
بسم الله الرحمن الرحيم

گزارش پروژه

(بخش اصلی)

عنوان پروژه:

کنترل سیستم سینگل کوپتر

عنوان درس:

طراحی سیستم های کنترل

استاد درس:

دکتر زارعی نژاد

تدریس‌یار:

مهندس استرکی

دانشجو:

فرزام جهانی ۹۷۲۶۰۱۵

مصطفی کویری ۹۷۲۸۰۷۶

نیمسال دوم ۱۴۰۱ – ۱۴۰۰

چکیده

با توجه به اهمیت فعالیت های فنی و عملی در علم کنترل و همچنین به منظور استفاده از مباحث تئوری دروس کنترلی برای سیستم های واقعی و فیزیکی، این پروژه تعریف شده است تا از مباحث آموخته شده در کنترل، برای شناسایی سیستم های واقعی و طراحی کنترلر مناسب استفاده کرده و به صورت عملی و با استفاده از بورد اردوینو، کنترل سیستم انجام شود.

واژه‌های کلیدی

کنترل، کنترلر، بورد اردوینو

فهرست مطالب

۱-
فصل اول
مقدمه

مقدمه

سیستم انتخابی این پروژه برای پیاده سازی کنترلر، با نام سینگل کوپتر شامل یک موتور،درایور، میله رابط، ملخ، سنسور زاویه، بورد اردوینو میباشد.

استراتژی کاری این سیستم به این صورت است که ملخ بر روی موتور سوار شده و موتور به کمک میله رابط به سنسور زاویه وصل شده و انتهای آن به لبه ی میز فیکس میشود.

پس از برقراری ولتاژ، درایور موتور را به حرکت در آورده و با گردش ملخ، میله به سمت بالا حرکت میکند. هدف ما کنترل زاویه میله است به نحوی که از حالت آزاد به یک زاویه مطلوب با اورشوت ماکزیمم ۱۵٪ و خطای ماندگار کمتر از ۳۰٪ برسد.

۲-

فصل دوم

ساخت و مونتاژ پلنت

داده‌ها و اطلاعات

همانطور که اشاره شد، روند ساخت پلنت به این صورت است:

ابتدا ملخ را در شفت موتور قرار داده و آن را فیکس میکنیم سپس به کمک پرینتر سه بعدی باید بستی را به صورت دو تکه پرینت کنیم تا موتور و میله رابط را با زاویه نود درجه بهم بچسبانیم.

انتهای میله رابط هم باید به سنسور زاویه(پتانسیومتر یا انکودر) وصل شده و سنسور هم به کمک بستی که با پرینتر سه بعدی پرینت گرفته شده، به زمین فیکس میشود

در این پروژه ابتدا به جای ملخ، از دم هواپیمای اسباب بازی استفاده شده بود ولی چون همراه با گیربکس بود ناحیه مرگ سیستم بالا بوده(حدود ۸ ولت) که تا حداکثر ولتاژ القایی ممکن به موتور(۱۲ ولت) فاصله زیادی ندارد که این هم کنترل سیستم را تقریبا غیر ممکن میکنددر نتیجه سیستم مذکور با سیستم دیگری جایگزین شد.

موتور جایگزین شده در این سیستم، یک موتور دی سی ۳.۷ ولت است و درایور موتور drv۸۸۳۳ میباشد.

رویکرد ابتدایی برای گرفتن فیدبک سیستم استفاده از یک انکودر دوفازی ۴۰۰ پالس بود که این رویکرد به دلیل نوع خروجی سنسور و عدم کفایت کلاک میکرو پروسسور برای خواندن پالس های خروجی کنار گذاشته شد.

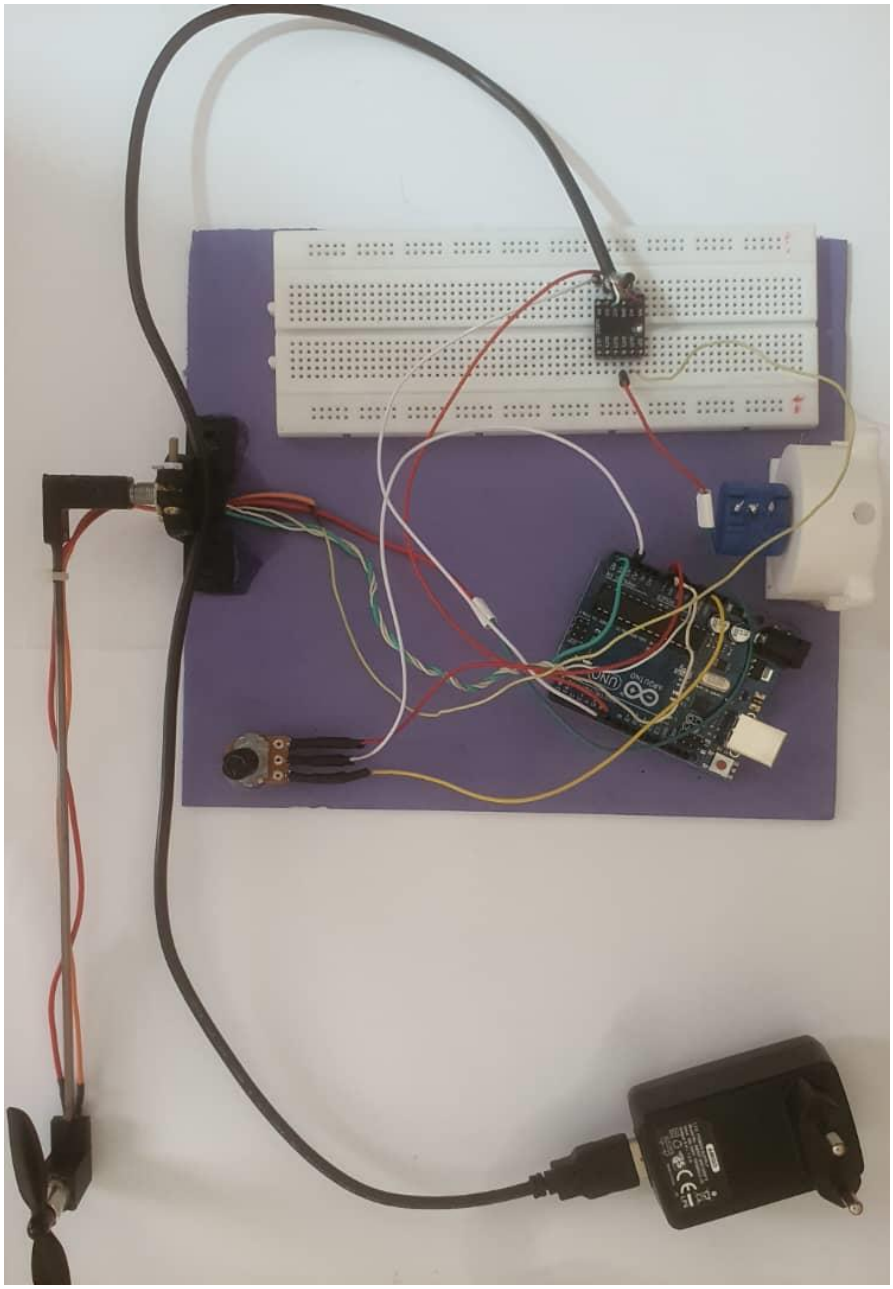
درنتیجه برای فیدبک، تصمیم گرفتیم که از پتانسیومتر استفاده کنیم که خروجی آنالوگ میدهد.

در شکل زیر تصویری از انکودر و دم هلیکوپتر اشاره شده قرار گرفته است.

شکل ۱ دم هلیکوپتر مدل مذکور

شکل ۲ انکودر اشاره شده

در شکل زیر تصویری از پلنت نهایی را میبینید.

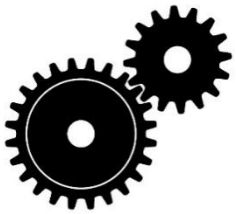


شکل ۳ پلنت نهایی به همراه ملخ، دم، موتور، میکروپروسسور، ولوم، درایور و منبع تغذیه

۳-

فصل سوم

شناسایی سیستم



روند شناسایی

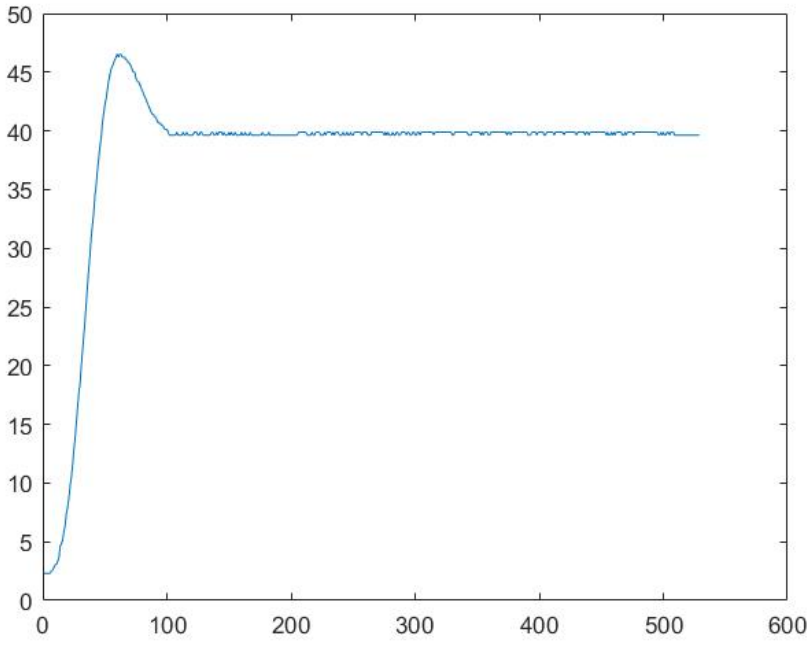
برای شناسایی سیستم به صورت تجربی، باید ورودی و خروجی هایی از آن داشته باشیم حال اگر ورودی فرکانس متغیر و به صورت chirp signal باشد، شناسایی سیستم بسیتر دقیقتر خواهد بود ولی در این پروژه متناسب با نیاز ما، با ورودی پله و سینوسی و گرفتن خروجی از آن سیستم را شناسایی میکنیم.

به این صورت که طبق کد اردوینو یک ورودی سرعت ثابت به پلنت اعمال شده و خروجی زاویه از طریق پتانسیومتر و با کمک excel data streamer ثبت میشود.

سپس داده های جمع آوری شده را در فضای workspace متلب به صورت ماتریس ذخیره کرده و از طریق system indentification toolbox متلب، تابع تبدیل پلنت به صورت تخمینی در فضای لاپلاس و فضای زد به دست می آید.

```
clc;clear;close all
T = csvread('test_final.csv',12);
% Rdata=T(:,1)
for ii=1:529
input(ii)=160
end
input=input'
```

شکل ۴ کد متلب مربوطه



شکل ۵ خروجی پاسخ ضربه سیستم

در قسمت بالا هم کد های مربوط به system identification و خروجی گرفته شده از پاسخ ضربه سیستم قرار گرفته است.

۴-
فصل چهارم
کنترل سیستم

پس از شناسایی سیستم، تابع تبدیل پلنت به صورت زیر است:

$$G(s) = \frac{-0.858s + 9.808}{s^2 + 6.4s + 39.44}$$

$$G(z) = \frac{0.0009882z}{z^2 - 1.933z + 0.9372}$$

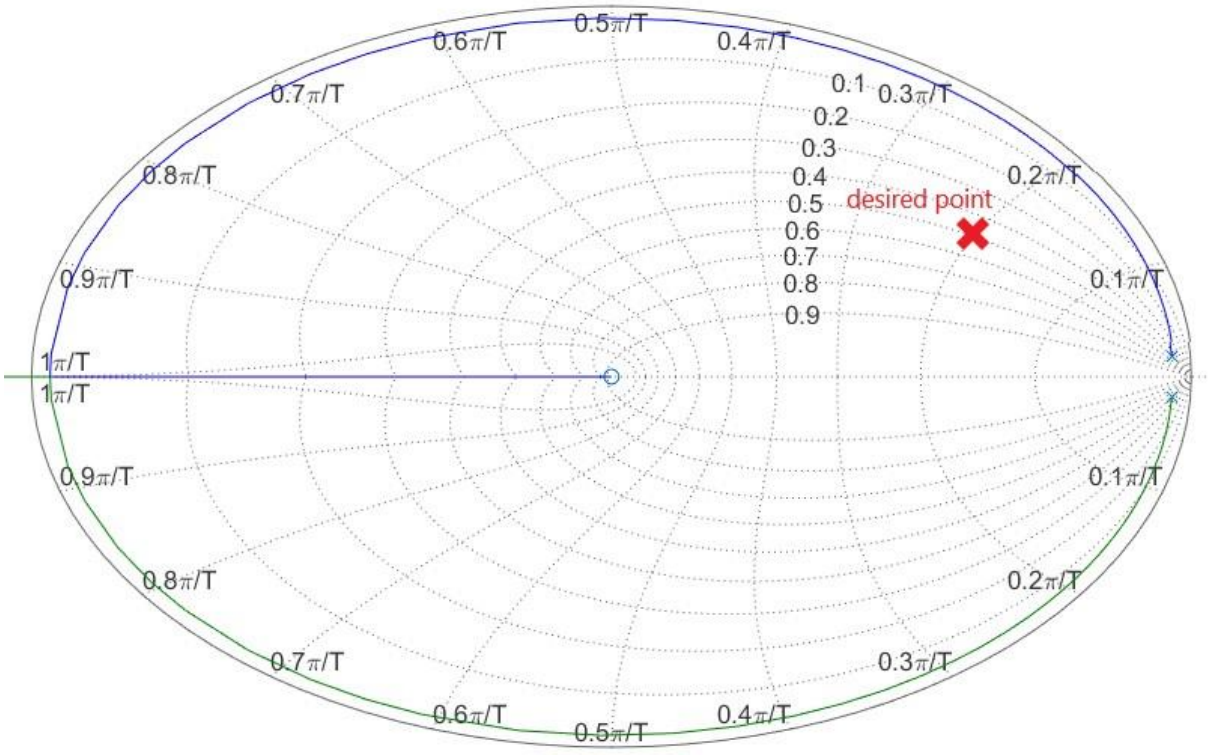
مطلوب ما برای طراحی کنترلر، اورشوت به ورودی پله ۱۵٪ و ثابت خطای استاتیکی بزرگتر از ۱ میباشد.

$$\text{desired} \begin{cases} overshoot = ۱۵٪ \rightarrow \zeta = ۰.۵ \\ ess < ۰.۱ \rightarrow k_p > ۹ \end{cases}$$

به کمک متلب نمودار روت لوکاس این تابع تبدیل را رسم میکنیم. قطب مورد نظر ما باید روی خط $\zeta = ۰.۵$ قرار داشته باشد

```
clc;clear;close all
T = csvread('forjavab۱.csv',۱۷);
% Rdata=T(:,۱)
time=۰:۱:۹۲۵;
time=(time*۰,۰۱)';
for ii=۱:۹۲۶
input(ii)=۱۵۰
end
input=input'
%% z-grid

g = tf([۰,۰۰۰۹۸۸۲ ۰],[۱ -۱,۹۳۳ ۰,۹۳۷۲],.۰۱)
rlocus(g)
zgrid
```



شکل ۶ روت لوکاس سیستم جبران نشده

طبق شرط زاویه، زاویه صفر ها و قطب ها با نقطه مطلوب باید -۱۸۰ شود.

$$\phi_{z1} - (\phi_{p1} + \phi_{p2}) = +۳۰ - (+۱۳۵ + ۱۳۰) = -۲۳۵$$

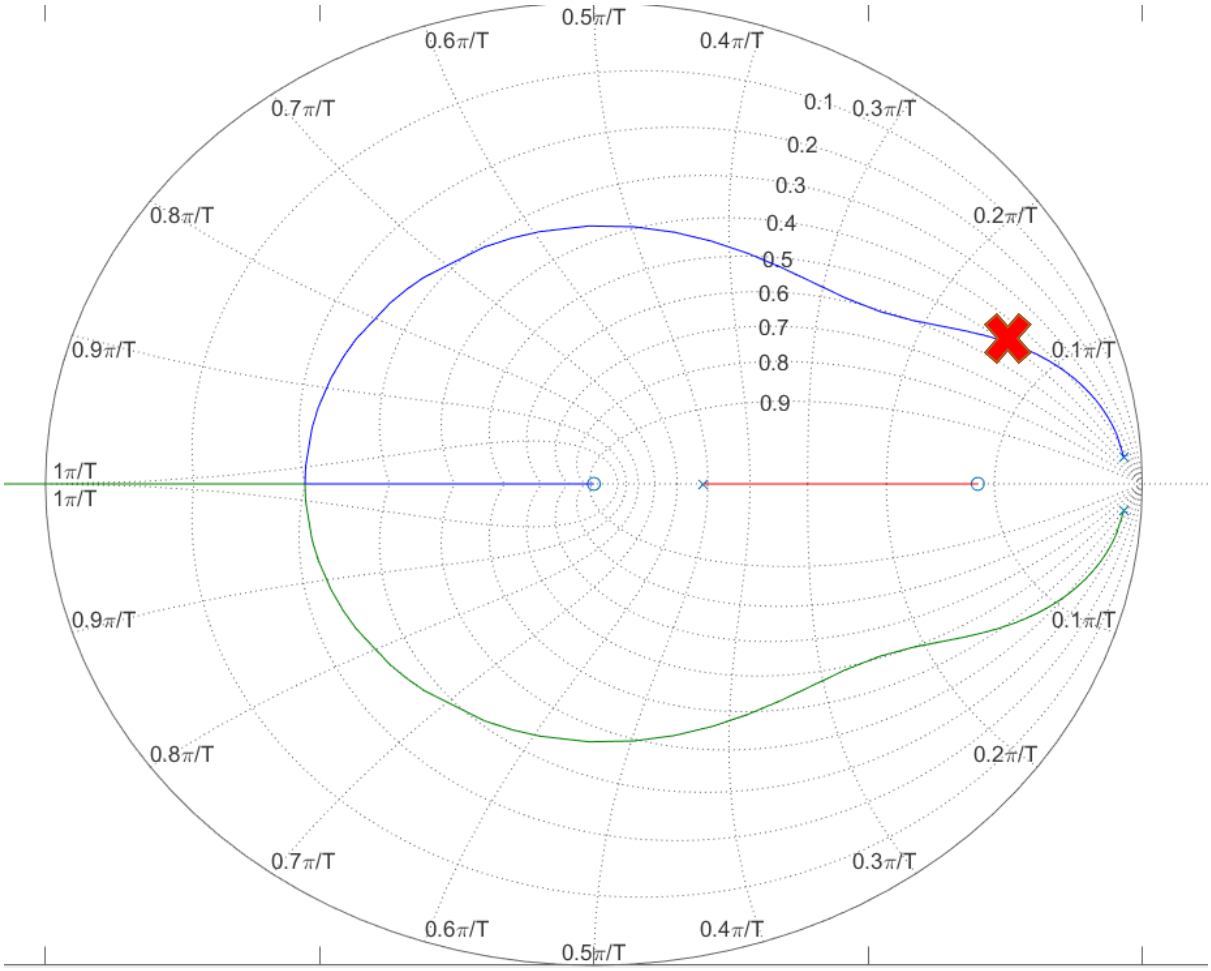
پس باید جبران سازی اضافه شود تا ۵۵+ درجه فاز مثبت اضافه کند مثلا

$$G_c(z) = \frac{z - ۰.۷}{z - ۰.۲} \rightarrow \phi_{zc} - \phi_{pc} = ۱۰۵ - ۵۰ = ۵۵$$

حال به کمک متلب چک میکنیم

```
clc;clear;close all
T = csvread('forjavab۱.csv',۱۷);
% Rdata=T(:,۱)
time=۰:۱:۹۲۵;
time=(time*۰,۰۱)';
for ii=۱:۹۲۶
input(ii)=۱۵۰
end
input=input'
%% z-grid
g = tf([۰,۰۰۰۹۸۸۲ ۰],[۱ -۱,۹۳۳ ۰,۹۳۷۲],.۰۱)
g_c۱=tf([۱ -۰.۹],[۱ ۰,۱],۰,۰۱)
gh=g*g_c۱;
```

```
rlocus(gh)
zgrid
```



شکل ۷ روت لوکاس سیستم جبران شده

پس زتا و اورشوت سیستم تایید میشود حال به محاسبه خطا میپردازیم

$$k_p = \lim_{z\to ۱} ۲ * \frac{z - ۰.۷}{z - ۰.۲} * \frac{۰.۰۰۰۹۸۸۲z}{z^۲ - ۱.۹۳۳z + ۰.۹۳۷۲} = ۰.۱۷۶۴$$

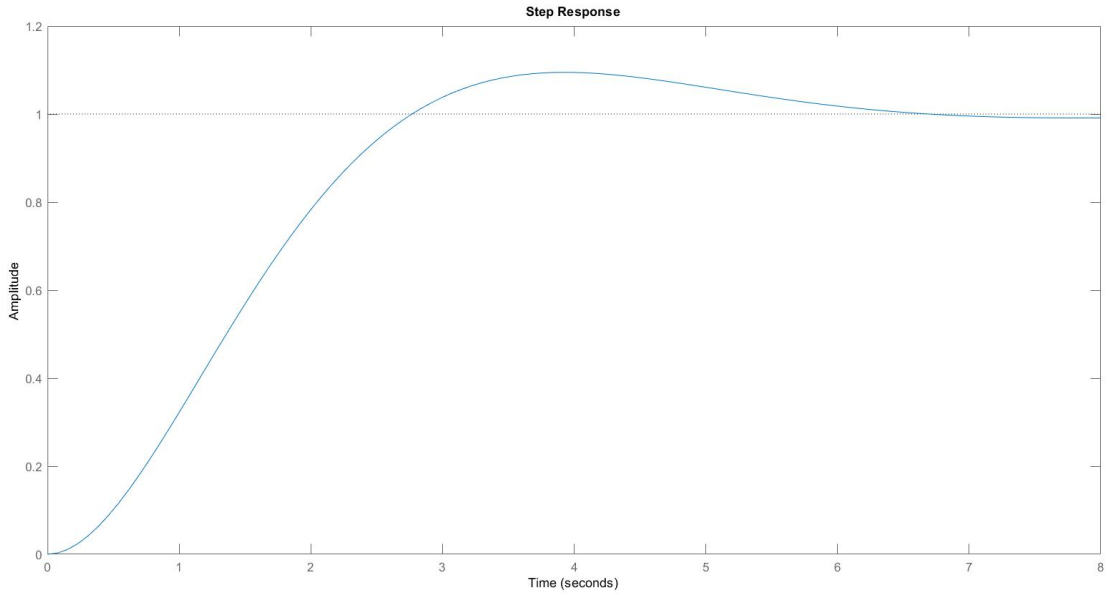
پس سعی میکنیم جبران ساز دیگری را اضافه کنیم تا گین DC حدودا برابر با ۱۰۰ داشته باشد تا ثابت خطای استاتیکی ۱۰۰ برابر شود :

$$G_{c2}(s) = \frac{s + ۰.۱}{s + ۰.۰۱} \quad \rightarrow \quad G_{c2}(z) = \frac{z - ۰.۹}{z - ۰.۹۹۹}$$

پس جبران ساز نهایی برابر است با:

$$G_c(z) = ۲ * \frac{z - ۰.۹}{z + ۱} * \frac{z - ۰.۹}{z - ۰.۹۹۹}$$

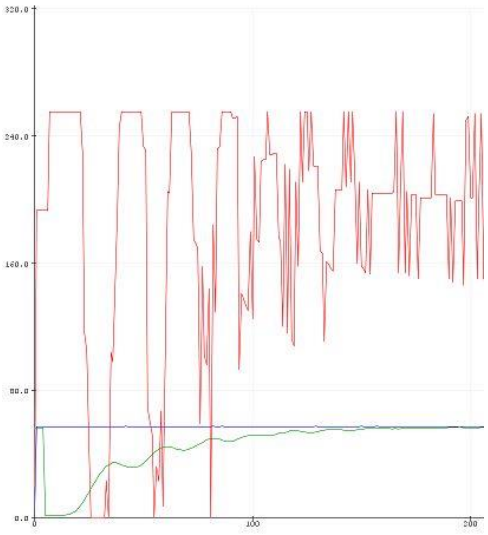
با یک ورودی پله، خروجی سیستم کنترل شده را چک میکنیم



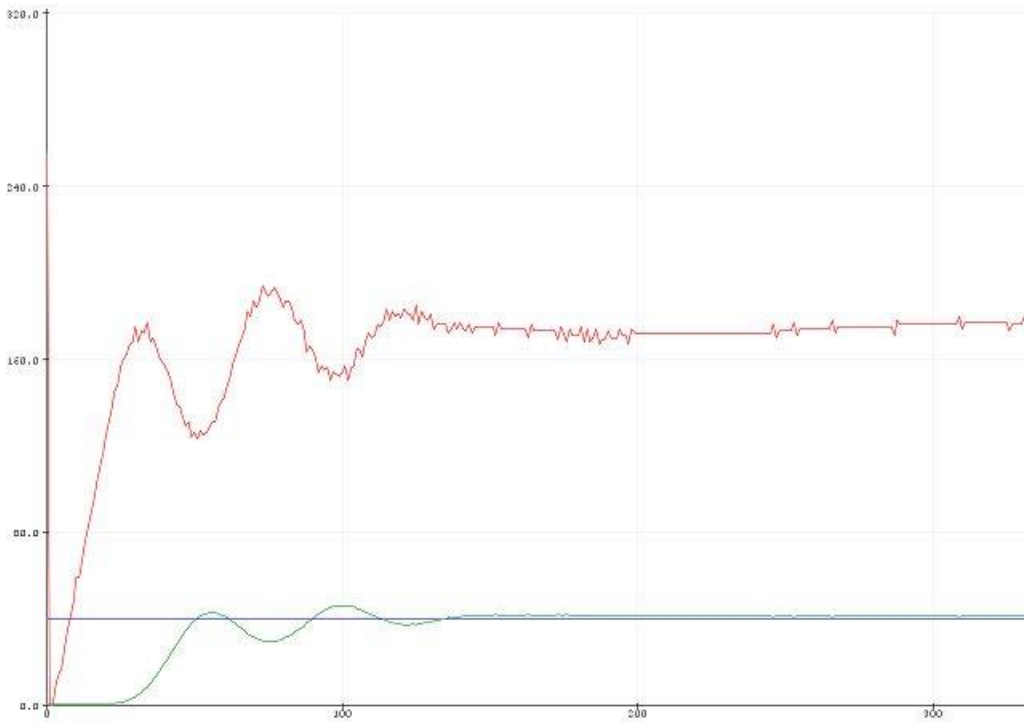
شکل ۸ پاسخ پله سیستم جبران شده به شکل تئوری

پس از تبدیل تابع تبدیل جبرانساز به فرم معادلات تفاضلی آن را روی کد آردوینو قرار داده و سیستم را کنترل میکنیم.

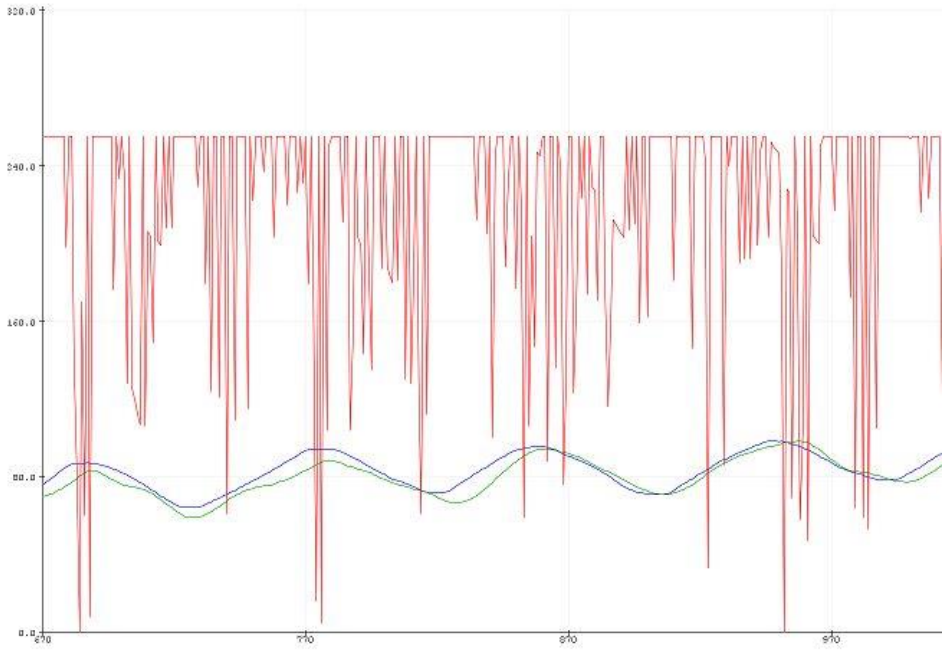
نتایج کنترلرهای لید-لگ و pid(تیون به صورت دستی) در شکل‌های زیر آمده است.



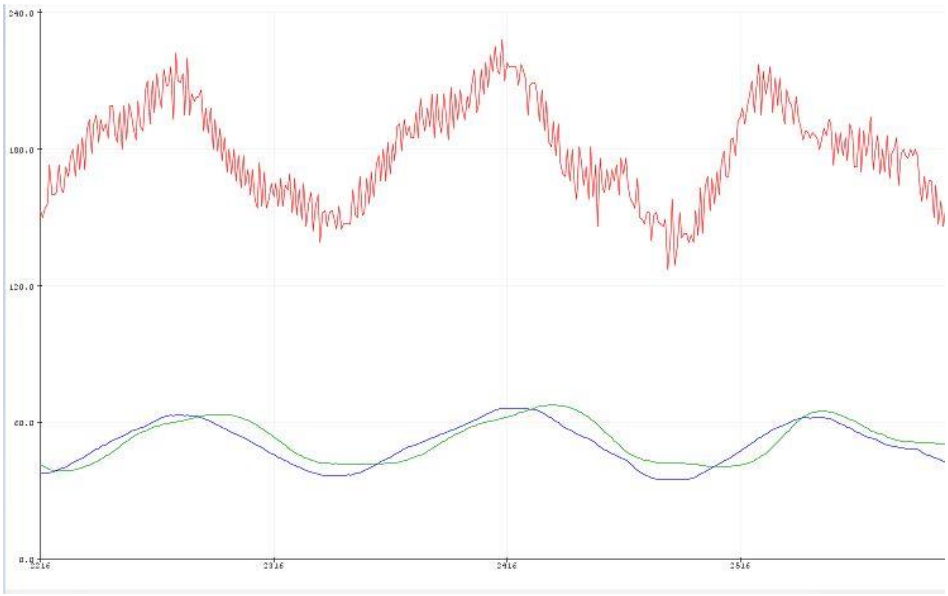
شکل ۹ پاسخ پله با pid پلنت تجربی



شکل ۱۰ پاسخ پله لید-لگ پلنت واقعی تجربی



شکل ۱۱ تعقیب نوسانی با pid



شکل ۱۲ تعقیب نوسانی با لید-لگ

تنظیمات و کد نهایی میکروکنترلر در زیر آمده است.

```
#define moto_pin_1 5
#define moto_pin_2 6

#define sensor A5

#define volum A3

boolean flag = true;
float k = 0;

float volumDesire = 0;

void setup() {
  Serial.begin (9600);

  pinMode(moto_pin_1, OUTPUT);
  pinMode(moto_pin_2, OUTPUT);
  pinMode(sensor,INPUT);
  pinMode(volum,INPUT);

}
float reads;
float degree;
int pwm = 0;
int desire;
float error = 0;
float last = 0;
int i=0;
float yn_2 = 0;
float yn_1 = 0;
float error_2 = 0;
float error_1 = 0;
```

شکل ۱۳ تنظیمات اردوینو

```
void loop() {
  if(flag){
    delay(5000);
    }
  flag = false;
  reads = analogRead(sensor);
  degree = (reads-248)*.2557+30-57.36 ;
  volumDesire = (float)analogRead(volum)*180/(float)1023;
  desire = volumDesire;
  error = desire - degree;
  i+=error;
 //pwm = 8*error +2*(error-last)/.01+0.08*i;
  k = 10;
  pwm = ceil(k*(error - 1.8 * error_1 + .81 * error_2) + 1.099 * yn_1 - .0999 * yn_2);
  //pwm = 160;
  if(pwm > 255){
    pwm = 255;
  }
  if(pwm < 0){
  pwm = 0;
  }
  yn_2 = yn_1;
  yn_1 = pwm;
  error_2 = error_1;
  error_1 = error;
  last = error;
  Serial.print(volumDesire);
  Serial.print(",");
  Serial.print(pwm);
  Serial.print(",");
  Serial.println(degree);
  motor(pwm);
  delay(10);
```

شکل ۱۴ کد نهایی آردوییِنو