SISTEMAS DE CONTROL EN TIEMPO CONTINUO

PEDRO LUIS SOLARTE CORREA

CONTROL PID DE PLANTA DE POSICIÓN

A:

ING. HERMES FABIAN VARGAS ROSERO

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
INGENIERÍA EN AUTOMÁTICA INDUSTRIAL
POPAYÁN, CAUCA
2016

INTRODUCCIÓN

Un controlador PID (Proporcional Integrativo Derivativo) es un mecanismo de control sobre una realimentación de lazo cerrado, que es usado ampliamente en la industria para realización de control de sistemas. El PID es un sistema al que le ingresa un error calculado a partir de la salida deseada, menos la salida obtenida y su salida es utilizada como entrada en el sistema que se quiere controlar. El controlador intenta minimizar el error ajustando la entrada del sistema.

OBJETIVOS DE LA PRACTICA

Realizar una planta tipo balancín, en la cual podamos aplicar el control PID a la posición de giro del motor, el cual se verá reflejado en la posición del balancín de acuerdo al *set point* al cual se desea tener ubicado el balancín.

CONCEPTOS A TENER EN CUENTA

- **Control PID**

  El controlador PID viene determinado por tres parámetros, los cuales son: el proporcional, el integral y el derivativo. Dependiendo de la modalidad del controlador alguno de estos valores puede ser 0. Por ejemplo, un controlador proporcional tendrá el integral y el derivativo igualados a 0 y un controlador PI solo el derivativo será 0.

  Cada uno de estos parámetros influye en mayor medida sobre alguna característica de la salida (tiempo de establecimiento, sobreoscilación, entre otras) pero también influye sobre las demás, por lo que por mucho que ajustemos no encontraríamos un PID que redujera el tiempo de establecimiento a 0, la sobreoscilación a 0, o que el error sea 0. sino que se

trata más de ajustarlo a un término medio cumpliendo las especificaciones requeridas.

- **Matlab**

  La plataforma de MATLAB está optimizada para resolver problemas de ingeniería y científicos. El lenguaje de MATLAB, basado en matrices, es la forma más natural del mundo para expresar las matemáticas computacionales. Los gráficos integrados facilitan la visualización de los datos y la obtención de información a partir de ellos. Una vasta librería de toolboxes preinstaladas le permiten empezar a trabajar inmediatamente con algoritmos esenciales para su dominio. El entorno de escritorio invita a experimentar, explorar y descubrir. Todas estas herramientas y prestaciones de MATLAB están probadas y diseñadas rigurosamente para trabajar juntas.

- **Simulink**

  Es un paquete de software para modelar, simular y analizar sistemas dinámicos, soporta sistemas lineales y no lineales, modelados en tiempo continuo, muestreados o en hibrido entre los dos. Los sistemas pueden ser también multifrecuencia, es decir, tienen diferentes partes que se muestrean o se actualizan con diferentes velocidades.

- **Arduino**

  Arduino es una plataforma de prototipos electrónica de código abierto basada en hardware y software que son flexibles y fáciles de usar. Arduino puede sentir el entorno mediante la recepción de entradas desde una variedad de sensores y puede afectar a su alrededor mediante el control de luces, motores y otros dispositivos.

- **Arduino Mega 2560**

  Arduino es una plataforma física computacional basada en una sencilla placa con entradas y salidas, analógicas y digitales, y en un entorno de desarrollo que implementa el lenguaje Processing/Wiring. El Arduino Mega está basado en el microcontrolador ATMega2560. Tiene 54 pines de entradas/salidas digitales (14 de las cuales pueden ser utilizadas como salidas PWM), 16 entradas análogas, 4 UARTs (puertos serial por hardware), cristal oscilador de 16 Mhz, conexión USB, jack de alimentación, conector ICSP y botón de reset.

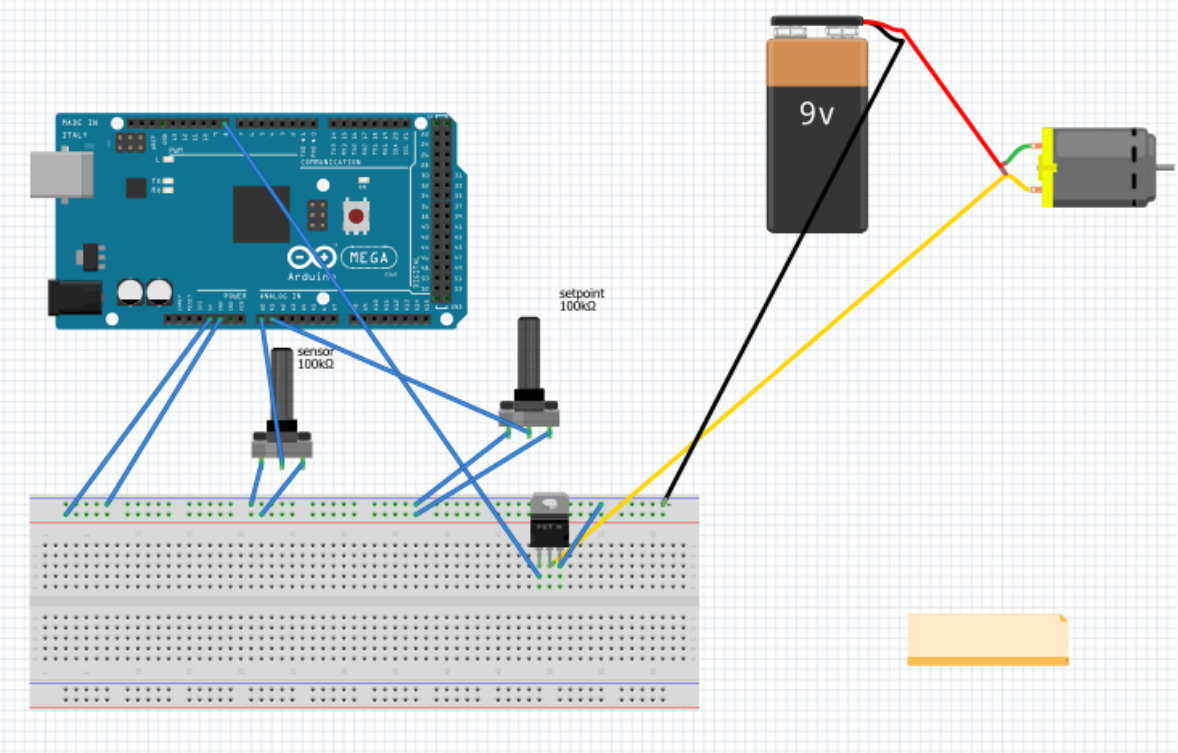
PRÁCTICA

Elementos utilizados

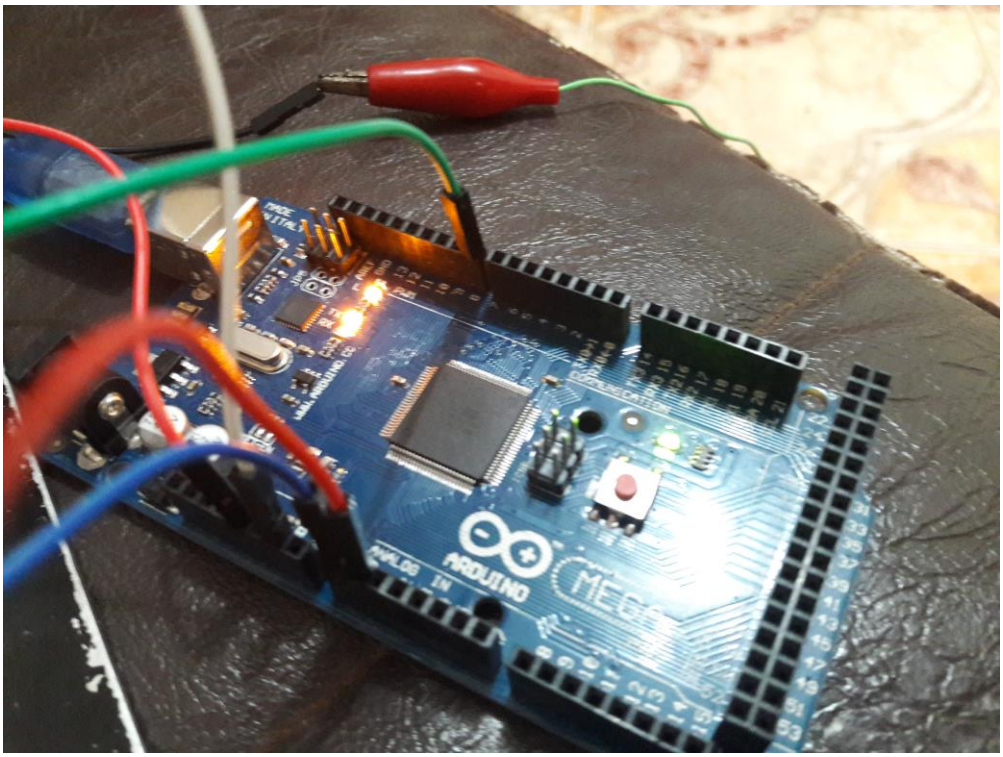
- Planta realizada en madera
- Motor DC 9V
- Hélice
- Potenciómetros de 20KΩ y 100KΩ
- Jumpers
- Protoboard
- TIP31C
- Caimanes
- Fuente 9V
- Matlab 2014ª y librerías de Simulink para Arduino
- Software Arduino
- Computador

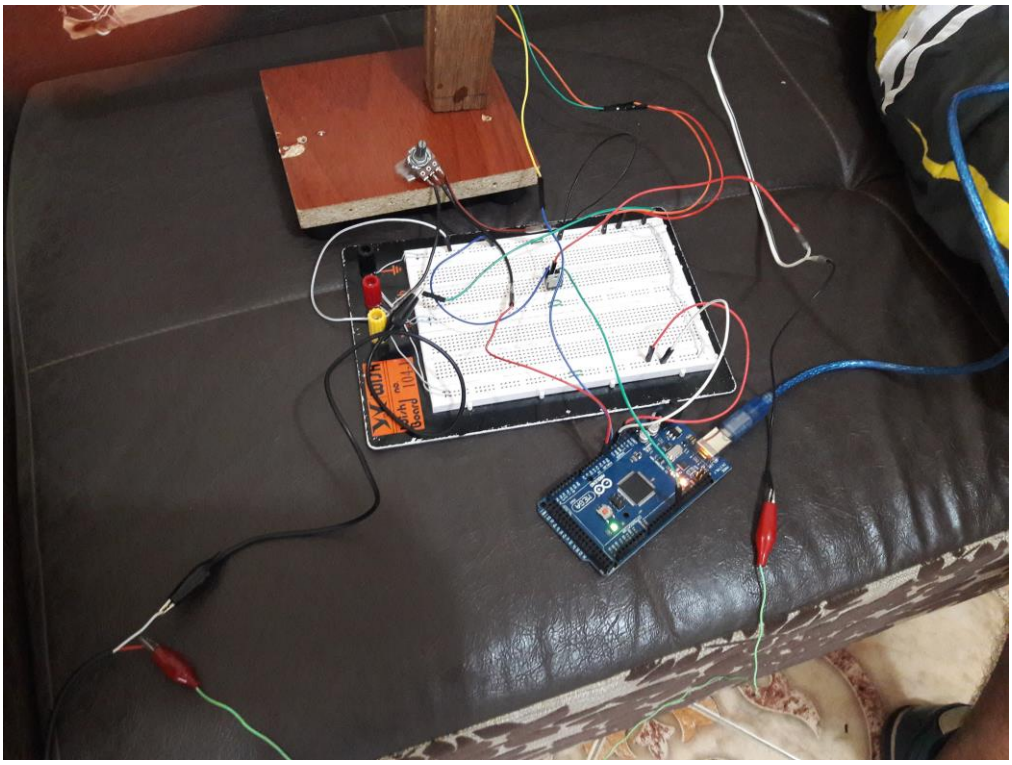
PROCEDIMIENTO

Se realizó el montaje de todos componentes que se mencionaron anteriormente en la protoboard, basados en el siguiente diagrama.



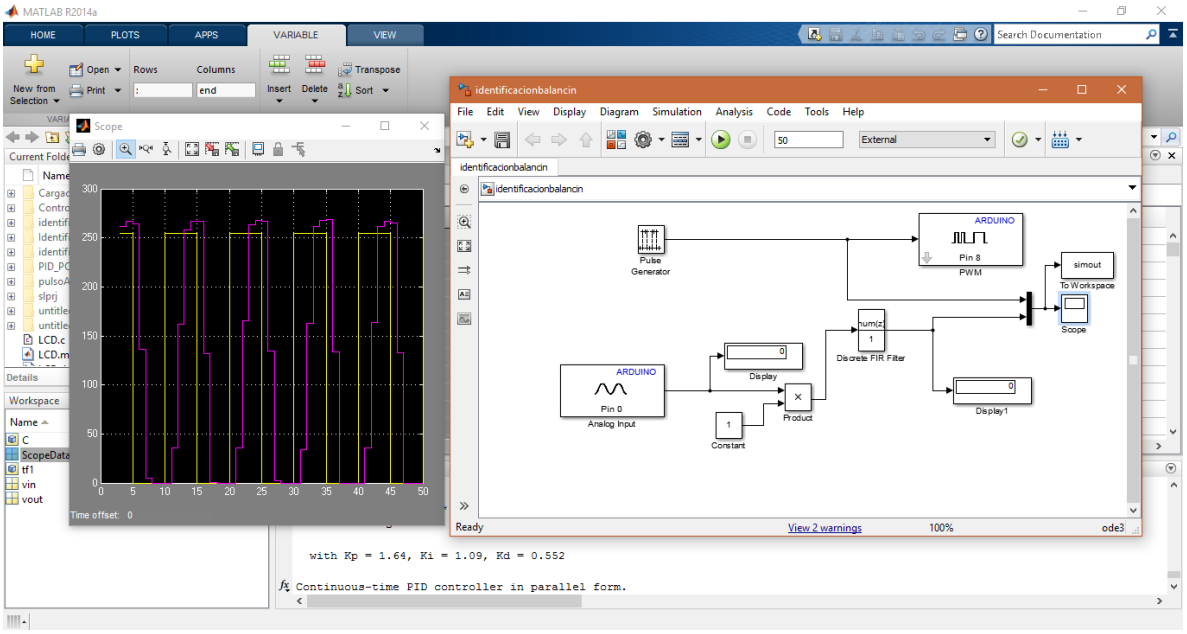
Luego, se procede a hacer el montaje de todos los componentes según la imagen anterior. La planta queda finalizada en cuanto a su montaje, como se muestra en la siguiente imagen.

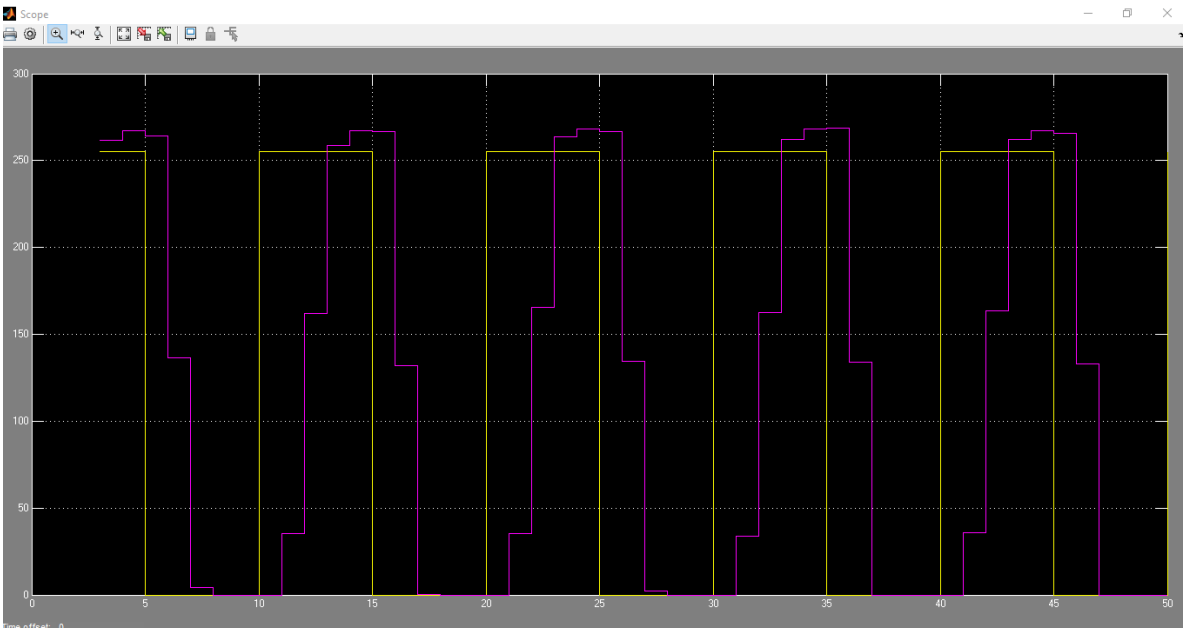

ARDUINO
MEGA



IDENTIFICACIÓN DE LA PLANTA

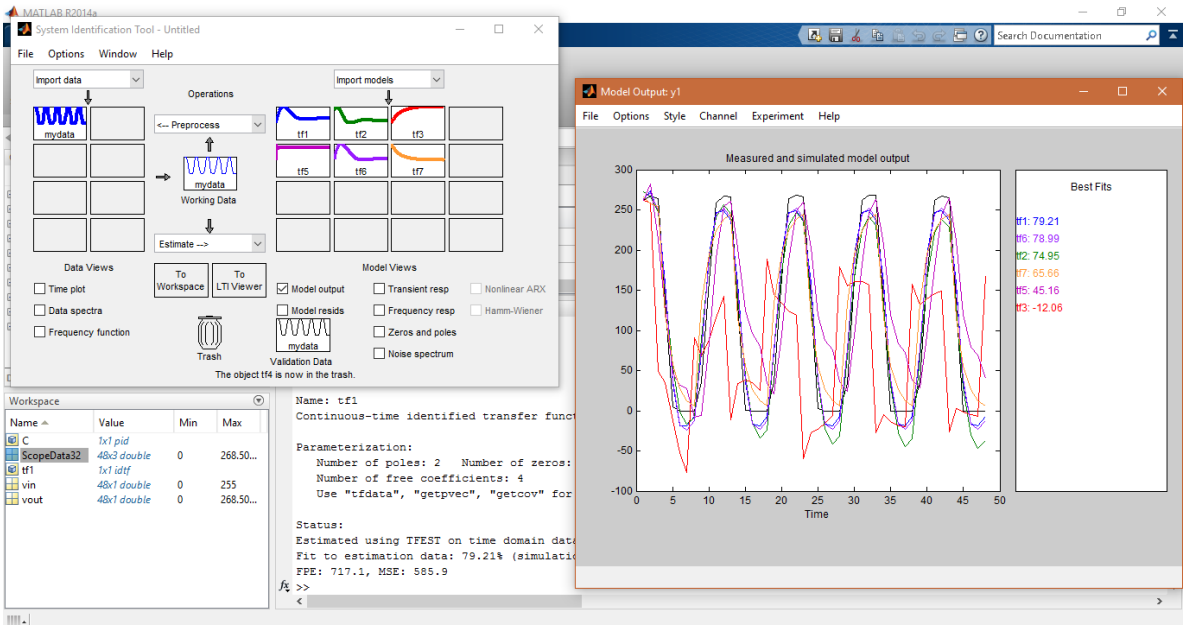
Primeramente, se generó un impulso al actuador, es decir que el motor registra una serie de voltajes obtenidos mediante los pulsos ingresados por MatLab con el modulo para Arduino.



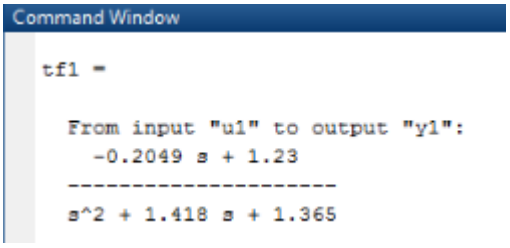


Luego se procede a crear dos vectores (uno para cada señal), con el rango de tiempo y los datos en ese rango para proporcionarlos a Matlab y crear una gráfica de I vs O. Este procedimiento se hace con la función **ident,** la cual estima una función de transferencia.

El paso siguiente es empezar a variar los polos y ceros de la función obtenida para obtener una exactitud como mínimo de 75%.



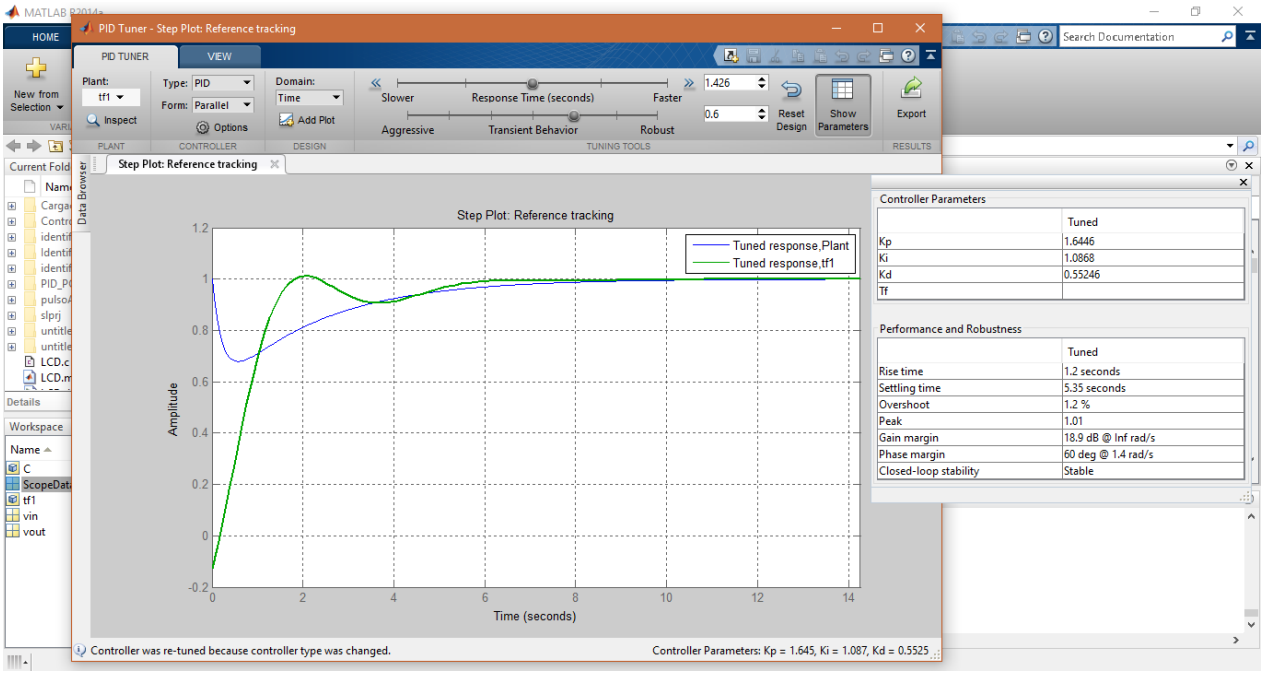
Como se puede apreciar en la anterior imagen, en nuestro caso obtener la función de transferencia que cumpliera con el anterior requerimiento no fue posible, ya que el sistema presentaba muchas oscilaciones, pero después de hacer muchas pruebas se obtuvo una función de porcentaje cercano siendo de 79.21%. En la siguiente ilustración se puede observar la función de transferencia estimada.



ESTIMACIÓN Y SINTONIZACIÓN DEL CONTROLADOR

Para este procedimiento se hace uso de la herramienta **pidtool,** para hacer la sintonización de la planta, tomando la función de transferencia estimada (en nuestro caso de 1er orden, que proporcionaba casi el 79.21% de exactitud) para modificar la respuesta del sistema siendo más rápido o más robusto, lo cual modifica las constantes del controlador que estima el **pidtool.**

Una vez obtenido un modelo que cumpla unas condiciones de rapidez, pero estable, se exporta la función de transferencia del controlador, con las variables del mismo. Como se puede observar en la siguiente imagen, la sintonización se hace para que el sistema cumpla unos requerimientos de rapidez y estabilidad, lo que se hace es hacer una compensación para que el sistema sea eficiente para estas dos características.



Una vez obtenidas las constantes del controlador y la función de transferencia de la planta, se procede a hacer las pruebas tanto en el código de Simulink como en la interfaz de Arduino.

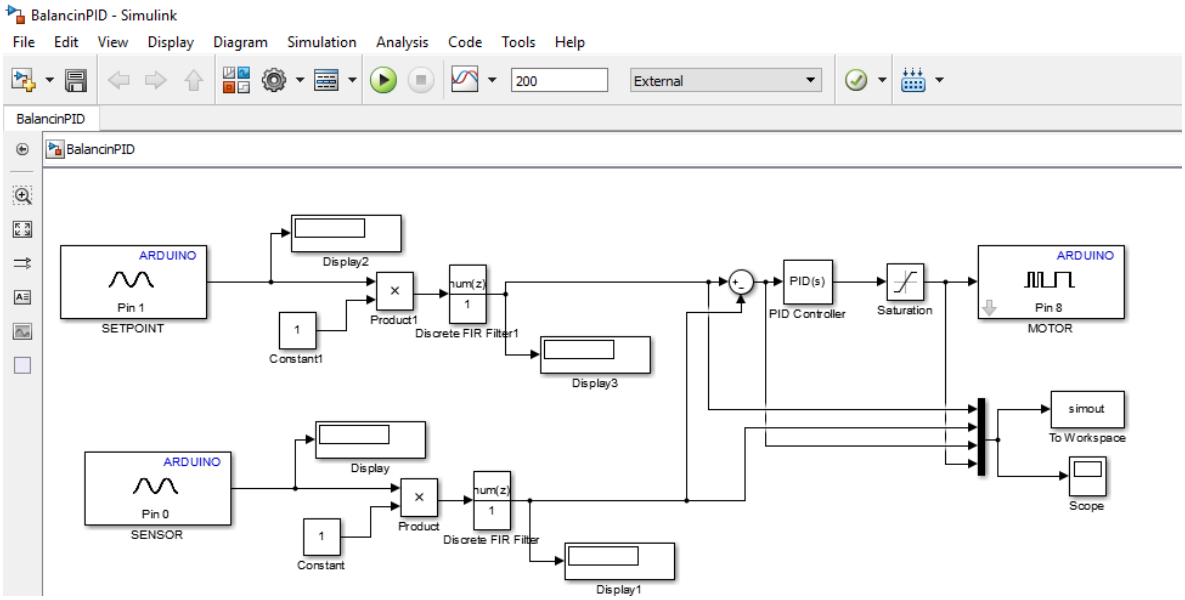
```
C =

                1
Kp + Ki * --- + Kd * s
                s

with Kp = 1.64, Ki = 1.09, Kd = 0.552
```

PRUEBAS DE FUNCIONAMIENTO EN SIMULINK Y ARDUINO

En el caso de Simulink se proporcionan las constantes obtenidas a la función/bloque PID y se procede a activar el sistema. En nuestro caso la planta se comporta de manera muy eficiente y para comprobar que sea el mismo comportamiento de manera teórica, se simula la planta con la función de transferencia estimada y las constantes del controlador para observar gráficamente la respuesta del sistema.



CODIGO EN ARDUINO

```
#include <PID_v1.h>
#define PIN_SENSOR 0
#define PIN_SETPOINT 1
#define PIN_MOTOR 8

double Setpoint, Sensor, Sensor2, Motor, Setpoint2;
double Kp=1.64, Ki=1.05, Kd=1;
PID BalancinPID(&Sensor, &Motor, &Setpoint, Kp, Ki, Kd, DIRECT);

void setup()
{
  Serial.begin(9600);
  Sensor2 = analogRead(PIN_SENSOR);
  Sensor = map(Sensor2,0,1023,0,270);
  Setpoint2 = analogRead(PIN_SETPOINT);
  Setpoint = map(Setpoint2,0,1023,0,73);

  BalancinPID.SetMode(AUTOMATIC);
}

void loop()
{
  Sensor2 = analogRead(PIN_SENSOR);
  Sensor = map(Sensor2,0,1023,0,270);
  Setpoint2 = analogRead(PIN_SETPOINT);
  Setpoint = map(Setpoint2,0,1023,0,73);
  BalancinPID.Compute();
  analogWrite(PIN_MOTOR, Motor);
  Serial.print(" SP: ");
  Serial.print(Setpoint);
  Serial.print("  Sensor: ");
  Serial.print(Sensor);
  Serial.print(" Motor: ");
  Serial.println(Motor);
}
```

CONCLUSIONES

- La realización de la práctica se hizo un poco difícil en cuanto a la identificación de la planta, ya que el potenciómetro usado como sensor no es suficiente para hacer la lectura de la posición del balancín respecto al tiempo debido a que es un potenciómetro logarítmico.

- Para la realización del control es mejor utilizar sensores más precisos para obtención de datos y así lograr que el control sea más robusto**.**

- Es necesario la utilización de un motor que genere más revoluciones, ya que cuando se utilizan valores de voltaje muy bajos, este sería capaz de girar más rápido y hacer que el sistema se estabilice de manera más eficaz.

- Lo logrado con la implementación de esta práctica nos despeja muchas dudas sobre la implementación de control PID y como saber manejar las diferentes variables que se presentan en un sistema.

BIBLIOGRAFIA

- https://control-pid.wikispaces.com/
- https://www.mathworks.com/products/matlab.html
- https://www.arduino.cc/