به نام خداوند

گزارش پروژه کنترل سرعت موتور DC

محمد امین محتشمی
امیر علی آقامحمدی

دکتر افشار

زمستان 1402

مشخصات سیستم:

ما برای کنترل موتور DC نیاز به انکودر(هرنوع انکودری مانند مطلق، افزایشی اثر هال و...) برای گرفتن فیدبک سرعت و یا موقعیت، درایور موتور برای ارسال دستورات از میکروکنترلر به موتور و همچنین میکروکنترلر برای پیاده سازی الگوریتم های مورد نظر داریم.

موتورDC و انکودر:

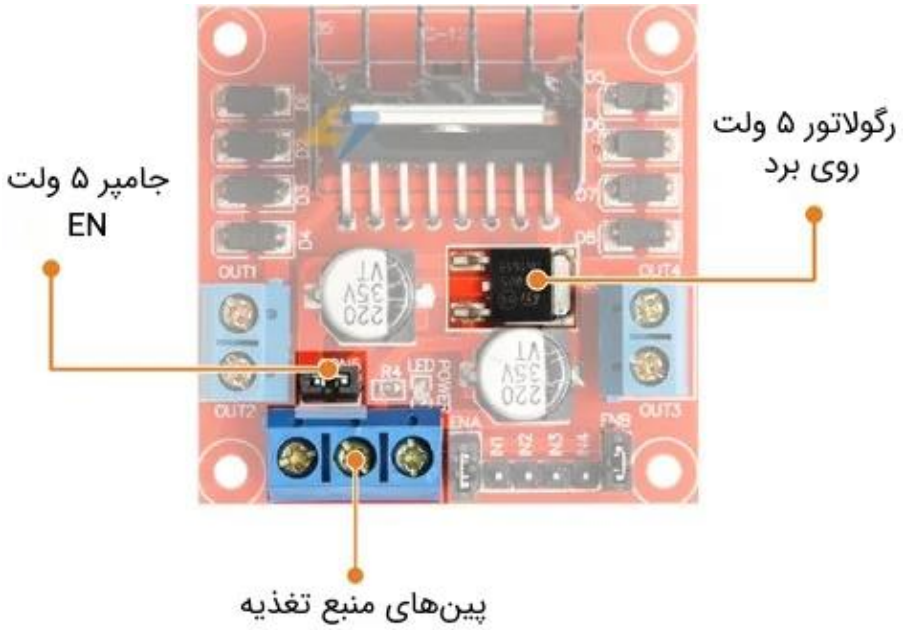
ما از موتور گیربکس انکودر دار YGY6138-R528C که محصول شرکت yuanruixin استفاده می کنیم. جریان این موتور 3 آمپر، ولتاژ12 آمپر و سرعت بدون بار 65 دور بر دقیقه می باشد. همانطور که گفته شد،این موتور با ولتاژ ۱۲ ولت کار میکند. در صورت استفاده از ولتازهای کمتر از ۱۲ ولت؛ دور خروجی، توان و در نهایت گشتاور تولیدی موتور کاهش می یابد. ای موتور دارای یک انکودر اثر هال دو کاناله می باشد که با استفاده از آن می توان موقعیت و سرعت موتور را تعیین کرد. این موتور با چرخش یک دور شفت خروجی، 1080 پالس به ما خروجی می دهد.

درایورL298 :

ماژول درایور موتور L298N از طریق 3 پین ۳،۵ میلی‌متری تغذیه می‌شود و شامل پین‌هایی برای منبع تغذیه موتور(Vs) ، زمین و منبع تغذیه 5 ولت (Vss) است.

تذکر :آی‌سی درایور 298 Nدر واقع دارای دو پایه توان ورودی است "Vss" :و ."Vs" پل اچ از پین Vs توان خود را برای هدایت موتورها دریافت می‌کند. ولتاژ این پین می‌تواند 5 تا 35 ولت باشد. ولتاژ Vss برای تغذیه مدارهای منطقی است که مقدار آن ممکن است بین 5 تا 7 ولت باشد. هر دوی این ولتاژها زمین مشترکی به نام "GND" دارند.

ماژول تصویر زیر، دارای یک رگولاتور ولتاژ 78M05 با مقدار ۵ ولت است که سازنده آن اس‌تی‌مایکروالکترونیکس (STMicroelectronics)است. این رگولاتور روی برد، از طریق جامپر فعال یا غیرفعال می‌شود.



وقتی این جامپر در مدار قرار داده شود، رگولاتور 5 ولت فعال می‌شود و منبع تغذیه منطقی (Vss) را از منبع تغذیه موتور (Vs) تأمین می‌کند. در این حالت، ترمینال ورودی 5 ولت به عنوان پین خروجی عمل می‌کند و 5 ولتِ 0،5 آمپری را تحویل می‌دهد. می‌توان از این رگولاتور برای تغذیه آردوینو یا مدارهای دیگری که به منبع تغذیه 5 ولت نیاز دارند، استفاده کرد.

هنگامی که جامپر را برداریم، رگولاتور 5 ولت از کار می‌افتد و باید 5 ولت را از طریق ترمینال ورودی 5 ولت جداگانه تأمین کنیم.

اخطار :اگر منبع تغذیه موتور کمتر از 12 ولت باشد، می‌توانید جامپر را در در مدار قرار دهید. اگر ولتاژ بیش از 12 ولت است، باید جامپر را بردارید تا از خراب شدن رگولاتور 5 ولت جلوگیری کنید. همچنین، دقت کنید زمانی که جامپر در جای خود قرار دارد، منبع تغذیه موتور و منبع تغذیه 5 ولت را جداگانه تأمین نکنید.

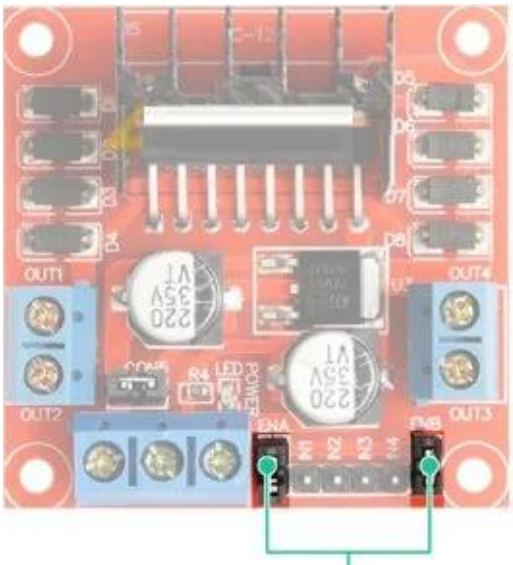
پین‌های کنترل ماژولL298

برای هریک از کانال‌هایL298N ، دو نوع پین کنترل وجود دارد که به ما امکان کنترل همزمان سرعت و چرخش موتورهای DC را می‌دهد. پین‌های کنترلِ جهت و پین‌های کنترلِ سرعت.

پین‌های کنترل جهت در شکل زیر مشخص شده‌اند.

با استفاده از پین‌های کنترل جهت، می‌توانیم موتور را در دو جهت بچرخانیم. این پین‌ها در واقع سوئیچ‌های مدار پل اچ داخل آی‌سی L298N را کنترل می‌کنند. ماژول L298 دارای دو پایه کنترلِ جهت برای هر کانال است. پایه‌های IN1 و IN2 جهت چرخش موتور A را کنترل می‌کنند، در حالی که IN3 و IN4 موتور B را کنترل می‌کنند. جهت چرخش یک موتور را می‌توان با اعمالِ منطقِ HIGH (5 ولت) یا منطقِ LOW (زمین) به این ورودی‌ها کنترل کرد. جدول زیر نحوه انجام این کار را نشان می‌دهد.

ورودی ۱	ورودی ۲	جهت چرخش
Low (0)	Low (0)	موتور خاموش
High (1)	Low (0)	مستقیم
Low (0)	High (1)	عکس
High (1)	High (1)	موتور خاموش

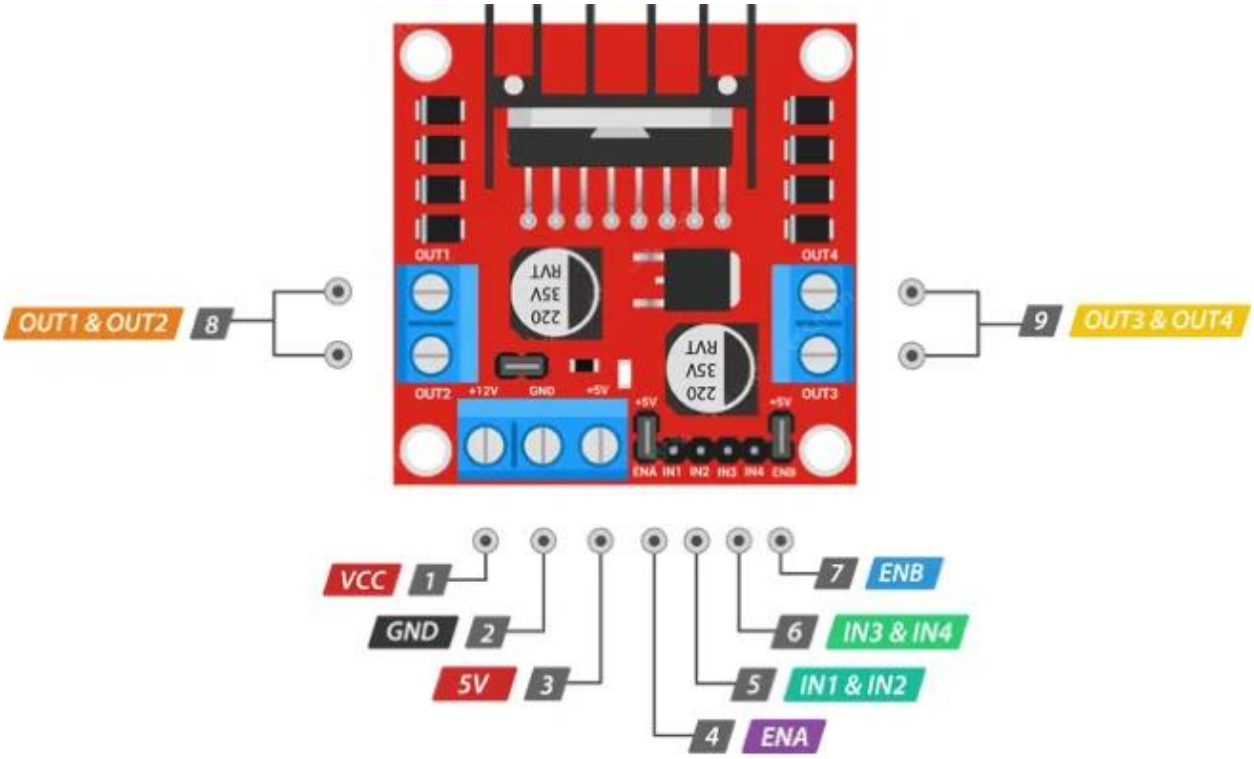


پین‌های کنترل سرعت، یعنی ENA وENB ، برای روشن یا خاموش کردن موتورها و کنترل سرعت آن‌ها استفاده می‌شوند.

با اعمالHIGH ، این پایه‌ها موتور را می‌چرخانند و اعمال LOW باعث متوقف شدن آن‌ها می‌شود، اما با PWM می‌توانیم سرعت موتورها را کنترل کنیم.

معمولاً جامپر در جای خود روی ماژول قرار دارد. در این صورت، موتور فعال است و با حداکثر سرعت می‌چرخد. اگر بخواهیم سرعت موتورها را به صورت برنامه‌ریزی‌شده کنترل کنیم، باید جامپر را برداشته و به PWM میکرو را به آن متصل کنیم.

پین‌های ماژول L298N

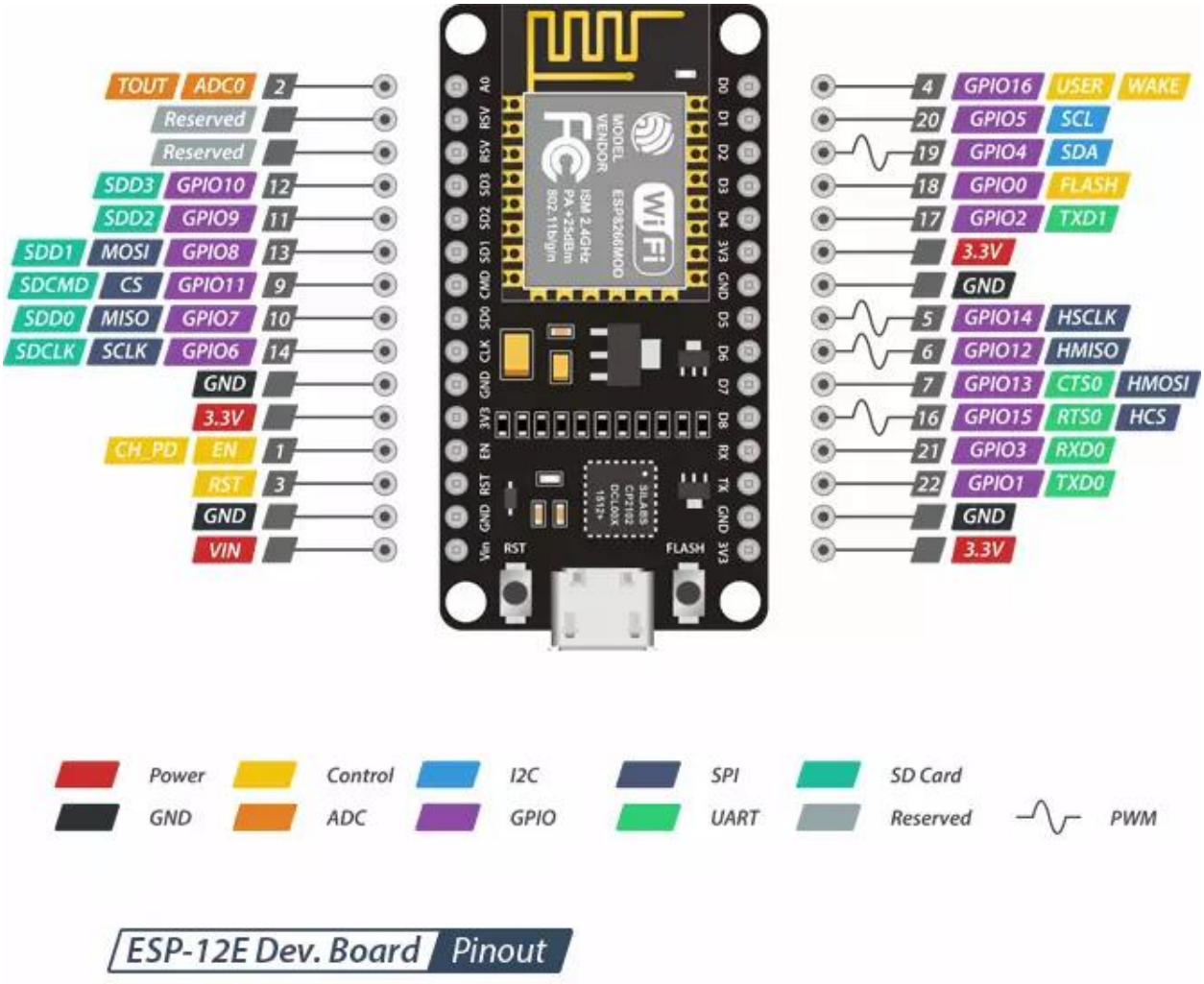


- پین **VCC**توان موتور را تأمین می‌کند. ورودی این پین می‌تواند هر مقداری بین 5 تا 35 ولت باشد. به یاد داشته باشید اگر جامپر 5 ولت EN در جای خود قرار داشته باشد، برای رسیدن به حداکثر سرعت، باید 2 ولت بیشتر از ولتاژ واقعی موتور را تأمین کنید.
- **GND**پایه زمین مشترک است.
- پین **5V**منبع تغذیه مدارهای منطقی سوئیچینگ داخل آی‌سی L298N است. اگر جامپر 5 ولت EN در جای خود باشد، این پین به عنوان یک خروجی عمل می‌کند و می‌توان از آن برای تغذیه آردوینو استفاده کرد. اگر جامپر ۵ ولت EN برداشته شود، باید آن را به پایه 5 ولت آردوینو وصل کنید.
- از پین‌های **ENA**برای کنترل سرعت موتور A استفاده می‌شود. اعمال HIGH به این پایه HIGH) نگه داشتن جامپر (باعث چرخش موتور A می‌شود و LOW شدن آن باعث متوقف شدن موتور می‌شود. برداشتن جامپر و اتصال ورودی PWM به این پایه، باعث می‌شود بتوانیم سرعت موتور A را کنترل کنیم.

- از پین‌های **IN1**و **IN2**برای کنترل جهت چرخش موتور A استفاده می‌شود. هنگامی که یکی از آن‌ها HIGHو دیگری LOW باشد، موتور A می‌چرخد. اگر هر دو ورودی HIGH یا LOW باشند، موتور A متوقف می‌شود.
- پین‌های **IN3**و **IN4**برای کنترل جهت چرخش موتور B به کار می‌روند. وقتی یکی از این پین‌ها HIGH باشد و دیگریLOW ، موتور B می‌چرخد. اگر هر دو ورودی HIGH یا LOW باشند، موتور B متوقف می‌شود.
- از پین‌های **ENB**برای کنترل سرعت موتور B استفاده می‌شود. اعمال HIGH به این پایه HIGH) نگه داشتن جامپر (باعث چرخش موتور B می‌شود، LOWکردن آن باعث متوقف شدن موتور می‌شود. با برداشتن جامپر و اتصال این پایه به ورودی PWM می‌توانیم سرعت موتور B را کنترل کنیم.
- پین‌های **OUT1**و **OUT2**به موتور A متصل می‌شوند.
- پین‌های **OUT3**و **OUT4**به موتور B متصل می‌شوند.

میکروکنترلر:

ما برای این پروژه از میکروکنترلر esp8266 استفاده کردیم. ESP8266یک SOC (سیستم روی تراشه) وای فای است که توسط Espressif Systems تولید شده است. این یک تراشه بسیار یکپارچه است که برای ارائه اتصال کامل به اینترنت در یک بسته کوچک طراحی شده است. این ماژول دارای 11 پین GPIO (پین های ورودی/خروجی عمومی) و یک ورودی آنالوگ نیز می باشد. این بدان معناست که شما می توانید آن را مانند هر آردوینو معمولی یا میکروکنترلر دیگری برنامه ریزی کنید. و علاوه بر آن، شما ارتباط Wi-Fi را دریافت می کنید، بنابراین می توانید از آن برای اتصال به شبکه Wi-Fi خود، اتصال به اینترنت، میزبانی وب سرور با صفحات وب واقعی و ... استفاده کنید.



پایه‌های تغذیه: در کل چهار پایه تغذیه شامل یک پایه VIN و سه پایه V3.3 وجود دارد. اگر یک منبع ولتاژ 5 ولت رگوله شده در اختیار داشته باشید، می‌توانید از پایه VIN برای اتصال مستقیم تغذیه ESP8266 و قطعات جانبی استفاده کنید. پایه‌های V3.3 ، خروجیِ رگولاتورِ رویِ برد هستند که می‌توانند تغذیه قطعات خارجی را فراهم کنند.

GND: پایه زمینِ بردِ NodeMCU ESP8266 است.

پایه‌های I2C: این پایه‌ها، برای اتصال انواع سنسورها و قطعات I2C در پروژه شما به کار می‌رود و هر دو حالت Master و Slave پروتکل I2C پشتیبانی می‌شوند. کارکرد رابط I2C از طریق برنامه قابل تنظیم است و فرکانس کلاک حداکثر 100 کیلوهرتز می‌باشد. در اینجا باید به این نکته دقت داشته باشید که فرکانس کلاک I2C باید از حداقل فرکانس دستگاه slave بیشتر باشد.

پایه‌های ورودی خروجی (GPIO): برد NodeMCU ESP8266، هفده پایه GPIO دارد که می‌توانند به کارهای مختلف مانند I2C، I2S، UART، PWM، کنترل از راه دور IR، LEDها و کلیدها اختصاص داده شوند. هر یک از پایه‌های ورودی خروجی دیجیتال می‌توانند به‌صورت pull up، pull down داخلی یا به‌صورت امپدانس بالا تنظیم شوند. زمانی‌که این پایه‌ها به عنوان ورودی تعریف شوند، می‌توانند به منظور ایجاد وقفه‌های پردازنده، برای تحریک در لبه یا سطح تنظیم شوند.

کانال ADC: در NodeMCU یک ADC (با دقت 10 بیتی و روش تقریبات متوالی) قرار داده شده است. از ADC می‌توان برای تست ولتاژ تغذیه از پایه VDD3P3 و تست ولتاژ ورودی از پایه TOUT استفاده نمود. با این حال، این دو عمل را نمی‌توان هم‌زمان اجرا کرد.

پایه‌های UART: برد NodeMCU ESP8266 دارای دو رابط UART به نام‌های UART0 و UART1 است که ارتباط آسنکرون (RS232 و RS485) را فراهم می‌کنند و می‌توانند تا سرعت 4.5 مگابیت در ثانیه (Mbps) کار کنند. UART0 (پایه‌های CTS0& TXD0, RXD0, RST0) از flow control پشتیبانی می‌کند و می‌توان از آن برای ارتباط استفاده کرد، در حالی‌که UART1 (پایه TXD1) تنها به ارسال داده می‌پردازد. به همین دلیل، معمولاً برای چاپ اطلاعات و گزارشات مورد استفاده قرار می‌گیرد.

پایه‌های SPI: ماژول ESP8266 دو رابط SPI و HSPI را در حالت‌های slave و master در اختیار کاربر قرار می‌دهد. به علاوه، این دو رابط از قابلیت‌های عمومی SPI نیز که در بخش زیر خواهید دید، پشتیبانی می‌کنند:

- 4 حالت زمان‌بندی انتقال با فرمت SPI
- فرکانس کلاک 80Mhz و تقسیم‌های فرکانسی آن
- تا 64 بایت حافظه First In First Out – FIFO، یکی از روش‌های سازمان‌دهی کنترل داده با توجه به اولویت‌بندی است. از این روش برای بافرکردن و کنترل سرعت انتقال اطلاعات استفاده می‌شود.

پایه‌های SDIO: ماژول ESP8266، رابط ورودی خروجی دیجیتال امن (SDIO) را ارائه می‌کند که برای ارتباط مستقیم با کارت‌های SD به کار می‌رود. SDIO بیست و پنج مگاهرتز، 4 بیتی نسخه 1.1 و SDIO پنجاه مگاهرتز 4 بیتی نسخه 2.0 نیز، توسط ESP8266 پشتیبانی می‌شوند.

پایه‌های PWM: برد ESP8266، چهار کانال مدولاسیون عرض پالس (PWM) دارد. خروجی PWM می‌تواند به‌صورت نرم‌افزاری پیاده‌سازی شود و برای درایو موتورهای دیجیتال و LEDها مورد استفاده قرار گیرد. رنج دوره تناوب PWM از 1000 میکروثانیه تا 10000 میکروثانیه قابل تنظیم است، یعنی از فرکانس 100 هرتز تا 1 کیلوهرتز.

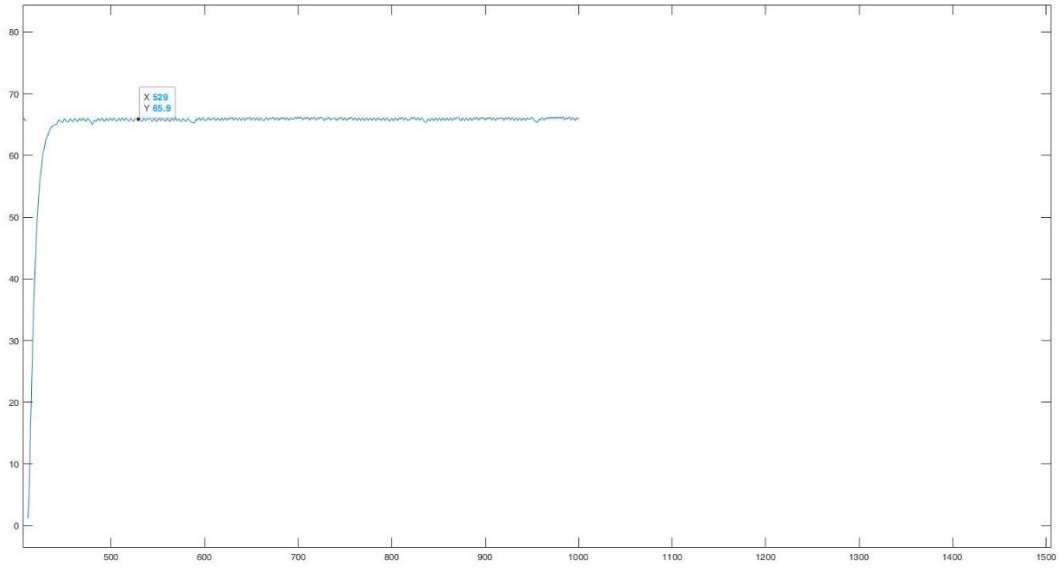
پایه‌های کنترل: از این پایه‌ها برای کنترل ESP8266 استفاده می‌شود. این پایه‌ها شامل پایه فعالسازی چیپ (EN)، ریست (RST) و پایه بیدارکردن (WAKE) هستند.

- پایه EN – زمانی‌که پایه EN یک شود، چیپ ESP8266 فعال می‌شود. زمانی‌که پایه EN صفر شود، چیپ با حداقل توان کار می‌کند.
- پایه RST – این پایه برای ریست کردن چیپ ESP8266 به کار می‌رود.
- پایه WAKE – این پایه برای بیدار کردن چیپ از خواب عمیق (deep-sleep) به کار می‌رود.

تست حلقه باز و بدست آوردن پارامتر های سیستم:

برای محاسبه پارامتر های سیستم و تاخیر، از تست حلقه باز موتور استفاده می کنیم. سایر روش ها به دلیل آسیب رساندن به موتور استفاده نکردیم (تست نوسان کامل) . ما برای خواندن دیتای انکودر برای کاهش نویز، از فیلتر RC استفاده کردیم. نویز باعث خطا بر روی دیتای خوانده شده از انکودر می شد.

با انجام تست حلقه باز پله، نمودار سرعت خروجی را plot کردیم. برای بدست آوردن پارامتر های سیستم از روش two pints method استفاده می کنیم.



K = 65

T1 = 0.008

T2 = 0.001

τ = 1.5(T1 – T2) = 0.0105

T0 = T1 – τ = 0.0025

$$G(s) = \frac{65\, e^{-0.0025s}}{1 + 0.0105s}$$

با انجام این محاسبات، معادله حالت سیستم بدست آمد.

بدست آوردن ضرایب PID :

برای بدست آوردن ضرایب PID از روش های مختلفی می توان استفاده کرد. از این روش های می توان به QDR (Quarter decade ratio) و ziegler-nicholes استفاده کرد.

Controller type	K_c	T_I	T_D
P	τ/Kt_0	-	-
PI	$0.9\tau/Kt_0$	$o.33t_0$	-
PID (series)	$1.2\tau/Kt_0$	$2t_0$	$o.5t_0$
PID (parallel)	$1.2\tau/Kt_0$	$2.5t_0$	$o.4t_0$

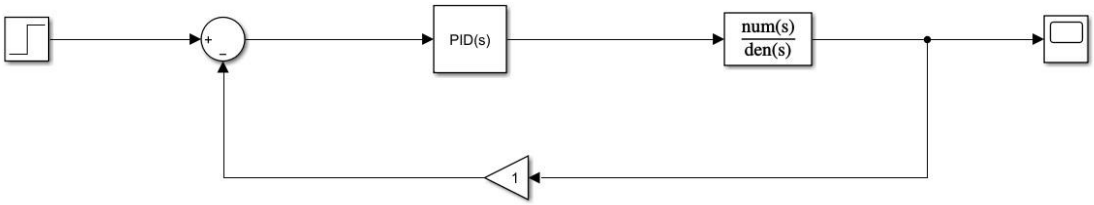
PID parallel :

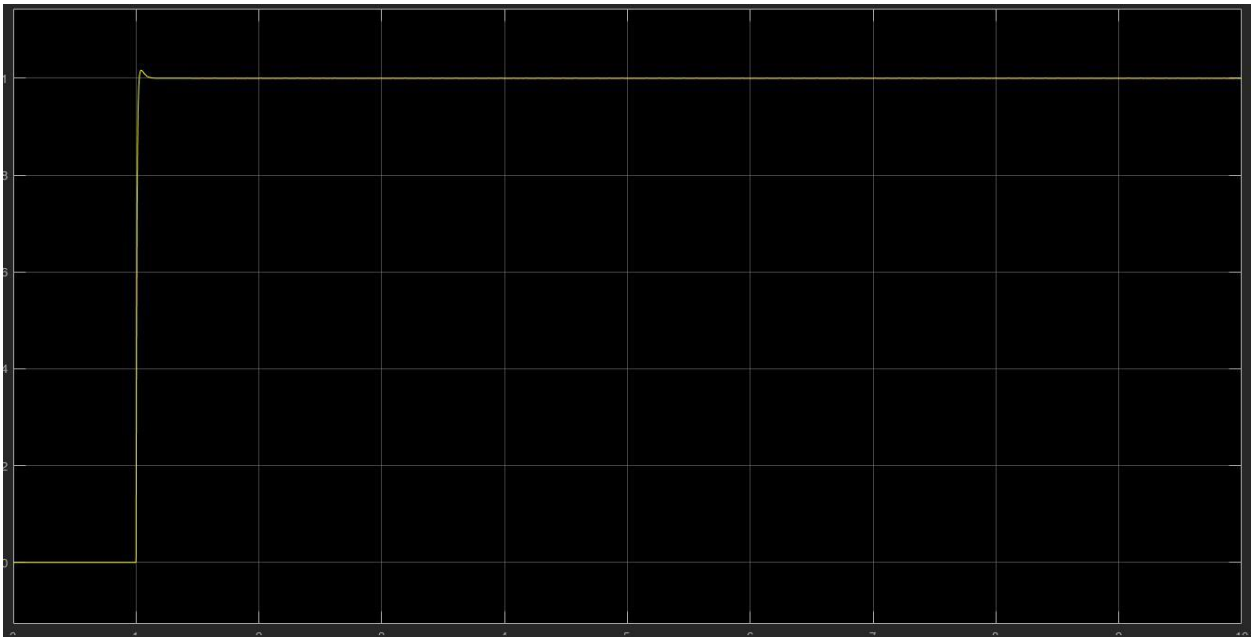
Kc = 4.89 Ki =Kc/Ti= 188.07 Kd =Kc*Td = 0.00489

شبیه سازی MATLAB :

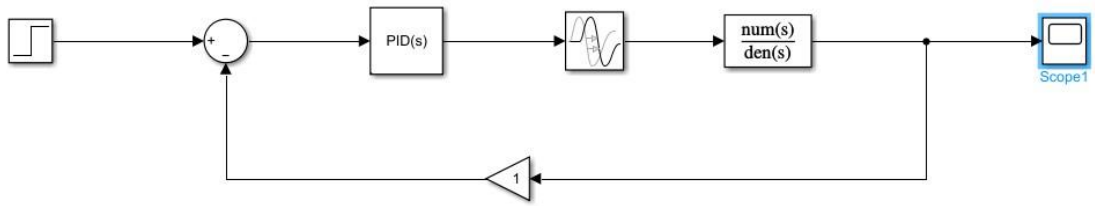
ابتدا سیستم بدون تاخیر را شبیه سازی کردیم:

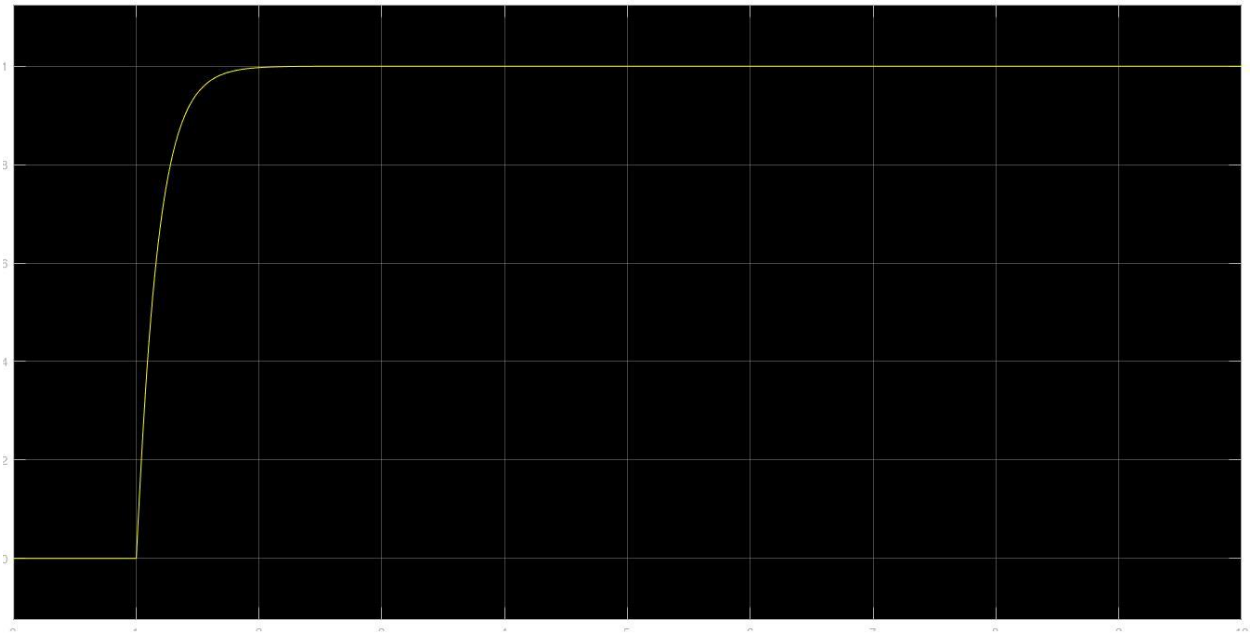
ضرایب PID که برای سیستم محاسبه شده اعمال شده است.



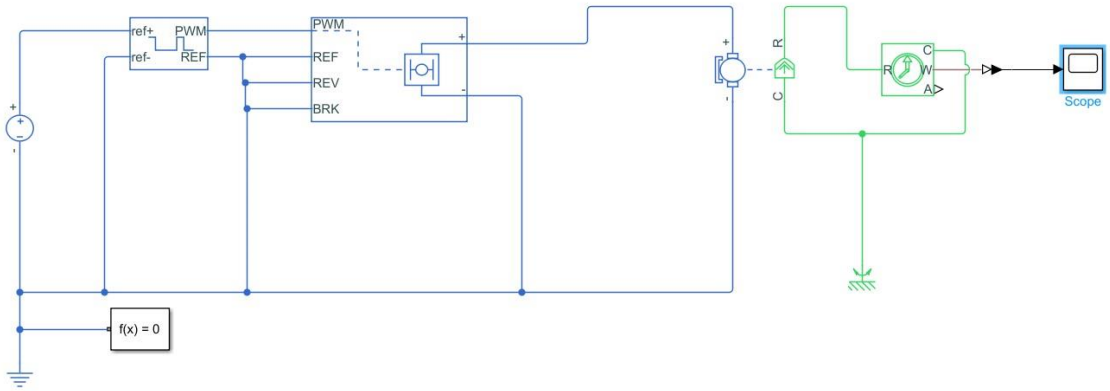


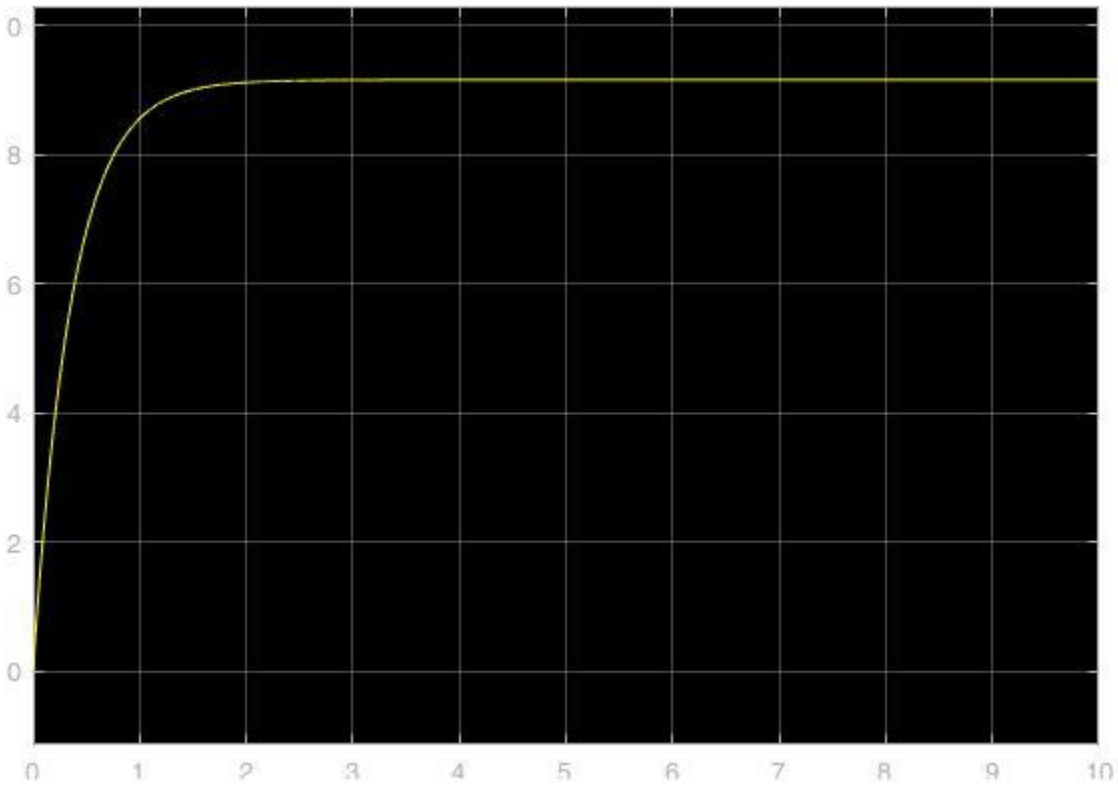
حال سیستم با تاخیر را شبیه سازی میکنیم:





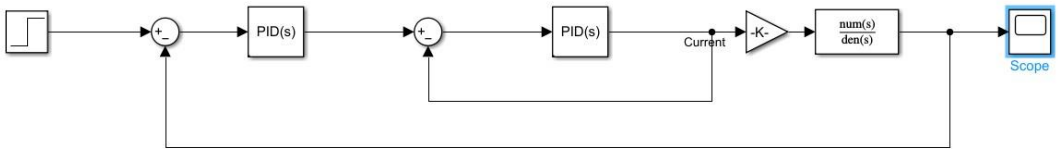
شبیه سازی مدل پیشنهادی با پلH :

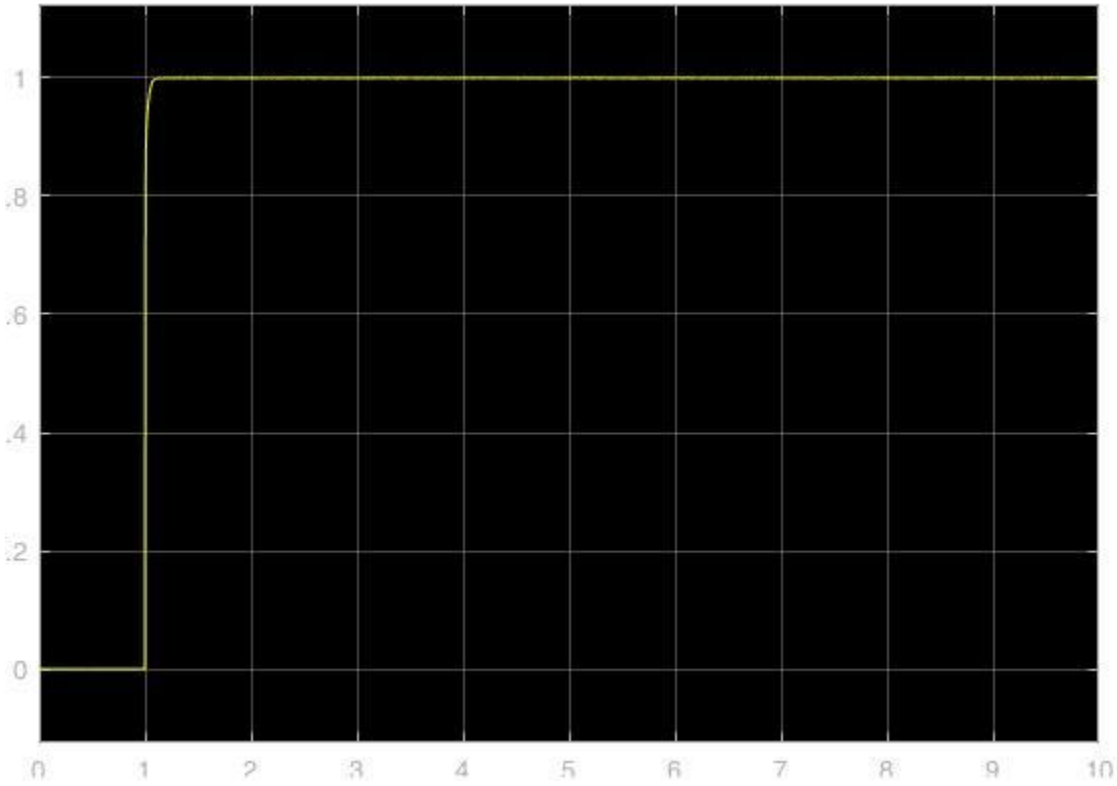




شبیه سازی کنترلر cascade :

در کنترل cascade از دو حلقه فیدبک و دو کنترلر PID استفاده می کنیم:





پیاده سازی کد میکروکنترلر:

در ابتدا لازم است ذکر شود که تمامی کد ها بصورت خوانا و با کامنت گذاری کامل می باشد.

در قسمت اول کد ، پایه های میکرو را انتخاب و به برنامه معرفی کردیم. در ادامه به ایجاد متغیر های مختلف که برای نوشتن کد نیاز داشتیم پرداختیم:

```
//Start Define PINS
//define Encoder pins for Encoder A and Encoder B
#define encoderPinA D1
#define encoderPinB D2

//Start define Driver(L298N) pins for Enable and IN1 and IN2
#define ENA  D5
#define IN1  D6
#define IN2  D7
//End Define PINS
//IS Ram dosent recegnise Interruption and we shoud use this line
void ICACHE_RAM_ATTR readEncoder();
//Start Define Global variables
// used in an interrupt for read encoder
// Use the "volatile" directive for variables
volatile int counter = 0;
int lastCounter = 0;
//Time used in velocity compute
unsigned long lastTime = 0;
//pulse per one round
float PPR = 150;
//motor shaft rotation & Gear ratio
float MSR = 6.7;
//Start define low pass Filter value
float Filter_RPM = 0;
float lastFilter_RPM = 0;
float lastRPM = 0;
float laste = 0;
```

در ادامه وارد قسمت void setup می شویم که تنظیمات و مود های اولیه پایه ها را معرفی میکنیم. در این قسمت یک interrupt برای خواندن خروجی انکودر نیز قرار داده ایم که وارد تابع readEncoderمی شود:

```
// for motor direction
int DirMotor;
//set point
float SP = 0;
float eintegral = 0;
float der = 0;
void setup() {

  Serial.begin(115200);

  pinMode(encoderPinA, INPUT);
  pinMode(encoderPinB, INPUT);

  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  attachInterrupt(digitalPinToInterrupt(5),readEncoder, RISING);

}
```

در ادامه وارد قسمت اصلی برنامه می شویم در این قسمت سرعت موتور را با استفاده از دیتای خروجی انکودر محاسبه کرده و با اعمال فیلتر بر آن، نویز را کاهش می دهیم:

```
void loop() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA,255);
  // Compute velocity with readEncoder function
  long currentTime = micros();
  float deltaTime = ((float) (currentTime-lastTime))/1.0e6;
  float velocity = (counter - lastCounter)/deltaTime;
  lastCounter = counter;
  lastTime = currentTime;

  // Convert velocity(count/s) to RPM
  //RPM = (count/s) * (round/pulses per one round) * (one output shaft rotation/motor shaft rotation) * (60s/1m)
  float RPM =fabs(velocity * (1/PPR) * (1/MSR) * (60));

   //set our low pass Filter in our plant
 Filter_RPM = 0.7284*lastFilter_RPM + 0.1357*RPM + 0.1357*lastRPM;
 lastFilter_RPM = Filter_RPM;
 lastRPM = RPM;

  Serial.print(Filter_RPM);
  Serial.println();
 SP = 60 ;
```

در این قسمت برای امنیت plant اجازه نمی دهیم موتور با سرعت بیش از حد مجاز خود بچرخد تا در مرحله اول امنیت سیستم حفظ شود و در مرحله بعد عمر مفید موتور کاهش نیابد :

```
 //setting max speed for security of motor
 if(SP>55)
 {
  SP = 55;
 }
 if(SP<-55)
 {
  SP = -55;
 }
 //setting motor direction
if(SP > 0){
      DirMotor = 1;
    }
    else if(SP < 0){
      DirMotor = -1;
      SP = fabs(SP);
    }
    else{
      DirMotor = 0;
    }
```

در این قسمت کد، وارد کنتل کننده PID می شویم. ابتدا ضرایبی که با سعی و خطا تعیین شده را وارد کرده و بعد ارور ،قسمت های مشتق گیر، انتگرال گیر را محاسبه می کنیم و قسمت اصلی برنامه به پایان می رسد:

```
// PID section
    float kp = 6;
    float ki = 12;
    float kd = 0.8;
    float e = SP - Filter_RPM;


    der = (e - laste)/deltaTime;


    eintegral = eintegral + e*deltaTime;

    float u = kp*e + ki*eintegral + kd*(der);

    int pwr = (int) fabs(u);

    laste = e;


    // using setmotor function for setting motor parameter and moving
    setMotor(DirMotor,pwr,ENA,IN1,IN2);

  delay(20);
}
```

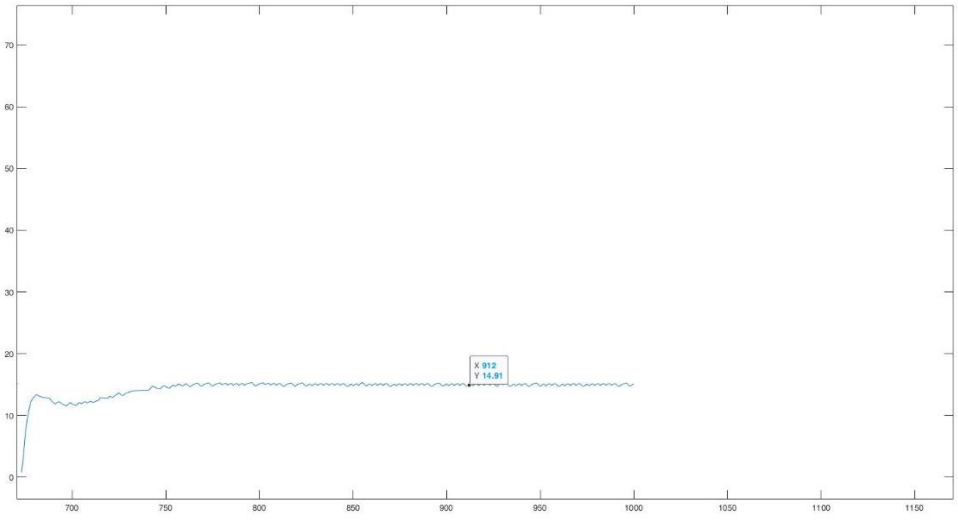
بعد از حلقه اصلی برنامه، دوتابع برای ست کردن پارامتر های موتور و یک تابع برای خواندن انکودر خواهیم داشت:

```
void setMotor(int dir, int pwmVal, int pwm, int in1, int in2)
{
    analogWrite(pwm,pwmVal); // Motor speed

    if(dir == 1){
    // Turn one way
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
  }
  else if(dir == -1){
    // Turn the other way
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
}
else{
    // Or dont turn
    digitalWrite(in1,LOW);
    digitalWrite(in2,LOW);
  }
}
```

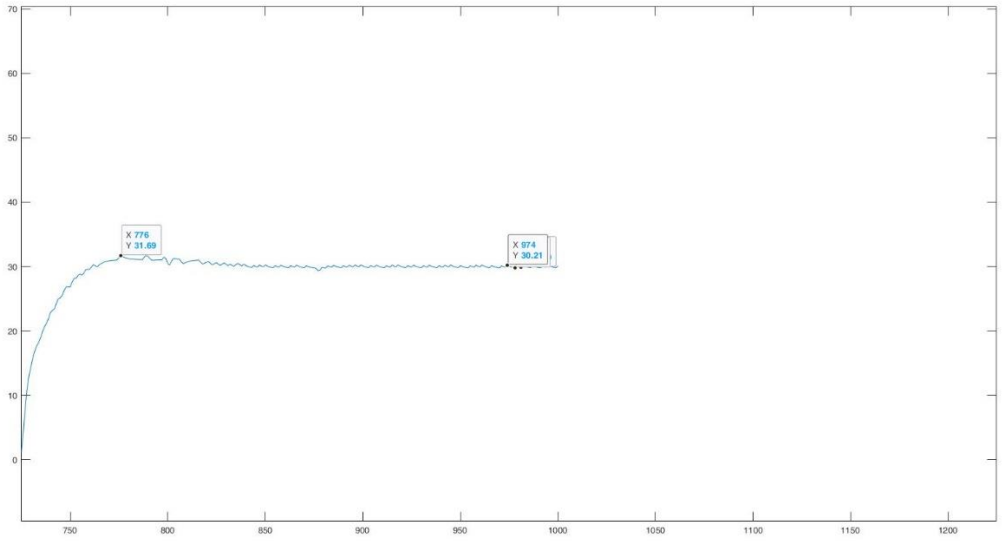
```
void readEncoder(){

  //metod2
  // Read encoder B when ENCA rises
  int b = digitalRead(encoderPinB);
  int increment = 0;
  if(b>0){
    // If B is high, increment forward
    increment = 1;
  }
  else{
    // Otherwise, increment backward
    increment = -1;
  }
  counter = counter + increment;

  ////test pulse with serial print
  // Serial.print(counter);
  // Serial.println();

}
```

در این قسمت خروجی را با Set point 3 مختلف نمایش می دهیم:

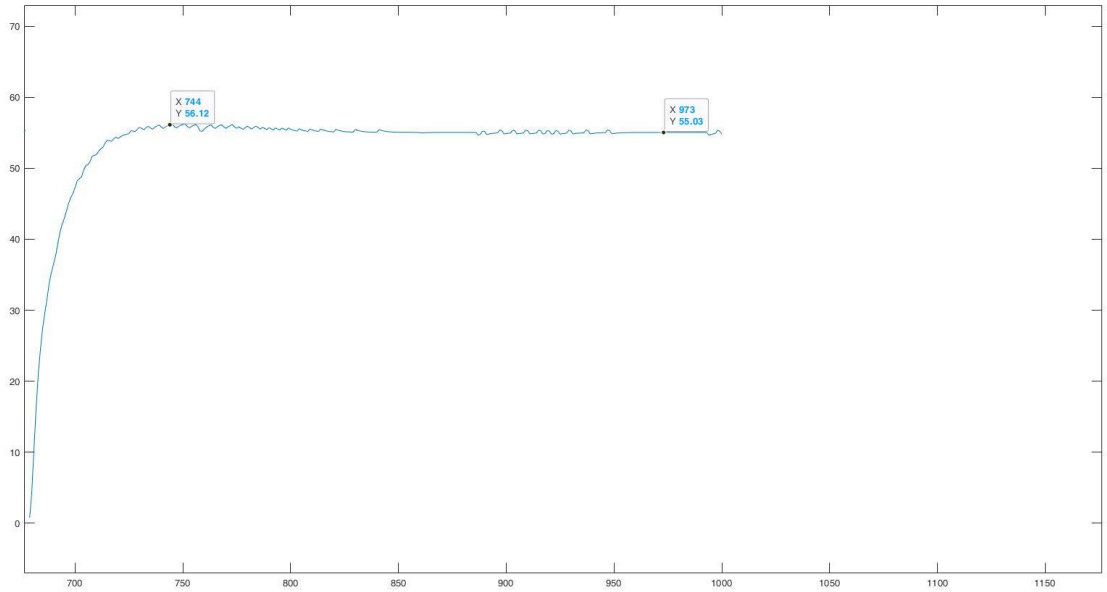
Set point = 15 rpm



Set point = 30 rpm



Set point = 55 rpm



کنترل موقعیت:

در قسمت اول کد ابتدا پین های میکروکنترلر را انتخاب و معرفی کردیم و متغیر های مورد نیاز برنامه را مشخص کردیم:

```
//Start Define PINS
//define Encoder pins for Encoder A and Encoder B
#define encoderPinA D1
#define encoderPinB D2

//Start define Driver(L298N) pins for Enable and IN1 and IN2
#define ENA  D5
#define IN1  D6
#define IN2  D7
//End Define PINS

void ICACHE_RAM_ATTR readEncoder();
volatile int counter = 0;
float PPR = 150;
float MSR = 6.7;
float lastTime = 0;
float eintegral = 0;
float der = 0;
float laste;
```

در ادامه وارد void setup شده و تنظیمات اولیه را انجام دادیم و یک interrupt برای خواندن دیتای انکودر نیز قرار دادیم:

```
void setup() {
  // put your setup code here, to run once:
Serial.begin(115200);

  pinMode(encoderPinA, INPUT);
  pinMode(encoderPinB, INPUT);

  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  attachInterrupt(digitalPinToInterrupt(5),readEncoder, RISING);
}
```

در ادامه وارد حلقه اصلی برنامه می شویم. در اینجا موقعیت موتور را بدست آورده و set point را برحسب درجه وارد میکنیم سپس وارد قسمت کنترلر PID شده و ضرایب را با سعی و خطا بدست آوردیم:

```
void loop() {
  // put your main code here, to run repeatedly:
  long currentTime = micros();
  float deltaTime = ((float) (currentTime-lastTime))/1.0e6;
  lastTime = currentTime;
  int motorposition = counter;
  float SP = 180;
  SP = SP * 3;

  float kp = 2.6;
  float ki = 2.55;
  float kd = 0.3;

  float e = SP - motorposition;

  der = (e - laste)/deltaTime;
  eintegral = eintegral + e*deltaTime;
  float u = kp*e + ki*eintegral + kd*(der);

  float laste = e;
```

این قسمت برای قرار دادن یک ماکسیمم مقدار برای خروجی کنترلر می باشد که تحت یک سرعت کنترل شده به سمت موقعیت حرکت کند :

```
// setting a maximum for u
  if(u>80)
  {
    u = 80;
  }

  if(u<-80)
  {
    u = -80;
  }

  int pwr = (int) u;
```

در ادامه شروطی برای حرکت موتور قرار دادیم تا تحت حرکات مختلف به set point نزدیک شود و بتواند سرعت مناسب و جهت مناسب را به موتور بدهد و در اینجا برنامه اصلی به اتمام می رسد

```
if(pwr>0)
{

  digitalWrite(IN2,HIGH);
  digitalWrite(IN1,LOW);
  analogWrite(ENA,pwr);

}
else if(pwr<0)
{
  digitalWrite(IN2,LOW);
  digitalWrite(IN1,HIGH);
  analogWrite(ENA,abs(pwr));
}
else if(pwr == 0)
{
  // Or dont turn

  digitalWrite(IN1,LOW);
  digitalWrite(IN2,LOW);
  analogWrite(ENA,abs(pwr));

}

// Serial.print("pose :");
Serial.print(motorposition/3);

Serial.println();
delay(20);
}
```

در قسمت اخر این کد یک تابع برای خواندن خروجی interrupt انکودر تعریف کردیم:

```
void readEncoder(){

  //metod2
  // Read encoder B when ENCA rises
  int b = digitalRead(encoderPinB);
  int increment = 0;
  if(b>0){
    // If B is high, increment forward
    increment = 1;
  }
  else{
    // Otherwise, increment backward
    increment = -1;
  }
  counter = counter + increment;

  ////test pulse with serial print
  // Serial.print(counter);
  // Serial.println();

}
```