Análisis de lenguajes programación orientados a niños.

Andrés Baamonde Lozano (andres.baamonde@udc.es)
Rodrigo Arias Mallo (rodrigo.arias@udc.es)

17 de noviembre de 2014

Índice

1. Lego Mindstorms EV3

1.1. Introducción

Mindstorms EV3 es un lenguaje de programación creado para interactuar con el kit Mindstorms EV3 de Lego. Esta diseñado para robots modulares que se construyen con piezas de Lego, por lo que son de coste «reducido». Estos robots utilizan piezas de la línea Technic con sensores y actuadores de bajo coste, con un módulo de control (brick) que tiene la potencia de un smartphone de gama baja.

1.2. Características del brick

- Procesador ARM9 @300MHz.
- 16 MB Flash (Linux).
- 64 MB RAM.
- 4 puertos de entrada para sensores y 4 puertos de salida para actuadores.
- Ranura microSDHC (hasta 32 GB). Se puede arrancar un S.O. diferente al que trae de serie desde una tarjeta en esta ranura.
- Bluetooth.
- Soporta wifi (sin encriptación o con WPA2) conectando dongle NetGear WNA1100 en puerto USB.
- Se pueden conectar hasta 4 bricks en daisy chaining para ampliar el número de puertos para sensores y actuadores (el programa se ejecuta solo en uno de los bricks).

1.3. Sofware de Programación

1.3.1. De ejecución del brick

- Mindstorms EV3(el que trataremos).
- RobotC for LEGO Mindstorms.
- MonoBrick(Alfa).
- LabVIEW(En desarrollo).
- BricxCC(En desarrollo).
- Python EV3(En desarrollo).
- LeJOS(Beta).

1.3.2. De control remoto

- Microsoft EV3 API.
- MonoBrick.
- RWTH Toolbox para Matlab(En desarrollo).

2. Mindstorms EV3

2.1. Características

2.1.1. Pros

- Muy fácil construir comportamientos reactivos / programar autómatas.
- Curva de aprendizaje muy rápida para quien no haya programado antes en otro lenguaje.
- Fomenta la documentación de los programas.
- Ejecución paralela trivial.
- Herramienta de data logging muy sencilla de utilizar.
- Muy buena documentación disponible en su propio servidor web (http://localhost:58401).

2.1.2. Contras

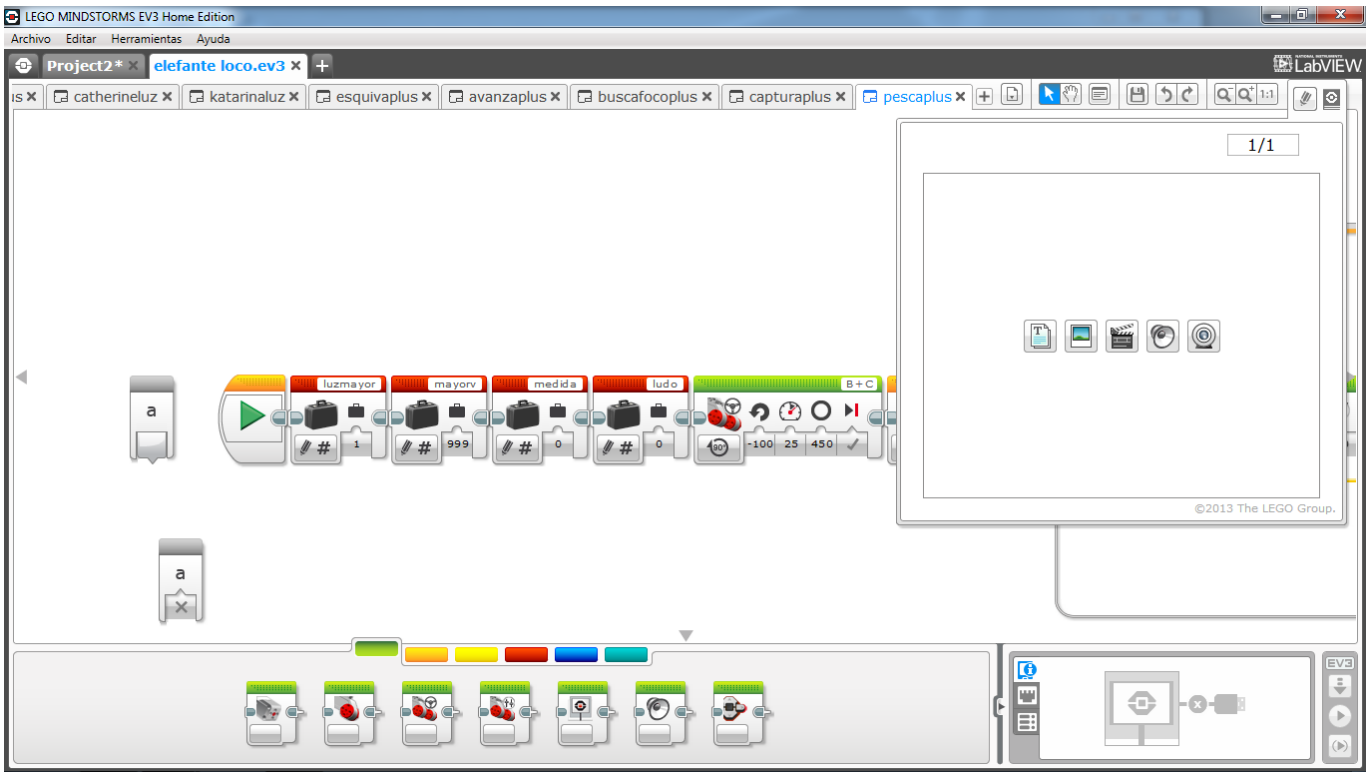
- Tedioso manejar estructuras de datos complejas.
- La programación es lenta si se compara con un lenguaje tradicional una vez que se domina éste.
- Programas medianos o grandes complicados de gestionar. Es necesario acostumbrarse a particionar todo en bloques propios.
- No hay simulador (aunque se puede acoplar al Robot Virtual Worlds).

2.2. El entorno de programación

2.2.1. Ventana

Al abrir el IDE, se nos muestra una pantalla donde comenzar a diseñar nuestro programa, empleando bloques. Se muestra en la figura 1.

Figura 1: Ventana principal.



2.2.2. Proyecto

Como en la mayoría de IDES los archivos se organizan en proyectos, que pueden contener programas experimentos imágenes y archivos de sonido. Los experimentos son programas que crea automáticamente el IDE para capturar datos. En la figura 2 se puede observar los componentes de la ventana de proyecto.

2.2.3. Página de hardware

Aquí muestra la información sobre el bloque EV3 que está conectado, versión de firmware, tipo de conexión (entre el pc y el bloque), configuración inalámbrica, barra de memoria, etc. También están las opciones de descargar programa al brick, descargar y ejecutar o ejecutar seleccionados.

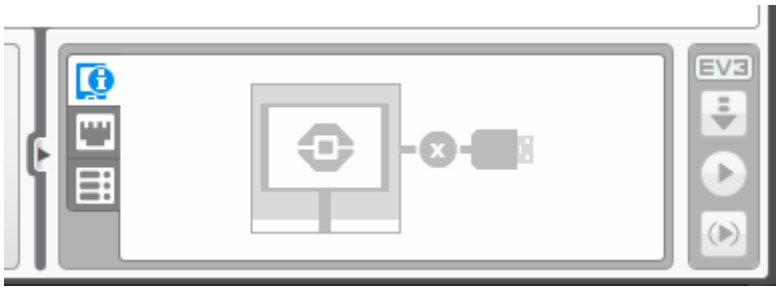
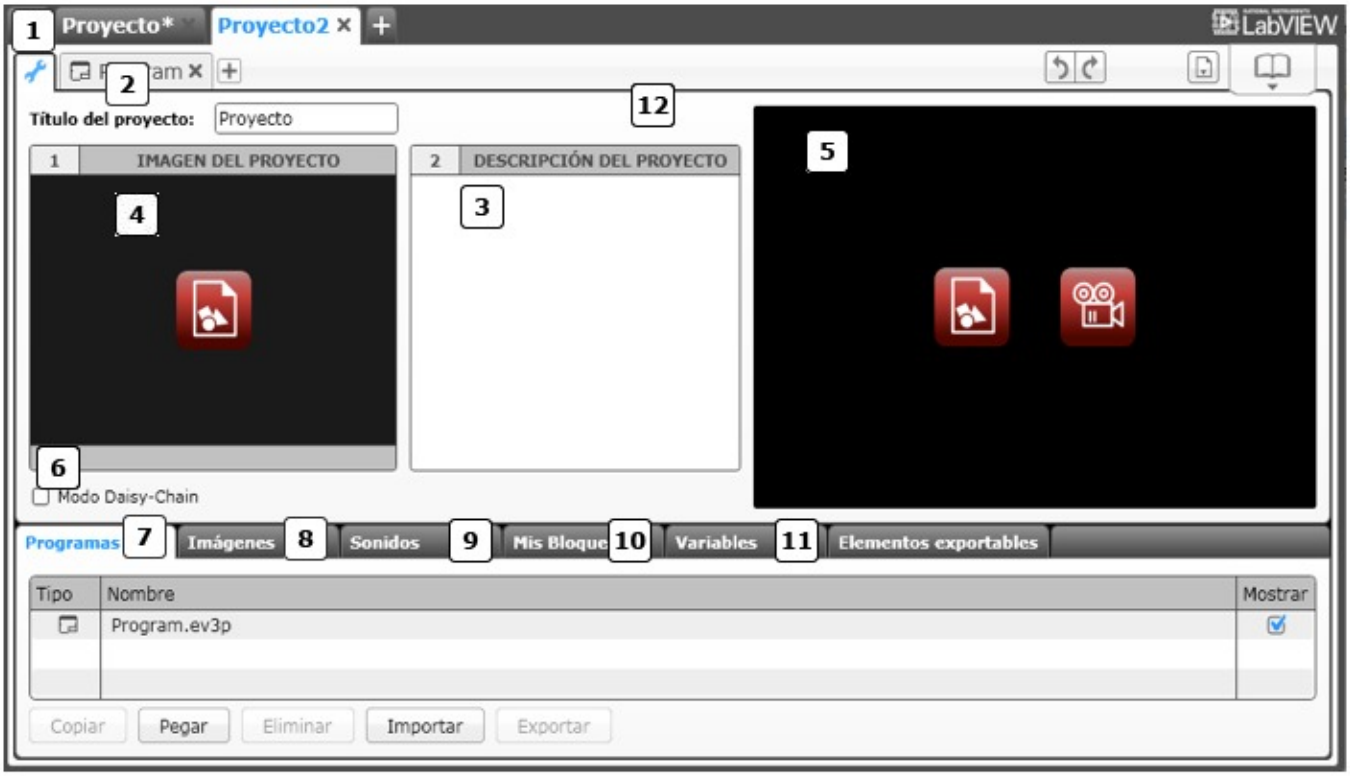


Figura 2: Ventana de proyecto.



1. Propiedades del proyecto
2. Título del proyecto: Este título aparece en el Vestíbulo
3. Descripción del proyecto: Escribe una descripción de su proyecto
4. Imagen del proyecto: Agrega una imagen JPG o PNG
5. Vídeo del proyecto: Agrega un vídeo y una imagen de cubierta
6. Modo Daisy Chain: Conecta varios Bloques EV3
7. Programas: Lista de Programas del proyecto
8. Imágenes: Lista de Imágenes del proyecto
9. Sonidos: Lista de Sonidos del proyecto
10. Mis Bloques: Lista de Mis Bloques del proyecto
11. Variables
12. Compartir proyecto: Haga clic en Compartir proyecto para ingresar al sitio web de Mindstorms. Aquí puede compartir su creación con el mundo.

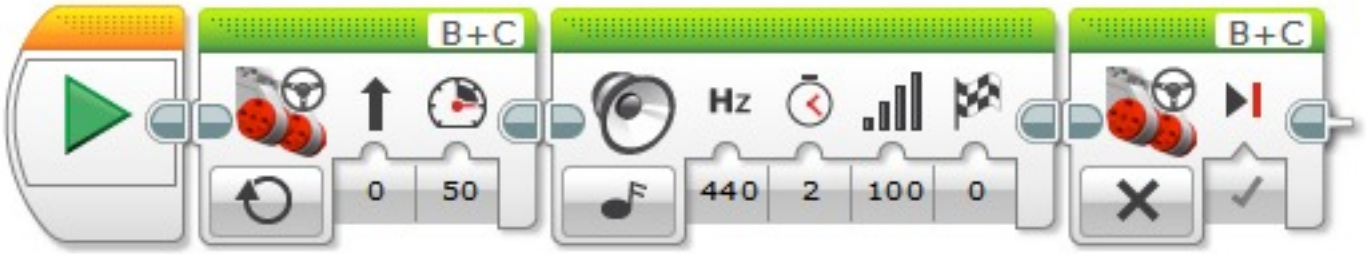
3. Lenguaje de Programación

3.1. Introducción

Los programas se crean arrastrando bloques (que tienen diferentes comportamientos como variables control de flujo, sensores o actuadores) que representan los elementos básicos del lenguaje. Todo programa empieza por el bloque iniciar y su ejecución va de izquierda a derecha. Algunos bloques tienen una ejecución no bloqueante es decir, su acción persiste mientras otro bloque posterior no la finalice explícitamente. La ejecución finaliza cuando se llega al extremo derecho y no hay mas bloques o explícitamente con el bloque finalizar.

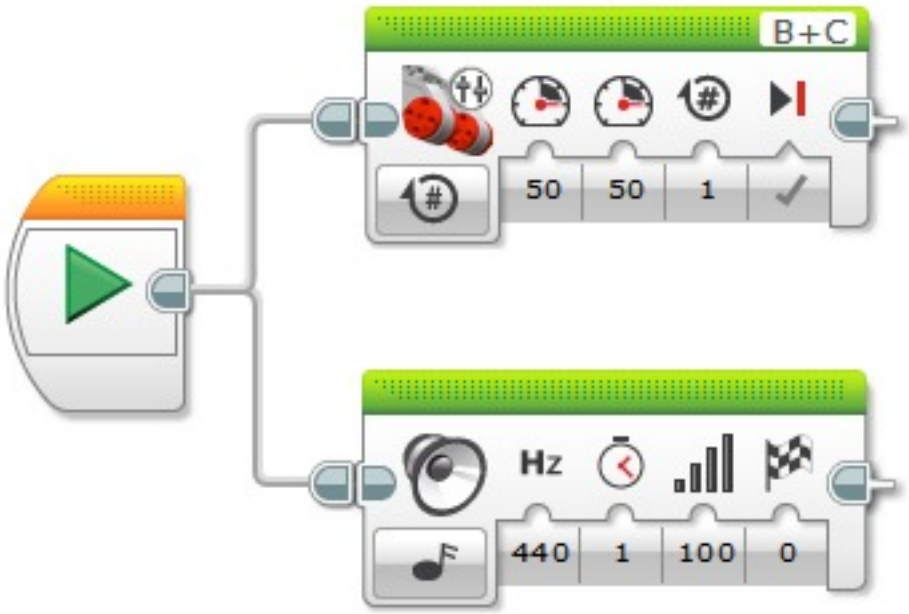
3.2. Secuencias de Bloques

Cuando los bloques están juntos o unidos por un cable forman una secuencia de bloques.



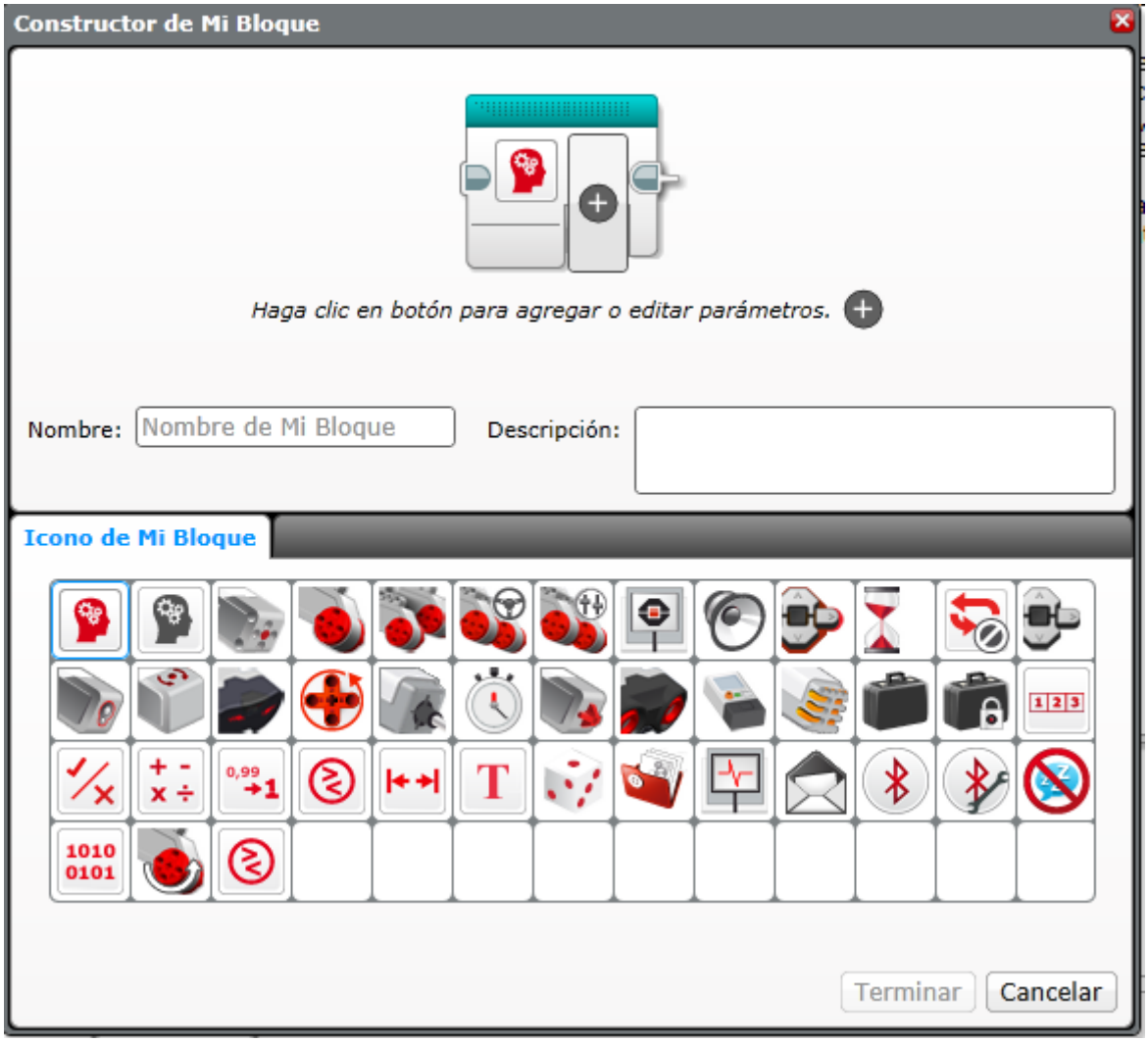
3.3. Secuencias Paralelas

Cuando los bloques están en paralelo, ya sea por el bloque iniciar o a partir de un bloque en concreto su ejecución es igual que en secuencia pero eso si, todas las secuencias comparten las variables y en caso de conflicto su comportamiento es impredecible ya que solo está garantizada la ejecución atómica de cada bloque individualmente.



3.4. Creación de bloques

Este lenguaje fomenta la aplicación de «funciones» que sería lo análogo a la creación de un bloque propio para agrupar secuencias de bloques con un comportamiento concreto. Para ello se emplea el «Constructor de Mi Bloque» en el que se definen: Nombre, icono, tipo de dato y valor por defecto para sus respectivas entradas y salidas.



Una vez creados, estos bloques se pueden cambiar pero no se pueden editar ni sus entradas ni sus salidas.

3.5. Tipos de datos

Existen cinco tipos de datos básicos para representar la información del programa. Son los siguientes.

- Numérico
- Lógico
- Texto
- Secuencia numérica
- Secuencia lógica

El texto puede ser unicode pero la pantalla del brick solo representa los caracteres ASCII de 7 bits. Las secuencias son arrays unidimensionales (no existe otro tipo)

3.6. Variables y constantes

El lenguaje soporta los conceptos tradicionales pero son tediosas de usar.

3.6.1. Variables

Se usan si un dato va a ser accedido y actualizado en diferentes puntos del programa o si se necesita un dato en diferentes secuencias (ya que se comparten).

3.6.2. Constantes

Se usan si un dato solo va a ser accedido.

3.7. Cables de datos

Este lenguaje trata de evitar uso de variables, los cables de datos con conexiones entre los bloques para pasar los resultados de otros bloques, es decir son un paso por valor entre bloques. El bloque origen debe estar antes que el bloque de destino en la secuencia y el bloque destino no se ejecuta hasta que el dato está disponible por lo que es posible usar cables de datos para sincronizar secuencias paralelas. Estos cables pueden pasar todos los tipos de datos, y se puede ver su contenido durante la ejecución de un programa.

3.8. Bloques predefinidos

3.8.1. Bloques de acción

Bloques de los motores (mediano y grande), mover la dirección, mover en tanque, pantalla, sonido y luz de estado EV3.

3.8.2. Bloques de control de flujo

Figura 3: Bloque iniciar y bloque finalizar

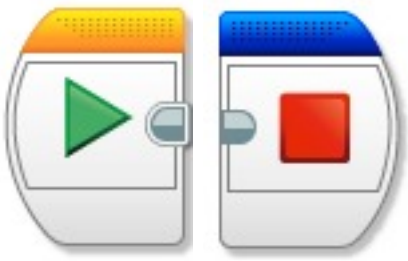


Figura 4: Bloque de flujo condicional if-then-else.

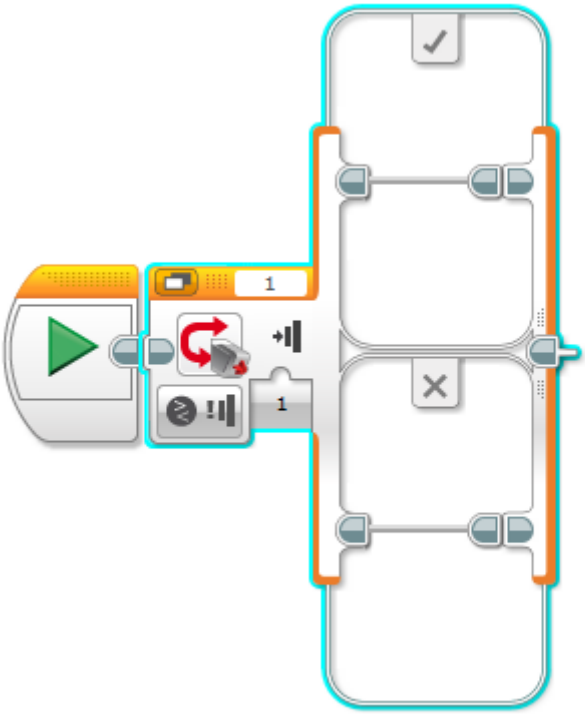


Figura 5: Bloque de bucle y condición para finalizar el bucle.

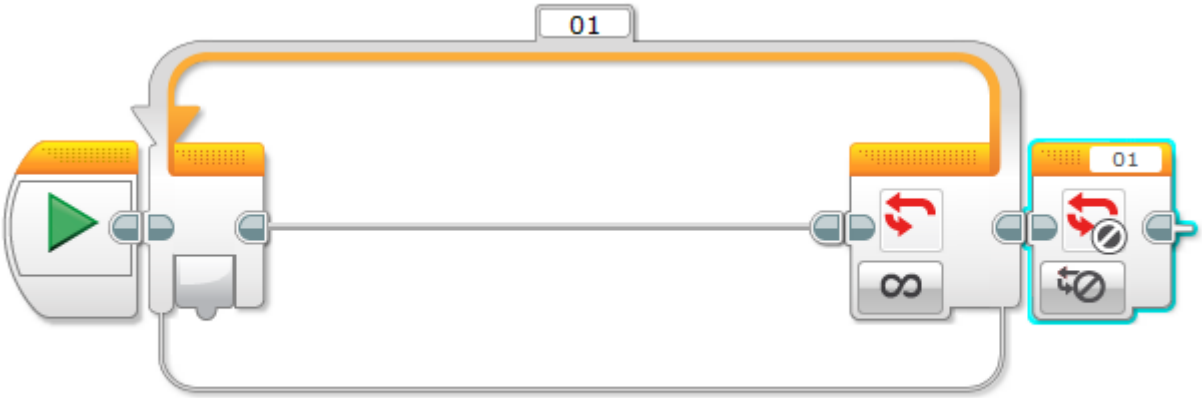


Figura 6: Bloque de retardo o sleep

3.8.3. Bloques de sensores

Botones del brick, sensor de color, sensor de giro, sensor infrarrojo, rotación del motor, sensor de temperatura, temporizador, sensor táctil, sensor ultrasónico, medidor de energía y sensor de sonido NXT.

3.8.4. Bloques de datos

Variable, constante, operaciones secuenciales, operaciones lógicas, matemáticas, redondear, comparar, alcance, texto y aleatorio.

3.8.5. Bloques de Avanzados

Acceso a archivo, registro de datos, mandar mensaje, conexión bluetooth,mantener archivo, valor del sensor sin procesar, motor sin regular, invertir el motor, detener el programa.

3.8.6. Mis bloques

Aquí estarán los bloques creados.

3.9. Flujo de un programa

3.9.1. Iniciar

Marca el inicio de una secuencia de bloques ya que puede haber más de una secuencia. Todas las secuencias se inician automáticamente cuando se inicia el programa y se ejecutan al mismo tiempo.

3.9.2. detener

Finaliza una secuencia de bloques aunque su uso no es obligatorio.Finaliza todas las secuencias en ejecución.

3.10. E/S básica

3.10.1. Pantalla

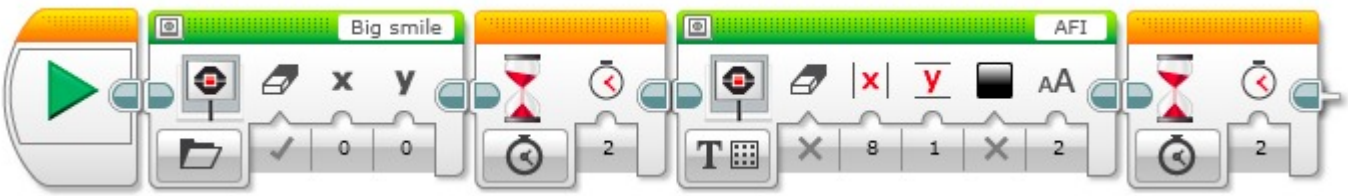
Muestra texto o gráficos.Modos de funcionamiento:

- Texto
- Formas
- Imagen
- Reiniciar

Ejemplo de uso :

- Iniciar.
- Escribir en pantalla - Borrar pantalla.
- Esperar 2 segundos.
- Escribir texto y no borrar la pantalla.
- Esperar 2 segundos.

Figura 7: Secuencia de ejemplo de uso pantalla



3.10.2. Luz de estado

Controla la luz de estado del brick puede ser un pulso continuo o con pulsos.

- Encendido
- Apagar
- Reiniciar

Ejemplo de uso :

- iniciar
- esperar
- luz de estado roja sin pulso
- esperar
- luz de estado naranja con pulso

- esperar

Figura 8: Secuencia de ejemplo de uso luz estado

3.10.3. Sonido

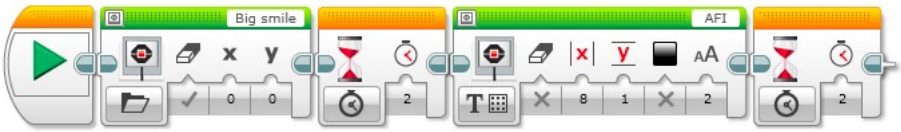
Reproduce sonido en el altavoz del brick, se puede ajustar el volumen y reproducir en bucle.Esta reproducción puede ser bloqueante o no bloqueante. Modos de funcionamiento :

- Archivo
- Tono
- Nota
- Detener

Ejemplo de uso:

- Iniciar
- Sonido (archivo)
- Sonido (tono)
- Sonido (nota)
- Esperar
- Sonido (detener)

Figura 9: Secuencia de ejemplo de uso Sonido



3.10.4. Botones del Brick

Obtiene los datos de los botones del brick. Modos de funcionamiento:

- Botones (muestra botón presionado).
- modo comparación para botones.

Figura 10: Bloque de botones del brick

3.10.5. Motores (mover la dirección)

Se asume que el robot tiene dos motores grandes, uno para el lado derecho del vehículo y otro para el lado izquierdo.Este bloque mueve el vehículo con un valor de potencia indicado(entre -100 y 100) y una dirección deseada. Modos de funcionamiento:

- Encendido.
- Encendido por segundos.
- Encendido por grados o rotaciones.
- Apagado

Figura 11: Bloque motor dirección

3.10.6. Motores (mover Tanque)

Este bloque mueve el vehículo con un valor de potencia diferente para cada motor. Modos de funcionamiento:

- Encendido.
- Encendido por segundos.
- Encendido por grados o rotaciones.
- Apagado

Figura 12: Bloque motor Tanque

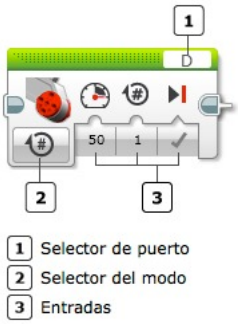
3.10.7. Motores (motor grande y motor mediano)

Estes bloques mueven el vehículo con un valor de potencia diferente para cada motor.Un número positivo implica un giro en el sentido de las agujas del reloj y un número negativo un giro en el sentido contrario al de las agujas del reloj. Modos de funcionamiento:

- Encendido.
- Encendido por segundos.
- Encendido por grados o rotaciones.
- Apagado

Figura 13: Motores mediano y grande

(a) Bloque motor Grande



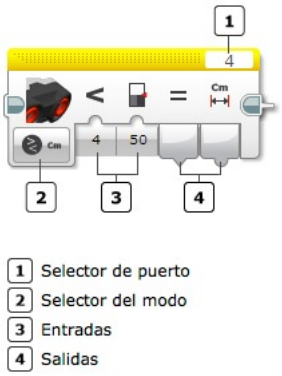
(b) Bloque motor mediano

3.10.8. Sensor de ultrasonidos

Envía ondas de sonido de alta frecuencia y mide cuánto tarda el sonido en reflejarse de vuelta al sensor con un rango de detección de 3 a 250cm. Modos de funcionamiento :

- Distancia (al objeto más cercano)
- presencia (detecta señales de ultrasonidos de otras fuentes)
- Para los anteriores modo existe un modo comparación con un valor especificado

Figura 14: Bloque sensor ultrasonidos



3.10.9. Sensor de infrarrojos

Envía luz en el espectro de los infrarrojos y detecta el reflejo de esta luz en otros objetos(hasta 70cm). Modos de funcionamiento :

- Proximidad
- Baliza(remoto)
- Baliza(posición)
- Para cada modo anterior existe un modo comparación respecto a un valor

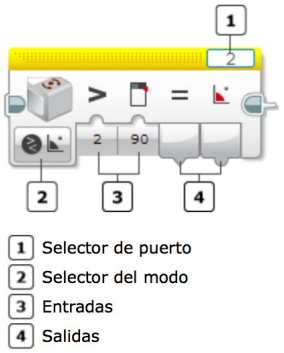
Figura 15: Bloque sensor de infrarrojos

3.10.10. Sensor giroscopio

Detecta el movimiento de rotación en un plano(máximo 400º/seg) Modos de funcionamiento :

- Angulo
- Velocidad de rotación
- Modo comparación y a mayores un reinicio para poner la referencia a 0º

Figura 16: Bloque giroscopio

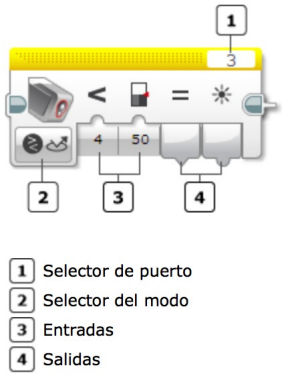


3.10.11. Sensor de color

Detecta color o intensidad de luz. Modos de funcionamiento :

- color (entre 7 colores : negro, azul, verde, amarillo, rojo, blanco, marrón y sin color)
- intensidad de luz reflejada
- intensidad de luz ambiente
- Modo comparación para todos estos métodos.
- Calibrar mínimo(intensidad de luz a 0)
- Calibrar máximo(intensidad de luz a 100)
- Reinicio

Figura 17: Bloque sensor de color

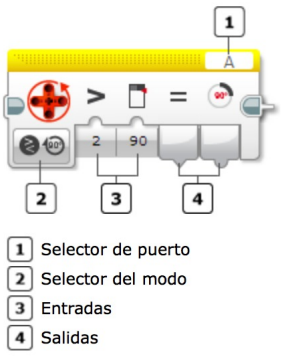


3.10.12. Sensor de rotación del motor

Obtiene información de un motor de lo que esta girando o de la potencia aplicada. Modos de funcionamiento :

- Rotación
- Potencia
- Modo comparación para estos dos últimos
- Reinicio

Figura 18: Bloque sensor rotación del motor

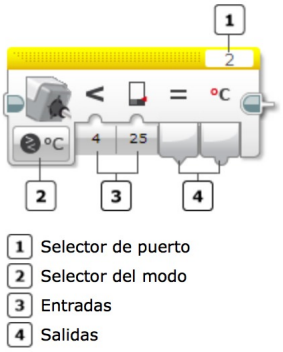


3.10.13. Sensor de contacto

Modos de funcionamiento :

- Estado (si está pulsado o no)
- Modo de comparación para el anterior modo

Figura 19: Bloque sensor de contacto



3.10.14. Sensor de temperatura

Mide la temperatura en la punta de su sonda metálica. El sensor mide en grados Celsius (de -20°C a 120°C) y Fahrenheit (de -4°F a 248°F) con una exactitud de 0,1°C. Modos de funcionamiento :

- Temperatura
- Para el modo anterior existe un modo comparación con un valor.

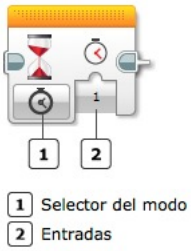
Figura 20: Bloque sensor de temperatura

3.10.15. Temporizador

Obtiene tiempo de alguno de los temporizadores del brick(que son 8) Modos de funcionamiento :

- Tiempo
- Modo comparación para el tiempo
- Reinicio

Figura 21: Bloque temporizador



3.10.16. Espera

Modo que espera a que suceda algo antse de continuar con el siguiente bloque de secuencia Modos de funcionamiento :

- Tiempo
- Sensor
- Mensaje(por un mensaje bluetooth de tipo especificado)

Figura 22: Bloque espera

3.11. Control de flujo

3.11.1. Condicionales

Bloque interruptor, contiene dos o mas secuencias de bloques de programación(casos). Estos casos se pueden mostrar en pestañas y puede existir un caso por defecto, puede actuar como if.. else o un switch.Modos de funcionamiento:

- Sensor(comprobar si el valor de un sensor cumple o no una condición)
- Lógico (True o False para una sentencia determinada)

- Texto(compara texto)
- Numérico(compara número)

3.11.2. Bucles

Repite una secuencia de bloques contenida en su interior. Modos de funcionamiento:

- Ilimitado (While True)
- Cuenta (repite número limitado de veces)
- Tiempo (durante un tiempo especificado)
- Lógico(hasta que una entrada logica se cumpla)
- Sensor (Hasta que un sensor tenga un valor determinado)
- Mensaje (Hasta que se recibe un mensaje determinado)

3.11.3. Break

Interrupción de un bucle, provoca la finalización de un bucle de manera anticipada. Se puede especificar que bucle quiere interrumpirse por medio de su etiqueta y desde cualquier parte del código(Muy útil para secuencias en paralelo)

4. Ejemplo de uso: Robot

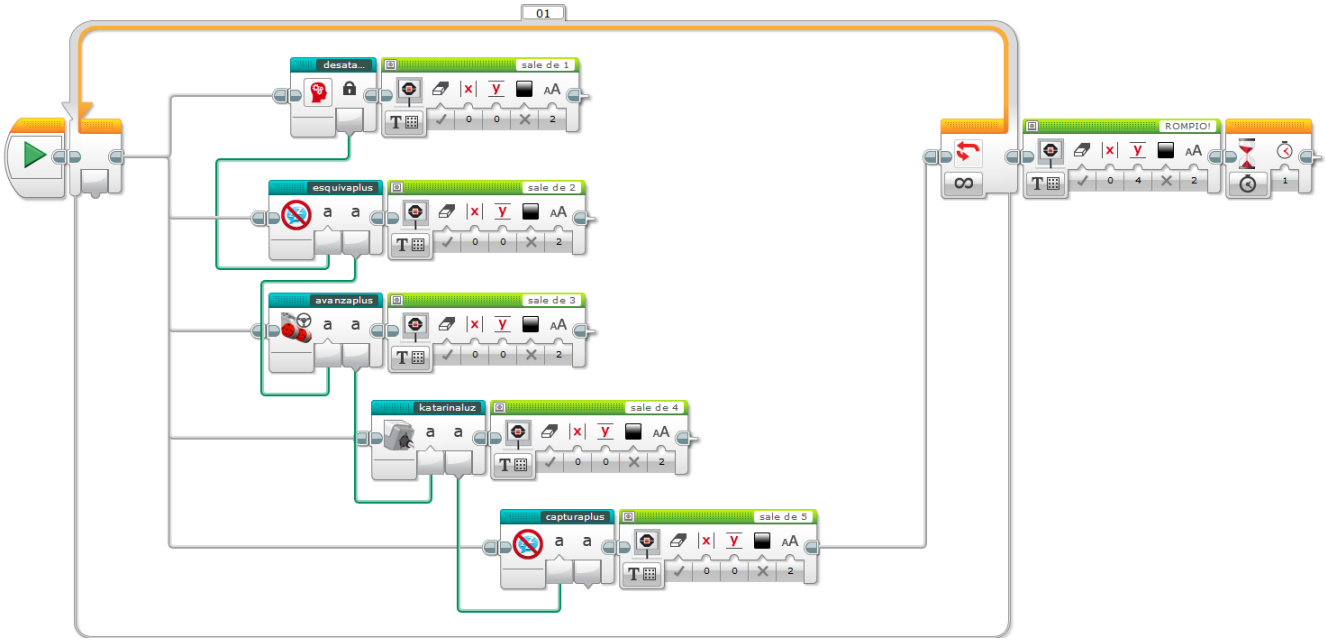
En este ejemplo veremos un robot que tiene el siguiente comportamiento :

- Sale de situaciones de colisión.
- Evita situaciones de colisión.
- Avanza
- Se dirige hacia la Luz.
- recoge una caja de determinado color.

Se incluyen estas imágenes en la carpeta img/ para poder verlas con un mayor detalle, sus nombres son de la forma Robot*.png

4.1. Controlador Global

Figura 23: Controlador global robot



Se realizan de forma paralela pero controlando su ejecución con variables bool, cada modulo activa el anterior para asegurarnos unas precondiciones antes de ejecutar un determinado comportamiento.Cuando cambia de modo se imprime en la pantalla del brick.

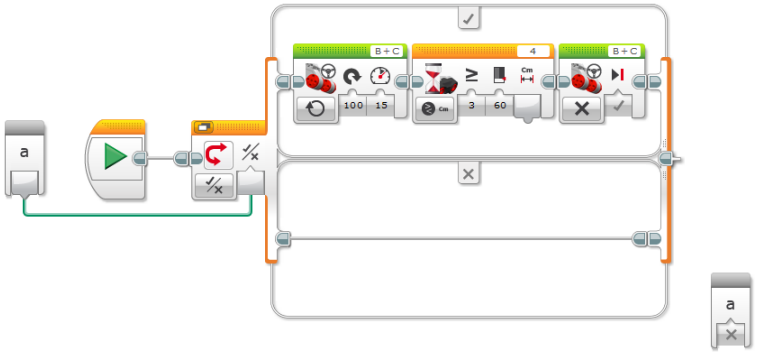
4.2. Salir de situaciones de Colisión

Figura 24: salir de situaciones de colisión

Gira hasta que su pulsador, situado en la parte delantera es soltado, este se presionaria si el robot chocase con algo.Este comportamiento aparece muy pocas veces ya que se suelen eviar las situaciones de colisión.De ahí su sencillez.

4.3. Evitar situaciones de colisión

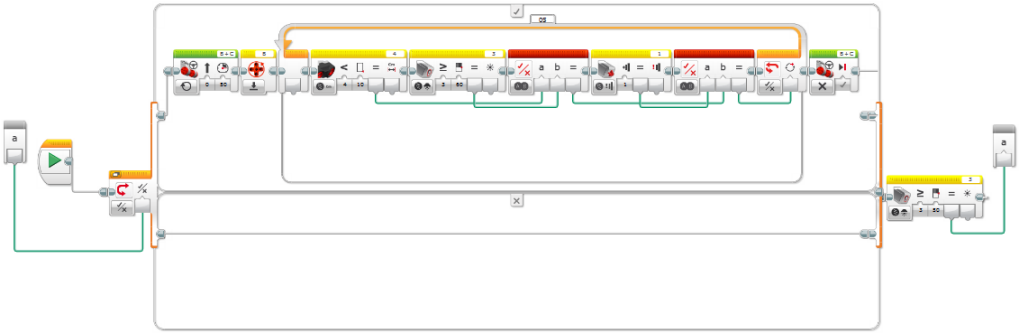
Figura 25: Evitar colisiones con ultrasonidos



Usa el sensor de ultrasonidos para saber a que distancia tiene la pared hacia la que está avanzando, cuando esta distancia llega a un umbral, gira.

4.4. Avanzar

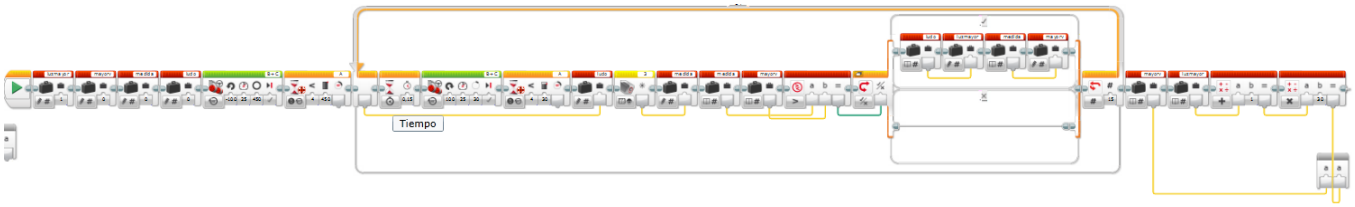
Figura 26: avanzar



Avanza mientras la luz no decrezca respecto a una medida anterior es decir, se esté alejando de la luz.

4.5. Se dirige hacia la Luz

Figura 27: capturar luz



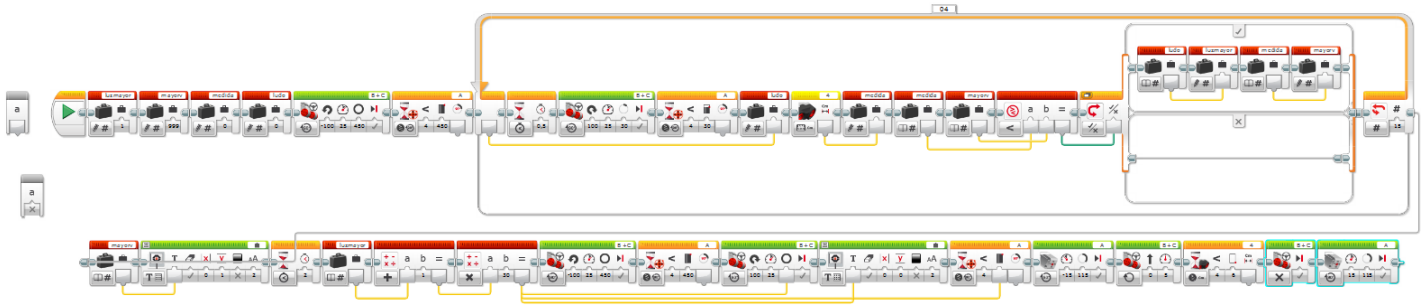
Gira hacia la luz, haciendo diferentes medidas previas.

4.6. Recoge una caja

Aquí diferenciamos dos comportamientos, encarar la caja y acercarse para posteriormente recogerla.

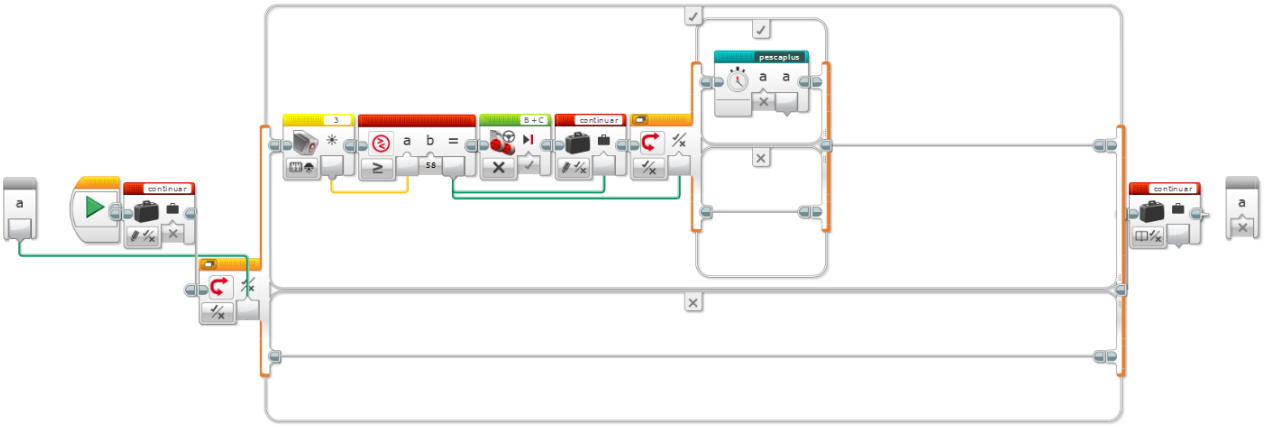
4.6.1. Acercarse a la caja

Figura 28: Acercarse a la caja



4.6.2. Recoger la caja

Figura 29: recoger la caja



5. Comparación con lenguajes tradicionales.

5.1. Memoria

En lenguajes tradicionales como Fortran, C o Pascal, es necesario reservar y liberar la memoria de forma manual.

Sin embargo en lenguaje de bloques de Lego, la memoria se maneja de forma automática. No es necesario reservar espacio explícitamente para almacenar una variable, y posteriormente liberarlo. De modo que incluye un recolector de basura para las variables que ya no se emplean.

5.2. Intérprete

El IDE es en realidad un compilador a un lenguaje intermedio llamado bytecode que es propiedad de Lego. Cada bloque es transcrito en una serie de instrucciones, que forman un «clump». Estos fragmentos, se tratan como tareas indivisibles, y son las que se ejecutarán siguiendo el orden establecido por la máquina virtual de Lego. Para la ejecución de un «clump», todos los bloques de los que depende, deben haberse ejecutado. Esto posibilita la paralelización de las ejecuciones.

5.3. Paralelismo

Algunos lenguajes de alto nivel, proporcionan un mecanismo sencillo para manejar operaciones que se realizan en segundo plano. Otros, sin embargo no incluyen dicha facilidad, relegando al programador la tarea de diseñar un mecanismo para realizar varias acciones a la vez.

En un lenguaje, donde se busca la simplicidad y facilidad de uso, el paralelismo se realiza de forma transparente para el usuario. De forma que los mecanismos para controlar que tarea se debe realizar, y de que manera se comparte el cpu entre tareas, quedan fuera del alcance del programador.

Para realizar una ejecución paralela, basta con emplear una bifurcación en el cable que conecta dos bloques, para incluir dos o mas bloques cuya conexión parte del mismo lugar. De esta forma cualquier bloque conectado, será ejecutado en paralelo.

El modo de resolver los problemas de acceso entre las variables se soluciona compartiendolas de forma global. Sin embargo, de forma interna, se mantiene un registro de todas las variables, protegidas mediante mutex, para el correcto funcionamiento de los bloques en paralelo.

5.4. Funciones

Las clásicas funciones, son ahora reemplazadas por bloques que engloban un funcionamiento concreto. Son aquellos creados por el mecanismo de bloques propios, y que realizan un conjunto de operaciones definidas en un plano aparte del programa. Lo que permite centrarse en la resolución de un problema, pequeño, y concreto.

Los argumentos de las funciones, son siempre pasados por valor, empleando las conexiones a unos bloques especiales.

5.5. Recursión

Lego Mindstorms EV3 no soporta la recursión. Sin embargo hay otros entornos como LeJOS EV3 (Java, gratuíto) y ROBOTC (C, no gratuíto), que sí la soportan.

5.6. Polimorfismo

De igual manera que lenguajes de alto nivel como Java, soportan el polimorfismo, el bytecode de Lego también lo hace. Sin embargo, se realiza de manera transparente para el usuario. Por ejemplo, en la comparación de dos enteros de distinto tamaño, 32 y 16 bits, se realiza previamente una conversión a tipos de 32 bits.