**ĐẠI HỌC QUỐC GIA**

**ĐẠI HỌC BÁCH KHOA TP HỒ CHÍ MINH**

🙞∙∙∙☼∙∙∙🙜



**BÀI TẬP LỚN MÔN NHẬP MÔN ĐIỀU KHIỂN THÔNG MINH**

ĐỀ TÀI: **Thực hiện bộ điều khiển PD mờ giữ cân bằng thanh và bóng.**

**LỚP L01--- NHÓM 02 --- HK 212**

**NGÀY NỘP 31/05/2022**

**Giảng viên hướng dẫn: Thầy Huỳnh Thái Hoàng.**

|  |  |  |
| --- | --- | --- |
| **Sinh viên thực hiện** | **Mã số sinh viên** | **Điểm số** |
| Đoàn Huỳnh Quát | 1914815 |  |
| Phan Thanh Huy | 1913551 |  |
| Đỗ Ngọc Tư | 1915846 |  |

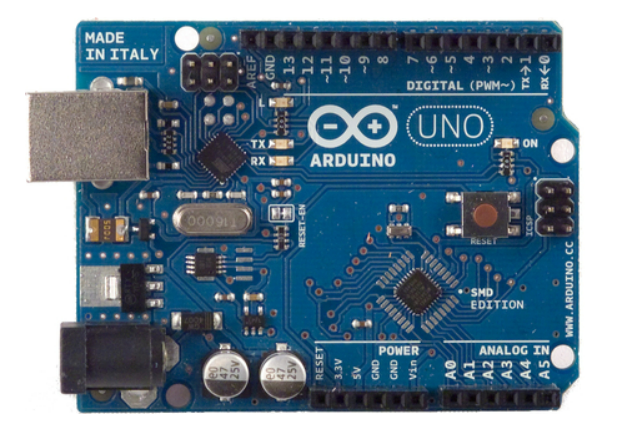
*Thành phố Hồ Chí Minh – 2021*

**I, Giới thiệu các thiết bị sử dụng:**

1. **Arduino UNO R3:**

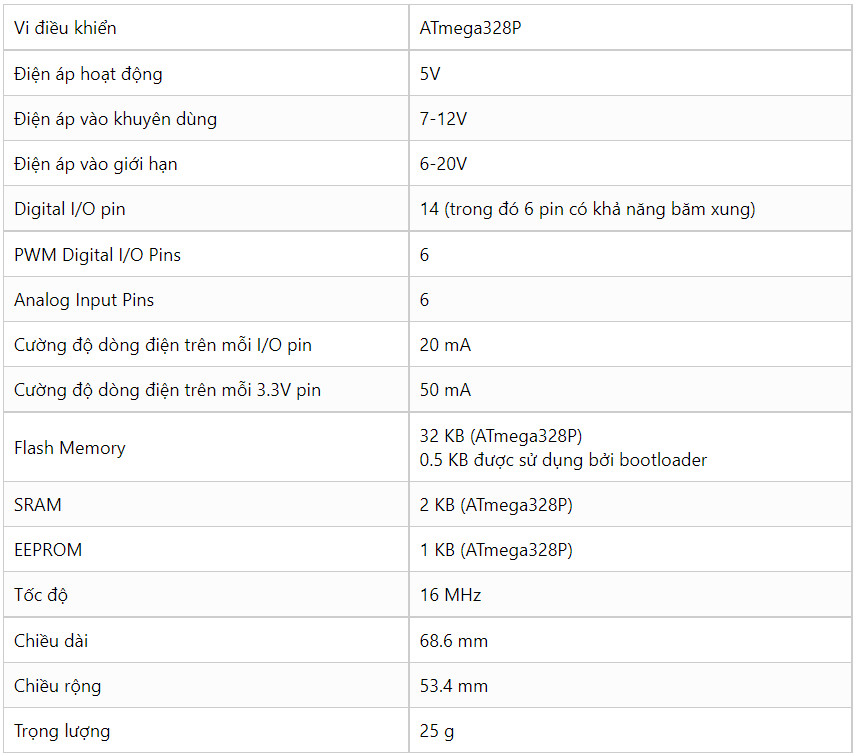
- Thiết kế tiêu chuẩn của Arduino UNO sử dụng vi điều khiển ATmega328.

- Arduino UNO có thể được cấp nguồn 5V thông qua cổng USB hoặc cấp nguồn ngoài với điện áp khuyên dùng là 7-12V DC và giới hạn là 6-20V. Thường thì cấp nguồn bằng pin vuông 9V là hợp lí nhất.



*Hình 1: Arduino UNO R3*

-Thông số kỹ thuật của arduino UNO R3:

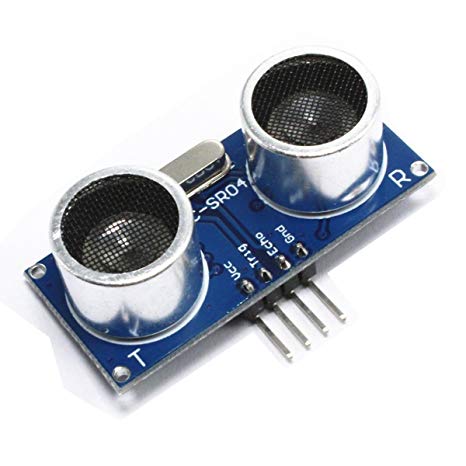


1. **Cảm biến siêu âm HS- CR04:**

- Cảm biến siêu âm HC-SR04 hay module cảm biến đo khoảng cách HC-SR04 bằng sóng siêu âm là một module arduino nhỏ gọn, rất thích hợp cho chúng ta tìm hiểu nghiên cứu về hoạt động và nguyên lý của dòng cảm biến siêu âm.

- Cảm biến siêu âm HC-SR04 thường được kết hợp với các bộ arduino, PIC, AVR,… để chạy một số ứng dụng như : phát hiện vật cản trên xe robot, đo khoảng cách vật,…

- Chính vì là một cảm biến siêu âm dạng module, cho nên hầu như ứng dụng hay độ chính xác của cảm biến đều phụ thuộc vào phần code mà người sử dụng lập trình và nạp vào bản mạch điều khiển.



*Hình 1: Cảm biến siêu âm HS- CR04*

1. **Động cơ JGA25- 370 130RPM :**

Động cơ DC giảm tốc GA25 Encoder thường được sử dụng trong các ứng dụng cần xác định tốc độ, vị trí, chiều quay của động cơ DC: Robot mê cung, robot xe hai bánh tự cân bằng…

Động cơ giảm tốc GA25-370 có encoder cấu tạo thực tế là động cơ giảm tốc DC bình thường có gắn thêm phần Encoder để có thể trả xung về vi điều khiển giúp xác định vị trí, vận tốc,... từ đó vi điều khiển tác động lại động cơ qua mạch công suất, sử dụng các thuật toán để điều khiển ngược lại động cơ.

Thông số kĩ thuật:

* Điện áp cấp cho động cơ hoạt động: 3 – 12VDC
* Điện áp cấp cho Encoder hoạt động: 3.3VDC
* Đĩa Encoder 11 xung, hai kênh A-B
* Đường kính động cơ: 25mm
* Đường kính trục: 4mm



*Hình 3: Động cơ JGA25- 370 130RPM*

1. **Mạch cầu H L298 :**

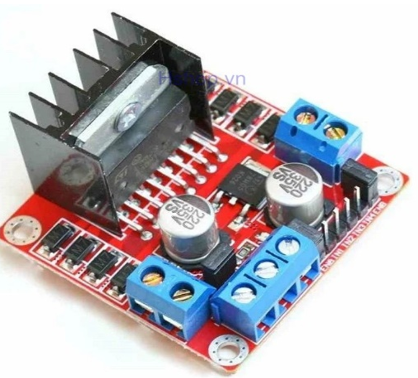
Mạch điều khiển động cơ DC L298 có khả năng điều khiển 2 động cơ DC, dòng tối đa 2A mỗi động cơ, mạch tích hợp diod bảo vệ và IC nguồn 7805 giúp cấp nguồn 5VDC cho các module khác (chỉ sử dụng 5V này nếu nguồn cấp <12VDC). Mạch L298 có khả năng điều khiển 2 động cơ quay hai chiều với mực điện áp 12V, đồng bộ với động cơ DC được sử dụng.

Thông số kỹ thuật:

* IC chính: L298 - Dual Full Bridge Driver
* Điện áp đầu vào: 5~30VDC
* Công suất tối đa: 25W 1 cầu (lưu ý công suất = dòng điện x điện áp nên áp cấp vào càng cao, dòng càng nhỏ, công suất có định 25W).
* Dòng tối đa cho mỗi cầu H là: 2A
* Mức điện áp logic: Low -0.3V~1.5V, High: 2.3V~Vss
* Kích thước: 43x43x27mm

Các chân của L298:

* ENB: cho phép băm xung động cơ A
* IN1: băm xung cho động cơ A chạy thuận
* IN2: băm xung cho động cơ A chạy nghịch
* IN3: băm xung cho động cơ B chạy thuận
* IN4: băm xung cho động cơ b chạy nghịch
* ENB: cho phép băm xung động cơ B



*Hình 4: Mạch cầu H L298*

**II. Cơ sở lý thuyết:**

1. **Điều khiển PD mờ:**

Bộ điều khiển PD mờ là một phần của bộ điếu khiển PID mờ sau khi lược bỏ thành phần tích phân để ứng dụng vào một số trường hợp. Thực tế đã chứng tỏ bộ điều khiển PD mờ tỏ ra vượt trội hơn so với PID mờ trong nhiều trường hợp nhất định

Bộ điều khiển PD mờ thường được sử dụng trong các trường hợp sau đây:

* Đối tượng có khâu tích phân lý tưởng
* Ổn định hóa trạng thái của đối tượng xung quanh điểm cân bằng

Do hệ chúng ta khảo xác là hệ Ball and Beam, bóng cần ổn định xung quanh điểm cân bằng, nên chọn PD mờ là phù hợp.

1. **Thiết kế bộ điều khiển PD mờ:**

***Bước 1:*** Vẽ sơ đồ khối HTDK xác định tầm giá trị của:



*Hình 4: Sơ đồ điều khiển PD mờ*

* Biến vào: sai số () và vi phân sai số ()
* Biến ra: tín hiệu điều khiển 

***Bước 2:*** Xác định hệ số chuẩn hóa biến vào, biến ra về miền giá trí [-1,1]

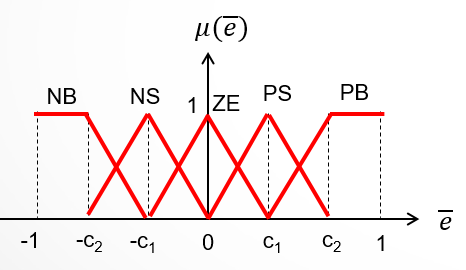
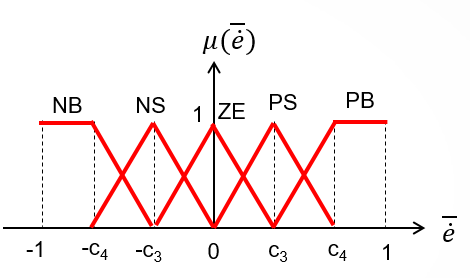
+ sai số (e) : E = y(t) - r(t) mà −20 ≤ 𝑦(𝑡), 𝑟(𝑡) ≤ 20 (c𝑚)

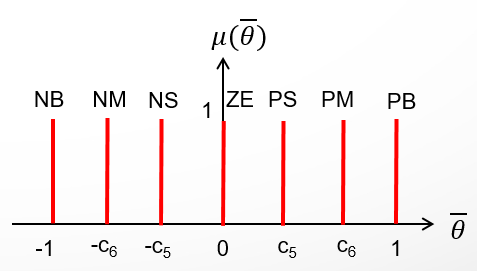
=> −40 ≤ 𝑒 ≤ 40 (c𝑚) => K1 = 1/40

+ vi phân sai số () : −1 ≤  ≤ 1 (𝑚/𝑠) => K2 = 1 ( điều chỉnh khi mô phỏng)

+ tín hiệu điều khiển 𝜃 : −𝜋/6 ≤ 𝜃 ≤ 𝜋/6 (𝑟𝑎𝑑) => 𝐾𝑢=𝜋/6

***Bước 3:*** Định nghĩa các giá trị ngôn ngữ cho biến vào và biến ra định lượng các giá trị ngôn ngữ bằng các tập mờ.

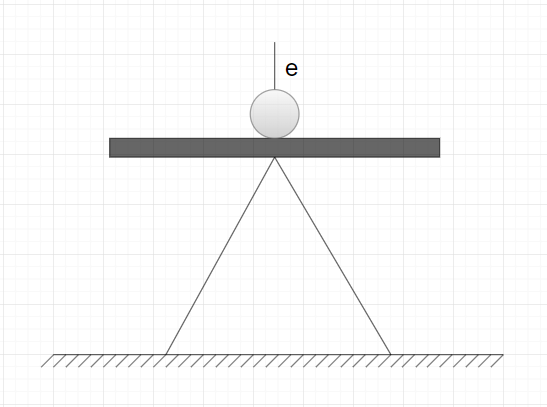
 



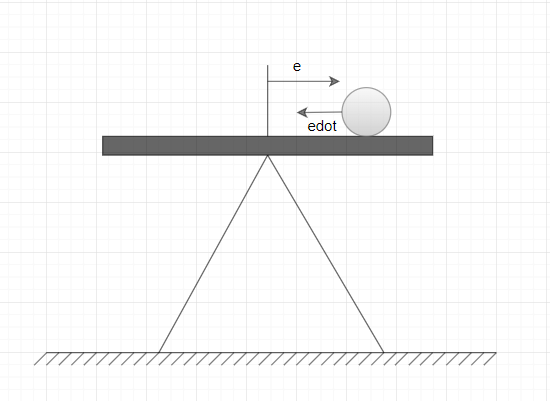
Các giá trị từ sẽ được tinh chỉnh thông qua thực nghiệm.

***Bước 4:*** Xây dựng hệ quy tắc mờ bằng cách vẽ hình minh họa để có ý tưởng đưa ra một số quy tắc điển hình, sau đó dùng tính liên tục của hệ mờ và tính đối xứng để suy ra các quy tắc còn lại.

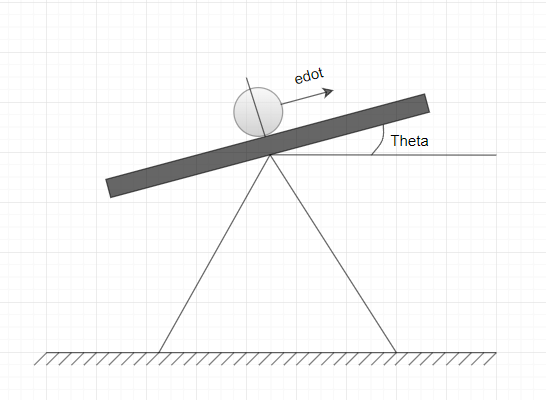
e: ZE, : ZE =>  : ZE



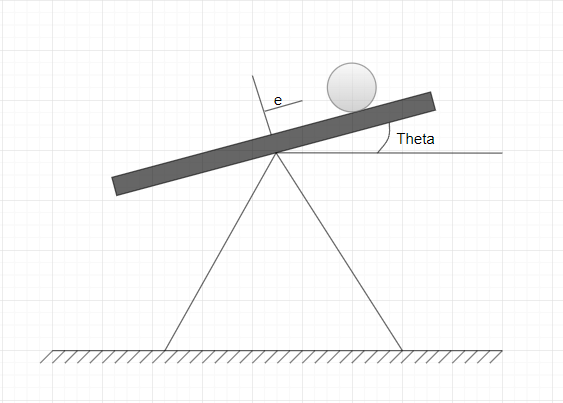
e: PS, : NS =>  : ZE



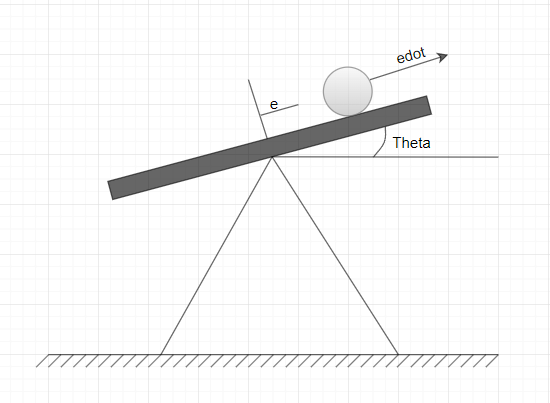
e: ZE, : PE =>  : PS



e: PS, : ZE =>  : PS



e: PS, : PS =>  : PM



Từ đó ta suy ra các quy tắc còn lại theo bảng sau:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| theta | | e | | | | |
| NB | NS | ZE | PS | PB |
| edot | NB | NB | NB | NM | NS | ZE |
| NS | NB | NM | NS | ZE | PS |
| ZE | NM | NS | ZE | PS | PM |
| PS | NS | ZE | PS | PM | PB |
| PB | ZE | PS | PM | PB | PB |

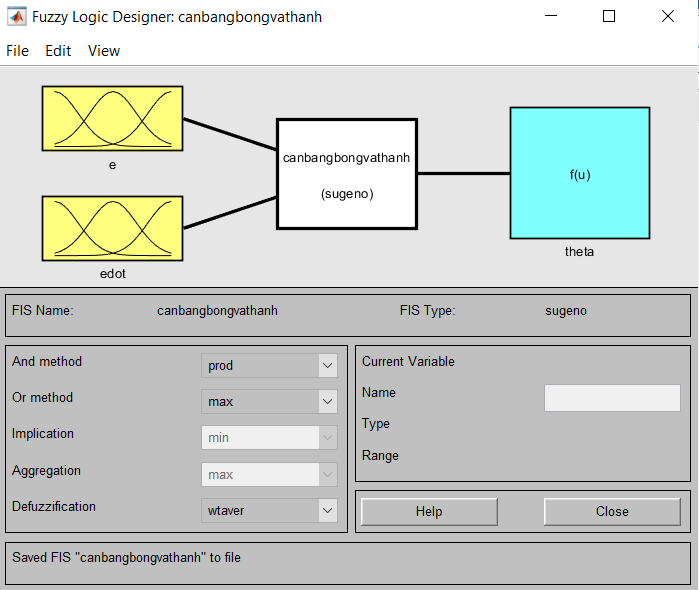
***Bước 5***: Chọn PP suy diễn MAX- PROD

***Bước 6***: Chọn PP giải mờ trung bình có trọng số

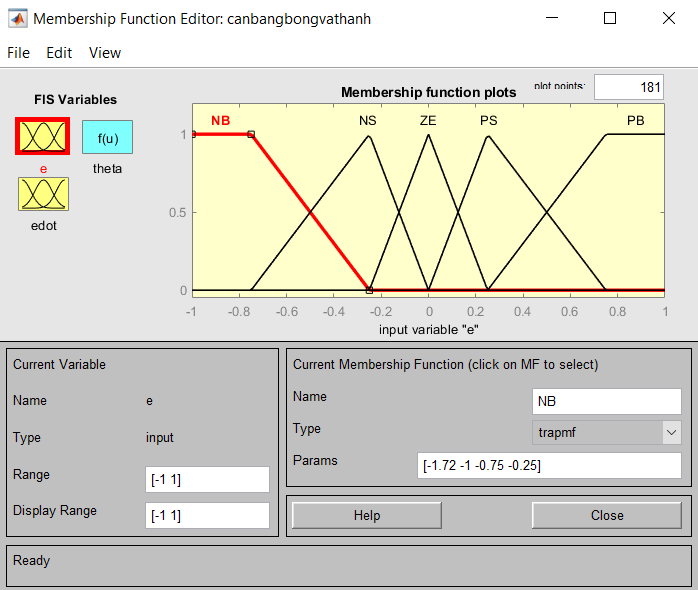
***Bước 7:*** Mô phỏng hoặc thực nghiệm đánh giá kết quả, tinh chỉnh các thông số của BĐK để đạt chất lượng mong muốn

1. Mô phỏng bằng Matlab:

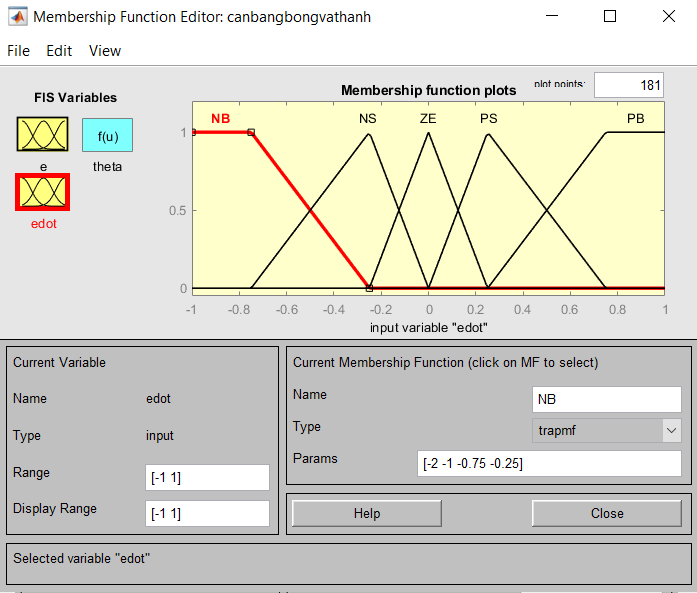
Khối fuzzy logic:



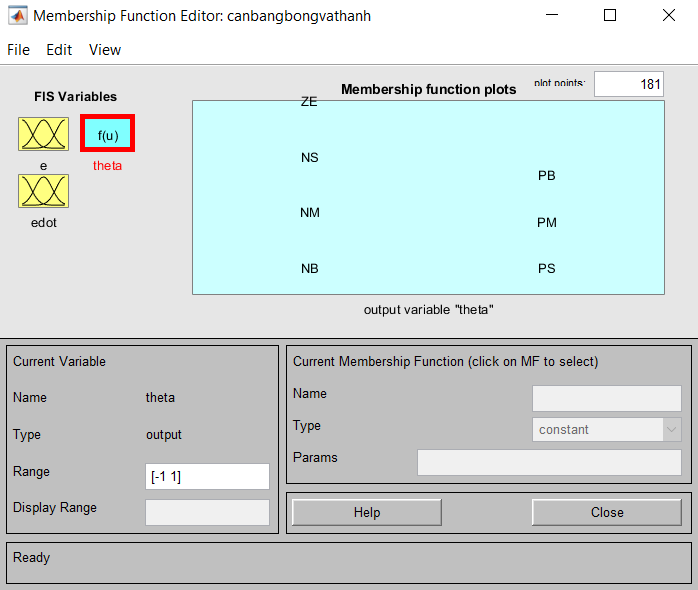
Input: e



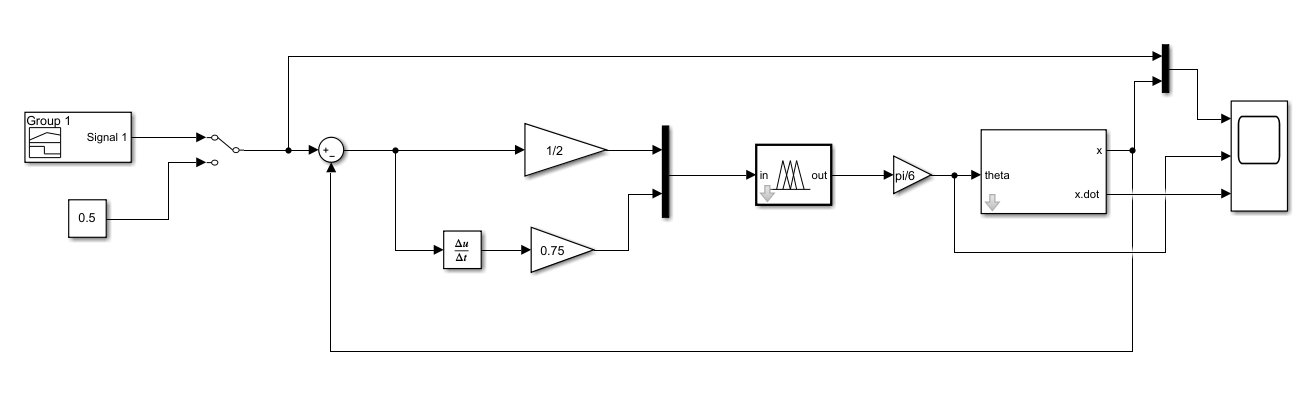
Input: 



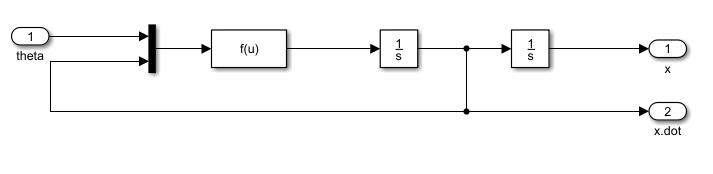
Output: 



Mô hình simulink:

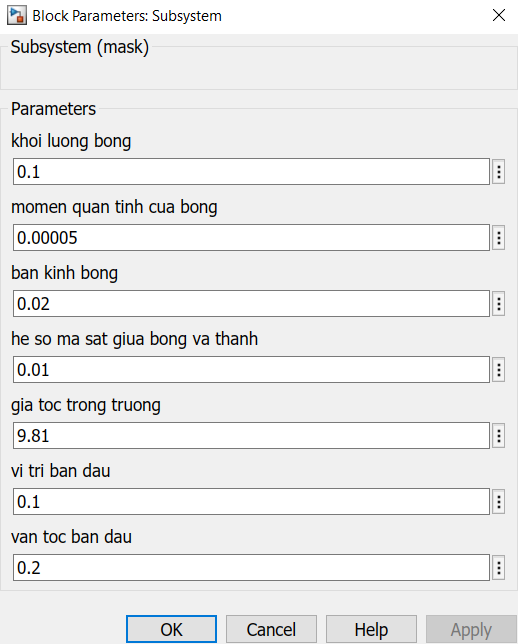


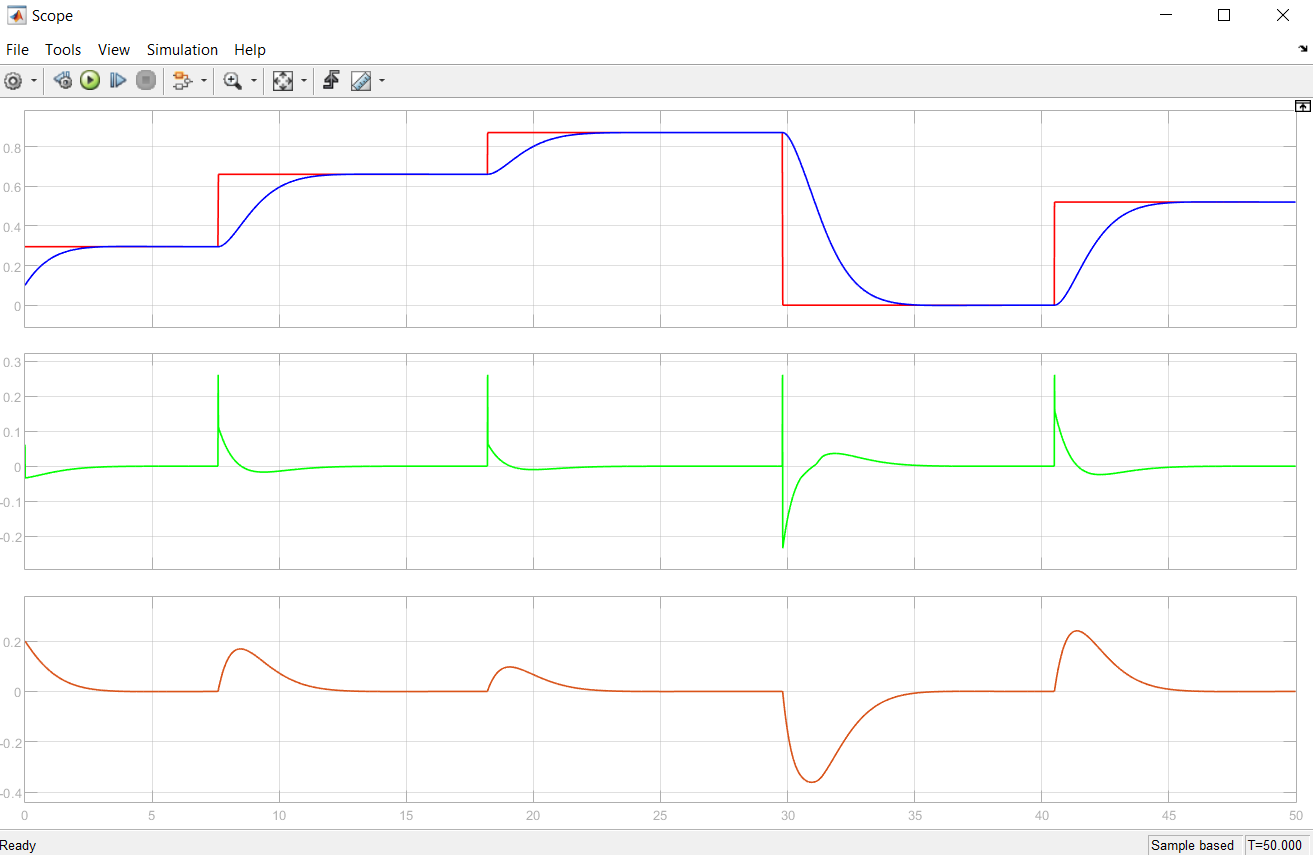
Hệ bóng và thanh:



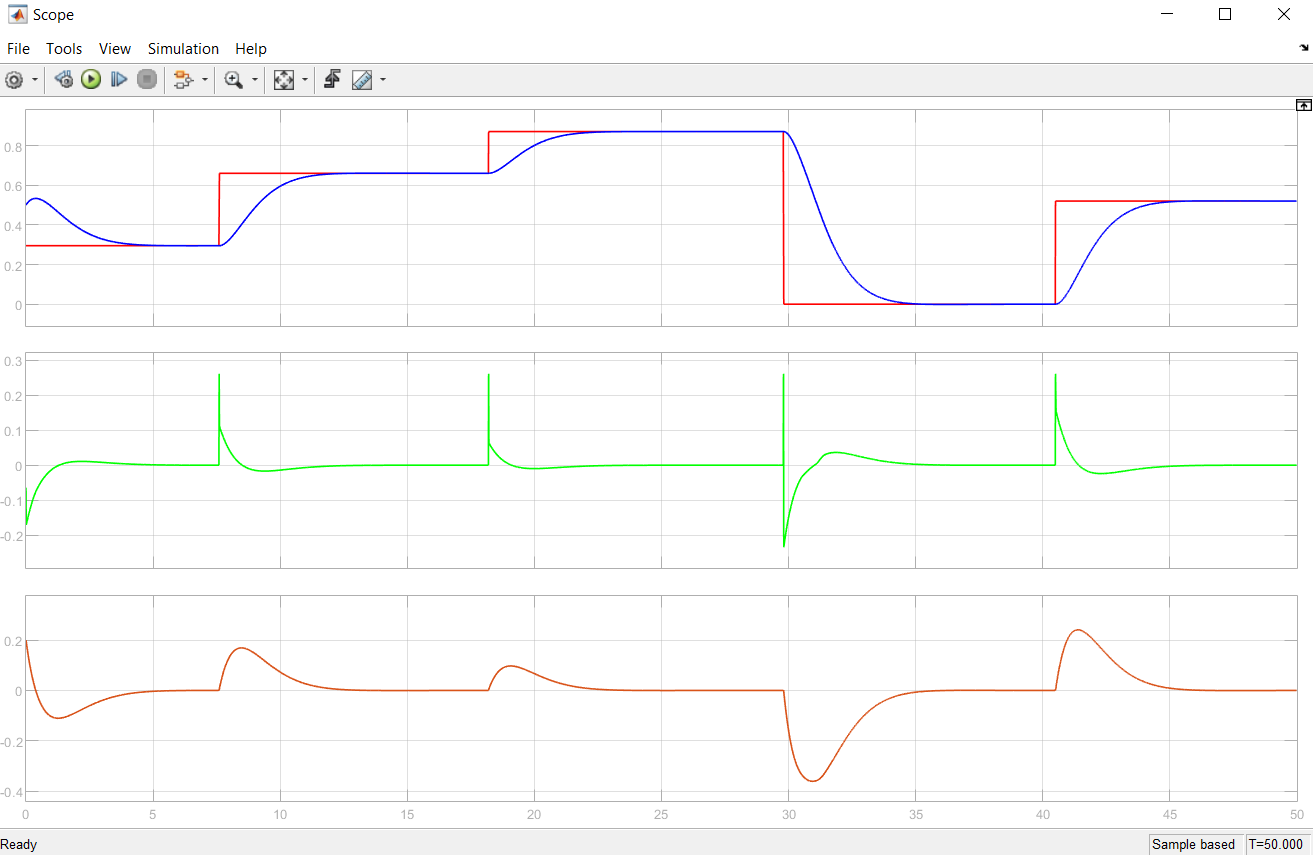
Kết quả mô Phỏng:

Thông số của hệ:





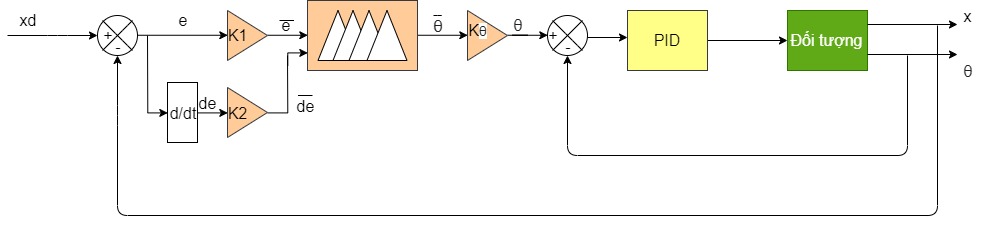
*Vị trí đầu =0.1 ; vận tốc đầu =0.2*



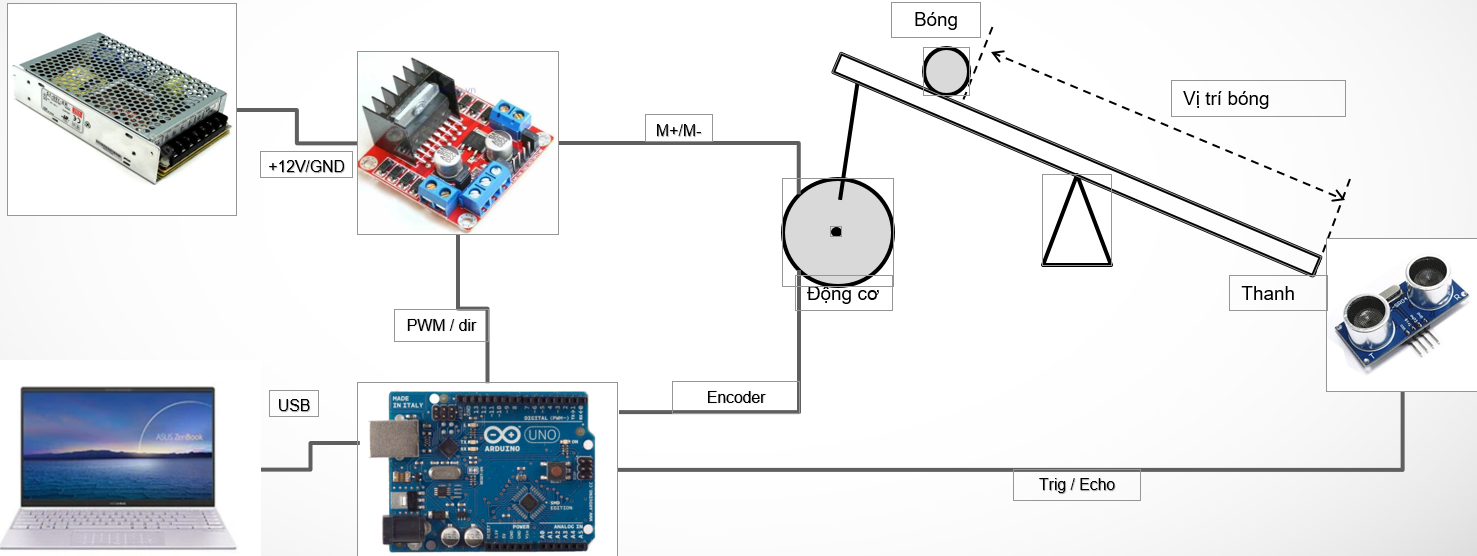
*Vị trí đầu =0.5 ; vận tốc đầu =0.2*

**III. Xây dựng hệ thống thực tế:**

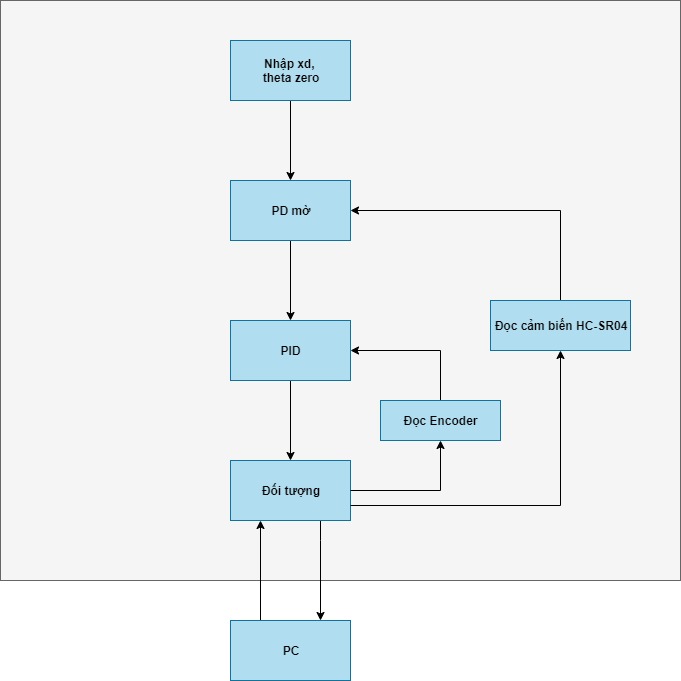
1. **Sơ đồ khối hệ thống:**



1. **Sơ đồ nối dây hệ thống:**

****

1. **Sơ đồ giải thuật:**



1. **Code chương trình**

Khai báo các biến:

int ena=11;

int in1=9;

int in2=10;

int giatri;

int bamxung;

const int interrupt0=0;

const int phase\_a= 2;

const int phase\_b=3;

float angle\_ref=0;

const int trigPin = A2;

const int echoPin = A3;

long duration=0;

float distance=0;

float e\_x = 0;

float n = 0;

float xd = 20;

float edot\_x = 0;

float pre\_e\_x = 0;

int x0=18;

long prevT=0;

int pos=0;

int pos\_zero=0;

float eprev=0;

float eintegral=0;

float u=0;

int target=0;

float pwr=0;

int dir=0;

Quy định các chân truyền nhận dữ liệu:

void setup() {

// put your setup code here, to run once:

Serial.begin(9600);

pinMode(ena,OUTPUT);

pinMode(in1,OUTPUT);

pinMode(in2,OUTPUT);

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

// khoi tao ham ngat

// ngat 0 tuong ung chan 2 cua arduino

attachInterrupt (interrupt0, Read\_Encoder, RISING);

pinMode(phase\_a, INPUT\_PULLUP);

pinMode(phase\_b, INPUT\_PULLUP);

}

Chương trình đọc encoder:

void Read\_Encoder(){

if (digitalRead(3)>0) pos++;

else pos--;

}

Chương trình đọc cảm biến siêu âm:

void ReadHC\_SRF04(){

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds

duration = pulseIn(echoPin, HIGH);

// Calculating the distance

// toc do am thanh trong khong khi la 340 m/s.

// do thoi gian tinh từ lúc phat hiện sóng cho tới khi sóng phản xạ lại

distance = duration \* 0.034 / 2;

}

Chương trình bộ PD mờ:

float hlt\_hinhthang(float x,float L, float C1,float C2, float R)

{

float y;

if(x < L)

y=0;

else if (x <C1)

y= (float)((x-L)/(C1-L));

else if(x <C2)

y=1;

else if(x<R)

y= (float)((R-x)/(R-C2));

else

y=0;

return y;

}

float e\_NB, e\_NS, e\_ZE, e\_PS, e\_PB, edot\_NB, edot\_NS, edot\_ZE, edot\_PS, edot\_PB;

float y\_NB, y\_NM, y\_NS, y\_ZE, y\_PS, y\_PM, y\_PB;

float beta1=0, beta2=0, beta3=0, beta4=0, beta5=0, beta6=0, beta7=0;

float beta8=0, beta9=0, beta10=0,beta11=0,beta12=0,beta13=0,beta14=0;

float beta15=0,beta16=0,beta17=0,beta18=0,beta19=0,beta20=0,beta21=0;

float beta22=0,beta23=0,beta24=0,beta25=0;

float fuzzy\_from\_dis\_to\_angle(float e, float edot)

{ float y;

e\_NB= hlt\_hinhthang(e,-1.5,-1,-0.75,-0.2);

e\_NS = hlt\_hinhthang(e,-0.75,-0.2,-0.2,0);

e\_ZE = hlt\_hinhthang(e,-0.2,0,0,0.2);

e\_PS = hlt\_hinhthang(e,0,0.2,0.2,0.75);

e\_PB = hlt\_hinhthang(e,0.2,0.75,1.25,1.5);

edot\_NB = hlt\_hinhthang(edot,-1.5,-1,-0.75,-0.2);

edot\_NS = hlt\_hinhthang(edot,-0.75,-0.2,-0.2,0);

edot\_ZE = hlt\_hinhthang(edot,-0.2,0,0,0.2);

edot\_PS = hlt\_hinhthang(edot,0,0.2,0.2,0.75);

edot\_PB = hlt\_hinhthang(edot,0.2,0.75,1.25,1.5);

y\_NB = -1;

y\_NM = -0.6;

y\_NS = -0.2;

y\_ZE = 0;

y\_PS = 0.2;

y\_PM = 0.6;

y\_PB = 1;

beta1 = e\_NB\*edot\_NB;

beta2 = e\_NB\*edot\_NS;

beta3 = e\_NB\*edot\_ZE;

beta4 = e\_NB\*edot\_PS;

beta5 = e\_NB\*edot\_PB;

/\*e= NS; edot = NB->PB \*/

beta6 =e\_NS\*edot\_NB;

beta7 =e\_NS\*edot\_NS;

beta8 =e\_NS\*edot\_ZE;

beta9 =e\_NS\*edot\_PS;

beta10 =e\_NS\*edot\_PB;

/\*e= ZE; edot = NB->PB \*/

beta11 =e\_ZE\*edot\_NB;

beta12 =e\_ZE\*edot\_NS;

beta13 =e\_ZE\*edot\_ZE;

beta14 =e\_ZE\*edot\_PS;

beta15 =e\_ZE\*edot\_PB;

/\*e= PS; edot = NB->PB \*/

beta16 =e\_PS\*edot\_NB;

beta17 =e\_PS\*edot\_NS;

beta18 =e\_PS\*edot\_ZE;

beta19 =e\_PS\*edot\_PS;

beta20 =e\_PS\*edot\_PB;

/\*e= PB; edot = NB->PB \*/

beta21 =e\_PB\*edot\_NB;

beta22 =e\_PB\*edot\_NS;

beta23 =e\_PB\*edot\_ZE;

beta24 =e\_PB\*edot\_PS;

beta25 =e\_PB\*edot\_PB;

y=( beta1\*y\_NB+ beta2\*y\_NB+ beta3\*y\_NM+ beta4\*y\_NS+ beta5\*y\_ZE+

beta6\*y\_NB+ beta7\*y\_NM+ beta8\*y\_NS+ beta9\*y\_ZE+ beta10\*y\_PS+

beta11\*y\_NM+ beta12\*y\_NS+ beta13\*y\_ZE+ beta14\*y\_PS+ beta15\*y\_PM+

beta16\*y\_NS+ beta17\*y\_ZE+ beta18\*y\_PS+ beta19\*y\_PM+ beta20\*y\_PB+

beta21\*y\_ZE+ beta22\*y\_PS+ beta23\*y\_PM+ beta24\*y\_PB+ beta25\*y\_PB

);

y=constrain(y, -1, 1);

return y;

}

Sử dụng bộ PD mờ điều khiển tín hiệu băm xung:

float fuzzy\_from\_angle\_to\_PWM(float e, float edot)

{

float y;

e\_NB= hlt\_hinhthang(e,-1.5,-1,-0.75,-0.25);

e\_NS = hlt\_hinhthang(e,-0.75,-0.25,-0.25,0);

e\_ZE = hlt\_hinhthang(e,-0.25,0,0,0.25);

e\_PS = hlt\_hinhthang(e,0,0.25,0.25,0.75);

e\_PB = hlt\_hinhthang(e,0.25,0.75,1.25,1.5);

edot\_NB = hlt\_hinhthang(edot,-1.5,-1,-0.75,-0.25);

edot\_NS = hlt\_hinhthang(edot,-0.75,-0.25,-0.25,0);

edot\_ZE = hlt\_hinhthang(edot,-0.25,0,0,0.25);

edot\_PS = hlt\_hinhthang(edot,0,0.25,0.25,0.75);

edot\_PB = hlt\_hinhthang(edot,0.25,0.75,1.25,1.5);

y\_NB = -1;

y\_NM = -0.67;

y\_NS = -0.33;

y\_ZE = 0;

y\_PS = 0.33;

y\_PM = 0.67;

y\_PB = 1;

beta1 = e\_NB\*edot\_NB;

beta2 = e\_NB\*edot\_NS;

beta3 = e\_NB\*edot\_ZE;

beta4 = e\_NB\*edot\_PS;

beta5 = e\_NB\*edot\_PB;

/\*e= NS; edot = NB->PB \*/

beta6 =e\_NS\*edot\_NB;

beta7 =e\_NS\*edot\_NS;

beta8 =e\_NS\*edot\_ZE;

beta9 =e\_NS\*edot\_PS;

beta10 =e\_NS\*edot\_PB;

/\*e= ZE; edot = NB->PB \*/

beta11 =e\_ZE\*edot\_NB;

beta12 =e\_ZE\*edot\_NS;

beta13 =e\_ZE\*edot\_ZE;

beta14 =e\_ZE\*edot\_PS;

beta15 =e\_ZE\*edot\_PB;

/\*e= PS; edot = NB->PB \*/

beta16 =e\_PS\*edot\_NB;

beta17 =e\_PS\*edot\_NS;

beta18 =e\_PS\*edot\_ZE;

beta19 =e\_PS\*edot\_PS;

beta20 =e\_PS\*edot\_PB;

/\*e= PB; edot = NB->PB \*/

beta21 =e\_PB\*edot\_NB;

beta22 =e\_PB\*edot\_NS;

beta23 =e\_PB\*edot\_ZE;

beta24 =e\_PB\*edot\_PS;

beta25 =e\_PB\*edot\_PB;

TS=( beta1\*y\_NB+ beta2\*y\_NB+ beta3\*y\_NM+ beta4\*y\_NS+ beta5\*y\_ZE+

beta6\*y\_NB+ beta7\*y\_NM+ beta8\*y\_NS+ beta9\*y\_ZE+ beta10\*y\_PS+

beta11\*y\_NM+ beta12\*y\_NS+ beta13\*y\_ZE+ beta14\*y\_PS+ beta15\*y\_PM+

beta16\*y\_NS+ beta17\*y\_ZE+ beta18\*y\_PS+ beta19\*y\_PM+ beta20\*y\_PB+

beta21\*y\_ZE+ beta22\*y\_PS+ beta23\*y\_PM+ beta24\*y\_PB+ beta25\*y\_PB

TS=(beta1+beta2+beta3+beta4+beta5+beta6+beta7+beta8+beta9+beta10+beta11+beta12+beta13+beta14+

beta15+beta16+beta17+beta18+beta19+beta20+beta21+beta22+beta23+beta24+beta25);

y=TS/MS;

return y;

Hàm set motor:

void setMotor( int dir, int pwmVal)

{

//dir: direction rotate

//pwmVal: pwm speed

// enz: pwm pin

analogWrite(ena,pwmVal);

if(dir==1){

digitalWrite(in1,HIGH);

digitalWrite(in2,LOW);

}

else if(dir==-1){

digitalWrite(in1,LOW);

digitalWrite(in2,HIGH);

}

else{

digitalWrite(in1,LOW);

digitalWrite(in2,LOW);

}

}

void PID(int target){

float kp=1;

float ki=0.1;

float kd=0.1;

// time different

long currT=micros();

long et=currT-prevT;

float deltaT=((float)(currT-prevT))/1.0e6;

prevT=currT;

int e1=pos-target;

float de1=(e1-eprev)/deltaT;

eintegral=eintegral+e1\*deltaT;

// control signal

u=kp\*e1+kd\*de1+ki\*eintegral;

eprev=e1;

//if ((e\_x<0.1 )&(e\_x>-0.1 )) u=0;

}

Chương trình chính:

void loop() {

// put your main code here, to run repeatedly:

if(Serial.available())

{

String data=Serial.readString();

//x0=Serial.readStringUntil('\n').toInt();

x0=data.substring(0,2).toInt();

//pos\_zero=data.substring(2,2).toInt();

//pos\_zero=Serial.readString().substring(2,5).toInt();

//x0=Serial.readStringUntil('\n').toInt();

pos\_zero=data.substring(2,5).toInt();

}

ReadHC\_SRF04();

e\_x=(distance-x0-3)/32.67;

edot\_x=(e\_x-pre\_e\_x)\*5;

angle\_ref=(fuzzy\_from\_dis\_to\_angle(e\_x, edot\_x))\*1

//target=angle\_ref;

//if ((pos<target+200 )&(pos>target-200 )) target=200;

//if((distance<20) & (distance>10)) angle\_ref=pos\_zero;

target=angle\_ref;

// if ((pos<target+10)&(pos>target-10 )) target=-48;

PID(target);

pwr=fabs(u);

//if(pwr>150) pwr=150;

if (u>5) dir=-1;

else if (u<-5){

dir=1;

}

else dir=0;

setMotor(dir,pwr);

pre\_e\_x=e\_x;

Serial.print(x0);

Serial.print("|");

Serial.print(distance);

Serial.print("|");

Serial.print(angle\_ref);

Serial.print("|");

Serial.print(pos);

Serial.print("|");

Serial.print(e\_x);

Serial.print("|");

Serial.print(edot\_x);

Serial.print("|");

Serial.print(pos\_zero);

Serial.print("\n");

}

**IV. Kết quả thực nghiệm:**

Thể hiện trong video.