UPC UNIVERSITAT POLITÈCNICA DE CATALUNYA

Pablo Samaura Aparicio
Jordi Pérez Almansa
Marc Planells Smith

Profesor: Pere Ponsa
K – 50 Robótica

INDICE

1.INTRODUCCIÓN

Lego Mindstorms es sin lugar a dudas algo más que un juguete, es la manera más sencilla de construir robots, a tu gusto, con la inteligencia que precises y en tu propia casa.

1.1.Historia y datos

La firma danesa **LEGO** creada en 1932 por **Ole Kirk Christiansen**, es la autora de los famosos *ladrillos* ensamblables, la cual nació bajo un lema:

"SÓLO LO MEJOR, ES LO BASTANTE BUENO"

Estos ladrillos están considerados, según la revista Fortune, como el Juguete del Siglo XX. Al principio, la carpintería ubicada en Billund fabricaba muebles y juguetes, siendo más importantes estos últimos por su calidad. En 1934 Ole convocó un concurso en su taller para buscarle un nombre a la empresa, el premio era una botella de vino. Lo ganó él mismo con lego, una contracción de la expresión danesa leg dogt (jugar bien) y que casualmente en latín significa ensamblar.

3 años es la edad mínima para concursar en el mundial Lego.
6 son las fábricas: Dinamarca, Suiza, EEUU, República Checa, Hungría y Corea.
8 son los colores Lego: azul, rojo, blanco, negro, amarillo, verde, gris claro y oscuro.
320 billones de ladrillos se han vendido en el todo el mundo.
5.000 millones de horas al año pasan los niños jugando en Lego.

En el otoño de 1998, sale al mercado Robotic Invention Systems (RIS), como una nueva línea de producto llamado **Mindstorms™.**

1.2.Alcance

Este nuevo producto va enfocado a un mercado adolescente, mayor de 12 años, antiguo consumidor de **Lego Technic**. El equipo contiene elementos necesarios para empezar a construir robots, emisor IR, *ladrillo*, motores, sensores, engranajes, poleas, etc.

1.3.Objetivo

El objeto de nuestro proyecto será construir los robots propuestos por el kit, siguiendo el sistema multimedia proporcionado por LEGO, para crear unas tablas comparativas de las capacidades reales de los mismos. Explicando también la mecánica, electrónica y programación del Robotics Invention System.

2. MECÁNICA

Lego a creado una forma de construcción flexible a base de pequeños módulos(ladrillos), a partir de los cuales se pueden construir diversos objetos, ya sea los que te propone lego cuando te compras el pack, o por la imaginación del propio constructor.

Las piezas que vienen en el pack son las necesarias para montar los objetivos que vienen en los planos de construcción, pero si el constructor quiere crear algún objeto personal que no esta en los planos puede verse un poco limitado según las piezas que necesite, en nuestro caso el Lego Mindstorms lleva todas las piezas necesarias para crear los robots móviles que ellos te sugieren, pero que pasa si en vez de hacer un robot que siga una línea queremos construir una maquina excavadora, el problema con el que nos encontramos es que nos faltan motores ya que los que lleva el pack solo nos serviría para realizar el movimiento de las ruedas pero no para mover loas ruedas y el brazo de la excavadora a la vez.

Debemos tener en cuenta también que si lego nos ofrece un pack con muchas piezas el precio de ese pack será mucho más elevado, y el precio del lego mindstorms no es que sea barato.

De todas formas en la parte mecánica lego nos dice que piezas coger y como juntarlas para realizar los objetos propuestos, nuestra pregunta es, si este tipo de productos son para enseñar, donde nos enseña lego el porque de las construcciones, así como porque juntamos un engranaje grande con otro pequeño o porque juntamos unas cosas con otras.

En esta parte explicaremos las bases mecánicas que utiliza lego para generar según que estructuras y movimientos físicos a partir de sus piezas.

2.1.Dimensiones del ladrillo lego

El elemento principal que hace servir lego en las estructuras es el ladrillo y sus derivados así como las bigas o bloques:

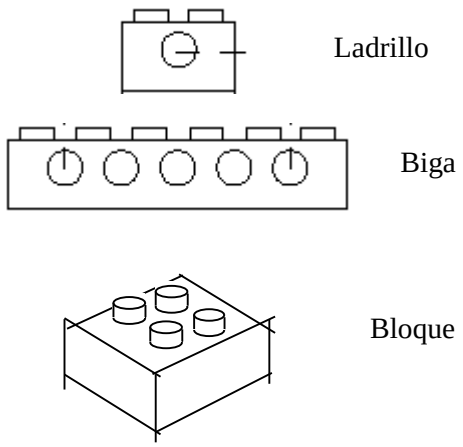


Fig 2.1

La parte horizontal (longitud) es 5/3 la parte vertical (altura) de esta forma 5 ladrillos puestos uno encima de otro es la misma longitud que una viga de 5 agujeros(3 ladrillos en horizontal), así que construir estructuras verticales equipara a integrar longitudes horizontales, esta particularidad hace que las estructuras no se vengan abajo, de todas formas, para asegurarnos podemos reforzar las estructuras mediante mas bigas, para ello también necesitaremos los llamados “platos”(figura 2) que son piezas como las vigas pero que miden 1/3 de la medida vertical:

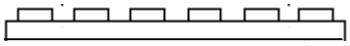


Fig 2.2

Esta pieza sirve para crear espacios verticales entre bigas, una de las cosas que se utiliza es para cuadrar los agujeros de vigas que están conectadas de forma perpendicular:

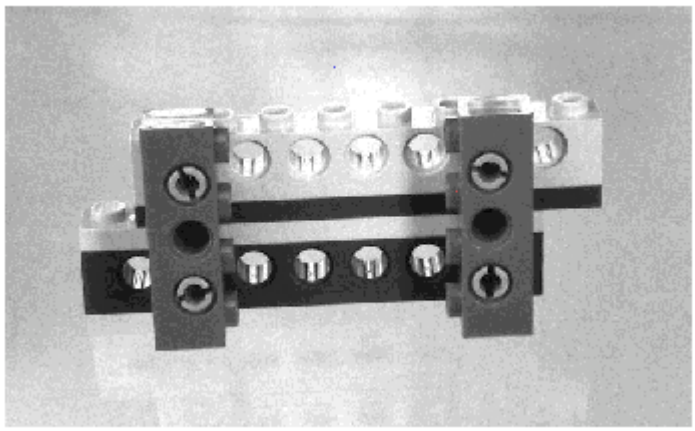


Fig 2.3

En la figura 2.3 vemos como una estructura vertical esta reforzada con bigas, sujetas con clavijas a la propia estructura, para que los agujeros cuadren hemos debido de colocar dos platos.

Normalmente los platos se utilizan para igualar ciertos agujeros para colocar las clavijas que hacen que la estructura sea mas robusta:

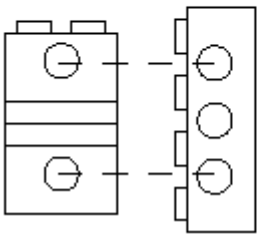


Fig 2.4

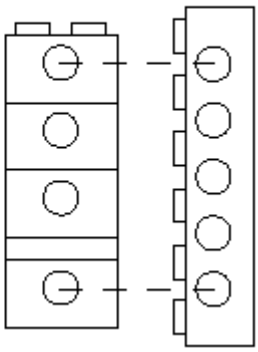


Fig 2.5

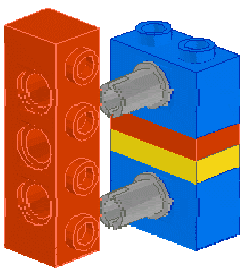


Fig 2.6

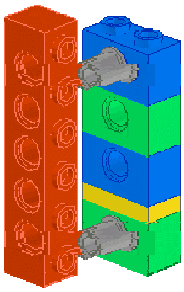


Fig 2.7

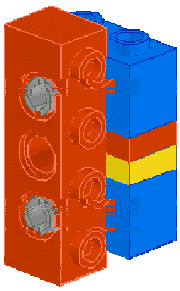


Fig 2.8

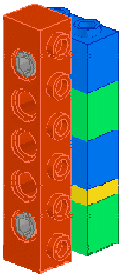


Fig 2.9

Pero otra cosa también para la que se utiliza y que veremos mas adelante, es para cuadrar las distancias de los engranajes.

Los refuerzos en las estructuras se pueden poner perpendicularmente como hemos visto anteriormente, pero también se pueden poner en diagonal, en este caso los cálculos para que cuadren los agujeros donde pondremos las clavijas se realizaran mediante el teorema de Pitágoras, de esta forma si la hipotenusa (refuerzo perpendicular) tiene cinco agujeros, los catetos deben tener 4 y 3 respectivamente.

Las clavijas que se utilizan, suelen ser de dos tipos negros o grises, la diferencia que existe, es que las clavijas negras se agarran mas a los agujeros en cambio los grises resbalan un poco, los negros se utilizan a la hora de que las estructuras queden bien robustas y los grises se utilizan para las articulaciones móviles.

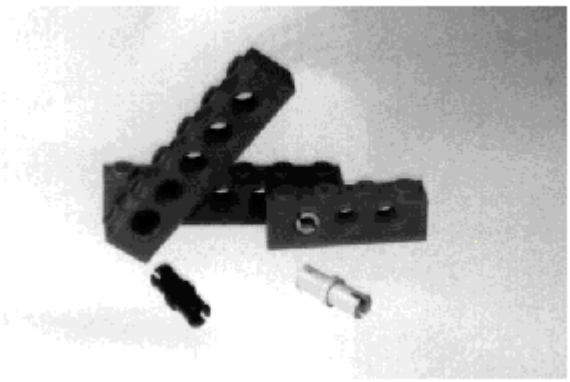


Fig 2.10

2.2. Uniones

Las uniones es una de las partes más importantes de la mecánica de lego, ya que sin ellas no podríamos realizar las estructuras, tendríamos muchas piezas pero no nos servirían de nada.

2.2.1. Uniones verticales

La unión mas conocida que poseen las piezas de lego son las que poseen los ladrillos, bigas, platos,etc..

Nos referimos a los tacos que tienen en la parte superior y los huecos que poseen en la parte inferior, al unirse cada taco con el agujero las piezas quedan fijadas.

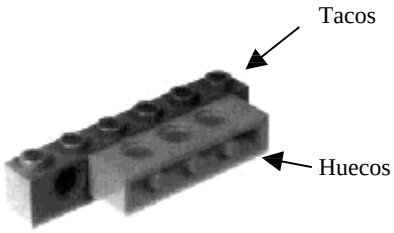


Fig 2.11

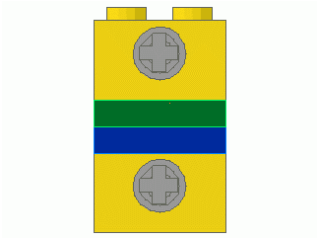


Fig 2.12

2.2.2. Uniones horizontales

Esta forma de unir es mediante clavijas(figura 13), ya comentados en el apartado 2.1, pero debemos comentar que existen diferentes tipos de clavijas:

-Las clavijas para unir horizontalmente dos piezas en el mismo sentido:

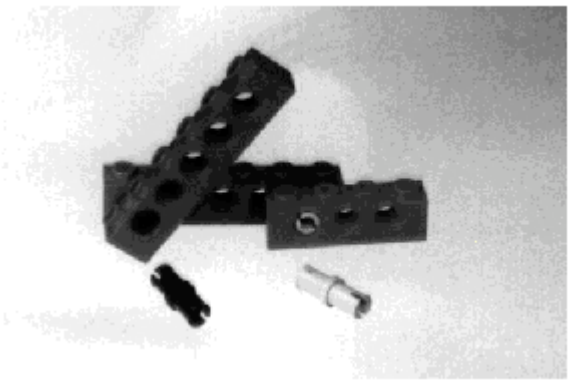


Fig 2.13

En este caso las piezas tienen los tacos mirando para arriba.
-Las clavijas para unir horizontalmente dos piezas que difieren en 45º:

Fig 2.14

2.2.3. Uniones para engranajes

Estas uniones se utilizan para juntar los engranajes con las piezas de lego como son las bigas, ladrillos,etc...

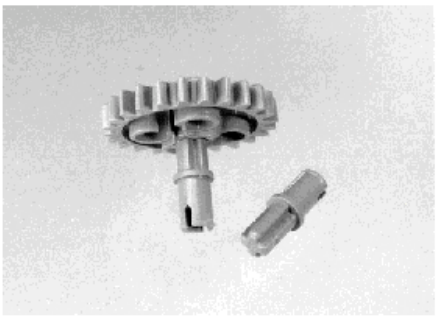


Fig 2.15

De esta forma no tenemos que utilizar ejes y topes para ejes.

2.2.4. Uniones para ejes

Existen topes de ejes que sirven para unir los ejes con las piezas, este tipo de uniones suele fijar el eje y no dejarlo rotar.

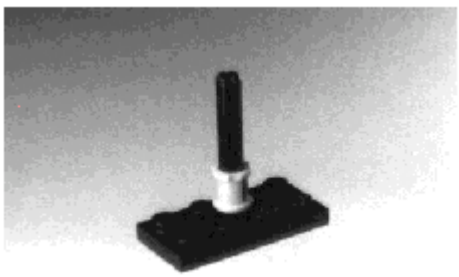


Fig 2.16

Existen uniones para ejes que desplazan angularmente el eje:

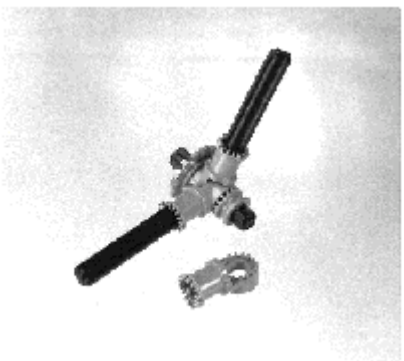


Fig 2.17

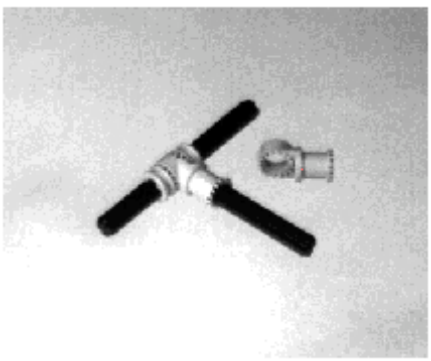


Fig 2.18

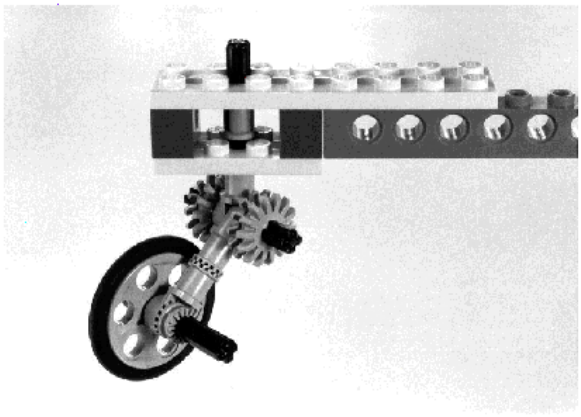


Fig 2.19

Para unir ejes con engranajes, utilizamos una clavija y una unión de tipo :

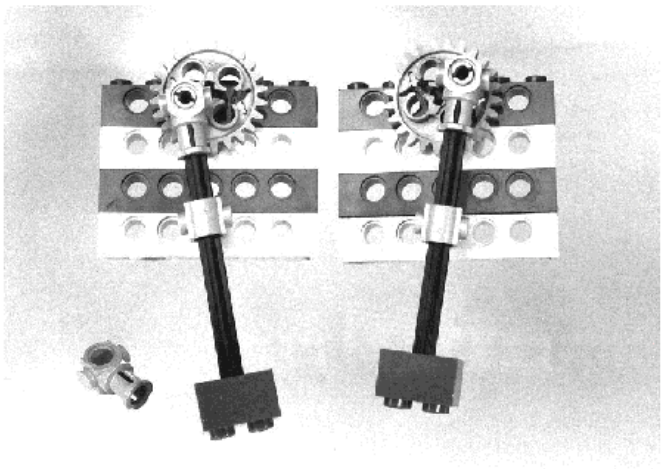


Fig 2.20

2.3. Motores y engranajes

Si ponemos en marcha uno de los motores, podemos observar que la velocidad del eje es muy elevada, pero el torque (fuerza de giro) es pequeño, pues podemos pararlo fácilmente con las manos mientras esta en marcha.

Mediante la utilización de reductores podemos hacer que esta rápida pero débil energía se convierta en una energía lenta pero fuerte.

Esta forma de transformar la energía nos va bien para potenciar las ruedas de los móviles, para dar fuerza a las manos de los robots que cogen objetos, para reforzar las articulaciones, y otras características de movimiento, tanto rotatorias como longitudinales.

2.3.1.Tipos de motores de lego

Los motores que utiliza lego son los siguientes, aunque en pack que hemos utilizado para el trabajo solo estaban los de la clase media:(8,9 N/cm) de torque:

The Motor	Descripción	Velocidad sin carga	Torque	Potencia de pico aproximada
	Gamma alta	1420 RPM 149 rad/s	1.0 N-cm 0.01 N-m 1.4 oz-in	0.37 N-m/s (Watts)
	Gama media (pack Mindstorm)	375 RPM 39 rad/s	8.9 N-cm 0.089 N-m 12.3 oz-in 0.06 ft-lb	0.87 N-m/s (Watts)
	Motor especial.	30 RPM 3.1 rad/s	1.9 N-cm 0.019 N-m 2.6 oz-in	0.015 N-m/s (Watts)

Figura A

2.3.2.Ruedas dentadas

2.3.2.1.Reducción

Las diferentes reducciones de lego son posibles mediante la conexión de diferentes tamaños de engranajes, sin que exista ningún problema con la compatibilidad de los dientes, en nuestro caso tenemos 4 tipos, de 8,16, 24y 40 dientes .

Tipos de reducciones:

- Ratio de reducción 3:1

Mediante un engranaje de 8 dientes y otro de 24 dientes, conseguimos la vuelta entera (una revolución) del engranaje de 24 dientes cuando el engranaje de 8 dientes ha dado tres vueltas o revoluciones;

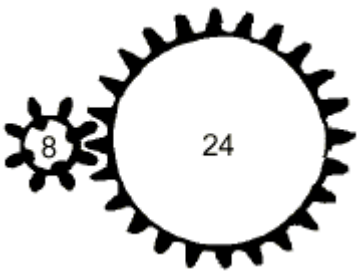


Fig 2.21

En este caso, al engranar desde el piñón pequeño al grande, la aguja corre a una velocidad de 117 rpm:

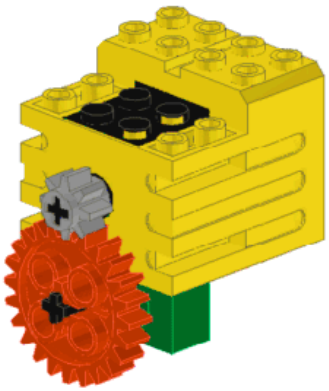


Fig 2.22

Número de dientes conductor:..... 8

Número de dientes conducido:..... 24

Velocidad conductor:..... 117 r.p.m.

Velocidad conducido:..... 39 r.p.m.

Torque conductor.....8,9 N/cm
Torque conducido.....26,7 N/cm

- Ratio de reducción 1:1

En este caso ambos engranajes son idénticos (16 dientes), con lo que no varia la velocidad ni el par de giro. Se puede emplear este sistema cuando queremos invertir el sentido de giro del eje.

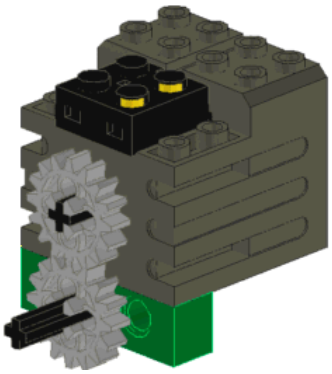


Fig 2.23

Número de dientes conductor:..... 16

Número de dientes conducido:..... 16

Velocidad conductor:..... 117 r.p.m.

Velocidad conducido:..... 117 r.p.m.

Torque conductor.....8,9 N/cm
Torque conducido.....8,9 N/cm

- Ratio de reducción 1:3

En este caso, al engranar desde el piñón grande (24 dientes) al pequeño (8 dientes), la aguja corre a una velocidad de 1.050 r.p.m.
Con este tipo de engranaje se consigue un modelo más rápido aunque menos eficaz en potencia de arrastre

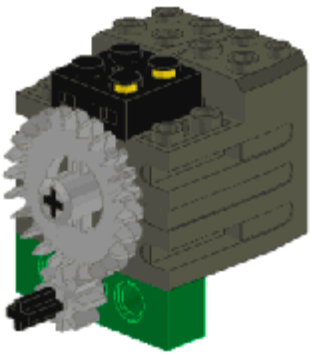


Fig 2.24

Número de dientes conductor:..... 24

Número de dientes conducido:..... 8

Velocidad conductor:..... 117 r.p.m.

Velocidad conducido:..... 351 r.p.m.

Torque conductor.....8,9 N/cm
Torque conducido.....2,96 N/cm

Hemos visto unas serie de ratios de reducción, pero como hemos dicho antes, podemos realizar las reducciones según las combinaciones que hagamos, por ejemplo, al engranar desde el piñón pequeño (8 dientes) al grande (40 dientes), la aguja corre a una velocidad de 23.4 r.p.m. , la velocidad se ha reducido pero el torque aumenta (44,5 N /cm).

¿Cómo podemos saber que combinación de piñones debemos realizar para obtener una velocidad de conducido específica?

Velocidad conductor ..... g1 **Velocidad conducido** g2

Número dientes conductor ..... n1 **Número dientes conducido** n2

$$\frac{g1}{g2} = \frac{n2}{n1}$$

Se observa que la relación de las velocidades es inversamente proporcional a la relación del numero de dientes de sus engranajes.

También debemos tener en cuenta, que si no tenemos los elementos necesarios para conseguir los ratios deseados, podemos realizar la multiplicidad de los engranajes(cadena de engranajes) para conseguir una relación de reducción , en este caso multiplicamos dos ratios de 3:1 (8-24) y conseguimos una relación de 9:1.

9 vueltas 3 vueltas 1 vuelta

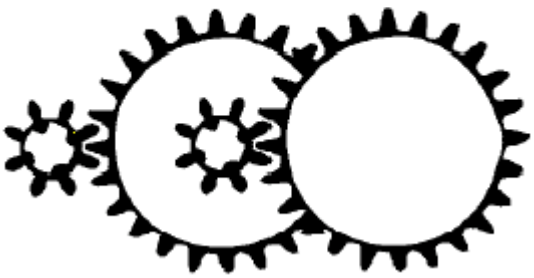


Fig 2.25

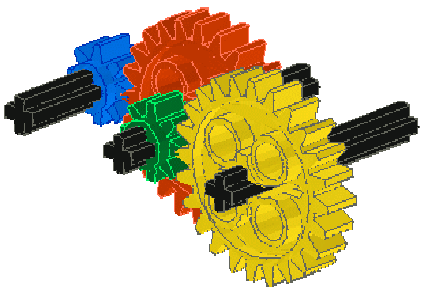


Fig 2.26

Podemos realizar numerosas reducciones, por ejemplo un ratio de reducción de 243:1 del eje del motor a una rueda, esta reducción nos dará una fuerza en la rueda muy grande pero tendremos el problema que ira muy lento, pero nos sirve para ver que podemos realizar reducciones prosperas a las que se utilizan en robots industriales.

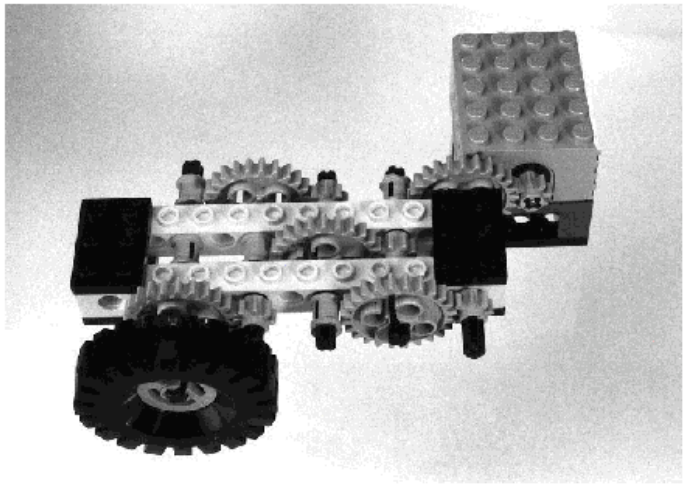


Fig 2.27

2.3.2.2. Sentido del movimiento

Con estos tipos de engranajes podemos realizar movimientos rotatorios sobre el mismo eje de rotación o en ejes paralelos, y según como coloquemos los engranajes el sentido del eje conducido será horario o antihorario en función si el numero de ruedas dentadas es par (sentido opuesto al giro del eje conductor) o impar (sentido del eje conductor)

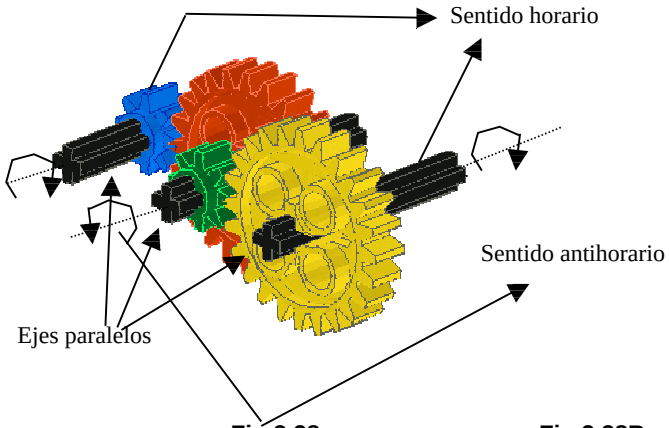


Fig 2.28

Fig 2.28B : casos de utilizar torque de 1:1500(motor 4.5v) 1:40 (motor 9 v)a grandes torques tienen facilidad de romperse

2.3.3. El tornillo sin fin

Este tipo de engranaje, nos ayuda a transformar una serie de movimientos a partir del eje del motor que los otros engranajes no nos permitían hacer, de esta forma la manera de dirigir el movimiento rotatorio a través del espacio se ve ampliado, además de las reducciones que le es posible hacer.

Debemos comentar que este tipo de engranajes producen mas perdidas por fregamiento que las ruedas dentadas y sufren mas desgaste, pero lo positivo que tienen es que no poseen anti-contradirección, esto quiere decir que si estas subiendo una carga mediante este engranaje, si quitas la tensión en el motor, la carga no se caerá, ya que el propio engranaje hace de freno.

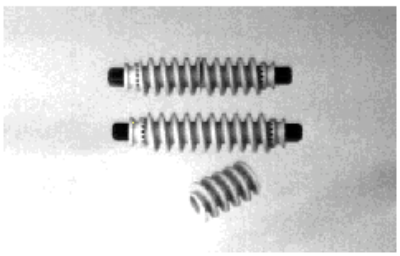


Fig 2.29

2.3.3.1. Reducción

Las reducciones que le es posible transformar a este elemento con las ruedas dentadas viene en función de los dientes de este ultimo, n:1, de esta forma si utilizamos una rueda dentada de 24, la relación de reducción será de 24:1 por una vuelta que de el engranaje oruga, la rueda dentada se moverá un diente, así que deberemos mover el engranaje 24 veces para que la rueda dentada se mueva 1 vuelta, si lo quisiéramos hacer con las ruedas dentadas(caso anterior), deberíamos tener tres grupos de reductores 3:1 para hacer el mismo efecto, de esta forma optimizamos espacio,aunque deberíamos tener en cuenta que el sentido de los ejes es diferente:

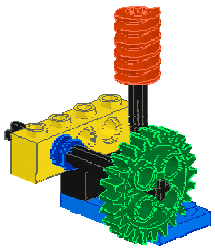


Fig 2.30

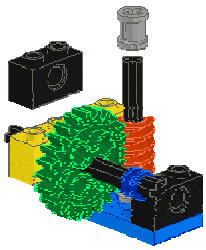


Fig 2.31

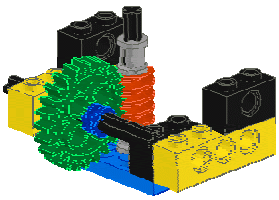


Fig 2.32

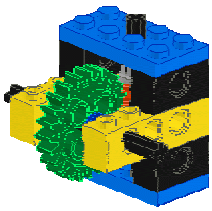


Fig 2.33

2.3.3.2. Sentido de movimiento

Este engranaje nos permite transformar un movimiento rotatorio en el mismo eje(reducción 1:1) o en ejes perpendiculares (n:1), si el sentido de giro será el mismo, así que la única diferencia es que el movimiento (sentido horario o antihorario) se traslada 90º.

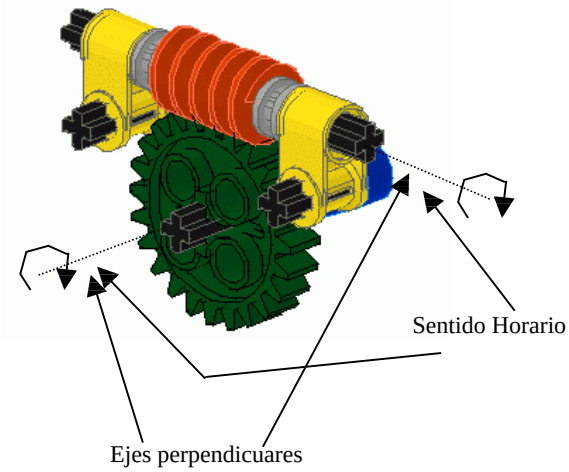


Fig 2.34

2.3.4. Engranajes de corona

Este tipo de engranaje es como el de rueda dentada, la única diferencia es el acabado de los dientes, los cuales permiten un movimiento rotatorio con cambio de sentido de ejes pero con los mismos tipos de reducciones que las ruedas dentadas.

Además tienen la misma medida que estas y se pueden utilizar para las mismas funciones.

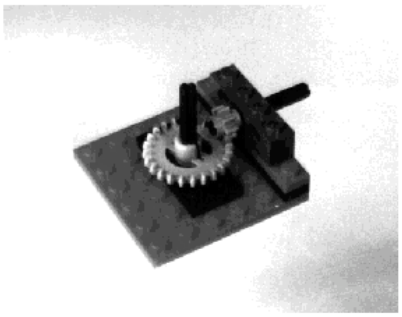


Fig 2.35

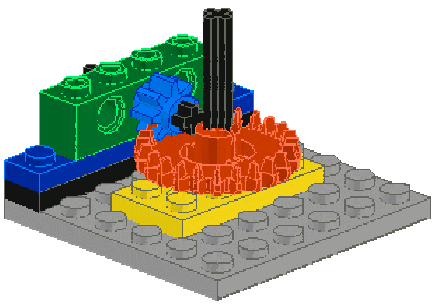


Fig 2.36

2.3.4.1.Sentido de movimiento

En este caso, además de poder realizar los mismos movimientos rotatorios que las ruedas dentadas, se puede realizar movimientos rotatorios en ejes perpendiculares, siendo el sentido de giro contrario al eje conductor.

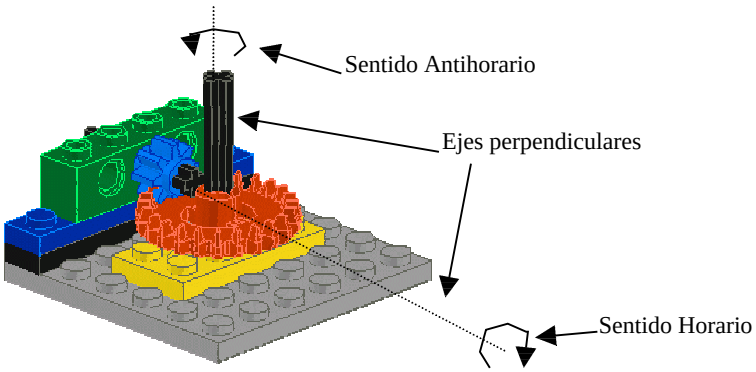


Fig 2.37

2.3.5. Engranaje de cremallera y piñón

Este engranaje se utiliza para transformar un movimiento rotatorio a otro lineal, aunque debemos tener en cuenta, que debemos poner un tope, suele utilizarse para movimientos transversales como a la hora de orientar las ruedas de un coche.

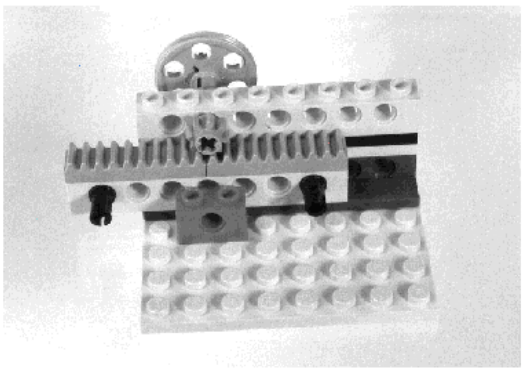


Fig 2.38

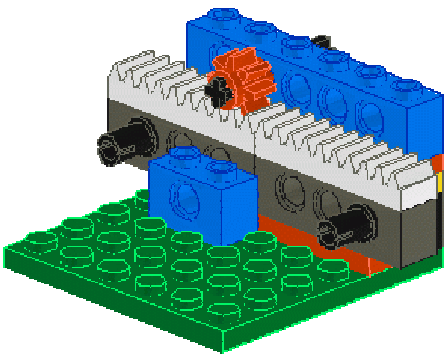


Fig 2.39

2.2.5.1. Sentido de movimiento

Podemos observar en la figura, que si realizamos giro horario, el desplazamiento se realiza hacia la derecha en sentido perpendicular al eje de rotación, y si el giro se realiza en sentido antihorario, el desplazamiento se realiza hacia la izquierda.

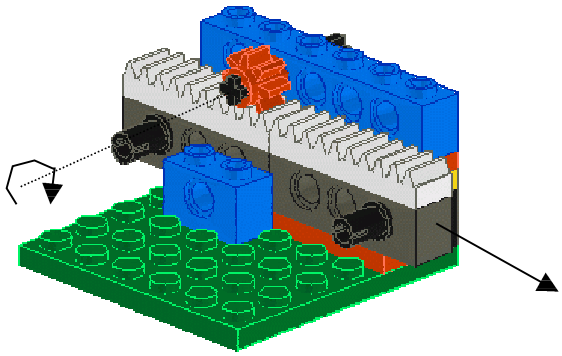


Fig 2.40

Otra forma de crear movimiento rotacional en traslacional, es mediante la construcción de un pistón:

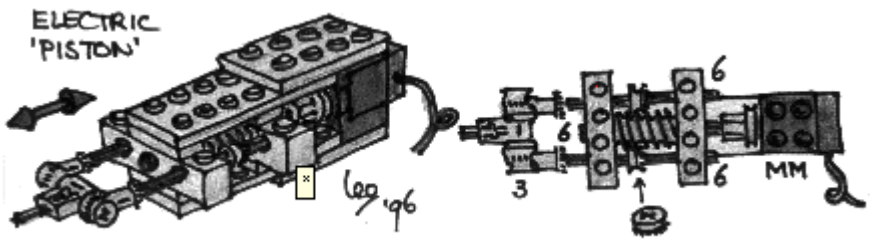


Fig 2.41 **Fig 2.42**

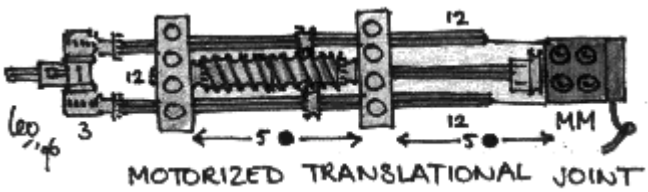


Fig 2.43

Este pistón esta basado en la rotación de un tornillo sin fin que traslada dos topes de ejes, que al mismo tiempo trasladan dos ejes que se unen en uno.

2.4. Dimensiones de los engranajes

Es importante saber las distancias que tienen los engranajes estándares a la hora de conectarlo con las vigas y demás elementos así como hemos demostrado la relación vertical y horizontal para mejores estructuras en puntos anteriores.

De los cuatro engranajes que posee el pack, solo tres(8,24,40) poseen un radio proporcional a la mitad de la medida vertical(altura)del ladrillo(8(1/2),24(1+1/2),40(2+1/2).

Los de 16 dientes tienen un radio igual a la unidad vertical (altura)

Engranaje de 8-24 horizontal

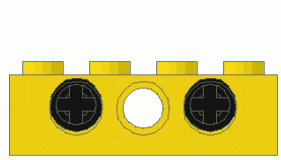


Fig 2.44

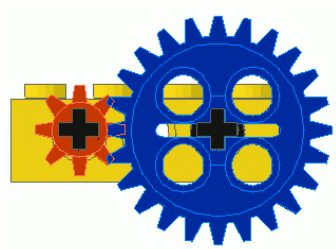


Fig 2.45

Engranaje de 16-16 horizontal

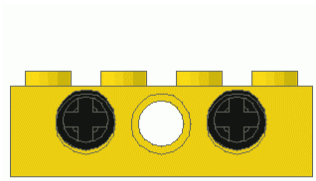


Fig 2.46

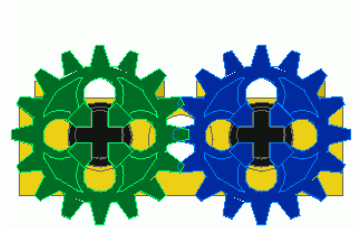


Fig 2.47

Engranaje de 24-40 horizontal

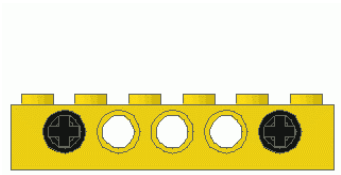


Fig 2.48

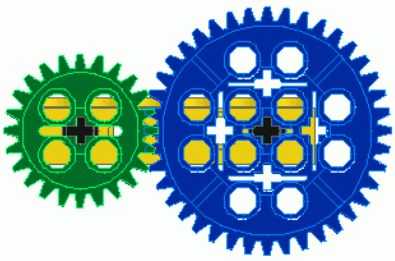


Fig 2.49

Engranaje de 40-40 horizontal

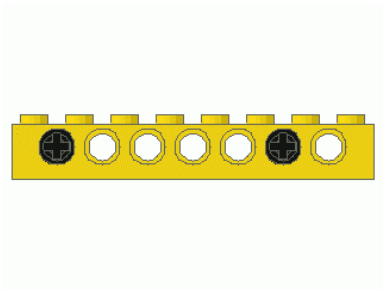


Fig 2.50

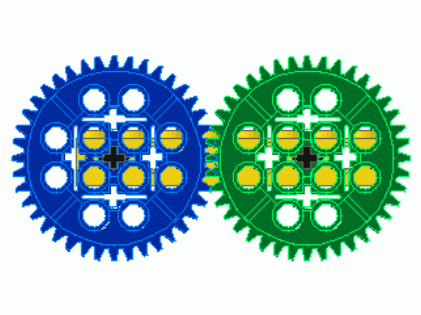


Fig 2.51

Engranaje de 8-24 vertical

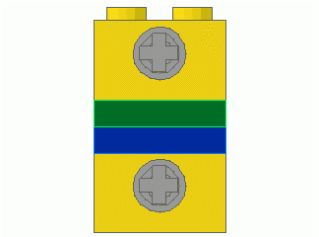


Fig 2.52

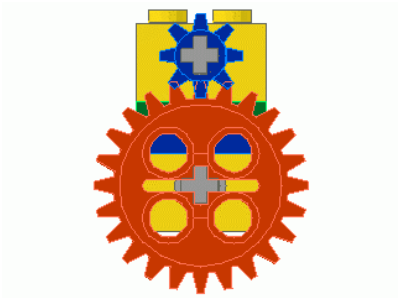


Fig 2.53

Engranaje de 16-16 vertical

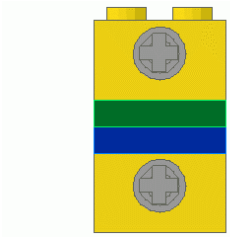


Fig 2.54

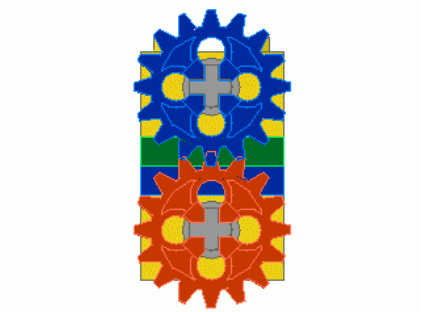


Fig 2.55

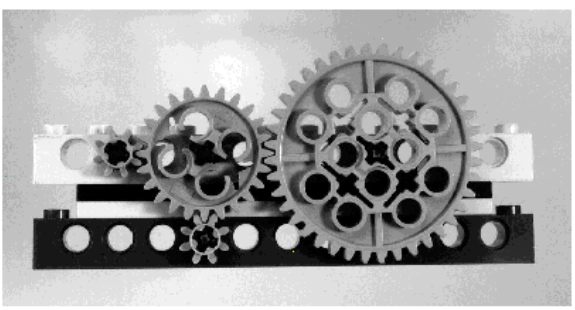


Fig 2.56

Otra forma para realizar el acoplo entre engranajes, es conectarlos en diagonal, incluso es mejor porque así los dientes no están tan cerca unos de otros y no se desgastan tanto.

Para saber la distancia que debemos interponer entre engranajes utilizaremos el teorema de pitágoras, por ejemplo para acoplar una rueda dentada de 8 y una de 16 necesitaremos una unidad vertical y otra horizontal para crear la distancia en diagonal.

2.5. Ejes

2.5.1. Soportes para ejes

Una vez que sabemos o tenemos claro los movimientos de nuestro robot móvil (saber las reducciones, sentidos de movimiento y engranajes a conectar), debemos tener en cuenta los soportes o estructuras en los cuales se fijaran los engranajes, las bigas son los soportes más frecuentes o más utilizados para los engranajes, estos componentes deben crear una estructura robusta, ya que las tensiones que crean los engranajes al moverse pueden hacer que la estructura se rompa si esta no esta fija, o es poco robusta.

En estos casos, las bigas van unidas entre sí por platos dobles, los platos son bigas con 1/3 de vertical, estos son platos más anchos para aumentar los puntos de apoyo de las estructuras, consiguiendo mayor robustez.

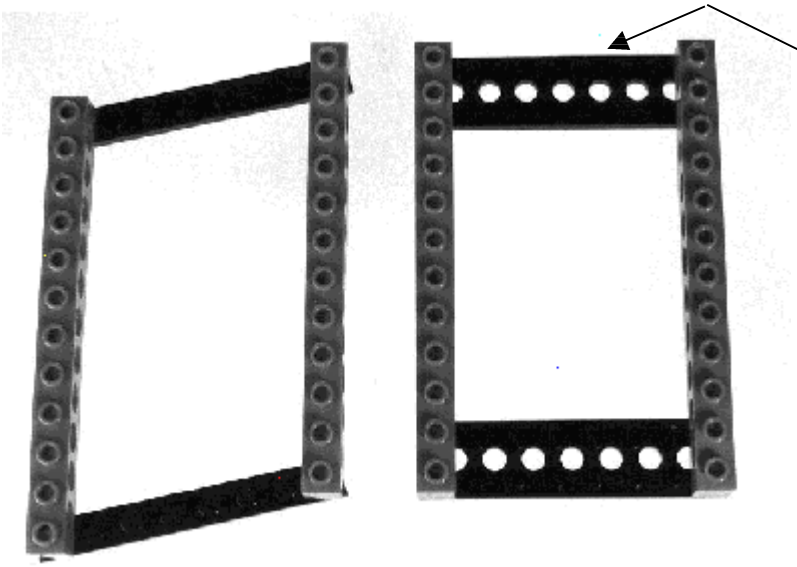


Fig 2.57

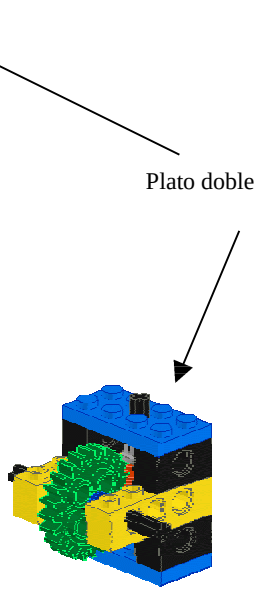


Fig 2.58

Existen componentes que la casa Lego posee pero no lo lleva el kit Mindstorms para la finalidad del soporte de ejes, esta compuesto de un cajón donde se introduce el engranaje y por unos agujeros colocas el eje del engranaje.

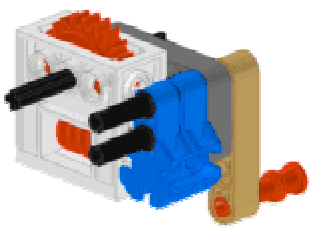


Fig 2.59

2.5.2. Topes para ejes

Los ejes pasan a través de los agujeros de las bigas, estas deben poder realizar movimientos rotatorios pero deben estar fijas en el sitio para que los dientes de una rueda encajen con otra, para esto debemos tener topes en el eje para que no se mueva traslacionalmente, estos topes se aprietan contra la biga, uno en contradirección del otro.

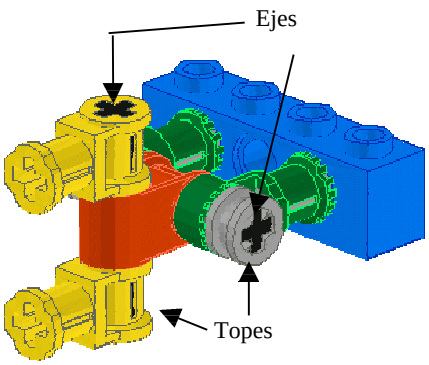


Fig 2.60

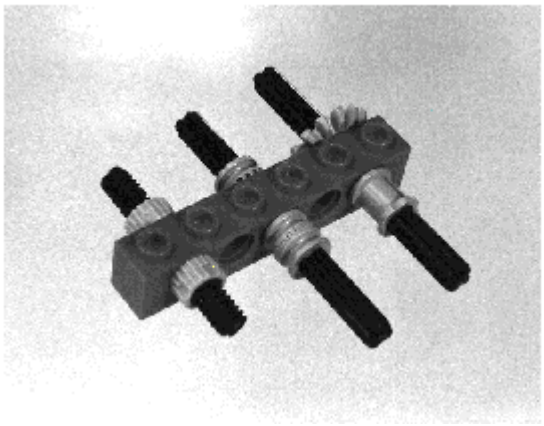


Fig 2.61

Los topes pueden ejercer la función de polleas o engranajes para aprovechar el movimiento de los ejes.

2.6. Poleas

La finalidad de la utilización de poleas, es la reducción de sonido que generan los engranajes en cadena de ruedas dentadas, de esta forma trasladamos la rotación con menos ruido, menos piezas y con las mismas posibilidades de reducción que las ruedas dentadas, pero como todo lo bueno tiene su parte mala, las poleas solo pueden utilizarse en aplicaciones de mucha velocidad y poco torque, debido a que la correa que une las poleas es de goma y pueden resbalar si el torque aumenta.

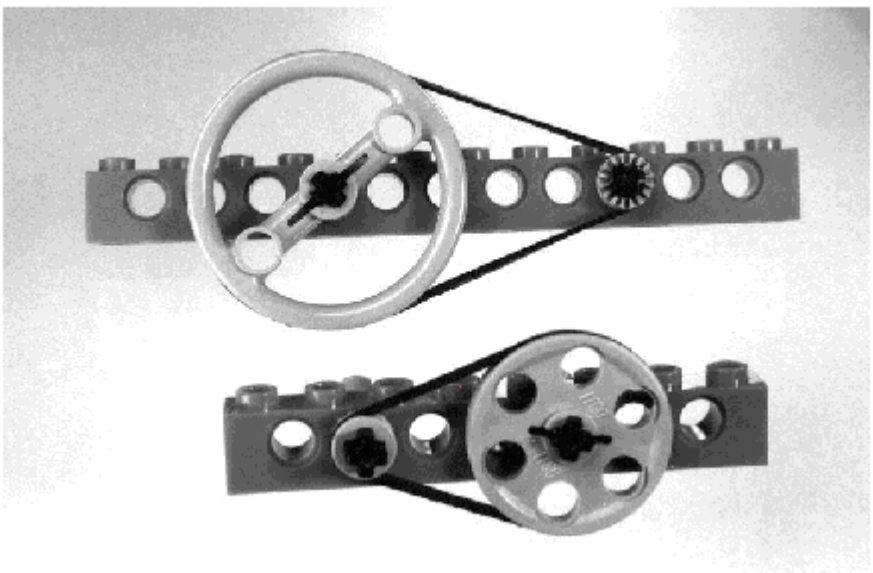


Fig 2.62

Para combatir el tema de la fricción, se puede poner correas pequeñas pero corremos el riesgo de que se rompan.

Existen otras correas como las cadenas que nos evitan ese problema además de no tener tantas perdidas de fricción, este tipo de correas no pueden estar ni muy apretadas(fuerzan los dientes) ni muy flojas(pueden saltar y soltarse).

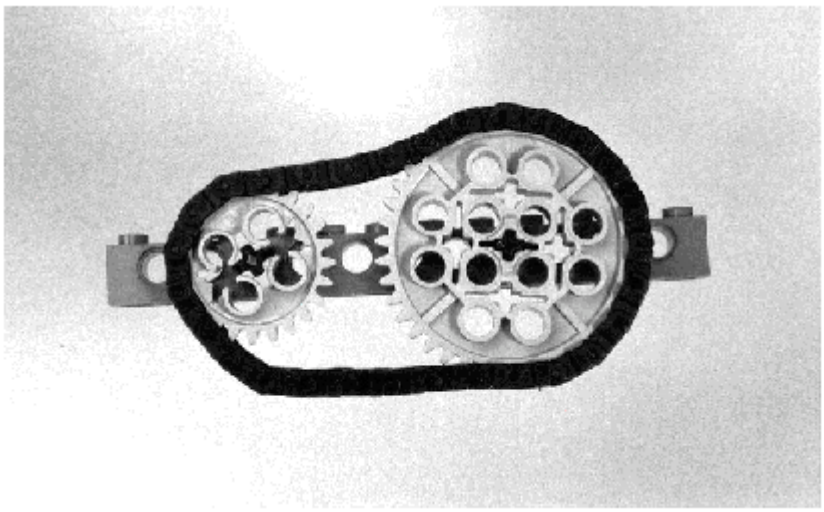


Fig 2.63

2.7. Neumática

Otra de las cosas que se pueden hacer dentro de la mecánica de lego es la neumática, de esta forma podemos crear amortiguadores para los coches, aunque esta opción no viene en el Kit pero es interesante que se conozca:

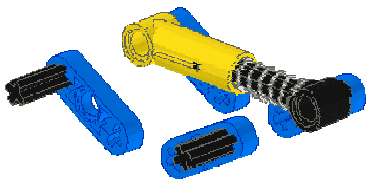


Fig 2.64

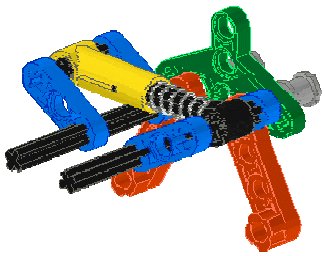


Fig 2.65

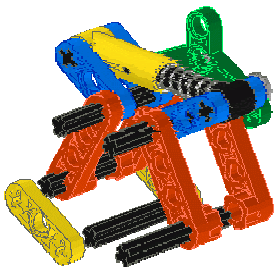


Fig 2.66

2.8. Utensilios para diseñar la mecánica de lego

Existen herramientas de software a la hora de construir un robot , la más útil esta que enseñamos en la figura 3.67, el diseño se realiza mediante el MLCAD que es un derivado del AUTOCAD para lego, de esta forma podemos plasmar una idea,ver la y modificarla para después poder montarla:

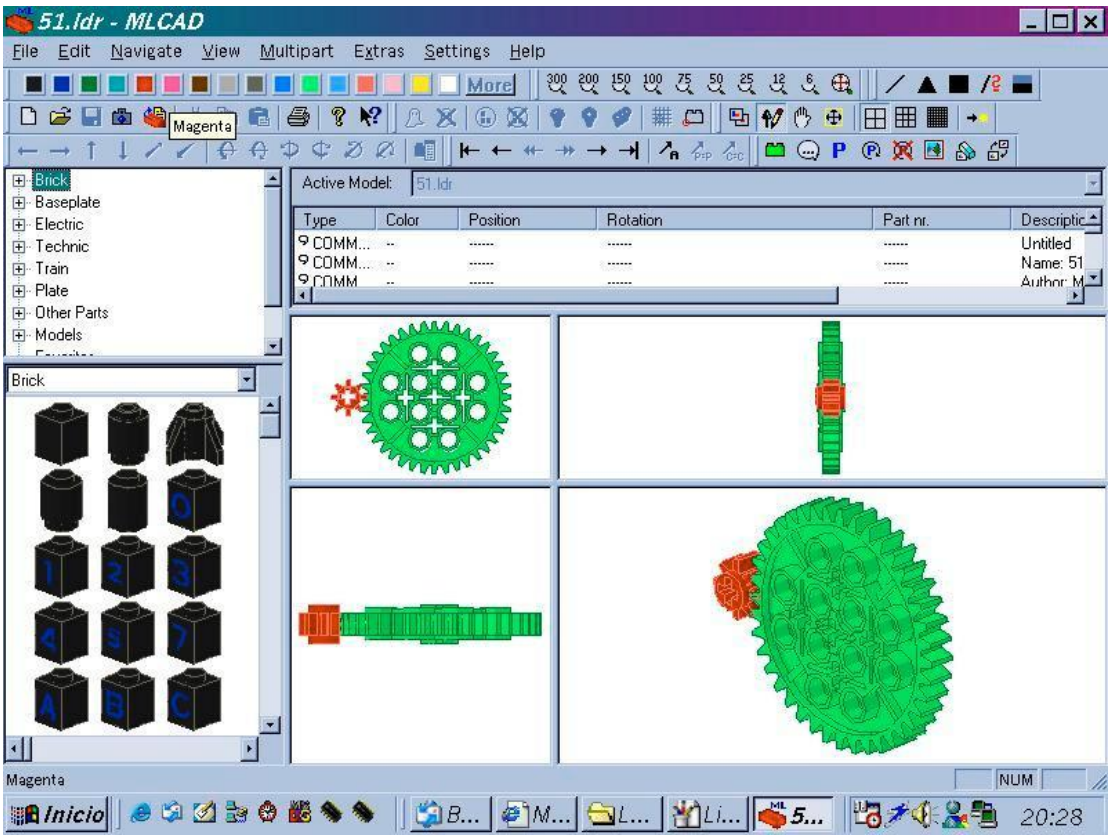


Fig 2.67

2.9 Tablas comparativas

Dimensiones del ladrillo de Lego		
Elemento	**Medida vertical**	**Medida horizontal**
Ladrillo	1 unidad	5/3 de la unidad
plato	1/3 de la unidad	Múltiplo de la medida horizontal
Biga	1 unidad	Múltiplo de la medida horizontal
Ejes	½ de la unidad	Múltiplo de la medida horizontal
Engranajes dentados (8,24,40)	Múltiplos de ½ de la unidad	Múltiplos de ½ de la unidad
Engranajes dentados (16)	1 unidad	1 unidad

Tabla 2.1

Uniones	
Tipo	**Finalidad**
Vertical	Tacos y huecos de los elementos
Horizontal	Clavijas (lateral-lateral)(lateral-inferior) unir a agujeros
Engranajes	Clavijas medio eje-medio clavija) para unir a agujeros
Ejes	Fijos o móviles y angulares
Engranajes y ejes	Mediante la inserción del eje en el engranaje y topes de eje

Tabla 2.2

Engranajes			
Tipo	**Desgaste**	**Reducción**	**Movimiento de giro**
Ruedas dentadas	Se rompen en torques elevados, mucha duración si el torque es normal y la velocidad también	Ratio (3:1) Conductor 8 dientes (117 rpm ,8,9 N/cm) Conducido 24 dientes(39 rpm, 26,7 N/cm) Ratio (1:1) Conductor 16 dientes(117 rpm,8,9 N/cm) Conducido 16 dientes(117rpm, 8,9 N/cm) Ratio (1:3) Conductor 24 dientes(117 rpm,8,9 N/cm)) Conducido 8 dientes(351rpm,2,96 N/cm) Ratio (n:1) Fórmula , Cadena de engranajes (8,16,24,40)	En el mismo eje o en ejes paralelos el giro del eje conducido va al contrario del conductor
Tornillo sin fin	No se rompen pero se desgastan mucho a torques elevados	Ratio (n:1) en función de los dientes de las ruedas Ratio (24:1) Conductor tornillo sin fin (117 rpm ,8,9 N/cm) Conducido 24 dientes(4,8 rpm, 213,6 N/cm)	En ejes perpendiculares pero el giro es el mismo que el conductor.
Engranaje corona	Como las ruedas dentadas	Igual que las ruedas dentadas	Posibilidad de realizar movimiento en ejes perpendiculares, giro al contrario al conductor.
Engranaje cremallera y piñón	Como las ruedas dentadas	Igual que las ruedas dentadas, en relación con la longitud de la cremallera (perímetro rueda) y la rueda conductora	Movimiento lineal Giro antihorario desplazamiento izquierda Ejes perpendiculares
Pistón	Como el tornillo sin fin	Igual al tornillo sin fin	Movimiento lineal en el mismo eje o paralelos al giratorio

Tabla 2.3

Ejes

Soportes de ejes	Topes de ejes
Estructuras basándose en bigas y platos dobles, los platos simples, no dan robustez a la estructura	Sujetan el eje pero dejan que gire Con fijación, para inmovilizar el eje Con polea, el mismo giro puede ser trasladado Engranaje, el mismo giro del eje puede trasladarse

Tabla 2.4

Poleas
Tienen misma reducción que las ruedas dentadas y pueden hacer de ruedas, eliminan ruido y engranajes

Correas	Cadenas
Solo para velocidades altas y poco torque, se desgastan mucho y si se tensan mucho posibilidad de romperse, eliminan ruido y engranajes en cadena	Mejoran las correas en operaciones de torque, pero se debe colocar bien ni muy apretado, no realiza bien la transmisión, ni muy flojo, pueden soltarse.

Tabla 2.5

3.ELECTRÓNICA

3.1.RCX

3.1.1.Introducción

El **RCX** conocido como ladrillo fig.3.1 por su forma y robustez física, es un microcontrolador programable capaz de controlar simultáneamente tres motores, tres sensores y un puerto serie de infrarrojos. Es la pieza cerebro de las 717 que contiene el kit de LEGO® Robotics Invention System.

Fig. 3.1 Unidad RCX

3.1.2.Descripción general

La utilización del sistema base RCX consiste en el mismo RCX, un emisor-receptor de infrarrojos y un PC. Si lo combinamos con componentes adicionales, como motores, sensores y piezas de construcción, podremos crear robots móviles funcionales autónomos.

En el corazón del RCX tenemos el microcontrolador **Hitachi H8** con 32K de memoria externa. El microcontrolador es usado para el control de los tres motores, tres sensores y el puerto serie de comunicaciones. Contiene un chip de 16K de memoria ROM con una pequeña ROM que se ejecuta la primera vez que ponemos en marcha el RCX. Este chip esta pensado para descargar 16K de firmware (sistema operativo del RCX). Ambos la ROM y el firmware aceptan y ejecutan comandos desde el PC a través del puerto de comunicaciones IR. Posteriormente los programas de usuario se descargan al RCX como código binario y se almacenan en una región de la **memoria de 6K**. Cuando el programa usuario se ejecuta, el **firmware** esta interpretando y ejecutando el código binario.

3.1.3.El Microcontrolador H8/3292

El microcontrolador utilizado para el RCX es un Hitachi H8/3292 de la familia H8/3297. El chip integra tres componentes principales, una CPU H8/300, memoria y entradas/salidas. Estos componentes están conectados por un on-chip controlador del bus de datos y direcciones.

La CPU H8/300 esta basada en una arquitectura de registros generales. El juego de instrucciones incluye una ALU(Unidad aritmético lógica). Los modos de direccionamiento son los más comunes modos de registro indirecto, directo, contadores relativos de programa y memoria indirecta. El espacio de direcciones es de 16-bit (64 Kbytes) combinado para programa y datos.

El chip principal de memoria consiste en una ROM programable de 16 Kbytes y una RAM de 512 bytes, el chip de registro de campos de 128 bytes se utiliza para mapear los registros de los periféricos de entradas/salidas. El mapa de memoria define como el chip interno de memoria y los posibles chips externos se mapean dentro del espacio de direcciones de 16 bits.

El mapa de memoria en parte es definido por el nivel de entrada sobre dos pins del chip (llamados MD1 Y MD0; se puede tener acceso con el bit 1 y el bit 0 del Registro de Control de Modo en la dirección 0xffc5 en el campo de registro). Estos dos bits seleccionan un modo de operaciones del microcontrolador (modo 1, 2 o 3). En todos los modos el chip de memoria RAM y el chip de registro de campos apuntan en la misma dirección. En el modo 1 y 2, modos llamado de expansión, es posible tener acceso al chip de memoria externa y registros de dispositivo por los pins de dirección / datos del microcontrolador. En el modo 3, modo simple, sólo están disponibles las memorias ROM, RAM y el chip de registro de campos.

Cuando la memoria externa es usada, es mapeada en el espacio de direcciones mediante la lógica externa para decodificar direcciones.

La entrada / salida del micro incluye tres tipos de temporizadores ,un temporizador de 16 bits, temporizadores de 8 bits, y un temporizador de perro guardián (Watch Dog), un interfaz de comunicación serie, un convertidor Analógico/digital, y puertos de entrada/salida. Los temporizadores pueden ser usados sin el conjunto de chips exteriores.

El interfaz de comunicación serie puede actuar como un controlador de dispositivo transmisor/receptor externo serie. Así mismo el convertidor analógico-digital de 10 bits puede ser convertido en un controlador de dispositivos de hasta ocho canales analógicos. La entrada externa analógica conectada a una de las 8 entradas de línea del convertidor analógico-digital puede ser sampleada por un circuito de muestreo-bloqueo y convertida a un valor de 10 bits disponible en un registro de dispositivo.

Un circuito multiplexor analógico puede operar en dos modos modo disparo y modo de exploración. El modo de exploración permite la conversión continua sobre más canales que entregan resultados digitales en registros de dispositivo diferentes. Los pins de los puertos de entrada/salida pueden ser unidos a la entrada de memoria externa o a las líneas de salida. Cuando el procesador lee de un registro de dispositivo correspondiente a un puerto de entrada de 8 bits, el valor de byte devuelto refleja unos y ceros sobre las líneas de entrada.

Con casi ningún componente añadido de circuitería externa, un fototransistor puede ser conectado directamente a una línea de entrada y el estado del fototransistor puede ser introducido desde el puerto de entrada. Cuando el procesador escribe en un registro de dispositivo correspondiente a un puerto de salida de 8 bits, el valor de byte de 1's y 0's colocado en el registro es la salida de los pins del puerto de salida y las líneas de salida unidas a los pins pueden ser usadas directamente con un circuito externo como un LED.

Un controlador de interrupciones proporciona un mecanismo de interrupciones internas y externas. Las interrupciones internas son generadas por eventos de los componentes de entrada / salida del microcontrolador. Cada acontecimiento por separado (p.ej. el final de la conversión analógico-digital y el desbordamiento de temporizador) interrumpen en vectores de interrupción diferentes. Las interrupciones externas son generadas por acontecimientos externos detectados por los pins del microcontrolador (p.ej. un pin para las interrupciones no enmascarables NMI). Todas las interrupciones a escala general o las interrupciones individuales pueden ser deshabilitadas exceptuando las NMI.

3.1.4.Capacidades reales del RCX en función del Firmware

Firmware de Lego	Firmware LegOS
Las subrutinas no admiten parámetros, por lo que no es posible emplear realmente los principios de la programación estructurada.	La carga dinámica de programas y módulos.
Sólo se pueden usar variables globales, es decir, el espacio de nombres es único, lo cual complica enormemente el desarrollo y mantenimiento de programas complejos.	El protocolo de comunicación basado en el transmisor infra-rojo.
Las subrutinas no pueden devolver valores, lo cual las limita a ser subconjuntos de código, sin poder aportar funcionalidades nuevas ocultando su implementación.	La posibilidad de realizar programas multitarea.
El número de variables está enormemente limitado, de hecho sólo se dispone de 32.	La gestión de memoria dinámica.
No existen las estructuras de datos, ni las estáticas ni las dinámicas, lo que imposibilita casi cualquier tipo de aproximación que necesite almacenar cualquier tipo de estado.	La existencia de *drivers* para todos los subsistemas del ladrillo.
	El uso de la velocidad nativa del micro-procesador, esto es 16 MHz.
	El acceso a los 32K de memoria RAM.
	Permite el uso completo del lenguaje de programación elegido, como por ejemplo C. Lo cual implica que se pueden usar punteros, estructuras de datos, etc.

·Tabla 3.2 Comparativa de capacidades reales del RCX en función del Firmware.

3.1.5.Estructura exterior del RCX

El ladrillo RCX, tiene forma de bloque fig.3.4 y esta compuesto de la siguiente manera:

Dispositivos de Entrada	Dispositivos de Salida
4 botones Run, OnOff, View, Prgm	Pantalla de cristal líquido
3 puertos de entrada nombrados 1, 2 , 3	Altavoz
Nivel indicador de baterias.	3 puertos de salida nombrados A, B, C
Temporizadores	Emisor de infrarojos
Receptor de Infrarojos	

• Tabla 3.2 Extraida de: http://gul.uc3m.es/gul/docs/iiicongreso_hispalinux/lego/hispalinux2000.pdf

Tabla 3.3 de Entradas/Salidas del RCX.

Fig.3.4

Una vez descargado firmware en el RCX, se puede utilizar el botón View del RCX (fig 3.5) para comprobar la lectura del sensor táctil conectado al puerto 1. Para obtener la lectura del sensor táctil, debemos descargar un programa que utilice un sensor táctil y ejecutarlo al menos una vez.

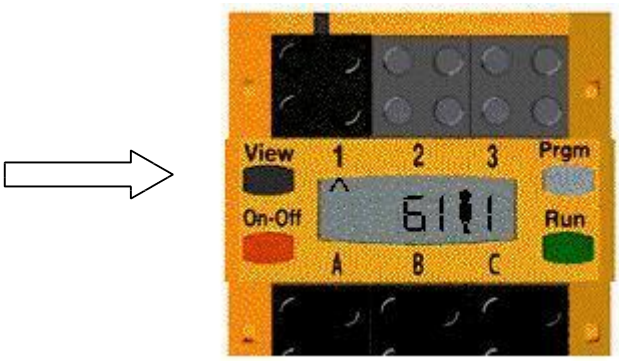


Fig.3.5

3.1.6.Estructura física interna del RCX

La composición interna esta formada por un compartimento para 6 baterías AAA, y el circuito impreso:

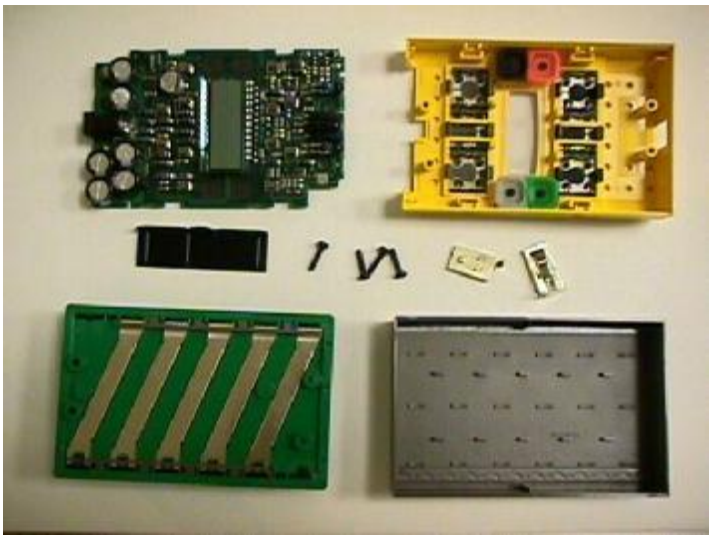


Fig. 3.6

3.1.7.Limitaciones del RCX

Las limitaciones más importantes en cuanto a nivel electrónico:

- Escaso nº de E/S (6).
- Escasa variedad de sensores y actuadores por parte de LEGO.
- Desaprovechamiento de las capacidades reales del microprocesador.

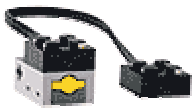
3.2. Sensores de LEGO

El RCX recibe información del exterior a través de sus sensores periféricos conectados a los puertos de entrada del RCX, los siguientes sensores están disponibles:

Sensores	Aplicaciones
Sensor de Contacto	Detectar objetos, señal de contacto físico
Sensor de Luz	Luz reflejada, luz ambiente
Sensor de Rotación	Número de vueltas por ruedas, movimientos troncales
Sensor de Temperatura	Temperatura corporal, temperatura exterior

Tabla 3.7 de Sensores

En la fig. 3.8 podemos apreciar sus formas y colores:

Entradas

Fig. 3.8 Sensores de Lego.

3.2.1. Sensores internos

Lego define como sensores internos, funciones internas del micro que son 4 temporizadores y 3 contadores:

Temporizadores:

Intervalo predefinido del temporizador será de 0,1 a 1 segundo. El máximo es 327,6 segundos. RCX ofrece temporizadores independientes con una resolución de 100 ms (10 tics por segundo).Con NQC los temporizadores van desde el tic 0 hasta el tic 32767 (unos 55 minutos).

Contadores:

Podemos asociar secuencias de programa que se ejecutaran cuando el contador RCX interno alcance un valor que se encuentre dentro del intervalo definido. El contador se pone a 0 cada vez que se ejecuta un programa. Podemos incrementar el contador de uno en uno. El valor máximo del contador es 32766.

3.2.2. Sensores externos

3.2.2.1.Sensor de luz

Características:

-Mide la cantidad de luz que llega a una célula foto-eléctrica (básicamente una resistencia).
-El sensor de luz mide desde los 0.6 Lux hasta los 760 Lux.
-El RCX escala las medidas en porcentajes de 0-100 .
-La resistencia es baja con luz y alta con oscuridad (sensor de oscuridad) .
-Usualmente el software invierte los valores (bajo oscuridad)

Se puede usar de diversas formas:

- Puede medir intensidad
- Puede orientar, enfocar, proteger.
- Su colocación influye.

Funcionamiento:

El sensor de luz mide la cantidad de luz en una dirección particular. El sensor de luz también emite luz. De este modo, es posible apuntar con el sensor de luz en una determinada dirección y distinguir la intensidad de luz reflejada por el objeto en esa dirección. Esto es especialmente útil cuando intentamos que un robot siga una línea en el suelo.

Fig. 3.9 Sensor de Luz

Interior del sensor:

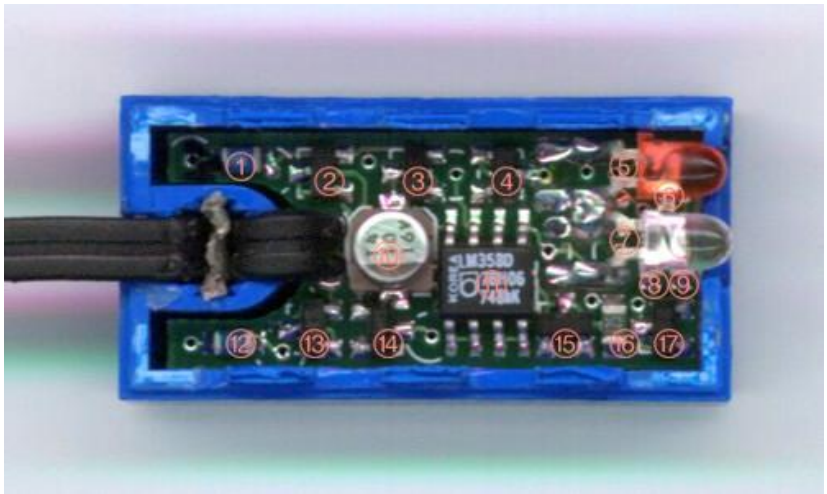


Fig. 3.10 Interior del sensor

Posible esquema electrónico:

Un chico llamado Sasuraibito de nacionalidad Japonesa, ha hecho un gran trabajo de ingeniería inversa, con el sensor de Luz de LEGO. Estudiando el interior del sensor de luz y con sus conocimientos ha descrito un posible diagrama de circuito (fig. 3.11). Los números sobre la fotografía corresponden a los dispositivos numerados sobre el circuito.

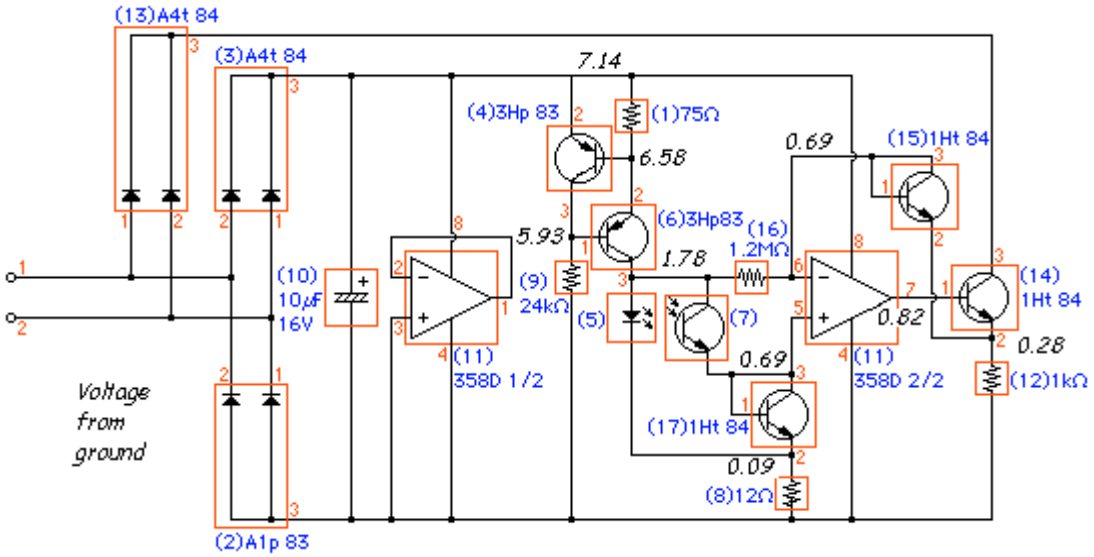


Fig. 3.11 Diagrama electrónico del sensor de Luz.

Análisis de Circuito:

2,3 y 10 componen el típico circuito de fuente de energía de sensor impulsado. A4t84 es un BAV70 y A1p83 son diodos dobles de Philips de alta velocidad BAW56 con una curva VI muy grande como la de un 1N4148. 11 1/2 es la mitad no usada de opamp dual. LM358 es un opamp dual de Philips de baja potencia . 1, 4, 6, 9 forman una fuente de corriente de 7.5ma para los 5 diodos y el fototransistor 7. Los dos 3Hp83's son BC857 PNP de silicio, transistores de objeto general de Philips. El fototransistor es probablemente algo como el Panasonic PN168 NPN de silicio, el fototransistor tiene una sensibilidad máxima de 800nm. Un efecto secundario interesante del circuito es que si el fototransistor es expuesto a una luz muy brillante como un indicador de láser, el led rojo se disparará. El voltaje sobre el led rojo es usado para influir en la segunda mitad del opamp 11 2/2 LM358. Dudo sobre la función de transistor 17, pero estoy bastante seguro de que ayuda a ampliar la gama de resplandor del sensor. 1Ht84 es un BC847W NPN de silicio, el transistor de propósito general de Philips es usado aquí como un diodo. Con los aumentos de corrientes por el fototransistor, la curva VI del diodo que limita el voltaje se cae y amplía el final brillante de la gama. Pienso que el transistor 15 (también se usó como un diodo) .Ambos 1Ht84's son también BC847W NPN de silicio, transistores de propósito general de Philips. 14 es un seguidor de voltaje, devuelve la lectura formada por el diodo 13 y la resistencia 12. A4t84 es también un diodo doble de alta velocidad de disparo BAV70 de Philips. Hay una operación común sobre los sensores de luz de LEGO que implica el retiro del led rojo.

Mejoras:

La creencia es que si el LED rojo se quita, el sensor de luz ambiental trabajaría mejor. El autor del articulo ha simulado el circuito del sensor de luz de LEGO con y sin el LED rojo para cuantificar el efecto. La Fig.3.12 muestra la lectura que realizaría el RCX . Conseguimos una escala de mayor precisión sin el LED rojo para lecturas de luz ambiental:

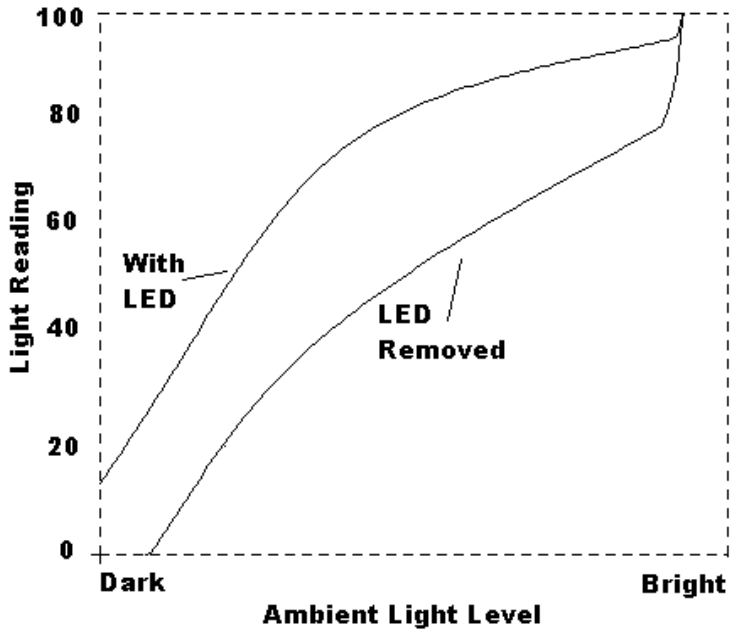


Fig. 3.12

3.2.2.2.Sensor de Rotación

Características :

- Mide la rotación angular: odómetro (número de vueltas), velocímetro (velocidad).
- El sensor de Rotación de LEGO es capaz de leer 16 posiciones por vuelta completa.
- La máxima lectura de velocida es a 500 Rpm con una resolución aceptable.
- El RCX leerá los grados de movimiento angular o 16 posiciones de una vuelta completa.
- Problemas de lectura a muy bajas velocidades.

Fig. 3.13 Sensor de rotación

Funcionamiento:

Durante 3ms se aplican 8V y seguidamente se hace una lectura de tensión mientras tanto se aplican 5V por la resistencia de 10,000 ohm durante 0.1ms. El sensor sólo tiene 4 salidas de valores analógicos 1.8v, 2.6v, 3.8v y 5.1v. La fig. 3.14 muestra la secuencia de voltaje para un sensor de rotación con el aumento de valores. Cada voltaje corresponde a 22.5 grados de rotación así hay 16 posiciones por una rotación. Se deduce que el rango de voltajes permite al RCX decir la dirección de cualquier punto. La gama total del RCX es -32767 a +32767 y esto se puede mantener incluso cuando directamente se une el sensor al eje de un motor.

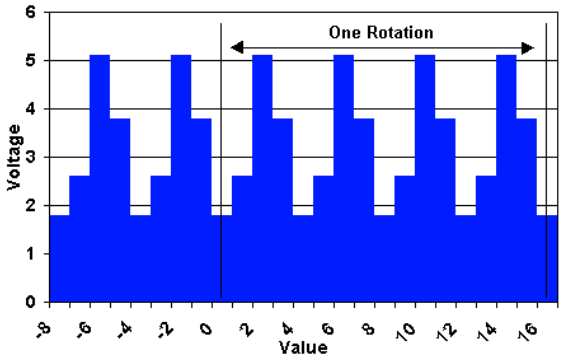


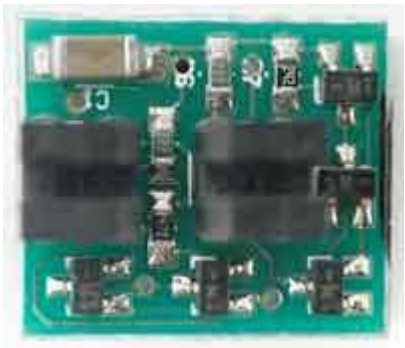
Fig 3.14 .Cuadro de Lecturas

Interior del sensor:

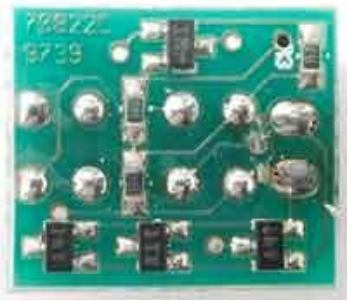
Interior del sensor de rotación.

Partes del sensor de rotación.

Circuito impreso.



Vista superior del cto.



Vista trasera del cto.

Esquema electrónico:

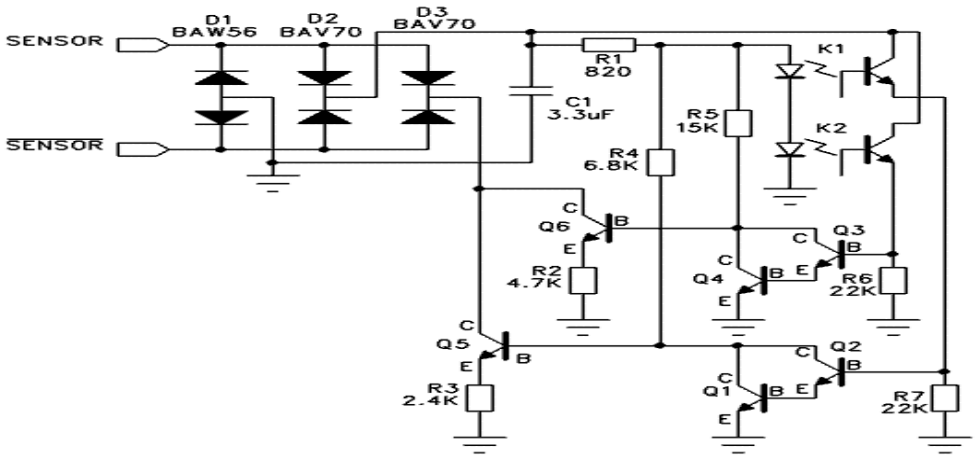


Fig. 3.15 Esquema electrónico del sensor de rotación

Estado de los transistores en las lecturas:

	Vout	Q5	Q6
Step 1	1.3V	ON	ON
Step 2	3.3V	OFF	ON
Step 3	2V	ON	OFF
Step 4	4.5V	OFF	OFF

Tabla 3.16 de estados de trt´s

Mejoras:

El sensor de rotación crea problemas de lectura en bajas velocidades, debido a lecturas erróneas en la transición de posiciones:

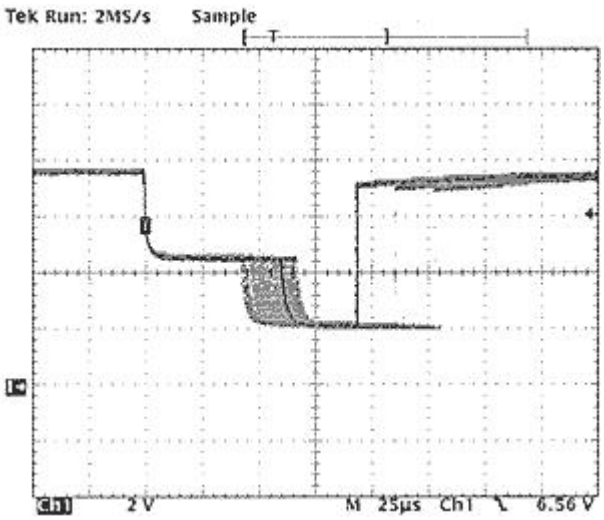


Fig. 3.17

Ya que el problema es causado por la sincronización entre el suministro y el umbral de amplificador fig.3.17, se puede aumentar la filtración de la fuente de energía, agregando un condensador de tántalo de 22 µF conectado en paralelo con C1 . Después de esta modificación, no se producen más cuentas erróneas.

3.2.2.3.Sensor de Contacto

El sensor de contacto, es un sensor digital a 9V calibrado para dar valores booleanos de cierto o falso.

Fig. 3.18

- Extraido de: http://philohome.free.fr/sensors/legorot.htm

3.2.2.4.Sensor de Temperatura

Temperature sensor

El sensor de tempertatura de LEGO es capaz de medir temperaturas comprendidas entre-20 y 50 grados Celcius . El RCX puede leer y mostrar la temperatura en grados Fahrenheit o Celsius.

·**Fig. 3.19**

Ejemplo:

Cuando colocamos un cubito de hielo en un vaso de agua, esta se enfría. Al alcanzar un nivel predeterminado, podríamos avisar de este cambio de temperatura conectando una lámpara o una alarma.

3.2.3.Sensores no fabricados por Lego

La Marca Lego, ofrece un número escaso de sensores y actuadores, debido a esta escasez , diferentes empresas han creado y comercializado sus propios sensores y actuadores. Al tener el RCX una arquitectura abierta de programación invita a los usuarios de Lego Mindstorms a crear libremente sus propios sensores y proyectos. Existe un gran oferta de paginas web, en las que se explica con detalle la elaboración de los mismos.

3.2.3.1.LEGO Dacta Sensores

LEGO Dacta en cooperación con DCP Microdevelopments Limited ofrecen los siguientes sensores que pueden ser usados con el sistema LEGO Dacta ROBOLAB.•

Para conectar los sensores DCP al RCX es necesario un cable adaptador.

•Extraido de: http://www.lego.com/eng/education/mindstorms/home.asp?menu=input&pagename=input

- Información adicional en: www.dcpmicro.com/lego
http://www.dcpmicro.com/lego/fr_sens.htm

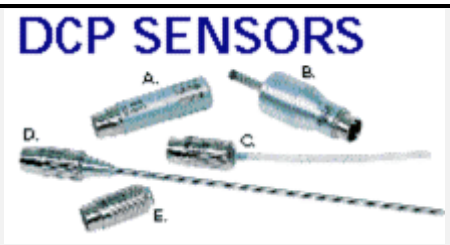
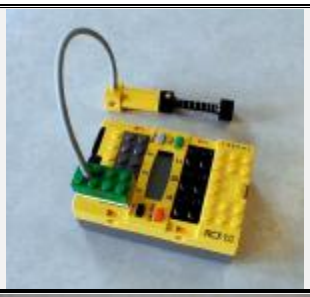
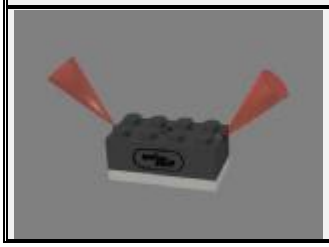
	 Cable adaptador.
A. **SENSOR DE HUMEDAD**	Rango: 0-100% humedad relativa en un rango de temperatura de los -20 grados Celsius a los +80 grados Celsius
B. **SENSOR DE PRESIÓN DE AIRE**	0-200kPa (aprox. 0-30 psi)
C. **SENSOR DE ROTACIÓN**	360 grados de rotación, precisión de 340 grados
D. **SENSOR DE TEMPERATURA**	-30 º, +130 º Celsius
E. **SENSOR DE SONIDO**	50 dBA -100 dBA; La respuesta frecuencial es una ponderación de la A.
SENSOR DE pH	Trabaja con electrodos pH con una salida de 59.1 mV por unidad de pH 25 º Celsius
SENSOR DE VOLTAGE	+/- 25 volts DC; resistencia sobre sondas: 410 Ohms

Tabla 3.20 Sensores LEGO Dacta

3.2.3.2. Techno-stuff Company

•Techno-stuff Company, es una de las empresas que se encarga de comercializar sensores totalmente compatibles con RIS:

 Sensor de Presión.	
 Detector dual de proximidad por infrarrojos.	

• Información extraida de: http://www.techno-stuff.com/sensors.htm

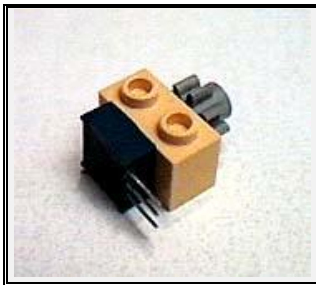
 Sensor de rotación.		
  Sensor de movimiento IR.		
 Sensor de Sonido.		

Tabla 3.21 Sensores Techno-stuff Company

3.2.3.3.Diseños propios

Es mucha, la gente que dedica parte de su tiempo al RIS y por ello son muchísimas las webs donde se describe con detalle la forma de elaborar de modo sencillo sensores "caseros", veamos un ejemplo:

·Sensor de luz

El material:

El único elemento electrónico que se necesita es un LDR. En la figura se puede ver el LDR y las dos piezas de LEGO necesarias (el LDR de la figura tiene la referencia MKY 76C348, y su precio es 2.48€+IVA).	

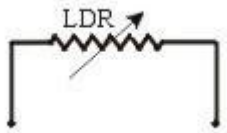
Fig. 3.22 Elaboración de un sensor de Luz con LDR

Un LDR es una resistencia con una característica muy particular: su valor depende de la intensidad de luz que incide en ella. Cuando la intensidad de la luz aumenta, el valor de la resistencia desciende, y viceversa. Una aplicación es la automatización de los sistemas de iluminación, de tal manera que al oscurecer se enciendan las luces.

Circuito

•Extraido de: http://www.donosgune.net/2000/gazteler/sensor.htm
Información adicional sobre sensores: http://www.hitechnicstuff.com/products.htm

El circuito es muy sencillo, y del mismo modo que sucede con el resto de resistencias no es necesario tener en cuenta la polaridad.



Montaje

Hay diferentes modos para montar este sensor. El más simple es cortar un cable de los utilizados para conectar motores ysensores, y soldar directamente el LDR a él. Otro es el que se puede ver en la figura. El LDR tiene sus terminales soldadas a los conectores de la pieza blanca inferior. De este modo, podremos conectar el sensor por medio de un cable estándar LEGO.

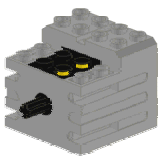
Nota:
Existe otro componente electrónico de semejante comportamiento al LDR, cuya resistencia varía con el cambio de temperatura. Con él es posible hacer un sensor de temperatura.

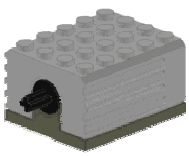
3.3.Actuadores

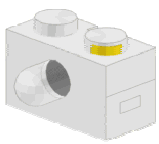
El RCX interactua con el ambiente a través de sus periféricos actuadores conectados a los puertos de salida del RCX, los siguientes actuadores están disponibles:

Actuadores	Ejemplos de respuesta
Motor	Movimientos rotatorios, movimientos lineales
Bombilla	Luz parpadeante, Luz roja de alarma
Altavoz	Sonidos Básicos

Vista de los elementos actuadores:







Salidas

Tabla 3.23 Actuadores

3.3.1.Motores

LEGO comercializa tres tipos de motores, de diferentes características:

Motor	Descripción	Velocidad sin carga	Torque	Par de arrancada
	Motor genérico Lego, ref. #5114.	1420 RPM 149 rad/s	1.0 N-cm 0.01 N-m 1.4 oz-in	0.37 N-m/s (Watts)
	Motor Reductor, ref. #5225.	350 RPM 37 rad/s	8.9 N-cm 0.089 N-m 12.3 oz-in 0.06 ft-lb	0.87 N-m/s (Watts)
	Micro Motor, ref. #5119.	30 RPM 3.1 rad/s	1.9 N-cm 0.019 N-m 2.6 oz-in	0.015 N-m/s (Watts)

Tabla 3.24 Características Motores

El motor incluido en el RIS, es el motor 5225 una combinación de motor y reductor. El reductor disminuye la velocidad del eje guía hasta una velocidad de 350 rpm, aumentando asi el par inicial.

Especificaciones:

- **El motor empieza a girar a partir de 1V.**
- **El voltaje máximo es de 9V.**
- **Velocidad nominal de 350 rpm.**
- **Consumo de corriente sin carga a velocidad nominal 5 mA.**
- **Consumo de corriente a velocidad nominal 350 mA**
- **Corriente de frenado 350 mA.**

Precauciones:

- **El motor no es impermeable.**
- **No exceder el voltaje máximo de 9 V.**
- **El motor no debería ser dejado en la condición de "parada" o bloqueo del motor por mucho tiempo.**

Usando el motor como generador:

Cuando se use el motor como generador , es conveniente no hacerlo girar a mayor velocidad de la que conseguiríamos con nuestras propias manos girando el eje.

Fig. 3.25 Motor 5225.

Conexión de los motores:

Con los cables de conexión podremos conectar motores o luces a los tres puertos de color negro del RCX: A, B y C. Se puede conectar un cable a un puerto de 4 formas distintas, de estas formas de conexión nos dependerá el sentido de giro de los motores.

Asimismo, se puede conectar el otro extremo del cable a un motor de 4 formas distintas.

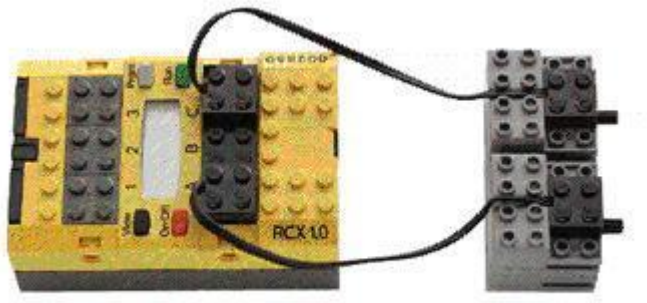


Fig. 3.26 Dos motores conectados al RCX.

El Motor-Reductor de LEGO es especial debido al engranaje interno del motor. Los engranajes están situados en el interior del motor en vez de por fuera. Es aplicable para usos en los que se exigen un alto momento de rotación a velocidades reducidas y con pocas pérdidas provocadas por la fricción. El Motor de Engranaje consigue un rendimiento del 80 %.

3.3.2.Lámpara

Ladrillo lampara.

La •lámpara de LEGO esta construida sobre un pequeño ladrillo de dos agujeros y se acopla perfectamente a cualquier otra pieza, Lego Mindstorms no la suministra con el kit.

La bombilla se controla mediante software de diferentes formas:

-Encender y apagar.
-Luz parpadeante durante un intervalo definido.
-Variación de la intensidad de luz.

• http://www.lego.com/eng/education/mindstorms/home.asp?menu=output&pagename=output

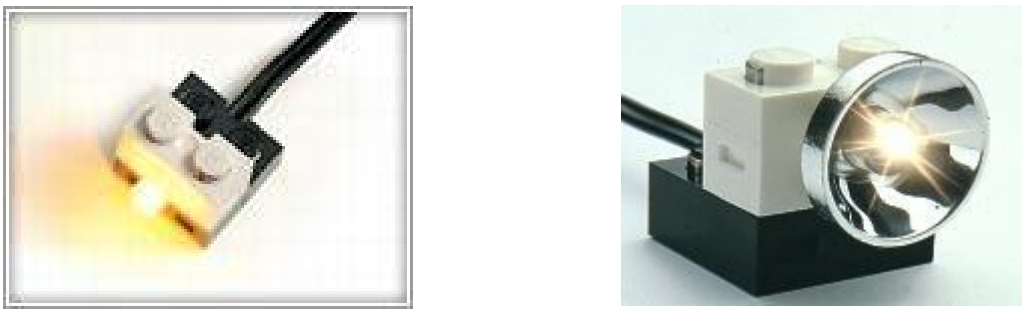


Fig. 3.27 Lámpara

3.3.3.Altavoz

Elemento de sonido(buffer)

El RCX incorpora internamente un altavoz, pero se puede incorporar como actuador externo un pequeño buffer.

El buffer se controla mediante software de diferentes formas:

- On y off;
- Producir dos tipos de sonidos diferentes;

Cuando se controla mediante software, es necesario girar el buffer hacia la derecha, para que se correspondan los sonidos con el software asociado. En la posición izquierda el buffer produce el sonido 'WEEWEE' conectado al canal de test del interfaz LEGO DACTA.

Fig. 3.28 Buffer

3.4.Transmisor/Receptor de infrarrojos

Características:

Diseño simple.
Apagado automático al cabo de 5 seg.
Uso inteligente de las señales de transmisión.

Funcionamiento:

Los datos del puerto serie son transmitidos por los leds de infrarrojos.
Los datos del RCX son recibidos por los leds de infrarrojos.

Partes del Transmisor/Receptor IR, empezando por la izquierda, la cubierta frontal, cubierta trasera y el circuito impreso, la cubierta frontal incorpora una ventanilla para filtrar infrarrojos.

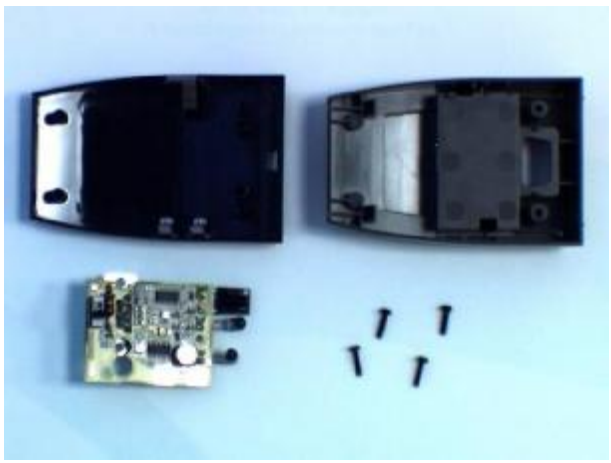


Fig. 3.29 Vista interior Transmisor

Distancia óptima entre el RCX y el transmisor de infrarrojos:

De 10 a 15 cm. También es posible comunicarse hasta unos 27 m aprox. usando el modo de larga distancia.

4. PROGRAMACIÓN

4.1. Introducción

El objetivo de esta parte del documento es mostrar las distintas opciones de programación alternativas al software de desarrollo que acompaña al kit.

Dado que el RCX es el núcleo de los robots, se ha considerado detallar su especificación técnica con el fin de conocer el funcionamiento interno del mismo y permitir mayor control a las aplicaciones desarrolladas.

Finalmente se ilustra el documento con numerosos ejemplos para mostrar las múltiples posibilidades de ambos kits de desarrollo. Como se podrá observar es posible construir sistemas de elevada complejidad, tanto funcional como estructural.

4.2. Introducción al LegoMindstorm

Los robots son más que simples juguetes. Comercialmente, los robots se utilizan hoy en día en campos que van desde la investigación médica a la exploración espacial. Aunque pueda parecer lo contrario, el Sistema de Invención Robótica de LEGO Mindstorms (LEGO Mindstorms Robotics Invention System) no es únicamente para niños. El cerebro de cada robot LEGO Mindstorms es el microcomputador RCX, dirigido por un procesador de 8 bits. El lenguaje oficial de programación del RCX fue diseñado teniendo en cuenta a los niños. Según Lego, «El código para RCX es un entorno de programación visual que permite a los niños coger, soltar y apilar comandos y trozos de código». Pese a ello, existen numerosos sistemas alternativos para la programación del RCX de una forma más avanzada y potente.

Para programar LEGO Mindstorms con el kit oficial se necesita al menos un computador personal con procesador Pentium (o compatible) a 90 MHz con el sistema operativo Windows 95 con una unidad lectora de CD. Con los sistemas de desarrollo no oficiales es posible programar el RCX desde otros sistemas operativos. Junto al Kit de lego tenemos un CD-Rom donde tenemos disponible el software para programar.

4.3. Herramientas de programación

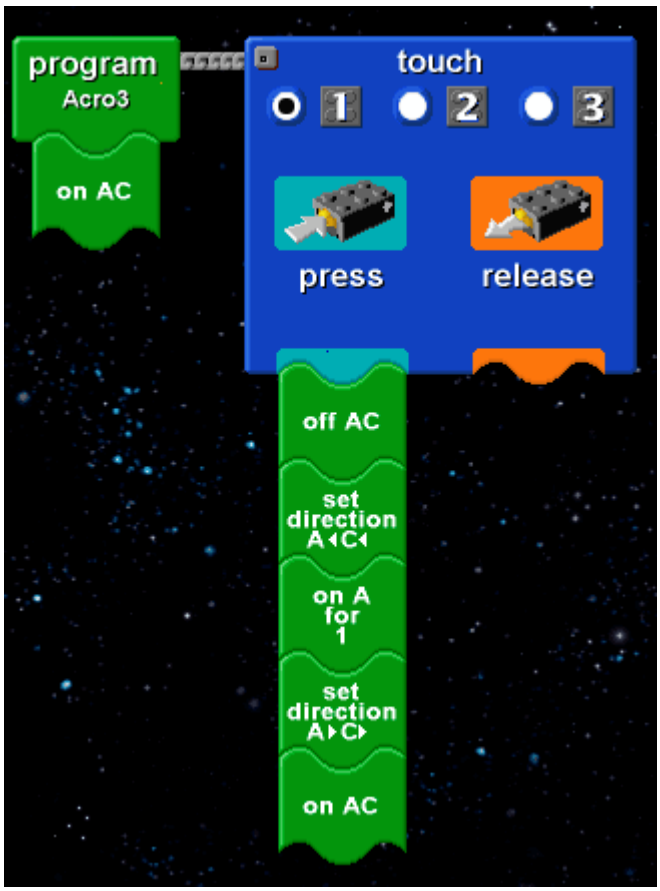
Cuando se instala el software que acompaña a LEGO MindStorms (versiones 1.0 ó 1.5) automáticamente se instala en el ordenador el control ActiveX **SPIRIT.OCX.** Este control premite controlar el RCX desde diferentes entornos de programación: Visual Basic, Visual C++, Delphi, Visual Java++... La versión 2.0 de LEGO MindStorms no contiene este control.

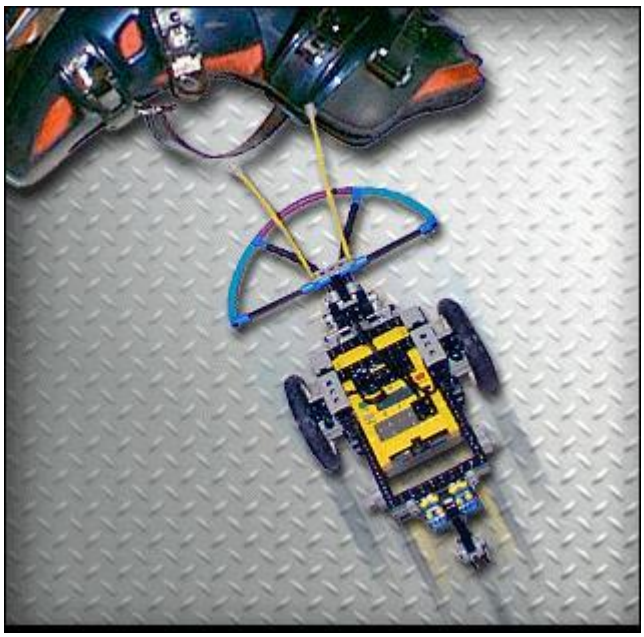
4.3.1. Código RCX

Código RCX es un entorno de programación para el RCX, el ladrillo programable de LEGO®. En el código RCX, cada bloque que se muestra en la pantalla representa una instrucción. Puedes crear un programa haciendo clic para seleccionar los bloques en la pantalla y después apilarlos uno debajo de otro.

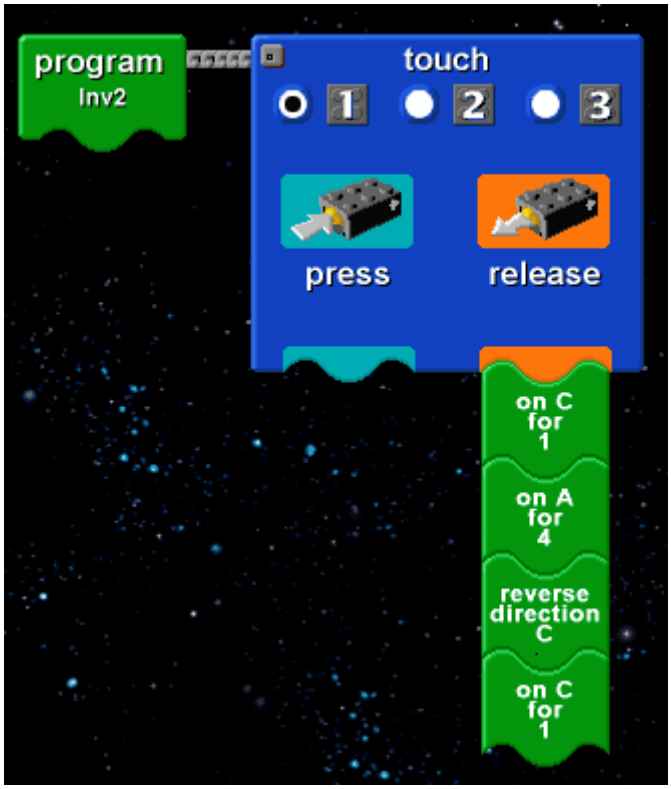
4.3.1.1.Programas en código RCX

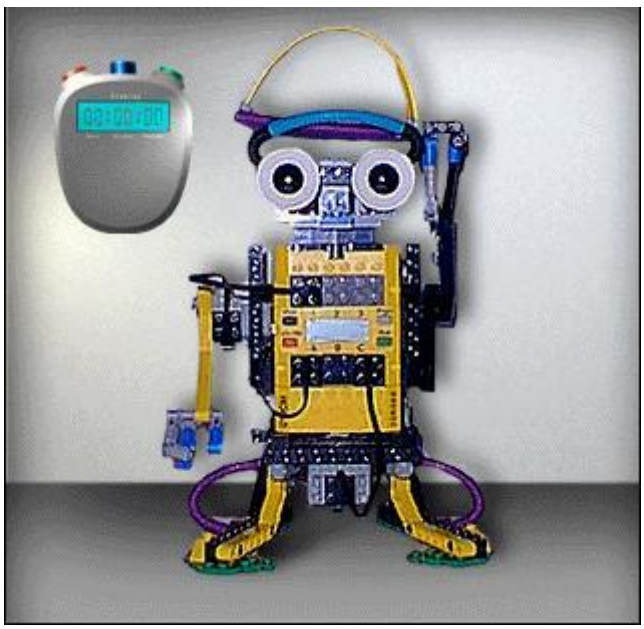
ACRO3. En este programa hacemos que el robot avance hacia adelante hasta que choque con un objeto. Después de chocar con un objeto, se da la vuelta durante 1 segundo y se gira hacia un lado. Finalmente, sigue avanzando de frente.



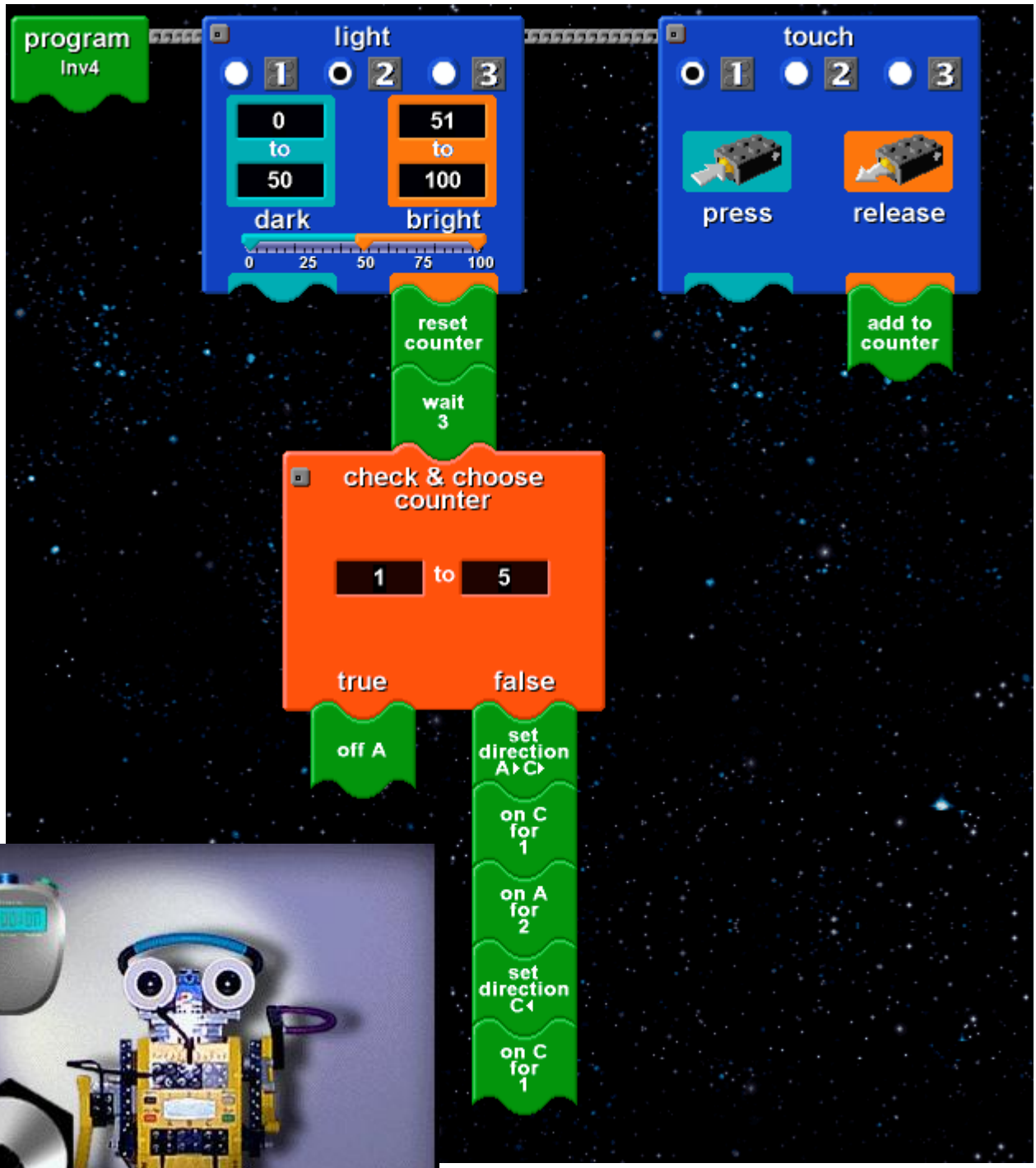


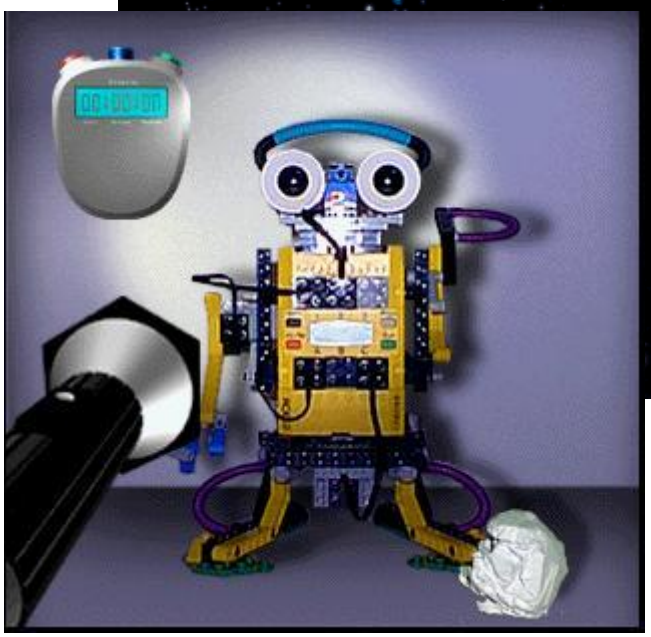
INVER2. Si aprietas o das un suave toque en la mano de este robot, se girará hacia un lado, se levantarà el sombrero y se volverá a girar de nuevo.





INV4. Cuando este robot detecta una luz brillante, espera durante tres segundos. Si durante ese periodo de tiempo no has tocado su mano para detenerlo, el robot se gira y dispara un bojeto con el brazo de lanzamiento. Después vuelve a colocarse en la posición inicial.





4.3.2.Sistemas que reemplazan el firmware

4.3.2.1. Programación con LegOS (brickOS)

El creador de **brickOS** fué *Markus L. Noga*. Él ha sido el que ha desarrollado la mayor parte del código, pero otra serie de personas han incorporado sus aportaciones. Es un software de libre uso bajo licencia Mozilla Public License, por lo que cualquier usuario o usuaria puede introducir los cambios que estime oportunos y añada sus aportaciones. El utilizar **C** en la programación del RCX ofrece muchas ventajas ya que combinandolo con **brickOS** proporciona flexibilidad, potencia y efectividad. El aspecto negativo de su uso es que si no se tienen conocimientos de **C** su aprendizaje es más lento.

LegoOS es un sistema operativo multitarea al estilo **POSIX** para el Robotic Invention System. Los programas se escriben en **C + C++** estándar compilados en un PC utilizando el gcc (construidos mediante compilación cruzada), y transmitidos al RCX donde son ejecutados. Básicamente, cualquier cosa que se pueda escribir en **C + C++** (en 32Kde RAM) puede ser escrito en legOS. Algunas características interesantes son la función random, emulación de números con coma flotante, gestión de hebra con semáforos POSIX, y la capacidad de almacenar varios programas. También incluye funcionalidad para enviar y recibir datos desde PCs con Linux y MS Windows. Toda esta potencia tiene un pequeño coste: debido a que usa gcc, legOs es el sistema con una configuración más complicada, y requiere bajar de Internet grandes herramientas.

La estructura del programa es equivalente que en NQC, pero en esta ocasión sólo se utiliza una tarea que espera en un bucle infinito a que se produzca un evento, la pulsación del sensor, para realizar el giro.

El entorno de programación bajo GNU/Linux incluye el compilador de C de GNU (gcc) compilado como cruzado para el Hitachi H8, para lo que hace falta usar las binutils. La distribución para GNU/Linux de LegOS incluye varias herramientas que permiten descargar el código de forma dinámica, descargar el firmaware o sistema operativo, así como varios ejemplos.

Además, alrededor del sistema operativo LegOS se han desarrollado múltiples herramientas auxiliares, como por ejemplo simuladores que hacen más fácil la depuración al permitir ejecutar programas en la propia plataforma de desarrollo usando un depurador tradicional de GNU/Linux como por ejemplo gdb.

Ofrece muchas ventajas además de mejores prestaciones y mayor flexibilidad:

- La carga dinámica de programas y módulos.
- El protocolo de comunicación basado en el transmisor infra-rojo.
- La posibilidad de realizar programas multitarea.
- La gestión de memoria dinámica.
- La existencia de drivers para todos los subsistemas del ladrillo.
- El uso de la velocidad nativa del micro-procesador, esto es 16 MHz.
- El acceso a los 32K de memoria RAM. Permitir el uso completo del lenguaje de programación elegido, como por ejemplo C. Lo cual implica que se pueden usar punteros, estructuras de datos, etc.

Por ejemplo :

```
#include <conio.h>
#include <unistd.h>
#include <dsensor.h>
#include <dmotor.h>

/*Declaro una función que se ejecutará al detectar la colisión*/
wakeup_t colision(wakeup_t dato);

int main(int argc, char *argv[]) {
  int dir=0;

  while(1) {
    /* arranco
       - Fijo velocidad*/
    motor_a_speed(MAX_SPEED);
    motor_c_speed(MAX_SPEED);
    /* - Avanzo*/
    motor_a_dir(fwd);
    motor_c_dir(fwd);

    /* Espero */
    wait_event(&colision,0);
    /* Me apunto en que lado fue la colisión*/
    if(SENSOR_1<0xf000)
      dir=0;
    else
      dir=1;

    /* reculo */
    motor_a_dir(rev);
    motor_c_dir(rev);

    /* - el ratito que reculo*/
    msleep(500);

    motor_a_speed(MAX_SPEED);
    motor_c_speed(MAX_SPEED);

    /* Una vez reculado, ahora giro un pelín*/
    if(dir==1) {
      motor_c_dir(fwd);
    } else {
      motor_a_dir(fwd);
    }
    /* - ratito para girar */
    msleep(500);
  }
}

wakeup_t colision(wakeup_t dato) {
       lcd_refresh();
       /* Los sensores están conectados a los puertos 1 y 2 */
       return SENSOR_1<0xf000 || SENSOR_2<0xf000;
}
```

4.3.2.2. pbForth

pbForth es un interprete completo del lenguaje de programación Forth que reemplaza el firmware por defecto. Una vez que se realiza el reemplazo, se pueden transmitir los programas en Forth al robot y el interprete los interpretará y ejecutará. No hay limitación en el número de variable y existen varias librerías que proporcionan funcionalidad adicional; como por ejemplo depuración interactiva. La configuración es muy sencilla, ya que únicamente se requiere la transferencia del binario. No hay otras herramientas, compiladores o intérpretes en el PC; aunque existe un entorno de desarrollo disponbile que simplifica la transferencia de los programas y la interacción con el PC.

4.3.2.3. TinyVM and leJOS

Como su nombre indica, TinyVm es una pequeña máquina virtual de Java que se transmite al RCX para reemplazar el firmware estándar. Los programas en Java se escriben y compilan a byte-code en el PC, tras lo que son transmitidos al RCX. Un programa de TinyVm puede usar algunas librerías estándar de Java, y una librería para el control de los sensores, motores, etc. TinyVM necesita, por supuesto, un compilador de Java que funciones.

LeJOS es un proyecto similar del mismo autor. Es más grande (5K adicionales de tamaño) pero también incluyen una funcionalidad substancial, como por ejemplo, soporte para variables en coma flotante y constantes de tipo String. Otras características planeadas son recolección de memoria no utilizada y almacenamiento de varios programas.

4.4.LIBRERIAS DE CONTROL REMOTO

4.4.1.spirit.ocx

spirit.ocx es un control ActiveX que se suministra conjuntamente con los LEGO Mindstorms y que se utiliza para comunicarse con el RCX. Al tratarse de un control basado en ActiveX es muy sencillo utilizarlo desde aplicaciones desarrolladas en Visual Basic o Visual C++.

4.4.1.1. Utilizando el control spirit.ocx desde Visual Basic

La utilización del control desde un proyecto en Visual Basic es muy sencilla, lo primero que se debe hacer es añadir este control a los disponibles para el proyecto. Para ello se selecciona en el menú Proyecto -> Añadir al proyecto -> Componentes y controles. Aparecerá una ventana de dialogo con una sección de "Controles ActiveX Registrados" dentro de esta sección se buscara el control "Spirit Control". Una vez seleccionado se pulsará el botón "Insertar" que cerrará la ventana de dialogo.

A partir de ahora el icono de LEGO deberá aparecer en la barra de herramientas y una nueva clase llamada CSpirit se habrá creado. Para añadir uno de estos controles a nuestra aplicación simplemente se inserta el control en la ventana de la aplicación desde la barra de herramientas y en las propiedades del mismo se indica que no sea visible.

Desde el inspector de clases se añade una variable de tipo CSpirit con la que se podrá acceder a las funcionalidades del control desde el código del programa.

EJEMPLO 1. INICIALIZANDO LAS COMUNICACIONES

Dado que el control spirit.ocx también se utiliza para controlar el CyberMaster Technic kit, primero se deben configurar unas pocas variables: El puerto de comunicaciones, el tipo de enlace y el "brick type".

```
m_cRCX.SetComPortNo(1); // COM port 1
m_cRCX.SetLinkType(0); // 0 = IR
m_cRCX.SetPBrick(1); // 1 = RCX, 0 = CyberMaster
m_cRCX.InitComm(); // Set up the comms
// Is the RCX responding?
if (!m_cRCX.PBAliveOrNot()) {
// ...
// Handle RCX Problem
// ...
}
```

EJEMPLO 2. ACCIONANDO LOS MOTORES

Es muy sencillo controlar los motores desde el control Spirit. Todos los procedimientos relativos a los motores admiten un único parámetro en forma de cadena. Ciertamente es una forma peculiar de indicar los motores, realmente la rutina extrae de la cadena los dos primeros números comprendidos entre 0 y 2.

```
m_cRCX.SetFwd("Motor 0 and Motor 2");
m_cRCX.On("02");
Sleep(100); // CSpirit::Wait not used,
m_cRCX.Off("02"); // since it is only downloadable.
```

EJEMPLO 3. UTILIZANDO LOS SENSORES

Antes de ver como se trabaja con sensores resulta conveniente realizar las siguientes definiciones que nos permitirán una mayor comprensión del código:

```
#define RCX_VARIABLE 0
#define RCX_CONSTANT 2
#define RCX_SENSOR 9
#define RCX_SENSOR_1 0
#define RCX_SENSOR_2 1
#define RCX_SENSOR_3 2
#define RCX_GREATER 0
#define RCX_LESS 1
#define RCX_EQUAL 2
#define RCX_NOTEQUAL 3
```

El acceso a los sensores se realiza mediante dos procedimientos: CSpirit::if y CSpirit::While. Estas funciones reciben 5 parámetros. Los dos primeros son el primer operando de la comparación en la forma tipo y valor. El tercero es el operador de comparación. Finalmente, el cuarto y quinto forman juntos el segundo operando de la comparación, también en la forma tipo y valor.

Por ejemplo, para comparar si el valor del sensor 3 es igual a 4:

```
bool sensor3 = m_cRCX.If(RCX_SENSOR, RCX_SENSOR_3, RCX_EQUALS, RCX_CONSTANT, 4)
```

Si se desea iterar hasta que el sensor1 no sea igual a 1 (sensor de contacto):

```
m_cRCX.While(RCX_SENSOR, RCX_SENSOR1, RCX_NOTEQUAL, RCX_CONSTANT,1);
//...
m_cRCX.EndWhile();
```

4.4.2. Lego::RCX.pm

Lego::RCX.pm es una librería del lenguaje de programación PERL para el control remoto vía RCX de la torre de infrarrojos. Toma el control de la torre de infrarrojos y manda comandos que el firmware estándar puede interpretar y ejecutar. Si ya se dispone de PERL instalado es sencillo y rápido utilizar este módulo. No es necesaria ninguna instalación especial; simplemente hay que copiar los archivos en los directorios de las librerías de PERL y añadir "use RCX.pm" en el inicio del programa en PERL.

4.4.3. Remote Java APIs

Existen unas librerías (teóricamente) multiplataforma de **Java** que envían señales que pueden ser usada para controlar remotamente un RCX usando el firmware original. Actualmente no se encuentran en desarrollo y no han sido muy probadas.

Aqui podemos ver un ejemplo de como usar el RCX Java API para controlar un Robot Lego Mindstorm:

```
import rcx.*;
public class Test
	{
		public static void main(String[ ] arg)
		{
		    new Test(arg[0]);
		}
		public Test(String portname)
		{
		    RCXPort port = new RCXPort(portname);
		    Motor.A.forward(  );
		    Motor.A.stop(  );
		    port.beep(  );
		    System.out.println( "Battery Level = "+port.getBatteryPower()+" volts.");
		}
	}
```

4.4.4. Pylnp

La librería **Pylnp** es bastante peculiar, no sólo por usar el lenguaje **Python**, sino porque en lugar de interactuar con el firmware estándar, interactúa con el de **legOs**. Debido a esto, proporciona una gran potencia de personalización que otras librerías para el control remoto del robot no ofrecen, ya que puede ser utilizado para invocar funciones propias cargadas en la parte de **legOS** almacenada en el robot. El inconveniente de esta potencia, por supuesto, es que hay que aprender **legOS** y hay que escribir programas en **C** para acceder a esta funcionalidad superior.

4.5. Diferentes entornos de programación

4.5.1. NQC , Not Quite C

El acrónimo NQC significa Not Quite C. Se trata de un simple lenguaje con una sintaxis muy similar a C que se puede usar para programar el RCX. Es, desde luego, la alternativa más sencilla al lenguaje de programación basado en iconos arrastrables que proporciona LEGO.

NQC se diseñó para ser muy simple y para ser utilizable por personas con limitados conocimientos de programación. Todo ello se ha conseguido, pero a costa de una serie de limitaciones, de entre las cuales las más importantes son:

- Las subrutinas no admiten parámetros, por lo que no es posible emplear realmente los principios de la programación estructurada.
- Sólo se pueden usar variables globales, es decir, el espacio de nombres es único, lo cual complica enormemente el desarrollo y mantenimiento de programas complejos.
- Las subrutinas no pueden devolver valores, lo cual las limita a ser subconjuntos de código, sin poder aportar funcionalidades nuevas ocultando su implementación.
- El número de variables está enormemente limitado, de hecho sólo se dispone de 32.
- No existen las estructuras de datos, ni las estáticas ni las dinámicas, lo que imposibilita casi cualquier tipo de aproximación que necesite almacenar cualquier tipo de estado.

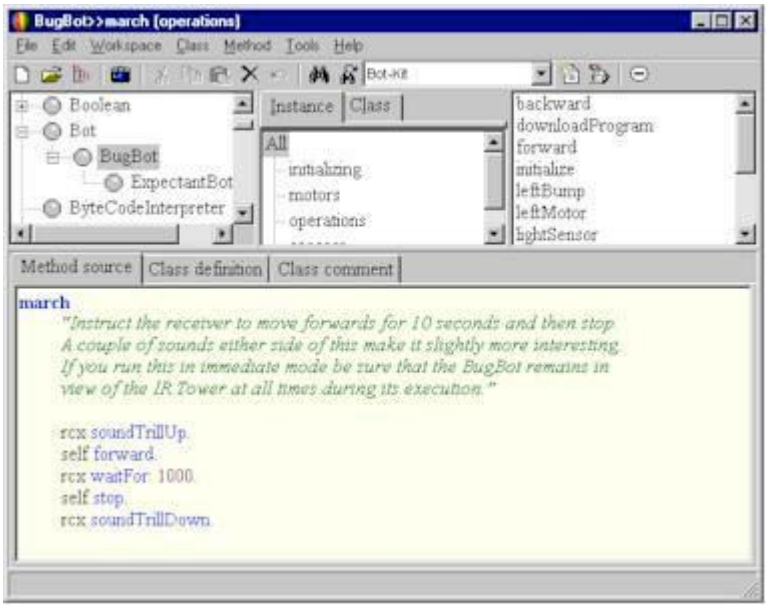
A pesar de estas limitaciones, NQC es un lenguaje muy popular entre los usuarios de baja formación informática, debido sobre todo a su simplicidad, pero también a la abundante documentación y a la existencia de herramientas como el ***RCX Command Center*** para MS-Windows que facilitan el desarrollo y la descarga de programas. En el caso de GNU/Linux, se dispone de un compilador que produce código directamente descargable en el robot, pudiendo utilizarse como editor cualquiera que edite texto ASCII, como por ejemplo **Emacs**.

4.5.2. Bot-Kit

Bot-Kit es un sistema de control para robots de LEGO que permite programarlos utilizando el lenguaje Dolphin Smalltalk. La aplicación permite controlar tanto robots Mindstorm como Cybermaster.

Como se ha dicho se utiliza el sistema de desarrollo Dolphin Smalltak que proporciona el entorno de programación y la posibilidad de utilizar un lenguaje fuertemente orientado a objetos como es Smalltalk.

Con Bot-Kit se ofrece la posibilidad de controlar los robots en dos modos. En el modo inmediato se interactúa con el robot a través del enlace de infrarrojos enviando comandos y recibiendo la información de los sensores. Y el modo programa en el que se compila un programa en Smalltalk y se descarga en el RCX de forma que el robot pueda operar independientemente del PC.



Codigo Bot-Kit en el navegador del Dolphin Smalltalk

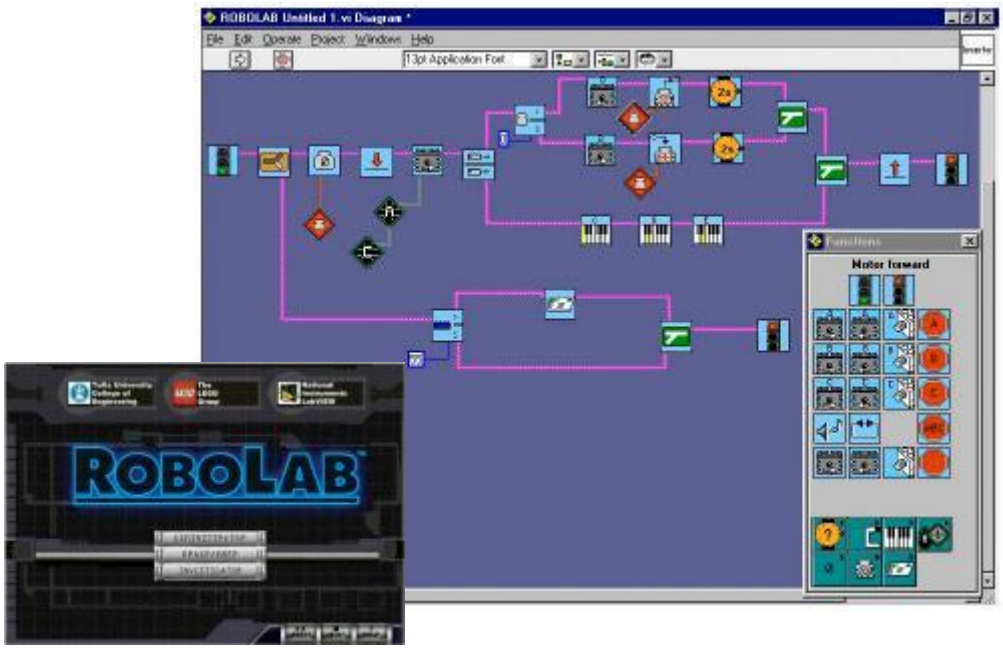
4.5.3. Robolab

ROBOLAB es un entorno de desarrollo disponible para PC y Mac desarrollado por Center for Engineering Educational Outreach en colaboración con Tufts CEEO, LEGO DACTA, y National Instruments con un enfoque didáctico.

Precisamente este enfoque es lo que lo diferencia del resto de aplicaciones aquí mencionadas y lo aproxima mas al entorno original de desarrollo que acompaña al Robot Invention System. Se trata de un entorno de programación grafico en el que los comandos se representan mediante iconos que se encadenan unos con otros. Bajo el mismo concepto de sencillez que el ofrecido por el entorno del RIS, ROBOLAB ofrece mucha mas funcionalidad al añadir numerosos comandos y posibilidad de combinarlos entre ellos.

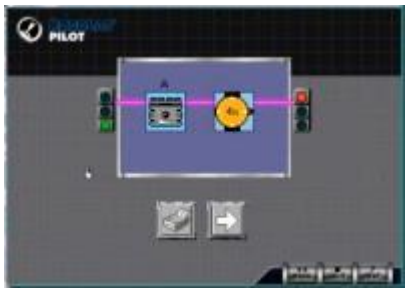
ROBOLAB ofrece modos diferentes de programación adaptados al nivel de aprendizaje del alumnado: **Pilot** e **Inventor**. Además, ofrece el modo **Investigator** orientado a su uso en el laboratorio de ciencias.

El entorno de ROBOLAB:



Pilot es el nivel básico. Por medio de una serie de plantillas introduce a niños y niñas en la lógica de la programación. Estas plantillas están protegidas, por lo que no pueden ser alteradas. Las modificaciones que se pueden hacer son pocas, pero se asegura que los programas siempre funcionarán, por lo que en muy poco tiempo pueden conseguirse resultados.El modo **Pilot** consta de cuatro niveles con dificultad creciente que abren el camino a utilizar **ROBOLAB**.

El entorno de ROBOLAB PILOT:

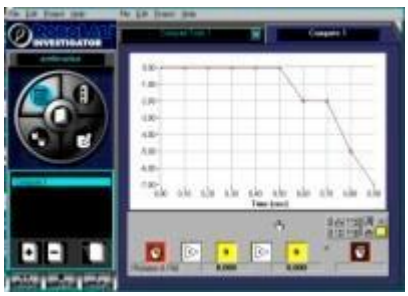


El modo **Inventor** constituye la segunda fase del aprendizaje. Usuarios y usuarias desarrollarán sus propios programas distribuyendo en la ventana de diagramas una serie de iconos. Inventor consta de cuatro niveles. Las diferencias entre ellos se centran en las opciones que ofrece cada uno: el cuarto nivel ofrece más herramientas de programación que el primetro.El último nivel de **Inventor** permite utilizar toda la potencia de ROBOLAB para desarrollar complicadas aplicaciones.

El entorno de ROBOLAB INVENTOR:

El modo **Investigator** está diseñado para ser utilizado en el laboratorio de ciencias. Utiliza para ello una versión adaptada de LabVIEW. Convierte el RCX en una interesante herramienta de trabajo en aquellas experiencias que requieren recoger datos. Si utilizamos el RCX para recoger datos (por ejemplo, la evolución de la temperatura en el aula a lo largo de la noche), Investigator nos ayudará a procesar dichos datos y a presentarlos. Además, permite editar por completo el informe de la experiencia, para imprimirlo a continuación, o si así se desea, convertirlo en una presentación por ordenador o en una página Web.

El entorno de ROBOLAB INVESTIGATOR:



Capacidades reales del RCX en función del Firmware:

Firmware de Lego	Firmware LegOS
Las subrutinas no admiten parámetros, por lo que no es posible emplear realmente los principios de la programación estructurada.	La carga dinámica de programas y módulos.
Sólo se pueden usar variables globales, es decir, el espacio de nombres es único, lo cual complica enormemente el desarrollo y mantenimiento de programas complejos.	El protocolo de comunicación basado en el transmisor infra-rojo.
Las subrutinas no pueden devolver valores, lo cual las limita a ser subconjuntos de código, sin poder aportar funcionalidades nuevas ocultando su implementación.	La posibilidad de realizar programas multitarea.
El número de variables está enormemente limitado, de hecho sólo se dispone de 32.	La gestión de memoria dinámica.
No existen las estructuras de datos, ni las estáticas ni las dinámicas, lo que imposibilita casi cualquier tipo de aproximación que necesite almacenar cualquier tipo de estado.	La existencia de *drivers* para todos los subsistemas del ladrillo.
	El uso de la velocidad nativa del micro-procesador, esto es 16 MHz.
	El acceso a los 32K de memoria RAM.
	Permite el uso completo del lenguaje de programación elegido, como por ejemplo C. Lo cual implica que se pueden usar punteros, estructuras de datos, etc.

Tabla 4.1. Tabla comparativa de capacidades reales del RCX en función del Firmware.

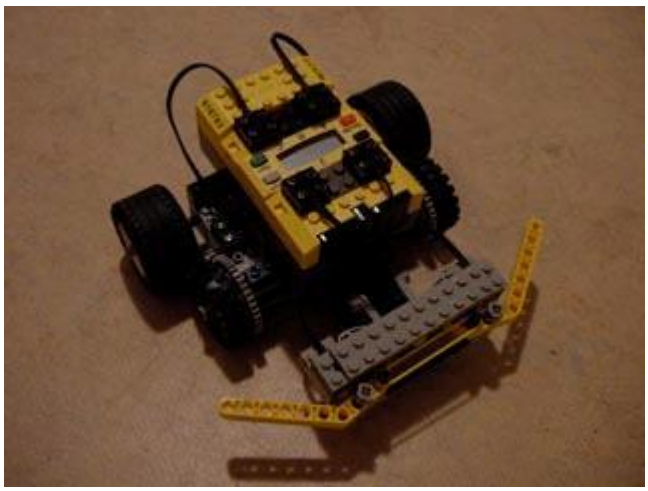
Extraida de:

http://gul.uc3m.es/gul/docs/iiicongreso_hispalinux/lego/hispalinux2000.pdf

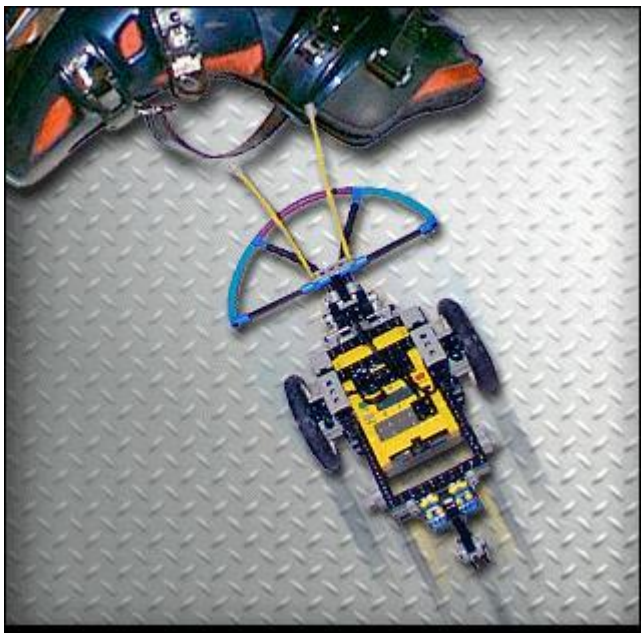
5. ROBOTS

PATHFINDER

ROVERBOT



ACROBOT



INVENTORBOT

Robots Mindstorm

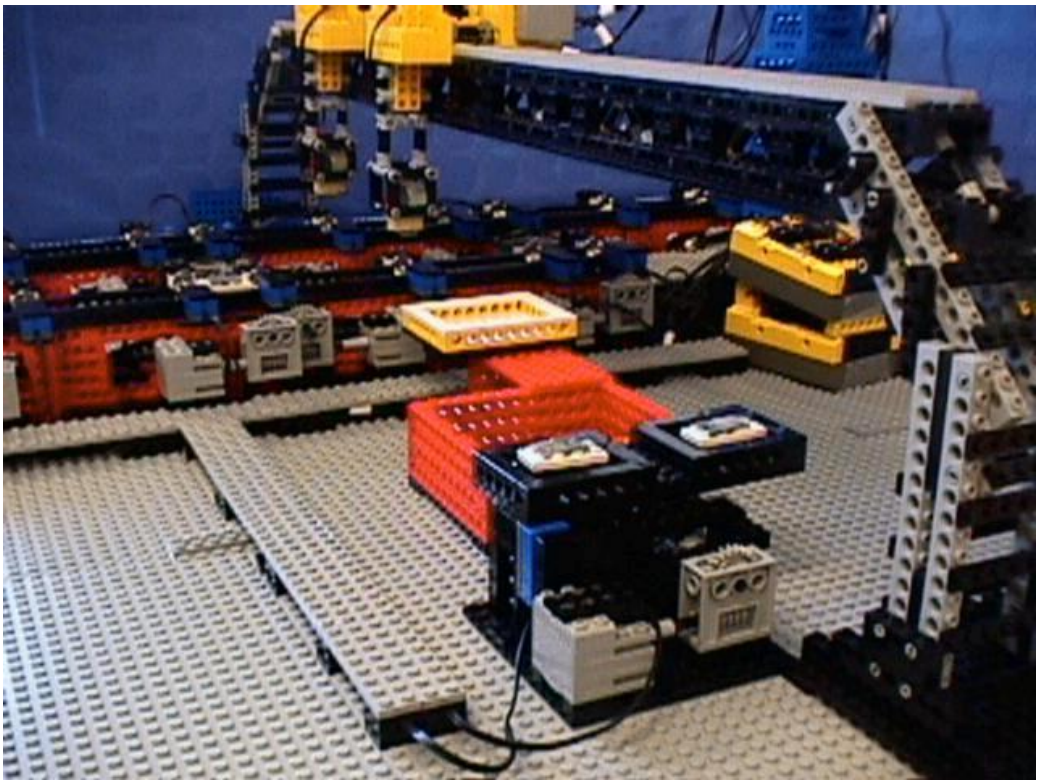
Tipo	Rotación	Lineal	Aceleración	Pendientes	Rastreo	Progr.	Electrónica	Veloc.	Mec.	Estructura	r
Pathfinder	Si	Si	Media	Medio 30%	No	Facil	Rcx 2motores 1 sensor	Alta	Simple	Debil	1:1
Roverbot con ruedas	Si	Si	Nula	Excelente 60%	No	Facil	Rcx 2motores 1 sensor	Redu cida	Media	Consis tente	5:1
Roverbot con patas	Si	Si	Nula	No	Si	Medio Facil	Rcx 2motores sensor luz	Redu cida	Media	Consis tente	5:1
Roverbot oruga	Si	Si	Nula	Excelente 60%	Si	Facil	Rcx 2motores sensor luz	Lenta	Buena	Consis tente	3:1
Acrobot	Si	Si	Si	Medio 30%	Si	Facil	Rcx 2motores sensor luz	Alta	Media	Muy buena	1:1
Acrobot con sensor de luz y choque	Si	Si	Si	No	Si	Se puede poner sub rutinas	Rcx 2motores sensor luz y choque	Alta	Buena	Rob- usta	1:1
Inventor bot	Si	Posible sin rotar	-	-	-	Medio	Rcx 2motores sensor luz	-	Muy buena	Rob- usta	Tronco 3:1 (Cadena de engranajes) Cadera 5:1 correa

Tabla 5.1.

6.Ejemplos

6.1.Modelado de una fábrica de acero

Una planta de producción de acero, es un experimento con el modelo físico utilizando piezas de LEGO. Se utiliza el RCX como mini-automatas temporizados para controlar la planta y simular las posibles situaciones.

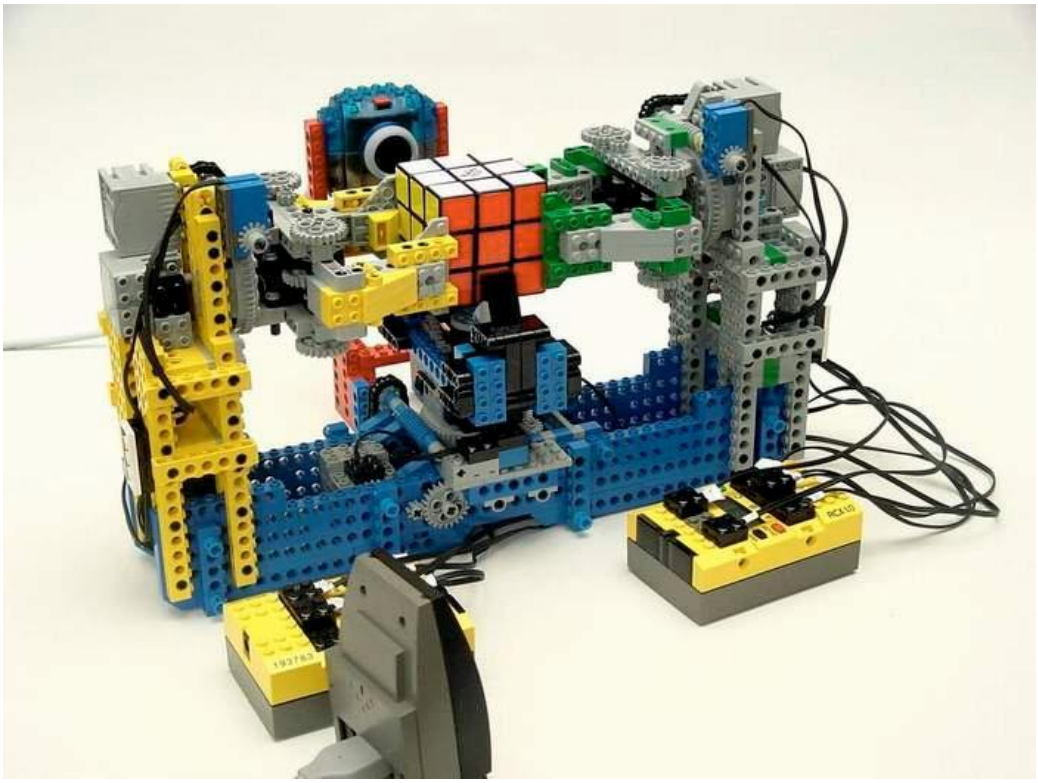


6.2. Robot bipedo

Los robots andadores también tienen grandes seguidores; son habituales criaturas con dos, cuatro o seis patas. Este tipo de robots son interesantes por sus similitudes con los animales, más que por poder realizar alguna tarea que no puedan desempeñar sus equivalentes con ruedas. Lo que realmente haría evolucionar a las creaciones con ruedas más allá de la consideración de juguetes sería la posibilidad de adaptar su modo de andar según el terreno por el que transitasen. La mayoría de los caminantes bípedos se limitan a repetir el mismo movimiento incesantemente. Algunos pocos son capaces de realizar giros.

6.3.Cubo de rubic

Este complejo sistema es capaz de encontrar las combinaciones necesarias para dar con la solución, utilizando dos RCX y una web-cam:



6.4. Robot hexápodo

A pesar de esta variedad de opciones, no han sido utilizadas, ni mucho menos, todos los posibles tipos de locomoción; aunque se están probando continuamente nuevas formas de movimiento. Existen robots serpenteantes que se extienden y contraen para moverse.

6.5.Robot escalador

7. ANEXO

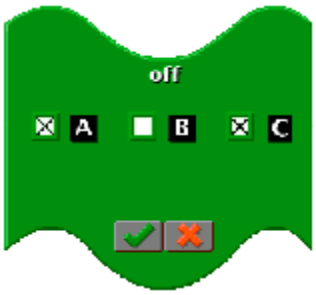
ANEXO A. CÓDIGO DE PROGRAMACIÓN RCX

A.1. COMANDOS EN CÓDIGO RCX

A.1.1.Off (“apagar”)

Utiliza el bloque Off para apagar los motores u otros dispositivos conectados a los puertos seleccionados:

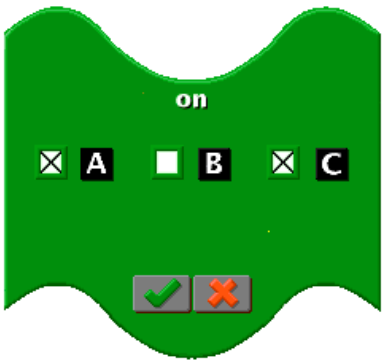
Haz clic con el botón derecho en el bloque Off para abrirlo. Haz clic en la letra o casilla de un puerto para activarlo o desactivarlo; una X en la casilla indica que el puerto correspondiente está seleccionado. Los motores conectados a los puertos con las casillas marcadas (A, B o C) se apagarán. Al terminar, haz clic en el cuadro que contiene la marca de verificación. La parte ampliada del bloque Off:



A.1.2.On (“encender”)

Utiliza el bloque On para encender los motores u otros dispositivos conectados a los puertos seleccionados:

Haz clic con el botón derecho en el bloque On para abrirlo. Haz clic en la letra o casilla de un puerto para activarlo o desactivarlo; una X en la casilla indica que el puerto correspondiente está seleccionado. Los motores conectados a los puertos con las casillas marcadas (A, B o C) se encenderán y permanecerán así hasta que los programes para que se apaguen. Al terminar, haz clic en el cuadro que contiene la marca de verificación. La parte ampliada del bloque On:



A.1.3. Add to Counter (“añadir al contador”)

El contador es un sensor interno del RCX que cuenta los eventos, por ejemplo, cada vez que una condición se cumpla (como podría ser que se toque un sensor táctil). Utiliza el bloque add to counter para agregar 1 al contador RCX.

A.1.4. On For (“encender durante”)

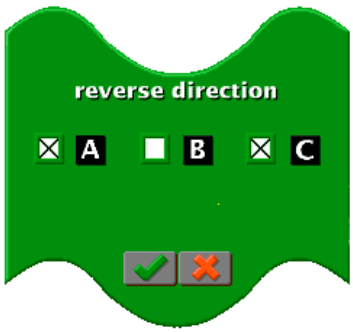
El bloque On for enciende los puertos A, B y C durante un período de tiempo determinado. Puedes conectar motores, luces u otros elementos a estos puertos. El bloque On for:

Haz clic con el botón derecho en el bloque On for para abrirlo. Haz clic en la letra o casilla de un puerto para activarlo o desactivarlo; una X en la casilla indica que el puerto correspondiente está seleccionado. En el campo de introducción de números, utiliza el signo + o -, o escribe el tiempo en segundos que deseas que el motor u otro elemento esté encendido. Puedes escribir un número entre 0,1 y 327,6. Puedes hacer clic en el dado para elegir una cantidad de tiempo aleatoria entre 0,1 segundos y el número que se muestra en el campo. Para volver al valor anterior, haz clic de nuevo en el dado. Al terminar, haz clic en el cuadro que contiene la marca de verificación. La parte ampliada del bloque On for:

A.1.5. Reverse Direction (“invertir dirección”)

El bloque Reverse direction permite cambiar la dirección de los motores conectados a los puertos seleccionados. No importa en qué dirección esté girando el motor, el bloque Reverse direction hace que gire en la dirección opuesta. El bloque Reverse direction:

Haz clic con el botón derecho en el bloque Reverse direction para abrirlo. Haz clic en la letra o casilla de un puerto para activarlo o desactivarlo; una X en la casilla indica que el puerto correspondiente está seleccionado. Al terminar, haz clic en el cuadro que contiene la marca de verificación. La parte ampliada del bloque Reverse direction:



A.1.6.Reset Counter (“restablecer contador”)

El contador es un sensor interno del RCX que cuenta los eventos, por ejemplo, cada vez que una condición se cumpla (como podría ser que se toque un sensor táctil). El contador se pone a 0 cada vez que se inicia la ejecución de un programa. Puedes utilizar el bloque Reset counter en cualquier otro lugar del programa para restablecer el contador RCX a 0. Por ejemplo, si tienes un contador Check and choose o un controlador de sensor Counter que cuenta los eventos en el programa, el bloque Reset counter pone el contador de nuevo a 0. El bloque Reset counter:

A.1.7.Reset Message (“restablecer mensaje”)

El bloque Reset message borra los números de mensaje recibidos previamente de otro RCX. Es necesario borrar el mensaje anterior para que el RCX pueda recibir un mensaje nuevo. Por ejemplo, es conveniente utilizar el bloque Reset message en los controladores de sensor o de secuencias que responden a mensajes. El bloque Reset message:

A.1.8.Reset Rotation (“restablecer giro”)

El sensor de giro realiza un seguimiento del giro de un eje que se efectúa a través del sensor. El sensor de giro se pone a 0 cada vez que el RCX inicia la ejecución de un programa. Puedes utilizar el bloque Reset rotation en otro punto del programa para volver a poner el sensor de giro a 0. El bloque Reset rotation:

Haz clic con el botón derecho en el bloque Reset rotation para abrirlo. Para elegir el puerto de sensor apropiado, haz clic en la casilla que se encuentra junto al sensor de giro 1, 2 ó 3. Si tienes varios sensores de giro conectados, puedes elegir más de un puerto a la vez. La parte ampliada del bloque Reset rotation:

A.1.9.Reset Timer (“restablecer temporizador”)

El temporizador RCX interno parte de 0 cada vez que el RCX inicia la ejecución de un programa y cuenta los segundos que se ejecuta el programa. Puedes utilizar el bloque Reset timer en otro punto del programa para volver a poner el temporizador interno a 0 y reiniciar el recuento del tiempo: El bloque Reset timer:

A.1.10. Send mesage (“Enviar mensaje”)

El bloque Send to RCX indica al RCX que envíe un mensaje de infrarrojos a otro RCX:

Haz clic con el botón derecho en el bloque Send to RCX para abrirlo. El mensaje puede ser cualquier número entre 1 y 255. Escribe un número en el campo de introducción de números o utiliza el signo + o - para seleccionar un número. Puedes hacer clic en el dado para elegir un número al azar entre 1 y el número que se muestra en el campo. Al terminar, haz clic en el cuadro que contiene la marca de verificación. La parte ampliada del bloque Send to RCX:

A.1.11. Set Direction (“establecer dirección”)

La primera vez que se activan los puertos A, B o C, los motores conectados giran en una dirección determinada. Siempre que no se haya invertido el cable que conecta el motor al RCX, esta dirección se indica en el bloque Set direction con una flecha que apunta a la derecha. Una flecha hacia la izquierda indica la dirección contraria. El bloque Set direction:

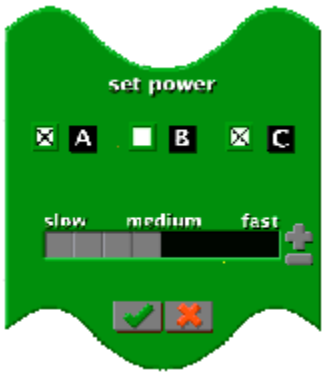
Haz clic con el botón derecho en el bloque Set direction para abrirlo. El bloque Set direction permite elegir la dirección de giro de los motores conectados a los puertos seleccionados. Haz clic en la letra o casilla de un puerto para activarlo o desactivarlo; una X en la casilla indica que el puerto correspondiente está seleccionado. Haz clic en la flecha o en el botón para elegir la dirección.

Al terminar, haz clic en el cuadro que contiene la marca de verificación. La parte ampliada del bloque Set direction:

A.1.12.Set Power (“establecer potencia”)

Si agregas el bloque Set power a una secuencia, podrás cambiar la potencia de un motor u otro elemento en un intervalo de 1 (menor) a 8 (mayor). El bloque Set power:

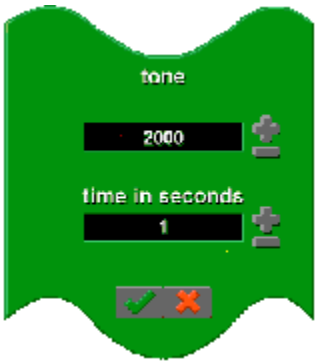
Haz clic con el botón derecho en el bloque Set power para abrirlo. Haz clic en la letra o casilla de un puerto para activarlo o desactivarlo; una X en la casilla indica que el puerto correspondiente está seleccionado. Utiliza el signo + o -, o haz clic en la escala de potencia para seleccionar un valor. Al terminar, haz clic en el cuadro que contiene la marca de verificación. La parte ampliada del bloque Set power:



A.1.13.Tone (“tono”)

El bloque Tone indica al RCX que emita un tono durante un tiempo preestablecido:

Haz clic con el botón derecho en el bloque Tone para abrirlo. Las frecuencias de tono oscilan entre 1 y 20.000. Escribe un número para indicar el tono deseado y otro para indicar su duración en segundos. Al terminar, haz clic en el cuadro que contiene la marca de verificación. La parte ampliada del bloque Tone:



A.1.14. Wait (“esperar”)

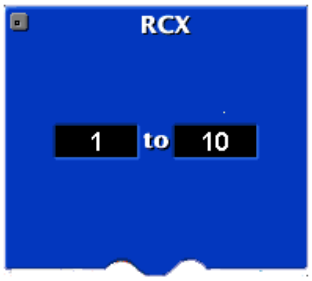
El bloque Wait hace que el RCX espere un período de tiempo determinado antes de ejecutar el siguiente comando.

Haz clic con el botón derecho en el bloque Wait para abrirlo. Escribe un número o utiliza el signo + o - para establecer los segundos que deseas que el RCX espere antes de pasar al siguiente comando. Puedes hacer clic en el dado para elegir una cantidad de tiempo aleatoria entre 0,1 y el número que se muestra en el campo. Al terminar, haz clic en el cuadro que contiene la marca de verificación. La parte ampliada del bloque Wait:

A.2. CONTROLADORES DE SENSORES EN CÓDIGO RCX

A.2.1.RCX message (“mensaje de RCX”)

Utiliza un controlador de sensor RCX message para activar una secuencia cuando se reciban mensajes de otro RCX:

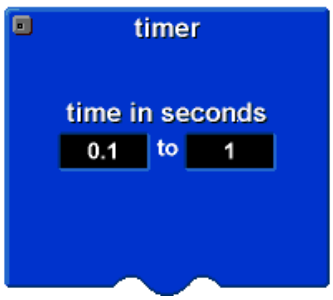


Haz clic en los números y escribe los valores mínimo y máximo del mensaje. La secuencia asociada se ejecutará cuando se reciba un mensaje con uno de los valores seleccionados de otro RCX. Los mensajes pueden tener un número del 1 al 255.

El controlador de sensor RCX message controla continuamente los mensajes. Una vez que se activa una secuencia, si se recibe un mensaje con un valor fuera del intervalo definido o se borra el mensaje y se recibe otro dentro del intervalo, la secuencia se reiniciará desde el principio.

A.2.2.Timer (“temporizador”)

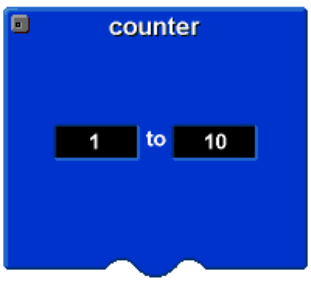
Utiliza un controlador de sensor Timer para activar una secuencia que utilice el temporizador RCX interno:



Haz clic en los números y escribe la hora de inicio y fin. La secuencia asociada se ejecutará cuando la hora alcance el intervalo definido para el temporizador. El temporizador RCX interno se pone a 0 cuando se inicia la ejecución de un programa y no se detiene.

A.2.3.Counter (“contador”)

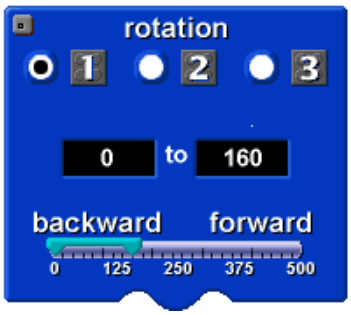
Utiliza un controlador de sensor Counter para activar una secuencia que utilice el contador RCX interno:



Haz clic en los números y escribe el valor inicial y el final del contador. La secuencia asociada se ejecutará cuando el contador RCX interno alcance un valor que se encuentre dentro del intervalo definido. El contador se pone a 0 cada vez que se ejecuta un programa. Utiliza el bloque de comandos add to counter para incrementar en 1 el valor del contador.

A.2.4.Rotation ("giro")

Utiliza un controlador de sensor Rotation para activar una secuencia cuando un eje gire para colocarse en su posición. El controlador de sensor Rotation:

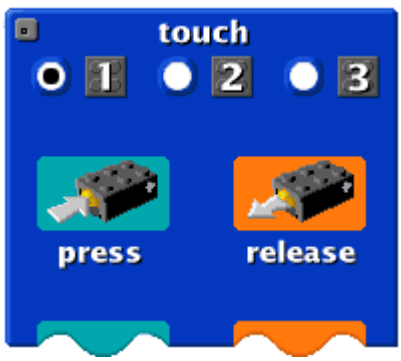


Haz clic en el botón de un puerto para elegir el sensor (1, 2 ó 3) que deseas controlar. Arrastra las flechas para cambiar el intervalo o haz clic en los números y escribe el recuento de giros mínimo y máximo. El sensor de giro cuenta 16 cada vez que el eje gira una vez. La secuencia asociada se ejecuta cuando el recuento del sensor de giro está dentro del intervalo.

El controlador de sensor Rotation controla continuamente el recuento de giros. Una vez que se activa una secuencia, si el recuento sale del intervalo y posteriormente vuelve a entrar en él, la secuencia se reinicia desde el principio.

A.2.5.Touch ("táctil")

Utiliza un controlador de sensor Touch para activar una secuencia cuando se presione un sensor táctil y otra cuando sea liberado.

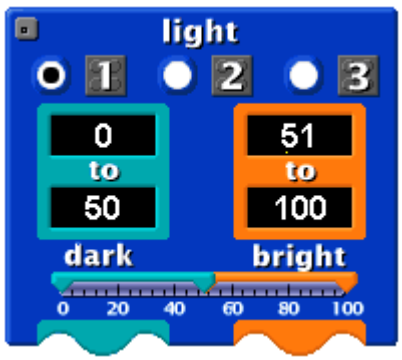


Haz clic en el botón de un puerto para elegir el sensor (1, 2 ó 3) que deseas controlar. Cuando el sensor táctil seleccionado sea presionado, se ejecutará la secuencia del controlador de sensor táctil izquierdo. Cuando el sensor táctil sea liberado, se ejecutará la secuencia del controlador de sensor táctil derecho.

El controlador de sensor Touch controla continuamente al sensor táctil. Se pueden ejecutar ambas secuencias a la vez si se presiona y se suelta rápidamente el sensor táctil. Una vez que se activa una secuencia, si el sensor táctil cambia dos veces de estado, la secuencia se reiniciará desde el principio.

A.2.6.Light (“luz”)

Utiliza un controlador de sensor Light para activar una secuencia cuando la lectura del sensor de luz se encuentre en un intervalo determinado.

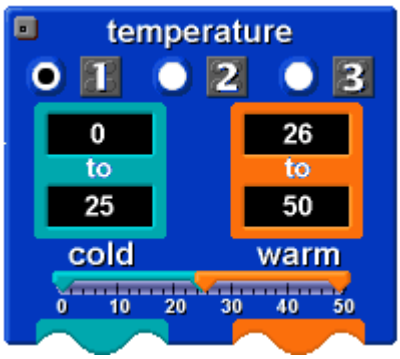


Haz clic en el botón de un puerto para elegir el sensor (1, 2 ó 3) que deseas controlar. Arrastra las flechas para cambiar los intervalos o haz clic en los números y escribe los valores mínimo y máximo para los intervalos de luz y oscuridad. La secuencia colocada debajo de un intervalo se ejecuta cuando la lectura del sensor está dentro de este intervalo.

El controlador de sensor Light registra continuamente la luz. Una vez que se activa una secuencia, si la lectura del sensor sale del campo de acción y posteriormente vuelve a entrar en él, la secuencia se reinicia desde el principio.

A.2.7.Temperature (“temperatura”)

Utiliza un controlador de sensor Temperature para activar una secuencia de comandos cuando la lectura del sensor de temperatura esté en un intervalo determinado.



Haz clic en el botón de un puerto para elegir el sensor (1, 2 ó 3) que deseas controlar. Arrastra las flechas para cambiar los intervalos o haz clic en los números y escribe los valores mínimo y máximo para los intervalos de frío y calor. La secuencia colocada debajo de un intervalo se ejecuta cuando la lectura del sensor está dentro de este intervalo.

El controlador de sensor Temperature registra continuamente la temperatura. Una vez que se activa una secuencia, si la temperatura sale del intervalo y posteriormente vuelve a entrar en él, la secuencia se reinicia desde el principio.

Si no cambias el controlador de sensor Temperature, el intervalo de frío será de 0 a 25 ºC (32 a 77 ºF) y el de calor de 26 a 50 ºC (78 a 122 ºF). El valor mínimo es –20 ºC (-4 ºF) y el máximo +50 ºC (+140 ºF).

Si lo deseas, puedes leer la temperatura en grados Fahrenheit en lugar de centígrados. Para ello, ve al menú principal, selecciona Inicio y, a continuación, Opciones de configuración. Haz clic en Avanzadas y después en el botón de grados "Fahrenheit", frente a "Temperatura medida en".

A.3.CONTROLADORES DE SECUENCIAS EN CÓDIGO RCX

A.3.1.Repeat Forever ("repetir indefinidamente")

El controlador de secuencias Repeat forever ejecuta una secuencia de comandos indefinidamente hasta que se detenga el programa. Coloca la secuencia entre los bloques Repeat forever y End repeat.

Nota: en las secuencias de controlador de sensor, Repeat forever dejará de ejecutarse si la secuencia se reinicia debido a que la lectura del sensor sale del intervalo y vuelve a entrar en él.

A.3.2.Check & Choose ("comprobar y elegir")

El controlador de secuencias Check and choose selecciona una de dos secuencias de comandos distintas utilizando una condición de sensor.

Elige un sensor para el bloque Check and choose: táctil, de luz, contador, temporizador o mensaje RCX. También estarán disponibles los sensores de giro y de temperatura, si se han seleccionado en Opciones de configuración. Al terminar, haz clic en el cuadro que contiene la marca de verificación. El controlador de secuencias Check and choose con el sensor de luz seleccionado:

Haz clic en el botón de un puerto para elegir el sensor (1, 2 ó 3) que deseas utilizar. Arrastra las flechas o haz clic en los números y escribe los valores mínimo y máximo del intervalo. Cuando la condición del sensor se cumpla (cuando la lectura del sensor de luz esté dentro del intervalo), se elegirá la secuencia de

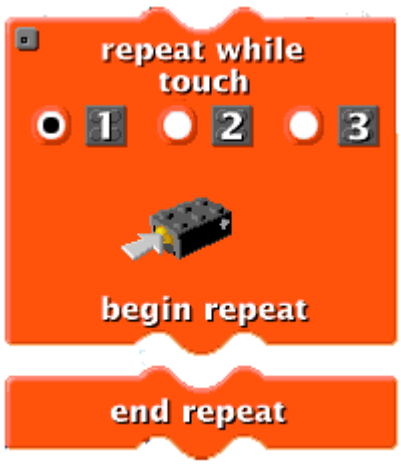
la izquierda. Cuando la condición del sensor no se cumpla (cuando la lectura del sensor de luz esté fuera del intervalo, es decir, sea superior o inferior al máximo o mínimo establecido), se elegirá la secuencia de la derecha.

A.3.3.Repeat While (“repetir mientras”)

El controlador de secuencias Repeat while repite una secuencia de comandos siempre que se cumpla una condición del sensor.

Elige un sensor para el bloque Repeat while: táctil, de luz, contador, temporizador o mensaje RCX. También estarán disponibles los sensores de giro y de temperatura, si se han seleccionado en Opciones de configuración. Al terminar, haz clic en el cuadro que contiene la marca de verificación.

El controlador de secuencias Repeat while con el sensor táctil seleccionado:



Haz clic en el botón de un puerto para elegir el sensor (1, 2 ó 3) que deseas utilizar. Coloca la secuencia entre los bloques Begin repeat y End repeat.

- Para los sensores de luz, temperatura o giro, arrastra las flechas o haz clic en los números y escribe los valores mínimo y máximo del intervalo.
- Para el sensor táctil, haz clic en el icono del sensor táctil para alternar entre presionado y no presionado.
- Para el temporizador interno, haz clic y escribe el tiempo entre 0,1 y 327,6 segundos.
- Para el contador interno, haz clic y escribe números entre 1 y 32766.
- Para el sensor de mensajes RCX, haz clic y escribe los valores de mensaje entre 1 y 255.

A.3.4.Wait Until (“repetir hasta”)

El controlador de secuencias Wait until espera hasta que se cumpla una condición del sensor antes de pasar al comando siguiente.

Elige un sensor para el bloque Wait until: táctil, de luz, contador, temporizador o mensaje RCX. También estarán disponibles los sensores de giro y de temperatura, si se han seleccionado en Opciones de configuración. Al terminar, haz clic en el cuadro que contiene la marca de verificación. El controlador de secuencias Wait until con el sensor táctil seleccionado:

Haz clic en el botón de un puerto para elegir el sensor (1, 2 ó 3) que deseas utilizar.

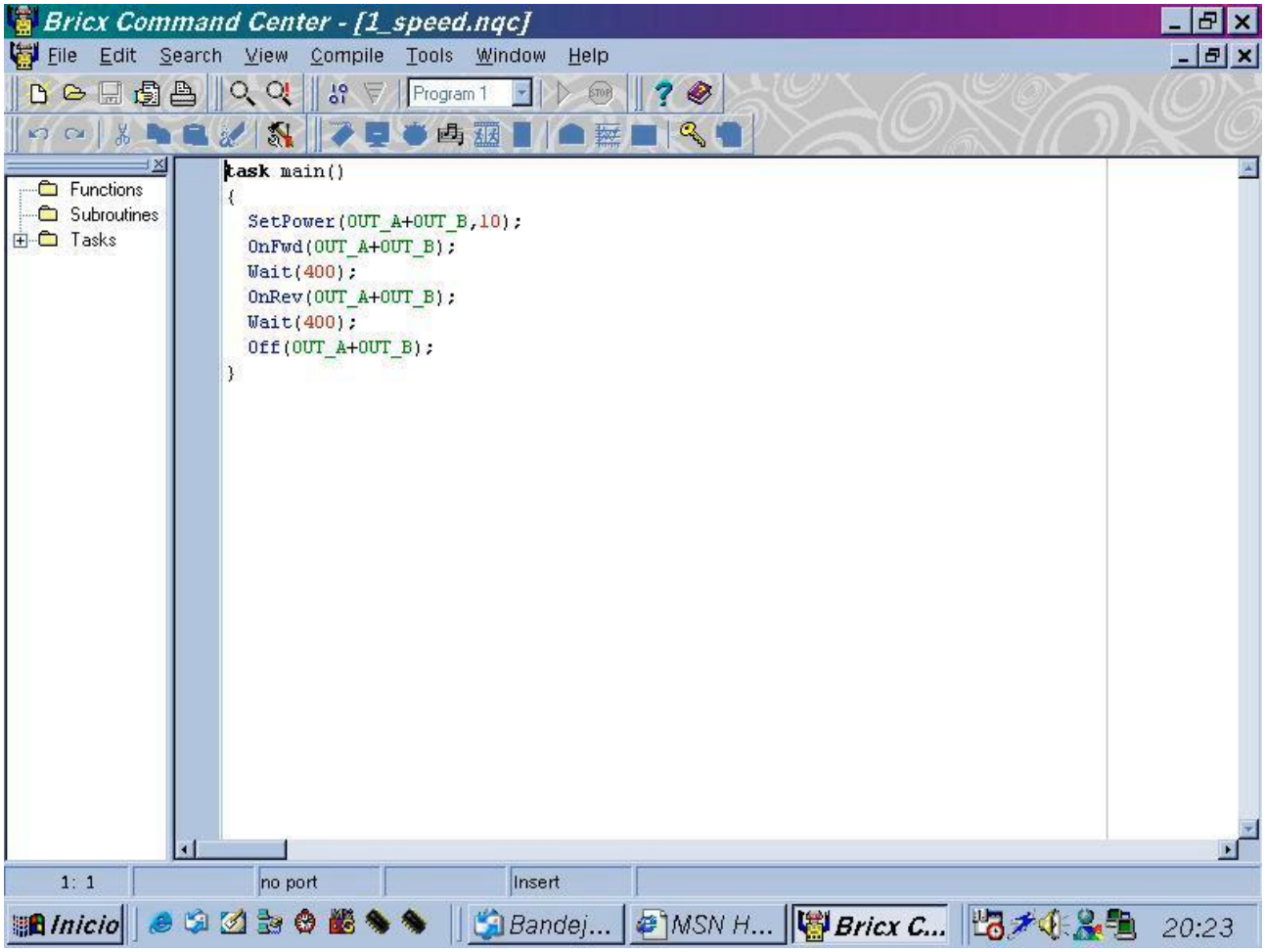
- Para los sensores de luz, temperatura o giro, arrastra las flechas o haz clic en los números y escribe los valores mínimo y máximo del intervalo.
- Para el sensor táctil, haz clic en el icono del sensor táctil para alternar entre presionado y no presionado.
- Para el temporizador interno, haz clic y escribe el tiempo entre 0,1 y 327,6 segundos.
- Para el contador interno, haz clic y escribe números entre 1 y 32766.
- Para el sensor de mensajes RCX, haz clic y escribe los valores de mensaje entre 1 y 255.

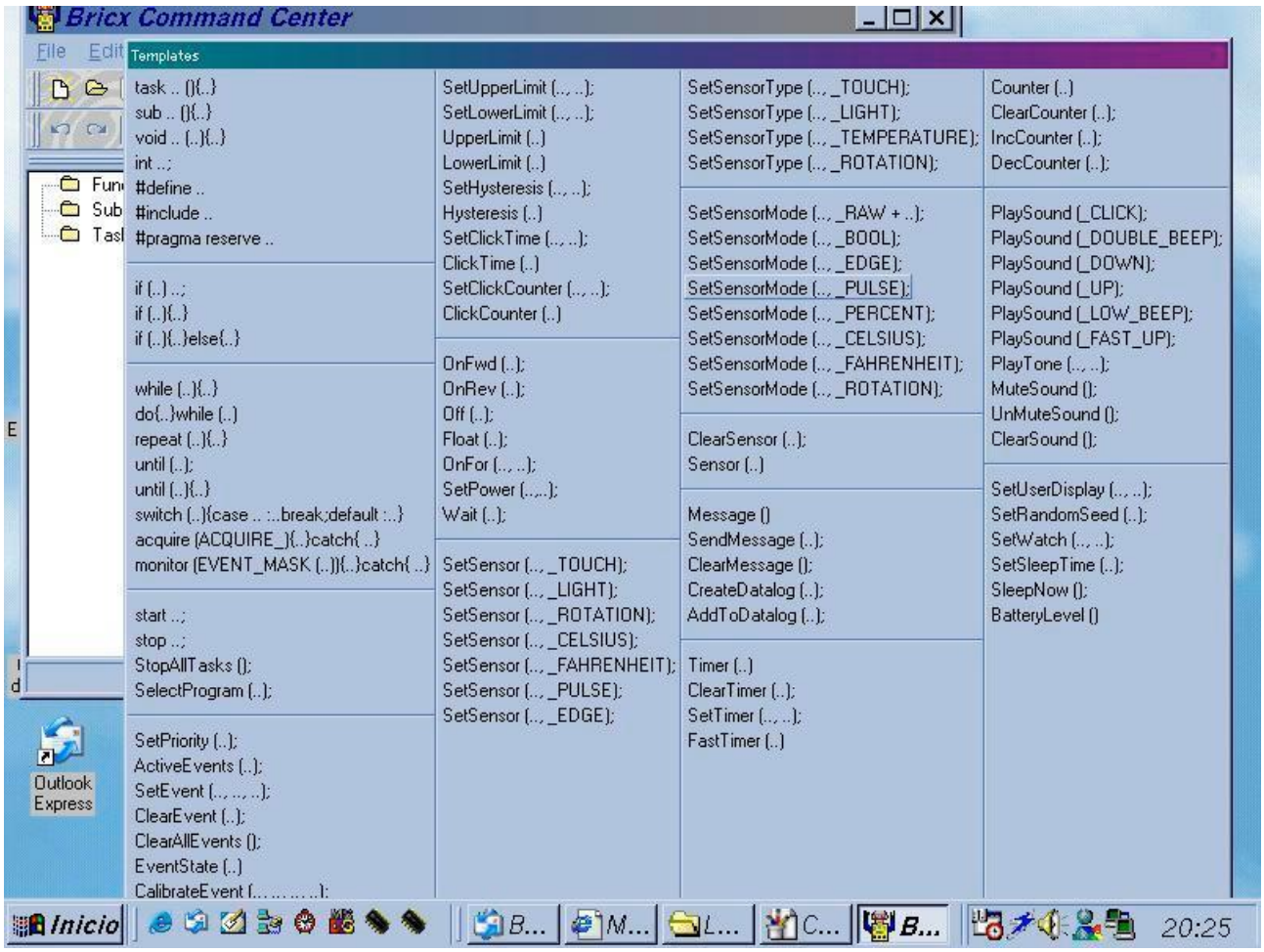
El bloque Wait until retardará la ejecución del comando hasta que se cumpla la condición del sensor. El bloque Wait until controla únicamente la secuencia en la que está colocado; otras secuencias (en los controladores de sensor, por ejemplo) continúan.

ANEXO B. NQC

B.1.Como utilizar NQC

Tal como hemos mencionado anteriormente, escribiremos nuestros programas utilizando el **RCX Command Center.** El interface se muestra como un editor de texto estándar. Hay menús para compilar y transferir los programas al robot y obtener información del robot, además de los clásicos de abrir, cerrar fichero y guardar entre otros.





Ahora introduciremos un programa en el NQC para después ejecutarlo. Todos los programas en NQC están compuestos por tareas, nuestro programa tiene una tarea denominada **main** (principal). Todo programa debe tener esta tarea porque es la que ejecuta el robot. Una tarea esta compuesta por varias ordenes, denominadas instrucciones.

Ejemplo: Desplazamiento hacia delante durante 4 segundos y atrás durante 4 segundos.

```
Task main ( )
{
OnFwd ( OUT_A );
OnFwd ( OUT_C );
Wait ( 400 );
OnRev ( OUT_A + OUT_C );
Wait ( 400 );
Off ( OUT_A + OUT_C );
}
```

OnFwd (OUT_A);
Esta instrucción indica que ha de iniciar la salida A, es decir, que el motor conectado en la salida etiquetada A en el RCX debe moverse hacia adelante. Se moverá a la velocidad máxima, a no ser que previamente se establezca otra velocidad.

OnFwd (OUT_C);
Esta instrucción indica que ha de iniciar la salida C, es decir, que el motor conectado en la salida etiquetada C en el RCX debe moverse hacia adelante. Se moverá a la velocidad máxima, a no ser que previamente se establezca otra velocidad.

Wait (400);
Esta instrucción nos produce un tiempo de espera de 4 segundos, el número entre paréntesis nos marca el número de tics. Un tic es una centésima de segundo. Por lo tanto durante 4 segundos el programa no realizará nada.

OnRev (OUT_A + OUT_C);
Con esta instrucción hacemos que el robot se desplace en dirección opuesta.

Off (OUT_A + OUT_C);
Apagamos los dos motores.

Una vez que tenemos el programa es necesario compilarlo, significa convertirlo en código que el robot pueda entender y ejecutar, enviarlo al robot por medio del puerto infrarrojos.

Para cambiar la velocidad utilizamos la instrucción:

SetPower (OUT_A + OUT_C ,2) , donde la potencia es un número entre 0 y 7.

B.1.2. Como hacer giros ?

Se puede hacer que el robot gire haciendo parar uno de los dos motores o modificando su dirección de giro. Por ejemplo: Avanzará un poco y dará un giro de 90º a la derecha.

```
Task main ( )
{
OnFwd ( OUT_A + OUT_C);
Wait ( 100 );
OnRev ( OUT_C);
Wait ( 85 );
Off ( OUT_A + OUT_C );
}
```

En NQC se pueden definir valores constantes, por ejemplo:

```
#define TIEMPO_DE_AVANCE 100
#define TIEMPO_GIRO 85

Task main ( )
{
OnFwd ( OUT_A + OUT_C);
Wait ( TIEMPO_DE_AVANCE );
OnRev ( OUT_C);
Wait ( TIEMPO_GIRO );
Off ( OUT_A + OUT_C );
}
```

Las dos primeras líneas son dos constantes, estas pueden ser usadas a lo largo del programa.
Definir constantes es bueno para hacer el programa más legible y es más fácil cambiar los valores.

B.1.3. Repetición de órdenes

Para hacer que una ejecución se repita 4 veces no hace falta escribir cuatro veces lo mismo, si introducimos la instrucción repeat ya nos lo hará.

Con este ejemplo podemos ver que el robot dará 10 vueltas describiendo una trayectoria cuadrada debido a que tiene que hacer 4 giros de 90º.

```
#define TIEMPO_DE_AVANCE 100
#define TIEMPO_GIRO 85
Task main ( )
{
 repeat (10);
 {
        repeat (4);
        {
        OnFwd ( OUT_A + OUT_C);
        Wait ( TIEMPO_DE_AVANCE );
        OnRev ( OUT_C);
        Wait ( TIEMPO_GIRO );
        }
 }
 Off ( OUT_A + OUT_C );
}
```

B.1.4. Variables

El uso de variables constituye un aspecto importante de todo lenguaje de programación. Las variables son posiciones de memoria en las que podemos almacenar un valor. Podemos utilizar este valor en diferentes lugares y también podemos modificarlo.

```
#define TIEMPO_GIRO 85

int  tiempo_de_avance                  // definir una variable
Task main ( )
{
  tiempo_de_avance = 20;               // asignar el valor inicial
  repeat (50);
 {
   OnFwd ( OUT_A + OUT_C);
   Wait (tiempo_de_avance);            // utilizar la variable para la espera
   OnRev ( OUT_C);
   Wait ( TIEMPO_GIRO );
    tiempo_de_avance += 5;             // incrementar la variable
 }
 Off ( OUT_A + OUT_C );
}
```

En este ejemplo podemos ver que el valor inicial es 20 y que al final del bucle se incrementa en 5 unidades, así la primera vez el robot avanza durante 20 tics, segunda durante 25, la tercera durante 30, etc...

B.1.5.Números aleatorios

Cuando el robot puede hacer cosas que desconocemos gana más interés. Ahora queremos algo de aleatoriedad en los movimientos. En NQC puedes generar números aleatorios con la instrucción **random**.

```
int  tiempo_de_avance, tiempo_giro
Task main ( )
{
  while ( true )
 {
   tiempo_de_avance = Random (60);
   tiempo_giro = Random (40);
   OnFwd ( OUT_A + OUT_C);
   Wait (tiempo_de_avance);
   OnRev ( OUT_A);
   Wait ( tiempo_giro);
 }
}
```

El programa define dos variables y les asigna números aleatorios. Random (60) representa un número aleatorio entre 0 y 60. Cada vez los números serán diferentes, sería posible evitar el uso de variables escribiendo Wait (Random (60)).

Otra forma de hacer un bucle de repetición es utilizando la instrucción While (mientras) repite las instrucciones que le siguen mientras la condición que se encuentra entre paréntesis se cumpla.

B.1.6. Estructuras de control

B.1.6.1.La instrucción IF

A veces interesa que una parte de un programa sea sólo ejecutada en ciertas situaciones, para ello utilizamos la instrucción if. Para ver el funcionamiento vamos a modificar el programa utilizado en todos los ejemplos anteriores.

Por ejemplo queremos que el robot circule por una línea recta y a continuación gire a izquierda o derecha, para hacer esto necesitamos utilizar los números aleatorios. Eligiéremos un número aleatorio entre 0 y 1, eso significa que será 0 o 1, si el número es 0 gira a la derecha, en caso contrario gira a la izquierda.

```
#define TIEMPO_DE_AVANCE 100
#define TIEMPO_GIRO 85
Task main ( )
{
  while ( true )
 {
   OnFwd ( OUT_A + OUT_C);
   Wait (TIEMPO_DE_AVANCE);
    If ( Random(1) = = 0 )                  // la condición será verdadera
    {
     OnRev ( OUT_C);
    }
    else
    {
     OnRev ( OUT_A);
    }
    Wait ( TIEMPO_GIRO);
```

```
  }
}
```

La instrucción **if** tiene cierta similitud con el while. Si la condición entre paréntesis se cumple, ejecutará la parte comprendida entre llaves, sino será la parte comprendida entre las llaves del **else**.

B.1.6.2. La instrucción DO

Las instrucciones entre llaves tras el do son ejecutadas mientras la condición sea verdadera. La condición tiene la misma estructura que la que se ha descrito anteriormente para la instrucción **if**. Por ejemplo, haremos circular al robot de modo aleatorio durante 20 segundos y entonces se parará.

```
int  tiempo_de_avance, tiempo_giro, tiempo_total

Task main ( )
{
  total_time = 0;
 do
 {
   tiempo_de_avance = Random (100);
   tiempo_giro = Random (100);
   OnFwd ( OUT_A + OUT_C);
   Wait (tiempo_de_avance);
   OnRev ( OUT_C);
   Wait ( tiempo_giro);
   total_time + = tiempo_de_avance;
   tiempo_total + = tiempo_giro;
 }
 while ( tiempo_total < 2000)
 OffFwd ( OUT_A + OUT_C);
}
```

La instrucción do se comporta casi del mismo modo que la instrucción while. Pero mientras que en la instrucción while las condiciones se comprueban antes de ejecutar las instrucciones, en la instrucción do esto se hace al final. En una instrucción while, las instrucciones que comprende pueden no ser nunca ejecutadas, pero con la instrucción do por lo menos se ejecutan una vez.

B.1.7.Sensores

Si utilizamos los sensores debemos comunicarle al robot que tipo de sensor utilizamos, por ejemplo SENSOR_1 es el número de entrada en la que está conectado el sensor, las dos otras entradas se denominan SENSOR_2 y SENSOR_3. Para indicar que tipo se utiliza la instrucción SENSOR_TOUCH el cual es de contacto o SENSOR_LIGHT el cual es de luz.

Veamos un ejemplo de como programarlo:

```
Task main ( )
{
 SetSensor(SENSOR_1,SENSOR_TOUCH);
 OnFwd ( OUT_A + OUT_C);
 Until (SENSOR_1 = = 1)
  OffFwd ( OUT_A + OUT_C);
}
```

Una vez especificado los sensores el programa pone en marcha los dos motores y el robot comienza a avanzar. La siguiente instrucción es una construcción muy útil, espera hasta que la condición entre paréntesis sea verdadera, el cual indica que el SENSOR_1 debe ser 1, sensor presionado, mientras el sensor no este presionado el valor será 0, por lo tanto cuando el sensor sea 0 los motores se detendrán.

B.1.7.1.Sensor de luz

Para el sensor de luz procederemos igual que el anterior, primero lo definiremos y después definiremos el valor de luminosidad, este valor debe superar el valor 40 debido a que se tiene que tener en cuenta la luz ambiente, entonces invertiremos el giro. Ejemplo:

```
#define UMBRAL  40
Task main ( )
{
  SetSensor(SENSOR_2,SENSOR_LIGHT);
  OnFwd ( OUT_A + OUT_C);
  While ( true )
   {
      if (SENSOR_2 > UMBRAL )
       {
         OnRev ( OUT_C);
         Until (SENSOR_2 < = UMBRAL )
         OnFwd ( OUT_A + OUT_C);
       }
   }
}
```

B.1.8. Subrutinas y tareas

B.1.8.1. Tareas

Un programa en NQC consiste como máximo en 10 tareas. Cada tarea tiene nombre propio. Una tarea he de tener el nombre main, y esta será la que se ejecutará. Las otras tareas solo serán ejecutadas cuando la tarea principal las llame para ser ejecutadas utilizando la orden start. A partir de este momento las tareas se ejecutarán a la vez, una tarea en marcha puede detener a la otra utilizando el stop.

Aquí ponemos un ejemplo de como hacer que un robot circule describiendo cuadrados pero cuando choque deberá reaccionar, para ello haremos una tarea para circular en cuadrados y otra que reaccione a los sensores.

```
Task main ( )
{
  SetSensor(SENSOR_1,SENSOR_touch);
  Start chequea_sensor;
  Start mover_cuadrado;
}

Task mover_cuadrado
{
 While ( true )
   {
         OnFwd ( OUT_A + OUT_C);
         Wait ( 100 );
         OnRev ( OUT_C);
         Wait ( 85 );
```

```
    }
}


Task chequea_sensor
{
 While ( true )
   {
    if ( SENSOR _1 = = 1 )
      {
       Stop mover_cuadrado
       OnRev ( OUT_A + OUT_C);
       Wait ( 50 );
       OnRev ( OUT_A);
       Wait ( 85 );
        Start mover_cuadrado
      }
   }
}
```

La tarea main sólo define el tipo de sensor e inicia la otras dos tareas, tras esto la tarea main finaliza .La tarea mover_cuadrado mueve el robot siempre en cuadrados. La tarea chequea_sensor comprueba si el sensor de contacto está pulsado, si es así se realiza lo siguiente: primero detiene la tarea mover_cuadrado, luego chequea_sensor toma el control del movimiento del robot, a continuación el robot retrocede un poco y gira entonces permite que la tarea mover_cuadrado funcione otra vez.

B.1.8.2. Subrutinas

En según que situaciones se necesita parte del código en múltiples lugares del programa. Para ello escribimos esta parte del código como una subrutina y le damos nombre. Entonces la podemos ejecutar con tan solo llamarla por su nombre desde el interior de la tarea. La NQC permite como máximo un máximo de 8 subrutinas. Veamos un ejemplo:

```
Sub gira( )
{
OnRev ( OUT_C);
Wait ( 340 );
OnFwd( OUT_A + OUT_C);
}
Task main ( )
{
OnFwd ( OUT_A + OUT_C);
Wait ( 100 );
gira ( );
Wait ( 200 );
gira ( );
Wait ( 100 );
Off ( OUT_A + OUT_C );
}
```

En este programa hemos definido una subrutina que hace que el robot gire en torno a sí. La tarea main llama tres veces a la subrutina. La subrutinas no pueden ser llamadas desde otras subrutinas, las subrutinas pueden ser llamadas desde otras tareas, producen problemas ya que puede estar ejecutándose por dos lados diferentes y por último debido a las limitaciones del firmware RCX, no se pueden utilizar expresiones muy complicadas. La parte buena de una subrutina es que solo se almacena una vez en el RCX.

Esto ahorra memoria, pero cuando las subrutinas son pequeñas es mejor utilizar las funciones, estas no se almacenan separadamente sino que se copian en cada lugar en que tienen que ser usadas, esto consume más memoria pero no tendremos problemas.

B.1.8.3. Funciones

La llamada a funciones se hace exactamente igual que a las subrutinas, solo que usamos la palabra void en lugar de sub. Veamos el ejemplo anterior pero con funciones:

```
Void gira( )
{
OnRev ( OUT_C);
Wait ( 340 );
OnFwd( OUT_A + OUT_C);
}
Task main ( )
{
OnFwd ( OUT_A + OUT_C);
Wait ( 100 );
gira ( );
Wait ( 200 );
gira ( );
Wait ( 100 );
gira ( );
Off ( OUT_A + OUT_C );
}
```

Las funciones tienen otra ventaja sobre las subrutinas, pueden tener argumentos, los argumentos pueden ser utilizados para transferir un valor a ciertas variables en una función. Por ejemplo: supongamos que en el ejemplo anterior hacemos que el tiempo de giro sea el argumento de la función.

```
Void gira( int tiempo_giro )
{
OnRev ( OUT_C);
Wait ( tiempo_giro );
OnFwd( OUT_A + OUT_C);
}

Task main ( )
{
OnFwd ( OUT_A + OUT_C);
Wait ( 100 );
gira ( 200 );
Wait ( 200 );
gira ( 50 );
Wait ( 100 );
gira ( 300 );
Off ( OUT_A + OUT_C );
}
```

B.1.8.4. Macros

Otra manera de dar nombre a pequeños fragmentos de código es utilizar macros. Ahora veremos el mismo código anterior pero utilizando macros.

```
#define  gira OnRev ( OUT_C) ; Wait ( 340 );OnFwd( OUT_A + OUT_C);

Task main ( )
{
OnFwd ( OUT_A + OUT_C);
Wait ( 100 );
gira;
Wait ( 200 );
gira;
Wait ( 100 );
gira;
Off ( OUT_A + OUT_C );
}
```

Las instrucciones define pueden tener argumentos. Por ejemplo, podemos poner el tiempo de giro como un argumento en la instrucción. Ahora definimos cuatro macros, una para avanzar, otra para retroceder, otra para girar a la izquierda y otra para girar hacia la derecha, cada una tiene dos argumentos: velocidad y tiempo.

```
#define gira_derecha ( s, t ) SetPower ( OUT_A + OUT_C ,s) ;OnFwd ( OUT_A);
                              OnRev ( OUT_C); Wait ( t );
#define  gira_izquierda ( s, t ) SetPower ( OUT_A + OUT_C ,s) ;OnRev ( OUT_A);
                                 OnFwd ( OUT_C); Wait ( t );
#define  avanza ( s, t )  SetPower ( OUT_A + OUT_C ,s); OnFwd ( OUT_A + OUT_C); Wait ( t );
#define  retrocede ( s, t ) SetPower ( OUT_A + OUT_C ,s); OnRev ( OUT_A + OUT_C); Wait ( t );

Task main ( )
{
avanza ( 3, 200 );
gira_izquierda ( 7, 85 );
avanza ( 7, 100 );
retrocede ( 7 ,200 );
avanza ( 7, 100 );
gira_derecha ( 7 ,85 );
avanza ( 3 ,200 );
Off ( OUT_A + OUT_C );
}
```

Las macros son muy útiles ya que convierte el código más compacto y legible.

B.1.9. Como hacer música

El RCX tiene en el interior un altavoz el cual puede generar sonidos. Los sonidos los cuales puede hacer el RCX son los siguientes:

0. Click de tecla
1. Pitido
2. Barrido de frecuencia decreciente
3. Barrido de frecuencia creciente
4. "Buhhh" sonido de error
5. Barrido de rápido crecimiento

Para generar dichos sonidos debemos utilizar la instrucción PlaySound( ). Ejemplo de un programa el cual reproduce todos los sonidos anteriores.

```
Task main ( )
{
PlaySound( 0 ) ; wait ( 100 );
PlaySound( 1 ) ; wait ( 100 );
PlaySound( 2 ) ; wait ( 100 );
PlaySound( 3 ) ; wait ( 100 );
PlaySound( 4 ) ; wait ( 100 );
PlaySound( 5 ) ; wait ( 100 );
}
```

Para crear una música más interesante , NQC tiene la orden PlayTone( ). Tiene dos argumentos, la frecuencia y la duración. Aquí tenemos una tabla con las frecuencias.

Sonido	1	2	3	4	5	6	7	8
G#	52	104	208	415	831	1661	3322	
G	49	98	196	392	784	1568	3136	
F#	46	92	185	370	740	1480	2960	
F	44	87	175	349	698	1397	2794	
E	41	82	165	330	659	1319	2637	
D#	39	78	156	311	622	1245	2489	
D	37	73	147	294	587	1175	2349	
C#	35	69	139	277	554	1109	2217	
C	33	65	131	262	523	1047	2093	4186
B	31	62	123	247	494	988	1976	3951
A#	29	58	117	233	466	932	1865	3729
A	28	55	110	220	440	880	1760	3520

```
Task main ( )
{
PlayTone( 262 , 40 ) ; wait ( 50 );
PlayTone( 294 , 40 ) ; wait ( 50);
PlayTone( 330 , 40 ) ; wait ( 50 );
PlayTone( 294 , 40 ) ; wait ( 50 );
PlayTone( 262 , 160 ) ; wait ( 200 );
}
```

Si quieres que el RCX toque música mientras circula, lo mejor es utilizar una tarea para cada cosa.

B.1.10. Otras instrucciones

B.1.10.1. Motores

Cuando utilizamos la orden Off( ), el motor se para inmediatamente. Para parar el motor más suavemente se utiliza la instrucción Float( ).

La orden OnFwd( ) hace dos cosas conecta el robot y establece la dirección de avance. La orden OnRev( ) también hace dos cosas, conecta el motor y establece la dirección del motor de retroceso. NQC tiene instrucciones para hacer esto separadamente. Las dos ordenes a utilizar por separado son SetDirection( ), que establece la dirección y SetOutput( ), que establece el modo.

SetDirection(OUT_FWD, OUT_REV o OUT_TOGGLE que invierte la dirección actual.)
SetOutput(OUT_ON, OUT_OFF o OUT_FLOAT)

```
Task main ( )
{
SetPower ( OUT_A + OUT_C , 7 );
SetDirection (OUT_A + OUT_C , OUT_FWD );
SetOutput (OUT_A + OUT_C , OUT_ON );
Wait ( 200 );
SetDirection (OUT_A + OUT_C , OUT_REV );
Wait ( 200 );
SetDirection (OUT_A + OUT_C , OUT_TOGGLE );
Wait ( 200 );
SetOutput (OUT_A + OUT_C , OUT_FLOAT );
}
```

B.1.10.2. Sensores

La orden SetSensor( ) hace dos cosas, establece el tipo de sensor y el modo en que opera. Estableciendo el modo y tipo de sensor por separado podemos controlar el comportamiento del sensor con más precisión.

El tipo se establece con la orden SetSensorType( ), hay cuatro tipos:

SENSOR_TYPE_TOUCH, sensor de contacto
SENSOR_TYPE_LIGHT, sensor de luz
SENSOR_TYPE_TEMPERATURE, sensor de temperatura
SENSOR_TYPE_ROTATION, sensor de rotación

El establecimiento del sensor es importante para indicar si el sensor necesita alimentación, tal como sucede con el sensor de luz.

El modo del sensor se establece con la orden SetSensorMode( ), hay ocho modos diferentes:

SENSOR_MODE_RAW, es el más importante, indica si esta totalmente pulsado o parcialmente.
SENSOR_MODE_BOOL, obtenemos valores de 0 o 1, sirve para los sensores de contacto.
SENSOR_MODE_CELSIUS
SENSOR_MODE_FAHRENHEIT, son útiles con los sensores de temperatura donde indican la temperatura en la unidad deseada.
SENSOR_MODE_PERCENT, convierte el valor sin procesar en otro comprendido entre 0 y 100.
SENSOR_MODE_ROTATION, sirve para el sensor de rotación.
SENSOR_MODE_EDGE

SENSOR_MODE_PULSE, cuentan transiciones, variaciones de un valor bajo a un valor de lectura sin procesar o viceversa.

Ejemplo:

```
task main()
{
  SetSensorType(SENSOR_1,SENSOR_TYPE_TOUCH);
  SetSensorMode(SENSOR_1,SENSOR_MODE_PULSE);
  while(true)
  {
    ClearSensor(SENSOR_1);
    until (SENSOR_1 >0);
    Wait(100);
    if (SENSOR_1 == 1) {Off(OUT_A+OUT_C);}
    if (SENSOR_1 == 2) {OnFwd(OUT_A+OUT_C);}
  }
}
```

B.1.11. Comunicaciones entre robots

Si disponemos más de un RCX podemos comunicarlos entre ellos a través del puerto de infrarrojos. La comunicación entre robots funciona, la orden SendMessage( ) para enviar un valor (0 - 255) por medio del puerto. El resto de robots reciben el mensaje y lo almacenan. El programa de un robot puede buscar el valor del último mensaje recibido utilizando Message( ).

Cuando disponemos de dos o más robots, uno es el líder. Le llamaremos el Master. Los otros robots son los esclavos. El Master envía órdenes a los esclavos y estos las ejecutan, otra veces los esclavos pueden devolver información al Master, por ejemplo el valor devuelto de un sensor. Así debemos escribir un programa para el Master y otro para los esclavos.

El programa del esclavo será el siguiente, esperará a que el mensaje se convierta en otro diferente de 0.

```
task main()          // ESCLAVO
{
  while (true)
  {
    ClearMessage();
    until (Message() != 0);
    if (Message() == 1) {OnFwd(OUT_A+OUT_C);}
    if (Message() == 2) {OnRev(OUT_A+OUT_C);}
    if (Message() == 3) {Off(OUT_A+OUT_C);}
  }
}
```

El Master tiene un programa más sencillo, simplemente envía los mensajes correspondientes a las órdenes y espera un poco. En el programa siguiente ordena al esclavo avanzar, tras dos segundos retroceder, y tras otros segundos, detenerse.

```
task main()          // PATRÓN
{
  SendMessage(1); Wait(200);
  SendMessage(2); Wait(200);
  SendMessage(3);
}
```

Si tenemos múltiples esclavos tenemos que transferir los programas a los esclavos por turnos.

B.1.12. Temporizadores

El RCX dispone de 4 temporizadores internos, estos temporizadores hacen tictac en incrementos de 0.1 segundo. Estos están numerados del 0 al 3. Se pueden reinicializar los temporizadores con la orden ClearTimer( ) y establecer el valor actual del temporizador con Timer( ). Por ejemplo, este programa hace que el robot circule de un modo aleatorio durante 20 segundos.

```
Task main ( )
{
 ClearTimer ( 0 );
 do
 {
   OnFwd ( OUT_A + OUT_C);
   Wait (Random ( 100 ));
   OnRev ( OUT_C);
   Wait (Random ( 100 ));
 }
 while ( Timer ( 0 ) < 200)
 OffFwd ( OUT_A + OUT_C);
}
```

Los temporizadores son muy útiles para reemplazar la orden wait( ). Puedes hacer una espera de una determinada cantidad de tiempo reinicializando el temporizador y esperando hasta que alcance un determinado valor. por ejemplo, en el siguiente programa permite que el robot circule hasta que pasen 10 segundos o el sensor de contacto toque algo.

```
Task main ( )
{
SetSensor ( SENSOR_1, SENSOR_TOUCH );
ClearTimer ( 3 );
OnFwd ( OUT_A + OUT_C);
Until ( (SENSOR_1 = = 1 ) | |  ( Timer ( 3) > 100 )) ;
OffFwd ( OUT_A + OUT_C);
}
```

B.1.13. Pantalla

Es posible controlar la pantalla del RCX de dos diferentes maneras. En la primera de ellas, puedes indicar que muestre el reloj del sistema, uno de los sensores o uno de los motores. Esto es equivalente a utilizar el botón negro (view) del RCX. Para establecer el tipo de pantalla , se utiliza la orden SelectDisplay( ). El programa siguiente muestra las siete posibilidades, una detrás de la otra.

```
task main()
{
  SelectDisplay(DISPLAY_SENSOR_1); Wait(100);  // Entrada 1
  SelectDisplay(DISPLAY_SENSOR_2); Wait(100);  // Entrada 2
  SelectDisplay(DISPLAY_SENSOR_3); Wait(100);  // Entrada 3
  SelectDisplay(DISPLAY_OUT_A);    Wait(100);  // Salida A
  SelectDisplay(DISPLAY_OUT_B);    Wait(100);  // Salida B
  SelectDisplay(DISPLAY_OUT_C);    Wait(100);  // Salida C
  SelectDisplay(DISPLAY_WATCH);    Wait(100);  // Reloj del sistema
}
```

El segundo modo por el cual podemos controlar la pantalla es controlando el valor del reloj del sistema. Puedes utilizarlo para mostrar, por ejemplo, la información de diagnóstico. Para ello utilizamos la orden SetWatch( ).Por ejemplo:

```
task main()
{
  SetWatch(1,1); Wait(100);
  SetWatch(2,4); Wait(100);
  SetWatch(3,9); Wait(100);
  SetWatch(4,16); Wait(100);
  SetWatch(5,25); Wait(100);
}
```

B.1.14. Registros de datos

El RCX puede almacenar valores de variables, lecturas de sensores, lectura de temporizadores, en un espacio de memoria llamado **datalog**. Los valores contenidos en el datalog no pueden ser utilizados en el interior del RCX, pero pueden ser leídos desde el ordenador. Esto es útil para chequear que ocurre en el robot. El RCX Command Center tiene una ventana especial en la que puedes ver el contenido actual del datalog.

El uso del registro de datos consiste en tres pasos:

1. El programa NQC debe definir el tamaño del datalog utilizando la orden CreateDatalog( ), esto también borra el contenido actual del datalog.
2. Los valores pueden ser escritos uno tras otro. Si se alcanza el final del datalog, no pasa nada los nuevos valores no serán almacenados.
3. Transferir el datalog al PC, para ello hay que seleccionar en el RCX Command Center la orden Datalog en el menú **Tools**, a continuación pulsar el botón denominado **Upload Datalog** y aparecerán todos los valores.

Con este comando se han hecho escáners con el RCX.

Por ejemplo: tenemos un robot con un sensor de luz, el robot circula por 10 segundos, y cinco veces por segundo le valor del sensor de luz es almacenado en el datalog.

```
task main()
{
  SetSensor(SENSOR_2,SENSOR_LIGHT);
  OnFwd(OUT_A+OUT_C);
  CreateDatalog(50);
  repeat (50)
  {
    AddToDatalog(SENSOR_2);
    Wait(20);
  }
  Off(OUT_A+OUT_C);
}
```

B.1.15. Referencia rápida en NQC

INSTRUCCIONES

Instrucción	Descripción
While (cond) cuerpo	Ejecuta el cuerpo cero o más veces mientras la condición es verdadera
Do cuerpo While (cond)	Ejecuta el cuerpo cero o más veces mientras la condición es verdadera
Until (cond) cuerpo	Ejecuta el cuerpo cero o más veces hasta que la condición es verdadera
Break	Salir del cuerpo de un while, do , until
Continue	Saltarse la siguiente iteración del cuerpo de un while, do , until
Repeat (expresión) cuerpo	Repetir el cuerpo un número determinado de veces
If (cond) inst 1 If (cond) inst 1 else inst 2	Ejecuta inst 1 si la condición es verdadera. Ejecuta inst 2 si existe si la condición no se cumple
Start nombre_de_tarea	Inicia la tarea especificada
Stop nombre_de_tarea	Detiene la tarea especificada
Function (args)	Llama una función utilizando los correspondientes argumentos
Var = expresión	Evalúa una expresión y la asigna a una variable
Var + = expresión	Evalúa una expresión y la suma a una variable
Var - = expresión	Evalúa una expresión y la resta a una variable
Var * = expresión	Evalúa una expresión y la multiplica a una variable
Var / = expresión	Evalúa una expresión y la divide la variable por ella
Var | = expresión	Evalúa una expresión y realiza una comparación OR bit a bit con la variable dejando el resultado en esta última
Var & = expresión	Evalúa una expresión y realiza una comparación AND bit a bit con la variable dejando el resultado en esta última
Return	Devuelve el control de la función al llamador
Expresión	Expresión a evaluar

CONDICIONES

Condición	Significado
true	Siempre verdadero
false	Siempre falso
Expr1 = = expre 2	Comprueba si las expresiones son iguales
Expr1 ! = expre 2	Comprueba si las expresiones son diferentes
Expr1 < expre 2	Comprueba si una expresión es inferior a la otra
Expr1 < = expre 2	Comprueba si una expresión es inferior a la otra o igual
Expr1 > expre 2	Comprueba si una expresión es superior a la otra
Expr1 > = expre 2	Comprueba si una expresión es superior a la otra o igual
! condition	Negación lógica de la condición
Cond 1 && cond 2	AND lógica entre dos condiciones verdaderas si y solo si las dos condiciones son verdaderas
Cond 1 || cond 2	OR lógica entre dos condiciones verdaderas si y solo si las dos condiciones son verdaderas

EXPRESIONES

Valor	Descripción
número	Un valor constante
variable	Un nombre de variable
Timer (n)	Valor del temporizador donde esta comprendido entre 0 y 3
Random (n)	Número aleatorio entre 0 y n
SensorValue(n)	Valor actual del sensor n donde n esta comprendido entre 0 y 2
Watch ( )	Valor del reloj del sistema
Message ( )	Valor del ultimo mensaje IR recibido

FUNCIONES

Función	Descripción	Ejemplo
SetSensor (sensor , config)	Configurar un sensor	SetSensor (SENSOR_1 , SENSOR_TOUCH)
SetSensorMode (sensor , modo)	Modo de configuración del sensor	SetSensorMode (sensor_2 , SENSOR_MODE_PERCENT)
SetSensorType (sensor, tipo)	Tipo de configuración de sensor	SetSensorType (sensor_2,SENSOR_TYPE_LIGHT)
SetDirection (salidas , dir)	Configura la dirección de salida	SetDirection (OUT_A , OUT_PWD)
SetPower (salidas , potencia)	Configura el nivel de potencia de salida (0-7). La potencia puede ser una expresión	SetPower (OUT_A, 6)
Wait (time)	Espera durante el tiempo especificado en 1/100 de segundo.El tiempo puede ser una expresión	Wait (x)
PlaySound (sonido)	Genera el sonido especificado (0-5)	PlaySound (SOUND_CLICK)
PlayTone (frec , duración)	Toca un tono de una especificada frecuencia durante un tiempo	PlayTone (440, 5)
ClearTimer (temporizador)	Reinicia el temporizador al valor 0	ClearTimer (0)
StopAllTasks ( )	Detiene todas las tareas que estén ejecutando	StopAllTasks ( )
SelectDisplay (modo)	Selecciona uno de los 7 modos de display de reloj del sistema	SelectDisplay (1)
SendMessage (mensaje)	Envía un mensaje IR	SendMessage (x)
ClearMessage ( )	Borra el buffer de mensaje	ClearMessage ( )
CreateDatalog (tamaño)	Crea Nuevo registro de datos del tamaño dado	CreateDatalog (100)
ClearSensor (sensor)	Borra el valor de un sensor	ClearSensor (SENSOR_1)
On (salidas)	Conecta una o más salidas	On (OUT_A + OUT_ B)
Off (salidas)	Desconecta una o más salidas	Off (OUT_C)
Float (salidas)	Detiene las salidas sin freno	Float (OUT_B)
Fwd (salidas)	Establece las salidas para dirección de avances	Fwd (OUT_A)
Rev (salidas)	Establece las salidas para dirección de retroceso	Rev (OUT_C)
Toggle (salidas)	Invierte la dirección de las salidas	Toggle (OUT_B)
OnFwd (salidas)	Arranca por un determinado número de segundos.	OnFwd (OUT_A)
OnRev (salidas)	Arranca con dirección de retroceso	OnRev (OUT_B)
OnFor (salidas , tiempo)	Arranca por un determinado tiempo	OnFor (OYT_A , 200)
SetOutput (salidas , modo)	Configura el modo de salida	SetOutput (OUT_A , OUT_ON)
SetDirection (salidas , modo)	Configura la dirección de salida	SetDirection (OUT_A , OUT_FWD)

8.BIBLIOGRAFÍA

http://lucas.hispalinux.es/Presentaciones/200002hispalinux/conf-16/16-html/programacion_LEGO_Mindstorm.html

http://www.akasa.bc.ca/tfm/lego_wr.html

http://www.daimi.aau.dk/dArk/Vaerktoejer.dir/RCX.vejledning.dir/Vejledning.html

http://www.ceeo.tufts.edu/graphics/robolab.html

http://graphics.stanford.edu/~kekoa/rcx/

http://www.pitsco-legodacta.com/products/robo-prog.htmhttp://www.pitsco-legodacta.com/products/robo-prog.htm

http://www.ni.com/company/robolab_video.htm

http://mindstorms.lego.com/community/resources/default.asp

http://ldaps.arc.nasa.gov/Workshop/VintageHills/programminggroup.html

http://www.microcontroladores.com/microbotica.asp

http://www.lego.com/dacta/robolab/downloadableinfo.htm

http://philohome.free.fr/sensors/rotsensor.htm

http://www.marioferrari.org/lego_mindstorm.html

http://usuario.tiscali.es/mindstorms/principal.htm

http://www.lego.com/eng/education/mindstorms/home.asp?menu=input&pagename=input

Http://Leo@wins.uva.nl **: fig: 2.41, 2.42, 2.43**

Http://el.eee.media.mit.edu/groups/el/Projects/constructopedia

Fig: 2.3, 2.4, 2.5, 2.10, 2.11, 2.13, 2.14, 2.15, 2.16, 2.17, 2.18, 2.19, 2.20, 2.21, 2.25, 2.27, 2.29, 2.35, 2.38, 2.56, 2.57, 2.61, 2.62, 2.63

http://usuario.tiscali.es/mindstorms/principal.htm

Fig: 2.6, 2.7, 2.8, 2.9, 2.12, 2.22, 2.23, 2.24, 2.26, 2.28, 2.30, 2.31, 2.32, 2.33, 2.34, 2.36, 2.37, 2.39, 2.40, 2.44, 2.45, 2.46, 2.47, 2.48, 2.49, 2.50, 2.51, 2.52, 2.53, 2.54, 2.55, 2.58, 2.59, 2.60, 2.64, 2.65, 2.66

Http://www.textbrick.com

Fig 2.28B

Http://www.lm-software.com/mlcad/

Fig 2.67