ghira

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	

ZUMO ROBOT

Autor: Ghiran Lorena

Grupa**: 30131**

AN UNIVERSITAR: 2019-2020

PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	

Cuprins

PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	

1. Introducere

Robotul ZUMO este un robot care are ca principal controler un dispozitiv Arduino. Măsoară aproximativ 10cm. Folosește două motoare și are o viteză maximă de 60cm/s. Robotul ZUMO include o lamă din oțel inoxidabil tăiat cu laser, montat pe partea din față pentru a împinge obiecte și un tablou de senzori montat de-a lungul marginii frontale a ZUMO(în spatele lamei). Acesta permite ZUMO-ului să detecteze caracteristici de pe sol în fața sa, cum ar fi linii pentru urmărire sau margini pentru evitare.

Robotul zumo are un compartiment intern pentru patru baterii AA.

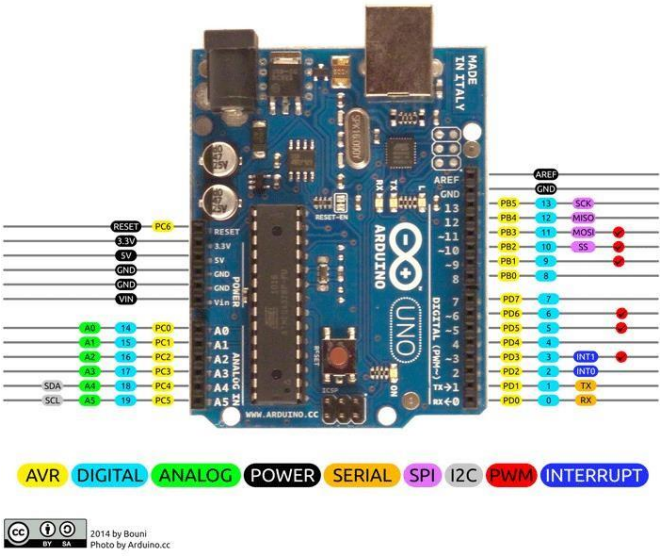
De asemenea, exista cinci leduri: un set de leduri de putere, unul albastru si unul rosu, care sunt amplasate in fiecare din cele doua colturi din spate ale scutului. Un led galben de utilizator, situate pe marginea din dreapta a scutului. Acesta este conectat la pinul digital 13 pe Arduino.

Butoane, unul de resetare situat pe marginea din stanga a scutului, si este conectat la pinul reset al placutei si poate fi apasat pentru a reseta placa si buton de utilizator. Butonul de utilizator este situate pe marginea din spate a scutului si este conectat la pinul digital 12.

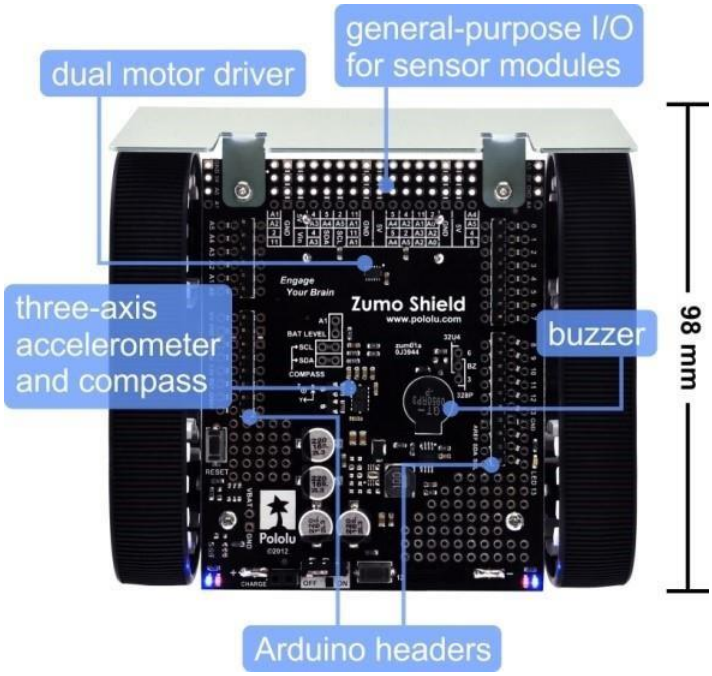
Pentru motor, sunt folositi patru pini: pinul 7 pentru controlul directiei drepte a motorului (low conduce inainte, high in sens invers); pinul 8 controleaza directia motorului stang; pinul 9 controleaza viteza motorului din dreapta cu PWM; pinul 10 controleaza viteza motorului stang cu PWM.

Robotul contine si un buzzer folosit pentru a genera sunete si muzica simpla.

O serie de conexiuni I/O permit montarea unor senzori suplimentari sau a altor componente.



PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	



PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	

2. Scopul Proiectului

a. Obiective

Principalul obiectiv al aplicației este de a programa robotul ZUMO pentru a controla poziția și viteza folosind placa Arduino.

Cel de al doilea obiectiv este Line Follower, prin care robotul este programat sa urmeze o linie neagra pe o suprafata alba.

Ideea pricipala este ca centrul robotului trebuie sa fie intotdeuna pe o linie neagra.Daca robotul se indeparteaza de linie din diferite motive,ar trebui sa facem robotul sa se intoarca la linie.

1.Identificam pozitia instantanee a robotului fata de linia neagra.Distanta cu care robotul s-a abatut de la linia neagra va fi referinta, semnal de eroare.Acesta este calculat folosind informatii de la senzorii de reflectare.

2.Pe baza erorii calculate mai sus, calculam cantitatea de rotatie necesara pentru ca robotul sa revina la linie.Semnalul trimis se numeste semnal de control,care va asigura faptul ca robotul nu se va mai abate de la linie.Acest semnal de control este calculat folosid un regulator de tip PID Iesirea de la PID va intoarce robotul inapoi la linie.

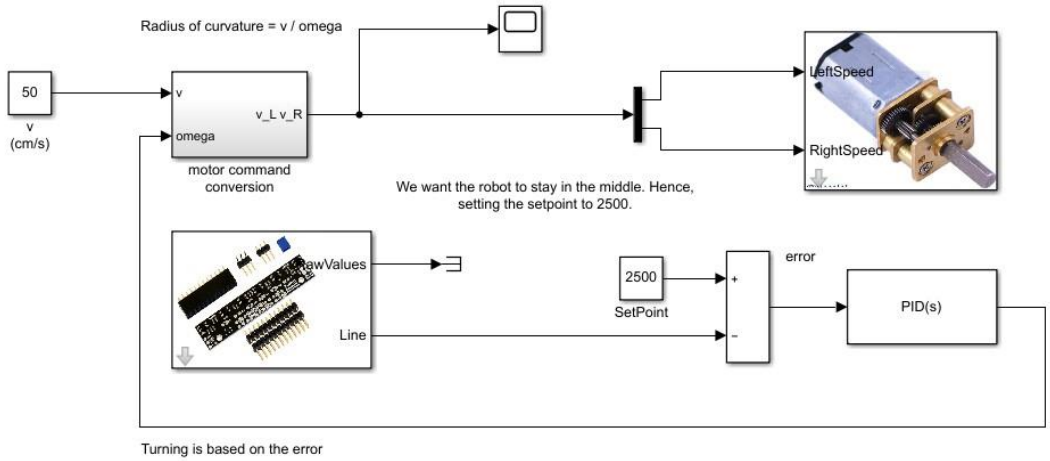
PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	

b. Specificații

In cazul cerintei LineFollower, folosim senzori cu infrarosu sau de reflectare. Acesti senzori detecteaza cantitatea de lumina reflectata pe o suprafata.O suprafata neagra reflecta mai putina lumina comparativ cu una alba.Putem folosi senzorul de reflectare pentru a detecta prezenta unei linii negre pe o suprafata alba.

Robotul Zumo asamblat are 6 senzori de reflectare pe suprafata inferioara. Daca folosim iesirea de la acesti senzori putem detecta pozitia relativa a robotului fata de linia neagra. Iesirele senzorilor se vor combina si vor da o comanda.

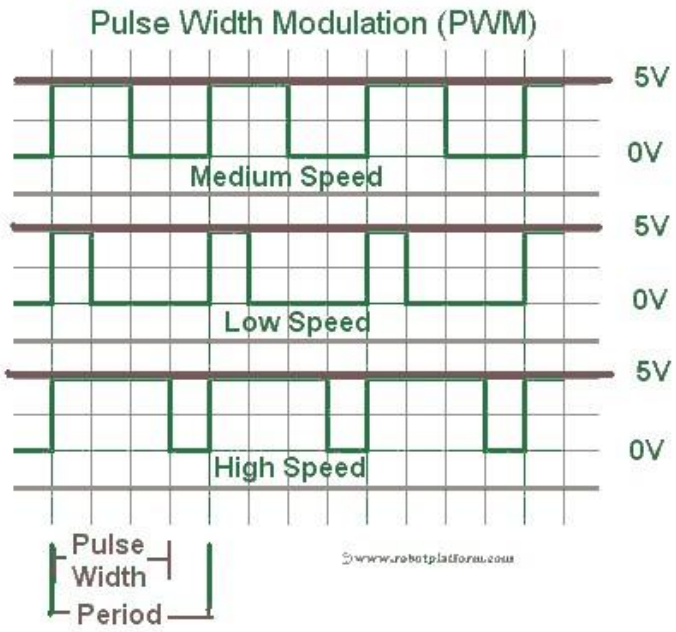
Biblioteca ZumoreflenceSensor face acest lucru,si va avea ca iesire un singur numar cuprins intre 0 si 500. Datorita faptului ca valoarea numerica 2500 repezinta centrul robotului,ceea ce inseamna ca robotul este situate pe linia neagra,vom folosi aceasta informatie ca referinta. Pentru 2500,robotul corespunde deplasarii inainte,fara rotire.Daca valoarea data de senzori este mai mica de 2500, inseamna ca robotul este pe o parte a liniei negre,iar daca valoarea este mai mare de 2500,inseamna ca robotul este pe cealalta parte a liniei.Am scazut la iesirea de la Sensori 2500, astel vom avea valori pozitive pentru o parte iar pentru celalata parte valori negative.Diferenta intre valoarea de referinta si valoarea reala se numeste semnal de eroare.



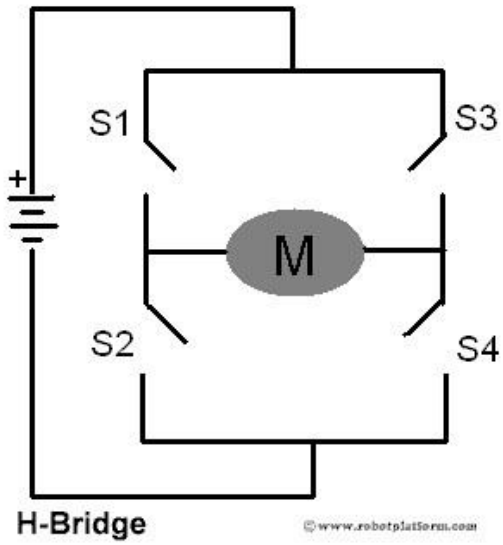
Acest semnal de eroare se va folosi ca intrare pentru PID, si se va decide cum sa se mute robotul pentru a se intoarce la linia neagra. Functia din PID transmite comanda la motor care va asigura faptul ca robotul urmareste linia.

PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	

Pentru a controla viteza unui motor cu curent continuu, cuplul sau este modulat folosind SCR(redresor controlat prin silicon) sau PWM(modul de latime a impulsurilor). SCR si alte metode sunt rareori utilizate in roboti, iar tehnica larg acceptata pentru controlul vitezei ete PWM.



Controlul unui motor presupun controlul vitezei si al rotatiei(fie in sensul acelor de ceasornic sau invers). Pentru motoarele cu curent continuu, controlul directiei se realizeaza printr-o punte H care conduce motorul in orice directive. O punte H tipica consta dintr-un minim de patru inrerupatoare mecanice sau in stare solida.



PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	

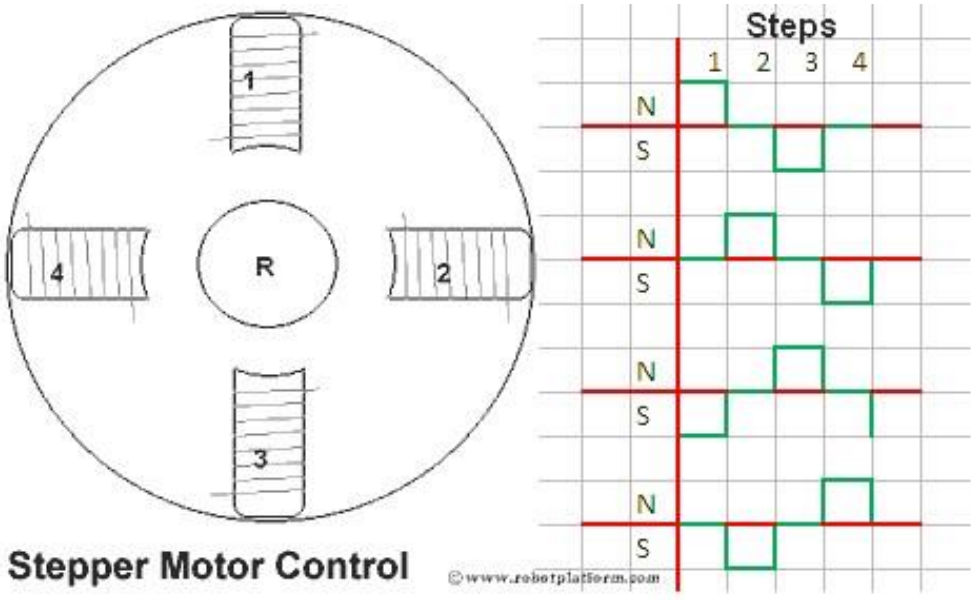
Acest circuit poate fi utilizat pentru a conduce motorul inainte, inapoi, frana sau rulare libera prin comutarea sau oprirea S1, S2, S3 si S4 conform tabelului.

S1	S2	S3	S4	Motor
1	0	0	1	Clockwise (Forwards)
0	1	1	0	Anti-clockwise (Reverse)
0	0	0	0	Free run
0	1	0	1	Brake
1	0	1	0	Brake

Pentru roboti mici, se poate construe o punte H cu o combinatie de tranzistoare PNP si NPN.

Motorul Pas cu pas contine mai multi electromagneti care rotesc arborele motorului atunci cand este alimentat. Prin transmiterea impulsurilor dorite catre acesti electromagneti, se poate controla directia si viteza unui motor pas cu pas.

In imaginea de mai jos, exista patru electromagneti care controleaza rotatia rotorului. Activarea fiecaruia dintre ele intr-un mod dorit muta axul rotorului intr-o anumita pozitie. Viteza cu care se schimba impulsurile determina viteza motorului si combinatia de alimentarare a electromagnetului decide directia rotorului.



PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	

3. Determinarea modelului matematic al sistemului

a. Analiza sistemului

Pentru a putea vedea impulsurile obtinute in urma miscarii senilelor trebuie rulat acest cod:

```
#include <Encoder.h>
Encoder knobLeft(2, 11);
Encoder knobRight(3, 4); void
setup() {   Serial.begin(9600);
  Serial.println(" Encoder Test:");
}
long positionLeft  = 999;
long positionRight = 999;
void loop() {   long
newLeft, newRight;
  newLeft = knobLeft.read(); // se citesc impulsurile
newRight = knobRight.read();
  if (newLeft != positionLeft || newRight != positionRight) { //daca pozitia se tot schimba
Serial.print("Left = ");
    Serial.print(newLeft);
    Serial.print(", Right = ");
    Serial.print(newRight); //se afiseaza datele
    Serial.println();
    positionLeft = newLeft; //se dau noile pozitii ale motoarelor
positionRight = newRight;
  }
  // if a character is sent from the serial monitor,
// reset both back to zero.
  if (Serial.available()) {
    Serial.read();
    Serial.println("Reset both knobs to zero");
knobLeft.write(0);
    knobRight.write(0);
  } }
#define PWM_R 9
//#define DIR_L 8
//#define DIR_R 7 #include
<Encoder.h>
int speed1=50;//se seteaza factorii de umplere int
speed2=200;//pentru a avea 2 trepte la iesire
```

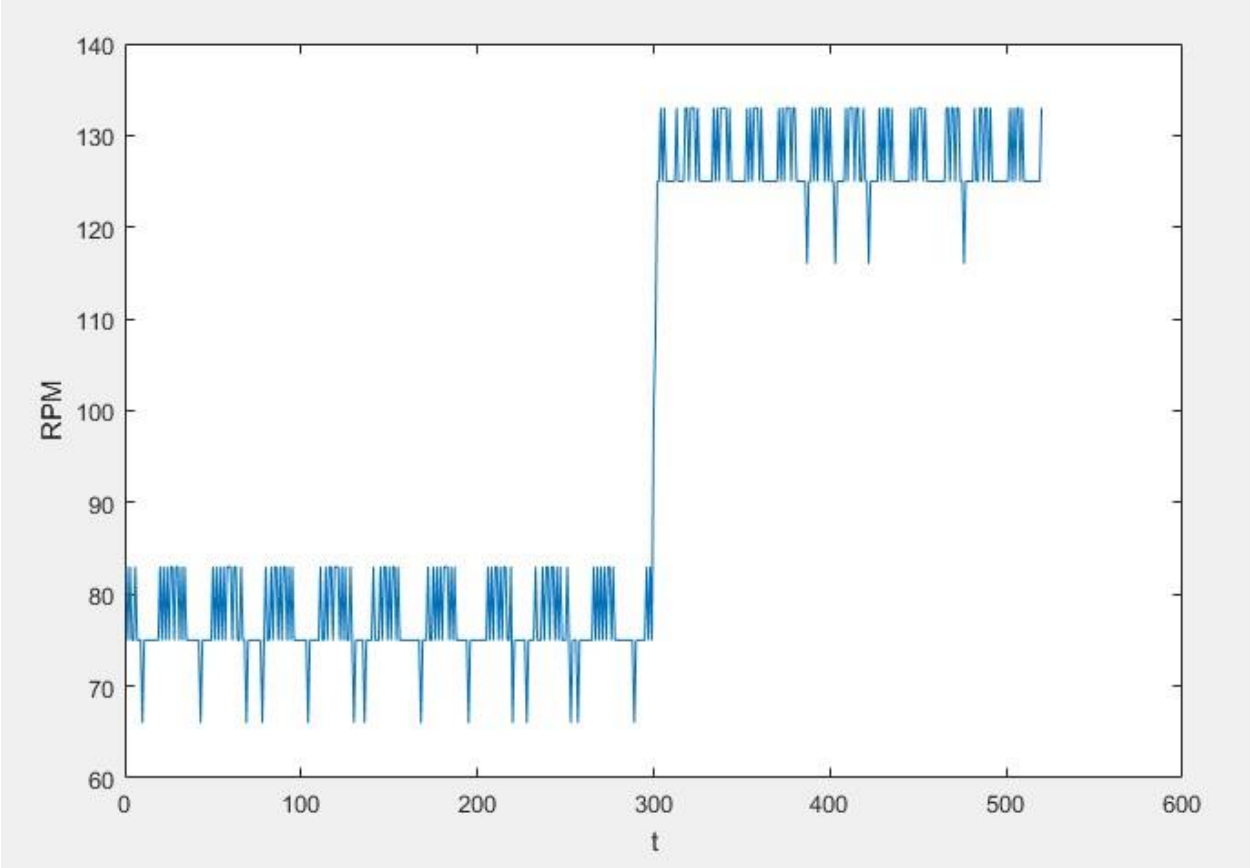
PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	

```
#if defined(_AVR_ATmega328P_)
  #define USE_20KHZ_PWM
#endif
Encoder knobLeft(2, 11); Encoder
knobRight(3, 4);
void setup() {
   pinMode(PWM_L,  OUTPUT);//pinii pe care se da comanda
   pinMode(PWM_R,  OUTPUT);
    Serial.begin(9600);
   Serial.println("TwoKnobs Encoder Test:");
}
long positionLeft  = 999; long
positionRight = 999;

void loop() {
  analogWrite(PWM_L, speed1); // se da comanda care este calculata
analogWrite(PWM_R, speed1);
  delay(1000);
  analogWrite(PWM_L, speed2 );
analogWrite(PWM_R, speed2 );   long
newLeft, newRight;
  newLeft = knobLeft.read();//se citesc impulsurile
newRight = knobRight.read();
  if (newLeft != positionLeft || newRight != positionRight) { //daca pozitia se tot schimba
Serial.print("Left = ");
    Serial.print(newLeft); //se afiseaza datele
    Serial.print(", Right = ");
    Serial.print(newRight);
Serial.println();
    positionLeft = newLeft;//se da noua pozitie
positionRight = newRight;

  }
}
// if a character is sent from the serial monitor,
// reset both back to zero.
  if (Serial.available()) {
    Serial.read();
    Serial.println("Reset both knobs to zero");
knobLeft.write(0);
    knobRight.write(0);
  } }
```

PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	

b. Achizitie semnale

Datele obtinute in Serial Monitor au fost salvate in Excel cu extensia .csv si introduse in Matlab. In urma plotarii datelor a rezultat graficul.



c. Identificare / Modelare analitica

Pe baza graficului am facut identificarea rezultand o functie de transfer pentru un sistem de ordin intai. Pentru a determina functia de transfer, am aproximat datele de pe grafic astfel : u=150-100=50 deoarece intrarea este o treapta intre 100 si 150(rpm) si apare la t=300 ;

y=125-75=50 conform graficului ;

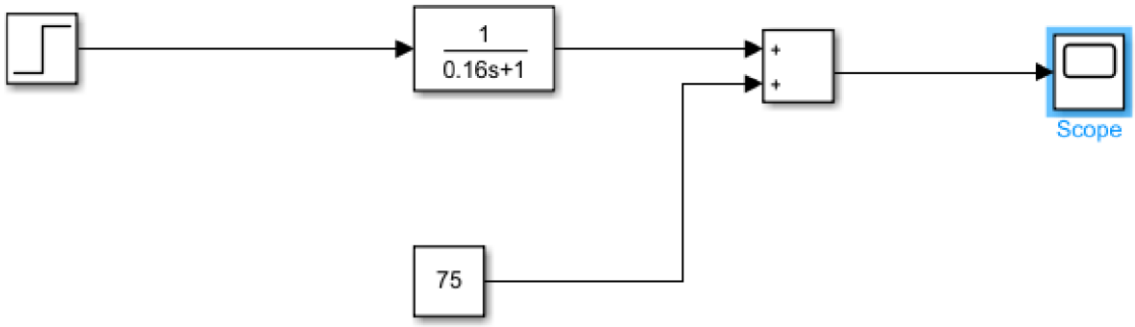
PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	

Factorul de proportionalitate K=y/u=1

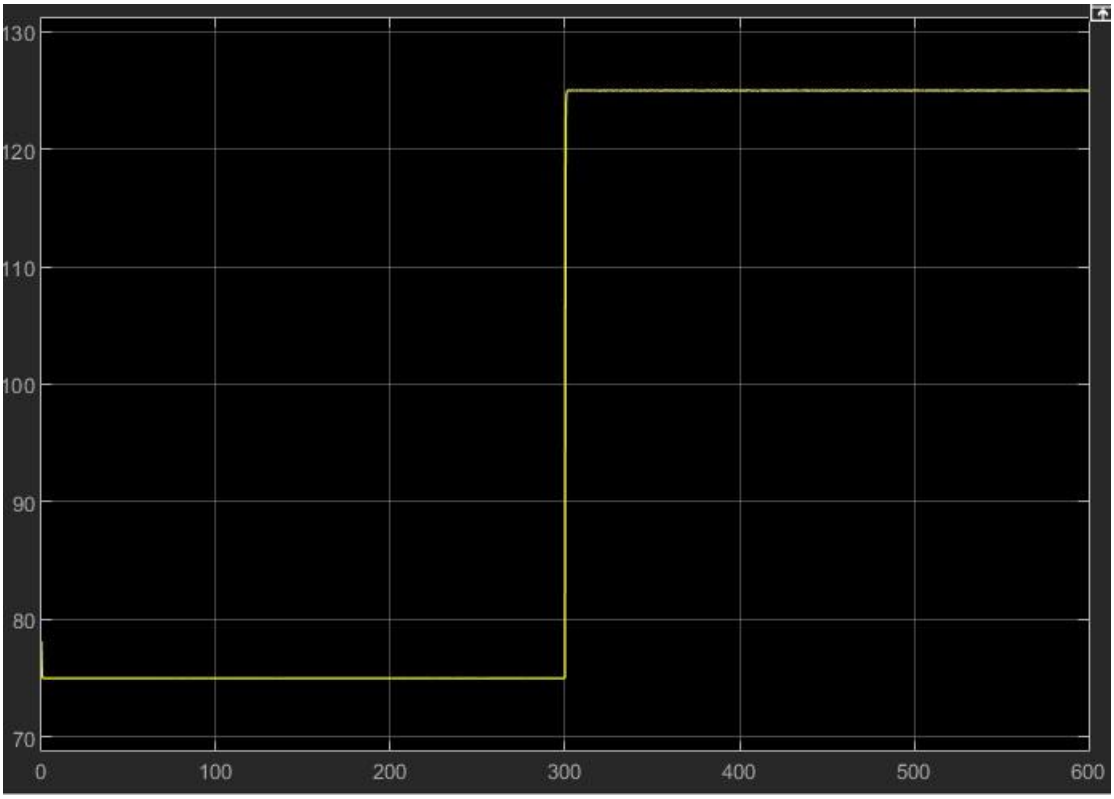
Perioada de esantionare este spusa in datele initiale ca fiind 16ms. Astfel am ales constanta de timp T=0.16sec.

H=tf(K,[T 1]) si am determinat o functie de transfer $H = \left(\frac{1}{0.16*s+1}\right)$

Ulterior, am verificat daca functia de transfer este corespunzatoare graficului pe baza caruia s-a calculat :



PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	



4. Proiectarea sistemului de control

Pentru proiectarea regulatorului am oscilat intre motoda modului si metoda simetriei. Metodele modulului si a simetriei sunt foarte utile in acordarea regulatoarelor pentru procese care nu contin timp mort si a caror structura contine una sau cel mult doua constante de timp dominante si o constata de timp mica, T_Σ.

Acordul regulatoarelor presupune adoptarea unei forme “optime” buclei directe, in cazul proceselor rapide, fara timp mort. Metodele modulului si a simetriei asigura o comportare buna a sistemelor de reglare automata la anumite clase de referinta si perturbatii.

In cazul metodei modulului din forma optima $H_{d}^{*}(s) = \left(\frac{1}{2*T_\Sigma \ \ s(T_\Sigma \ +1)}\right)$ se deduce $H_R(s)$ in functie de diferitele variante ale partii fixate:

$$H_R(s) = \left(\frac{H_{d}^{*}(s)}{H(s)}\right)_f \quad ;$$

Astfel, folosind metoda modului am ajuns la un regulator de forma $H_R(s) = \left(\frac{1}{0.32*s}\right)$, folosind T_Σ =0.16.

Performantele se evalueaza analizand functia de transfer a sistemului in bucla inchisa.

$H_o = \frac{H_{d^*}(s)}{1+H_{d^*}(s)} = \frac{1}{0.0512*s*s+0.32*s+1}$

Din relatia anterioara, prin identificarea coeficientilor am obtinut:

$w_n = \frac{1}{\sqrt{2}*T_\Sigma}$ =4.4 1rad/sec;

$\zeta = \frac{1}{\sqrt{2}}$=0.78;

$t_r = \frac{4}{\zeta * w_n}$=8*$T_\Sigma$=1.28;

Suprareglajul: σ=4.3%

Performantele obtine, cum ar fi timpul de raspuns mic, suprareglajul redus fca ca metoda modulului sa fie atractiva atunci cand se doreste urmarirea unei referinte de tip treapta. In cazul unei referinte de tip rampa, performantele sunt cu atat mai bune cu cat constanta de timp este mai mica. Coeficientul la viteza poate fi estimate folosind formula:

$= \frac{w_n}{*} = \frac{1}{*T_\Sigma} = 3.125$

Cv ζ 22

De unde rezulta direct eroarea stationara la viteza ca fiind

$\zeta_{stv} = \frac{1}{2*Cv} = 2*T_\Sigma = 0.32$

In cazul metodei simetriei forma optima a buclei directe este data de:

$H_{d^*}(s) = \frac{4*T_\Sigma *s+1}{8*T_\Sigma^2 s2(T_\Sigma *s+1)}$, parametrii de accord ai regulatorului rezultand si acum prin identificare din relatia: $H_R(s) = \left(\frac{H_{d^*}(s)}{H_f(s)}\right)$;

In acest caz, performantele superioare se obtin la urmarirea unei referinte de tip rampa

$$\zeta_{stv} = 0$$

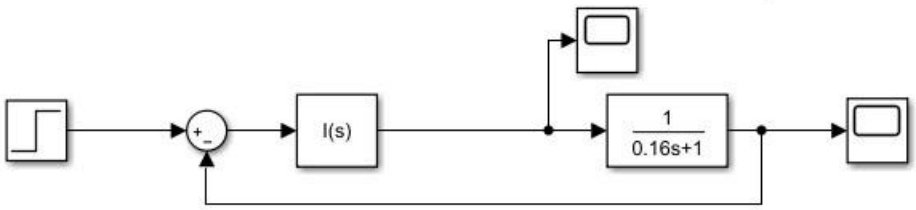
In cazul urmaririi unei referinte de tip treapta, acordarea regulatorului prin metoda simetriei duce la performante proaste, caracterizate printr-un suprareglaj ridicat, de aproximativ 43%, si un tip e raspuns $t_r = 11 * T_\Sigma$.

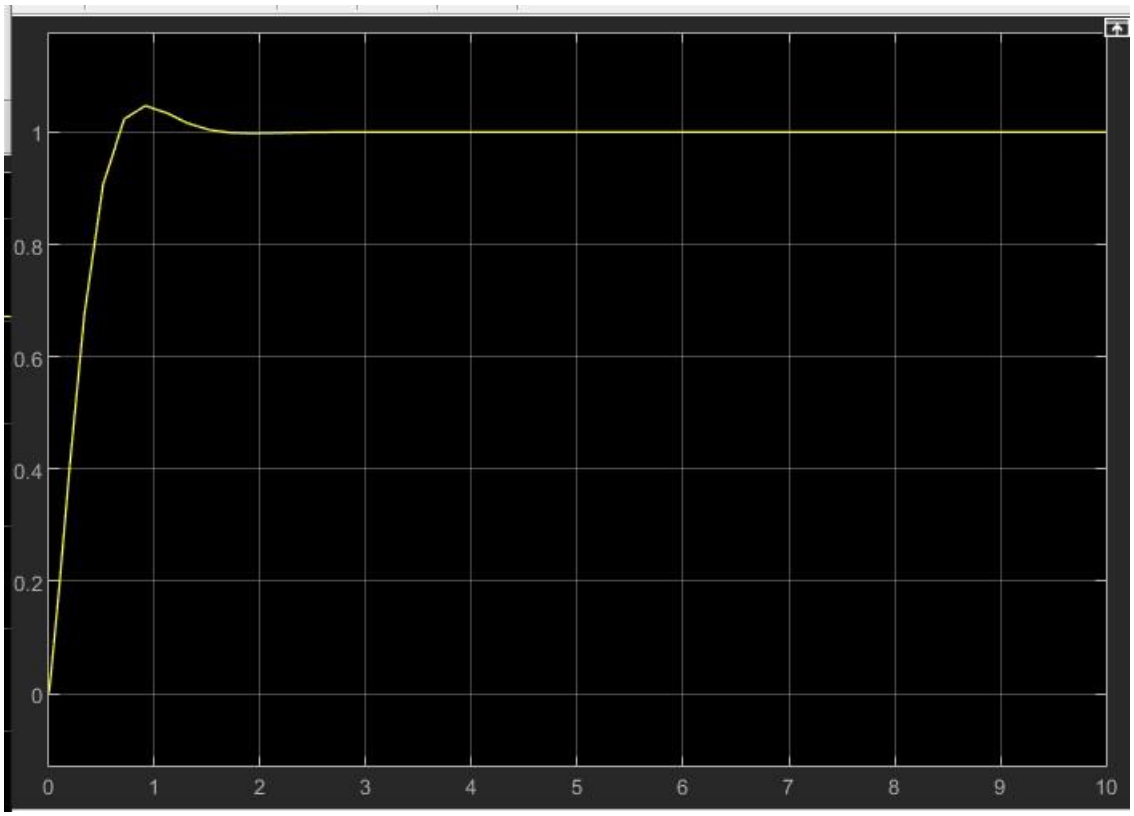
5. Implementarea sistemului de control

Pentru implementarea sistemului de control am realizat doua scheme in simulink, pentru cele doua regulatoare rezultate anterior.

Metoda modulului

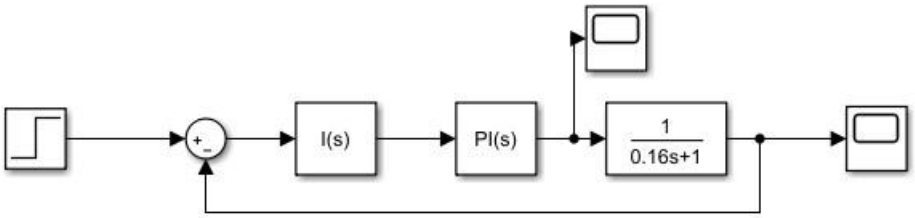
PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	

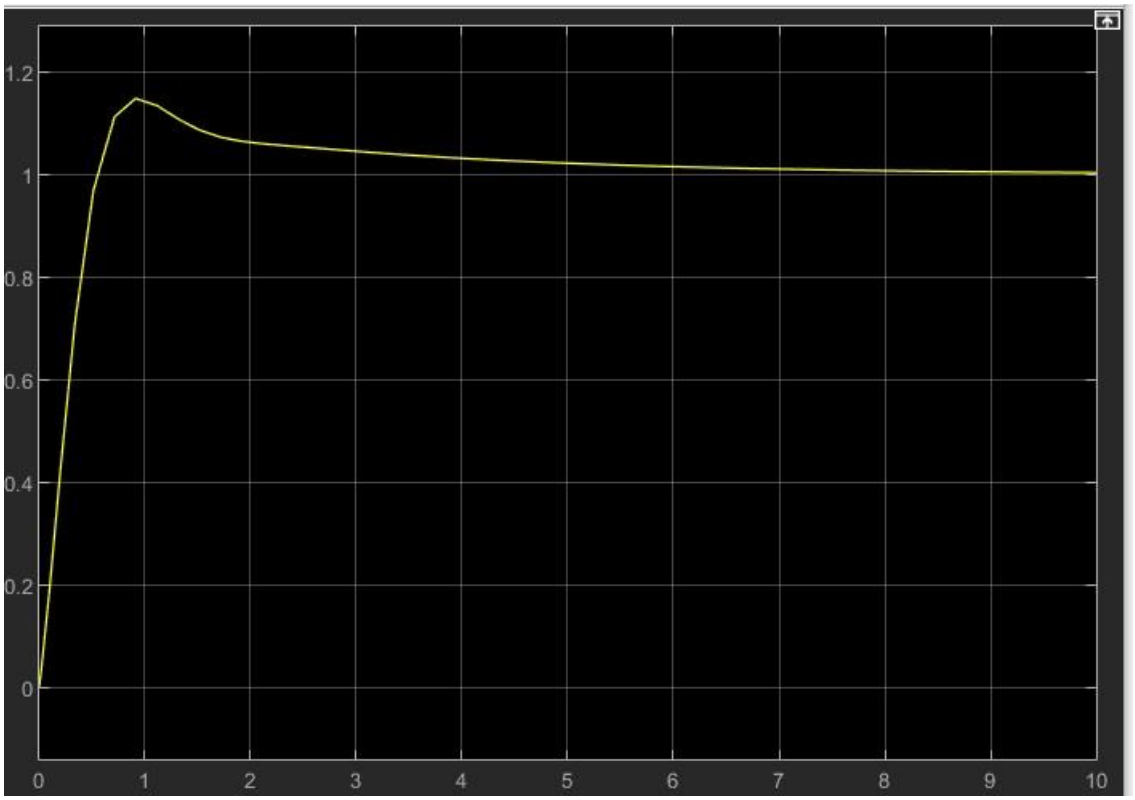




Metoda simetriei

PROIECT INGINERIA REGLARII AUTOMATE II					
NUME student	Ghiran Lorena	GRUPA:	30131	Nota	





Se observa ca suprareglajul in ambele cazuri nu este deranjant, dar timpul de raspuns este mai mic pentru regulatorul calculat prin metoda modulului.

Am discretizat functiile regulatoarelor folosind metoda zero order hold

```
hc1=tf([1],[0.32 0]); hc2=tf([0.64
1],[0.2048 0 0]);
te1=0.32; te2=0.2048;
hcz1=c2d(hc1,te1,'zoh')
hcz2=c2d(hc2,te2,'zoh')
```

Rezultand regulatorul discretizat:

$$Hcz1=\frac{1}{1-z^{-1}}\quad;\quad hcz2=\frac{0.7424*z-0.5376}{z*z-2*z+1}$$

Cod arduino pentru controlul vitezei si al pozitiei:

```
#include <Wire.h>
#include <Encoder.h>
ZumoMotors motors;
Encoder knobRight(3, 4);//encoder pe motorul drept
#define PinM_R 9 unsigned long
durata_m_dreapta;
float c[3]={0,0,0}; float
e[3]={0,0,0}; int ref=20;
//Viteza dorita void
setup()
{
 pinMode(PinM_R, OUTPUT);
 Serial.begin(9600);
 Serial.println("MotorDrept e[0] c[0]");
delay(1000);
}
long newRight1=0; void
loop()
{
 durata_m_dreapta=pulseIn(PinM_R, HIGH);
 if(millis()<7000){        //timp de 7 sec
 long newRight = knobRight.read();//se citesc impulsurile si se calculeaza viteza
e[0] = ref - abs(newRight);//eroarea este diferenta dintre referinta si viteza citita
Serial.print(e[0]);//se afiseaza   c[0]=c[1]+e[0];//se aplica comanda
Serial.print(c[0]);//se afisaza
 if(c[0]>400)//se pune o limitare;comanda sa nu depasasca capacitatea maxima a motoarelor
 {
```

```
   c[0]=400;
  }

  if (c[0]<-400)
  {
   c[0]=-400;
  }
analogWrite(PinM_R,c[0]); //se aplica comanda pe motor
for(int i=2;i>0;i--) //se realizeaza shiftarea
{   c[i]=c[i-
1];
  e[i]=e[i-1];
}
 knobRight.write(0); //se reseteaza la 0 impulsurile
if(millis()>7000){  //dupa 7 sec
analogWrite(PinM_R,0); //se opreste motorul
 }
}

}
```

Cod Arduino pentru detectare miscare, folosind senzorul de proximitate. Motorul robotului functioneaza normal, iar cand senzorul detecteaza miscare, motorul se opreste. Se observa in Serial Monitor ca atunci cand senzorul nu detecteaza miscare, impulsurile au valoarea 1023, iar cand apare miscare valorile sunt mult mai mici.

```
#include <ZumoShield.h>
ZumoMotors motors; void
setup() {
Serial.begin(9600);
}
void loop() {
int detectare = analogRead(A3);//se citeste senzorul pe pin-ul A3
Serial.println(detectare);//se afisaza  datele primite delay(10);
if (detectare<500)//daca impulsurile<500 inseamna ca impulsurile sunt 181(exista miscare)
 {
   motors.setRightSpeed(0);//motorul se opreste
 }
 else  //daca impulsurile>500 inseamna ca datele sunt 1023(nu exista miscare)
  {
   motors.setRightSpeed(134);//motorul are viteza e 134
```

```
  }
}
```

Cod Arduino pentru line follower:

```
#include <ZumoMotors.h>
#include <QTRSensors.h>
#include <ZumoReflectanceSensorArray.h>
#include <Pushbutton.h>
ZumoReflectanceSensorArray reflectanceSensors;
ZumoMotors motors;
Pushbutton button(ZUMO_BUTTON);
#define set_motors() = (setLeftSpeed,setRightSpeed);
#define NUM_SENSORS 6
#define KP 1 #define KD 0 unsigned int
sensorValues[NUM_SENSORS];
void setup()
{
  reflectanceSensors.init();   delay(500);   pinMode(13, OUTPUT);
pinMode(2, OUTPUT);   digitalWrite(13, HIGH);       // aprinde led
cand s-au calibrat senzorii
  pinMode(2,HIGH);
  unsigned long startTime = millis();
delay(1000);
}
void loop()
{
```

```
 unsigned int sensors[6]; // vector cu cei 6 senzori
unsigned int last_proportional=0;   long
integral=0;   while(1)
 {//pornim cu robotul pus pe linie si senzorii calibrati     unsigned int
position = reflectanceSensors.readLine(sensorValues);    // P ar trebui
sa fie 0 cand suntem cu centrul robotului pe linie     int proportional =
((int)position) - 2500;     int derivative = proportional -
last_proportional;     integral += proportional;     // pastram ultima
pozitie     last_proportional = proportional;
//daca val data de senzori e negativa, robotul merge la stanga,daca e neg la dreapta
int power_difference = proportional + integral + derivative;     // Compute the
actual motor settings.  We never set either motor
   // to a negative value.     const
int max = 250;
if(power_difference > max)
{power_difference = max;}     else
if(power_difference < -max)
{power_difference = -max;}     else
if(power_difference < 0)
    {motors.setLeftSpeed(max+power_difference);
                motors.setRightSpeed (max);}
else
    {motors.setLeftSpeed(max);
motors.setRightSpeed (max-power_difference);}}}
```