PROYECTO DE CONTROL, POSICÓN RESPECTO A UN OBSTACULO

Ismael Vega Rojas 2162931, Wilson Danilo Malaver Fonseca 2162496

Resumen – A lo largo de este documento se ira mostrando el paso a paso para la caracterización de una planta, diseño del controlador y sus respectivas pruebas de desempeño respecto a los parámetros esperados, La planta consta de dos motores que por medio del microcontrolador Arduino nano y un puente-H L298, se buscara que se mantenga a una distancia determinada de un obstáculo.

I. INTRODUCCIÓN

Una de las grandes inconvenientes de la movilidad en todo el mundo es la gran cantidad de choques que se presentan entre coches o directamente consta obstáculos fijos, por medio de los avances tecnológicos se ha permitido implementar vehículos autónomos en la movilidad de países desarrollados, sin embargo cada día se presenta un problema por resolver para optimizar el funcionamiento de estos así como lograr cada día que el vehículo sea en su totalidad autónomo si dejar de lado la seguridad para sus ocupantes y para el alrededor del mismo, por eso se ha propuesto el prototipo de un vehículo el cual se debe mantener a una distancia mínima de un obstáculo, para efectos didácticos se propone mantenerse a una distancia especifica del obstáculo sin importar si este cambia de posición repentinamente, buscando obtener una respuesta ante todo tipo de variaciones en la posición de la referencia. Para este diseño se usará un prototipo el cual solo se desplazará en dos direcciones, es decir se puede acercar o alejar de la referencia de forma lineal. Después del éxito en la caracterización y control del prototipo, se abren las posibilidades de aumentar las aplicaciones implementando algunas otras funciones como dirección por ejemplo para poder seguir un objeto en todo el plano **X, Y.**

II. CONSTRUCCIÓN Y MODELADO E LA PLANTA

A. *MATERIALES*

En este caso se hace el diseño de un móvil, con dos ruedas de 4cm de diámetro (1) adaptadas a dos motores de las mismas características (2), conectados en paralelo, que sostienen una plataforma a la cual se le añaden los de más materiales como lo son un sensor ultrasonido HC-SR04 (3), un Arduino nano (4), un módulo LM298(5) y un bloque de dos baterías de 3.7 V (6) en serie, a continuación, se visualizan los materiales

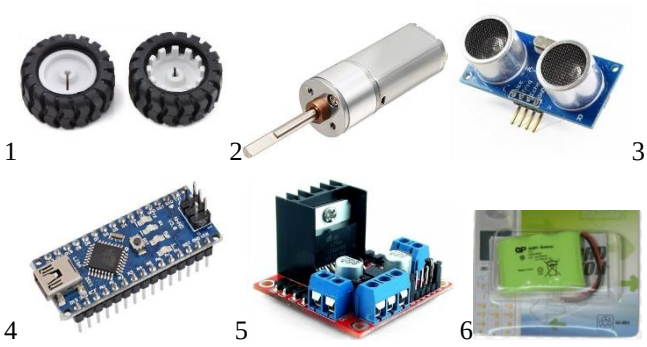


Ilustración 1: -Materiales usados.

B. *Características principales de los materiales*

1. Ruedas de goma de 42 mm de diámetro, 19 de ancho y de bajo peso.
2. Motorreductor Dc de 6 a 12 V, 15-30 RPM, Corriente nominal de 0.55 A
3. Sensor ultrasonido HC-SR04, 5V Dc, con un rango de medición entre 4 y 450 cm, consume alrededor de 15 mA, Precisión de ± 3m.
4. Arduino nano, opera a 5 V Dc, corriente de los pines máxima de 40mA, consumo de alrededor d 19mA, frecuencia del reloj de 16 MHz.
5. Control de motores LM298. Admite tensiones de entre 3 y 35V Dc, con niña intensidad de hasta 2 A, consta de un regulador interno para tener una salida de 5V para alimentar por ejemplo el Arduino si fuese necesario. Los pines de control trabajan a 5v y permiten hacer control PWM con valores entre 0 y 255.
6. 2 bloques de baterías recargables de 3 celdas, cada bloque de 3.7 V, conectadas en serie para obtener 7.4 V, suficientes para alimentar los motores por al menos 30 minutos en funcionamiento optimo.

C. *Ensamble y conexiones eléctricas*

En esta sección se describe principalmente las conexiones eléctricas usadas para interconectar los materiales eléctricos, así

como el ensamble de todos los componentes buscando un funcionamiento óptimo de la planta.

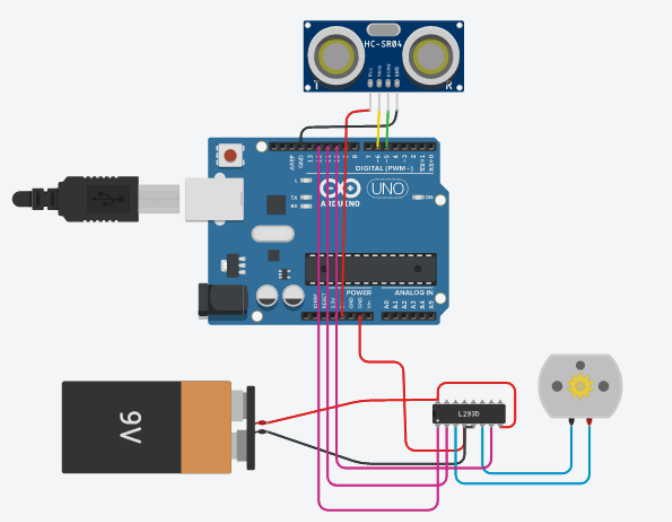


Ilustración 2: Conexiones eléctricas. (Los dos motores se conectan en paralelo, por ese solo se muestra uno de ellos.)

Ilustración 3: Montaje de la planta.

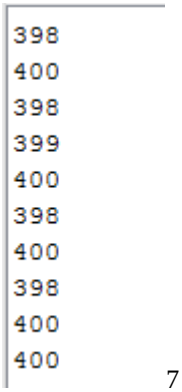
D. Caracterización de los elementos principales

1. Sensor de distancia

Después de tener la planta armada y conectada, se procede con las pruebas de funcionamiento para lo cual con ayuda de la programación en Arduino se realizan pruebas sencillas como mover adelante y atrás por un intervalo de tiempo corto, así como obtener lecturas del sensor y verificar la calidad de la medida observando la medida en el monitor Serial.

Medida [mm]	Mínimo [mm]	Máximo [mm]	Variación [±mm]
20	15	24	5
50	45	47	5
100	99	101	1
200	198	202	2
300	296	203	4
400	398	400	2
500	497	502	3
1005	1002	1007	2

Tabla 1. Medidas reflejadas por el sensor, tomadas del monitor serial



7

Ilustración 4. Distancia medida por el sensor HC-SR04

2. Driver LM298.

En al caso del controlador de motor y la conexión directa del motor, mediante pruebas se halló el mínimo PWM con que los motores realizan desplazamiento con las condiciones ya establecidas por la planta, es decir en una superficie nivelada, sin inclinación, sin rugosidades exageradas que pudiesen actuar como baches u obstáculos, teniendo en cuenta el tamaño de la rueda, además de eso con alimentación ya mencionada y el peso de la misma planta. Partiendo de esto se encontró que el mínimo PWM debe estar entre 65y 70 de 255, por lo cual se define un PWM de trabajo de 70 a 255.

Finalmente se debe tener en cuenta que el tiempo de trabajo está limitado por la batería, por lo que se considera un máximo de 30 minutos de funcionamiento antes de cargar de nuevo las baterías.

E. Caracterización de la planta

Para la caracterización de la planta se tiene en cuenta primero que todo el tiempo de muestreo mínimo al que pude responder nuestra planta, dado que si el periodo de muestreo es demasiado pequeño entonces no se alcanzaran a procesar los datos, visto de una forma más clara, se debe programa de manera sencilla la planta de tal manera que siga una referencia establecida, teniendo en cuenta la capacidad de respuesta en el tiempo establecido. Para esto se hace pruebas usando la función *millis* de Arduino, buscando que el periodo de muestreo no fuera en aumento a medida que se cumplen las diferentes condiciones del programa. Lara lo cual se observó que el máximo el periodo de respuesta aumentaba hasta casi 60ms, por lo cual se determinó T=60ms.

Después de obtener el periodo de muestreo se procede a definir una referencia y tomar los datos de la variación de la distancia a medida que pasa el tiempo, esto se hace con ayuda de la aplicación ***CoolTerm.exe.*** La cual nos permite visualizar y almacenar los datos del puerto serial en uso.

Para esto se coloca la planta a una distancia de aproximadamente 510 mm y se establece una referencia de 100 mm y se toman los datos de como la planta se acerca a la referencia y cuando llega a esta, debido a la ausencia de un controlador, permanece oscilando alrededor de la referencia.

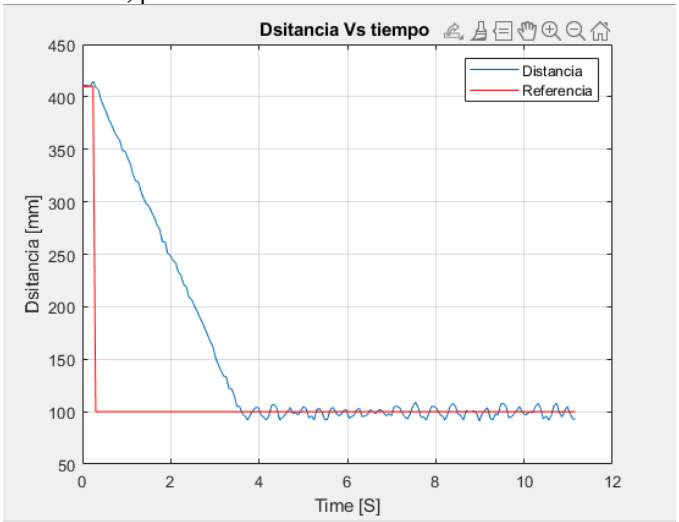


Ilustración 5: Grafica del comportamiento del sistema (Maltlab)

Después de importar lo datos en Matlab, se procede a hacer uso de la herramienta de "***System Identification toolbox"*** Para obtener una función de transferencia, la cual tenga una un seguimiento de los valores superior al 80%, antes de eso se les realiza un filtrado a los datos para mejorar el resultado

Se importan los datos teniendo en cuenta el periodo de muestres de 0.06 segundos, luego se hace estimaciones con diferentes parámetros, es decir para 1 polo 1 cero, 2 polos 2 ceros, 2 polos 1 cero, y 3 polos dos ceros, esto con el fin de encontrar la mejor estimación.

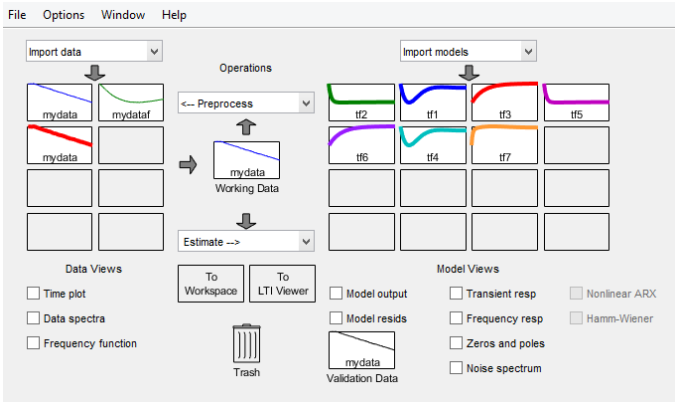


Ilustración 6. Importación de datos y estimación de la función de transferencia con diferentes parámetros.
Verde: tf2 2 polos, 1 cero= (morado claro tf5)
Azul: tf1 2 polos, 2 ceros = (celeste tf4)
Rojo: tf3 1 polo, 1 cero = (morado tf6)
Naranja tf7: 3 polos, 2 ceros

Después de estimar diferentes funciones de transferencia se visualizan en función del tiempo, para poder visualizar cuál de ellas tiene el mejor comportamiento, y cuál de ella queda totalmente descartada, también se observa cómo se comportan estas respecto a los datos reales tomados anteriormente, esto se puede ver seleccionando el recuadro **Model output**, este gráfico nos presenta tanto el seguimiento a los datos como el porcentaje de ajuste, siendo mejor el más alto. Estas gráficas se presentan a continuación, Ilustraciones 7, 8, 9

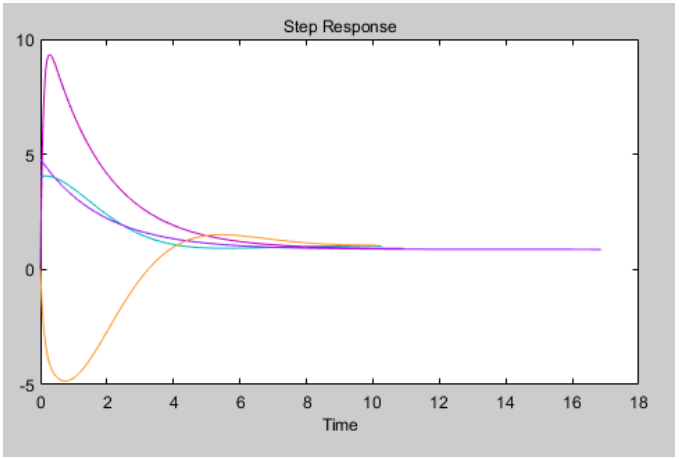


Ilustración 7. Análisis en respuesta temporal de las diferentes funciones estimadas.

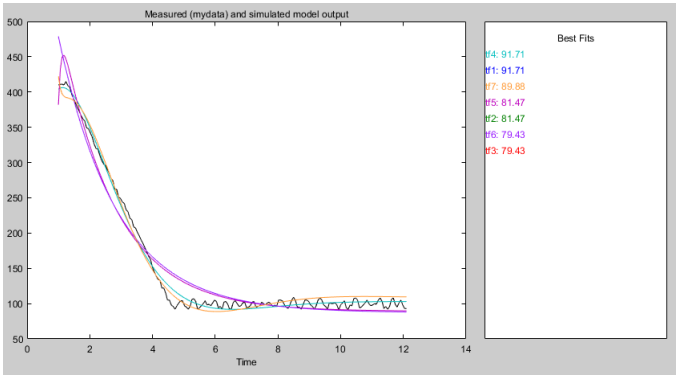


Ilustración 8. Análisis del modelo de salida, respecto a los datos reales, se observa que la mejor estimación es del 91.71%

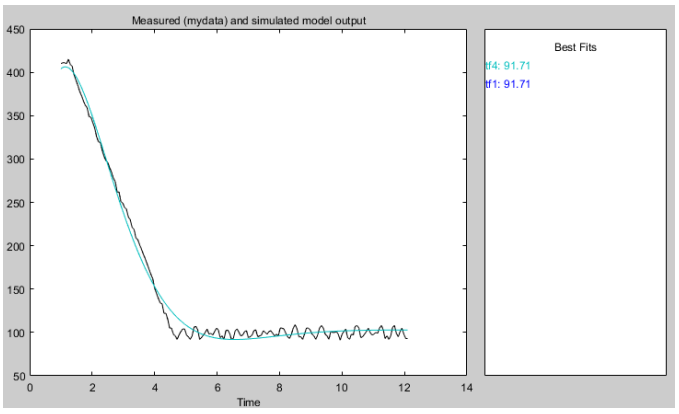


Ilustración 9. Se muestra el comportamiento de salidas de la función de transferencia elegida.

Finalmente se procede a mostrar la funcion de transferenecia estimada y seleccionada para continual con siguiente paso que sería el diseño del controlador.

```
  4.041 s^2 + 5.333 s + 0.7372
  ----------------------------
     s^2 + 1.239 s + 0.7197
Name: tf1
Continuous-time identified transfer function.

Parameterization:
   Number of poles: 2   Number of zeros: 2
   Number of free coefficients: 5
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using TFEST on time domain data "mydata".
Fit to estimation data: 91.71% (stability enforced)
FPE: 68.64, MSE: 63.66
```

*Ilustración 10. Función de transferencia estimada por "**System Identification Toolbox**"*

Segunda forma: Finalmente se usó esta metodología para el control de la planta

Primero que todo se linealizan los datos, teniendo en cuenta dos aspectos importantes.

- La salida, PWM debe tener un rango de excursión adecuado, en este caso el Arduino (microcontrolador) maneja 8 bits para las salidas análogas, por lo tanto, el máximo valor del PWM es de 255, el mínimo se calcula como el pwn desde el cual los motores realizan movimiento, para esto se considera 70.
- La entrada en este caso se considera en un rango que va desde la referencia para la que se tomaron los datos con el control on-off. Que en este caso fue desde 410mm hasta 100 mm que fue el punto de referencia

En los dos casos anteriores, para linealizar se debe tener en cuenta que el punto de partida será de 0. Por lo tanto, la salida tendrá un rango entre 0 y 185, mientras que la entrada será desde -310 hasta 0.

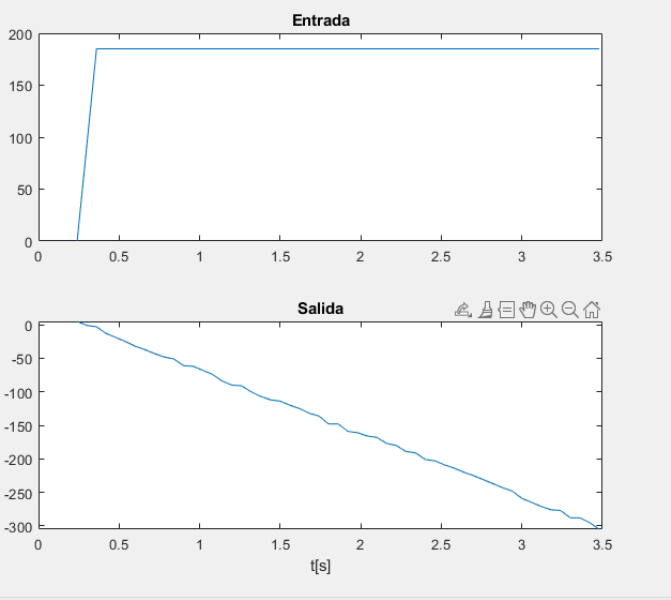


Ilustración 11: Entrada y salida linealizadas.

Después de linealizar, se procede a intentar buscar la funciona adecuada con el método "***process model***", ya que este nos permite ingresar un integrador desde el inicio, pero al tener intentos fallidos de aproximar una función a los datos reales, se procese a suponer una función de transferencia, sabiendo cómo se comporta en modelo y que se trata de un modelo similar al de posición motor estudiado anteriormente.

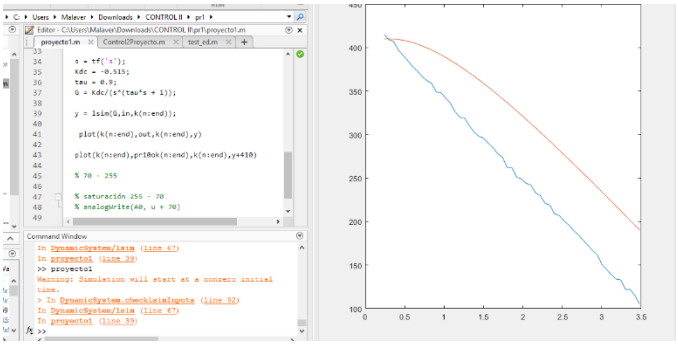


Ilustración 12: buscando los valores adecuados para G

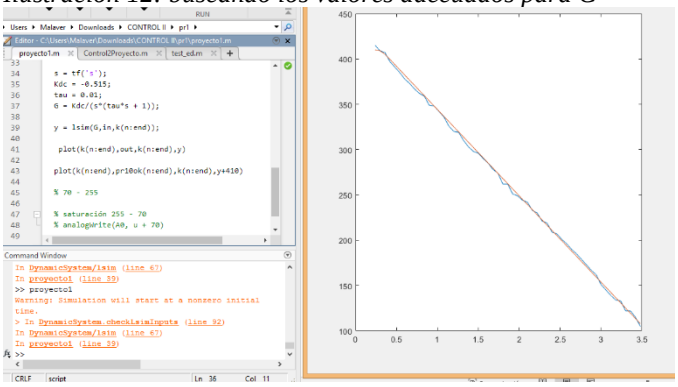


Ilustración 13: se visualiza que la función de transferencia hallada coincide con los datos reales

Encontrando una muy buena aproximación con los siguientres valores

```
s = tf('s');
Kdc = -0.515;
tau = 0.01;
G = Kdc/(s*(tau*s + 1));

y = lsim(G,in,k(n:end));

 plot(k(n:end),out,k(n:end),y)

plot(k(n:end),pr10ok(n:end),k(n:end),y+410)
```

Ilustración 14: Código usado para determinar la función de transferencia de la planta.

Finalmente se presenta la función de transferencia hallada en función *de S.*

```
G =

      -0.515
  ------------
  0.01 s^2 + s

Continuous-time transfer function.
```

Ilustración 15: Función de transferencia continua de la planta

A continuación, y con la ayuda de Matlab se discretiza la función para luego poder diseñar el controlador.

```
GGho =

   -0.02576 z - 0.005061
  -----------------------
  z^2 - 1.002 z + 0.002479

Sample time: 0.06 seconds
Discrete-time transfer function.
```

Ilustración 16: función de transferencia discretizada.

III. DISEÑO DEL CONTROLADOR

Se continua con el diseño de controlador esperando obtener un tiempo de establecimeinto de 2.4 segundos. Para eso se usa la herramneinta sisotool importando la funcion GGho, ya que esat coirespondea ala plnata que sera nuetro objeto a acontrolar.
Para finaemtne obtener lña función de transferencia del controlador y poder impplementarla en el microcontroladore de la planta.

A continuacion se presenta el controlador discreto diseñado.

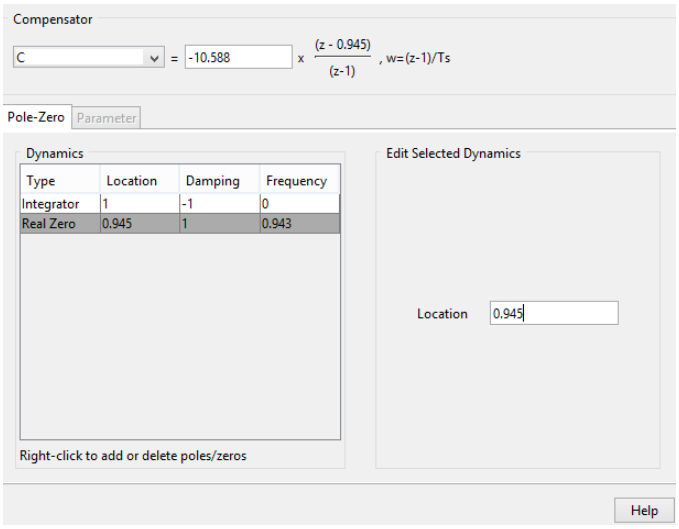


Ilustración 17: Controlador discreto.
Se muestra la respuesta al escalón donde se observa que se cumple con el tiempo de establecimiento.

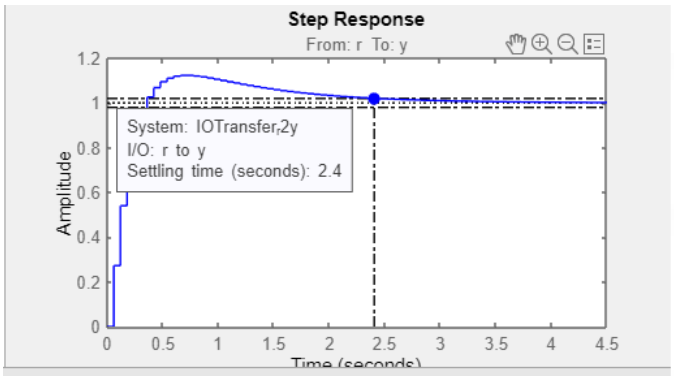


Ilustración 18: Respuesta al escalón, teniendo en cuneta el tiempo de establecimiento.
Despues de obtener la función de transferencia del controlador, se extrae la ecuacion en diferencias para poder implementar dicho controlador en el microcontrolador.

$$u[n] = u[n-1] - 10.588 * e[n] + 0.542 * e[n-1] \quad (1)$$

Donde:

$u[n]$: Valor de la salida, PWM
$u[n-1]$: $Valor\ de\ la\ muestra\ anterior$
$e[n]$: $error$
$e[n-1]$: $Valor\ anterior\ del\ error$.

Luego se implementa la ecuacion en diferencias (1) en arduino.

```
//Inicializando variables.
  ref=100; % Referencia [mm]
  e=0;
  pwm=1;
  cm=100;

  e=(cm-ref); //Error

  pwm = abs(((a*pwm1 + b*e + c*e1)/6));
  pwm1=pwm;
  e1=e;
```

Ilustración 19: Ecuación en diferencias y variables.

Se apaga el PWM cuando este no es suficiente para mover los motorwes, para evitar gasto de energia incncesesario.

```
  if (pwm >=(250))
  pwm = 250;
  if(pwm<=15)
  pwm=0;
```

Ilustración 20: Acotamiento del PWM

Se definen condiciones ya que el controlador tiene la capacidad de controlar la inversión de giro de los motores, para esto la condición debe ser el error, ya que a la ecuación en diferencias se le calcula el valor absoluto.

```
if (e>=0){
      digitalWrite(En, HIGH);
      analogWrite(Rt, 0);
      analogWrite(Av, pwm);
      delay(50);
}

if (e <= 0){
      digitalWrite(En, HIGH);
      analogWrite(Rt, pwm);
      analogWrite(Av, 0);
      delay(50);
}


if (e==0);{
  digitalWrite(En, LOW);
    analogWrite(Av, 0);
    analogWrite(Rt, 0);
    }
```

Ilustración 21: Condiciones para inversión de giro de los motores.

Fiablemente se prueba el comportamiento de la planta desde el "**Monitor Serial**" de Arduino donde se visualiza un correcto desempeño del PWM respecto al valor de la salida, en este caso directamente del error en la salida.

```
Eror = -47  pwm= 250
Eror = -45  pwm= 250
Eror = -45  pwm= 250
Eror = -42  pwm= 250
Eror = -36  pwm= 250
Eror = -34  pwm= 250
Eror = -32  pwm= 250
Eror = -31  pwm= 250
Eror = -25  pwm= 250
Eror = -22  pwm= 250
Eror = -17  pwm= 210
Eror = 10  pwm= 47
Eror = 44  pwm= 206
Eror = 69  pwm= 250
Eror = 64  pwm= 195
Eror = 69  pwm= 250
Eror = 51  pwm= 126
Eror = 41  pwm= 140
Eror = 45  pwm= 157
Eror = 30  pwm= 68
Eror = 40  pwm= 169
Eror = 42  pwm= 127
Eror = 33  pwm= 100
Eror = 20  pwm= 47
Eror = 14  pwm= 45
Eror = 11  pwm= 32
Eror = 14  pwm= 55
Eror = 38  pwm= 170
Eror = 19  pwm= 0
Eror = 12  pwm= 56
Eror = 3  pwm= 0
Eror = 1  pwm= 0
```

Ilustración 22: Prueba de funcionamiento del PWM respecto al error.

Y finalmente se comprueba el funcionamiento correcto de la planta.

IV. CONCLUISONES

- La herramienta **System Identification Toolbox**, es de gran ayuda gracia a la cantidad de herramientas que tiene a nuestra disposición, no solo para obtener la función de transferencia adecuada, sino también para la visualización y comparación de estas.
- Para que la estimación sea adecuada se debe hacer una correcta selección y linealización de los datos.
- En algunos casos, si el comportamiento del modelo es conocido, puede resultar mas favorable, suponer la función de transferencia de la planta y modificar

algunos parámetros para sintonizar o hacer coincidir esta con los datos reales y de esta manera caracterizar la planta.

- **Sisotool ()** Permite diseñar controladores de manera rápida, ya que si se tiene claros los conceptos será fácil visualizar el comportamiento y ajustar la ubicación de los polos y ceros de manera rápida.
- Finalmente se pudo comprobar que al multiplicar entrada por una constante como por ejemplo 2 o 3, se puede aumentar la velocidad de repuesta de la planta sin afectar el comportamiento. Ya que se acerca más rápido a ala referencia sin perder su comportamiento ya controlado.
- El error a la entrada rampa o cambio rápido de referencia es notorio, ay que el sistema tiene un tiempo de establecimiento dado en segundos.

REFERENCES

[1] G. O. Young, "Synthetic structure of industrial plastics (Book style with paper title and editor)," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

[2] W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.