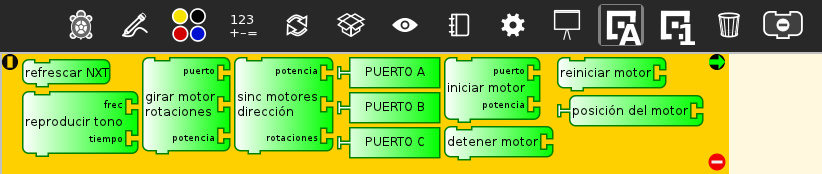
**LEGO Nxt-tortuga**

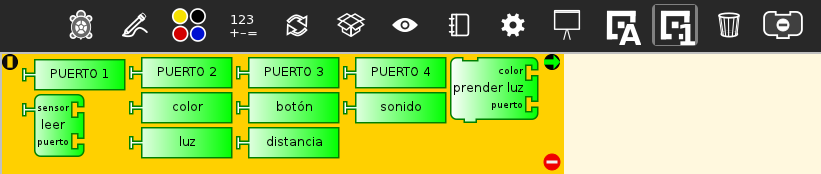
Tortugarte cuenta con un nuevo plugin para poder utilizar el kit Lego Mindstorms NXT, y controlar nuestro robot desde allí. Descarga: [nxt\_plugin.tar.gz](http://wiki.sugarlabs.org/go/File:Nxt_plugin.tar.gz)

El plugin nos agrega dos paletas para trabajar con el robot: una de motores y otra de sensores.

* La paleta de motores, identificada con el logo y una letra A (referida a que los motores son conectados en los puertos A, B o C).

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Paleta_nxt_motores.png)

* Y una paleta de sensores identificada con un 1 (también referida al sistema Lego que conecta sensores en los puertos 14, 2, 3 y 4):

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Paleta_nxt_sensores.png)

Bloques de motores:

* refrescar nxt: Reestablece la conexión con el Brick y repinta la paleta de acuerdo al estado: Verde si está disponible y Gris en caso c contrario.
* reproducir tono: Reproduce un tono a cierta frecuencia y cierta cantidad de tiempo.
* girar motor: Gira un motor cierta cantidad de vueltas.
* sinc motores: Sincroniza dos motores para que giren sincronicamente. Los motores deben estar en los puertos B y C.
* PUERTO A, B y C: Representan los puertos A, B y C del Brick para conectar motores.
* iniciar motor: Pone a un motor en rotación continua a una potencia especificada.
* detener motor: Frena el motor especificaado.
* reiniciar motor: Resetea el contador interno del motor y lo deja en modo libre.
* posición del motor: Obtiene la posición desde el último reinicio. El valor es en grados absolutos, por ejemplo, 1 vuelta y media serán: 360 + 180 = 540.

Bloques de sensores:

* PUERTO 1, PUERTO 2, PUERTO 3, PUERTO 4: Respresentan los puertos 1, 2, 3 y 4 del Brick para conectar sensores.
* leer: lee un sensor específico en un puerto específico.
* color: sensor de color RGB
* luz: sensor de luz
* botón: sensor de botón
* distancia: sensor de distancia (ultrasónico)
* sonido: sensor de sonido
* prender luz: prende el led correspondiente en el sensor de color

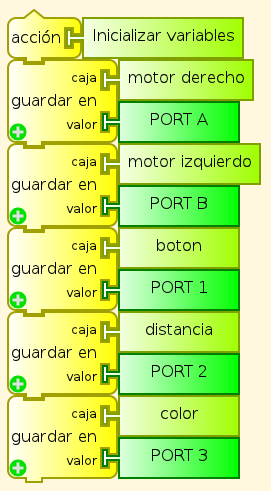
**Cómo comenzar ...**

En todo momento, durante nuestro código, para utilizar los bloques del robot nxt deberemos indicar el puerto en donde está conectado un motor a mover, o un sensor del que se quiere obtener una medición; lo que puede significar una complicación si luego se quiere ejecutar el mismo código en otro robot, que no tiene conectados los motores/sensores en el puerto que habíamos indicado en nuestro programa para el robot inicial.

Para evitar tener que modificar todas las ocurrencias del puerto en nuestro programa, o re-conectar los motores/sensores en el nuevo robot, podemos definirnos variables que identifiquen a los motores/sensores a utilizar, y almacenar inicialmente los valores del puerto en el que están conectados. De esta manera, cuando se modifican las conexiones de los mismos, solo alcanzará con actualizar estos valores en la inicialización.

Por ejemplo, tenemos construido un robot con dos motores para mover las ruedas, un sensor de distancia, un sensor botón y un sensor de color.

Definimos así una acción que inicialice las variables:

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Varsinicializar.png)

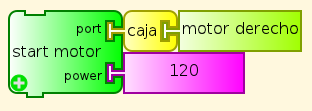
Y en nuestro código nos referimos al motor que está conectado en el *PUERTO\_A* como **motor derecho**, al sensor conectado en el *PUERTO\_2* como **distancia**, y así con todas las variables definidas.

**Primeros movimientos y ejemplos**

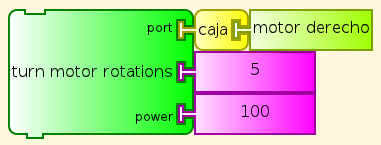
Estos son algunos movimientos por los cuales podemos comenzar, y así familiarizarnos con el robot y Tortugarte.

**Mover un motor**

* Podemos darle instrucciones a un motor para que comience a girar. Por ejemplo, utilizamos el bloque **start motor**, le colocamos el puerto en el cual está conectado nuestro motor, en el ejemplo debajo le pasamos la variable *motor derecho* donde previamente almacenamos dicho puerto. Además podemos indicar la velocidad a la que queremos que gire, entre el rango [-128,127], donde para valores negativos gira en un sentido, y para valores positivos en el sentido contrario.

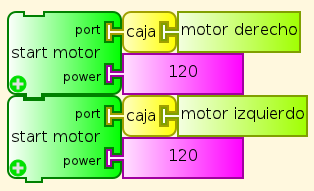
[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Movermotor1.png)

* También podemos mover un motor una cantidad de vueltas dadas, para esto utilizamos el bloque **turn motor**, conectamos el puerto del motor, la cantidad de vueltas que deseamos que gire, y la velocidad.

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Movermotor2.png)El motor derecho gira cinco vueltas.

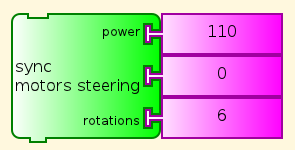
**Mover dos motores**

* Para mover dos motores podemos encastrar dos bloques **turn motor**, uno con cada motor a mover e indicando la velocidad de giro de cada uno.

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Movermotor3.png)

En este ejemplo se logra avanzar ambos motores en la misma dirección. Pero se pueden lograr diferentes giros variando las velocidades entre los motores. Para este caso también tenemos el mismo rango de variación de la velocidad [-128,127].

* Este plugin además cuenta con un bloque que permite mover dos motores conectados específicamente en los puertos B y C en forma sincrónica. Se especifican las rotaciones de los motores y su velocidad

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Movermotores4.png)

**Cuadrado**

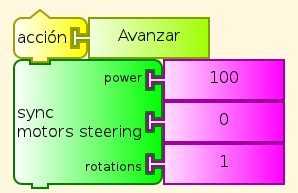
repetir 4 veces:

avanzar robot

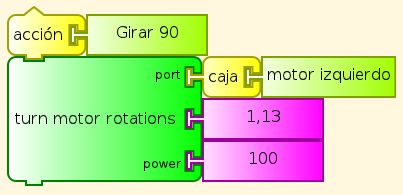
girar 90 grados

* Definimos acciones individuales para los movimientos *avanzar* y *girar 90 grados*. Es importante comprender la utilidad de modularizar nuestros programas, ya que permite que estos sean más claros, y que su corrección sea más fácil.

Se conectan los motores en los puertos B y C, ya que en la acción avanzar se utiliza el bloque **sync motors** que sincroniza estos dos puertos justamente.

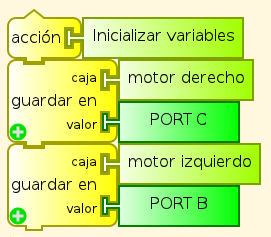
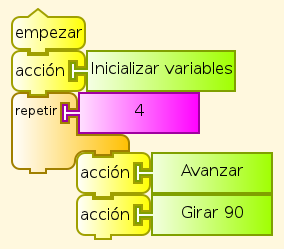
[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Avanzar1.png)

Para el caso de girar 90 grados el robot, se utiliza el bloque **turn motor**, hay que tener en cuenta que la rotación del motor (*rotations*) se debe ajustar hasta conseguir el giro que queremos. Más adelante se presentará una forma más eficiente de conseguir este mismo resultado con la utilización del bloque Python

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Girar90.png)

También se puede definir este mismo giro hacia la derecha, haciendo uso de la variable *motor derecha* en vez de la correspondiente *motor izquierdo*.

Nuestro programa principal queda entonces:

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Initvarscuad.png)[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:EjCuadrado.png)

**Utilizando sensores**

* El kit cuenta con tres sensores:

**Botón**

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Boton.png)

Este sensor devuelve *1* cuando el botón esta presionado y *0* si no lo está.

Un ejemplo sencillo con este sensor:

Por siempre

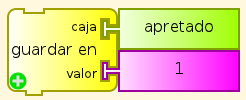
avanzar

si botón == apretado

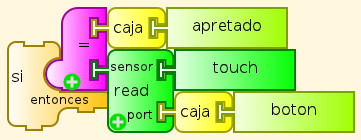
retroceder 1 vuelta

girar

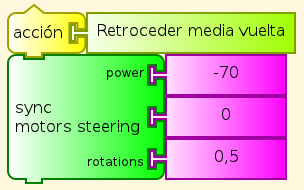
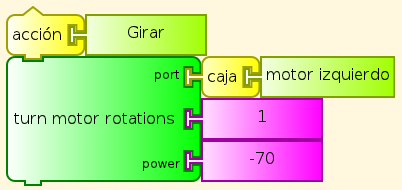
De esta forma podemos definirnos una variable *apretado*, la cual almacene el valor constante 1, y nos sea útil para comparar el estado del botón en cualquier momento en nuestro programa.

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:B_apretado.png)

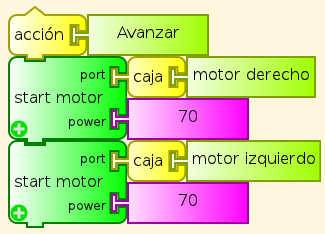
Se utiliza la estructura de control *si-entonces* y se compara la lectura del sensor botón con la variable *apretado*

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:If-then-boton.png)

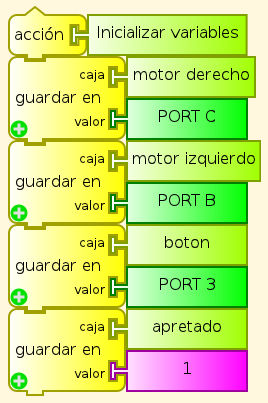
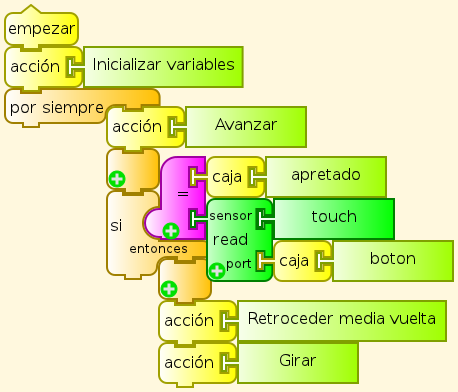
en el caso que el botón esté presionado se retrocederá media vuelta y se girará. Notar la utilización de velocidades negativas, para que retroceda el robot.

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Retrocedermediavuelta.png)[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Girarmediavuelta.png)

Este bloque se incluye en el programa, en un bucle infinito *Por-siempre* a continuación de la instrucción que ordena al robot que avance:

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Avanzarboton.png)

El programa principal y la definición de la función *inicializar variables* nos queda:

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Initvarsboton.png)[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Ejsensorboton.png)

**Distancia**

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Distancia.png)

Con este sensor podemos conocer la distancia del objeto mas próximo a nuestro robot, y utilizar esta información para movernos en un entorno. Es capaz de detectar objetos que se encuentren desde 0 a 233 cm, con una precisión relativa de +/- 3 cm, y funciona mejor cuando las señales ultrasónicas que recibe provienen de objetos que son grandes, planos o de superficies duras. Los objetos pequeños, curvos o suaves, como pelotas, pueden ser muy difíciles de detectar.

Podemos escribir el ejemplo anterior con este sensor en lugar de utilizar el sensor botón.

Por siempre

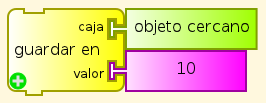
avanzar

si distancia < objeto cercano

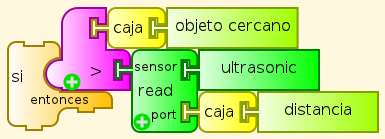
retroceder 1 vuelta

girar

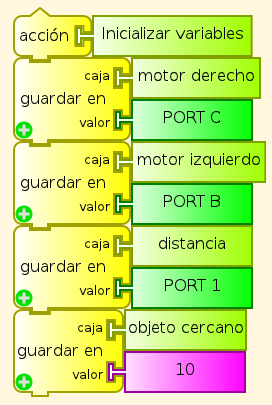
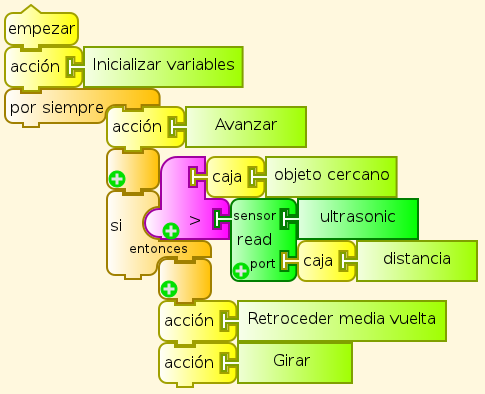
En este ejemplo definimos una variable que almacene una distancia mínima, frente a la cual el robot deba buscar otro rumbo.

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Objcercano.png)

Utilizamos la estructura de control *si-entonces* para poder indicarle al robot que retroceda y gire cuando encuentra un objeto cercano.

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:If-then-dist.png)

Reutilizamos las funciones definidas en el ejemplo anterior para los movimientos del robot ("Avanzar", "Retroceder media vuelta" y "Girar"). El programa principal y la definición de la función *inicializar variables* nos queda:

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Inicvarsdist.png)[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Ejsensordistancia.png)

**Color**

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Color.png)

Este sensor tiene dos funcionalidades:

* Permite encender un led del color que le pasemos como parámetro (verde, azul, rojo, o blanco) al bloque **set light**

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Setlight.png)

* Y permite detectar colores: *blanco* = -9998, *negro* = -9999, *rojo* = 0, *verde* = 40, *azul* = 70, y *amarillo* = 20.

Ejemplo:

Por siempre

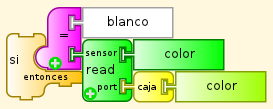
avanzar

si color sensado == blanco

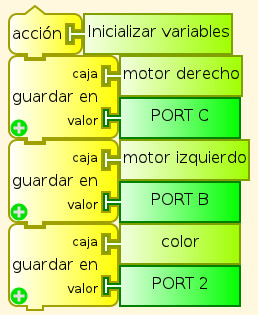
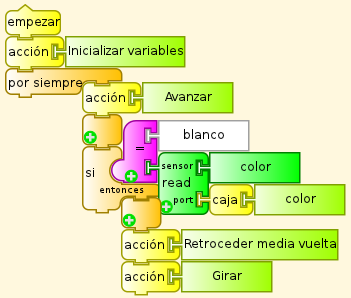
retroceder 1 vuelta

girar

El ejemplo se podría aplicar en una zona delimitada por líneas blancas, el robot avanzando siempre, y que cuando detecte una de estas líneas borde, retroceda y gire, manteniéndose siempre entre los límites de la zona. Utilizamos el bloque **read** para leer el valor del sensor color en todo momento, y cuando el valor sensado se corresponda con *blanco* retrocedemos, giramos y continuamos avanzando en otra dirección.

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:If-then-color.png)Aquí comparamos el valor sensado con el de los límites de la zona (*blanco*).

El programa principal nos queda: (reutilizando las funciones *Avanzar*, *Retroceder* y *Girar* que definimos anteriormente).

[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Initvarscolor.png)[](http://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Archivo:Ejsensorcolor.png)

**Nota.** Estos son algunos ejemplos para tener una idea de las funcionalidades de la paleta con el robot Lego NxT, la idea es probar todos los bloques que tiene con diferentes valores y ver los resultados que se obtienen.