند).

موتورهای DC دارای گشتاور راه‌اندازی بالا و قابل تغییر هستند و کاربرد بسیاری در صنعت دارند. سرعت موتور DC با روش‌های مختلفی قابل کنترل است؛ از جمله با تغییر ولتاژ تغذیه (مثلاً به روش PWM) ، یا تغییر جریان سیم‌پیچ‌های موتور.

موتورهای DC با ابعاد کوچک در ابزارآلات برقی و لوازم برقی خانگی، اسباب‌بازی‌ها ...، و با ابعاد بزرگ در آسانسورها، خودروهای الکتریکی، بالابرها و در کارخانه‌های نَوَرد لوله و فولاد استفاده می‌شوند. همچنین این موتورها در کاربردهای صنعتی که به کنترل دقیق سرعت و گشتاور نیاز دارند، استفاده‌های فراوان دارند.

گشتاور

گشــتاور نیرو عاملی است که باعث دَوَران یا چرخش جسم می‌شود، همان‌گونه که نیرو باعث حرکت جسم می‌شود. به عبارت دیگر، اثر گشتاور در حرکت چرخشی، مانند اثر نیرو در حرکت انتقالی است.

برای مثال، برای باز کردن در، نیرویی از راه دستگیره (عمود بر در) به آن وارد می‌کنیم تا در به دور محور چرخش (لولا)، بگردد. اگر نیرو به لولا نزدیکتر باشــد یا زاویه نیرو با در کمتر از ۹۰ درجه باشد، باید نیروی بیشتری نسبت به حالت قبل به در وارد کنیم تا در به مانند قبل بگردد. پس گشتاور نیرو هم با مقدار نیرو و زاویه اعمال آن و هم با فاصله آن از محور چرخش رابطه مستقیم دارد.

گشتاور از رابطه $r \times F$ به‌دست می‌آید که در آن، r فاصله نقطه وارد شدن نیرو تا تکیه‌گاه، یا فاصله تا مرکز جرم جسم است.

گشتاور، کمیت بُرداری است. یکای آن در سامانه استاندارد بین‌المللی یکاها، نیوتن متر است.

کنترلر PI

کنترلر PI از رایج‌ترین نمونه‌های الگوریتم کنترل بازخوردی است که در بسیاری از فرایندهای کنترلی نظیر کنترل سرعت موتور DC، کنترل فشار، کنترل دما و... کاربرد دارد. کنترل‌کنندهPI مقدار «خطا» بین خروجی فرایند و مقدار ورودی مطلوب (setpoint)محاسبه می‌کند. هدف کنترل‌کننده، به حداقل رساندن خطا با تنظیم ورودی‌های کنترل فرایند است.

ضریب K_p سرعت سیستم را افزایش می‌دهد و خطای دائم را تا حدودی کاهش می‌دهد (اما صفر نمی‌کند). افزودن جمله انتگرالی (ضریب K_i) خطای حالت دائم را صفر می‌کند، اما مقدار زیادی نوسانات ناخواسته (overshoot) به پاسخ گذرا اضافه می‌نماید.

$$u(t) = Kp * e(t) + Ki \int_0^t e(x)dx$$

Inertia

اینرسی یا لختی یا Inertia خاصیتی از یک جسم است که در برابر تغییر سرعت یا تغییر جهت حرکت جسم مقاومت می‌کند.

هر چه جرم یک جسم بیشتر باشد لختی آن بیشتر است. به قانون اول نیوتون قانون لختی نیز گفته می‌شود.

تمایل اجسام به حفظ حالت قبلی را لختی گویند.

قانون اول نیوتن می‌گوید هرگاه شـی با سـرعت ثابت در مسیری در حال حرکت باشد تا مادامی که نیروی خارجی به آن وارد نشود به حرکت خود در همان مسـیر ادامه خواهد داد. توجه کنید که حرکت دایره‌ایِ یکنواخت، شتاب‌دار است و بردار سرعت دائم تغییر می‌کند.

اینرسـی، یا نیروی ذاتی ماده، قدرت مقاومتی اسـت که با آن هر جرمی، به همان اندازه که آن توان در آن نهفته‌اسـت، برای حفظ وضعیت کنونی خود تلاش دارد، چه این حالت، وضعیت سکون، یا حالت حرکت یکنواخت رو به جلو در یک خط مستقیم باشد.

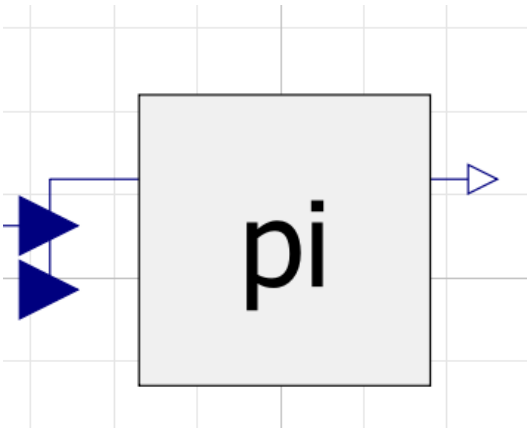
فنر و Damper

برای جلوگیری از دریافت مستقیم ضربه، از فنر استفاده می‌شود. ضربه مستقیم آسیب جدی به سیستم وارد میکند و به همین دلیل حتما باید از فنر استفاده بشود.

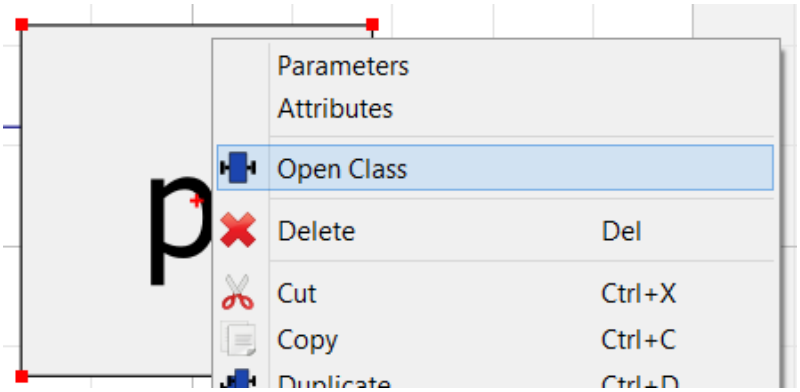
اما Damper خودش کنترل کننده فنر است. درواقع فنر، ضربه را کنترل می‌کند و Damper نوسان را. این باعث می‌شود که فنر در نرود و کنترل سیستم از دست خارج نشود.

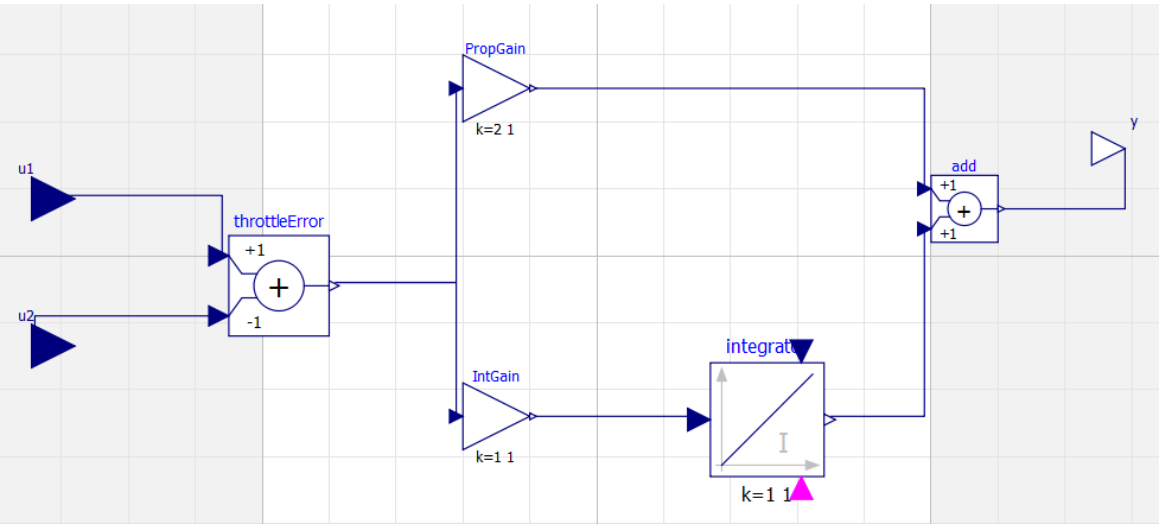
طراحی و پیاده‌سازی

نرم‌افزار OpenModelica و کتاب‌خانه آن؛ و زبان Modelica در طراحی مورد اســتفاده قرار گرفته اســت. ابتدا طراحی PI آغاز شد.



با کلیک راست روی بلوک و انتخاب گزینه Open Class، وارد بلوک PI و طراحی آن می‌شویم.





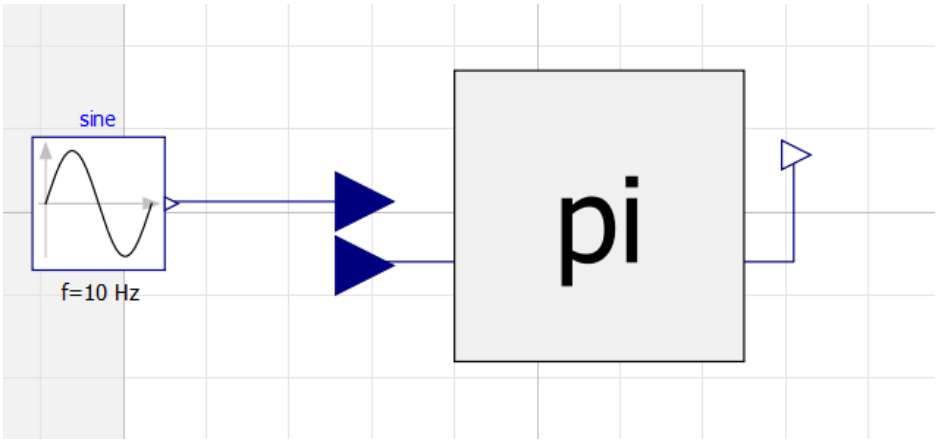
دو ورودی داریم و آن را به یک Adder متصـل کردیم. در Adder قابلیت تعریف Gain وجود دارد. پس Gain یکی از ورودی‌ها را برابر منفی یک میگذاریم تا مقدار دو ورودی از یکدیگر کم بشـود. اسـم این Adder بخصـوص را ThrottleError گذاشتیم تا نمایان‌گر این باشد که خروجی، همان e(t) است.

سـپس e(t) را به دو Gain مختلف متصـل میکنیم. یکی با نام PropGain که همان Proportional Gain ماست و دیگری با نام IntGain که همان Integral Gain ماست.

سـپس خروجی Integral Gain به یک انتگرال‌گیر وصـل میشـود و در ادامه خروجی آن، به سـمت یک Adder میرود که دیگر ورودی آن همان $K_P * e(t)$ می‌باشد.

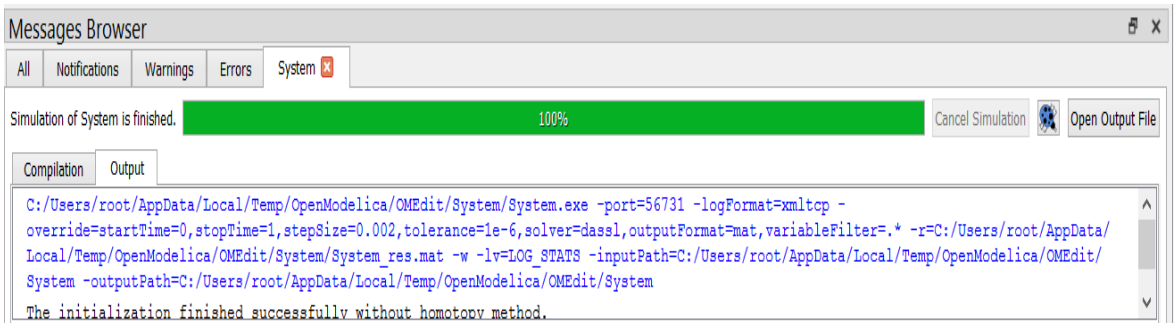
حاصل نهایی، خروجی PI است.

هر بخشی از سیستم، جدا جدا تست شد اما تست کردن ما، صرفا برای اطمینان حاصل کردن از درستی اتصالات بوده و تست نهایی که هر سه بخش تکمیل است، منطق درستی دارد. اما به هرحال، تست بخش کنترلر PI به صورت زیر انجام شد:

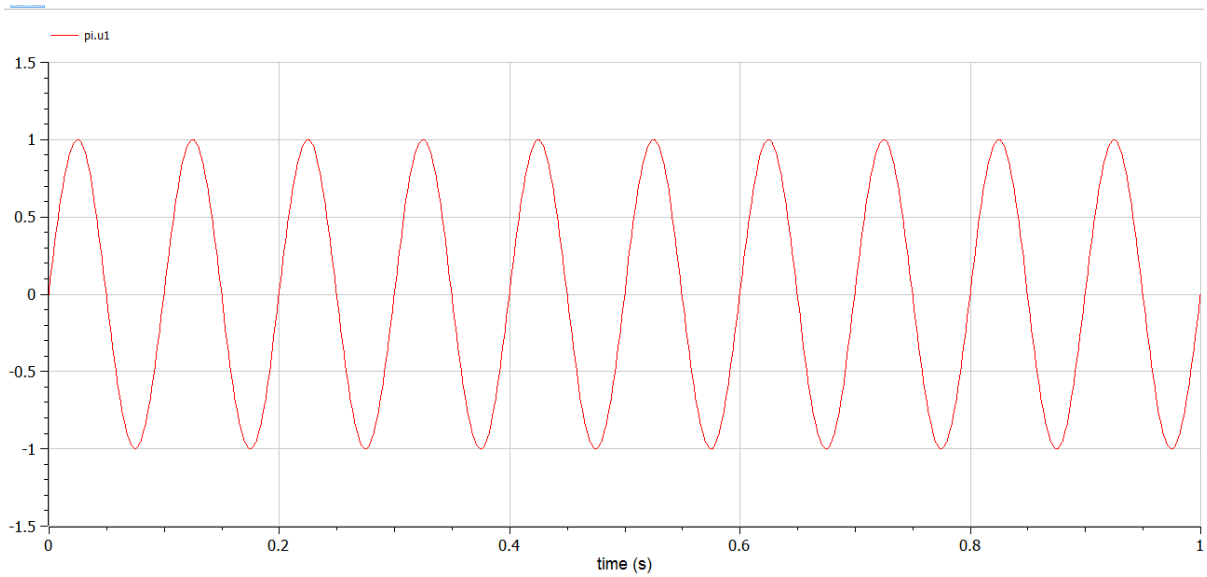


خروجی به ورودی متصل شده است؛ و ورودی دیگر را هرچیزی میتوانیم بگذاریم. به عنوان مثال یک تابع سینوسی را به ورودی داده‌ایم.

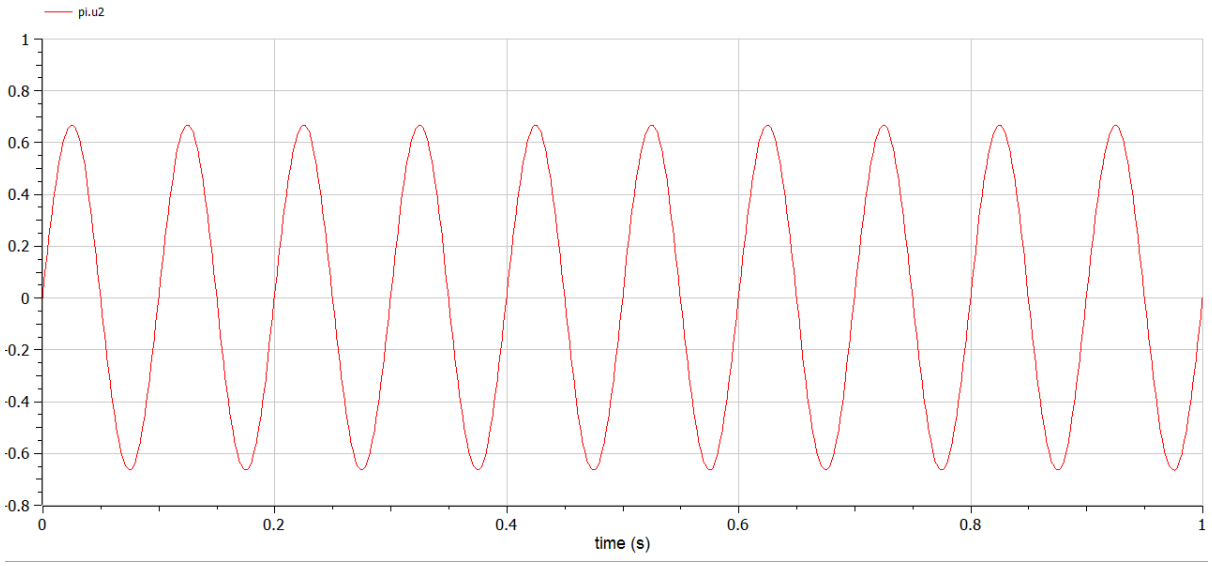
مدل را کامپایل و سپس simulate میکنیم. اگر موفقیت آمیز باشد، این پنجره را خواهیم دید.



ورودی اول که به صورت سینوسی دادیم، به شکل زیر است:



ورودی دوم که به صورت فیدبک داده بودیم، از صفر شروع شده و به صورت زیر ادامه میابد:

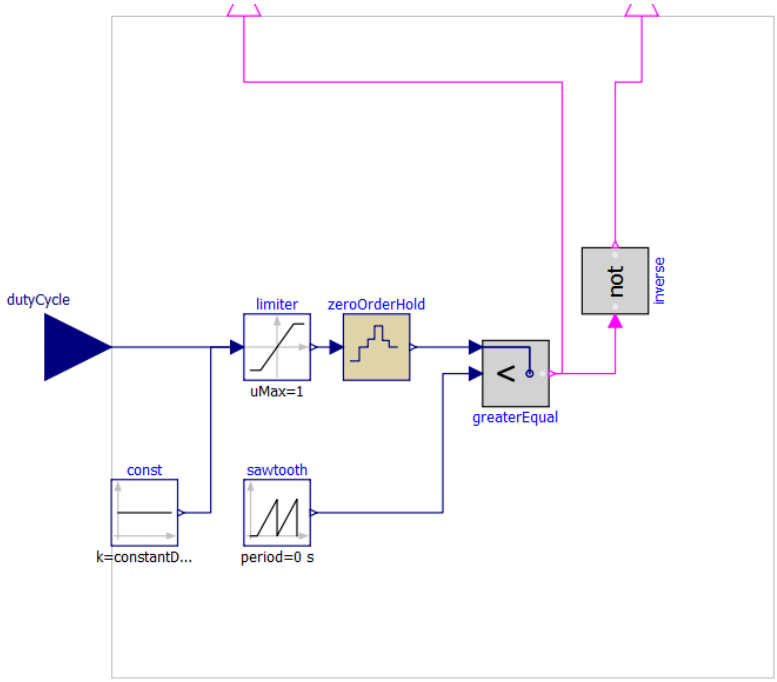


مشخصا در این نوع تست، ورودی دوم مقدار برابری با خروجی PI دارد چون به همدیگر مستقیم متصل است.

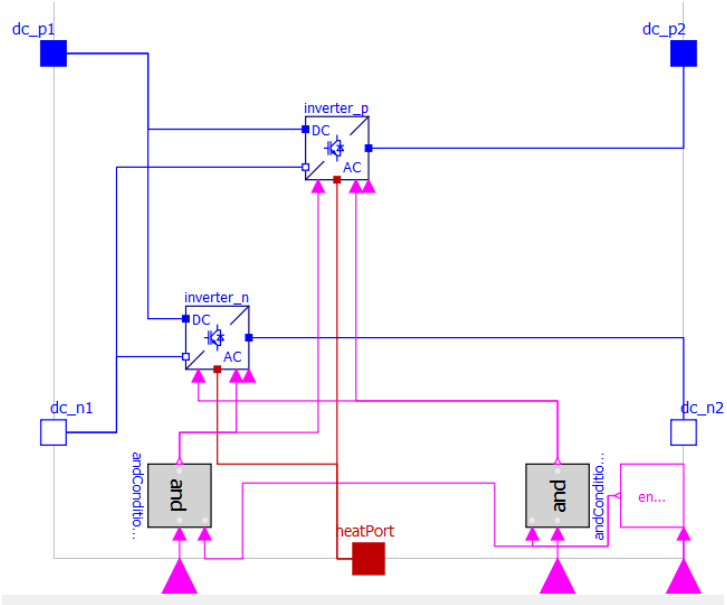
اما متوجه شـدیم که PI درست کار می‌کند. در ادامه که بخش‌های دیگر طراحی شوند و به هم متصل شوند، خروجی PI بهتر قابل درک خواهد بود.

پس به طراحی موتور DC می‌پردازیم.

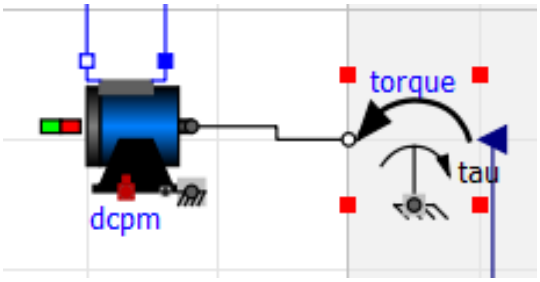
ابتدا به یک قطعه‌ای نیاز داریم با نام Pulse Width Modulation یا به اختصـار PWM. این قطعه برای کنترل قدرت استفاده می‌شـود به این صـورت که سـیگنالی که دریافت میکند را تکه تکه میکند و به صـورت Discrete تحویل میدهد. درواقع انگار خروجی به صورت دیجیتال خواهد بود. این قطعه در OpenModelica موجود است و به صورت زیر طراحی شده است.



برای کنترل جهت/قطبیت و سرعت DC Motor از قطعه‌ای با نام H-Bridge استفاده می‌شود که شکل مدارش شبیه حرف H است. این قطعه نیز در OpenModelica موجود است و به صورت زیر طراحی شده است.



در انتها نیز یک موتور DC میخواهیم که به آن یک Torque یا گشتاور یا نیروی پیچشی متصل است و این نیروی پیچشی قدرت موتور را مشخص میکند.

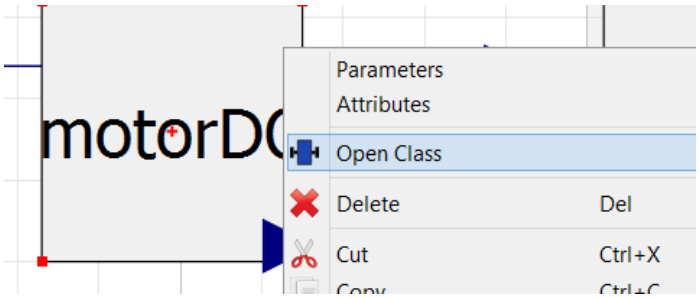


در بخش قبل کنترلر PI را طراحی کردیم. ورودی به DC Motor باید از جنس ولتاژ باشـد. پس خروجی کنترلر PI که به عنوان ورودی به DC Motor داده می‌شـود، باید به ولتاژ تبدیل شـود. از قطعه‌ای با نام signalVoltage اسـتفاده می‌کنیم که یک سیگنالی را ورودی میگیرد و خروجی را به صورت ولتاژ تحویل می‌دهد. حال ولتاژی که داریم را به PWM متصل کرده تا خروجی دیجیتال دریافت کنیم و سپس آن‌را به H-Bridge خود متصل میکنیم.

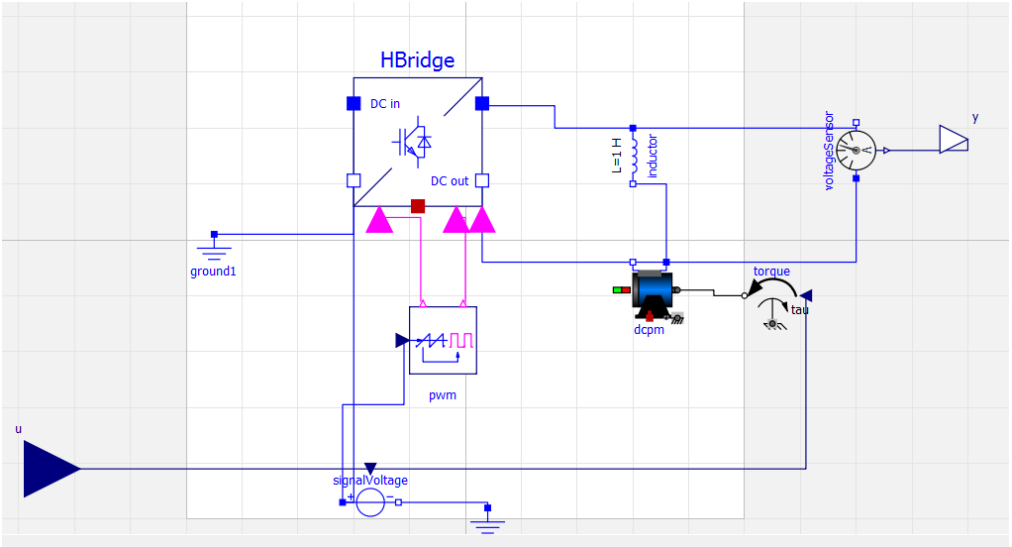
چالشـی در اینجا به وجود آمد. اینکه کدام پورت به کدام پورت در قطعات باید متصـل شـود کمی گنگ بود اما خوشبختانه در خود OpenModelica چندین مثال وجود داشت و مانند همان نیز اتصالات را انجام دادیم. همچنین قطعه H-Bridge نیاز به اتصال به یک ولتاژ ثابت را داشت که آن نیز انجام شد.

چالش بعدی اتصال گشتاور و موتور و H-bridge بود. و موضوع چالش نیز باز چگونگی اتصالات بود که آن هم با مثال‌های موجود در OpenModelica و کمک از آن انجام شد. اما در تمام مثال‌ها یک قطعه دیگر نیز به کار گرفته شده بود که در ویدیویی که برای پروژه Reference داده شده بود، استفاده نشده بود. آن قطعه، القاگر یا Inductor بود. همانطور که میدانیم، القاگر یک قطعه ذخیره کننده انرژی است. در اغلب مدارها استفاده می‌شود و در مثال‌های OpenModelica نیز استفاده شده بود. به همین دلیل در طراحی خود نیز از القاگر استفاده کردیم.

سپس یک سنسور اندازه‌گیری ولتاژ قرار دادیم تا به عنوان خروجی از DC Motor خارج شود به سمت دریچه گاز برود.



بلوک مورد نظر با نام motorDC در مدل System قرار داده شـــده اســـت. اگر Open class را انتخاب کنید، با طراحی‌ای که انجام دادیم مواجه خواهید شد.



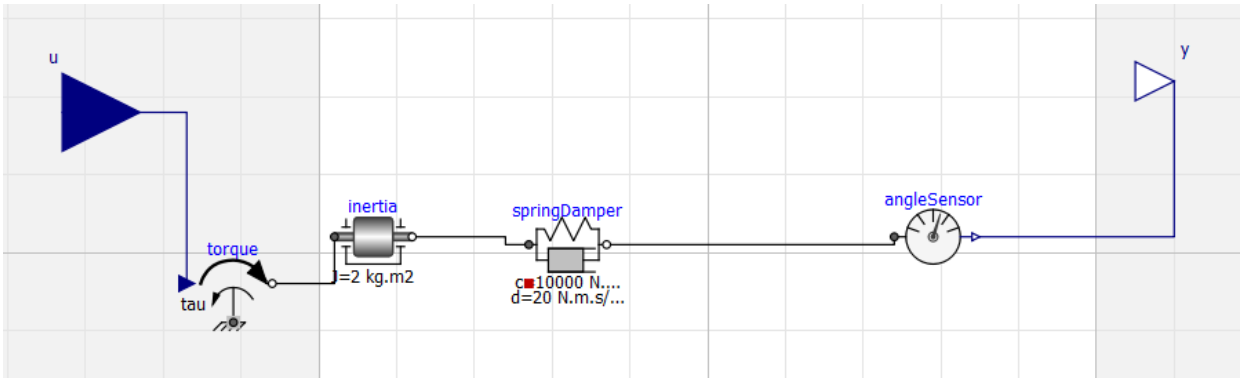
و اما طراحی بدنه دریچه گاز:

بدنه Throttle یا دریچه گاز، طبق رفرنسی که داده شد، شامل اجزای زیر است:

فنر – Damper – قطعه‌ای با نام inertia – hard stop – سنسور تشخیص زاویه.

مشکل این است که قطعه‌ای با نام Hard Stop در OpenModelica وجود ندارد.

به همین دلیل طراحی این بخش صرفا بدون Hard Stop بوده و همچنین چالش دیگر، نحوه اتصالات بوده و همچنین اعدادی که به عنوان پارامتر باید به فنر و damper و ... داد.



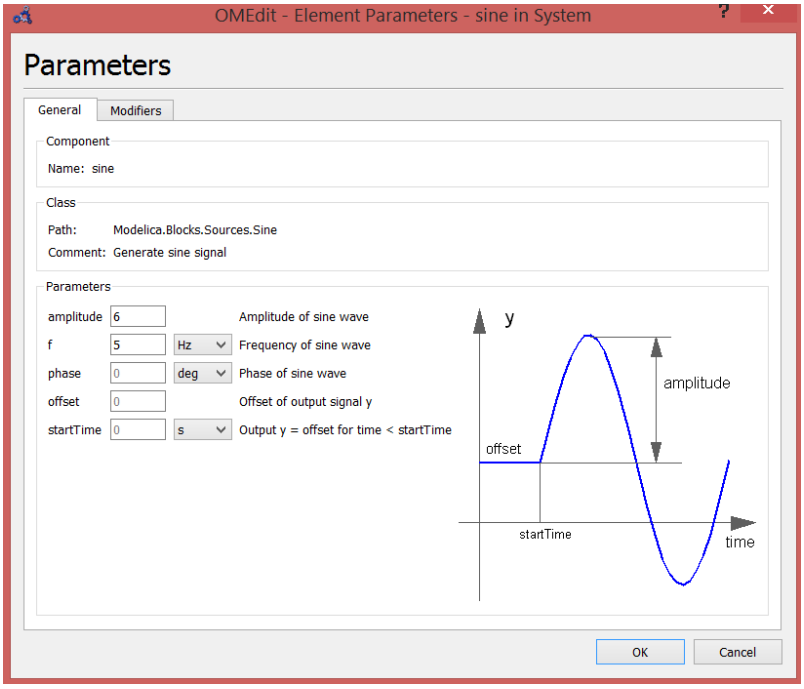
این شکل در ادامه تغییر خواهد کرد (به سبب تست)

همچنین برخلاف رفرنس، طبق مثال‌هایی که در OpenModelica مشاهده شد، همواره به Inertia یک گشتاور متصل بوده. در ادامه تمام بخش‌ها به هم متصل شده و تست گرفته خواهد شد.

تست

تست اول:

ورودی به صورت سینوسی با فرکانس 5 هرتز و amplitude برابر 6 است.

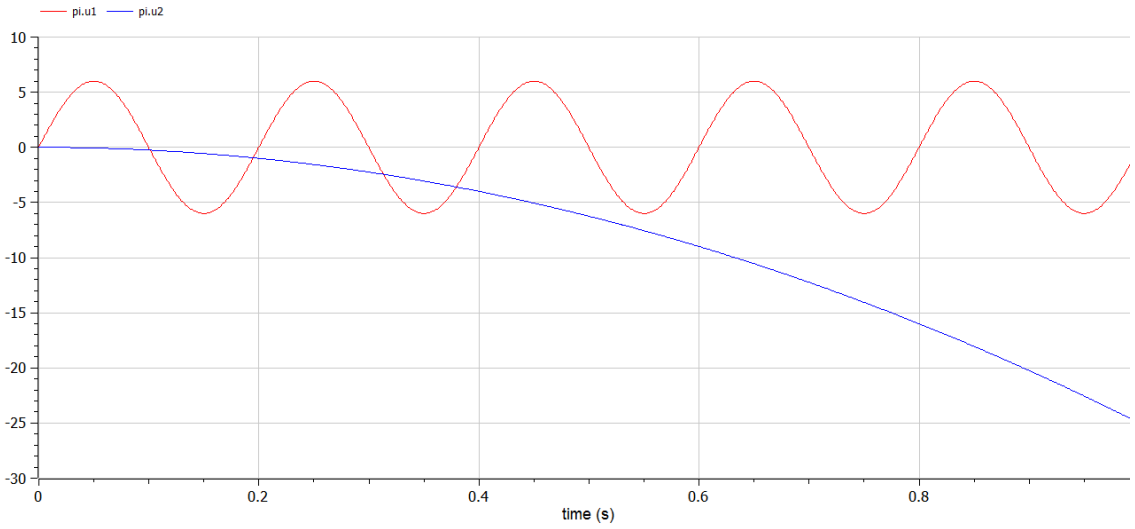


درون مـدل PI، نیـاز به دادن دو نوع مقدار داریم. Proportional gain و Integral gain. اولی را برابر دو و دومی را برابر یک قرار دادیم.

درون مدل motorDC، یک ولتاژ ثابتی به H-Bridge متصل است که مقدار آن را 100V قرار دادیم.

همچنین به گشتاور متصل به موتور، یک ورودی سینوسی مشابه به پدال گاز متصل است.

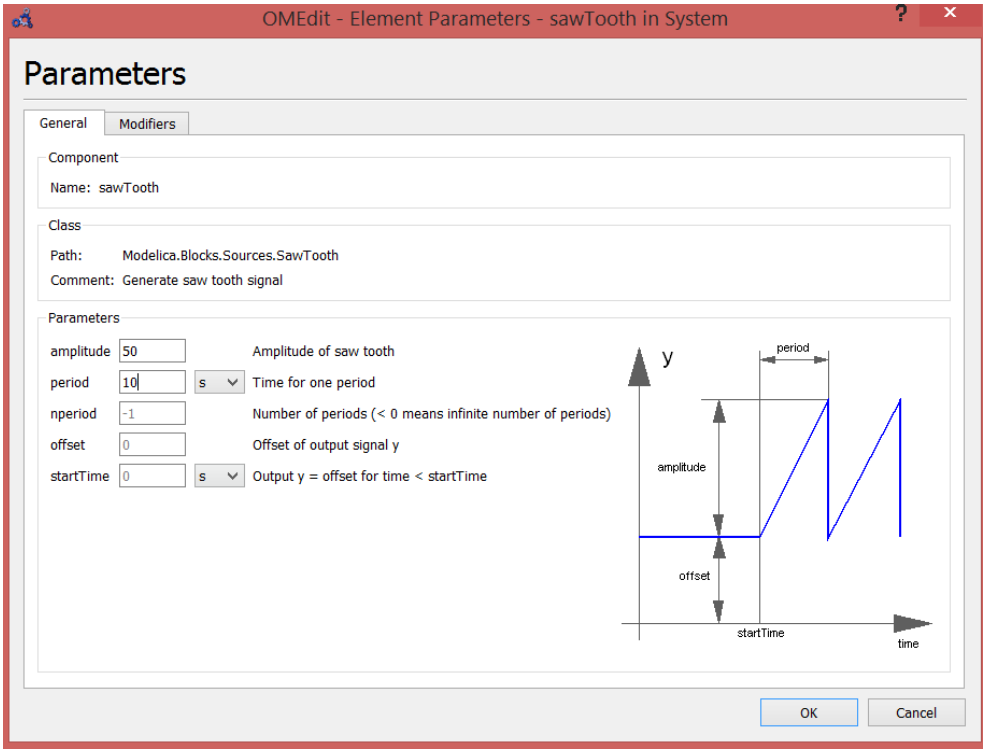
درون مدل ThrottleBody، پارامتر اینرسی با عنوان moment of inertia را برابر J = 2 kg.m2 قرار دادیم و همچنین یک فنر و damper داریم. در فنر، ثابت فنر را برابر c = 10000 N.m/rad قرار داده و ثابت damper را برابر d = 20 N.m.s/rad.



نمودار تست به شکل بالاست. خط نارنجی، ورودی پدال و خط آبی زاویه دریچه گاز است.

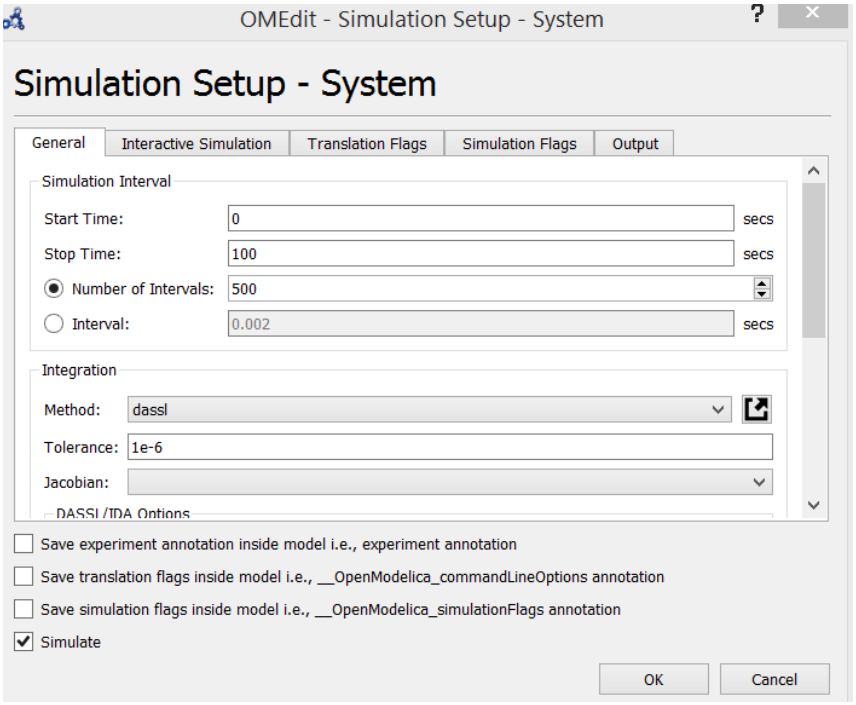
تست دوم:

ورودی پدال شـبیه یک ورودی به صـورت مثلثی است. در OpenModelica چنین تابعی را یافتیم با نام SawTooth که شبیه اره و دندان است.



متغیرهای آن یعنی amplitude را برابر 50 و period را برابر 10 گذاشتیم.

همچنین زمان شبیه‌سازی آزمایش را 100 ثانیه گذاشتیم به صورت زیر:

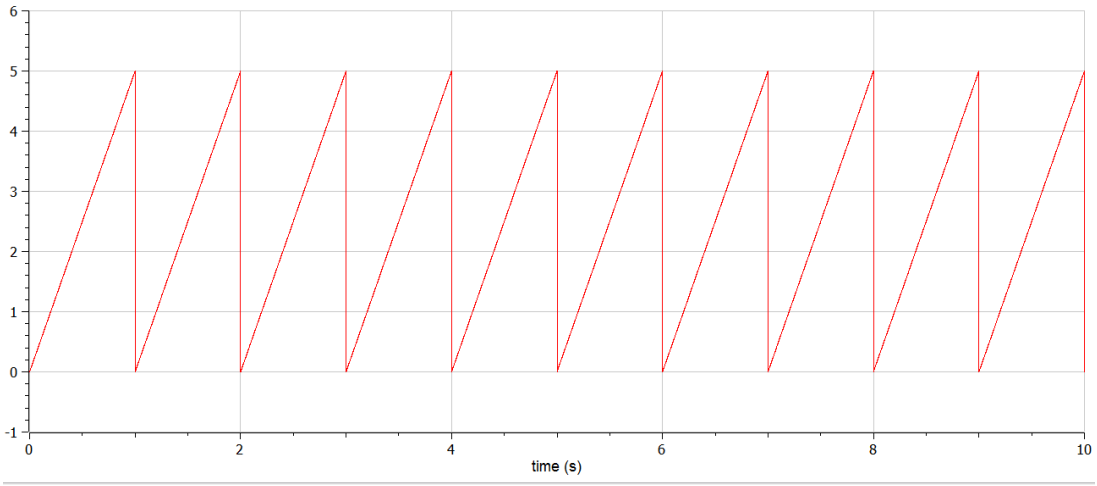


سایر موارد را دست نمیزنیم تا مشاهده کنیم چه جاهایی نیاز به تغییر دارد. پس نمودار ورودی و خروجی:

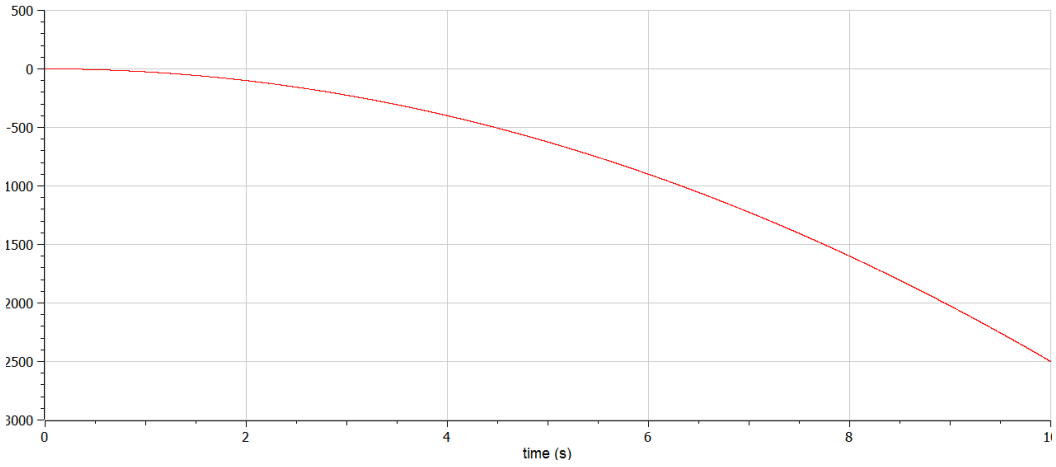
متاسفانه کامپایل و ران کردن آن بسیار طول کشید و جواب نداد. زمان خیلی زیاد، بسیار کند اجرا می‌شود. علت شکست این تست به این دلیل است.

تست سوم:

ورودی را با amplitude برابر 5 و period برابر 1 میگذاریم و شبیه‌سازی را به مدت 10 ثانیه انجام میدهیم. ورودی و خروجی:



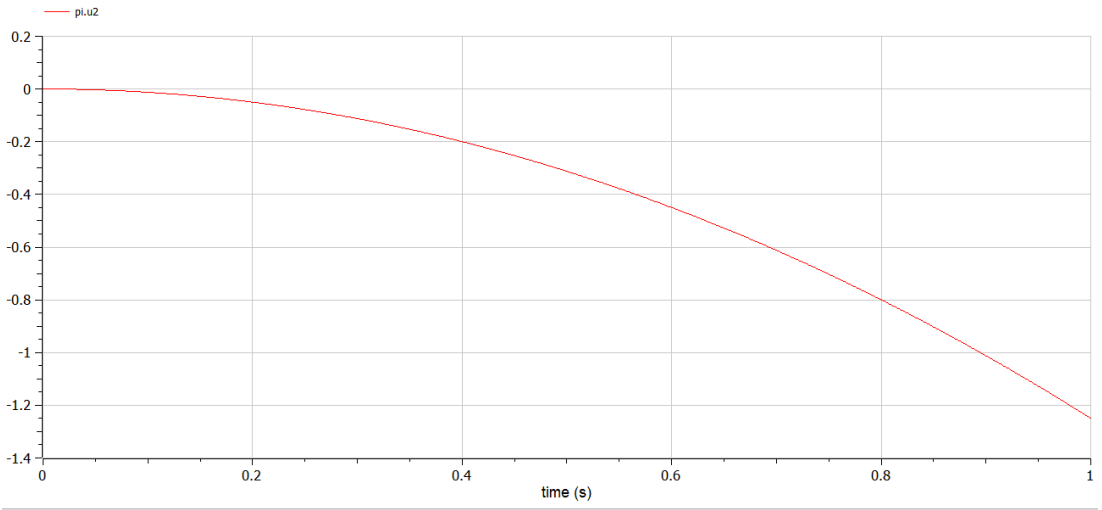
ورودی به صورت بالاست اما خروجی تغییری نکرد:



پس مشخص شد که مشکل از ورودی نیست و باید سایر متغیرها را تغییر داد.

تست چهارم:

گفتیم به H-Bridge یک ولتاژ ثابت 100 ولتی متصل است. مقدار آن را 5 ولت کردیم. تغییری ایجاد نشد جز اینکه انگار این ولتاژ ثابت، amplitude خروجی را مشخص میکند؛ زیرا که صرفا خروجی 1/20 شد.

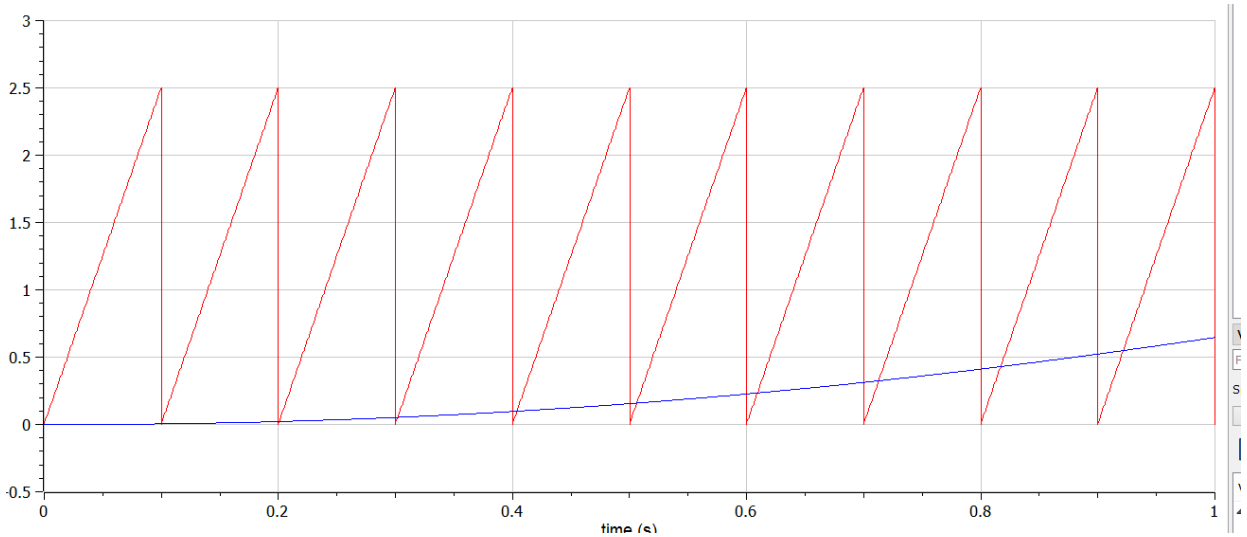


در ادامه چندین تست انجام شد(حدود 7 تست) و همگی ماهیتی یکسان داشتند و یکسری تغییرات در تست بعدی یعنی تست دوازدهم انجام شد.

تست دوازدهم:

ورودی SawTooth را با amplitude برابر 2.5 و period برابر 0.1 دادیم. همچنین گشتاور درون motorDC که یک ورودی نیاز داشت را به خروجی PI متصل کردیم. علت آن این است که حس کردیم خروجی PI باید به نوعی به طور مستقیم به موتور DC نیز تاثیر بگذارد. به‌هرحال میتوانستیم هرنوع ورودی‌ای بدهیم.

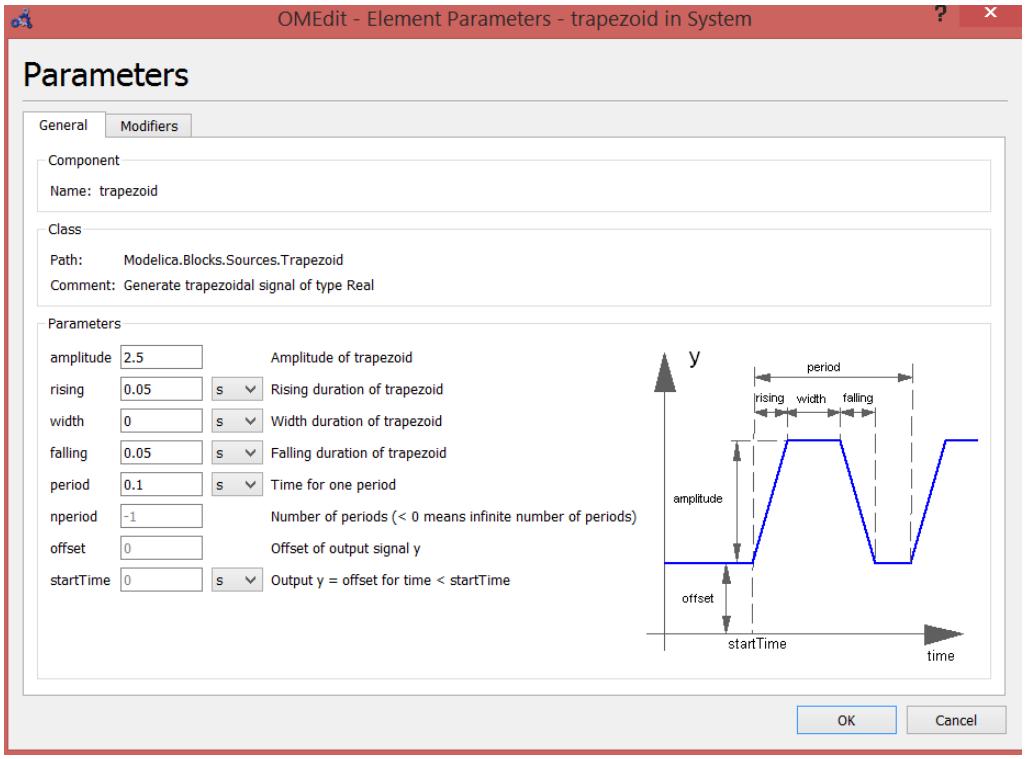
با این کار، خروجی اینبار تغییر یافت. نمودار ورودی و خروجی به شکل زیر است. ورودی به رنگ نارنجی و خروجی به رنگ آبی:



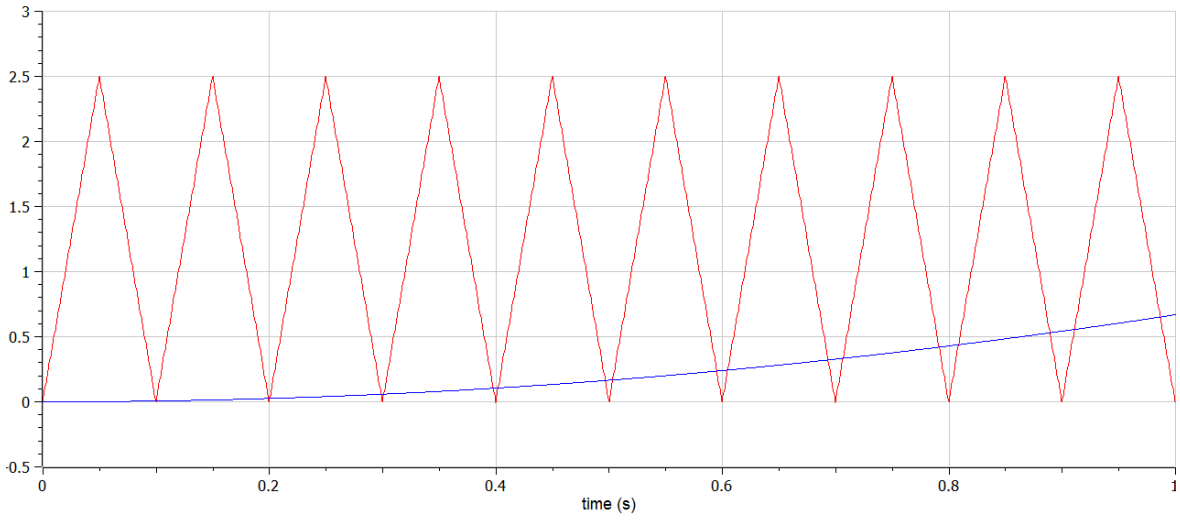
تست سیزدهم:

ورودی پدال را در آزمایشات قبلی به صورت مثلث قائم‌الزاویه یا اره‌ای میدادیم اما باید متساوی‌الساقین باشد. در OpenModelica چنین تابعی نداریم ولی یک راهکاری پیدا شد.

ورودی را بـه صـــورت Trapezoid تعریف کردیم، اما مدت زمان پهنای Trapezoid را برابر صـــفر قرار داده تا عملا یک مثلث تشکیل شود.



ورودی مثلثی و سایر موارد را تغییر ندادیم. خروجی به شکل زیر است. ورودی با رنگ نارنجی و خروجی به رنگ آبی:

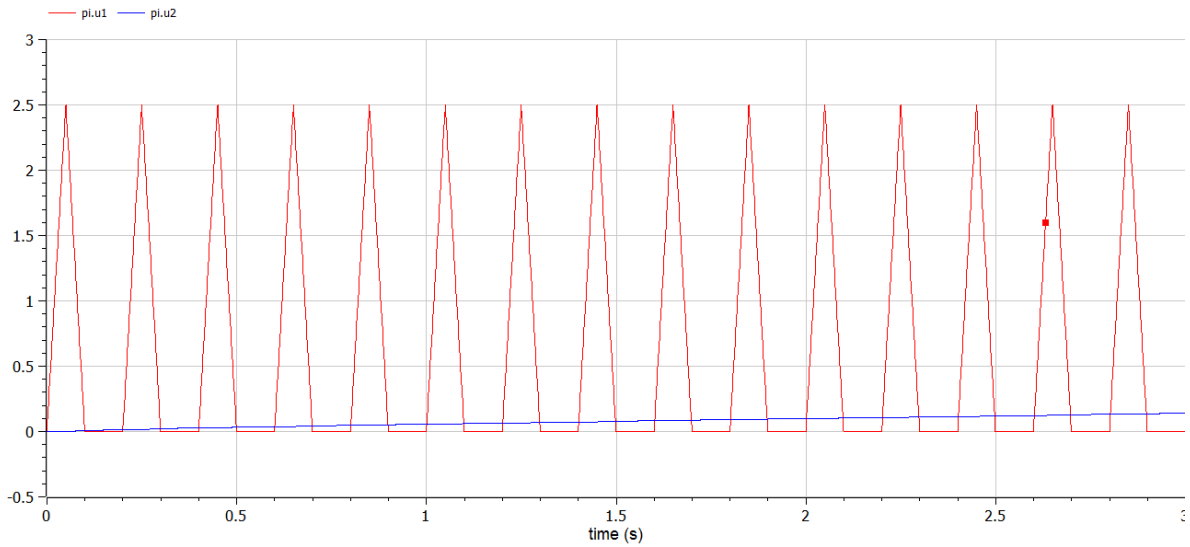


خروجی با تست قبلی تفاوتی نکرد.

چندین تست انجام شد و به نتایج زیر رسیدیم:

Proportional Gain را طبق ویدیو رفرنس برابر 2.5 و Int Gain را برابر 8 قرار می‌دهیم. همچنین در تست‌ها متوجه شدیم که مشکل اینکه خروجی تا بی‌نهایت در حال زیاد شدن است، از طراحی Throttle Body است. بنابر دلایلی، هرنوع ورودی‌ای که وارد Throttle body میشد، همواره تا بینهایت زیاد میشد. از آنجا که طراحی شبیه مدل سیمولینک انجام شد، احتمال دادیم که OpenModelica جور دیگری سیستم را شبیه‌سازی میکند.

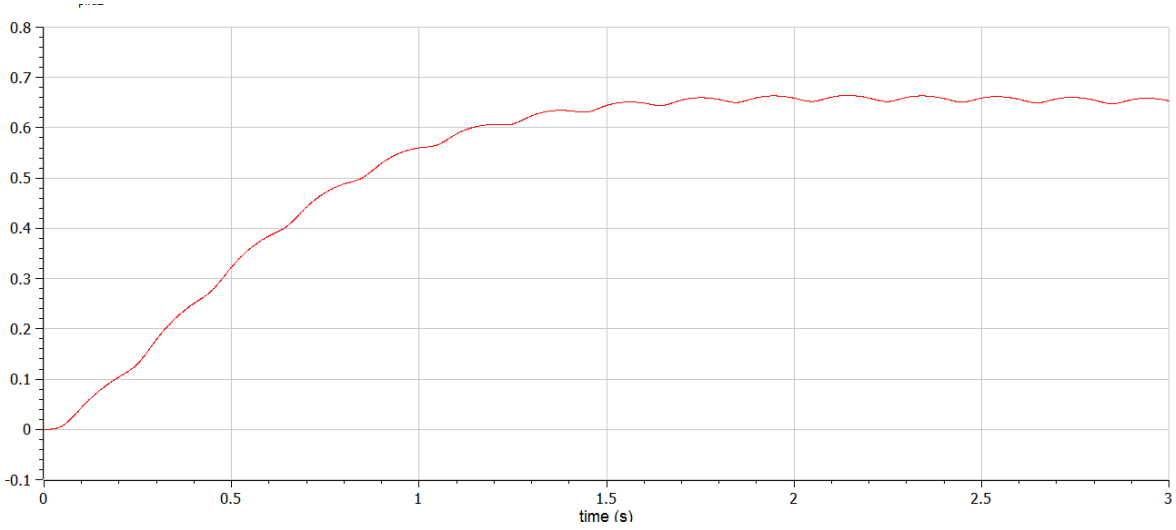
فنر و Damper را بعد از چند تست متوجه شدیم که اگر یک سر آن به یک محیط fixed متصل باشد، خروجی قابل قبول‌تر است. همچنین ثابت فنر را برابر 100 گذاشتیم. همچنین ورودی را مثلث‌هایی با فاصله از هم گذاشتیم. شکل به صورت زیر است. ورودی به رنگ نارنجی و خروجی به رنگ آبی:



مشخص است که خروجی یک حدی در بالا دارد و مانند خروجی قبل نیست. حال باید متغیرها را دوباره عددگذاری‌های مختلف کرد تا شکل بهتری دربیاید.

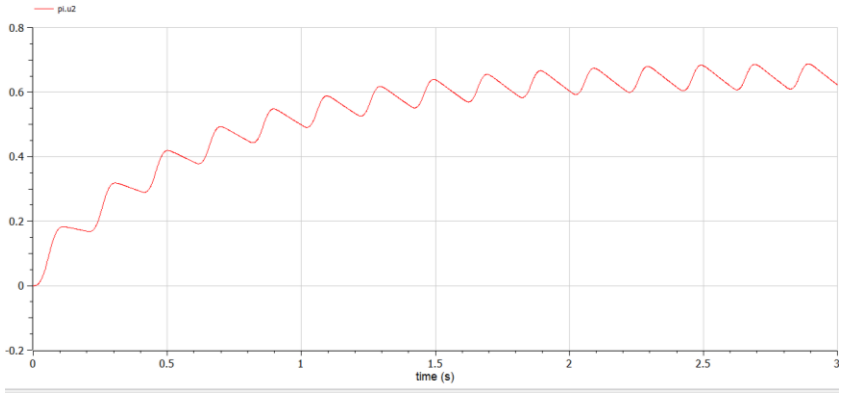
تست بعدی:

Prop gain را برابر ۱۶ و int gain را برابر ۸. همچنین ثابت فنر برابر 10 و ثابت دمپر برابر 10. برای درک بهتر خروجی، فقط خروجی را نمایش میدهیم که دوباره شکل قابل قبول‌تری به خود گرفت.

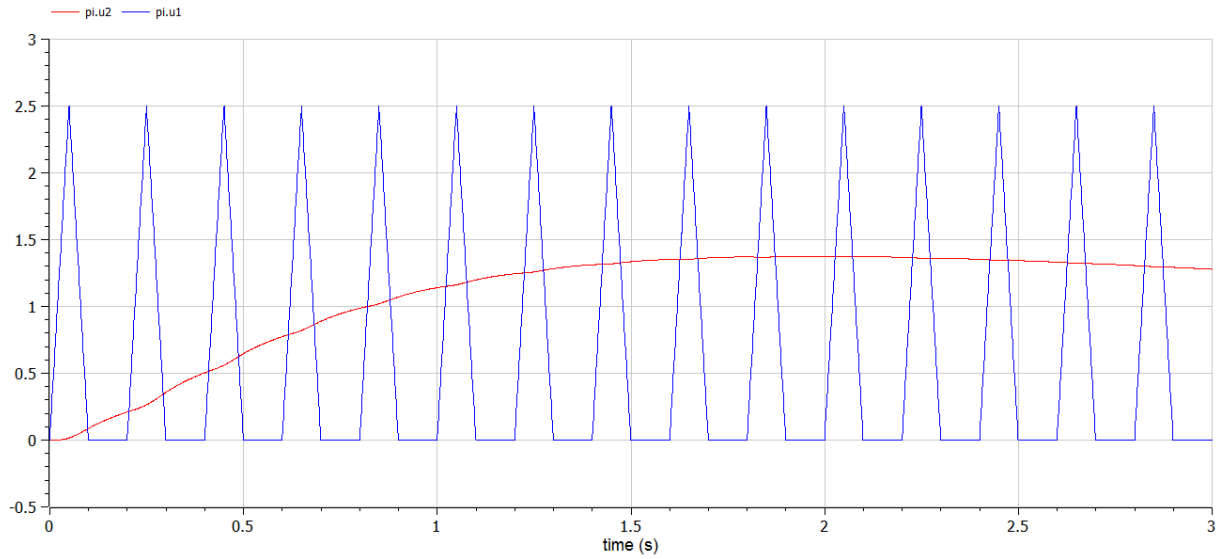


این شـکل به دقت خروجی ویدیو رفرنس نیست اما قابل مشاهده است که صرفا متغیرهای عددی همانند ثابت فنر یا Prop Gain مناسبی نیاز است تا شکل بهتر شود؛ وگرنه منطق سیستم درست است.

در شکل بعدی، ثابت inertia را از 2 به 0.1 تغییر دادیم و خروجی به این شکل شد:

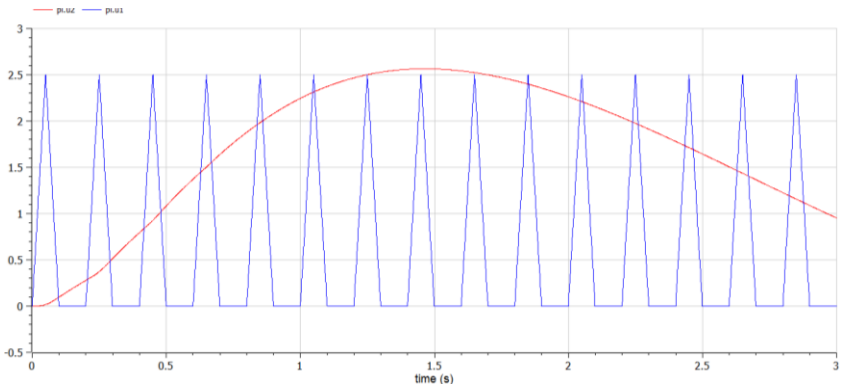


تست بعدی:

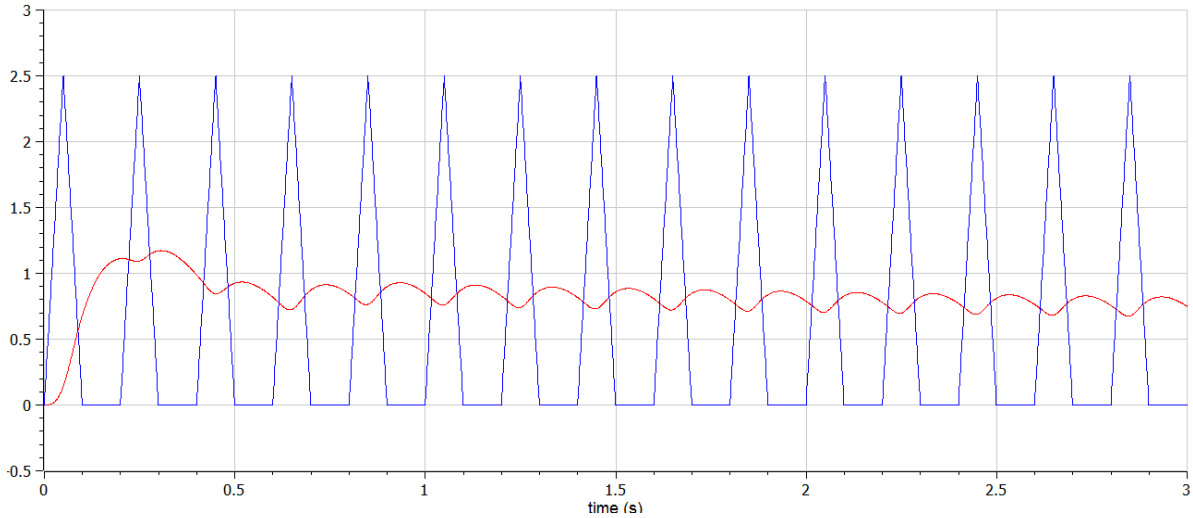


با چندین تغییر در متغیرها و چندین تســت مختلف، شــکل بالا به وجود آمد. به مرور زمان حتی خروجی کم نیز شــد. در این مثال ثابت فنر و inertia برابر 1 و ثابت damper برابر 5 است.

اگر ثابت damper هم 1 بگذاریم، شکل به صورت زیر میشود:

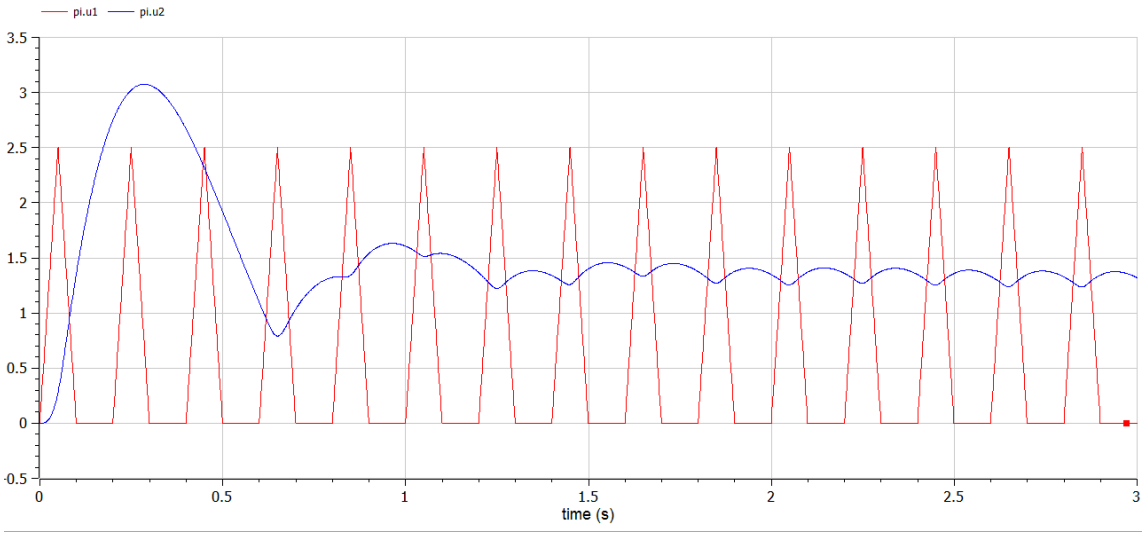


ان شکل خوبی‌ست چون خروجی میخواستیم از ورودی هم بالا بزند ولی کم کم در همان حوالی ماکسیموم ورودی باقی بماند. این شکل ولی به مرور زمان در حال کاهش است پس باید دوباره متغیرهارا عوض کنیم.

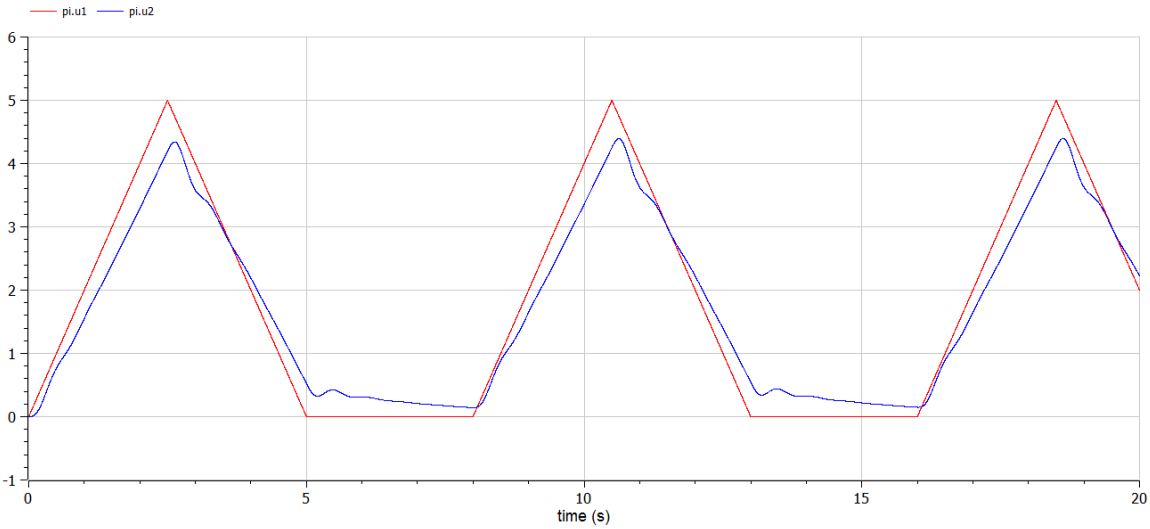


ثابت فنر 5 و ثابت inertia 0.1 و شکل به صورت بالا در آمد.

بعد از ده‌ها تست، نزدیک‌ترین شکل به شکل ویدیو رفرنس را به‌دست آوردیم. Int gain برابر 5 و prop gain برابر 60 و ثابت فنر برابر 5 و ثابت inertia برابر 0.2 و ثابت damper برابر 1 گذاشته شده است. شکل به صورت زیر است. ورودی با رنگ نارنجی و خروجی با رنگ آبی:



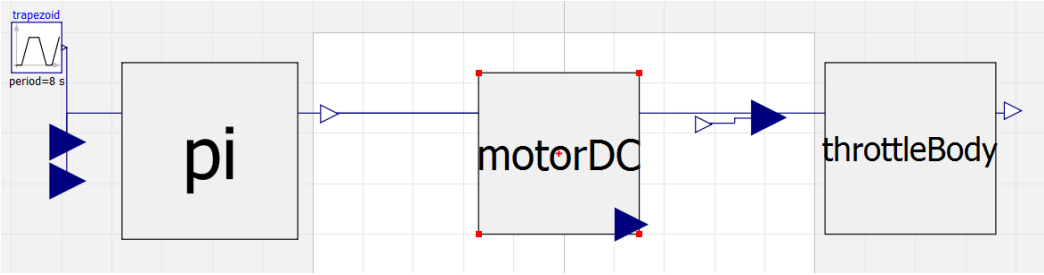
اما یک مشـکلی داشتیم. باید شبیه‌سازی مثل دنیای واقعی باشد. نمیتوانیم به این سرعت با روی پدال بگذاریم و برداریم. پس زمان شـبیه‌سـازی را بیشـتر کردیم و بین فشار دادن گاز فاصله انداختیم. آخرین تست ما به صورت زیر است و متغیرها از دیتاشیت قابل مشاهده می‌باشد.



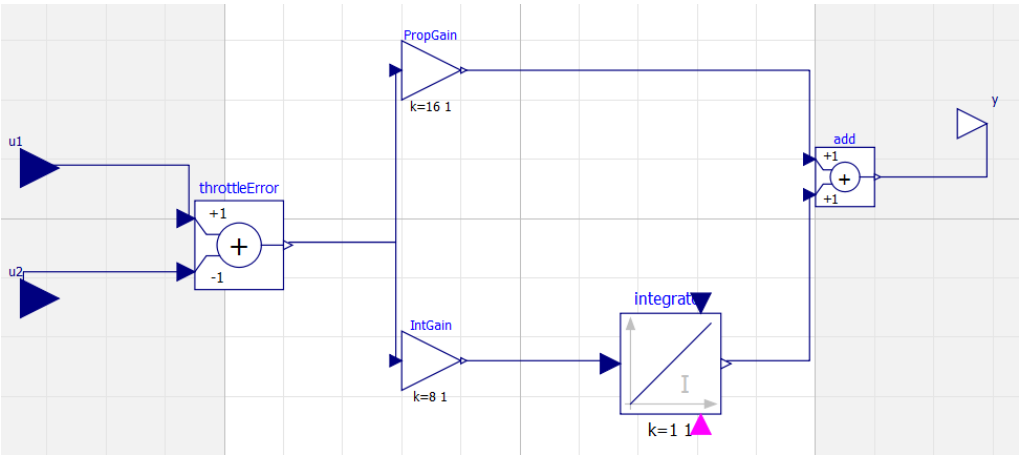
کد برنامه

کد برنامه کنترل دریچه گاز الکترونیکی که به زبان Modelica نوشته شده است، در گیتهاب در دسترس عموم قرار گرفته است.

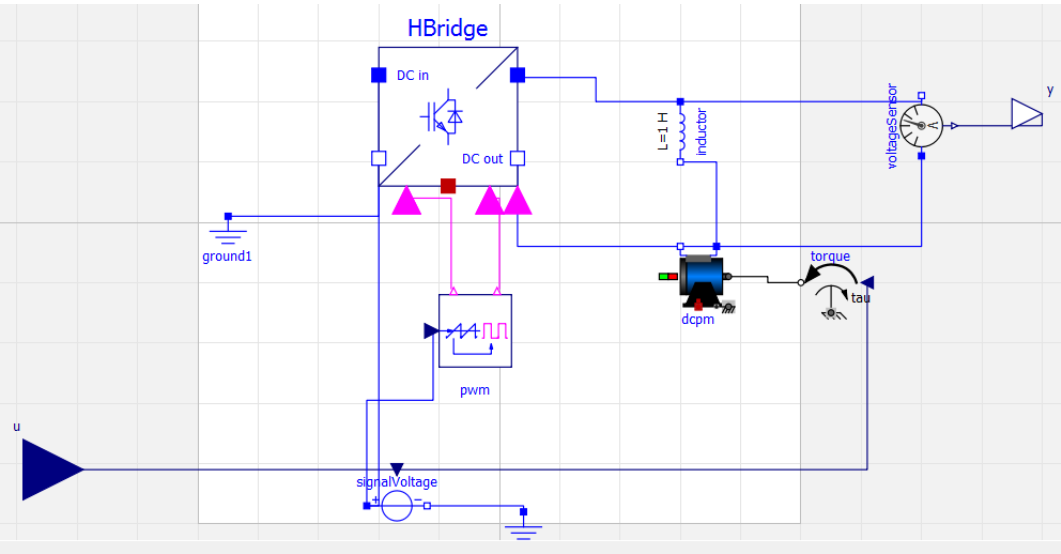
ورژن نهایی سامانه را در این بخش نشان خواهیم داد.



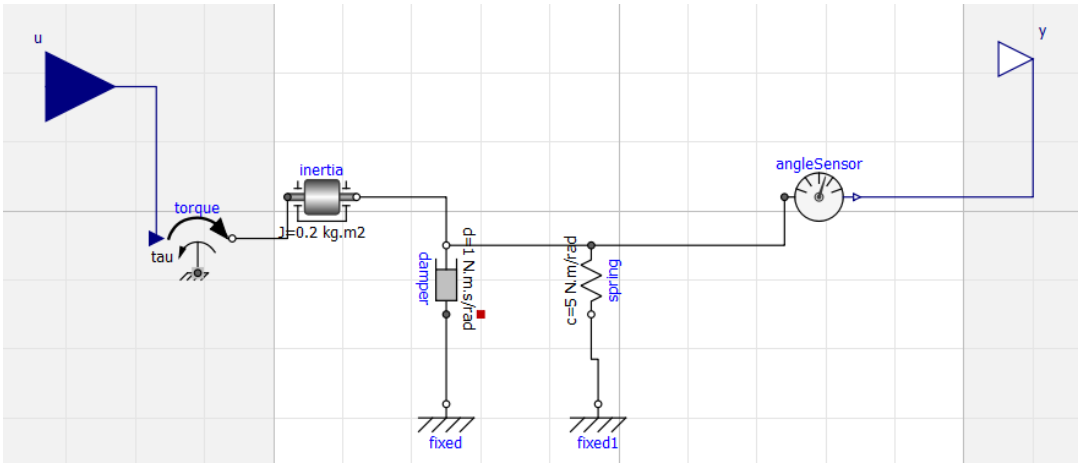
ورژن نهایی سامانه (سیستم)



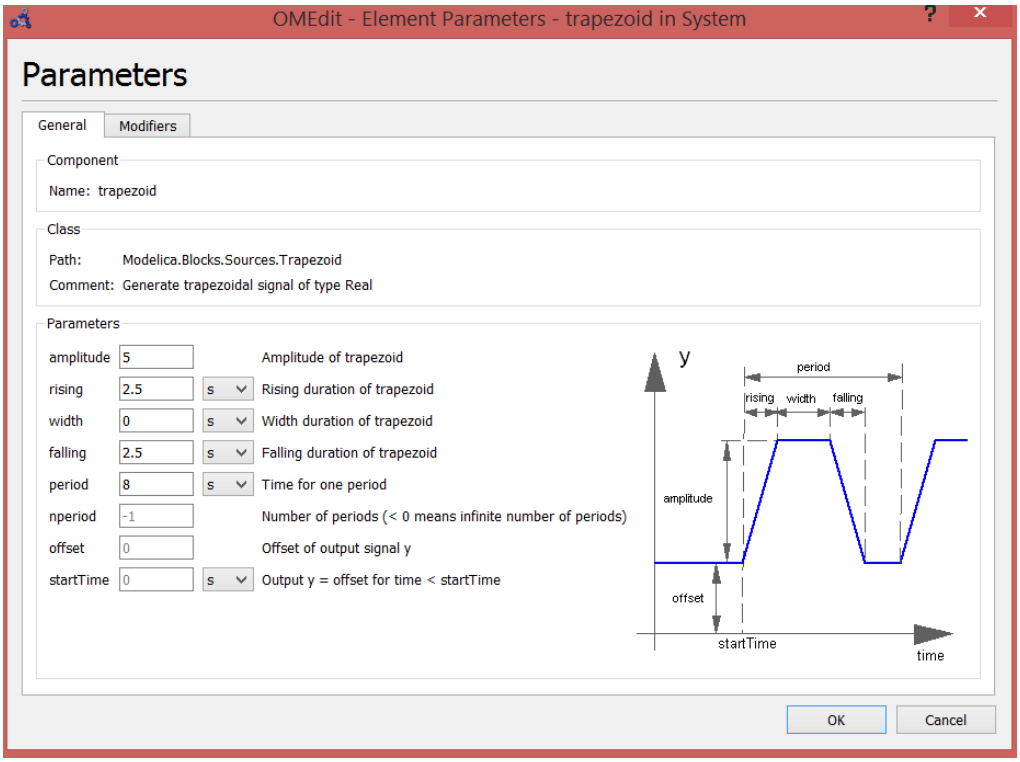
ورژن نهایی کنترلر PI



ورژن نهایی موتور DC



ورژن نهایی بدنه دریچه گاز



تنظیمات پدال گاز

قیمت

این بخش اهمیت و مزیت شبیه‌سازی را نشان خواهد داد. در دنیای واقعی، ساخت چنین سامانه‌ای و تست و آزمایش آن، فوق‌العاده گران و زمان‌بر می‌شـد. با نرم‌افزار OpenModelica که خود رایگان و متن‌باز است، میتوانیم هر سامانه‌ای که می‌خواهیم را تا حد قابل قبولی طراحی کنیم. تنها هزینه‌ای که این پروژه می‌تواند برای شما داشه باشد، هزینه دانلود نرم‌افزار بوده که نسبت به طراحی چنین پروژه‌ای در دنیای واقعی، بسیار بسیار ناچیز است.

جدا از هزینه مالی، هزینه زمانی شـما هم تا حد خوبی کاهش میابد. البته این هزینه زمانی کم، بدی هم دارد. نتیجه شـبیه‌سـازی قطعا با واقعیت فرق دارد ولی یک دید کلی به شـما خواهد داد. ممکن است بعضی موارد را نتوان شبیه‌سازی کرد (به دلیل امکانات ناکافی شبیه‌ساز یا ...). به عنوان مثال، این پروژه نیازمند قطعه‌ای با نام Hard Stop بود اما چنینی قطعه‌ای در OpenModelica موجود نبود. اما به طور کلی، استفاده از شبیه‌سازی کاملا ارزشش را دارد.

راه‌حل مشکل بالا، استفاده از شبیه‌سازهای قوی‌تر است. حتی اگر این شبیه‌سازها رایگان نباشند، همچنان ارزان‌تر از ساخت سامانه در دنیای واقعی تمام می‌شود.

جمع‌بندی

در این پروژه بـه طراحی ســـامانه کنترل دریچه گاز پرداختیم. از آنجا که طراحی چنین ســـامانه‌ای در دنیای واقعی، هزینه مالی و زمانی زیادی دارد و امکان تســت و خطا تا حدی کم اســت، از نرم‌افزار OpenModelica اســتفاده شــد. این نرم‌افزار، مخصــوص شبیه‌سازی بوده و از زبان Modelica استفاده می‌کند.

این پروژه نیازمند درک نحوه کار دریچه گاز و همچنین نحوه طراحی یک شبیه‌سازی از آن با کمک OpenModelica می‌باشد.

سامانه کنترل دریچه گاز الکترونیکی از سه بخش تشکیل شده است:

- کنترلر PI
- موتور DC
- بدنه دریچه گاز

برای کنترل بهتر روی ســامانه، پروژه به صــورت Closed-Loop طراحی شد. درواقع خروجی بدنه دریچه گاز به عنوان فیدبک به ورودی داده می‌شود. ورودی دیگر، طبیعتا پدال گاز است.

با شبیه‌ساز OpenModelica ضمن طراحی پروژه، نمودار ورودی و خروجی آن را نیز می‌توان مشاهده نمود.

با کمک Simulation Setup، هرنوع شــبیه‌ســازی مدنظر شما قابل پیاده‌سازی است و می‌توان با تغییر متغیرات، سامانه را Re-Simulate کرد و نتایج جدید را در نمودار مشاهده کرد.

در تست‌های انجام گرفته، به این نتیجه رسیدیم که باید ورودی پدال به صورت مثلثی باشد و فاصله و زمان لازم برای فشردن پدال نیز باید درنظر گرفته شود.

همچنین اعداد به دست آمده همچون ثابت فنر، جنبه محاسباتی نداشته و صرفا از تست فراوان و رسیدن به شکل موردنظر به دست آمده است.

از آنجا که نحوه اتصالات و برخی بلوک‌ها با سایر نرم‌افزارهای شبیه‌سازی متفاوت است، ممکن است نتایج مشابهی با سایر نرم‌افزارها به دست نیاید. همچنین برخی بلوک‌ها مانند hard stop که در سیمولینک موجود است، در OpenModelica وجود ندارد و مشخصا در طراحی سامانه ما به‌کار نرفته است.

برای آشنایی بیشتر با این پروژه و محیط OpenModelica و نحوه کار با آن می‌توان به ویدیو مشخص شده مراجعه نمود. این ویدیو توسط تیم طراح این پروژه ضبط شده است و به زبان انگلیسی می‌باشد.

همچنین فایل‌های مرتبط با پروژه نیز همانطور که قبلا ذکر شد در گیتهاب پروژه قابل دسترسی است.