Seguidor de Líneas Ultrafast

Volver al Índice

Diseño

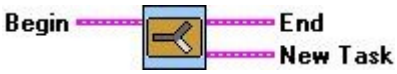
El diseño del robot no ha sido un factor clave. Tomando como base el robot standard que viene en la caja de LEGO NXT hicimos algunas modificaciones tales como añadir un segundo sensor de luz, elevar un poco el ladrillo, colocar las ruedas de cubiertas planas y diseñar una rueda loca que elevase un poco la parte de atrás para que los sensores se inclinasen un poco hacia adelante. Sin embargo, probando el mismo programa en el robot standard (más un segundo sensor de luz) las diferencias eran inapreciables. Podemos afirmar que la eficacia del robot reside en el programa.

Los sensores y los motores se controlan por parejas. En lo que sigue, el sensor 2 se asocia al motor A y el 3 al motor C.

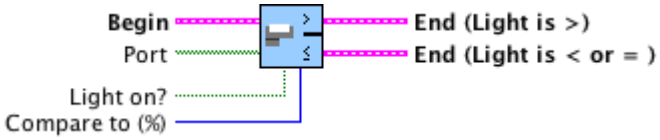
Bloques básicos

Hemos programado el robot en ROBOLAB. Los bloques básicos de programación a tener en cuenta son:

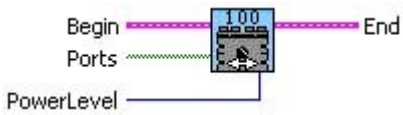
Este bloque, teóricamente, permitía una mejor respuesta de los motores (aunque sirve para los antiguos RCX leímos que también mejoraba los motores en el NXT. Lo mantuvimos en el programa, aunque no observamos ninguna diferencia apreciable).



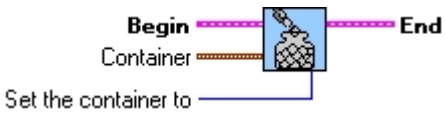
Cada sensor controla un motor, así que pensamos en dividir el programa en dos tareas simétricas. Cada tarea reconocía los valores de un sensor y actuaba en su motor. Sin embargo, posteriormente, nos dimos cuenta que la programación "en serie" era igual de efectiva, así que no es importante.



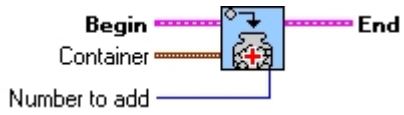
El próximo icono importante es el condicional basado en el sensor de luz. La opción "light on" debe ponerse en su valor verdadero (TRUE) para que el sensor encienda su piloto y detecte luz reflejada. Si el porcentaje de luz es superior a un valor de corte (Compare to %) sigue un camino de programación. Si es ese valor o inferior, continúa por otro.



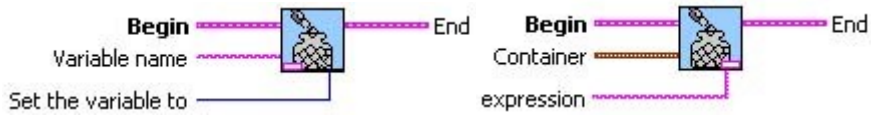
El siguiente icono sirve para controlar los motores mediante porcentaje de potencia. Si "Power Level" se establece a 0, el motor se para. Si a 100 el motor gira a máxima potencia. También es muy útil reseñar que con este icono se puede invertir el giro del motor introduciendo valores negativos.



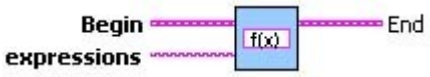
El siguiente icono introduce un valor en un contenedor. Los contenedores son de colores rojo, amarillo, azul o ya genéricos.



El icono del contenedor con un signo más permite añadir una cantidad a un contenedor. Si por ejemplo añado "2" al contenedor rojo donde había un "5", tras la adicción el resultado en el contenedor será "7"

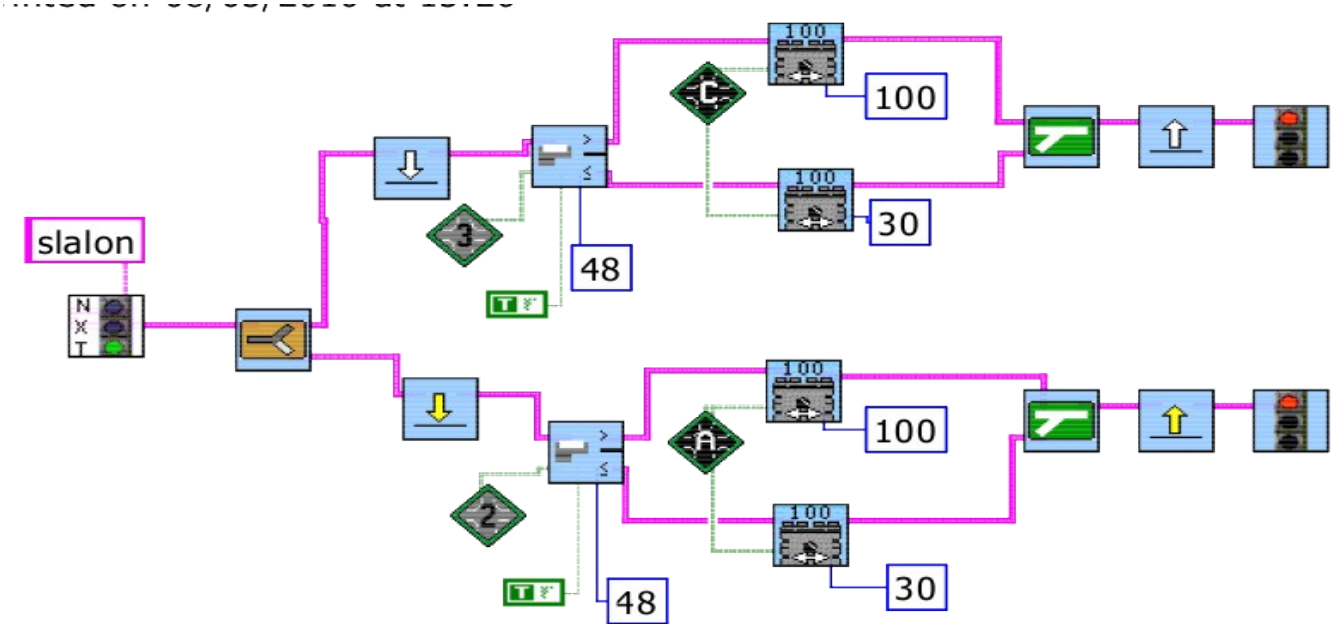


Estos dos comandos permiten asociar el valor de un contenedor a una variable con un nombre, y al contrario, el valor de una variable a un contenedor. son importantes porque...



...con este icono puedo calcular una variable en función de otra. Con los comandos anteriores asigno el valor de un contenedor a una variable, con éste hago el cálculo de una segunda variable usando la primera y, con el otro de los anteriores comandos, puedo asignar al valor de un contenedor la variable calculada.

Programa previo



Paso a explicar brevemente el programa básico que sigue la línea con dos sensores. Este programa es muy sencillo. Dividido en dos tareas independientes, la primera controla el motor C a través de su sensor asociado en la entrada 3. Si detecta un valor de luz inferior a una cota (oscuridad: línea negra), reduzco la potencia del motor (en el esquema a un tercio aprox.) y si detecta valores superiores a la cota (brillante: zona blanca) permite al motor ponerse a máxima potencia. Simétricamente, la segunda tarea realiza lo mismo sobre el motor A usando el sensor en la entrada 2.

El problema de este programa es que las curvas las toma siempre girando de la misma manera. Sabemos por nuestra experiencia conduciendo que una curva ligera necesita un leve giro. Una curva pronunciada necesita un giro más acentuado y una reducción de velocidad para evitar que la fuerza centrífuga nos haga salirnos de la trayectoria. Este programa, ante un circuito de curvas pronunciadas y curvas ligeras, puede salirse en las primeras debido a que su factor de giro (relación entre la potencia del motor que se frena y la potencia del motor que realiza el giro) no es suficiente o bien puede perder demasiado tiempo rectificando demasiado en curvas ligeras en las que no necesita tanto factor de giro.

Así que necesitamos una nueva idea que permita que los motores ajusten automáticamente sus potencias a la excentricidad (curvatura) de la curva.

Objetivo

Conseguir un diseño de Seguidor de Líneas rápido en las rectas, y preciso y rápido en las curvas, controlando el giro en las mismas según sea su trazado.

En las curvas necesitamos que

1. Vaya el robot a la velocidad que vaya en tramo recto, al tocar la zona negra con un sensor

rectifique la trayectoria.

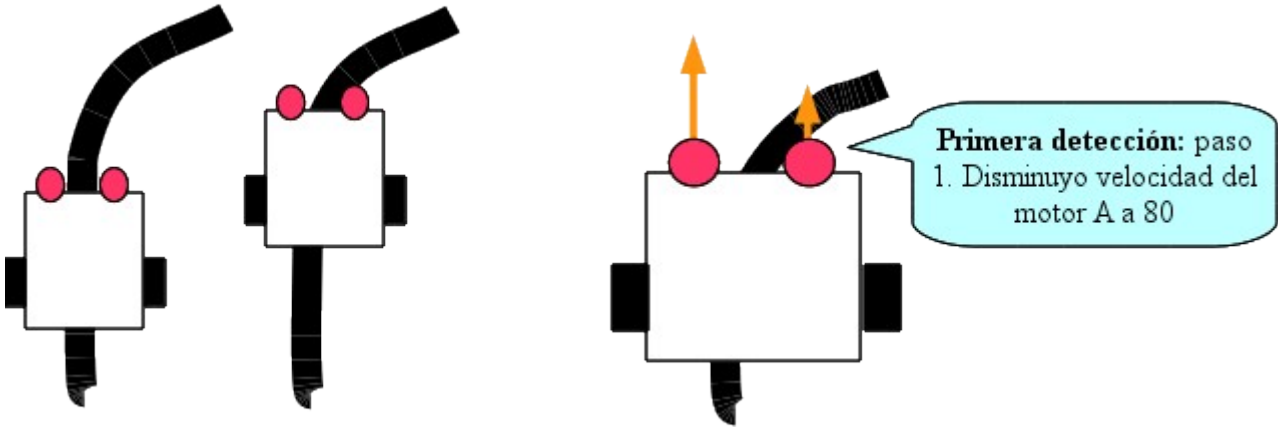
2. La rectificación de la trayectoria se hace disminuyendo la velocidad del motor donde está colocado el sensor (motor freno) respecto del otro motor que provoca el giro (motor giro).
3. Caben muchas posibilidades:

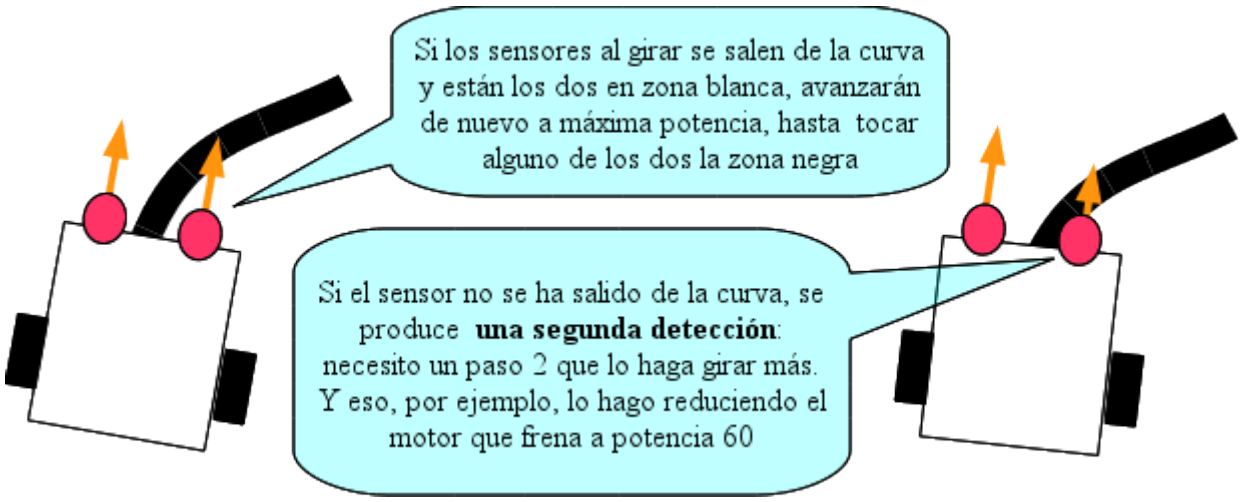
- El motor frenado disminuye su velocidad. El motor que gira continua a la velocidad que llevaba en recta.
- El motor de frenado disminuye su velocidad. Aunque menos, también frena el que gira.
- El motor de frenado disminuye su velocidad incluso hasta hacerse negativa. Aunque menos, también frena el que gira.
- El motor de frenado disminuye su velocidad. El que gira la aumenta.
- El motor de frenado disminuye su velocidad e incluso gira hacia atrás. El que gira la aumenta.
- En todo caso, siempre se debe cumplir que la relación entre la potencia de frenado y la potencia del motor que gira es menor que 1. **Potf / Potg < 1**

Y de todas formas, si la frenada es corta y sigue el sensor detectando línea negra, deberemos hacer que dicha relación sea aún menor, y si tras calcular una relación menor sigue aún estando en línea negra, aún otra menor... y así recursivamente hasta que gire lo que necesite y se salga de la línea.

Ejemplo de situación...

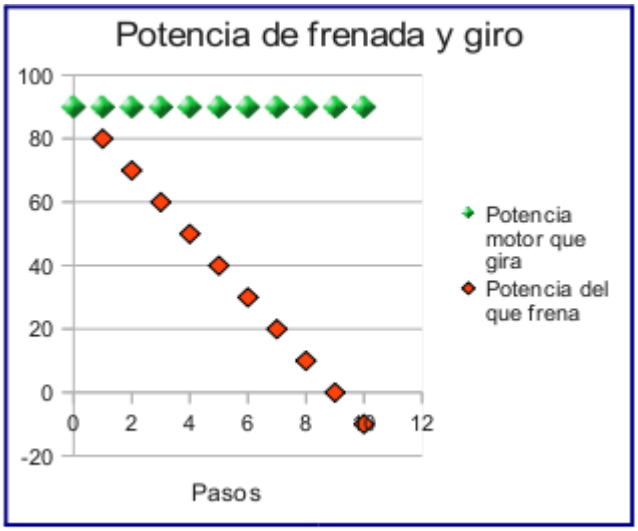
Vamos a suponer que nuestro robot se encuentra con una curva a derechas, donde está el sensor 2. Si en la recta desde donde proviene iban los motores a 90 de potencia, la primera vez que detecte y calcule una reducción de potencia suponemos que el motor que frena lo hace a 80 y el otro sigue igual.





Y si de nuevo al girar no se ha salido de la zona negra, deberé hacer que el motor que frena frene aún más respecto del que gira. Por ejemplo, disminuyendo en 50, 40, etc. Voy a suponer que ya en diez pasos se ha salido de la curva. Si represento en cada paso la potencia de cada uno tengo:

Paso	Potencia motor que gira	Potencia del que frena
0	90	90
1	90	80
2	90	70
3	90	60
4	90	50
5	90	40
6	90	30
7	90	20
8	90	10
9	90	0
10	90	-10



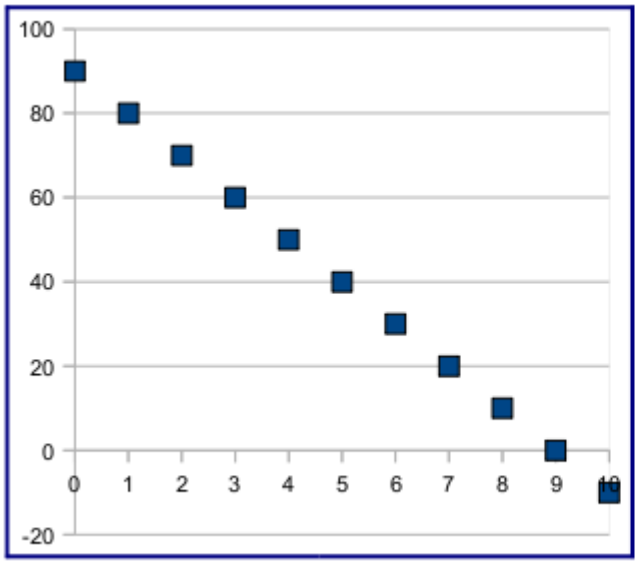
El paso 0 significa que ambos van rectos... El problema es: ¿cómo calculo una reducción de potencia de ese tipo automáticamente en el programa cuando voy contando los pasos?

Contar pasos....

Los pasos (variable X) se consiguen recorrer asociándolos a un contenedor, e ir sumando de uno en uno...

Matemáticas en la frenada

Observando el gráfico anterior observamos que la potencia de frenado parece ir disminuyendo siguiendo una recta, en la que la "X" son los pasos y la "Y" son las potencias. Parte de un valor inicial de 90 y termina en un valor final de -10.Calculo la pendiente, la ordenada en el origen y la ecuación de la recta.



Cálculo de la pendiente:

$$m=\frac{y_f-y_o}{x_f-x_o}=\frac{(-10)-90}{10-0}=\frac{-100}{10}=-10$$

Ordenada en el origen:

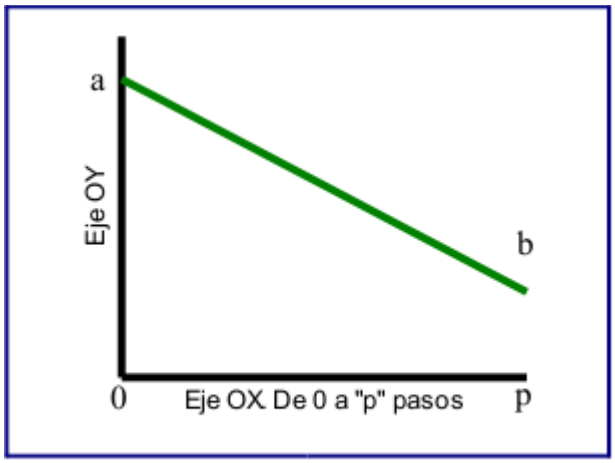
Es el valor de y cuando x=0, entonces n = 90

Fórmula de la recta

$$y=mx+n=-10x+90$$

Matemáticas en la frenada. Recta genérica.

Convendría calcular la ecuación de la recta según ciertos parámetros genéricos; así parto de una potencia "a" hasta una potencia "b" (a>b). Calculo pendiente, ordenada en el origen y ecuación de la recta.



Cálculo de la pendiente:

$$m=\frac{y_f-y_o}{x_f-x_o}=\frac{b-a}{p-0}=\frac{(b-a)}{p}$$

Ordenada en el origen:

Es el valor de "y" cuando x=0, entonces n = a

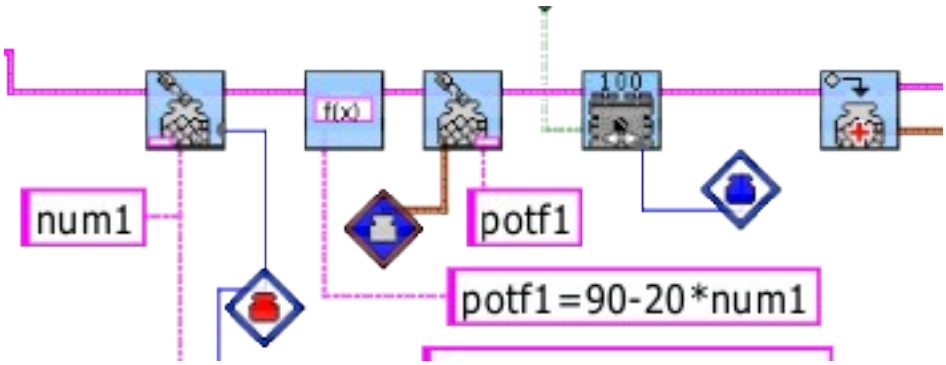
Fórmula de la recta

$$y=mx+n=\frac{(b-a)}{p}x+a$$

Así consigo una recta genérica...

Diseñar el programa que disminuya la potencia en el motor de frenado.

El programa que permite entoces hacer que cada vez que el sensor detecte línea negra disminuya la potencia del motor que tiene que frenar es el siguiente:



Este conjunto de órdenes se encuentra en la rama del condicional en la que el **sensor 3** encuentra línea negra. Si es la primera detección, el valor del contenedor rojo es 0. Ese valor pasa a la variable **num1**. Con esa variable calculo una potencia **potf1** mediante el comando fórmula (en el ejemplo: potf1 = 90-20*num1). Ese valor de potf1 se asigna al contenedor azul y dicho valor se aplica al motor. Es importante que al final, **sumo un valor de 1 al contenedor rojo** (no hace falta poner 1.Si el valor que se suma se omite por defecto suma 1). Si num1=0, potf1=90 que es el valor que lleva en zona blanca.

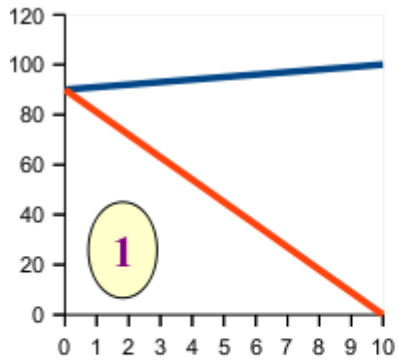
Si es la segunda detección, ahora num1 =1 y calculo potf1=70. El motor frena, y el contenedor rojo aumenta a 2.
Si es la tercera detección, ahora num1=2 y potf1=50. El motor sigue frenando...

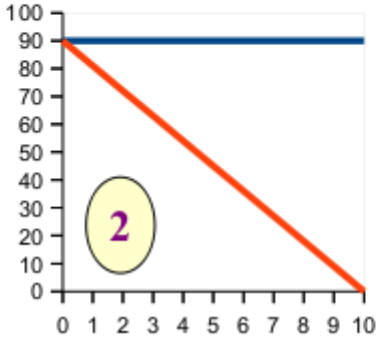
Y así sucesivamente hasta que el giro dado haya sido capaz de sacar al detector de la línea negra.

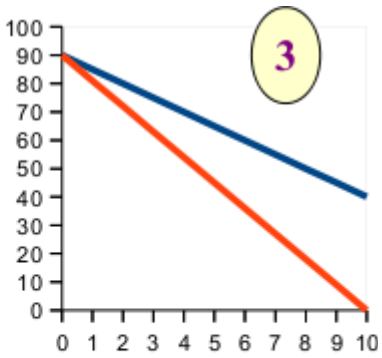
Evidentemente, el problema se reduce a conseguir una recta de frenada eficiente, que combine seguridad al cogerla y rapidez. Por ello, y para ir investigando, necesito la fórmula genérica de la recta que antes expongo. Puedo calcular rectas rápidamente eligiendo tres parámetros: **potencia en recta (a), potencia de frenada máxima (b) y número de pasos (p).** ¿Qué recta es la adecuada? Antes debo pensar en el otro motor, el que gira... Si en el ejemplo a=90... ¿debo aumentar su velocidad en la curva? ¿debo disminuirla o mantenerla? Por una parte podría querer que se aumente a 100 (a=100) porque así en recta va lo más rápido posible. ¿Podré conseguir ese valor y que en las curvas no se salga?

El motor de giro

El motor de giro necesita también ser modificado. Mientras su potencia sea mayor que la potencia del que frena hará su trabajo. Sin embargo podremos jugar a subirle la potencia más que la que llevaba en la recta (1), dejarla igual que en recta (2) o bajarle la potencia para controlarlo más aunque siempre siendo mayor que la del motor que frena (3) .

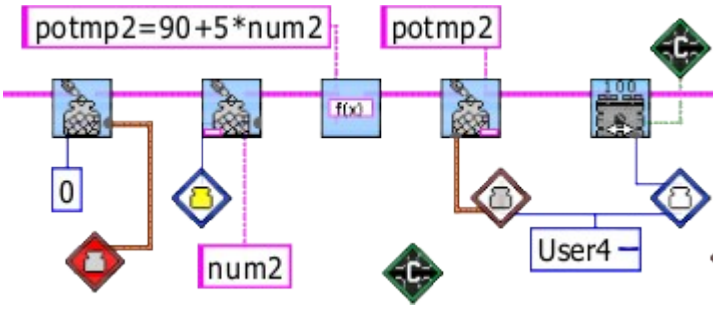






Diseñar el programa que controla la potencia del motor que gira

De forma análoga, podemos controlar el motor que realiza el giro:



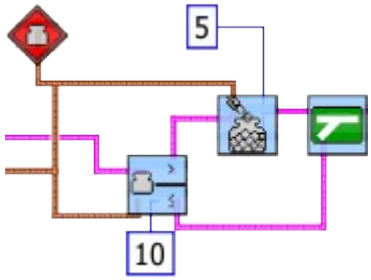
El motor que realiza el giro es controlado por el sensor contrario ¿cómo? cuando el otro sensor (en este caso el 2) detecta línea negra, **num2** empezará a cambiar y asímismo la potencia del motor C. Si num2=0, la potencia se fijará a 90. Al cambiar num2, en la otra tarea se frena el motor A y en esta tarea se modifica el motor C.

El primer icono asegura que el contenedor rojo es 0. Cuando el sensor 2 detecta zona blanca debe reiniciar este contador para cuando detecte línea negra empiece de nuevo el conteo.

En este caso, la recta aplicada aumenta rápidamente de 90 a 100. Los valores superiores a 100 no tienen sentido fijándose el tope en 100. Estaríamos en el caso de la recta (1) del apartado anterior.

¿Qué ocurre si el programa no se sale de la línea en el último paso?

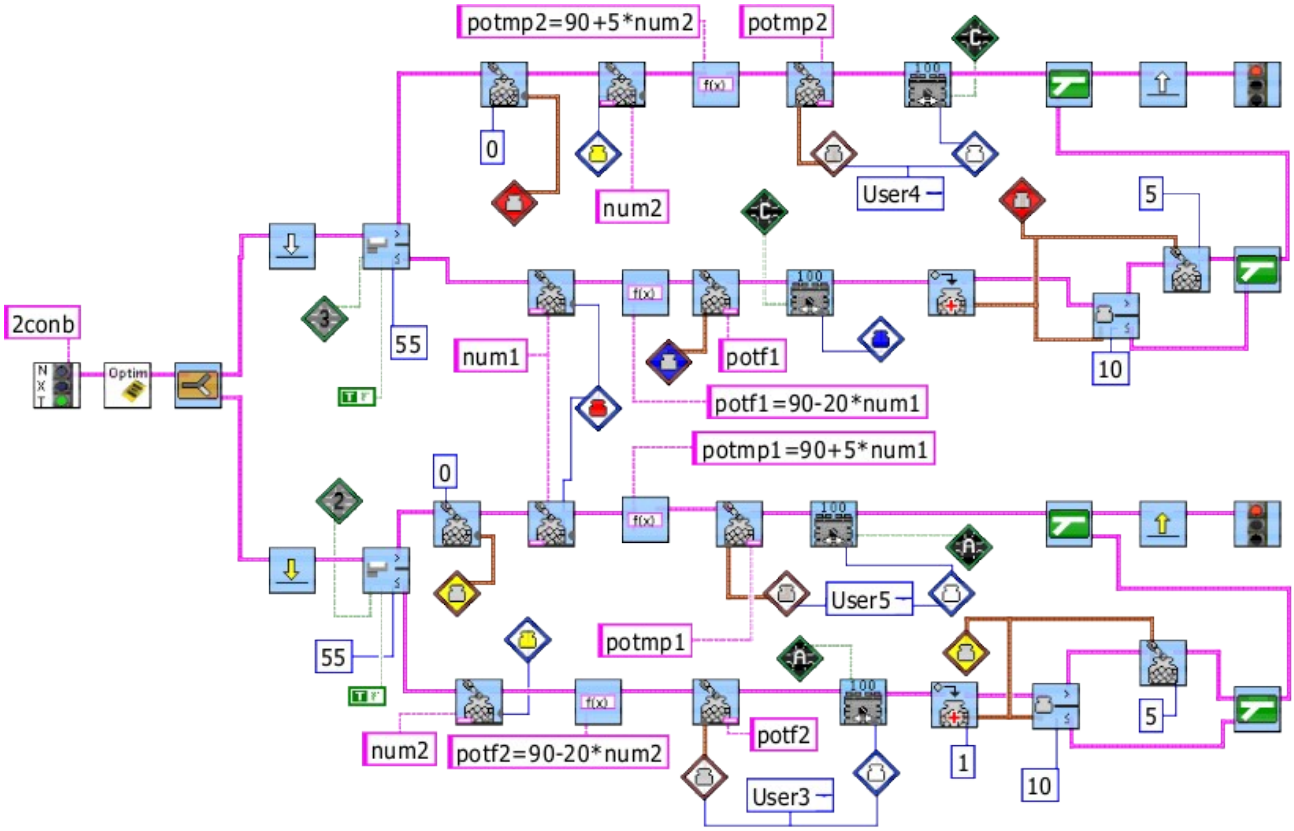
Esta circunstancia es difícil que se produzca (incluso para algunos valores de recta es imposible), pero si lo hace, debo provocar un salto brusco y puntual en la trayectoria. Simplemente lo consigo rompiendo la relación que van llevando las potencias de una con otra. Si por ejemplo he calculado una recta en 10 pasos, y he llegado al 11, salto bruscamente a la mitad, al 5. Eso haría que el robot avanzase caóticamente un instante sacándolo de su posible bloqueo.



El bloque es un simple condicional según el valor del contenedor rojo. Si es mayor que 10, se fuerza a 5, y el programa continúa.

Programa completo

Finalmente, os dejo el programa completo. Cada tarea es simétrica con la anterior.



Cálculo de varias rectas. Cálculo de parámetros.

NOTA: tengo que dar más datos de los cálculos que hicieron los chicos de varias rectas que investigaron.

1. (Unos 24-25 s. por vuelta) Recta de frenado: y=90-20x ; Recta de giro: y=90+5x ; número de pasos 10.
2. El mínimo tiempo logrado (22-21 s. por vuelta): recta de frenado y=100-20x ; Recta de giro y=100; pasos 10.

En el segundo caso se obvia una recta y se pone la potencia del motor que gira al máximo. También se observa que con pasos = 10 el motor gira todo lo posible ya que una rueda avanza a 100 y la otra retrocede a 100, con lo que los bloques de programación que controlan la potencia del motor que gira y el corte de número de pasos se pueden obviar simplificando mucho el programa si se elige este caso concreto.

Una vez obtenido el programa, lo único que habría que modificar son los valores de corte de los sensores de luz. Éstos no suelen dar valores simétricos, con lo que conviene hacer mediciones por separado. También es conveniente que estos valores de corte sean próximos a los valores máximos en zona blanca. Si por ejemplo en zona blanca tenemos 60 de luz, un valor de corte apropiado es 58.

Otros resultados:

1. Pasos=10 ; frenado: y=95-7,5x ; giro: y=95+10x. Tiempo 24"
2. Pasos = 20; frenado: y=100-10x ; giro: y=100. Tiempo 20" . Inestable.
3. Pasos = 5; frenado: y=100-40x ; giro: y=100. Inestable.
4. Pasos = 50; frenado: y=100-4x ; giro: y=100. Se sale.
5. Pasos = 3; frenado: y=100-66x; giro: y=100. En torno a 21". No tan bueno como el de y=100-20x.

Eficacia de las distintas rectas.

A pesar de que el programa ha funcionado bien la mayoría de las ocasiones si es cierto que con velocidades máximas de 90 en recta es más fiable que a 100. En el primer caso, ha llegado a completar 20 vueltas sin salirse del circuito.

En el segundo observamos ciertas alteraciones "inexplicables". Una de cada diez veces, más o menos, el programa hace que el robot se frene continuamente, como si entrara en los dos bucles de frenado a la vez detectando "línea negra" en los dos sensores, aunque parezca que físicamente ambos estén en zona blanca. También es cierto que a veces se sale del circuito. Cuando se da esta circunstancia, sospechamos que de alguna manera existe una mínima variación de la luz (es muy sensible a perturbaciones).

Es curioso observar que, además, con este programa el robot avanza en línea recta si los dos sensores están en zona blanca y al llegar a la línea negra rectifica su trayectoria siguiéndola. Además, si se encuentra el robot con el borde de una mesa el robot no se cae.