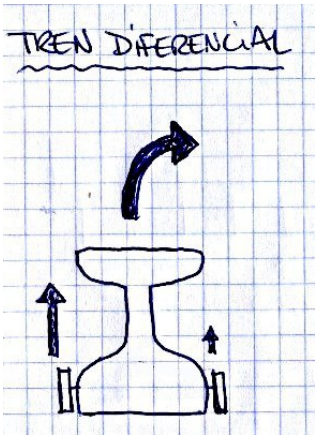
¿Cómo funciona un control proporcional derivativo (PD)?

Adaptación del artículo: http://iesseveroochoa.edu.gva.es/severobot/2011/01/29/como-funciona-un-controlador-pd/ para el Taller de Iniciación a la Robótica 2012 organizado por el CRM-UAM.

El tren de tracción diferencial

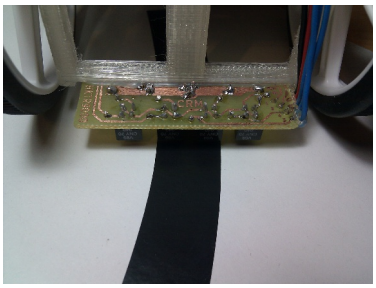


La forma más simple de mover un robot es usando dos motores independientes, uno a cada lado. **Variando la velocidad de uno y otro conseguiremos que el robot gire, mientras que si los dos motores giran a la misma velocidad, el robot irá recto.**

*A esto se le llama **configuración diferencial** y es el mecanismo utilizado, por ejemplo, por las orugas de un tanque o una excavadora. El tercer punto de apoyo se consigue con una rueda loca, por ejemplo en robótica se usa una bola que gira libremente dentro de un casquillo de plástico.*

Esta configuración es la más utilizada, ya que en todo momento los 3 puntos están apoyados en el suelo aunque haya irregularidades, confiriéndole al robot mayor estabilidad y agarre.

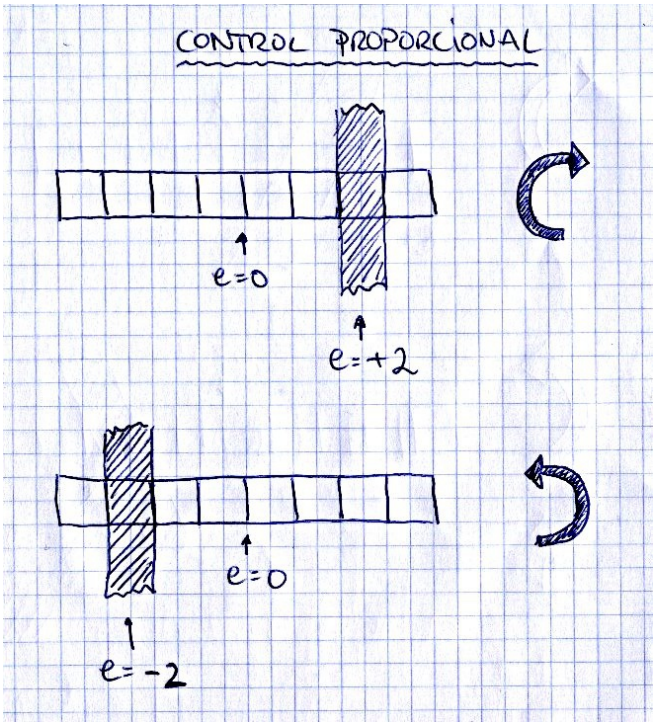
Control proporcional



Nuestro robot tiene una tira de 4 sensores separados más o menos 0.5 cm, así que podemos detectar la posición de la línea con una precisión de medio centímetro.

Si la línea se mueve, tendremos un "error", que puede ser positivo o negativo según si el desplazamiento es a la izquierda o a la derecha.

Si la línea está situada exactamente entre los dos sensores centrales, el error es 0 (lo deseado para un seguidor de línea).



Pero si el error es positivo, significa que la línea se está moviendo a la derecha, por tanto nosotros también nos moveremos a la derecha para corregir el error. En caso de error negativo es lo mismo, sólo que corregiremos girando a la izquierda.

El error nos indica el radio de la curvatura: cuánto más alto sea, significa que la curva es más cerrada.

Para evitar salirnos en las curvas, lo lógico es girar más o menos según sea la curva más cerrada o más abierta, multiplicando el error por una constante.

Esta será la **constante proporcional Kp:**

$$\Delta v = K_p \cdot e$$

A la velocidad que lleva el robot en línea recta le llamamos **velocidad de crucero**. Cuando queremos girar, simplemente a un motor le sumamos la Δv anterior, y al otro se la restamos:

$$V_{IZDA} = V_{CRUCERO} + \Delta v$$
$$V_{DCHA} = V_{CRUCERO} - \Delta v$$

Cuando el robot va despacio, el control proporcional funciona muy bien y es suficiente. El problema a altas velocidades es que los motores llevan **inercia** y cuando les obligamos a cambiar de velocidad, este cambio no sucede inmediatamente, sino que tarda un tiempo.

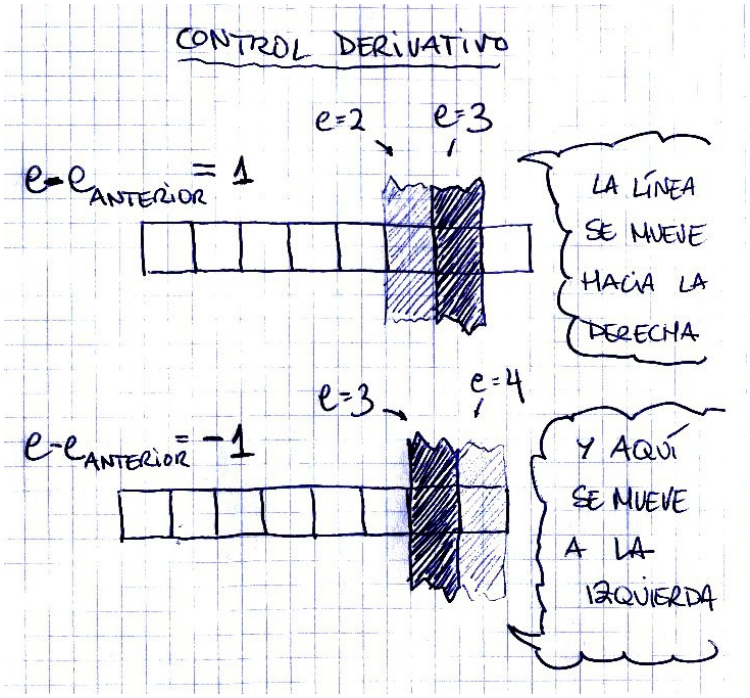
Si, por ejemplo, el robot se sale por la izquierda de la línea y corregimos girando a la derecha, probablemente el robot se pasará la línea y se saldrá por la derecha. Volveremos a corregir hacia la izquierda, pero la inercia le hará salirse otra vez. Veríamos que el robot va haciendo eses por encima de la línea, y si las oscilaciones son muy bruscas, hasta se saldría.

(Fuente: http://elm-chan.org/works/ltc/report.html)

Control derivativo

El control derivativo se llama así porque en vez de utilizar el error, utiliza la derivada del error respecto del tiempo, es decir la velocidad con la que varía el error.

Para calcular la velocidad con la que cambia el error, basta con restar el error anterior al actual y dividir por el tiempo transcurrido. Si hacemos esto a intervalos fijos, el tiempo transcurrido siempre es el mismo, se convierte en factor común y la división se puede omitir haciendo mucho más simples los cálculos.



En la figura hay dos dibujos. En ambos la línea **está sobre el tercer sensor**, con lo que **el error es 3**.

Como la línea está a la derecha, **nuestro controlador proporcional decidirá mover el robot a la derecha, pero ¡hay una diferencia importante!**

En el primer dibujo, la línea **se está alejando** del centro (error actual – error anterior = 1) y **deberíamos reforzar el giro**, ya que aparentemente nos vamos a salir pronto.

Por el contrario, en el segundo caso la diferencia de errores es -1, lo que significa que **ya nos estamos acercando al centro** y, por tanto, **hay que amortiguar un poco el giro para no pasarnos** de la línea.

En definitiva, **el control proporcional se encarga de decidir la dirección y velocidad del giro**, mientras que **el control derivativo tiene cierto carácter predictivo que le permite amortiguar o reforzar el giro para evitar oscilaciones**.

La suma de ambos nos da un **control proporcional-derivativo**:

$$\Delta v = K_p \cdot e + K_D \cdot (e - e_{ANTERIOR})$$

Aquí aparece una nueva constante: **la Kd o constante derivativa**, que es el factor que multiplica a la velocidad del error.

¿Cómo se calculan las constantes Kp y Kd?

Pues, lamentablemente, **a ojo**. *Un robot de estas características es muy difícil de modelar matemáticamente. Hay que tener en cuenta el par de los motores, la tensión de la batería, el agarre de las ruedas (que a su vez depende de la superficie), la distribución de pesos, longitud y un sinfín de parámetros más.*

Hay varios métodos empíricos para ajustar un controlador PD, pero a nosotros nos ha funcionado éste, que es bastante fácil:

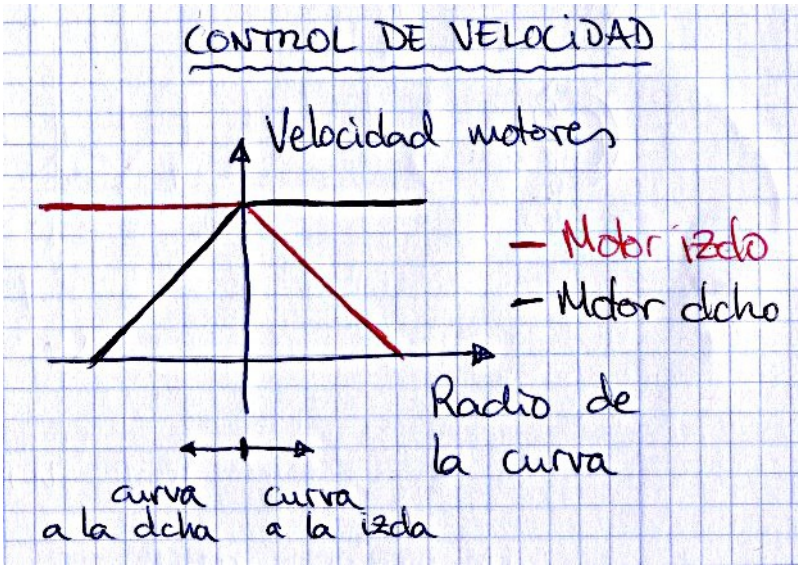
- Ajustar la Kp de manera que el término proporcional cubra todo el rango de velocidades desde 0 hasta la velocidad crucero. **Es decir, que si el robot se sale por un lado, la rueda interior debería pararse mientras la otra gira a Vcrucero.**
- Al principio probar con la Kd = 0. El robot debería oscilar y salirse.
- **Ir subiendo la Kd hasta que deja de oscilar.**

Control de la velocidad

La Δv anterior se utiliza para variar las velocidades de los motores según las fórmulas que hemos visto antes:

$$V_{IZDA} = V_{CRUCERO} + \Delta v$$
$$V_{DCHA} = V_{CRUCERO} - \Delta v$$

Pero esto no es del todo correcto y **hay que poner unos límites**: <u>la velocidad nunca puede ser negativa, y tampoco queremos superar en ningún momento la velocidad de crucero (ya que en las curvas el robot podría acelerar demasiado y salirse).</u>



Como se ve, la velocidad de cada motor puede estar entre 0 y la velocidad de crucero, pero nunca podrá sobrepasar esta última.

Algunas notas sobre el control PD

Todos los puentes en H tienen dos modos de "parar" un motor: una es frenándolo y otra dejándolo girar libremente. La diferencia es que en el modo "frenado", internamente se cortocircuitan las dos patas del motor. Haciendo esto se genera una fuerza electromotriz que se opone al giro del motor, frenándolo. Cuando usamos PWM para controlar el motor conviene usar esta configuración porque en teoría tendremos una respuesta más precisa de los motores. En la práctica los motores de robótica son tan pequeños que la f.e.m. autoinducida apenas frena, pero algo es algo.

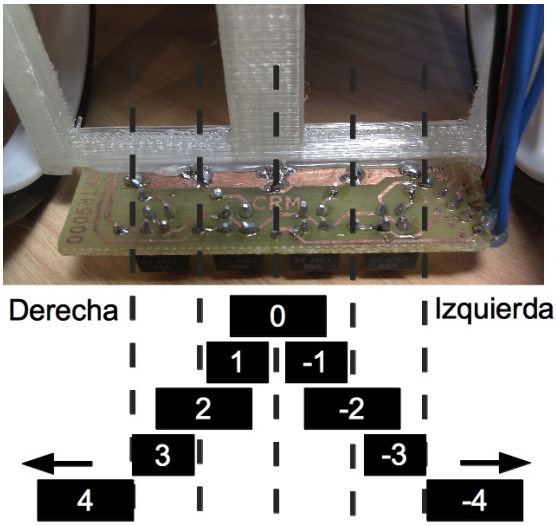
En robótica de competición se necesitan velocidades muy elevadas, y para conseguir un mejor rendimiento del control PD incluso se fuerza a retroceder a los motores, logrando así frenarlos de forma casi instantánea.

Para este taller no usaremos ni el modo de parada rápida ni el retroceso de los motores, ya que suponen un esfuerzo adicional para las baterías, lo que resulta en picos de tensión que acaban por reiniciar la Arduino.

Adaptado por C.G.S.

Ejercicios de control proporcional

1. **Implementar la función `int error_linea()`.** Debe devolver un valor comprendido entre -4 y 4 indicando la posición de la línea negra. Es decir:



   *Nota: Defina una variable global **`int ultima_posicion_linea`** para poder identificar por que lado escapa la línea de la zona de visibilidad de los sensores (si no se ve la línea y la última posición era -3, entonces la nueva posición será -4), y para poder devolver ese valor en caso de una lectura problemática (p.e: se activan a la vez los sensores 1 y 4).*

2. **Probar la función `int error_linea()`**, leyendo el valor retornado a través del puerto serie y asegurándose de que es correcto.

3. **Implementar la función `void cabeceo_motores(int velocidad, int desviacion)`.** Debe ajustar la velocidad de los motores según:

   `Velocidad_motor_izquierdo = velocidad_crucero + desviacion;`

   `Velocidad_motor_derecho = velocidad_crucero – desviacion;`

   Importante limitar: 0 ≤ velocidad_motor ≤ velocidad_crucero

   Nota: Debe definir la variable global `int velocidad_crucero`. Un buen valor inicial puede ser 40.

4. **Probar la función `void cabeceo_motores()`**, asegurándose de que los resultados son exactamente los indicados.

5. **Hacer que el robot siga una línea negra.** Usando las dos funciones creadas y una constante proporcional `int kp`.

   El valor de la constante puede ser `kp=velocidad_crucero/4;` para así conseguir que *"el término proporcional cubra todo el rango de velocidades desde 0 hasta la velocidad crucero: Es decir, que si el robot se sale por un lado, la rueda interior debería pararse mientras la otra gira a Vcrucero."*

Se debe experimentar a cambiar los valores de la constante y de la velocidad de crucero para observar cómo influyen en el comportamiento del robot.

Ejercicios de control proporcional derivativo

6. **Implementar la función `int velocidad_error_linea(int posicion_linea).`**
   Debe devolver un valor comprendido entre -8 y 8 indicando la velocidad a la que se mueve la línea sobre los sensores. Se puede utilizar la función **`millis()`** de Arduino (http://arduino.cc/es/Reference/Millis) para realizar la medición una vez cada `t_medicion` milisegundos (esto es necesario para permitir un intervalo de tiempo en el que el robot pueda reaccionar).

   *Nota: Debe definir una variable global **`int t_medicion`**. Puede usar 150ms como valor inicial, y ajustarlo experimentalmente en los ejercicios posteriores para que sea lo suficiente para que reaccione el robot.*

7. **Probar la función `int velocidad_error_linea(int posicion_linea)`**, leyendo el valor retornado a través del puerto serie y asegurarse de que el valor devuelto es correcto.

8. **Implementar el control proporcional derivativo.** Usando las funciones creadas y las constantes `int kp`, `int kd`, y ajustarlas empíricamente, siguiendo el método explicado en este documento:
   - *Ajustar la Kp de manera que el término proporcional cubra todo el rango de velocidades desde 0 hasta la velocidad crucero. **Es decir, que si el robot se sale por un lado, la rueda interior debería pararse mientras la otra gira a Vcrucero.***
   - *Al principio probar con la Kd = 0. El robot debería oscilar y salirse.*
   - *Ir subiendo la Kd hasta que deje de oscilar.*