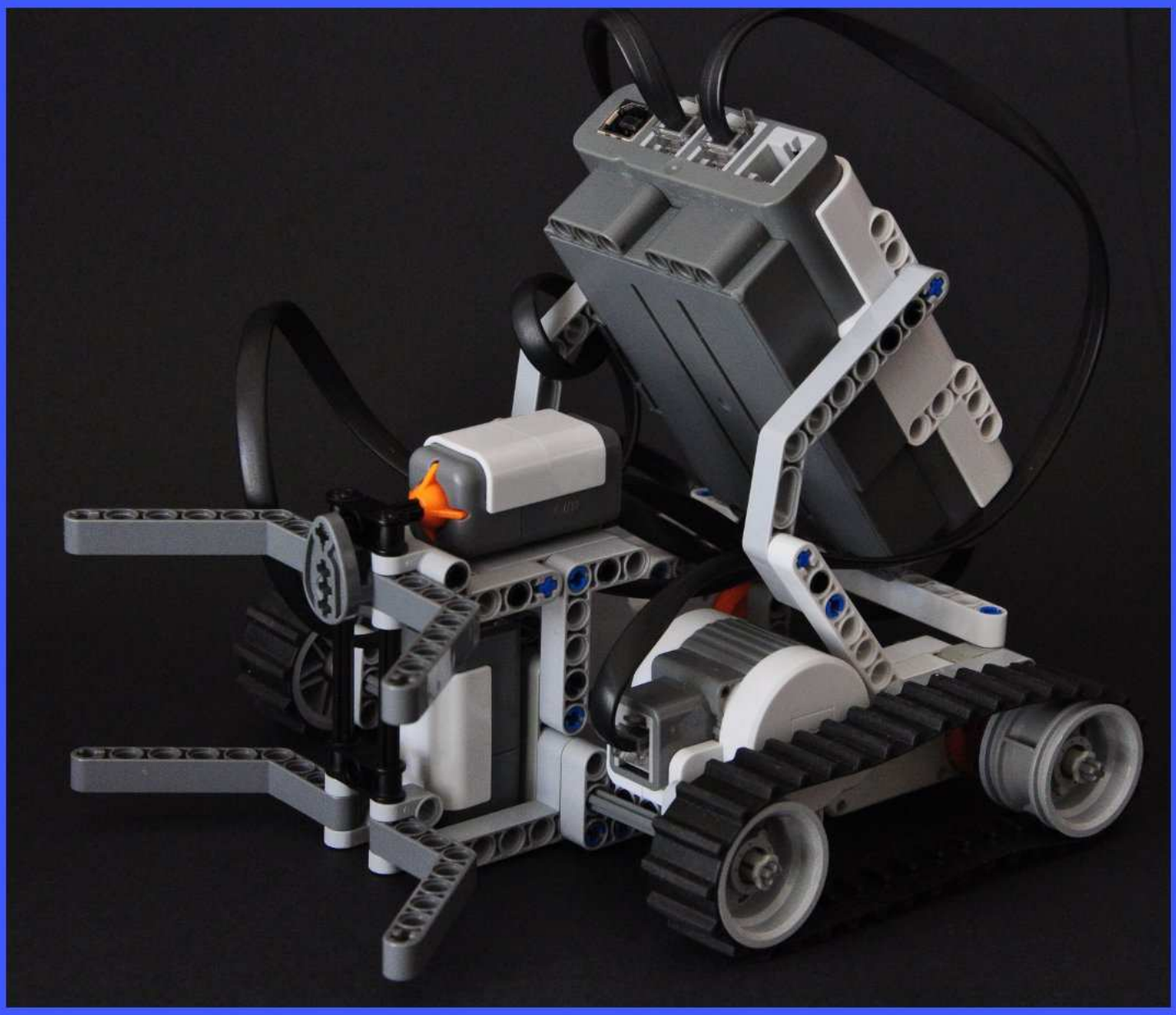
Retos con LEGO MINDSTORMS 1

Nivel Medio

Latas fuera



Koldo Olaskoaga
http://lrobotikas.net

En este cuadernillo se presenta un reto a resolver con un robot LEGO MINDSTORMS programado con NXT-G, así como los pasos necesarios para llegar a una de las posibles soluciones. Y digo una de ellas porque del mismo modo que sucede con cualquier problema en ingeniería siempre hay más de una solución, algunas más eficientes que otras.

La descripción del proceso incluye algunos de los errores que he cometido y cómo los he solucionado.

Atribuciones

LEGO y MINDSTORMS son marcas registradas de LEGO Group.

Las instrucciones de montaje las he dibujado con MLCad para a continuación generar las instrucciones con LPub.

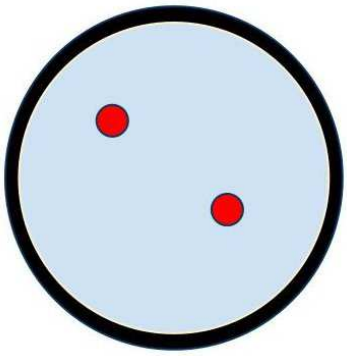
Contacto

Si deseas ponerte en contacto con su autor (sugerencias, propuestas de mejora…) lo encontrarás en http://lrobotikas.net.tienes dudas sobre el reto puedes compartirlas en el foro de la misma web.

1.-Propuesta

En el interior de un área circular limitada por una línea negra hay dos latas de bebida de 330cc llenas. Se trata de montar y programar un robot que saque las dos latas fuera del área circular y que tras ello emita un sonido y se detenga.

El material disponible es una caja de LEGO MINDSTORMS NXT 2.0 pero por diversas circunstancias solo se dispone de un sensor de contacto y uno de color (el sensor de color se puede sustituir por uno de luz).



NOTA: El área circular se puede sustituir por una rectangular. El interior no tiene porqué ser blanco, con que sea un tono claro que se diferencie bien del negro es suficiente. En la figura las latas se representan por medio de los círculos rojos.

2.-Por dónde empezar

La planificación es fundamental si queremos alcanzar un buen resultado. Reflexionar y determinar los pasos a dar debe ser el punto de partida. A la hora de hacerlo no hay que olvidar que cualquier reto con robots tiene dos partes bien diferenciadas y a la vez estrechamente relacionadas entre sí: el hardware (el robot propiamente dicho) y el software (el programa). Cada una de estas partes condiciona la otra, así que en un proceso de ensayo y error habremos de tener en cuenta las dos.

Las tareas a realizar serán las siguientes:

1. Diseñar y montar el robot
2. Escribir el programa (en el caso de NXT-G enlazando bloques de programación). Para ello primero habrá que escribir el algoritmo para a continuación convertirlo en un programa.
3. Probarlo y cuando sea capaz de desarrollar la tarea asignada
4. Mejorarlo

3.-El robot

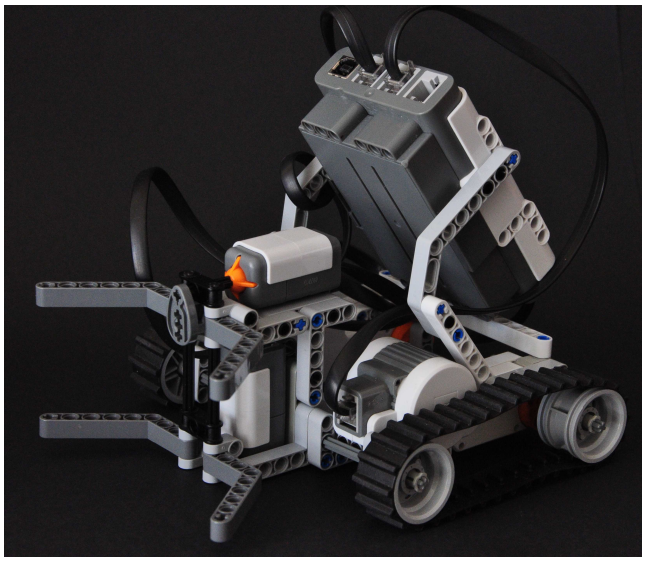
Para empezar conviene tener claro cuáles son las restricciones (límites que no podemos sobrepasar es su montaje y programación) que plantea el reto y las habilidades que ha de poseer el robot, que en este caso son las siguientes:

Restricciones

- El robot sólo puede utilizar un sensor de contacto y uno de luz. No hay ninguna en cuanto a tamaño y otras piezas a utilizar.

Habilidades

- El robot ha de ser capaz de desplazarse y girar sobre el área de juego.
- Será capaz de detectar la línea negra que limita el área manteniéndose dentro del área de juego.
- Será capaz de detectar cuándo choca contra una lata y sacarla del área de juego.
- Será capaz de saber cuándo ha sacado las dos latas.



Una vez que esté montado y comencemos a probarlo probablemente se presente algún problema que requiera cambios. Por ejemplo, un problema que se me ha presentado está relacionado con el sensor de color: cuando he probado el primer programa el sensor leía cualquier color como negro. Tras varias pruebas, suspiros, cambios… me he dado cuenta que el problema era que el sensor estaba demasiado pegado al suelo, lo que impedía que la luz que emitía se reflejase y volviese al sensor. En consecuencia todo era negro para él. Elevando un poco el sensor el problema se ha resuelto.

4.-El algoritmo

Antes de empezar a combinar los bloques que definirán el programa conviene escribirlo en nuestro propio lenguaje natural, es decir, escribir los pasos que pensamos que ha de dar el robot para conseguir su objetivo. No hay un algoritmo único para una tarea del mismo modo que las personas realizamos nuestras tareas de diferentes maneras. Así que después de escribirlo conviene reflexionar por si se nos ocurre otro modo de conseguir el objetivo de un modo más eficiente.

En este caso las tareas a programar serán las siguientes:

1. Crear y poner un contador a cero (almacenará el número de latas que ya han sido sacadas del área circular)
2. Avanzar recto hasta que el robot detecte raya negra u objeto
    a. Si detecta objeto seguir avanzando hasta que detecte la línea negra (lo cual querrá decir que la lata está fuera) y sumar al contador de latas una unidad
3. Detenerse, retroceder y girar
4. Si el contador es 2, emitir un sonido y detenerse sino volver al principio del programa

Este no es mas que un punto de partida que puede requerir modificaciones o mayor concreción en alguno de los puntos a lo largo de la programación.

5.-El programa

Ya sabemos cuáles son los pasos que ha de dar el robot para conseguir su objetivo, pero no es necesario que nos enfrentemos al programa completo desde el principio, ni recomendable. Una estrategia muy útil es descomponer los problemas complejos en problemas sencillos; en este caso descomponerlo en dos retos que resolveremos previamente antes de combinarlos en uno solo:

1. robot que se mueve de modo aleatorio en el área de juego sin salir fuera de ella.
2. robot que empuja una lata fuera del área de juego y que retrocede y emite un sonido una vez lo haya conseguido.

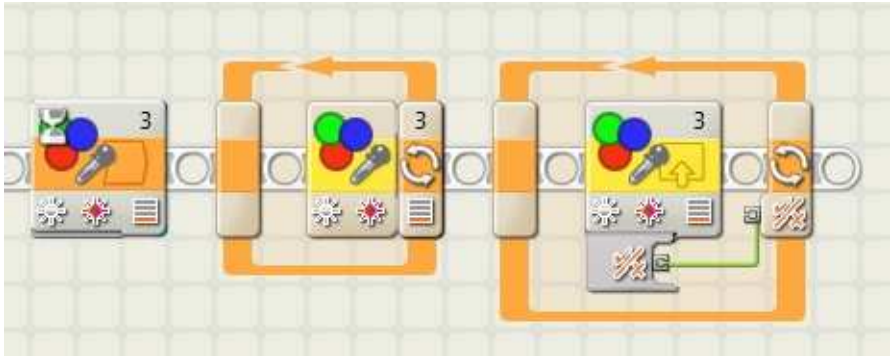
Reto básico 1

Veamos el algoritmo del primero antes de ver el código:

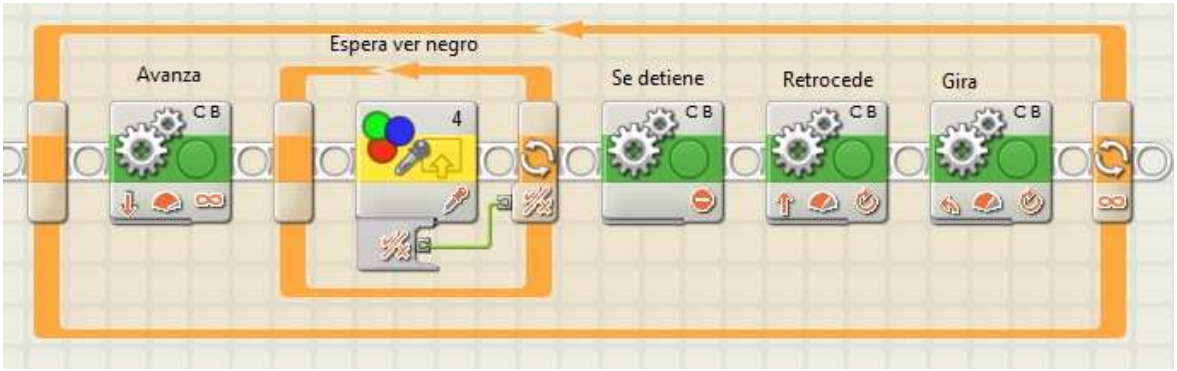
1. Avanza hasta que el sensor de luz lee negro
2. Se detiene

3. Retrocede
4. Gira
5. Repetir los mismos pasos.

Para decirle qué ha de esperar a que el sensor lea negro se puede utilizar cualquiera de las tres siguientes opciones. La funcionalidad es exactamente la misma, sin embargo, la tercera nos abre el camino para la combinación de los dos retos.



Dado que se tienen que repetir los mismos pasos una y otra vez habrá que colocar los bloques en el interior de un bucle tal y como se ve en la siguiente figura:

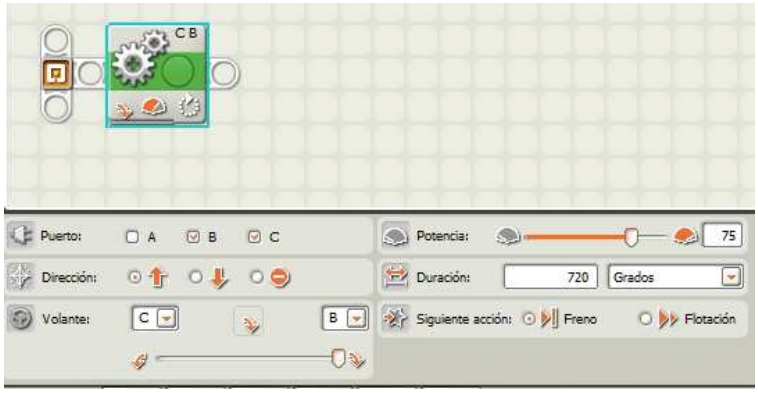


Hay que probarlo y observar si su comportamiento es el esperado.

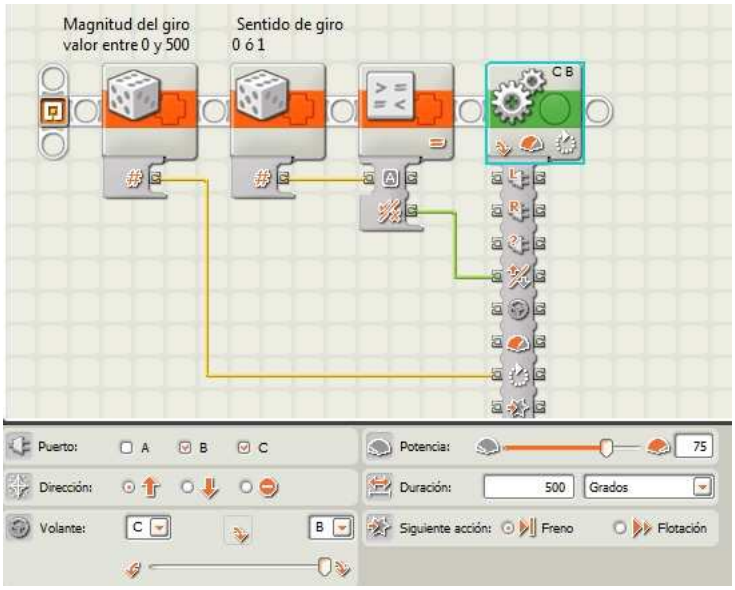
Mejora al reto básico 1

Si se observa cómo navega el robot con el anterior programa, se puede ver que los giros siempre los hace iguales. Si se quiere que la navegación no sea tan predecible hay que permitir que sea el robot el que decida sobre su giro.

Vamos a programar el robot decida si gira a izquierda o derecha y que ese giro esté comprendido entre 0º y 90º. Antes que nada hay que determinar cuánto han de girar los motores para que el robot gire 90º, para ello en este caso la única manera es el tanteo. En mi caso he empezado con un giro de 720º que al probar se ha visto excesivo.

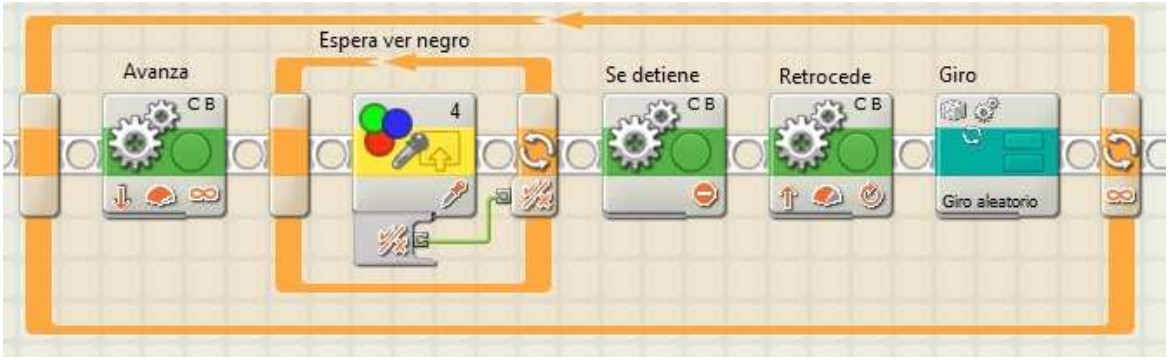


Tras varias pruebas he comprobado que con el valor de 500 el giro se acerca lo suficiente a 90º. Una vez que ya conocemos la magnitud del giro máximo será necesario generar dos valores aleatorios: uno para determinar el sentido de giro (0 ó 1) y el segundo para su magnitud (entre 0 y 500). En la siguiente figura puede verse el fragmento de programa correspondiente al giro.



El cable que lleva la magnitud del giro está conectado a la entrada ***Duración*** del bloque ***Mover***. Para que interprete este valor como grados habrá que seleccionar en su panel la opción ***Grados*** (***Segundos*** si lo que entra es tiempo).

Para simplificar el programa se puede convertir este fragmento de programa en un nuevo bloque de programación de tal manera que el nuevo programa para el Reto básico 1 será el siguiente:

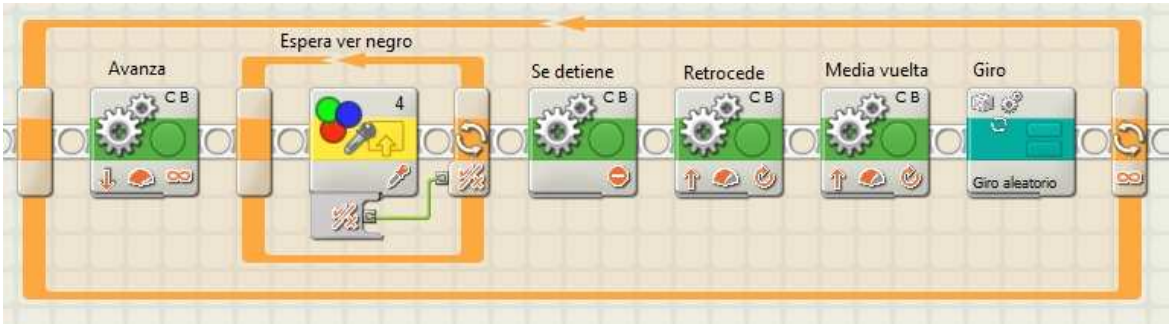


Una vez hecho el programa toca probarlo. Y… ¿cuál ha sido el resultado? Pues… no ha sido el deseado, me he dado cuenta que el robot una vez alcanzada la línea retrocedía y a continuación a partir de esa posición giraba a la izquierda o derecha con un máximo de 90º a cada lado, cuando en realidad lo que quería era que hiciera eso una vez que estuviese mirando en sentido contrario. Las posibles soluciones son dos:

1. Girar primero 180º hasta mirar en el sentido contrario y después girar a un lado u otro.

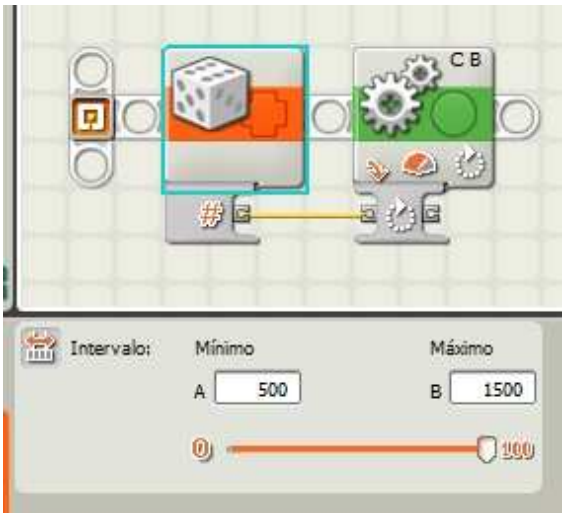
2. Girar un ángulo entre 90º y 270º con lo que se consigue el mismo resultado.

Para la primera solución sólo hace falta introducir un bloque de programación que genere un giro de 180º (1000º de los motores en base a lo que hemos calculado) antes del giro aleatorio. El programa quedaría así:



Pero si reflexionamos un poco podemos deducir que con la segunda el programa resultante va a ser más sencillo. Únicamente hay que modificar el nuevo bloque que hemos creado: **Giro aleatorio**. De este modo ya no hará falta que echemos a dados a qué lado ha de girar, será suficiente con que lo programemos para que gire entre 90º (500º motor) y 270º (1500º del motor). Para modificarlo hay que hacer un doble clic sobre el nuevo bloque **Giro aleatorio** para abrirlo, hacer los cambios que se deseen y guardarlo otra vez.

Con ello el bloque **Giro aleatorio** se habrá reducido a lo siguiente:

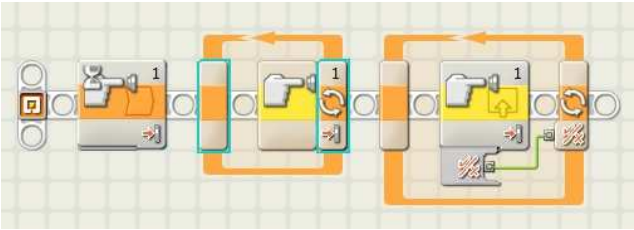


Reto básico 2

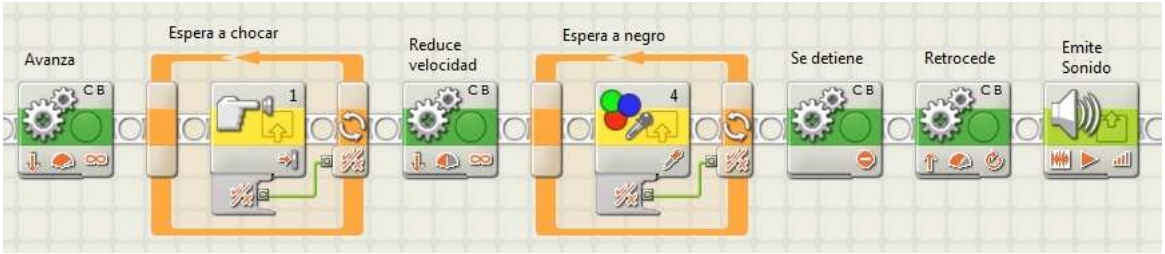
Vamos a ver el segundo, vamos a suponer como estado inicial que el robot está apuntando a la lata (para que no tenga que buscarla), lo cual permite que el algoritmo sea el siguiente:

1. Avanzar hasta que choque con la lata
2. Continuar avanzando hasta que alcance la línea oscura que limita el borde
3. Detenerse
4. Retroceder
5. Emitir un sonido

El acercamiento hasta que encuentra la lata lo he hecho de la misma manera que en el caso anterior, aunque cualquiera de los otros dos de la figura siguiente generarían idéntico comportamiento.



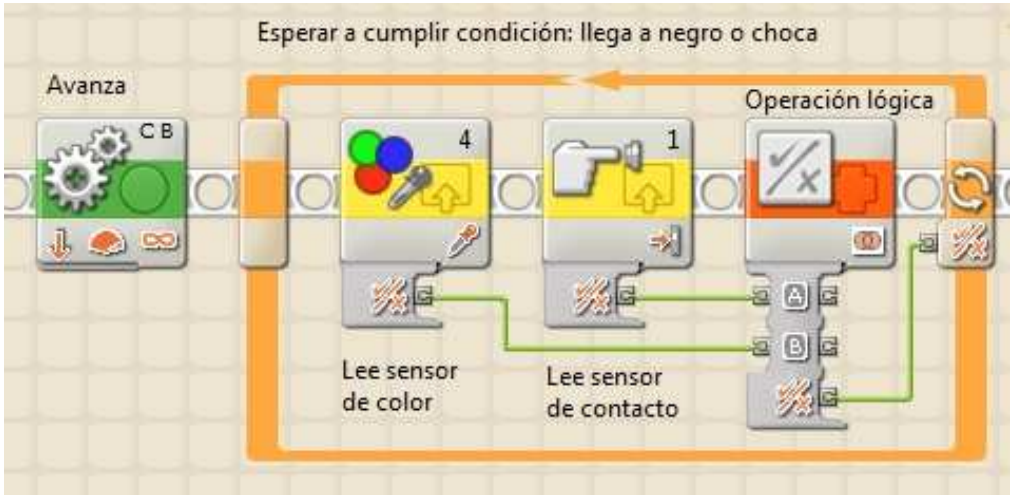
El bloque presente entre los dos bucles lo único que hace es reducir un poco la velocidad del robot una vez que la detecta.



Programa completo

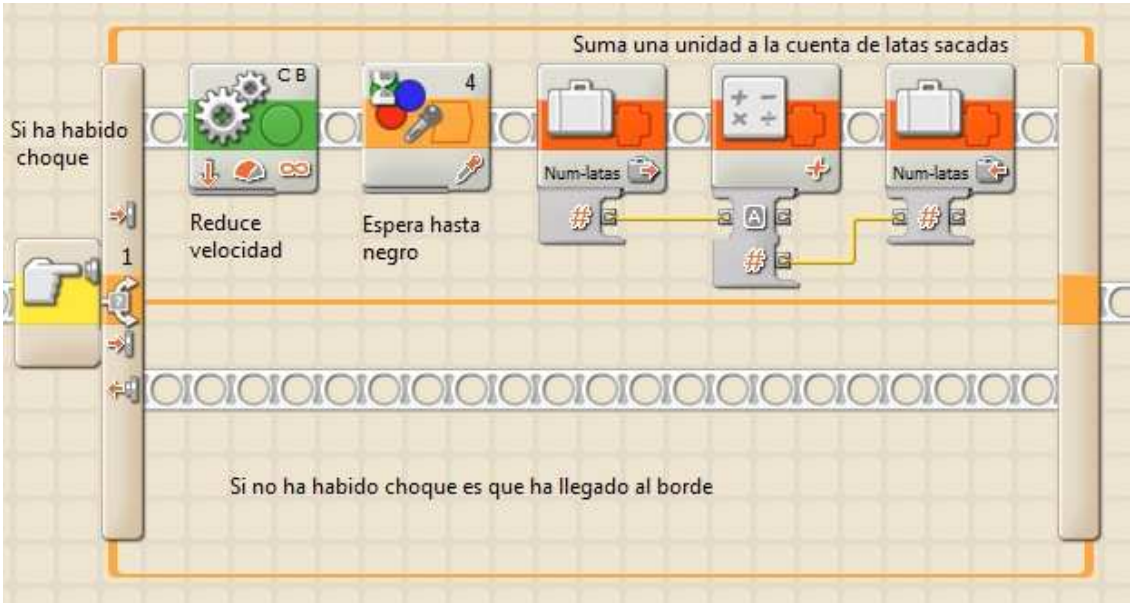
Si lo anterior va bien, ahora es el momento de combinarlo en un nuevo programa. Vamos a empezar creando una **variable** (representada por el bloque con aspecto de maletín) en la que podamos almacenar el número de latas sacadas del área de juego. La variable se crea por medio de la opción *Definir variable* del menú *Herramientas* y se inicializa con el valor cero de la siguiente manera:

Vamos a ver primero cómo se pueden monitorizar dos sensores a la vez y tomar decisiones en base a dichos valores (paso 2 del algoritmo).



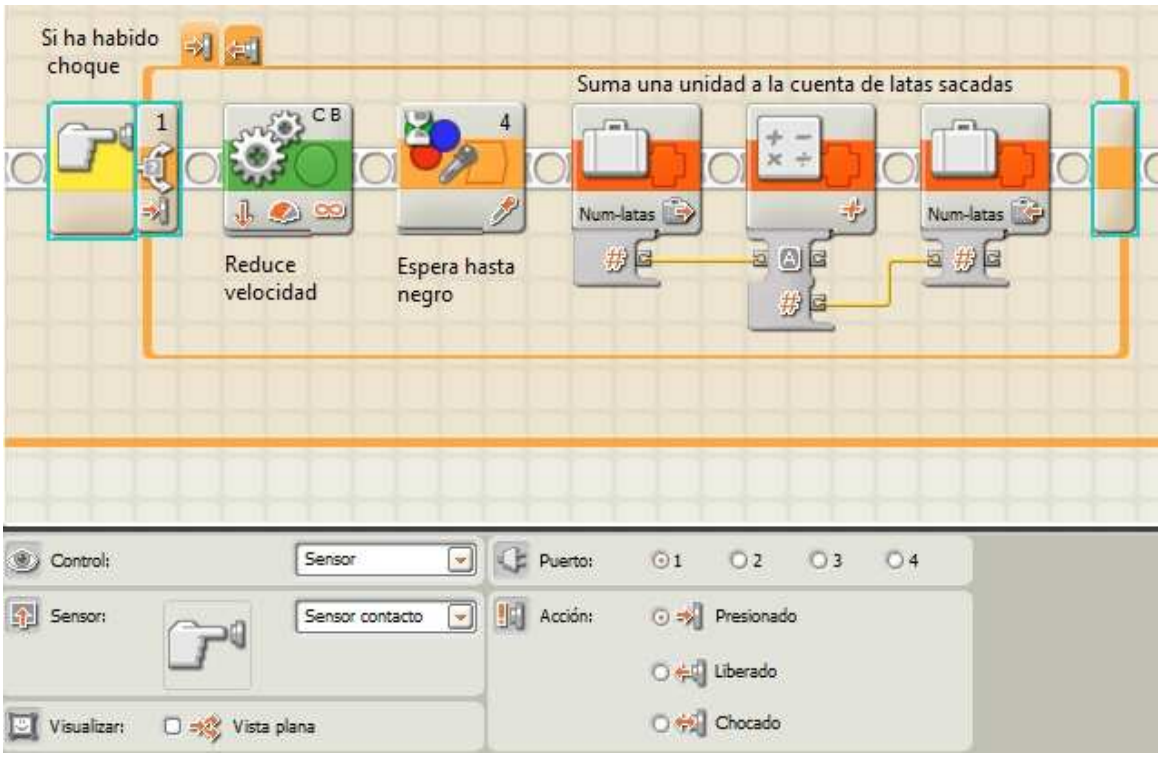
El fragmento de programa de la imagen comienza poniendo en marcha el robot antes de dar inicio a un bucle que se repetirá hasta que el sensor lea negro (llega al borde) o el sensor de contacto sea presionado (contacto con la lata). Para ello en el interior del bucle lee continuamente los dos sensores y por medio de una operación lógica (en este caso OR), combina dichas lecturas generando un valor lógico que será **Verdadero** si hay contacto con la lata, llega a la línea o si se cumplen las dos condiciones de modo simultaneo.

El robot ha de saber si ha llegado al borde o ha chocado con una lata, para lo que resulta útil un condicional que sólo se ejecutará en el caso en que haya habido contacto con la lata. En ese caso empujará hasta que llegue a la línea y sumará una lata a la cuenta (es el paso 3).



Si lo que había hecho era llegar al borde (o lo que es lo mismo, no ha chocado con la lata) evitará ejecutar esta secuencia de bloques.

¡TRUCO! : Si se desea que el programa se muestre de un modo más ligero (en el de la figura el ramal inferior no aporta nada), es posible hacerlo haciendo clic sobre Vista Plana del panel de configuración de la **Bifurcación**. Se puede ver el resultado en la figura siguiente.



Tras detener el robot, retroceder y girar (paso 4) hay que comprobar cuántas latas están ya fuera. Para ello comprueba el valor del contador que en el caso de que sea 2 finaliza el bucle y genera un sonido antes de finalizar el programa (paso 5).

6.-Probar el robot

Ahora ya podemos probar el robot y valorar los resultados, si no se corresponden a lo que esperábamos habremos de valorar si hemos de modificar el robot, el programa o los dos.

¡Recomendación! No conviene aplicar múltiples cambios a la vez, ya que no sabremos cuál de dichos cambios es el que ha causado la mejora o ha provocado que lo que iba bien haya dejado de hacerlo.

7.-Por dónde seguir

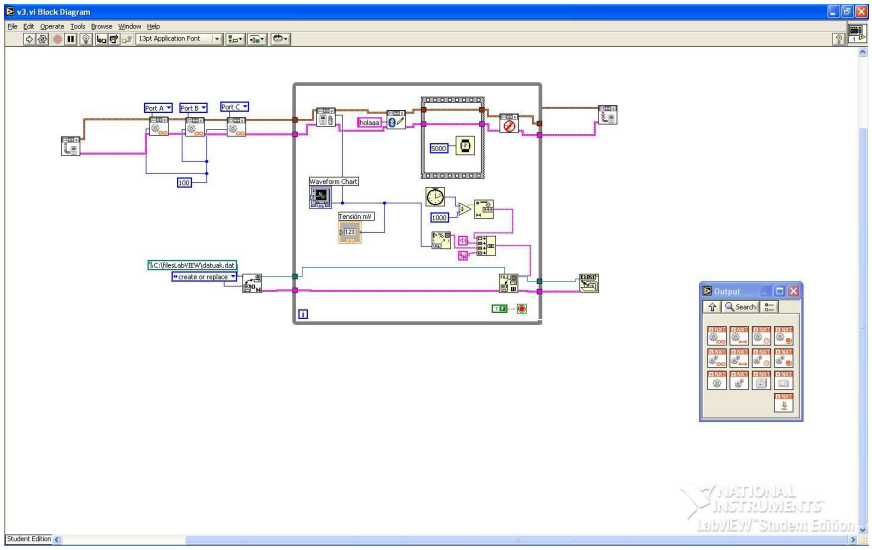
Si has llegado hasta aquí todavía le puedes sacar más jugo a este reto intentando incluir las siguientes modificaciones:

1. Tienes un sensor de ultrasonidos, así que puedes hacer que tu robot no se mueva a ciegas. Con este sensor podrá localizar las latas sin necesidad de tocarlas.

2. Vas a cambiar la lata por otro objeto más pesado, puede ser una lata de tomate de 800gr u otro objeto similar (o más pesada). Este ejercicio es interesante para acercarse a lo que es una competición de sumo entre robots, en este caso con un peso muerto.

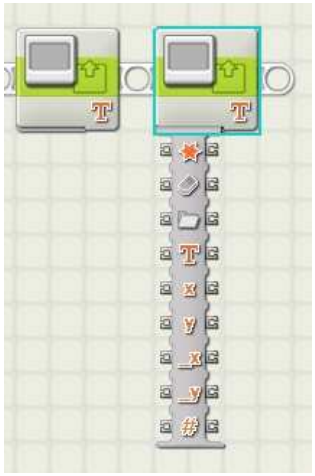
A.-Cables de datos

NXT-G está basado en LabView, un lenguaje de programación gráfico que utiliza "cables" para unir los diferentes bloques de programación y transportar datos.

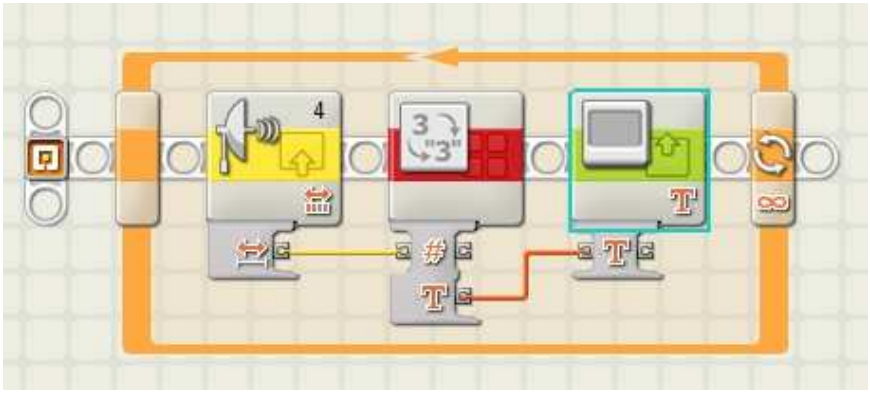


NXT-G también utiliza cables para transportar datos, aunque no son necesarios al iniciarse en su programación. Sin embargo, si se quiere avanzar en el dominio de NXT-G y desarrollar programas más complejos es necesario dominar el uso de cables de datos.

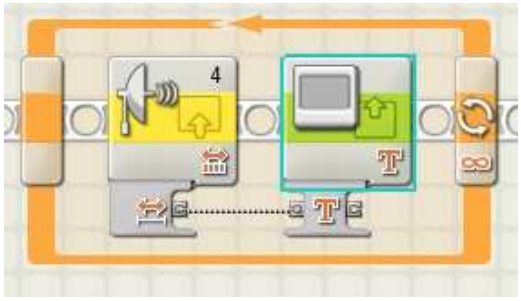
Para las conexiones hay que desplegar los concentradores de datos de los bloques de programación haciendo clic en su parte inferior. En la siguiente imagen puede verse el bloque Visualizar con el concentrador cerrado y abierto.



Un ejercicio básico de cableado es el convertir un número en texto para poder mostrarlo en el monitor del NXT, por ejemplo la lectura del sensor de ultrasonidos. El programa será el siguiente:



Los cables presentan diferentes colores en función del tipo de datos que transportan. El amarillo se corresponde a valores numéricos, el naranja a texto y el verde a valores lógicos. Si se realizan conexiones ilegales, las que unen conectores de diferentes tipos, el cable adquiere el aspecto de la siguiente figura. En dicho ejemplo se ha intentado unir una salida con valor numérico con una entrada que espera un texto.



B.-Monitorización de sensores

Cuando se programa con sensores en muchos casos lo significativo no es el valor que lee el sensor, sino saber si esa lectura cumple una condición, por ejemplo, si la distancia que mide el sensor de ultrasonidos es menor que 20 cm o no lo es.

En otros casos el valor de lectura sí será importante, por ejemplo si queremos que un robot vaya reduciendo su velocidad cuando se acerca a una pared. Aquí voy a centrarme en el primero de los casos.

Algunos ejemplos de condiciones

Veamos algunos ejemplos de condiciones de la vida diaria que pueden ayudar a entender lo que va a continuación:

- La temperatura en el balcón supera los 20º
- La luz del baño está apagada
- Ha sonado el timbre de la puerta
- Han pasado más de 5 minutos desde que la clase ha empezado
- Miguel ha llegado tarde

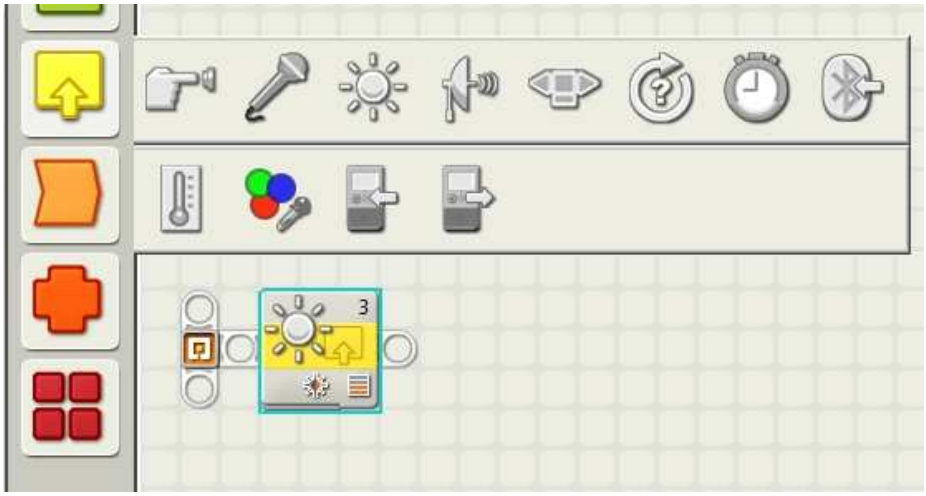
Todas estas expresiones se pueden cumplir o no, es decir tener como valor ***Verdadero*** o ***Falso***. En función de ese valor se procederá de una manera o de otra, por ejemplo:

Si "Ha sonado el timbre de la puerta" ***entonces*** "ábrela"

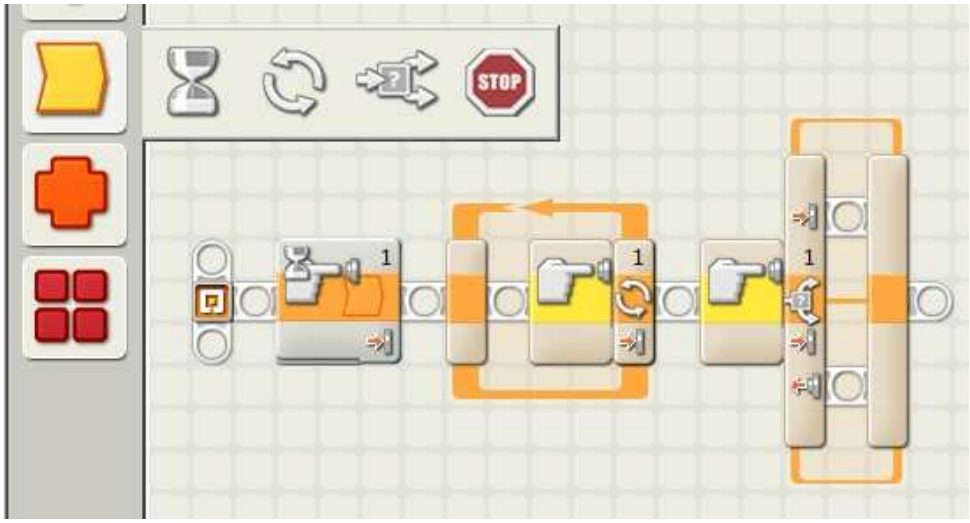
Dependiendo del valor de "Ha sonado el timbre de la puerta" habrá que abrirla o no.

Bloques de programación para sensores

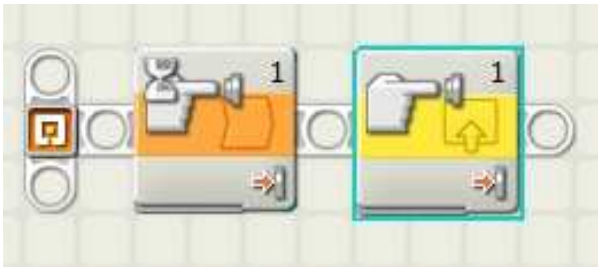
NXT-G dispone de un bloque de programación para cada sensor que permite acceder al valor de la lectura. En la figura siguiente puede verse el menú Sensor abierto con los bloques preinstalados en NXT-G Educación 2.1. Vía Por medio del menú ***Herramientas > Asistente de Importación y Exportación de bloques*** se pueden instalar los correspondientes a otros sensores.



Además de este, se pueden utilizar las lecturas de los sensores ligadas a los bloques que controlan el flujo del programa: ***Espera***, ***Bucle*** y ***Bifurcación***, que pueden verse en la siguiente imagen asociados a un sensor de contacto.



¡¡Atención!!: Una fuente de errores suele ser el confundir el uso del bloque que detiene el flujo del programa hasta que la lectura del sensor es la deseada (menú ***Flujo*** con franja naranja). y el que proporciona la lectura de un sensor (menú ***Sensor***, con franja amarilla) En el segundo caso el programa lee la lectura del sensor y sigue adelante con el siguiente bloque. En la figura se puede apreciar la diferencia entre los dos en el caso del sensor de contacto.



Algebra de Boole

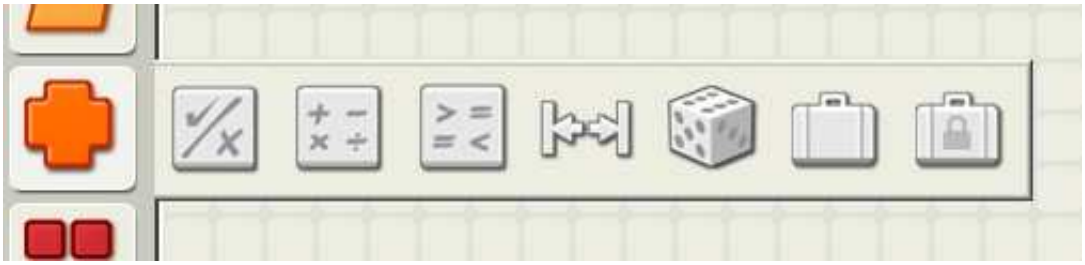
Las funciones del álgebra de Boole son muy útiles en programación y NXT-G permite utilizar varias de ellas. Estas funciones permiten operar con valores lógicos, es decir, con valores tales como Verdadero o Falso (estos valores también pueden ser representados como 1 ó 0).

Los resultados de las condiciones mencionadas anteriormente (por ejemplo, la temperatura en el balcón ha superado 220º) son valores lógicos. Si se utiliza una sola condición, o un solo sensor, se puede programar sin recurrir a estas funciones, pero cuando se combinan dos o más condiciones, como en este reto, las funciones del álgebra de Boole son muy útiles.

En los cuatro operadores que se presentan a continuación, se indica cómo se calcula el resultado y se presenta la correspondiente tabla la verdad. Una tabla de verdad es algo similar a lo que en aritmética es la tabla de multiplicar, esta tabla nos da el resultado de aplicar estos operadores para distintos valores de entrada.

Bloques de programación

Los bloques a utilizar con valores lógicos se encuentran en el menú ***Datos*** que puede verse desplegado en la siguiente figura. En los siguientes ejemplos se utilizará el primero de ellos, denominado ***Lógica***.

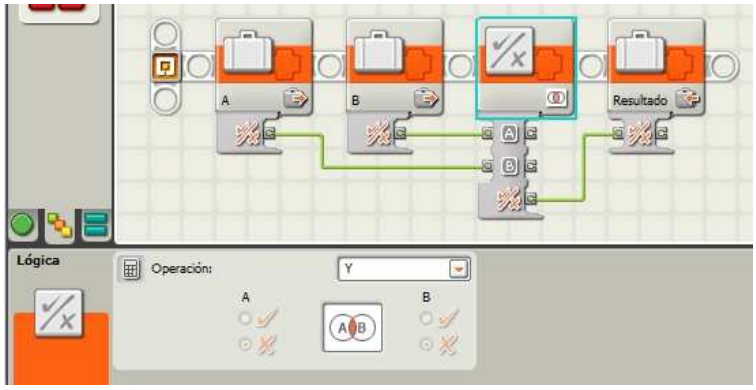


Operador Y (AND)

El operador ***Y*** relaciona dos entradas. Para que el resultado sea ***Verdadero*** las dos entradas deberán tener el valor ***Verdadero***. Si cualquiera de las dos es ***Falso*** la salida también será ***Falso***. La tabla de verdad de este operador es la siguiente:

A	B	Resultado
V	V	V
V	F	F
F	V	F
F	F	F

En la imagen siguiente puede verse el modo en el que se utiliza este operador. Los maletines representan variables que contienen valores lógicos.

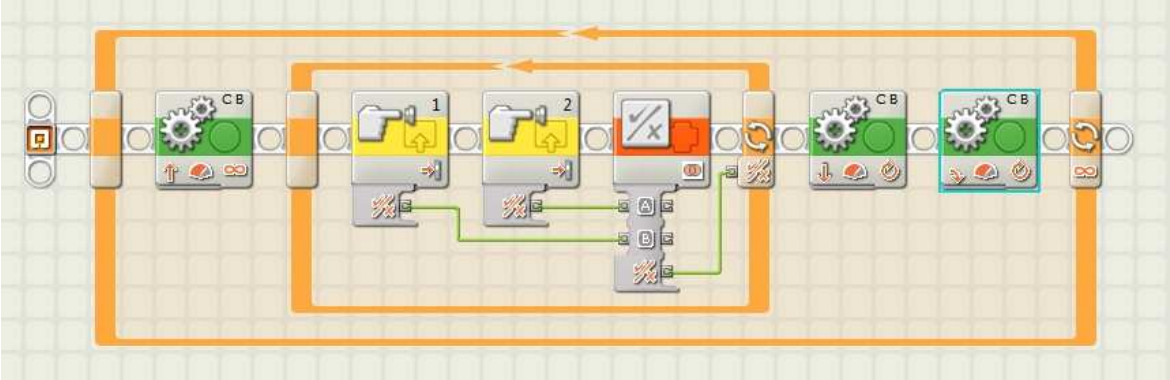


Operador O

Este operador relaciona dos entradas, la salida es ***Verdadero*** en el caso que al menos una de ellas sea ***Verdadero***.

A	B	Resultado
V	V	V
V	F	V
F	V	V
F	F	F

Un ejemplo de uso de este operador es el caso en el que un robot tiene dos sensores de contacto en la parte frontal uno en cada lado. Utilizando este operador el robot podrá controlar los dos sensores a la vez tal y como se muestra en el siguiente programa.



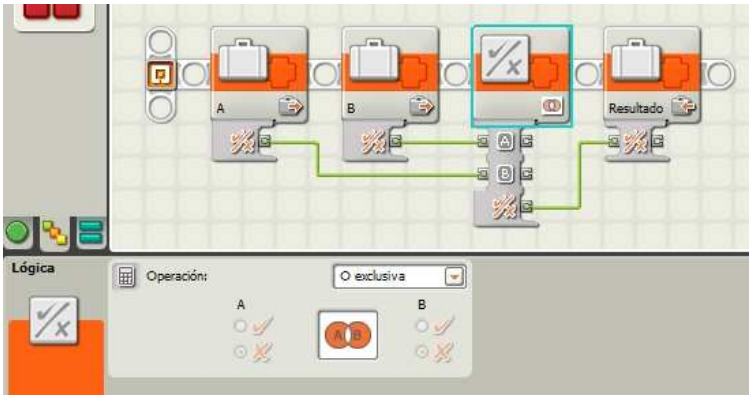
El robot se pone en marcha y lee continuamente los dos sensores, combina sus lecturas por medio de la función O y repite este proceso hasta que la función devuelve un Verdadero.

Operador O exclusivo

En este caso para que el resultado sea ***Verdadero*** una y solo una de las condiciones ha de ser ***Verdadero***, en el resto de los casos el resultado será ***Falso***.

A	B	Resultado
V	V	F
V	F	V
F	V	V
F	F	F

En la imagen siguiente puede verse el modo en el que se utiliza este operador. Los maletines representan variables que contienen valores lógicos.



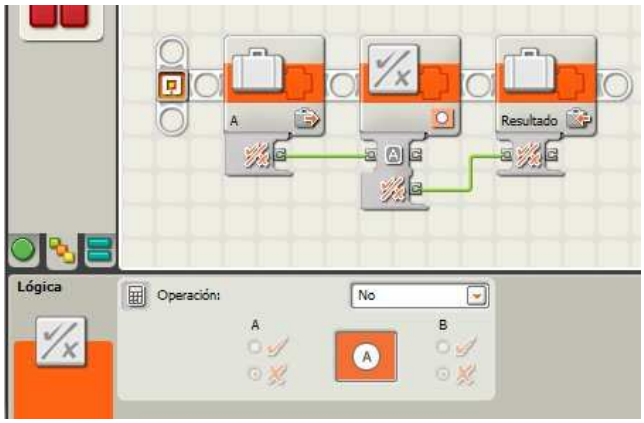
Operador NO

El operador NO da como resultado el valor opuesto al de entrada, es decir, si la entrada es ***Verdadero***, la salida será ***Falso***.

La tabla de verdad de este operador será la siguiente:

A	Resultado
V	F
F	V

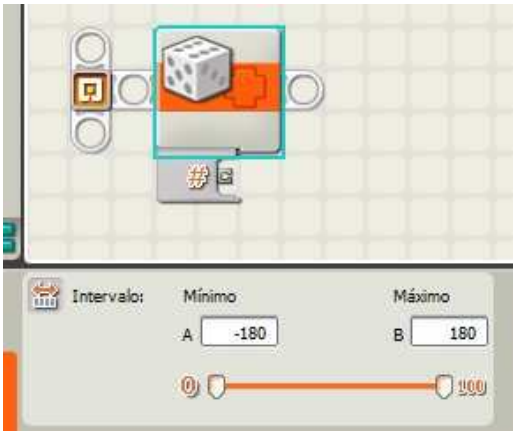
En la siguiente imagen puede verse un ejemplo con NXT-G.



C.-Números aleatorios

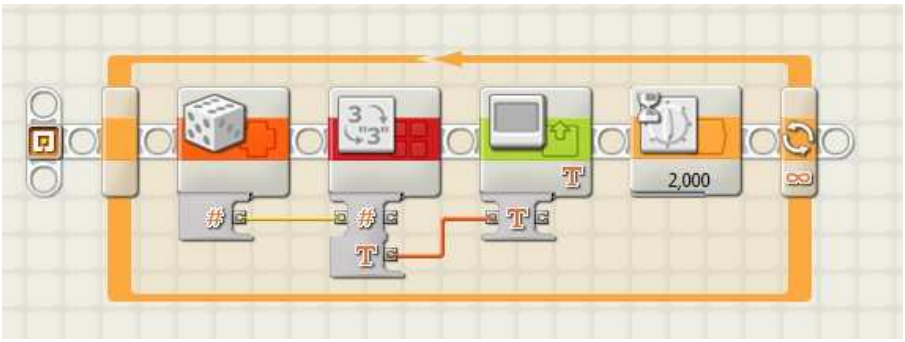
Obtener un valor aleatorio es como echar a suertes a ver cuál sale, de la misma manera que se hace con un dado o una moneda. Utilizarlos permite que el robot no sea tan previsible en su comportamiento. De este modo es posible que gire a un lado u otro, que gire más o menos…

Para generar valores aleatorios se utiliza el bloque de programación ***Aleatorio*** (Random en inglés) del menú ***Datos***.

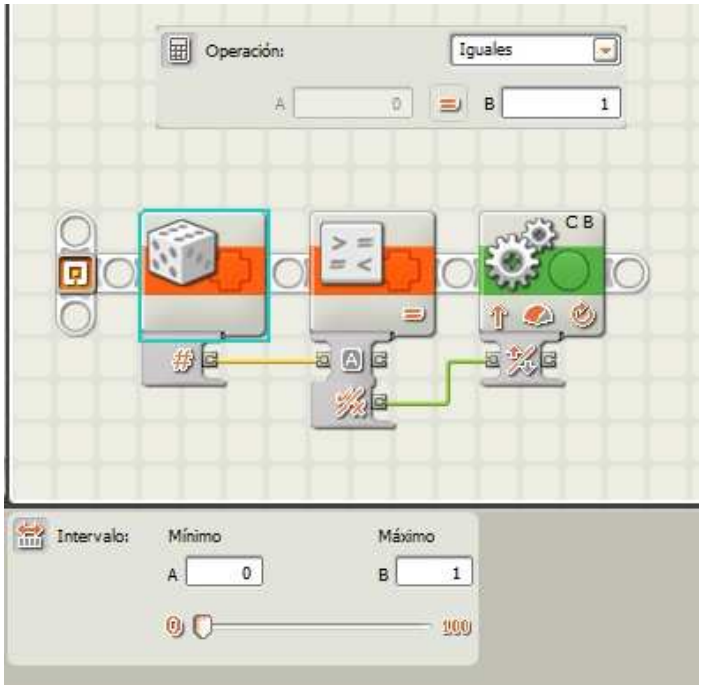


Este bloque devuelve un número aleatorio comprendido entre los valores que se introduzcan en su panel, en el ejemplo de la figura un número comprendido entre -180 y 180.

El siguiente programa permite probar este bloque y conocer su funcionamiento. En este ejemplo el bloque ***Aleatorio*** devuelve un número entre -180 y 180 que se muestra en la pantalla del NXT. Dado que la pantalla del NXT solo acepta texto y no números (cantidades), el programa convierte el valor devuelto por el bloque ***Aleatorio*** en una cadena de caracteres antes de mostrarlo. El bloque de espera de dos segundos facilita la lectura de los resultados que se irán generando hasta que se detenga el programa.

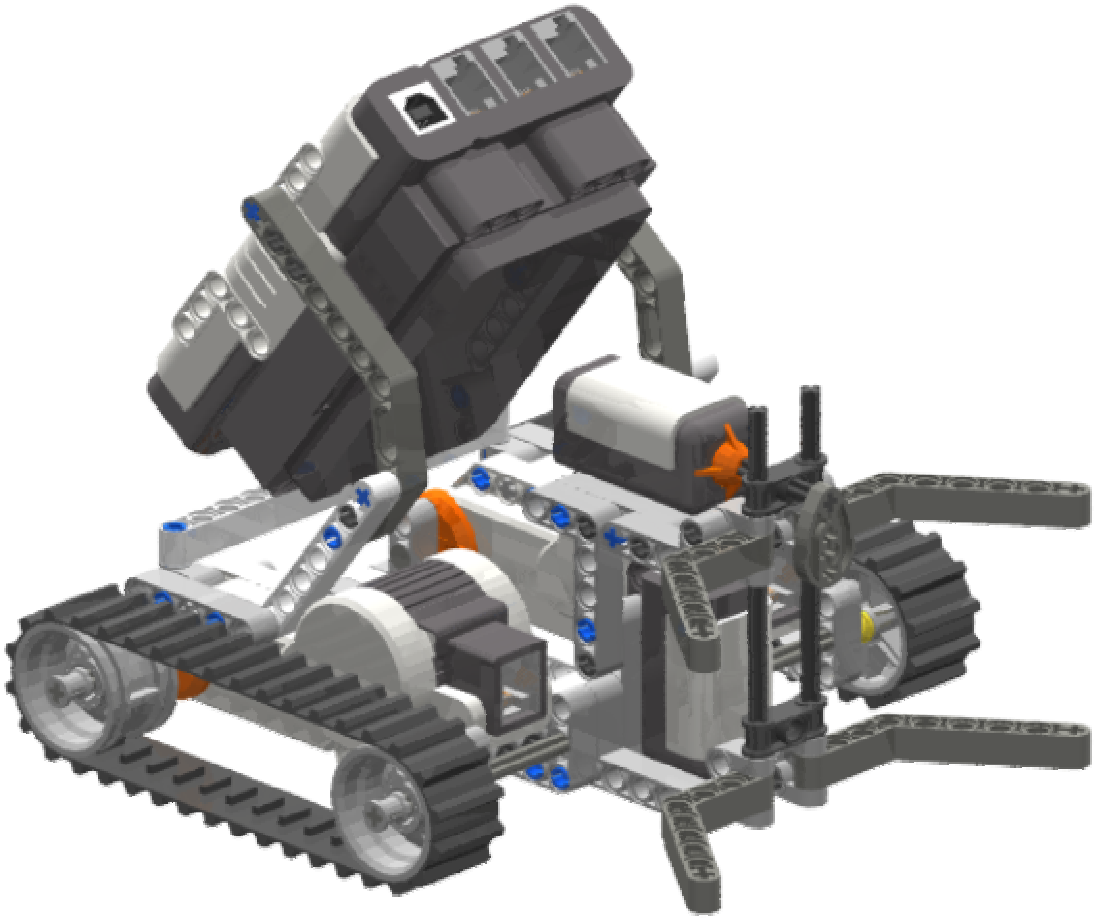


Si lo que se desea es generar un valor lógico, es decir, ***Falso*** o ***Verdadero***, lo que habrá que hacer es generar primero un valor aleatorio entre 0 y 1, y convertirlo después en un Falso o Verdadero utilizando el bloque ***Comparar*** del menú ***Datos***. Esto es lo que hay que hacer si se quiere echar a suertes hacia qué lado queremos que gire un robot tal y como puede verse en el programa de la siguiente figura. La razón es que el bloque ***Mover*** requiere de un ***Verdadero*** para moverse hacia adelante y un ***Falso*** para moverse hacia atrás

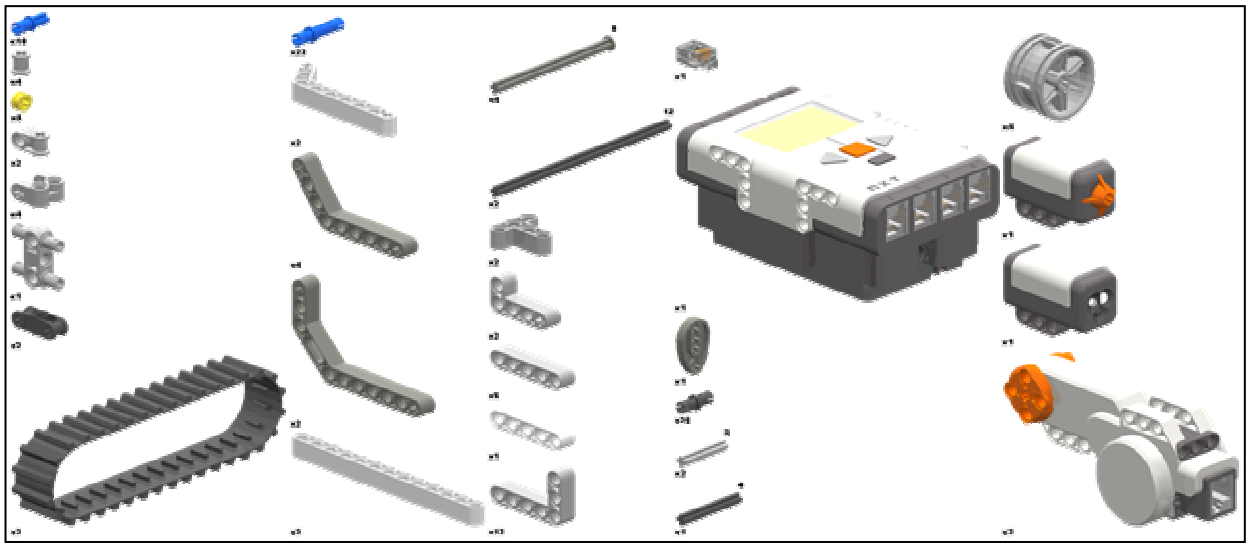


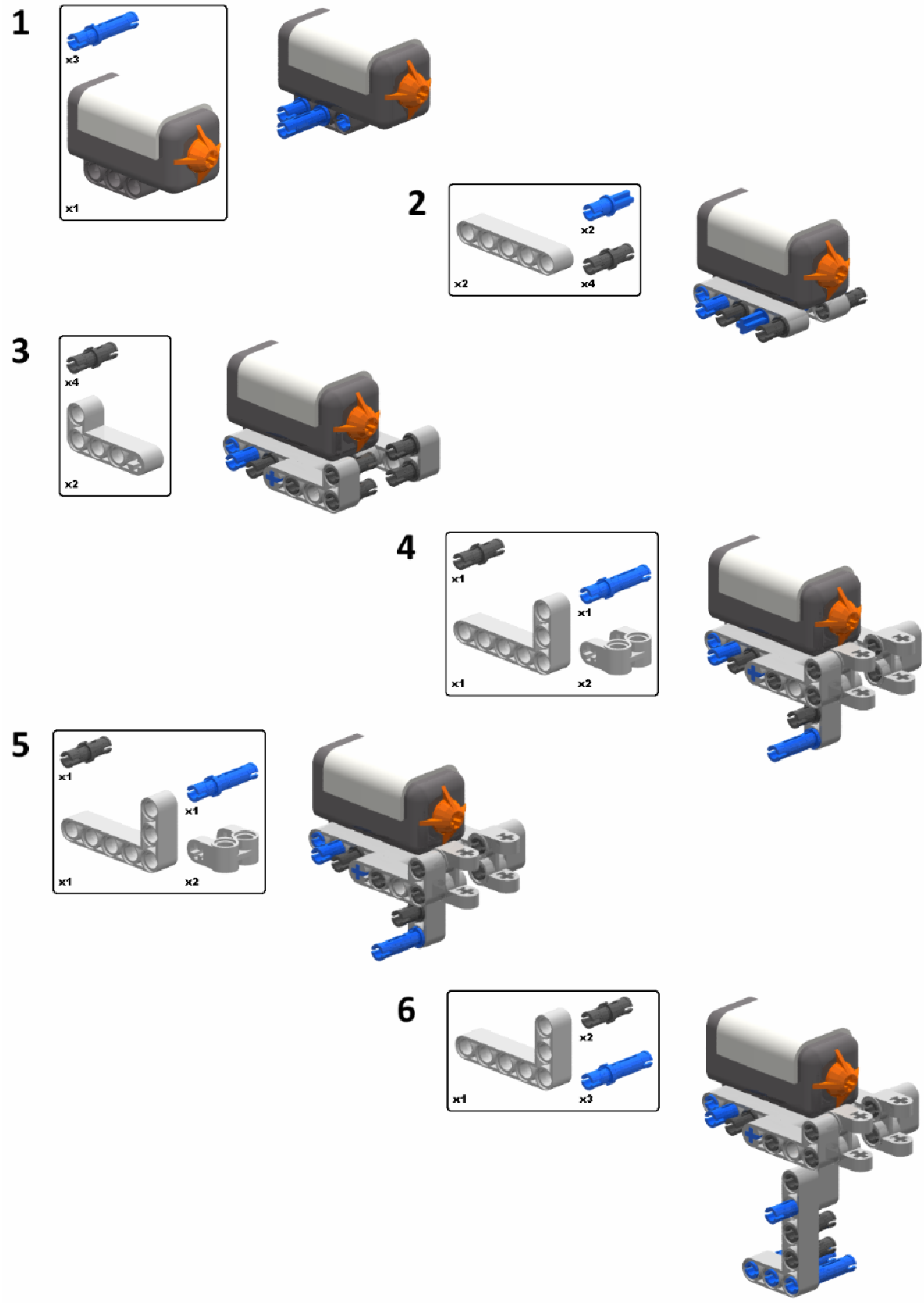
En la figura se puede ver en la parte superior el panel de configuración del bloque ***Comparar***. Compara el número aleatorio con 1 con el operador ***Iguales*** de tal modo que convertirá el 1 en ***Verdadero*** (movimiento de avance) y el 0 en ***Falso*** (retroceso).

Instrucciones de montaje

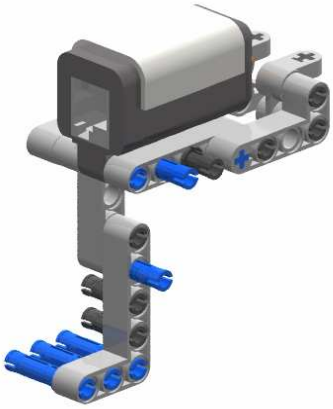


Piezas necesarias

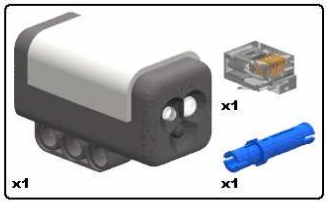


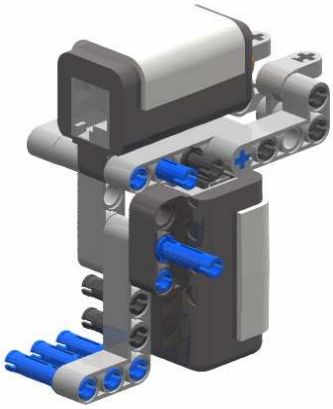

1
x3
x1
2
x2
x2
x4
3
x4
x2
4
x1
x1
x1
x2
5
x1
x1
x1
x2
6
x2
x1
x3

GIRAR

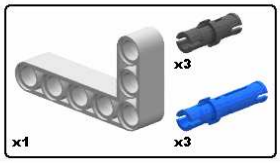


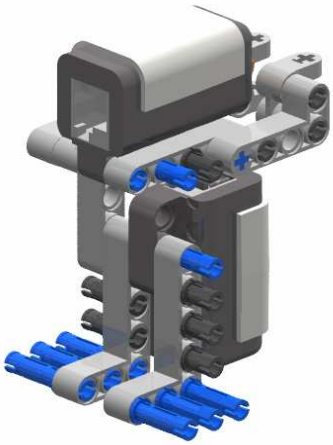
7



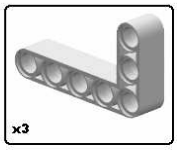


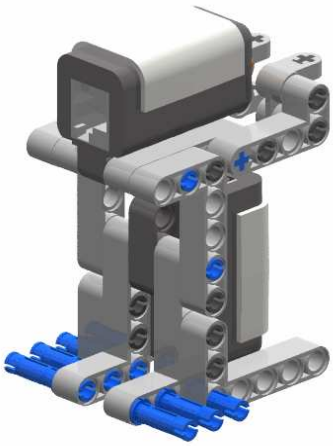
8



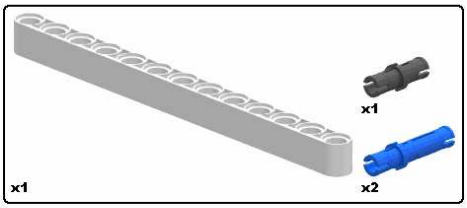


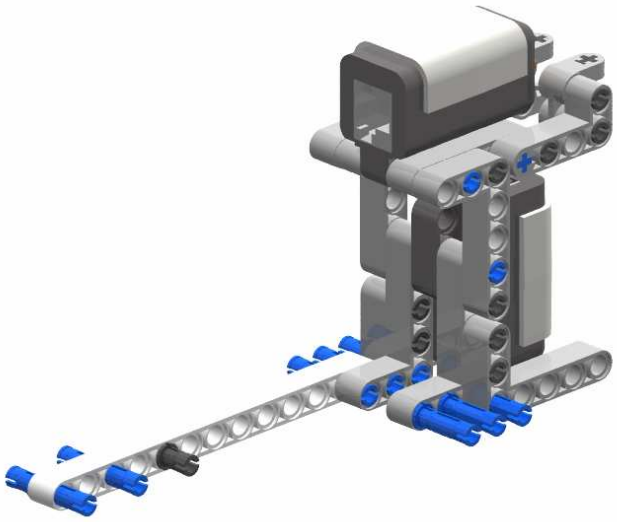
9



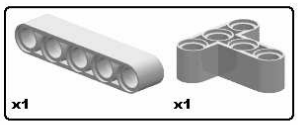


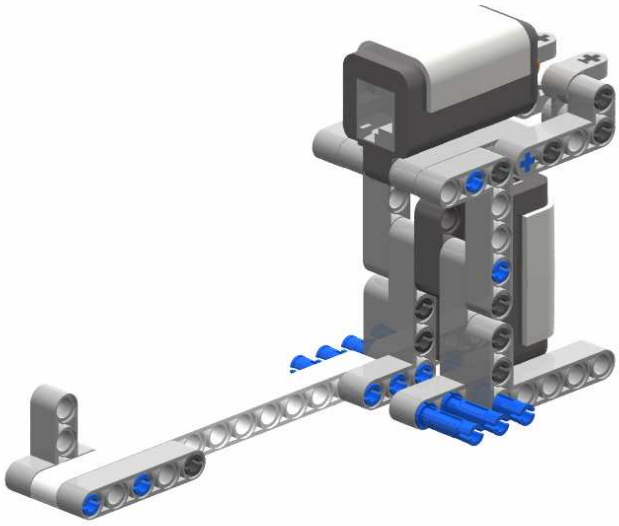
10



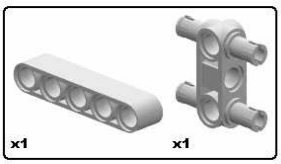


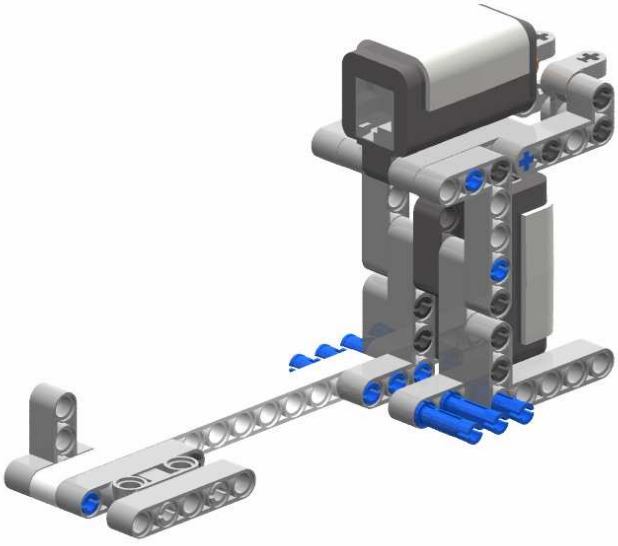
11



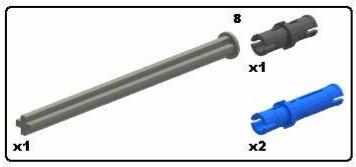


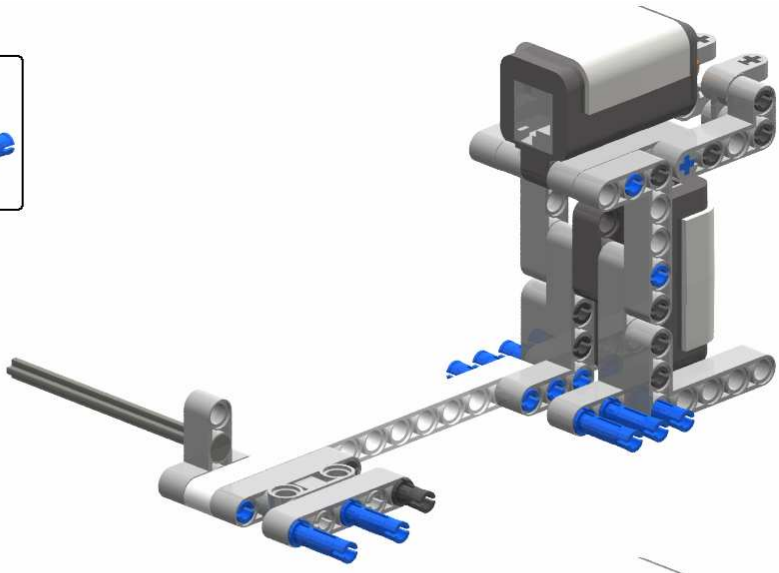
12



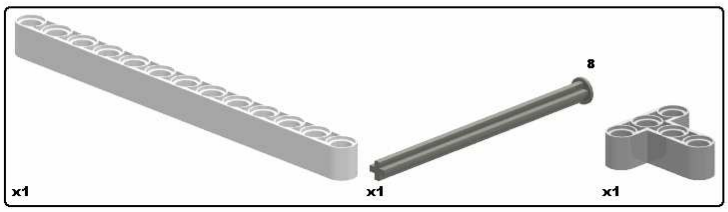


13

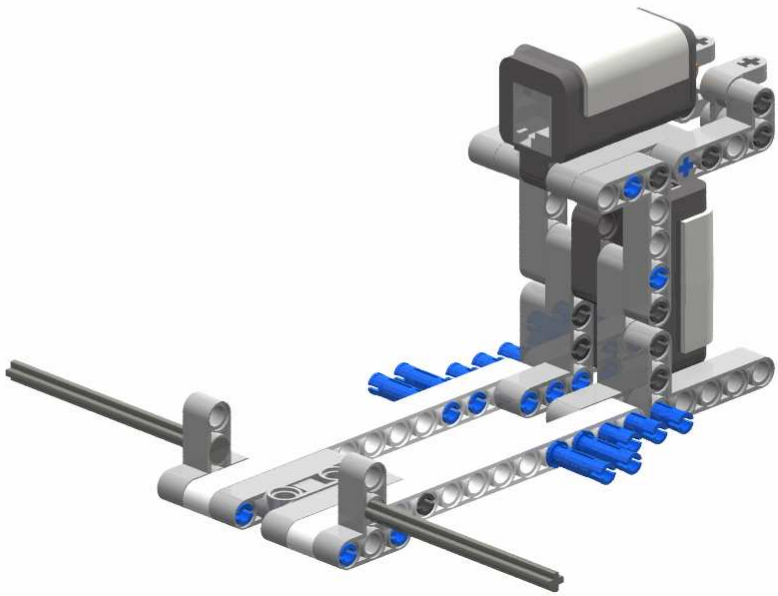




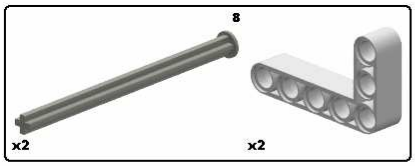
14

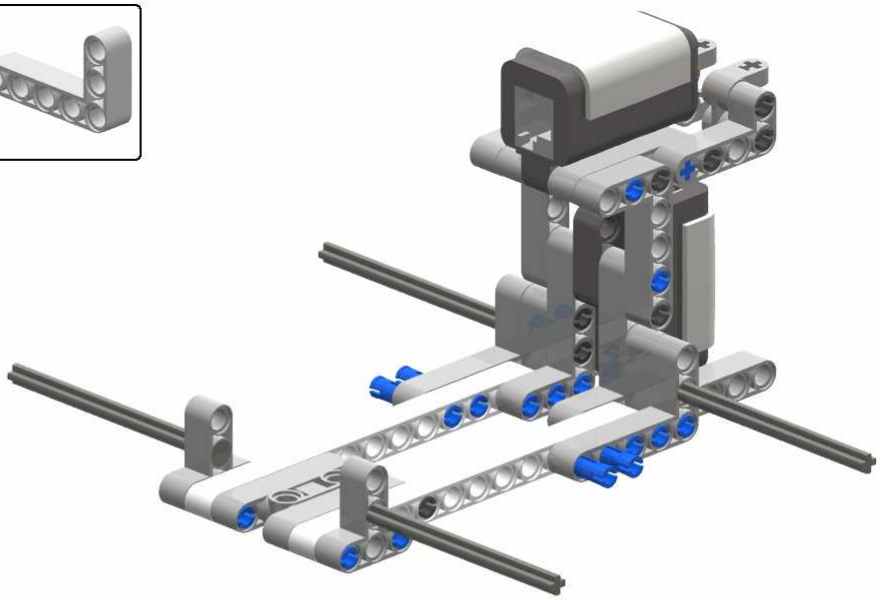


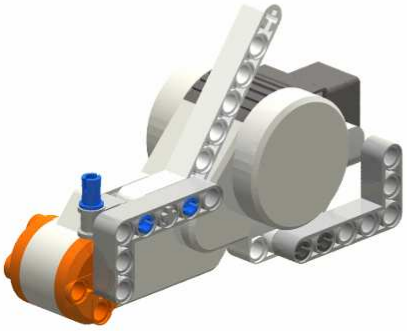
15



16

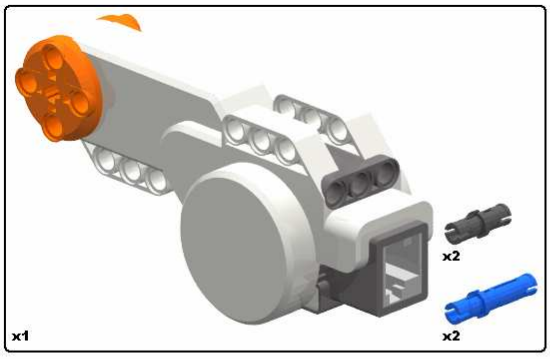


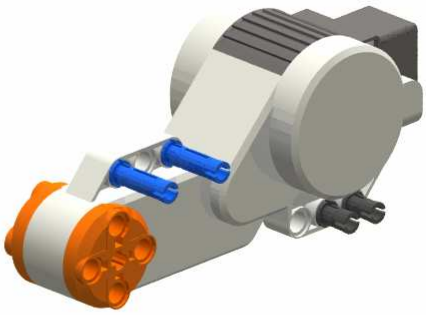




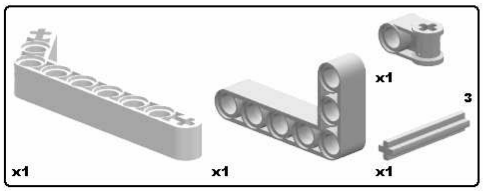
Lateral Derecho

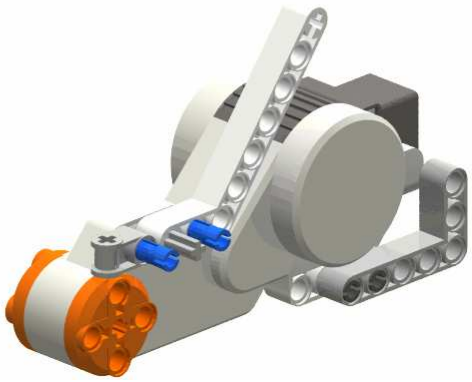
A



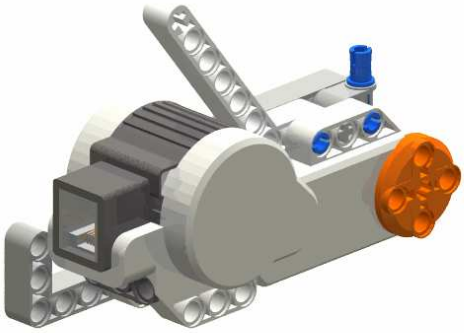


B

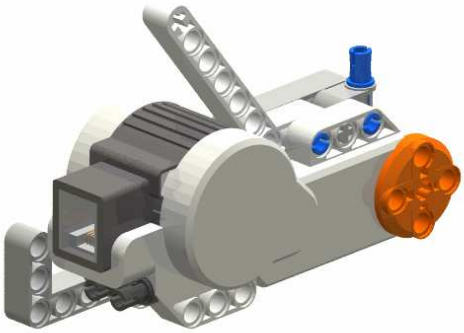




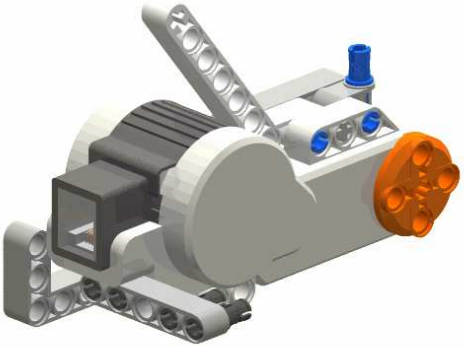
GIRAR



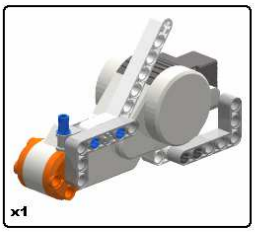
C

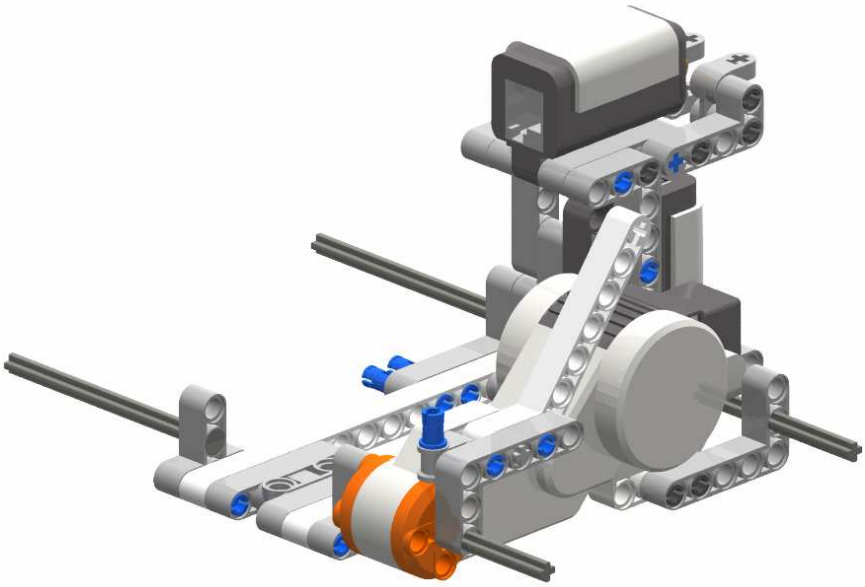


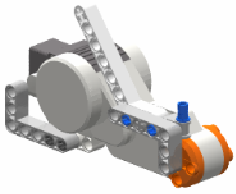
D



17

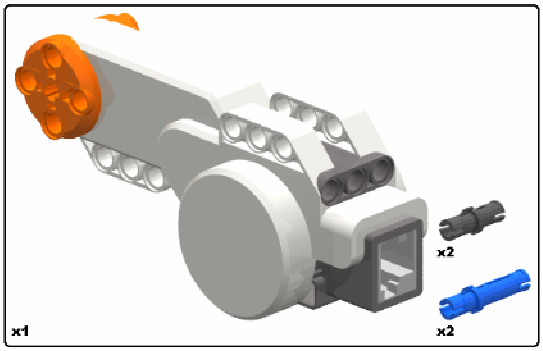


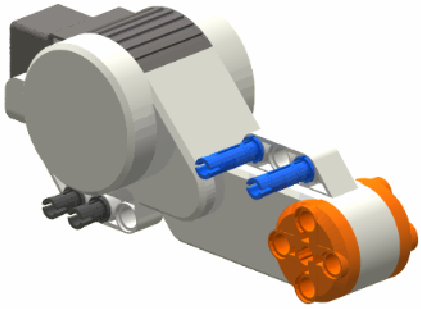




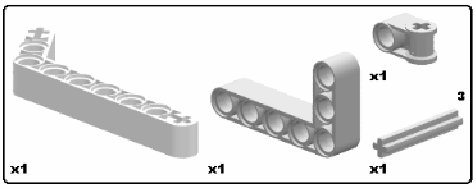
Lateral Izquierdo

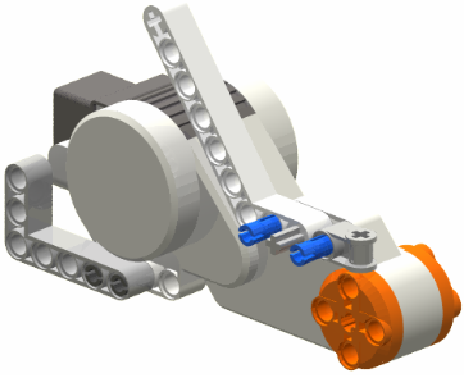
A



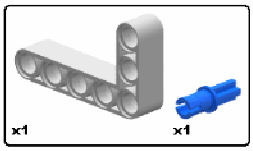


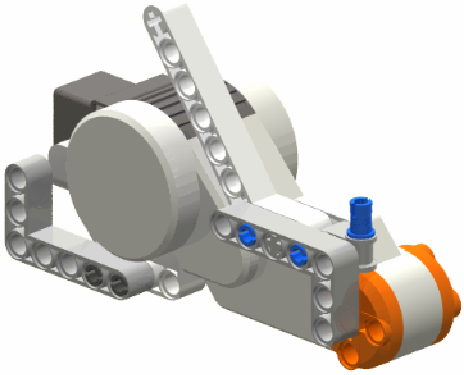
B

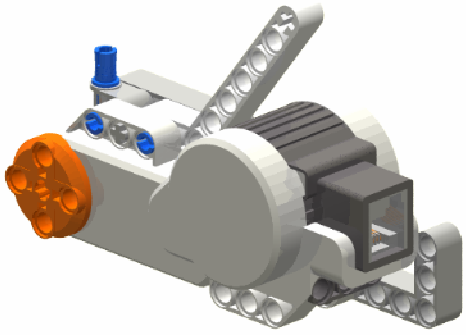




C

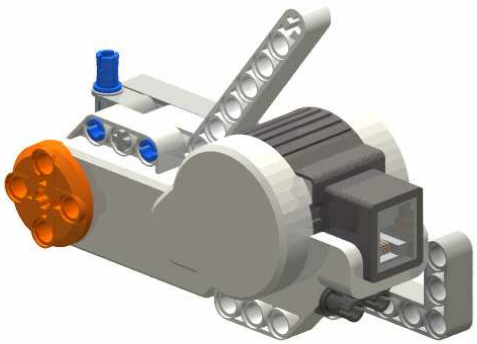




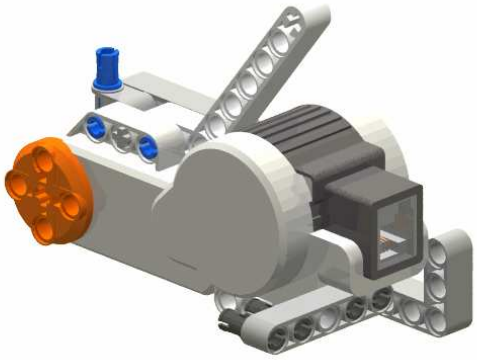


GIRAR

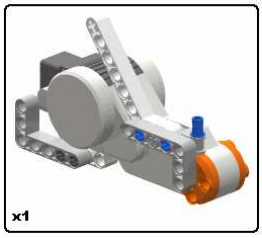
D

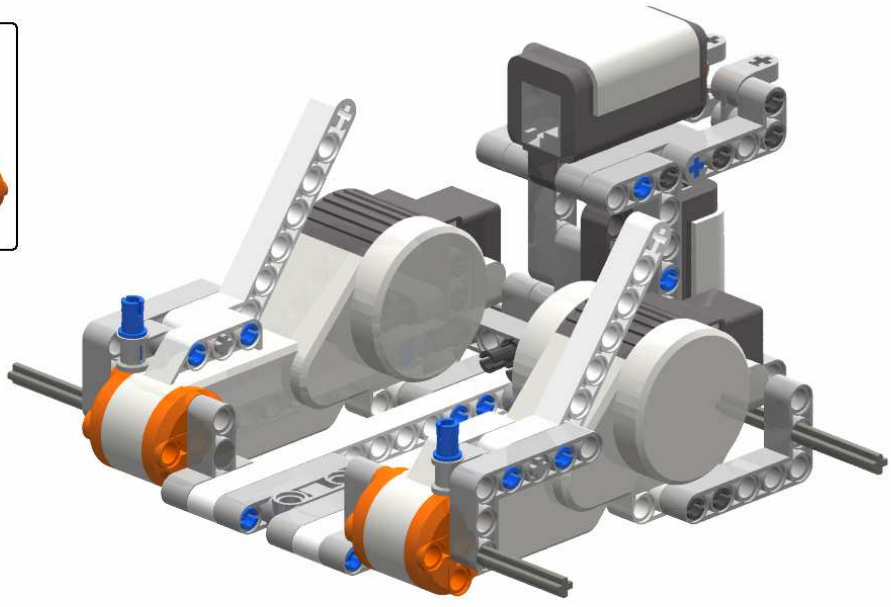


E

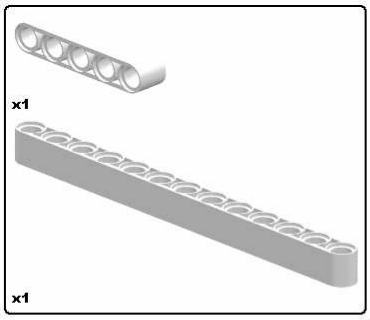


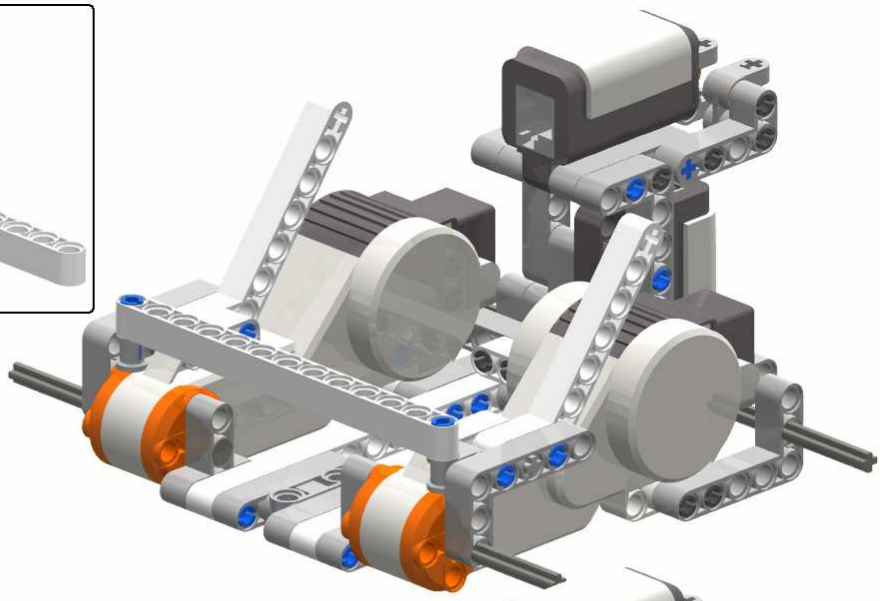
18



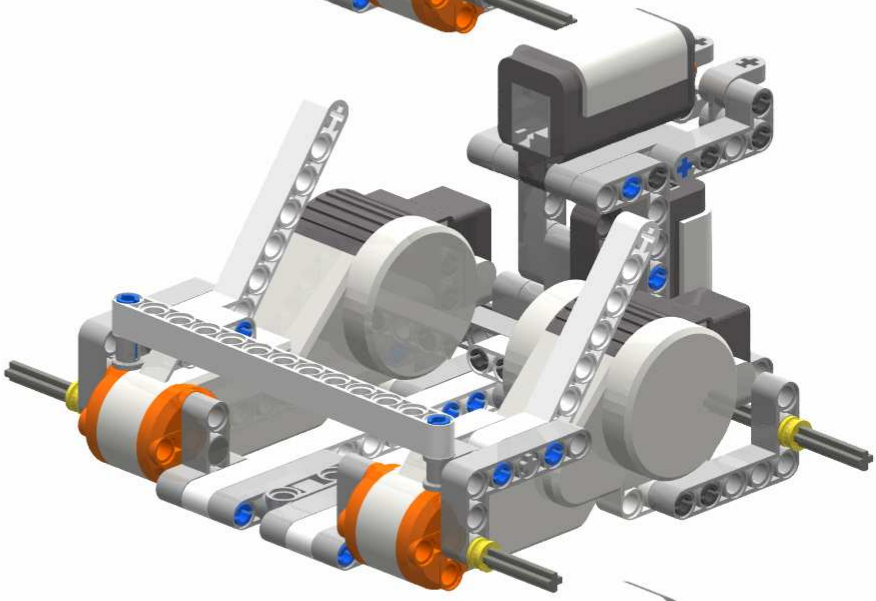


19

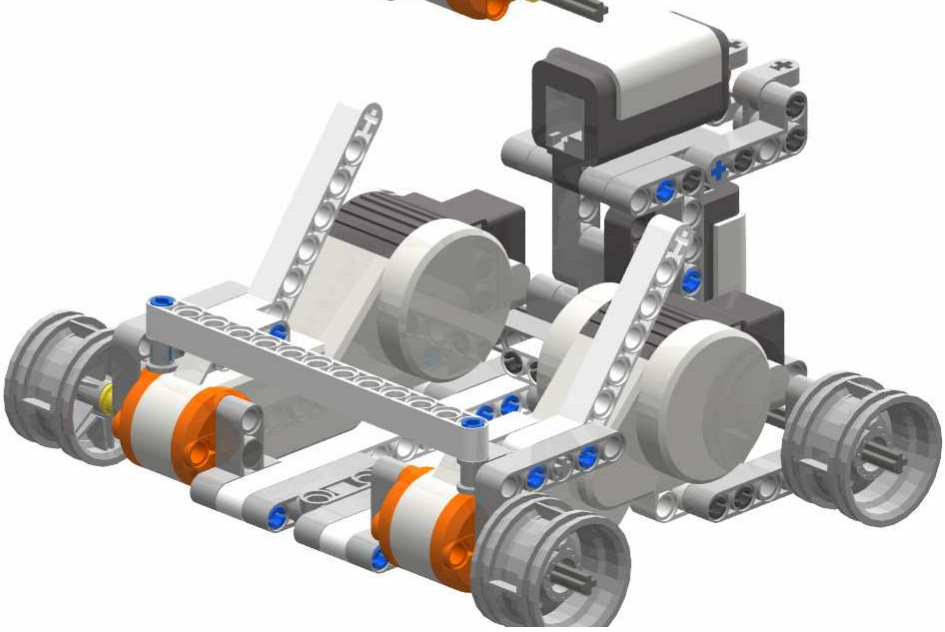




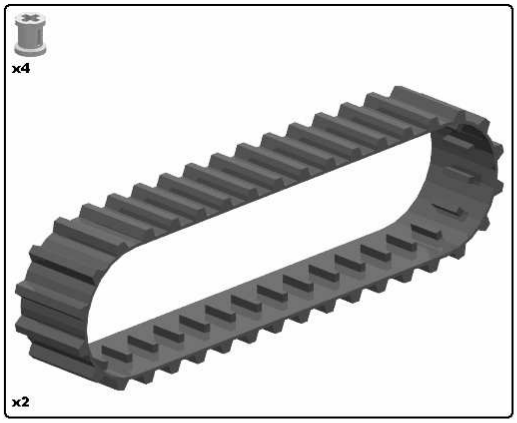
20

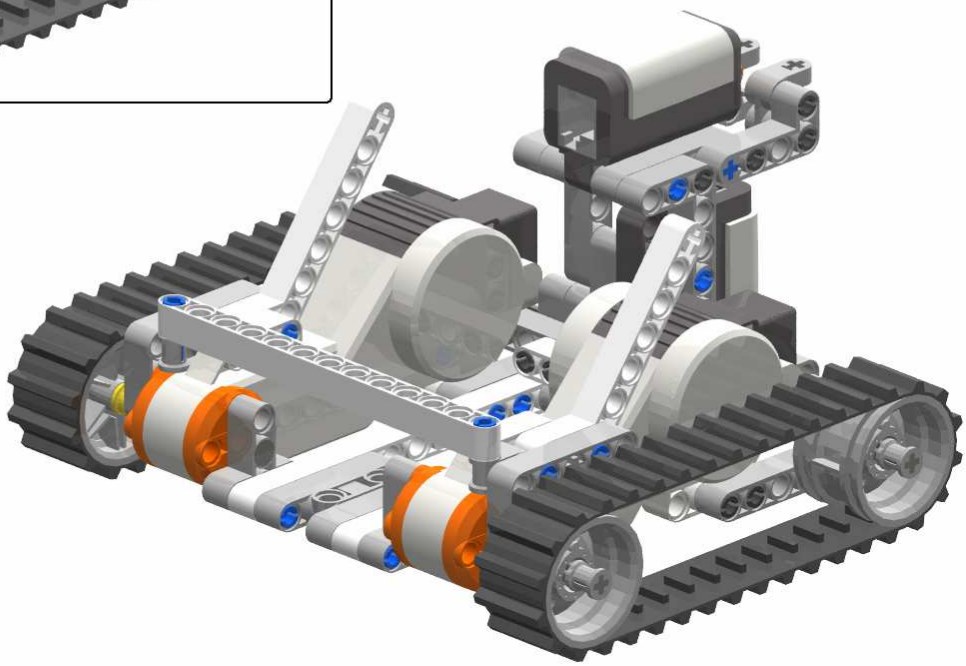


21

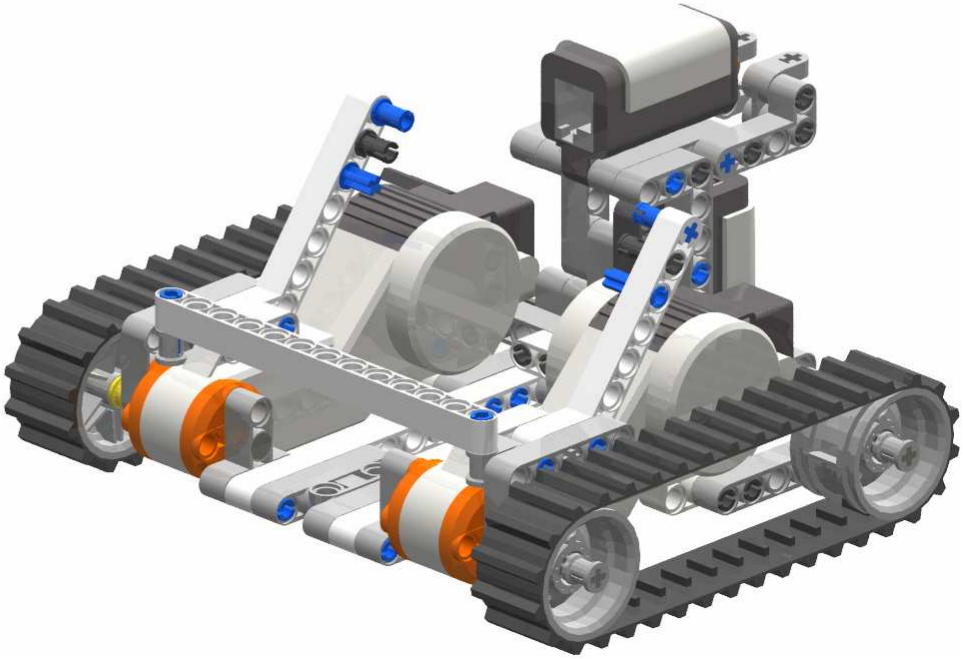


22

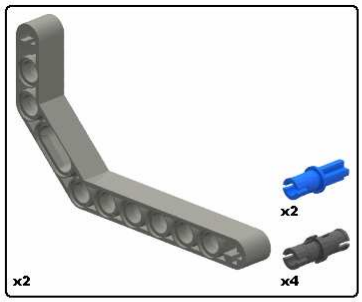


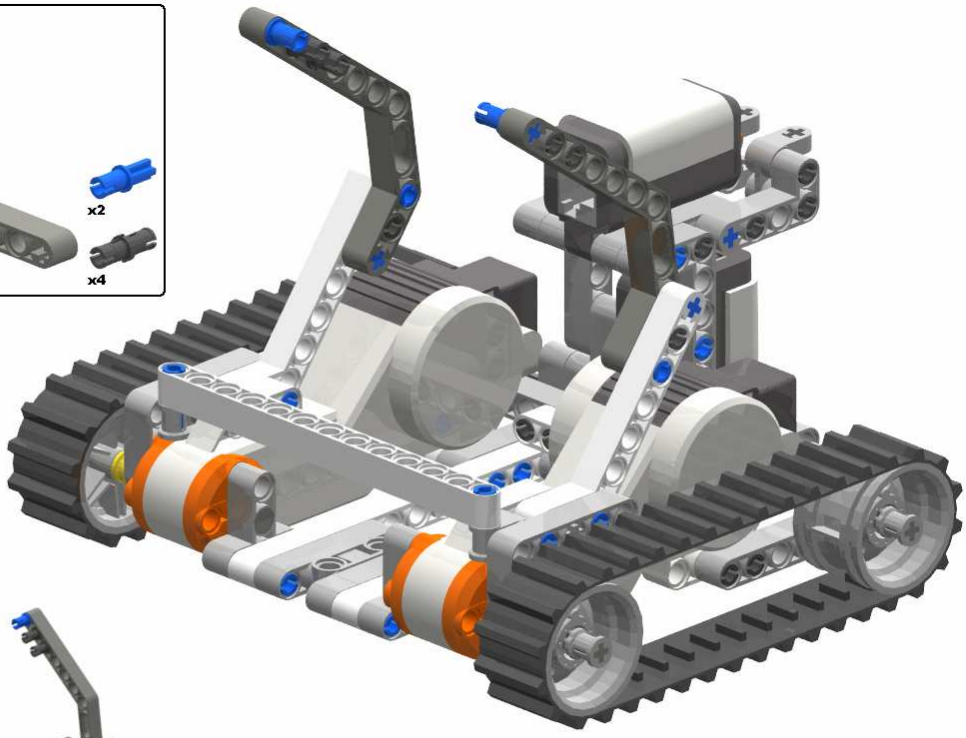


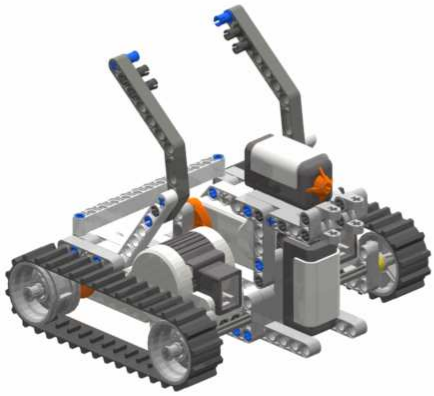
23



24

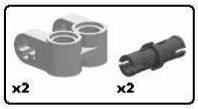


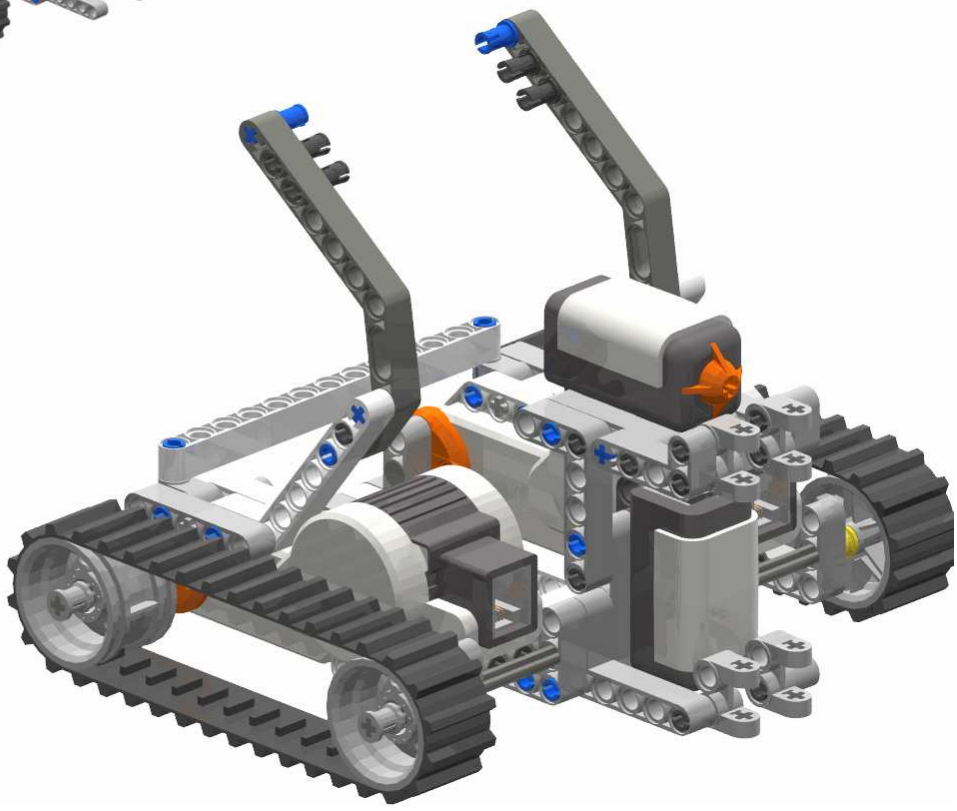




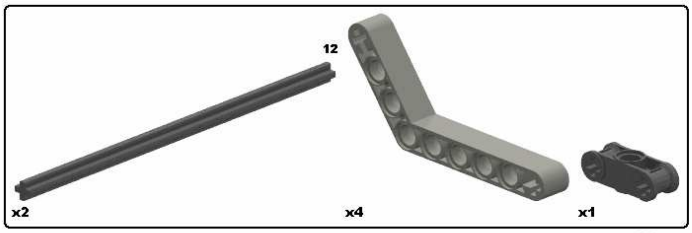
GIRAR

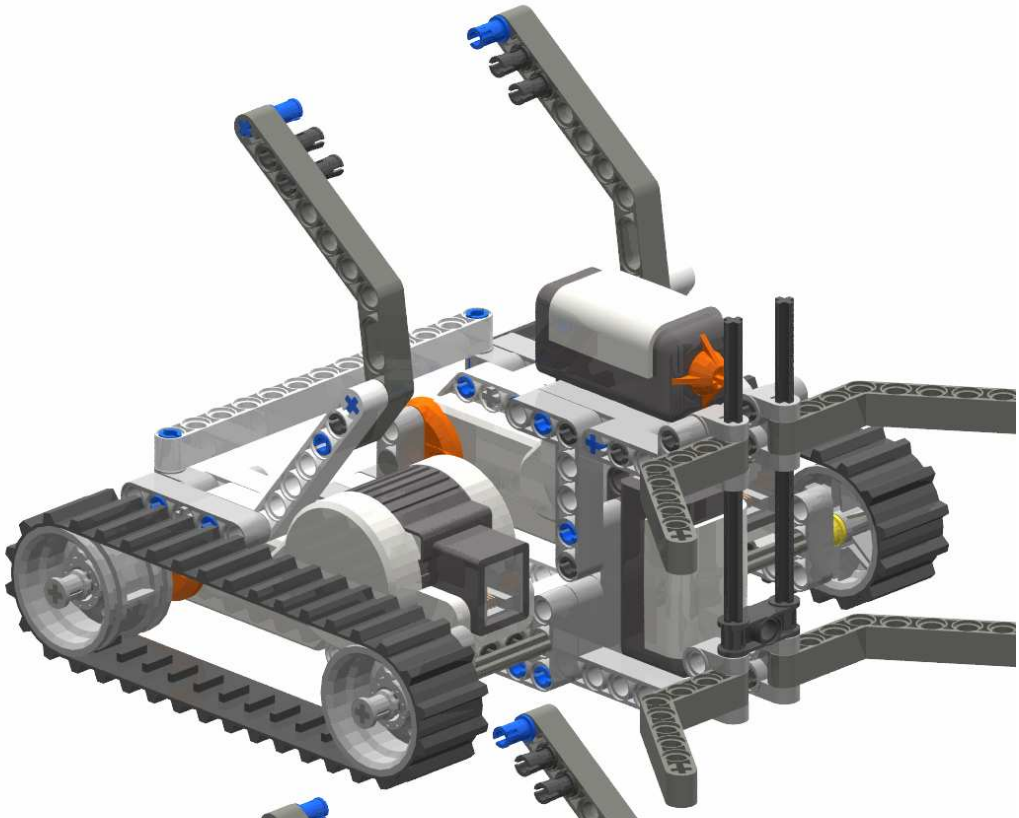
25



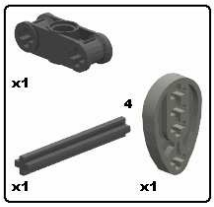


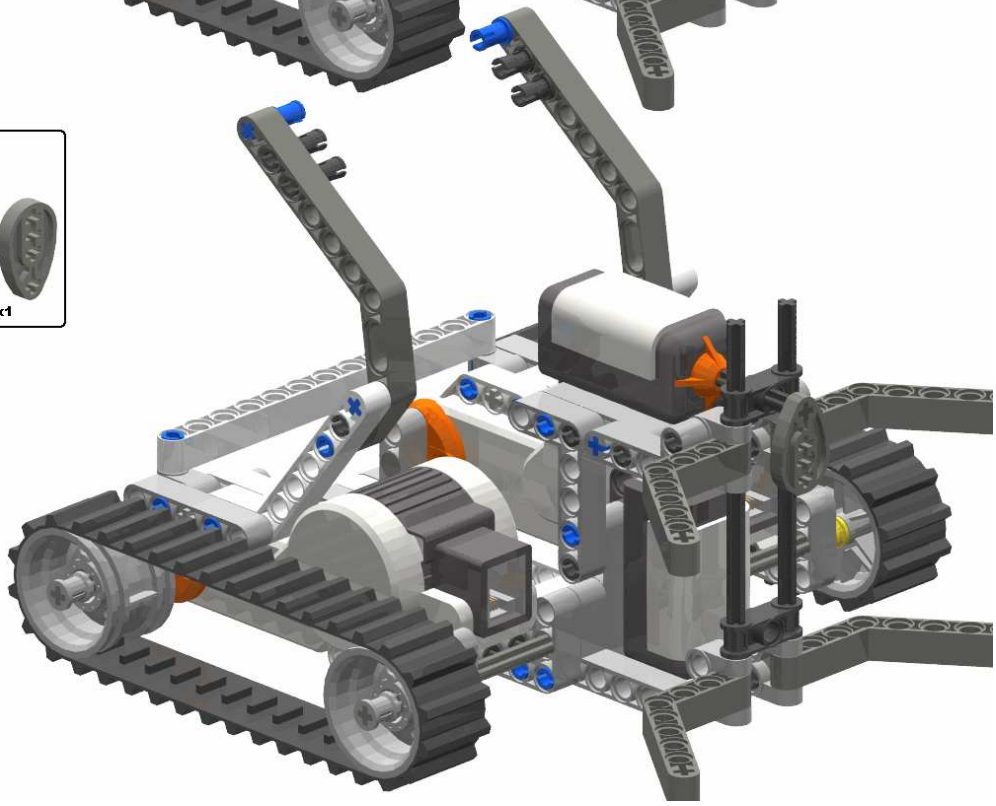
26



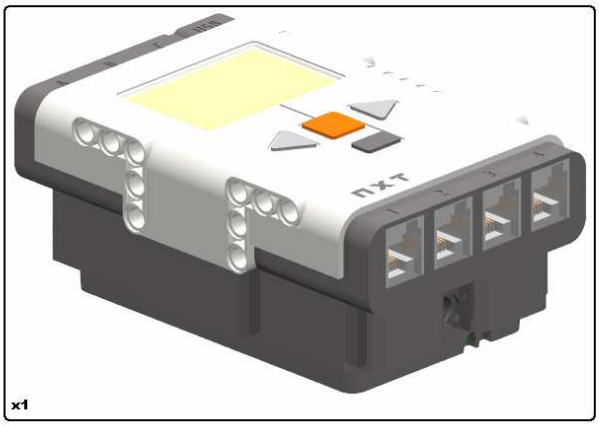


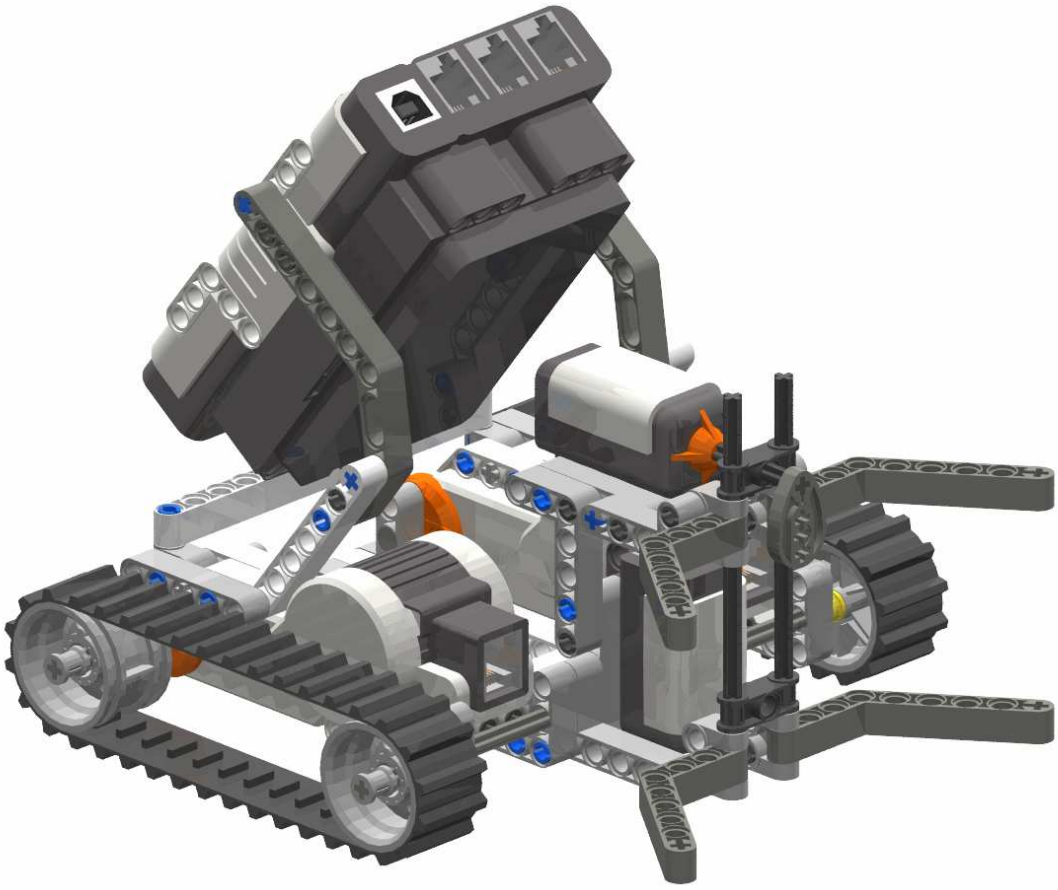
27





28





Lrobotikas.net

Robótica Educativa y Recreativa