# PROGRAMACIÓN DE ROBOTS

*Programación de microbots*
*PRÁCTICA 3.- Seguimiento de líneas basado en PID*

El objetivo de esta práctica es implantar un algoritmo de seguimiento de líneas basado en un controlador PID. A lo largo del texto de la práctica se desarrolla el esquema básico que debe seguirse.

## 1.- Planteamiento del programa general

El seguidor de líneas que vamos a desarrollar (basado en http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html) va a seguir, en realidad, el borde de la línea negra, y no la línea en sí. De esta manera será más fácil detectar, cuando se abandona la línea, por qué lado nos hemos salido; si intentáramos ir por el centro de la línea, necesitaríamos dos lecturas del robot para averiguar por qué lateral hemos abandonado la línea.

El algoritmo general que debe seguir el programa es el siguiente:

- Definición de constantes (entre ellas, Kp, Ki, Kd)
- main
  - Declaración de variables
  - Calibración de la luz del sensor, ya que dependiendo del nivel de luz que haya en la habitación, las lecturas del sensor correspondientes al negro y al blanco son diferentes
  - Algoritmo de control en un bucle
    - Leer el sensor
    - Ajuste de la lectura a los niveles de luz del sensor
    - Actualización de la actuación del controlador
    - Ajuste de la actuación con Wind-Up
    - Ajuste de la potencia de los motores con Saturación

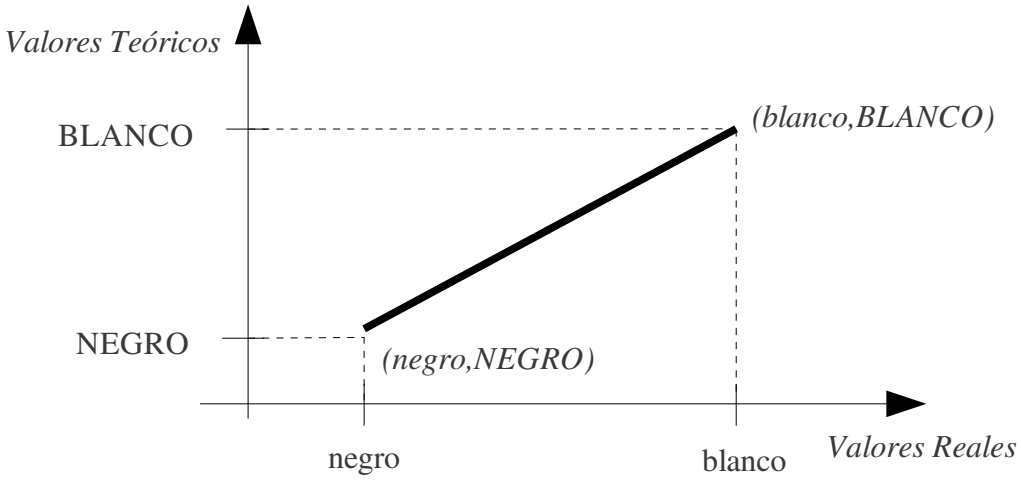
## 2.- Calibración de la luz del sensor

Lo primero que haremos será definir qué niveles de luz serán nuestros valores de blanco y negro; para ello definimos dos constantes BLANCO y NEGRO con valores 70 y 35, respectivamente (puede comprobarse con el menú Try-Me del LEGO, que valores bajos devueltos por el sensor corresponden a colores oscuros, y valores altos, a colores claros).

Sin embargo, las condiciones de luz del entorno donde realicemos el seguimiento de líneas no serán las mismas siempre (por ejemplo, las lecturas del sensor variarán de un día soleado a otro nuboso), por lo que los valores reales de luz para el blanco y el negro que obtengamos pueden diferir mucho de nuestras constantes, con lo que el seguimiento de la línea podría ser totalmente incorrecto. Para evitar este problema, vamos a normalizar y escalar los valores reales de luz que nos devuelva el sensor de la siguiente forma, para que nuestro algoritmo pueda funcionar independientemente de las condiciones lumínicas. Por ello, el primer paso de nuestro programa será leer el valor real *blanco* en nuestro entorno (situando el robot en una zona blanca y leyendo el valor devuelto por el sensor), y el

valor real *negro* (situando el robot en una zona negra, y leyendo el valor devuelto por el sensor)

A continuación, definiremos el siguiente sistema, donde los valores reales (blanco, negro) devueltos por el sensor estarán en el eje *x*, mientras que los valores teóricos previamente definidos (BLANCO y NEGRO) se situarán en el eje *y*.




El valor de luz definitivo que usaremos en nuestro programa se situará en la línea recta definida entre los puntos (negro,NEGRO) y (blanco,BLANCO), definida según la expresión de la recta que pasa por dos puntos, donde y es el valor de luz escalado y normalizado que buscamos y que usaremos en el resto del programa, y x, el valor de luz real leído del sensor:

$$\frac{(x-x_1)}{(x_2-x_1)}=\frac{(y-y_1)}{(y_2-y_1)}$$

$$\frac{(x-negro)}{(blanco-negro)}=\frac{(y-NEGRO)}{(BLANCO-NEGRO)}$$

Por tanto, los valores de luz que usemos en nuestro controlador serán las y de la recta anterior, obtenidas a partir de los valores de luz x devueltos por el sensor, y escalados y normalizados según la expresión anterior.

Por tanto, en nuestro programa tendremos que hacer lo siguiente:

1.- Definir las constantes BLANCO y NEGRO.
2.- Tomar los valores reales de luz blanco y negro en nuestro entorno.
3.- Escalar y normalizar cada lectura que hagamos del lector según la expresión de la recta anterior.

```
lectura_recta = (lectura*(BLANCO-NEGRO)/(blanco-negro))+((-negro*(BLANCO-NEGRO))
                    +(NEGRO*(blanco-negro)))/(blanco-negro);
```

4.- Determinar el error como la diferencia entre la lectura escalada y normalizada, y la media de los valores BLANCO y NEGRO.

## 3.- Controlador PID

La actuación del controlador PID se basa en la siguiente expresión:

$$G_c(s) = K_P + K_D s + K_I \frac{1}{s}$$

de manera que será necesario multiplicar el error, la derivada del error, y la integral del error por las constantes correspondientes. Como el controlador P actúa directamente sobre el error, será necesario definir variables que almacenen la integral del error (como suma acumulada del error) y la derivada del error (como diferencia entre el error actual y el error de la pasada anterior del bucle).

## 4.- Actuación con Wind-Up

La acción de control integral puede generar el llamado efecto windup, explicado a continuación. La acción de control actúa, y lleva al actuador a la saturación (es decir, en nuestro caso, determina que la potencia de los motores sea 100); sin embargo, si el error no desaparece, se seguirá acumulando, con lo cual la acción de control integral sigue aumentando, a pesar de que el actuador no puede hacer nada más para poder eliminar el error. A partir de aquí pueden pasar dos cosas: o que la acción de control siga creciendo indefinidamente, con lo cual hemos perdido el control del bucle, o que el error comience a decrecer, pero cueste mucho trabajo eliminar el efecto de la acción integral ya que se había acumulado mucho error hasta el momento.

Para evitar esta situación existen diferentes métodos, aunque nosotros vamos a implementar uno propio: consiste en determinar un valor umbral WINDUP, de forma que si la acción de control total (la suma de las acciones de control proporcional, integral y derivativa) lo supera, se elimina de dicha acción de control la acción de control integral, y al error integral se le resta el último valor del error. De esta manera, se evita que la acción de control integral provoque el windup.

## 5.- Ajuste de los motores con saturación

Una vez determinada la acción de control, la potencia que se suministra a cada motor es la suma de una potencia base (no muy alta) más (o menos, dependiendo del motor) la nueva acción de control. Para evitar movimientos muy bruscos del robot, conviene restringir el valor de potencia no al máximo que los motores del robot pueden dar, sino a un cierto valor de SATURACIÓN máximo.