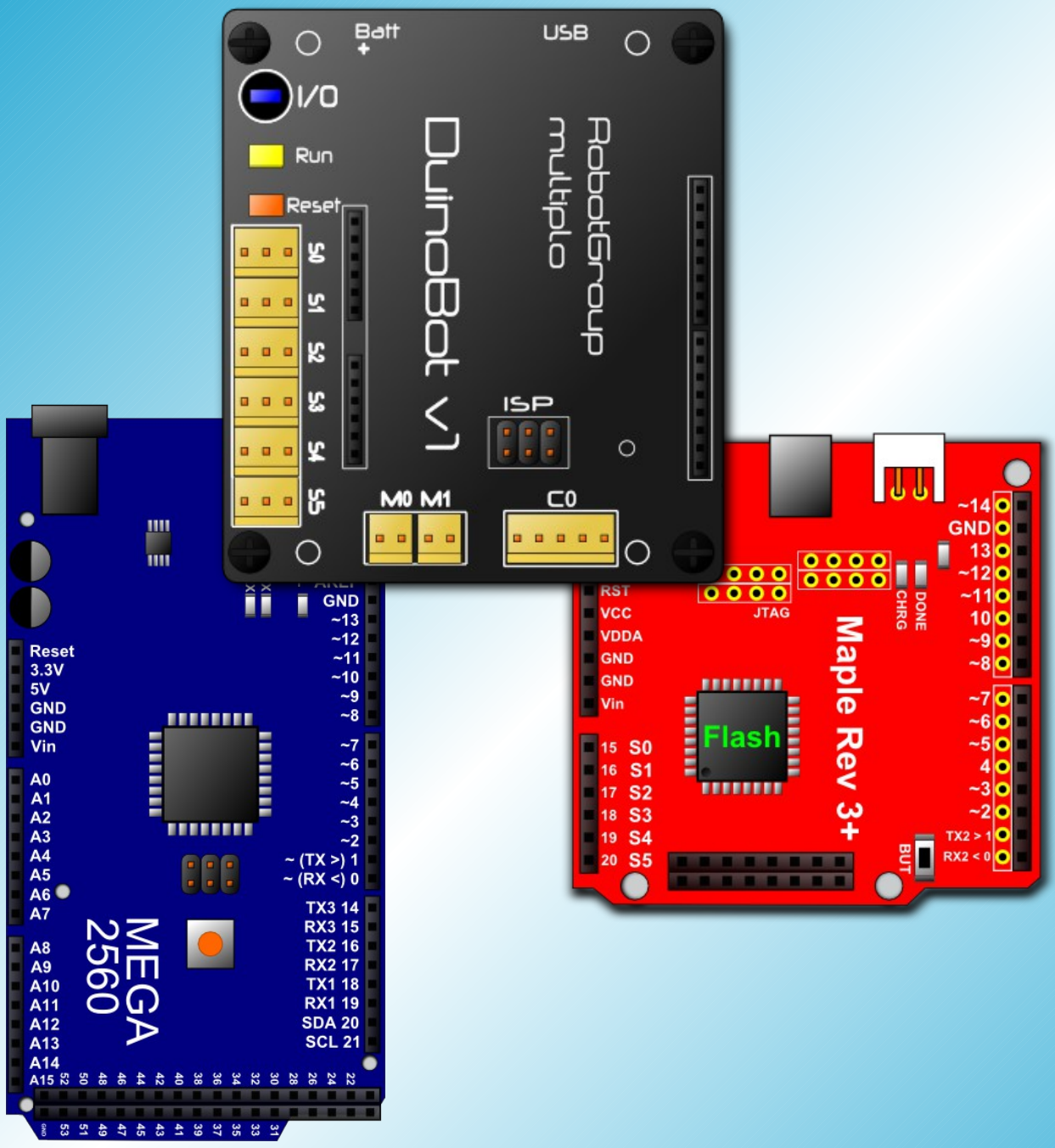



# Minibloq

## v0.81.Beta

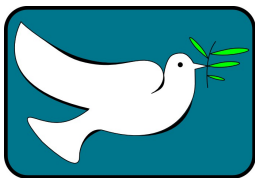
## Manual del usuario

## Autor

Julián U. da Silva Gillig

## Copyright y licencias





## Marcas / Trademarks



## Descargo de responsabilidad / Disclaimer

# Contenido

# 1. Introducción

Minibloq es un entorno de programación gráfica (o icónica) que tiene por principal objetivo facilitar la programación y su aprendizaje, ya sea a niños, principiantes, y personas con pocos conocimientos de informática. Minibloq es un proyecto activo y en permanente evolución, y es posible que en el futuro sirva también como lenguaje genérico para todo tipo de aplicaciones y para usuarios avanzados. Actualmente está orientado especialmente a la programación de dispositivos de computación física y robótica, tales como kits Multiplo, placas Arduino™, etc..

Algunas de las principales características de Minibloq son:

- **Fácil**: Sólo algunos clicks, y tu primer programa estará funcionando.

- **Generación de código en tiempo real**: Crea el código a medida que agregas bloques o modificas parámetros, mostrándolo instantáneamente en una ventana con coloreo de sintaxis. De esta forma, Minibloq facilita la transición hacia la programación basada en texto.

- **Control de errores en tiempo real**: Minibloq marca en rojo todos los errores y parámetros incompletos a medida que creas el programa.

- **Interfaz de usuario avanzada**: Drag & drop con autopan, Zoom, cortar y pegar, ventanas acoplables y navegación en el editor tanto por mouse como por teclado son sólo algunas de las características de la interfaz de usuario (GUI) de Minibloq. Y hay más...

- **Terminal embebido**: Permite recibir y enviar datos desde y hacia tu placa a través de puertos USB o serie.

- **Solución "todo en uno"**: El software viene listo para usar, incluyendo ya todo lo necesario. No hay necesidad de instalar otras librerías, herramientas de software, etc..

- **Portable**: No requiere instalación (excepto para los drivers de dispositivos, como los necesarios para que funcionen placas como Arduino™). Puede incluso corrrer desde un pendrive flash. Además, no requiere conexión a Internet para funcionar, ya que reside completamente en la máquina en la que se está ejecutando. Es más: Puedes tener diferentes copias de Minibloq, incluso configuradas para trabajar con diferente hardware, todas funcionando a la vez en la misma computadora.

- **Rápido**: Minibloq es una aplicación nativa, compilada con C++ (gcc), utilizando una librería llamada wxWidgets. Por esta razón, es perfectamente apto para funcionar en pequeñas computadoras, tales como netbooks. Además, compila realmente rápido tus programas, ya que contiene código precompilado.

- **Modular y expandible**: Los usuarios avanzados pueden crear sus propios bloques (en próximas versiones habrá facilidades nuevas para crear nuevos bloques de forma rápida y sencilla).

# 2. Instalación

## 2.1. Descarga

La versión actual del software se encuentra disponible para descargar de forma gratuita en diferentes sitios web:

- http://minibloq.org
- http://www.robotgroup.com.ar/web/index.php?option=com_content&view=section&layout=blog&id=2&Itemid=6&lang=es
- http://www.mexchip.com/2011/10/minibloq-beta-disponible-para-descargar/

## 2.2. Instalación

### 2.1.1. Drivers de las placas controladoras

Minibloq es software portable en el sentido de que en verdad no requiere instalación (al menos en Windows, ya que en Linux necesita ciertas dependencias, como se describirá luego). Ahora bien, que Minibloq no requiera instalación, no quiere decir que el hardware soportado, el cual proviene además de diferentes fabricantes, no requiera de procesos de instalación para sus drivers (o controladores de dispositivos, como se los suele llamar también). En general, tanto las placas de RobotGroup, como Arduino, Maple, o las fabricadas por Seeed Studio, tienen todas su propio proceso de instalación de drivers, el cual varía además ligeramente entre una versión y otra de Windows. Esta operatoria es común a todos los entornos de programación para estas plataformas open source de computación física y robótica. Es por esto que con Minibloq se han incluido drivers para todas las placas soportadas, los cuales se pueden encontrar en el siguiente subdirectorio (dentro del directorio donde se haya instalado Minibloq):

**\Components\Drivers**

Allí se encontrará a su vez un subdirectorio para cada familia de hardware soportada. La documentación de instalación de drivers la suele proporcionar cada fabricante en su sitio web, y no forma parte de la distribución de Minibloq. De todos modos, siempre es posible acudir al forum de Minibloq por problemas específicos en la instalación de drivers de hardware:

http://minibloq.net/forum

### 2.1.2. Minibloq en Windows

Minibloq es software portable en el sentido de no requiere instalación. Puede incluso en teoría correr desde un pendrive, y en las pruebas realizadas sobre medios similares ha respondido satisfactoriamente. Actualmente corre en Windows XP, Vista y Seven, tanto en versiones de 32 como de 64 bits. Para instalarlo en entornos Windows, basta con descomprimir el archivo zip de la distribución estándar (al cual se lo puede obtener, como se mencionó antes, de

minibloq.org, o de los sitios espejo -mirrors-). Pero para comodidad de los usuarios menos experimentados, ha sido desarrollado también un pequeño instalador ejecutable, que hace la tarea de descompresión de forma automática. Una vez descomprimido e instalado, el usuario debe buscar y ejectuar el archivo **MinibloqRun.exe**.

### 2.1.3. Minibloq en Linux

Si bien la versión actual de Minibloq corre bajo Windows, es posible en general, utilizarlo en Linux, aunque quizá con algunas limitaciones. Para esto debe instalarse Wine (se está trabajando en una versión nativa para Linux, pero no estaba lista a la hora de publicar este documento). En distribuciones basadas en Debian (como por ejemplo Ubuntu, o la distribución argentina para educación Lihuen), en general los pasos son similares a los que se describen a continuación:

```
sudo apt-get install wine
winecfg
sudo usermod -aG dialout <myuser>
sudo ln -sf /dev/ttyUSB0 ~/.wine/dosdevices/com1
sudo ln -sf /dev/ttyACM0 ~/.wine/dosdevices/com2
unzip Minibloq.v0.81.Beta.zip
cd Minibloq.v0.81.Beta
wine MinibloqRun.exe
```

En otras plataformas similares, como la versión de Fedora que se incluye en algunas de las computadoras del programa OLPC (One Laptop Per Child), los pasos son los siguientes:

```
sudo su
yum install wine
winecfg
sudo usermod -aG dialout <myuser>
sudo ln -sf /dev/ttyUSB0 ~/.wine/dosdevices/com1
sudo ln -sf /dev/ttyACM0 ~/.wine/dosdevices/com2
```

Luego Minibloq debe ser descomprimido, lo cual se puede realizar en la misma interfaz gráfica con el **Archive Manager**.

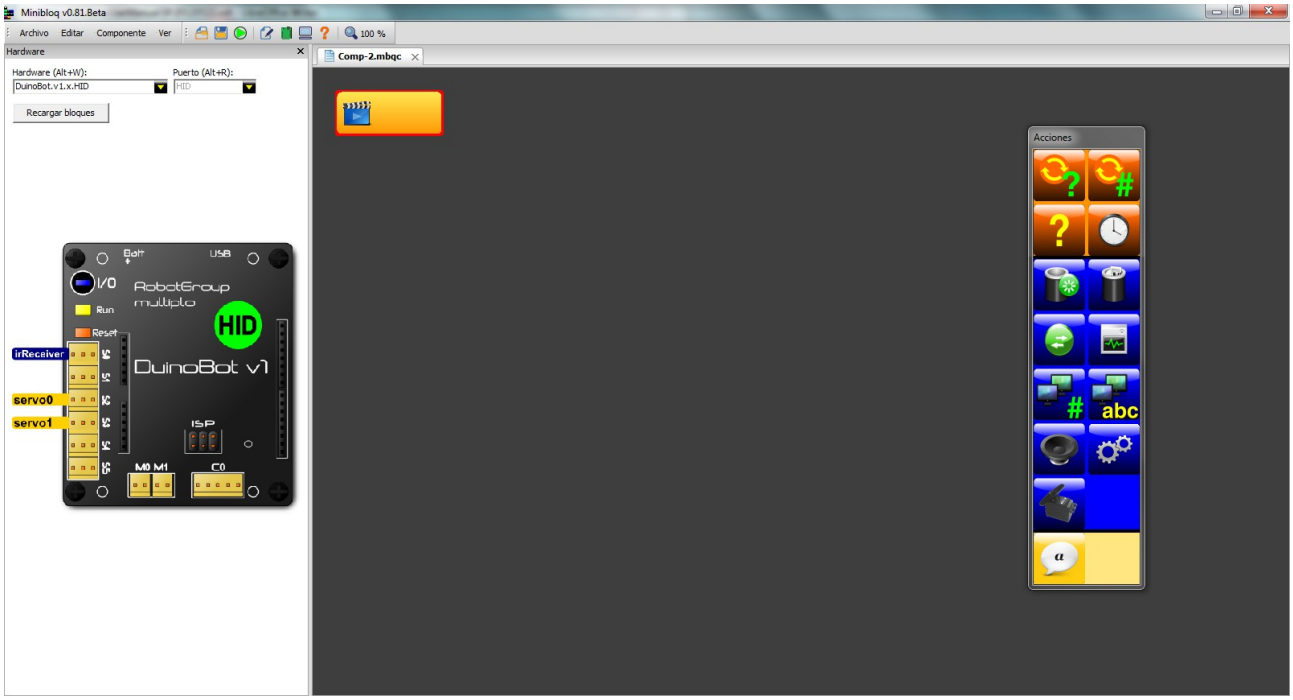
Finalmente debe hacerse click con el botón derecho sobre el archivo **MinibloqRun.exe** y debe seleccionarse la opción **Open With Wine Windows Program Loader** (del menú contextual que aparecerá).

# 3. Uso

## 3.1. Elementos de la interfaz de usuario

### 3.1.1. Zonas de la pantalla

A continuación se puede ver una captura de pantalla del entorno cuando recién inicia:




En los siguientes apartados se irán describiendo los diferentes elementos de la interfaz de usuario del programa.

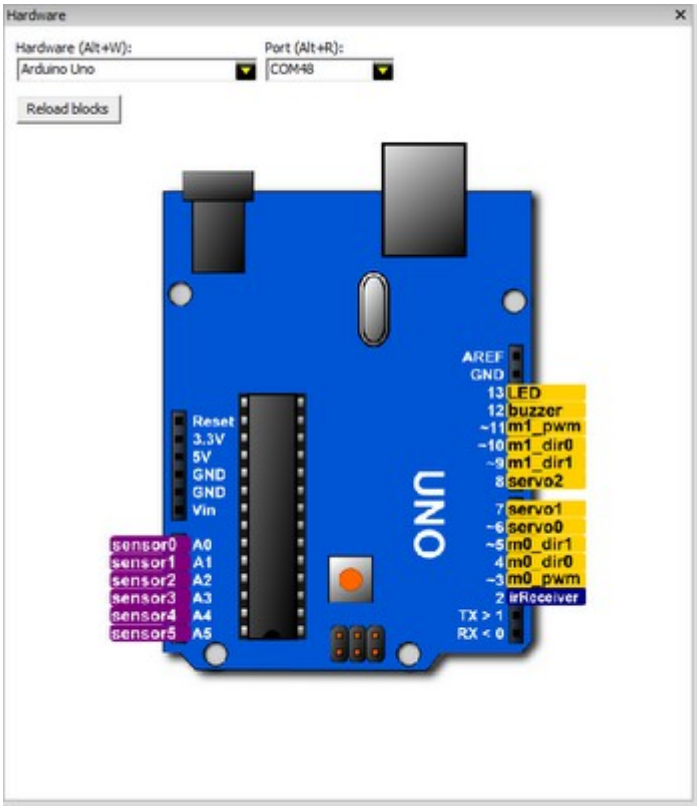
### 3.1.2. Menú y barra rápida de herramientas

Tanto el menú principal como la barra de herramientas rápida son en sí mismos ventanas acoplables. La barra de herramientas pueden incluso ocultarse si se lo desea, ya sea con el menú **Ver->Barra de herramientas** o con las teclas rápidas **Alt + Q**:


En Minibloq, se ha optado por un diseño minimalista, de modo que se han agregado a la barra rápida sólo las funciones más comunes, tales como grabar o ejecutar el programa que se está realizando, hacer zoom o visualizar las ventanas más comunes (código generado, configuración del hardware o la terminal). El resto de las funciones se pueden acceder por medio del menú o utilizando las teclas rápida (las cuales se describen más adelante en este documento).

### 3.1.3. Hardware

Esta ventana (también acoplable) permite al usuario seleccionar el tipo de placa o controlador que se desea programar (con lo cual Minibloq carga automáticamente los bloques que son compatibles con el hardware seleccionado). Es también en esta ventana donde el usuario selecciona el puerto de comunicaciones (que puede ser cualquier puerto serie tanto físico como virtual montado sobre un bus USB o sobre Bluetooth) para interactuar con el hardware.

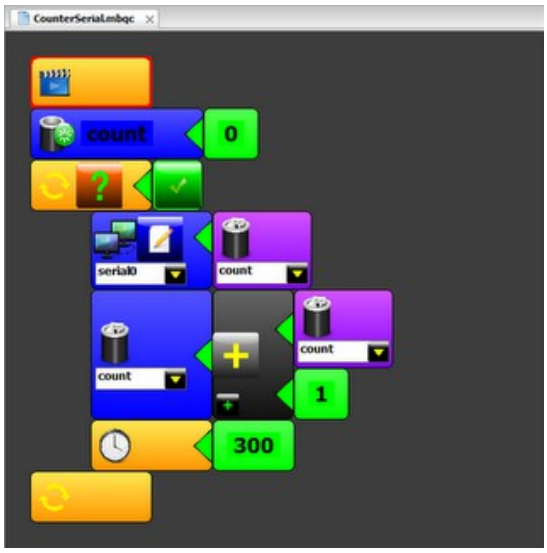



Adicionalmente, esta ventana presenta un dibujo conceptual del hardware seleccionado, el cual puede resultar muy útil a la hora de ver rápidamente, sin necesidad de consultar documentación extra, qué números de entradas y salidas (pines) accedidos desde el software, se corresponden con la placa física. También se han incorporado etiquetas para que el usuario sepa rápidamente dónde debe conectar ciertos dispositivos externos (tales como servos, o ciertos tipos de sensores). La ventana de hardware se muestra por defecto al iniciarse Minibloq, pero puede ser ocultada/visualizada con el menú **Ver->Hardware** o con la teclas rápidas **Alt + H**.

### 3.1.4. Editor de Componentes

Los programas realizados con Minibloq se llaman Componentes. El Editor de Componentes es por tanto la zona donde el usuario puede agregar los bloques que forman el Componente. En la misma es posible cortar y pegar bloques, desplazarlos, hacer zoom, etc.. El Editor de Componentes, tal como se verá más adelante soporta navegación por teclado para seleccionar el bloque actual y otras operaciones, así como también permite realizar ciertas operaciones (principalmente zoom y desplazamiento –o scroll-) utilizando la rueda del mouse y diferentes

combinaciones de teclas. Los bloques son además indentados automáticamente a medida que el usuario los va colocando.



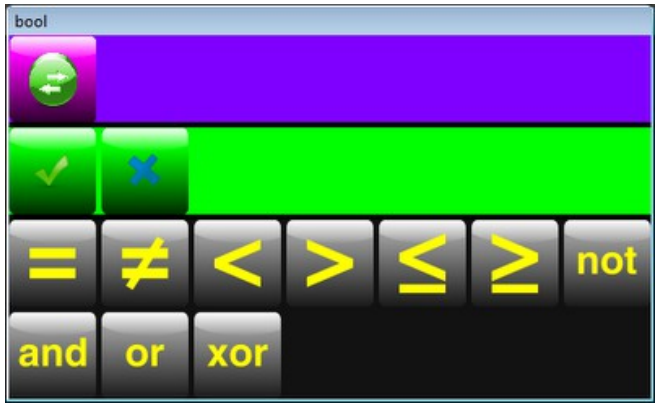

### 3.1.5. Selector de acciones (Actions picker)

El selector de acciones es la barra con bloques que permite ir agregando al programa (o componente) los bloques que representan comandos imperativos, o acciones (como por ejemplo la escritura de una salida digital, o de una salida analógica, o el establecimiento de la velocidad de un motor). Entre otras cosas, muestra pequeños "tooltips" de ayuda cuando el usuario se posiciona sobre un bloque por cierto corto intervalo de tiempo con el cursor, y sus botones cambian de color al moverse sobre ellos con el mouse. Esta ventana es flotante, y está siempre visible. Cada nuevo bloque se insertará con un solo click de mouse, y se hará a continuación del "bloque actual", que es el seleccionado con un pequeño rectángulo rojo en el Editor de componentes.

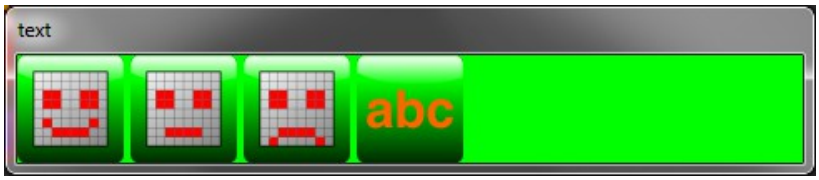
### 3.1.6. Selectores contextuales (Contextual pickers)

Una de las características más importantes de Minibloq es que presenta los bloques de forma *contextual*. Esto significa que cuando el usuario va a agregar un bloque como parámetro de otro, Minibloq sólo le mostrará aquellos del mismo tipo de datos. De esta forma, no sólo la interfaz de usuario es más limpia y clara, sino que se reducen las posibilidades de errores y el tiempo de búsqueda de cada bloque, ya que el usuario no debe seleccionar el bloque entre **todos** los bloques, sino sólo entre los que podría colocar. A continuación se muestra el selector contextual para bloques que devuelven números (tipo de datos *number*, en Minibloq):

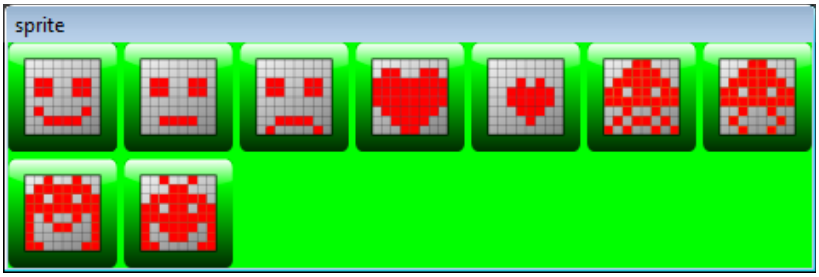

Y el siguiente es el selector de bloques que devuelven un valor booleano (tipo de datos *bool*):



Para poder enviar texto por el puerto serie, la versión v0.81.Beta dispone del selector de constantes de texto. Además de texto en sí mismo, es posible seleccionar "emoticones", los cuales son mostrados gráficamente por la terminal de Minibloq.v0.81.Beta:



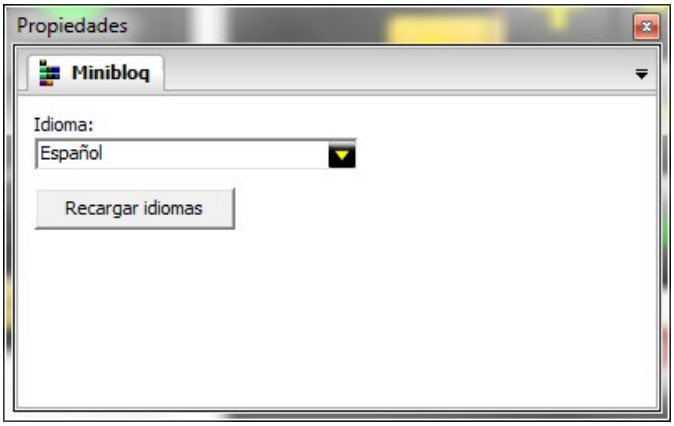

Por último, en placas con display de matriz de puntos, tales como DuinoBot.Kids, existe también un selector para los llamados *sprites* gráficos:




Algunas características importantes de los selectores (tanto del de acciones como de los contextuales), es que dividen automáticamente por color los bloques de acuerdo a la función que cumplen. Por ejemplo en el de bloques numéricos, la línea violeta es la de los "getters" (bloques que devuelven un valor de un método de una instancia, o de funciones del sistema), la línea verde es de constantes, la gris oscura de operaciones, y la gris clara de sensores específicos. Esta división el sistema la hace en *run-time*, durante la carga inicial de los archivos XML que describen a los bloques.

### 3.1.7. Ventana de propiedades

La ventana de propiedades permite al usuario configurar Minibloq. En la versión actual, sólo muestra el lenguaje seleccionado para la interfaz de usuario (y lista todos los paquetes de lenguajes instalados). Pero en el futuro, permitirá configurar muchos otros aspectos de Minibloq:

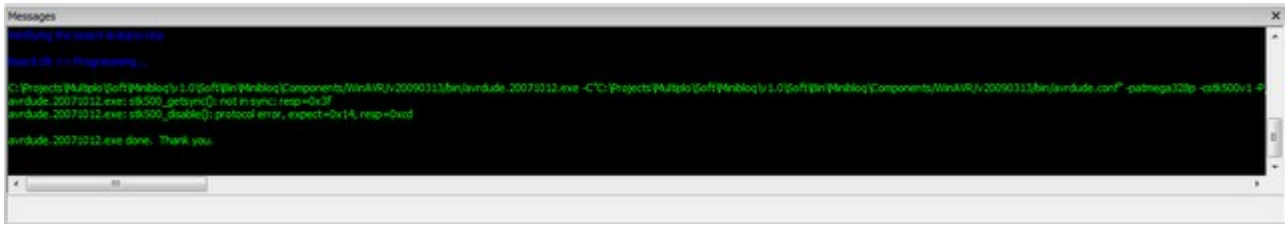


La ventana de propiedades es acoplable, y se la puede mostrar u ocultar con las teclas rápidas **Alt + M** o por medio del menú **Ver->Propiedades**. Es importante notar que Minibloq posee la capacidad de detectar el lenguaje del sistema operativo automáticamente. Pero es también posible para el usuario seleccionar otro lenguaje manualmente. Una característica interesante es que no es necesario reiniciar la aplicación cuando se cambia de lenguaje.

### 3.1.8. Ventana de mensajes

La ventana de mensajes tiene 2 zonas: Una de texto para los mensajes en sí, y una barra de progreso, para indicar el avance de tareas como la carga de bloques al seleccionar otra placa controladora, etc.. Esta ventana muestra el texto en diferentes colores:

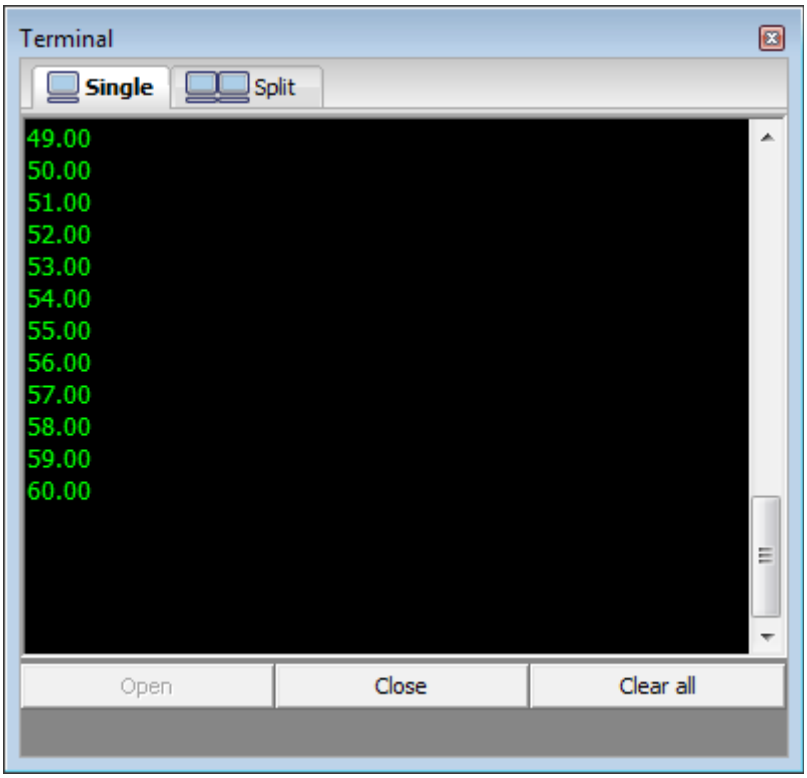
- Los comandos generados o ejecutados por Minibloq, tales como una llamada a un compilador se muestran en verde.
- La salida devuelta por herramientas externas llamadas por Minibloq se presenta en azul cuando fue exitosa.
- Si hubo errores en la salida producida por herramientas externas, el texto es rojo.




La ventana de mensajes además cuenta con zoom (utilizando la rueda del mouse y la tecla **Ctrl**). Para ocultar o visualizar la ventana de mensajes se puede, o bien presionar **Alt + M**, o bien utilizar el menú **Ver->Mensajes**.

### 3.1.9. Terminal

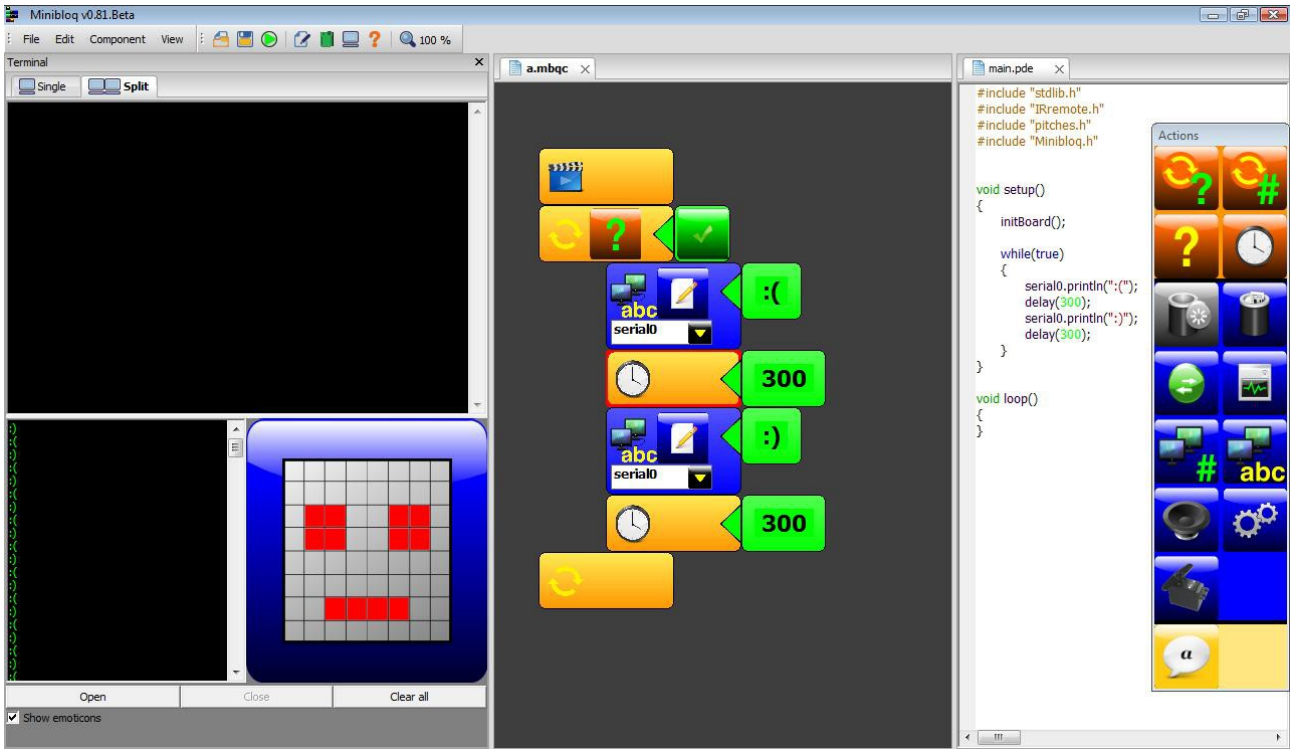
Es frecuente que los programas (componentes) envíen datos por el puerto USB (generalmente configurado como puerto serie virtual, o dispositivo USB CDC en las placas controladoras soportadas por Minibloq). Para visualizar estos datos en tiempo real, Minibloq cuenta con un terminal embebido:




El terminal puede ser visualizado ya sea utilizando las teclas **Alt + T**, o mediante el menú **Ver->Terminal**. La misma (como se puede ver en la figura) tiene botones para abrir y cerrar el puerto, lo que permite dejar el control del puerto a otras aplicaciones sin cerrar el terminal. Además, al bajar un nuevo programa a una placa, la terminal se desconecta automáticamente del dispositivo (cierra el puerto), de modo que su uso está completamente integrado al entorno, resultando así muy cómoda. Algunas características adicionales de la ventana de terminal son las siguientes:

- Muestra el texto recibido en verde y el enviado en azul.
- Posee zoom (rueda del mouse combinada con la tecla **Ctrl**).
- Presenta dos formas de visualizar los datos: Una donde el texto enviado y recibido están en la misma área de texto, y otra (*split*) donde están divididas (algunos usuarios pueden preferir una u otra, dependiendo también de lo que estén realizando).

Por último, queremos mencionar que el terminal de Minibloq soporta algunos "emoticones" gráficos. La idea de incluir esto surgió de la buena experiencia en aula que produjo el controlador DuinoBot.Kids, que cuenta con un pequeño display gráfico de matriz de LEDs. Al no contar con esto la mayoría de los controladores soportados por Minibloq, se decidió emular esta capacidad, al menos en lo que respecta a los emoticones ("sprites", como se los llama en el display). Aquí se puede ver la terminal mostrando estos gráficos. Para esto debe seleccionarse la solapa "Doble" (o "Split" en la versión en inglés del software):



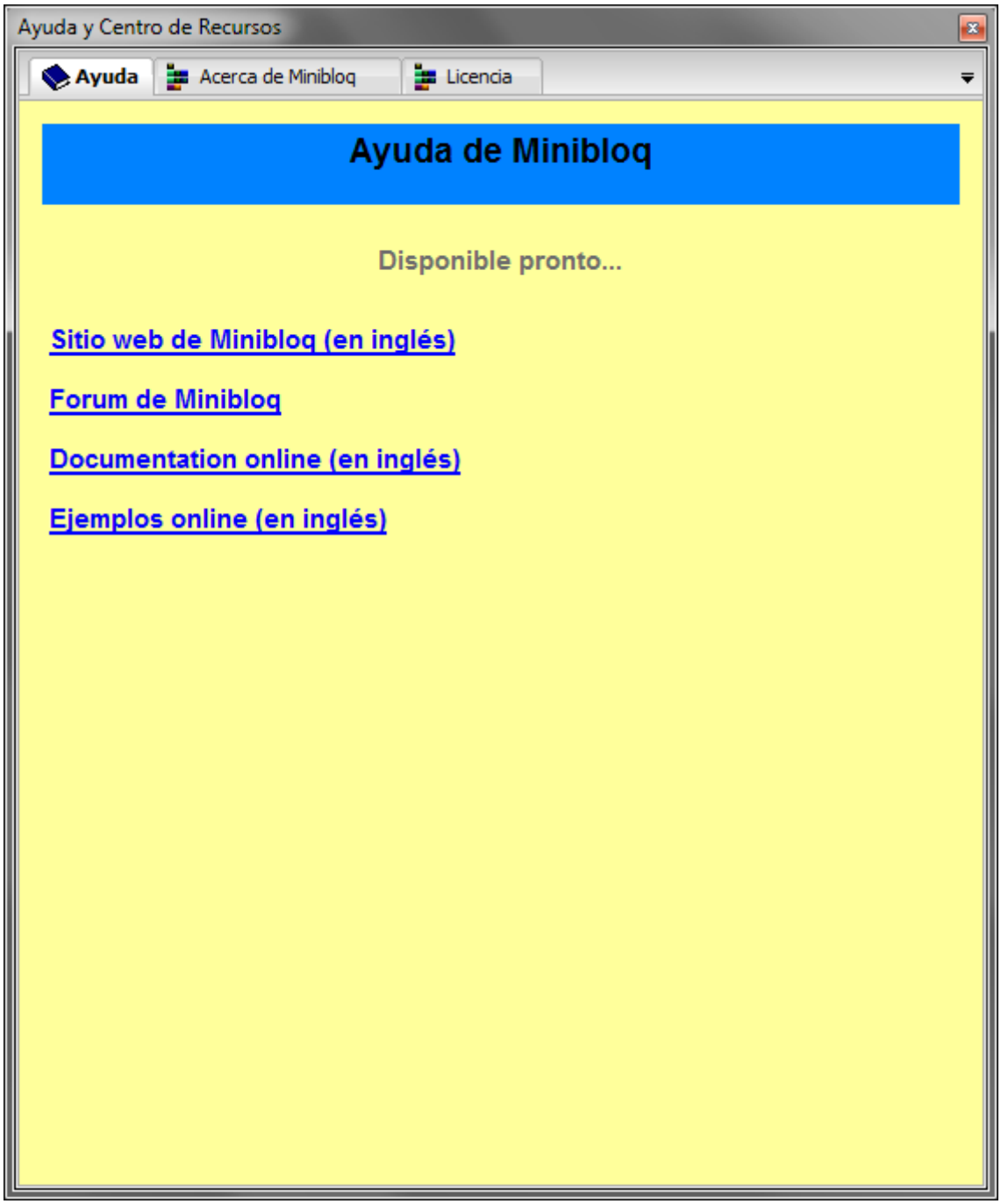

### 3.1.10. Visor del código generado

El usuario puede ver el código generado siempre que lo desee, en su editor de texto preferido, ya que todos los archivos de código creados automáticamente son de texto plano (Minibloq genera tanto los archivos de texto como los binarios -hex- y los coloca en un subdirectorio del directorio donde el usuario ha guardado el componente en el que está trabajando). Pero además, Minibloq brinda la posibilidad de visualizar todo el código generado en **tiempo real**. Ésta (la generación en tiempo real del código) es una característica única con respecto a otros entornos open source de programación visual para computación física (hasta donde sabemos al menos). La ventana del visor de código generado es también acoplable, y se la puede ver/ocultar con las teclas rápidas **Alt + G** o con el menú **Ver->Código generado**. A continuación se puede ver una captura de pantalla:


Como se puede apreciar en la imagen, la ventana cuenta con coloreo de sintaxis, y marca además con puntos verdes sobre el margen izquierdo el último código que fue modificado (coloca un punto verde por cada línea entre la primer línea modificada y la última). También es posible hacer zoom, dando foco a la ventana con un click de mouse y luego utilizando la rueda del mouse en combinación con la tecla **Ctrl**.

### 3.1.11. Ayuda

La ventana de ayuda no presenta en la versión actual mayor información que el link al sitio web de Minibloq (http://minibloq.org), los créditos (necesarios no sólo por la cantidad de gente que apoyó el proyecto sino también por las licencias open source utilizadas), y la licencia del software. Actualmente, la documentación más actualizada puede encontrarse en el link mencionado arriba, así como en el foro del proyecto: http://minibloq.net/forum. En futuras versiones del software (quizá más allá del alcance del proyecto) se irán incorporando más funciones a la ayuda. La siguiente es una captura de pantalla de la ventana de ayuda, la cual también es acoplable y a la que se puede visualizar con las teclas rápidas **Ctrl + H** o por medio del menú **Ver->Ayuda y Centro de Rercursos**:

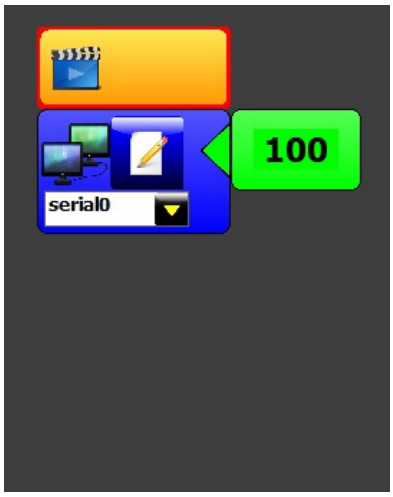



## 3.2. Verificación de errores en tiempo real

Muchas veces, la verificación de errores es problemática en los entornos de programación gráficos. Minibloq verifica errores en tiempo real: muestra en rojo aquellos elementos no completos a medida que el usuario va agregando bloques. A continuación se muestra este mecanismo para los diferentes casos que Minibloq detecta en la versión actual:

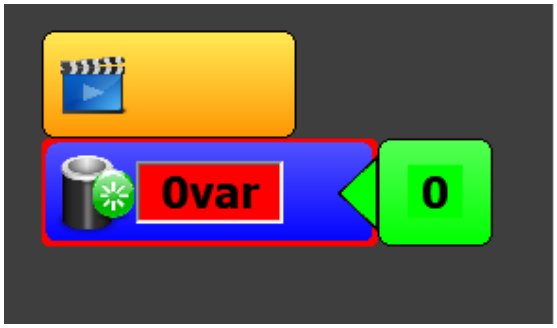
### 3.2.1. Parámetros incompletos

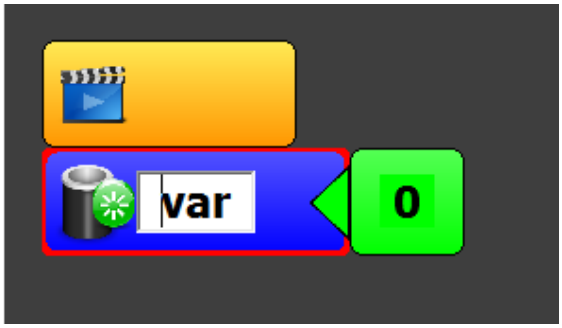
Cuando se agrega un bloque que requiere parámetros, el botón de agregado de parámetro (que tiene una flecha, indicando el sentido en el que fluirán los datos) queda en rojo hasta que dicho parámetro esté completo, como se puede ver en las siguientes imágenes (nótese el botón con la flecha en rojo en la imagen de la izquierda):


### 3.2.2. Nombres de variables no válidos

Minibloq valida los nombres de variables mientras éstos son escritos (tiempo real), de modo que si el nombre no es válido, la casilla del bloque de inicialización de variables queda en rojo:

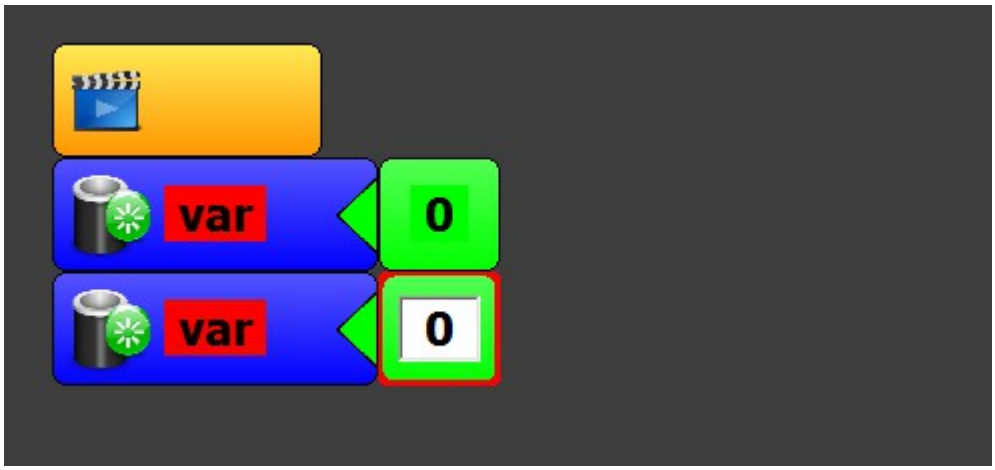



Además, los caracteres no válidos directamente no pueden ser ingresados al cuadro de textos.

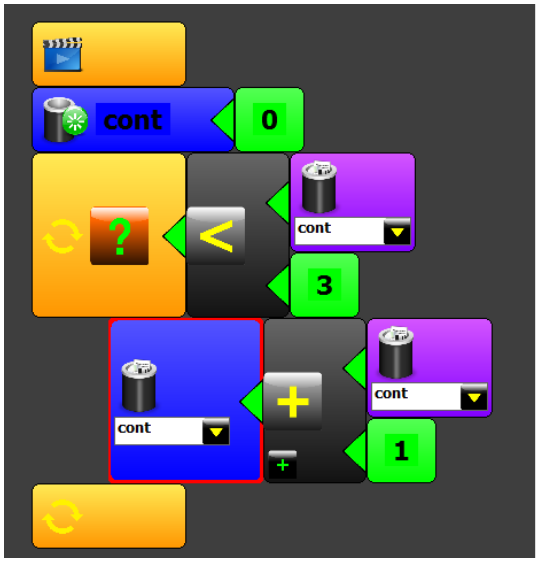
### 3.2.3. Nombres de variables repetidos

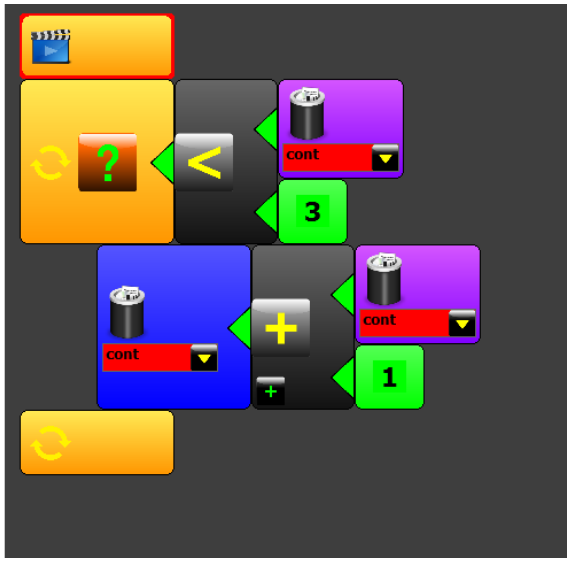
El entorno detecta automáticamente cuando se repiten nombres en las declaraciones de variables:




### 3.2.4. Nombres de variables eliminadas

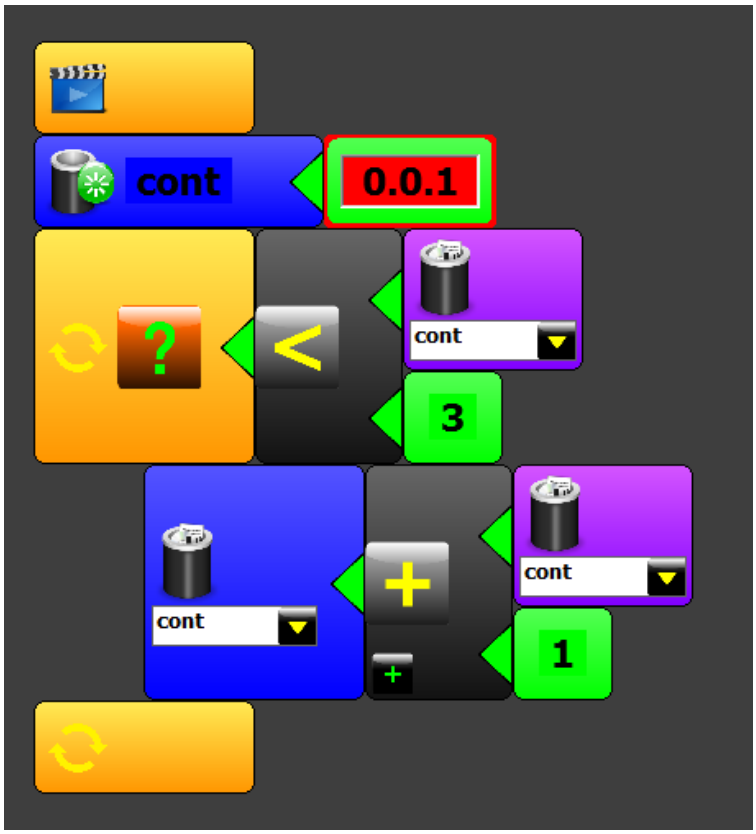
Si el usuario declara una variable, luego coloca bloques de asignación o de obtención del valor de la variable y finalmente borra la declaración, Minibloq coloca en rojo todas las ocurrencias de la los bloques de asignación y acceso. Esto se puede ver en la siguiente imagen:




### 3.2.5. Números con formato erróneo

Los caracteres no numéricos no pueden ser ingresados en las casillas de edición de constantes numéricas, pero aún así es posible que un usuario pretenda ingresar un número con dos separadores decimales (carácter ".") o errores similares. Minibloq también detecta estos problemas, indicándolos en pantalla con rojo:




### 3.2.6. Comentarios finales sobre los errores

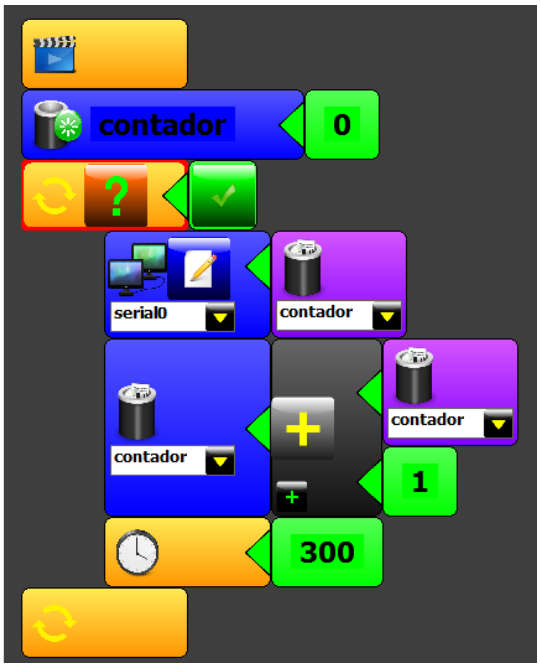
Alguien podría preguntarse si no son pocos casos de error los detectados, pero la realidad es que el lenguaje mismo evita que otros errores sean incluso posibles. Por ejemplo, los siguientes son errores muy comunes en la programación basada en texto cuando se programa en C/C++ (que es la sintaxis que genera la versión actual de Minibloq, aunque no está limitado a ella, por la que esto sólo es a modo de ejemplo):

- Falta de ";". Esto no puede ocurrir en Minibloq, porque el generador de código agrega los ";" cuando son necesarios de forma automática.
- Confusiones con tipos de operadores, por ejemplo el igual de asignación "=" y el de comparación "==". Minibloq coloca el operador correcto de acuerdo al contexto, cuando genera el código.
- Parámetros de tipos incorrectos. Esto tampoco es posible, debido a la forma contextual de trabajo descripta antes.
- Errores por letras mayúsculas o minúsculas. No ocurren, pues se programa con bloques.

Adicionalmente, en la ventana de mensajes se muestra la salida de la llamada al compilador, por lo que se indican allí también los errores que éste haya podido detectar en el código generado.

## 3.3. Manejo automático de nombres de variables

Otra característica que apunta a facilitar la programación cuando se utilizan variables, es el manejo automático de los nombres. El entorno detecta cuando se cambia el nombre de una declaración, renombrando de forma automática todas las ocurrencias de los bloques de acceso y asignación:


## 3.4. Uso del mouse y teclas rápidas

### 3.3.1. Uso del mouse

Muchas de los entornos de programación gráfica actuales utilizan alguna forma de arrastrar y soltar (*drag & drop)*. En Minibloq, es posible arrastrar los bloques de acciones para reorganizarlos. Incluso el entorno tiene autopaneo vertical cuando se realiza el arrastre. Pero la adición de bloques no se basa en arrastrarlos desde el selector, ya que basta con dar click sobre el bloque en cuestión en el selector de acciones, o sobre los selectores contextuales para agregar un nuevo bloque. Esta operatoria, a nuestro entender tiene las siguientes ventajas sobre el *drag & drop*:

- Reducción de las distancias recorridas con el cursor del mouse.
- Eliminación de la incomodidad asociada a mantener uno de los botones del mouse presionado durante el arrastre (la cual es mucho mayor en aquellas computadoras donde el principal dispositivo de manejo del cursor es un pad, como en netbooks).
- Agregado contextual de bloques (ya descripto antes con sus correspondientes ventajas).

Dentro de la zona de programación, aquellos mouses equipados con rueda (*mouse-wheel*) agilizan además las operaciones de zoom y desplazamiento (*scroll*), tanto vertical como horizontal. A continuación se describe el uso del mouse en detalle:

- Rueda del mouse sóla: Desplazamiento vertical.

- Rueda del mouse + **Shift**: Desplazamiento horizontal.
- Rueda del mouse + **Ctrl**: Zoom (in / out, dependiendo del sentido de giro de la rueda).

Adicionalmente, en el Editor de componentes, el mouse puede utilizarse del siguiente modo:

- Click con botón izquierdo sobre un bloque: Selecciona dicho bloque como el bloque actual. El bloque actual es aquel tras el cual se van a insertar los bloques al seleccionarlos en el selector de acciones.
- Click con botón derecho sobre un bloque: Presenta el menú contextual del Editor de componentes.
- Presionando el tercer botón del mouse (la rueda en la mayoría de los botones) sobre el fondo del Editor de componentes, y arrastrando con el mouse, se pueden desplazar tanto vertical como horizontalmente todos los bloques en conjunto.

### 3.3.2. Teclas rápidas

El manejo por teclado muchas veces agiliza mucho la programación. Las siguientes son las teclas rápidas del Editor de componentes:

- **Alt + Delete**: Borra el bloque actual.
- **Alt + Arrow keys (flechas)**: Se mueve hacia los bloques adyacentes.
- **Alt + Home (Inicio)**: Va hasta el primer bloque en la misma línea.
- **Alt + End (Fin)**: Va hasta el último bloque en la misma línea.
- **Ctrl + Home (Inicio)**: Va hasta el primer bloque.
- **Ctrl + End (Fin)**: Va hasta el último bloque.
- **Ctrl + Arrow keys (flechas)**: Desplazamiento.
- **Ctrl + Alt + Home (Inicio)**: Desplaza el programa a su posición inicial.
- **Ctrl + "+" / Ctrl + "-"**: Zoom in / Zoom out.
- **Ctrl + 0 (número cero)**: Restaura el zoom al100% (como en algunos navegadores web).

Las siguientes combinaciones son las teclas de acceso rápido de los menús:

- **Alt + F**: Despliega el menú Archivo.
- **Alt + E**: Despliega el menú Editar.
- **Alt + C**: Despliega el menú Componente.
- **Alt + V**: Despliega el menú Ver.

- **Ctrl + O**: Archivo->Abrir.
- **Ctrl + S**: Archivo->Guardar.
- **Alt + F4**: Archivo->Salir.

- **Ctrl + X**: Editar->Cortar.
- **Ctrl + V**: Editar->Pegar.
- **Alt + Del**: Editar->Eliminar.

- **Ctrl + U**: Componente->Ejecutar.
- **Ctrl + R**: Componente->Construir.

- **Alt + D**: Componente->Abrir la carpeta.

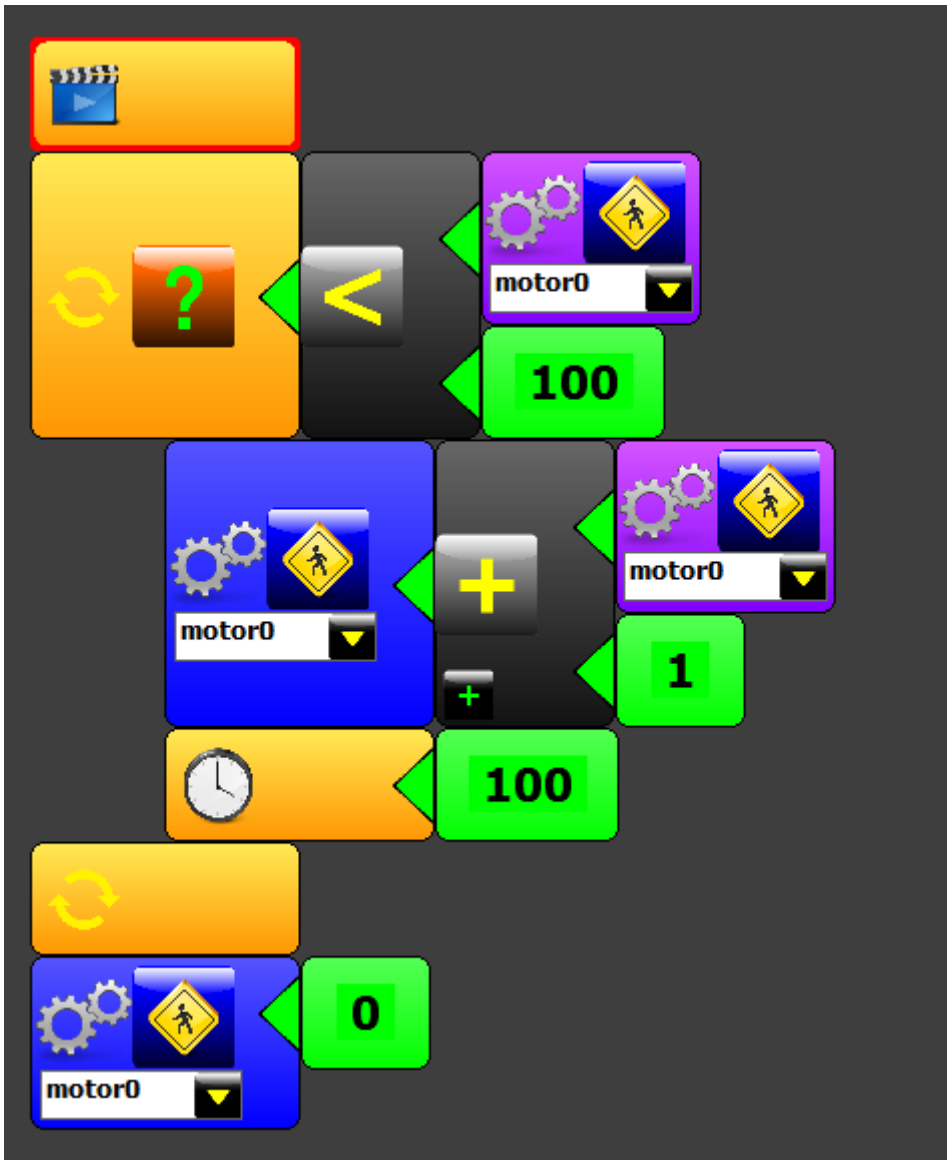
- **Ctrl + H**: Ver->Ayudar y Centro de Recursos.
- **Alt + Q**: Ver->Barra de herramientas.
- **Alt + G**: Ver->Código generado.
- **Alt + H**: Ver->Hardware.
- **Alt + M**: Ver->Propiedades.
- **Alt + M**: Ver->Mensajes.
- **Alt + T**: Ver->Terminal.

# 4. Programación con Minibloq

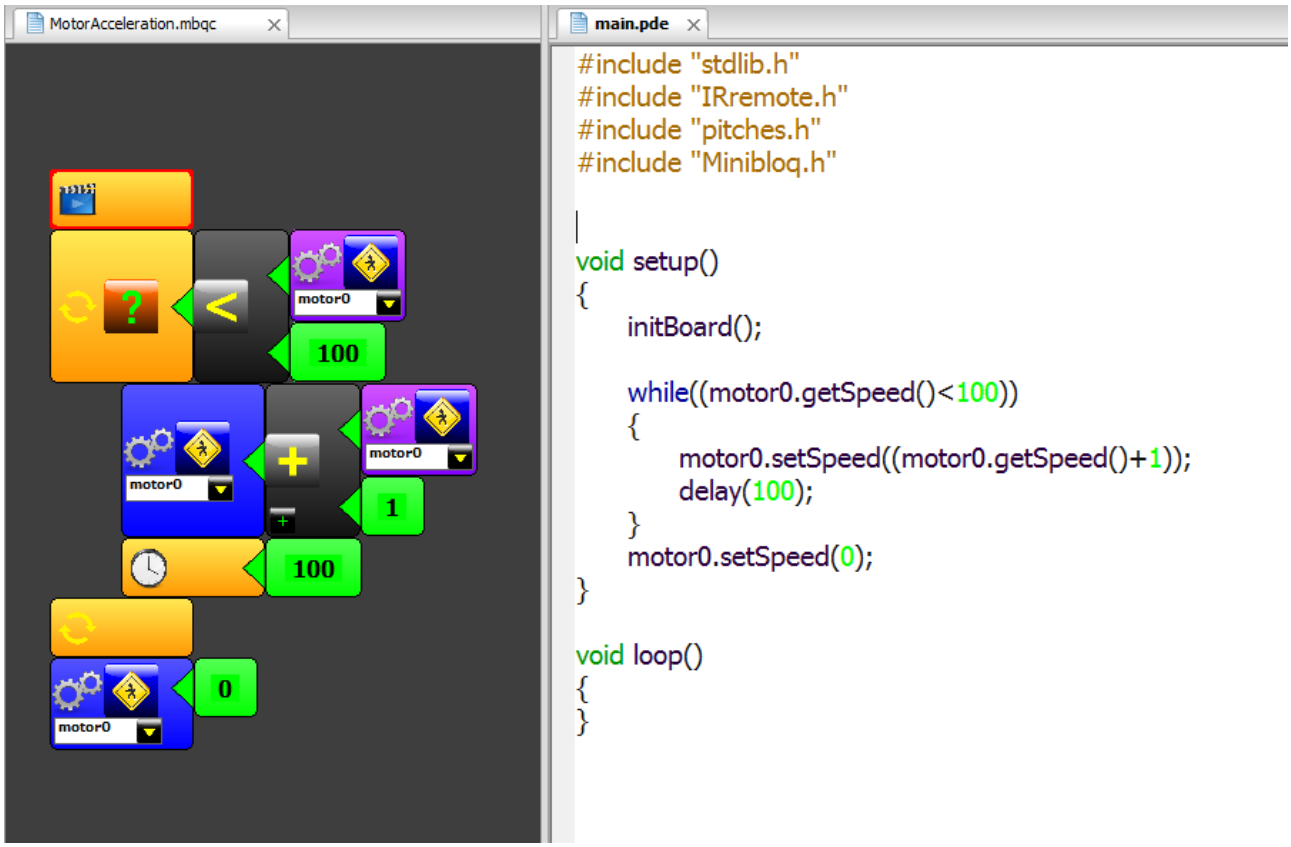
## 4.1. Conceptos básicos

### 4.1.1. Flujos de datos y de ejecución

En Minibloq, el flujo de ejecución es vertical, y los datos fluyen horizontalmente. Si bien esto no es así en todos los entornos gráficos, hemos optado por esta forma ya que es afín a la forma en que ejecución y datos fluyen en la gran mayoría de los lenguajes imperativos (paradigma para el cual Minibloq es en general capaz de generar código):




De manera conceptual, las flechas de los parámetros simbolizan el flujo de los datos. Debido a esta forma de funcionamiento, cuando se visualiza el código generado en tiempo real, la relación entre bloques y código se hace más clara:




### 4.1.2. Normalización de valores en dispositivos analógicos

En Minibloq todos los valores relacionados con periféricos con cierto tipo de entrada/salida analógica están normalizados entre 0 y 100. Por ejemplo, en el caso del bloque PWM, el parámetro de ciclo de trabajo (o *duty cycle*), éste puede ir de 0.0 a 100.0. En el bloque de control de motores, la velocidad se puede establecer entre -100.0 y +100.0 (float), siendo el signo el que determina el sentido de giro del motor. Los sensores analógicos, por su parte, devuelven también valores que van de 0.0 a 100.0, siempre en punto flotante. Esto se ha hecho así por varias razones. Algunas de ellas son:

- Mayor independencia del hardware: El usuario no se vé obligado a lidiar con cuestiones tales como si el conversor analógico del microcontrolador está configurado en 8 ó 10 bits. O si se migra a otra familia de microcontroladores luego, donde los ADCs son de 16 bits, no hay en general que portar nada, porque el código está escrito para floats de 0.0 a 100.0. En otras palabras, se trabaja en cierta forma con "porcentajes". De este modo, se gana independencia de las resoluciones de timers, PWM, conversores analógico-digitales, conversores digital-analógicos, etc..

- Se logra código más compacto cuando se trata de leer un sensor y realimentar directamente un motor o una salida de PWM, u otras aplicaciones similares (en los ejemplos que se incluyen con Minibloq, esto se puede ver más claramente). Evita por ejemplo, el uso de la función ***map***, porque ya está todo, tanto entradas como salidas, normalizado.

- Hace al código más claro para niños y principiantes. Por ejemplo, creemos que el "0 a 100" resulta bastante más natural que el "0 a 255" en PWMs de 8 bits, o el "0 a 1023" en sensores montados sobre ADCs de 10 bits (números que además cambian si se migra a una placa tipo Maple, por ejemplo).

Por otra parte, la principal contra que presenta esto es que el código es menos óptimo en tiempo de ejecución y en uso de memoria de programa, pero hemos visto código de este tipo (con otros entornos de programación) corriendo incluso sobre intérpretes y en microcontroladores mucho más chicos y antiguos que los AVR, y funciona muy bien de todos modos. Así que en un AVR o un ARM (o cualquier otro procesador moderno) con código nativo, salvo en aplicaciones extremadamente críticas (que difícilmente sean programadas utilizando Minibloq), esto no presenta diferencias realmente perceptibles de ejecución. Y como Minibloq soporta diferentes arquitecturas (y esperamos que soporte más aún en el futuro) esta forma resulta mucho más portable.

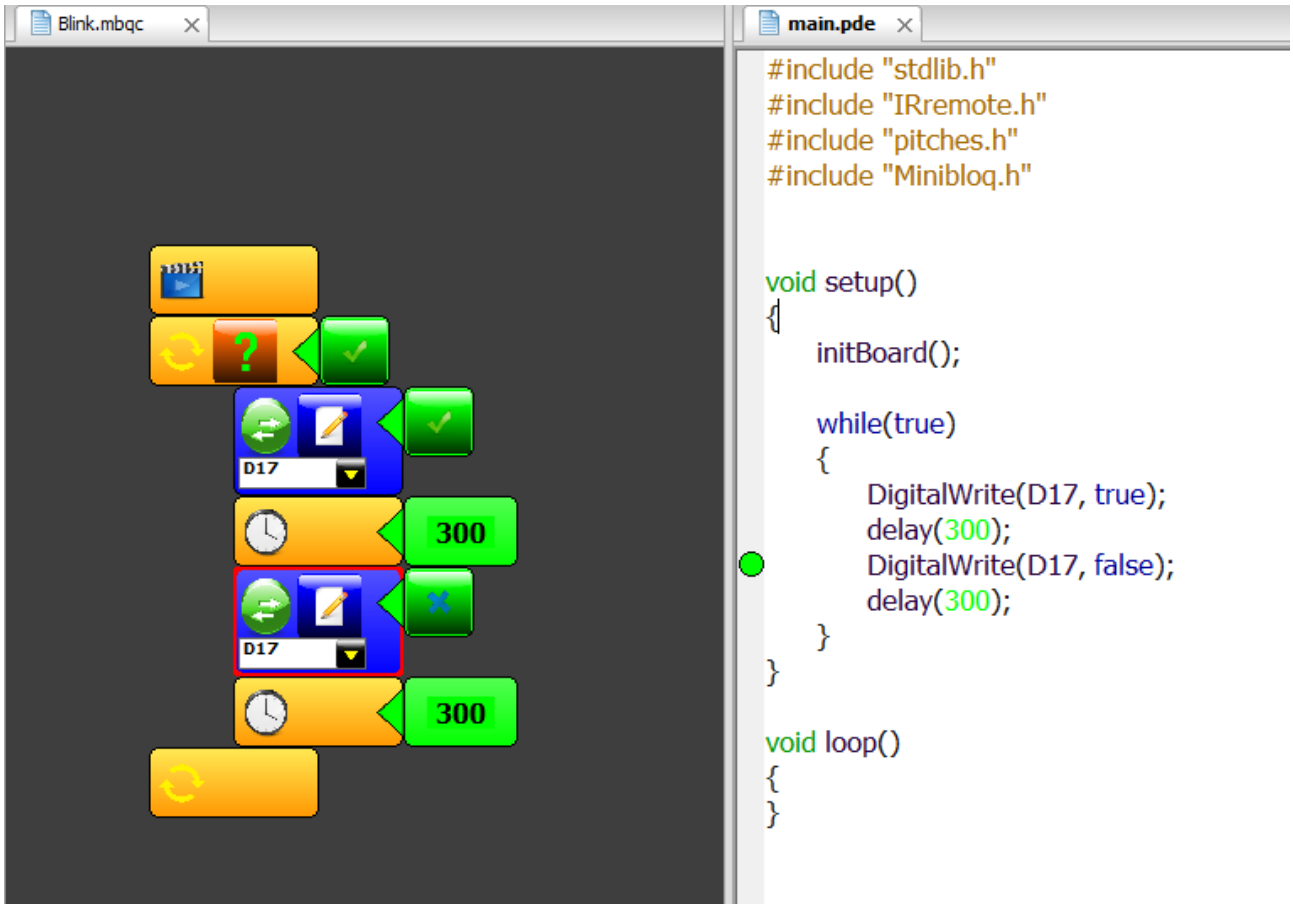
### 4.1.3. Generación de código en setup

La versión actual de Minibloq genera código C/C++ para dispositivos embebidos (o "placas controladoras", o "dispositivos de computación física", como se prefiera) utilizando la API de Arduino (http://arduino.cc). Ésta es hoy por hoy la API de código abierto para computación física más utilizada del mundo, y no está limitada a placas Arduino, como se puede ver en el mismo Minibloq, pues es prácticamente la misma API la utilizada por Maple, y muchos otros controladores disponibles actualmente en el mercado. Los programas en Arduino, si bien son C/C++, ocultan la función main() del usuario, reemplazándola (por decirlo de alguna manera) por dos funciones: setup() y loop(). La función setup se ejecuta una vez al inicio del programa, mientras que loop se llama de forma periódica y constituye lo que se suele llamar el "ciclo principal de la aplicación". Es así como el mínimo programa Arduino válido, suele ser algo como esto:

```
void setup()
{
}

void loop()
{
}
```

Minibloq genera todo el código dentro de la función setup. Por ejemplo, el código típico generado por Minibloq es algo como esto:




Esta decisión de diseño responde a varias razones, entre las cuales podemos mencionar las siguientes:

- Ayuda a evitar que el usuario cometa un error común cuando realiza programas que deben ejecutarse una sóla vez (y no en forma repetitiva): dejar al procesador en un estado indeterminado. Colocando el código en setup si el usuario no desea un ciclo infinito para su aplicación, su programa simplemente se ejecutará una vez (que era el comportamiento deseado) y luego entrará al ciclo infinito dado por la función loop. Allí el procesador permanecerá ejecutando hasta que alguien lo apague, evitando así el estado indeterminado, que puede ser peligroso (dependiendo de qué hardware esté siendo controlado por el procesador).

- La forma en que la API de Arduino resuelve internamente el loop no nos parece satisfactoria. Esto no es suficiente motivo para no utilizar esta API, ya que es la más extendida en computación física open source, y ha demostrado flexibilidad y potencia sobradas en sus ya varios años de uso (se estiman unos 300.000 usuarios de Arduino hoy en el mundo, sin contar los usuarios de clones de Arduino). Además, hemos integrado nuestras librerías para robótica con dicha API. Pero la implementación del loop nos parece problemática. Por ejemplo: es posible declarar variables locales dentro de loop(), pero éstas serían "realocadas" en cada iteración, lo cual está muy lejos de ser una solución óptima. En implementaciones con la función main() estándar y un ciclo infinito, como un while(true), por ejemplo, este problema no ocurre. Así que en Minibloq, la función loop()

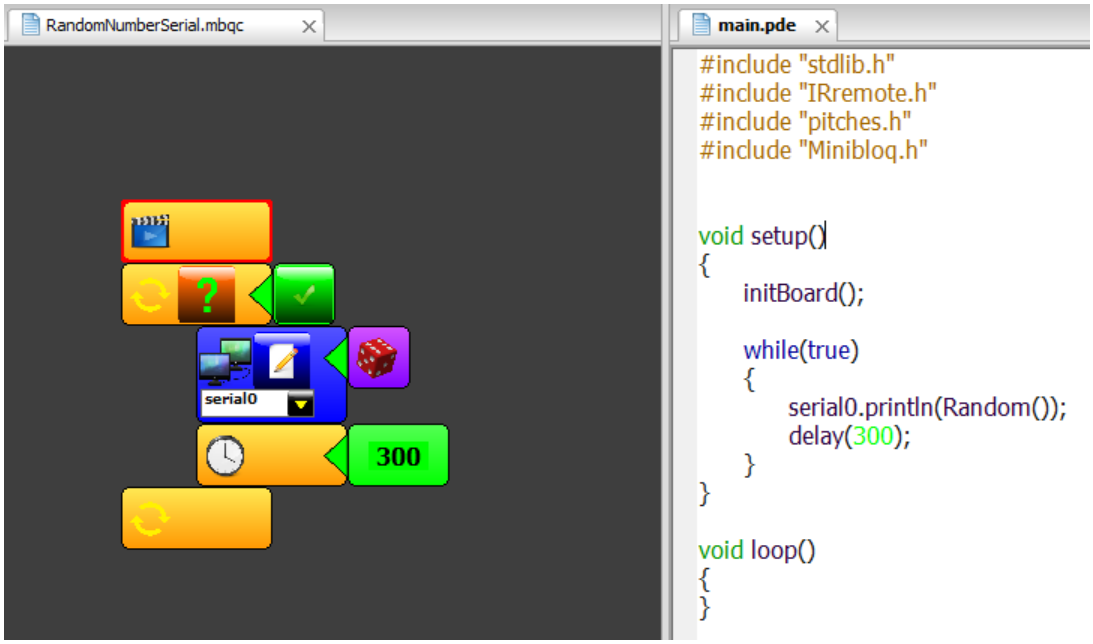
queda vacía, y se le da el uso descripto en el punto anterior (dejar al microncroncontrolador en un estado determinístico).

- Una característica importante de Minibloq (y que no muchos entornos de programación gráfica poseen) es que ni bien se inicia el entorno, el programa en pantalla (que consta sólo del bloque "inicio" o *start*) constituye un **programa válido**. Es así como el usuario podría presionar el botón **Run**, y el programa compilará y será enviado a la memoria de la placa que se encuentre conectada. Otros entornos que también generan sintaxis Arduino, requieren que el usuario agregue un loop manualmente. Ese bloque, que sólo tiene la función de generar el código para la función loop, aquí no es necesario, reduciéndose así la complejidad inicial, ya que Minibloq generará automáticamente el código para setup() y loop(), pero si el usuario quiere un ciclo infinito con acciones adentro, deberá agregar un while(true), lo cual ha constituido en general la práctica habitual en programación de sistemas embebidos en C/C++.

- Al no estar ligada la sintaxis Minibloq a la API Arduino exclusivamente, sino que la genera en esta forma donde podría tranquilamente reemplazarse al conjunto setup / loop por el main estándar con ciclo infinito, Minibloq resulta más independiente del hardware y específicamente más independiente de la API de Arduino. Es posible que en el futuro, Minibloq genere programas con otras sintaxis, no sólo otras APIs, sino incluso otros lenguajes imperativos (ya que en principio tiene potencia para realizar esto sin mayores problemas, gracias a su motor genérico de generación de código).

## 4.2. Acciones

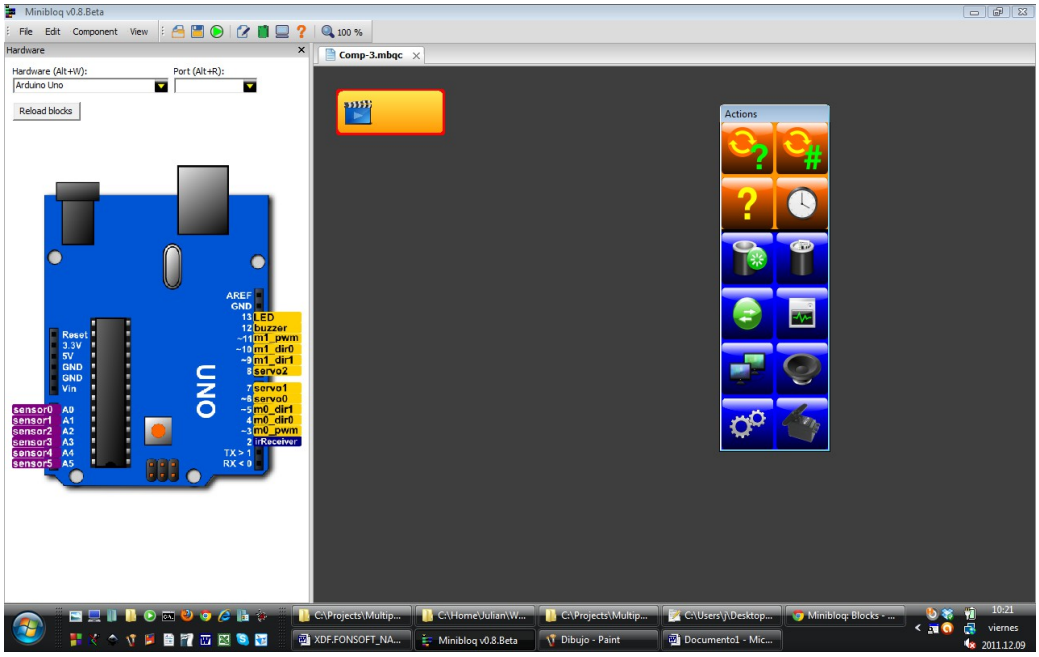
Minibloq tiene 2 tipos de bloques: Las acciones (de alguna manera equivalentes a las funciones y métodos que retornan *void* en C/C++, o a los *procedures* de Pascal; aunque en Minibloq las acciones también engloban declaraciones y asignaciones de variables, por ejemplo) y los que devuelven algún tipo de datos. Estos últimos se presentan en los selectores contextuales (contextual pickers) que aparecen en los parámetros de otros bloques. Varios de los bloques de acciones generan código orientado a objetos, llamando a *setters* de instancias. Por ejemplo, el bloque para enviar valores numéricos por la conexión serie/USB genera código relacionado con una instancia predeclarada (de la clase Serial):



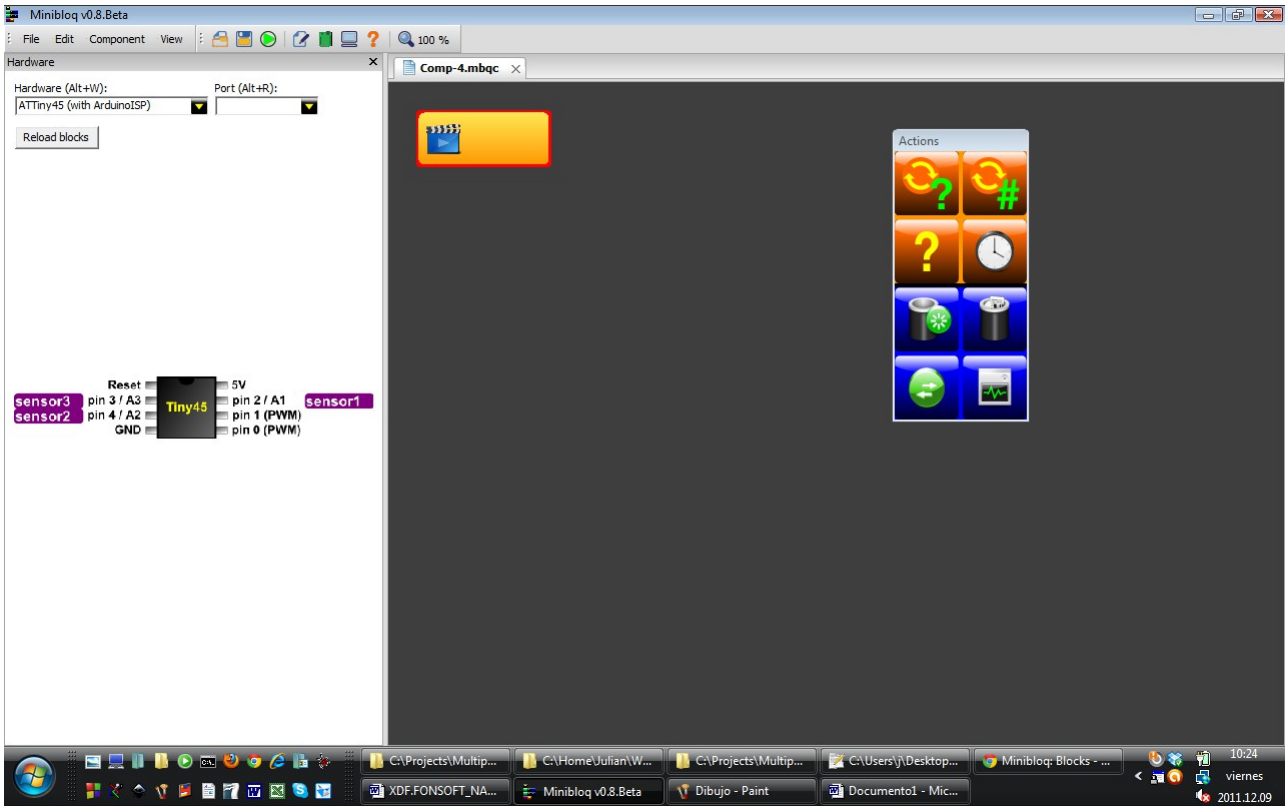

Las acciones, como todos los bloques en Minibloq, se cargan de acuerdo a las características del hardware que se ha seleccionado. Este mecanismo brinda mucha flexibilidad a la hora de agregar nuevo hardware al entorno, ya que éste último se adapta a las posibilidades de aquel. Las siguientes imágenes muestran claramente cómo varía el Selector de acciones en base a diferentes selecciones de hardware:

1. La placa DuinoBot.Kids dispone de un display de matriz de puntos, por lo que el entorno carga los bloques correspondientes (últimos 3 bloques del Selector de acciones):


2. La placa Arduino estándar tiene casi los mismos bloques que la placa anterior, pero sin el display:



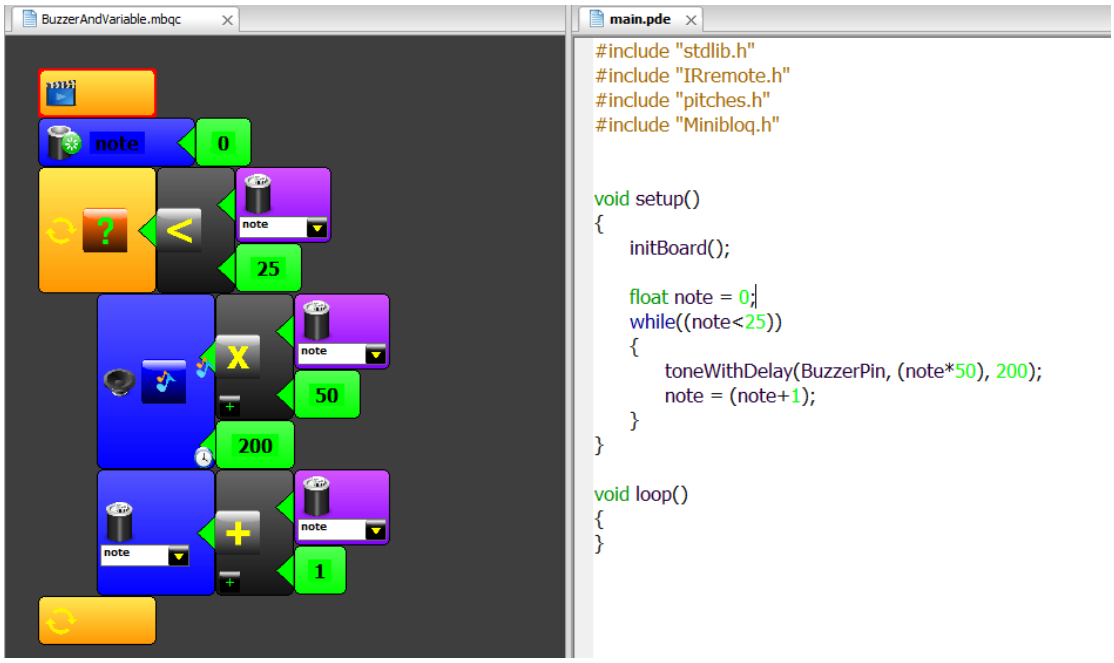

3. Los pequeños microcontroladores ATTiny tienen posibilidades mucho más limitadas, como se puede ver en el Selector de acciones:



## 4.3. Ciclos

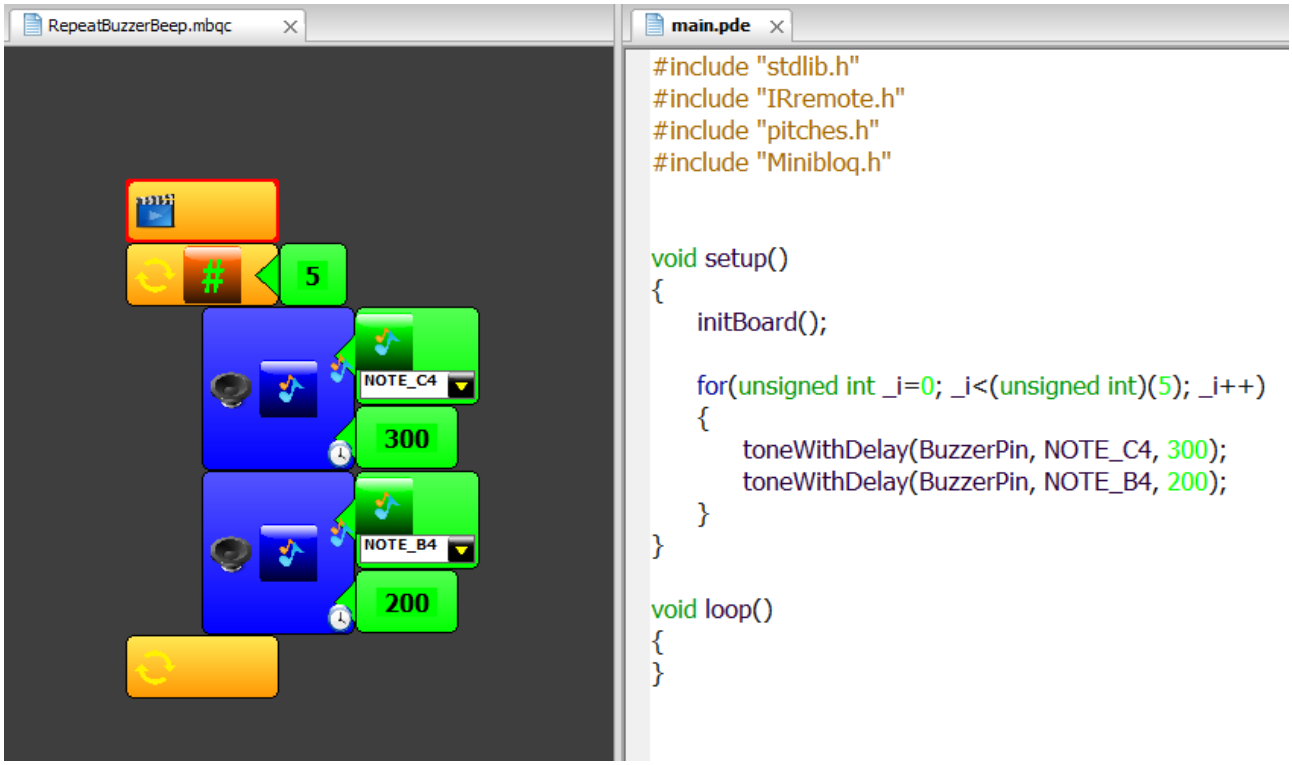
### 4.3.1. while

Él ciclo while de Minibloq es equivalente al *while* de C/C++, recibiendo como único parámetro una expresión booleana. El siguiente ejemplo incrementa una variable, que luego utiliza para generar una frecuencia en el buzzer, mientras dicha variable sea menor que 25:




### 4.3.1. repeat

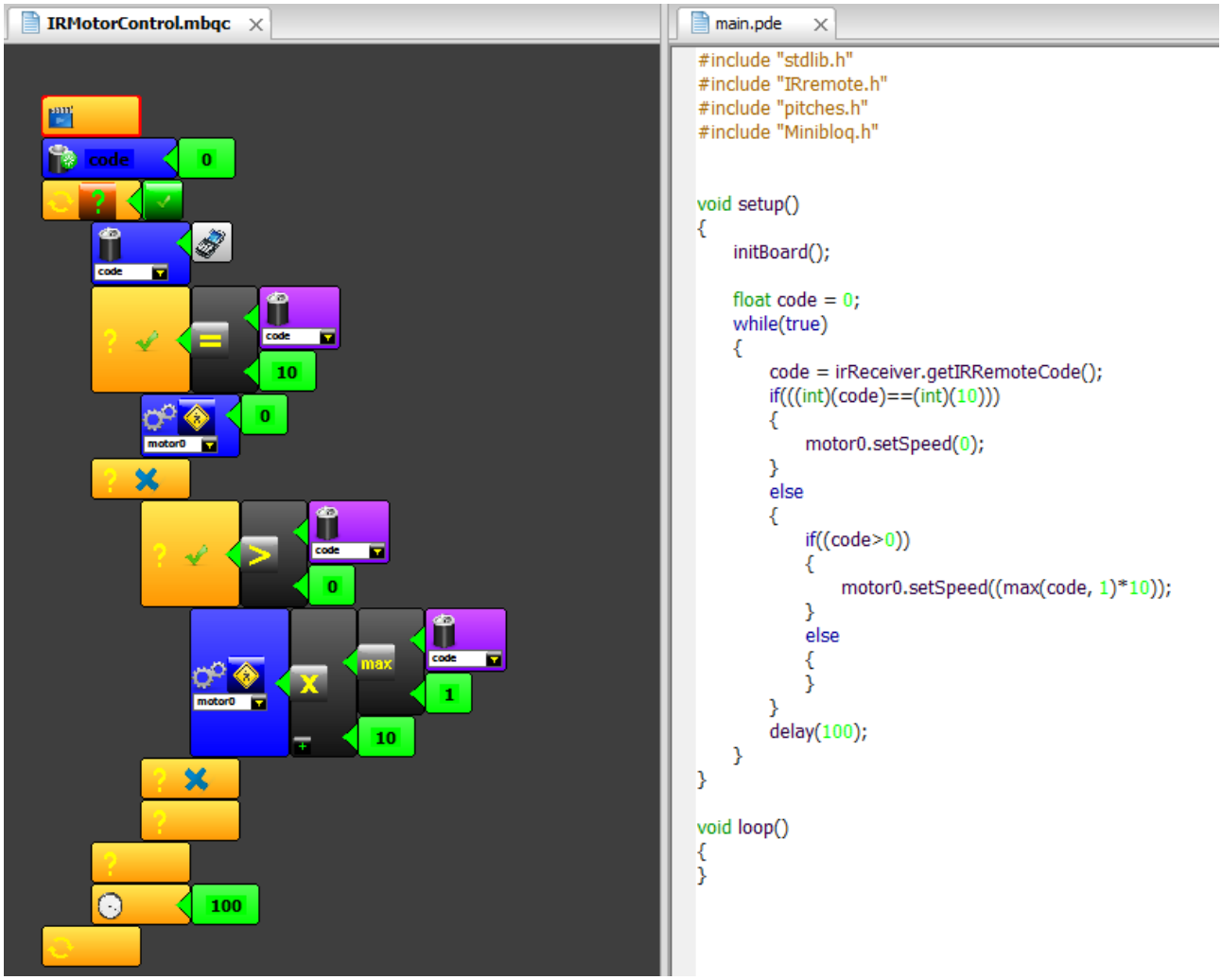
El ciclo repeat se asemeja al *for* de C/C++ (y de hecho genera un *for* en el código), pero no es directamente equivalente. Más bien se ha modelado sobre el repeat de lenguajes como Logo, donde el parámetro que se toma es directamente la cantidad de iteraciones a realizar. Este parámetro puede de todos modos no ser un simple número sino una expresión numérica más compleja, donde también podrían utilizarse variables que sean modificadas adentro del propio ciclo. Pero en general es utilizado para repeticiones sencillas, donde se conoce a priori cuántas iteraciones se harán. A continuación puede verse un ejemplo sencillo con el código que genera:




La variable _i del código es interna y el usuario no puede accederla en la versión actual de Minibloq, decisión de diseño tomada en base a la idea de mantener la sencillez del entorno, y al hecho de que el bloque repeat es independiente del lenguaje que se genera (como se dijo antes, Minibloq podría generar otras sintaxis más allá del C/C++ en el futuro). De esta forma, el repeat no es un ciclo *for*, sino precisamente un *repeat*. El *for* visto aquí en el código generado es simplemente la **implementación C/C++** de este concepto.

## 4.4. Decisiones (if)

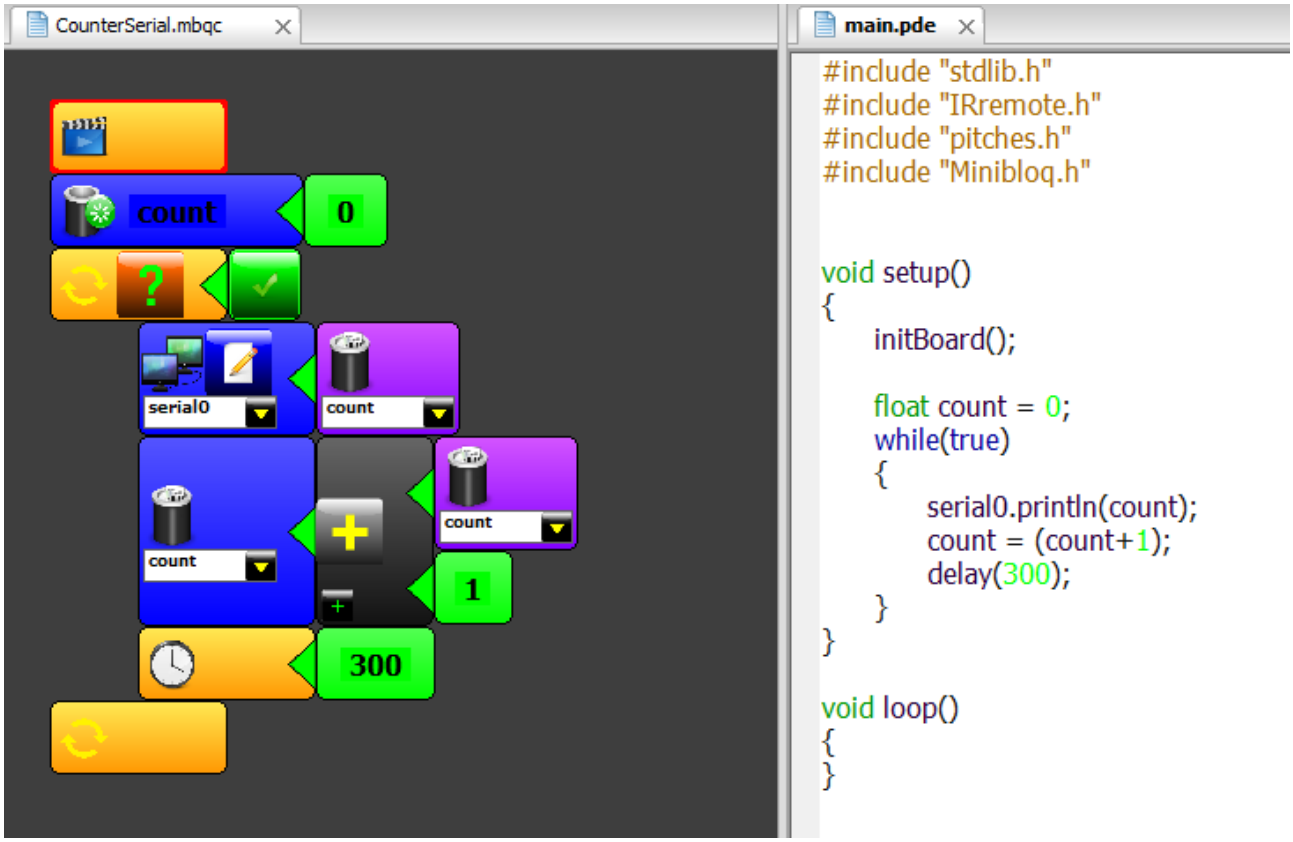
Los bloques de decisiones en base a condiciones boolenas (if) son equivalentes a las construcciones *if-else* de C/C++. La única salvedad es que generan siempre el *else* en la versión actual (de forma similar a cómo lo hacen otros entornos de programación gráfica). A continuación se puede ver el bloque en uso, junto con el código que genera:



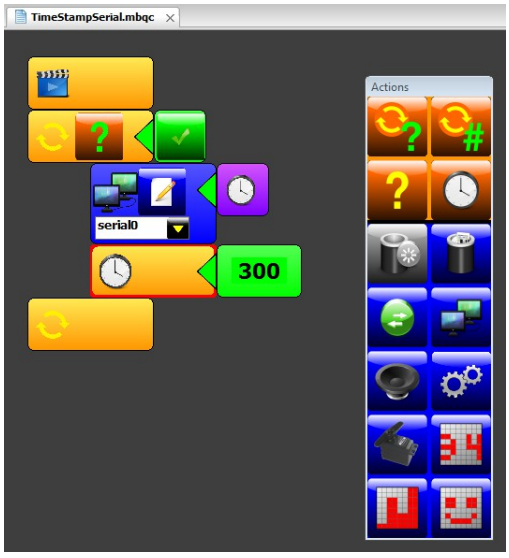

## 4.5. Variables

Minibloq posee en su versión actual variables numéricas. Se han tomado algunas decisiones de diseño con respecto al sistema de variables, para impulsar buenas prácticas de programación. Por ejemplo:

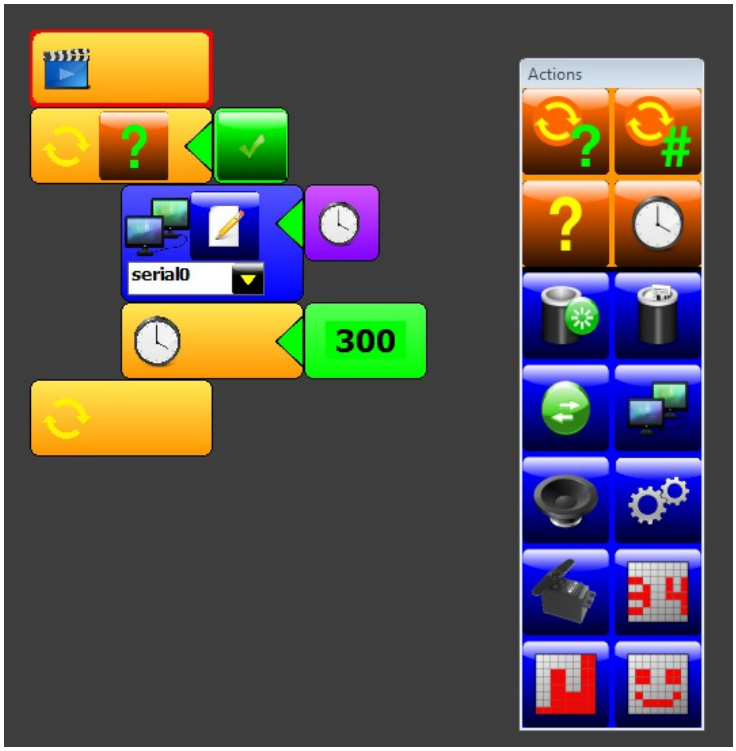
- Las variables requieren **declaración** e **inicialización**. Como se puede ver en la imagen a continuación, hay un bloque específico para declarar variables, y éste requiere inicialización obligatoria. Como ya se explicó antes, Minibloq tiene varias características para ayudar al usuario cuando renombra una variable, o si borra una declaración, renombrando automáticamente las demás ocurrencias de la variable en el programa, o notificando de errores (por borrado por ejemplo) en tiempo real. La siguiente imagen muestra un sencillo programa con una variable que funciona como contador. Allí se puede apreciar el bloque de declaración / inicialización (o *varInit*), y los bloques de asignación y acceso a la variable:



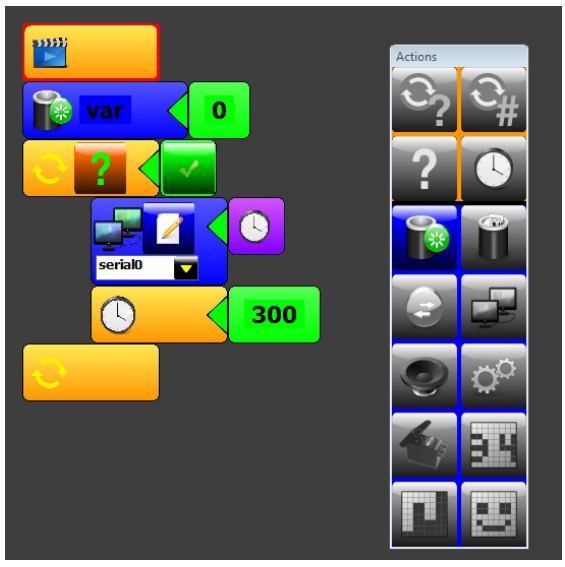

- Las declaraciones e inicializaciones van obligatoriamente en el principio del programa, tras el bloque de inicialización. El entorno habilita o deshabilita los bloques del selector de acciones también en relación al contexto del bloque seleccionado actualmente. Por ejemplo, si el usuario tiene seleccionado un bloque cualquiera del programa, que no sea el primer bloque, no podrá agregar un bloque de declaración/inicialización, puesto que las variables sólo pueden declararse al principio. Nótese en la siguiente imagen que el bloque de declaración/inicialización está deshabilitado (gris) en el Selector de acciones, ya que el bloque seleccionado (bloque con el borde rojo) en el Editor de componentes es un bloque intermedio (tras el cual no está permitido insertar declaraciones de variables):



En cambio, cuando el usuario selecciona el primer bloque, se habilita también el botón del bloque de Declaración/Inicialización de variables:

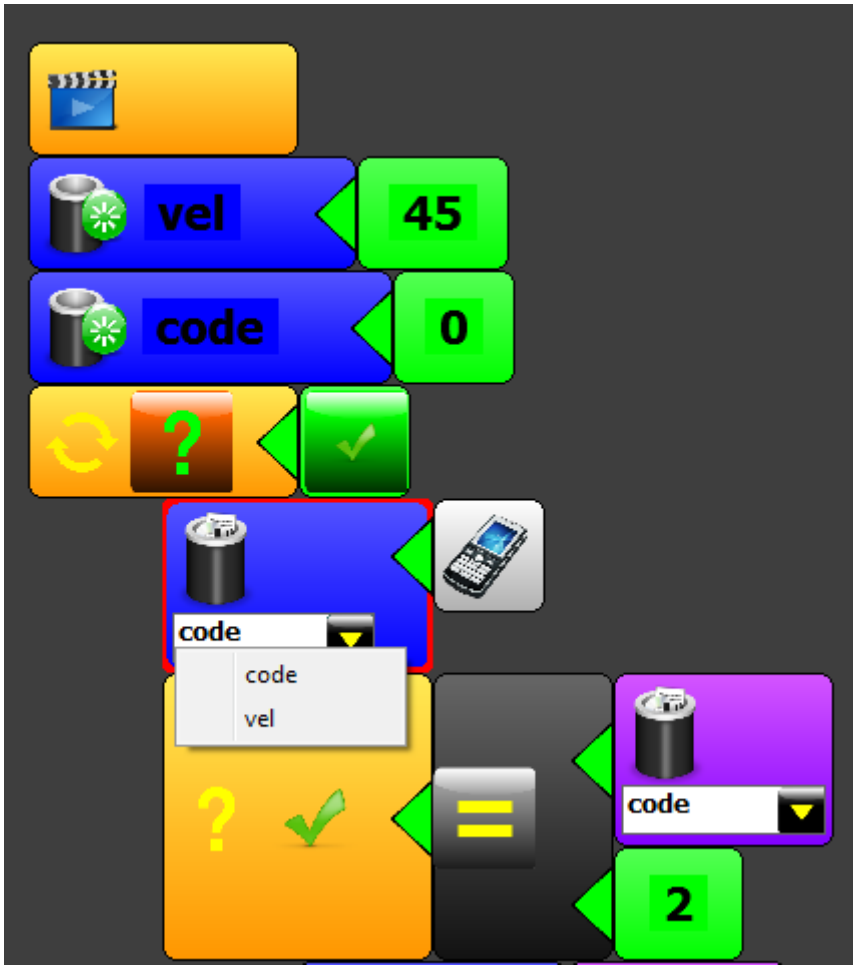



Ahora bien, si se agrega un bloque de Declaración/Inicialización, entre éste y el bloque de inicio, sólo podría agregarse otro bloque de Declaración/Inicialización. Si fuera de otro modo, el usuario podría declarar una variable y luego colocar bloques antes de la declaración, lo cual como se explicó antes, no se permite en Minibloq por no considerarse una buena práctica de programación. Esta situación también es detectada por el entorno, de modo que cuando hay bloques de declaración al principio, y la posición seleccionada de inserción es **antes** de uno de éstos, el Selector de acciones deshabilita automáticamente todos los bloques, excepto los de declaración:



- No hay variables globales. Si en el futuro se incorporan bloques de usuario (a modo de procedimientos y funciones), las variables permanecerán siendo locales. En la versión actual, las variables tienen alcance local dentro de la función setup() del código generado.

- Por último, cabe destacar que Minibloq lista automáticamente sólo los nombres de variables declaradas en los bloques de acceso y asignación, de modo que el usuario no puede ingresar a mano allí nombres inválidos, protegiéndolo de este modo de posibles errores. Nótese el menú contextual en la siguiente imagen, que sólo lista las variables "Code" y "Vel", declaradas antes:




## 4.6. Nota acerca de los bloques de sensores

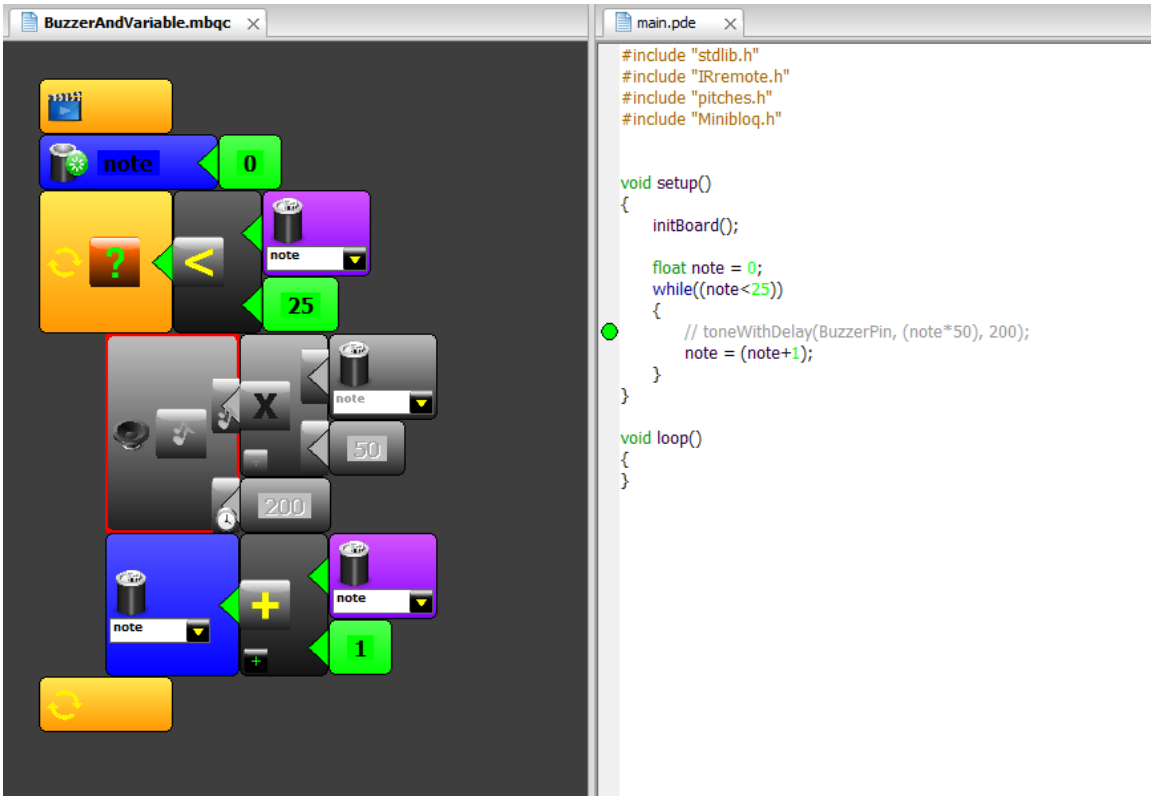
La mayoría de los sensores que se consiguen en el mercado pueden utilizarse con el bloque AnalogRead() o con el bloque de lectura digital (el *getter* IOPin):

Una de las ideas más importantes detrás del diseño de Minibloq es la simpleza. Minibloq tiene que ser tan simple como sea posible. Es por esta razón, que si bien el entorno soporta muchos tipos de sensores utilizanod los bloques mencionados, sólo se incluyen bloques específicos para aquellos sensores que requieren código especializado. Por ejemplo, los sensores de distancia (o *rangers*) Sharp, ciertos sensores de distancia ultrasónicos, o los receptores de control remoto de 38 KHz son algunos de ellos:

Esperamos seguir agregando (y contando también con agregados de la comunidad de usuarios) más bloques de sensores específicos, pero en general no habrá bloques de este tipo cuando un sensor pueda leerse por medio de los bloques estándar AnalogRead o IOPin. Esto además mantiene los selectores (pickers) más compactos y facilita así al usuario el poder encontrar los bloques que necesita más rápidamente.

## 4.7. Código comentado

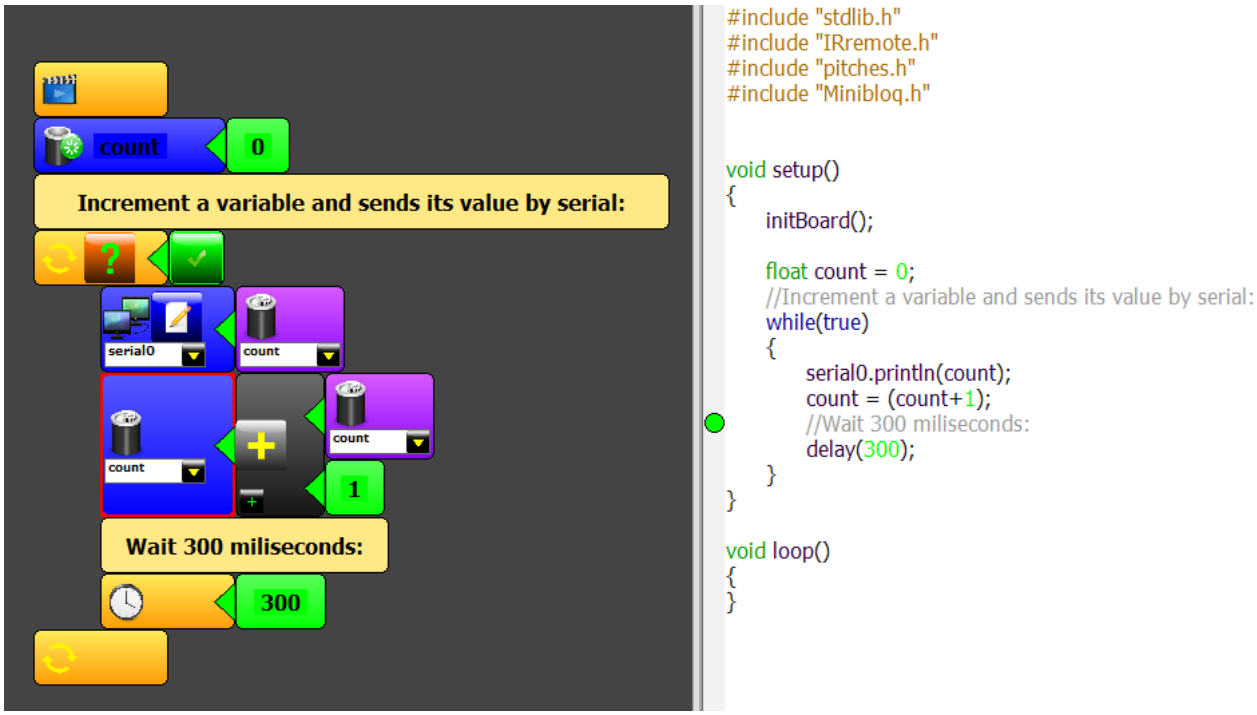
Si se quiere dejar sin efecto un bloque, es posible comentarlo, utilizando el menú contextual (con click derecho sobre el bloque en cuestión), o con el menú **Edit->Comment/Uncomment**. Al comentar un bloque, éste aparecerá en gris, y el código generado automáticamente será comentado también, como se puede ver en la siguiente imagen:



## 4.8. Comentarios

Además de la posibilidad de comentar el código, Minibloq permite también agregar comentarios arbitrarios, para documentar el programa. Ésto se realiza por medio del bloque de comentarios, en el selector de acciones:

Dicho bloque permitirá agregar comentarios de una sóla línea entre los bloques. Y como se puede ver en la siguiente imagen, Minibloq automáticamente agregará el comentario al código:




## 4.9. Ejemplos

Minibloq incluye numerosos ejemplos que se distribuyen con el entorno. Para encontrarlos fácilmente, se puede utilizar el menú **File->Examples**, el cual mostrará un cuadro de diálogo de apertura de archivos posicionado en el subdirectorio \Components_Workspaces_Examples de Minibloq. Allí hay a su vez 5 subdirectorios (en la versión actual de Minibloq), donde se pueden encontrar los ejemplos clasificados por el tipo de hardware sobre el que pueden utilizarse. Dichos subdirectorios son los siguientes:

- **Arduino_Seeed_DuinoBot_DuinoBot.Kids**: Contiene ejemplos para las siguientes placas:
    - DuinoBot.v1.x.HID
    - DuinoBot.v1.x
    - DuinoBot.Kids.v1.x
    - Seeeduino v2.2x Mega328
    - Seeeduino Mega 1280
    - Arduino Uno
    - Arduino Duemilanove Mega328
    - Arduino Duemilanove Mega168
    - Arduino Mega 2560
    - Arduino Mega 1280

- **Arduino_Seeed_DuinoBot_DuinoBot.Kids_Maple**:
    - DuinoBot.v1.x.HID
    - DuinoBot.v1.x
    - DuinoBot.Kids.v1.x
    - Seeeduino v2.2x Mega328
    - Seeeduino Mega 1280
    - Arduino Uno
    - Arduino Duemilanove Mega328
    - Arduino Duemilanove Mega168
    - Arduino Mega 2560
    - Arduino Mega 1280
    - Maple Rev 3+ (to Flash)
    - Maple Rev 3+ (to RAM)

- **Arduino_Seeed_DuinoBot_Maple**:
    - DuinoBot.v1.x.HID
    - DuinoBot.v1.x
    - Seeeduino v2.2x Mega328
    - Seeeduino Mega 1280
    - Arduino Uno
    - Arduino Duemilanove Mega328
    - Arduino Duemilanove Mega168
    - Arduino Mega 2560
    - Arduino Mega 1280
    - Maple Rev 3+ (to Flash)
    - Maple Rev 3+ (to RAM)

- **DuinoBot.Kids**:
    - DuinoBot.Kids.v1.x

- **Tiny**:

- ATTiny25 (with ArduinoISP)
- ATTiny45 (with ArduinoISP)
- ATTiny85 (with ArduinoISP)
- ATTiny25 (with Doper)
- ATTiny45 (with Doper)
- ATTiny85 (with Doper)

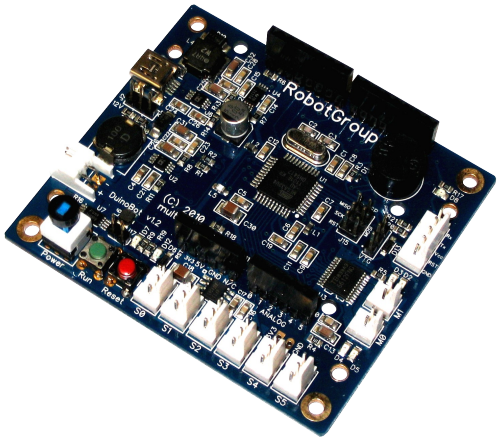
Por otro lado, de forma continua se están adicionando ejemplos al sitio de Minibloq, específicamente a la página http://blog.minibloq.org/p/tutorials-and-examples.html. Estos ejemplos on-line, tienen la ventaja adicional de que los hemos enriquecido con los esquemáticos (también creados con software open source, en este caso con Frizting http://fritzing.org), la lista de partes de hardware y en la mayoría, con al menos un video de la aplicación funcionando.

# 5. Equipamiento físico y Hardware soportado por Minibloq

En este capítulo se muestran las principales placas controladoras y microcontroladores soportados por la versión actual de Minibloq.

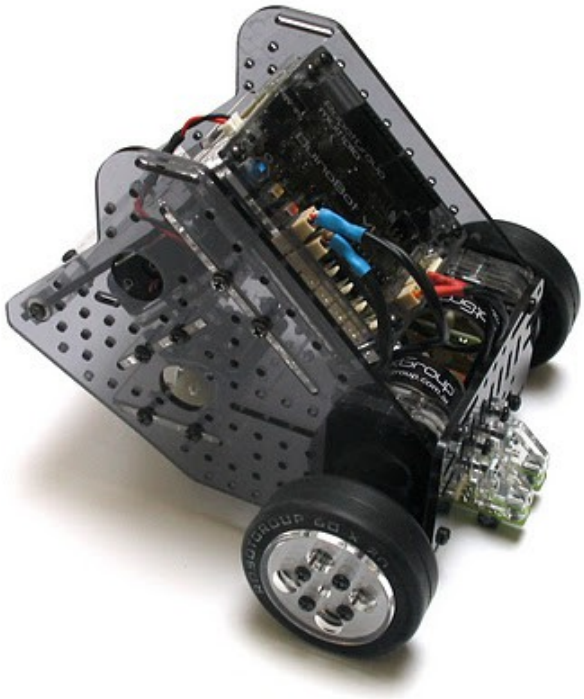
## 5.1. DuinoBot y DuinoBot.HID

El controlador DuinoBot es el más representativo de los controladores para robots que fabrica RobotGroup para los kits open source Multiplo. Está basado en un procesador AVR





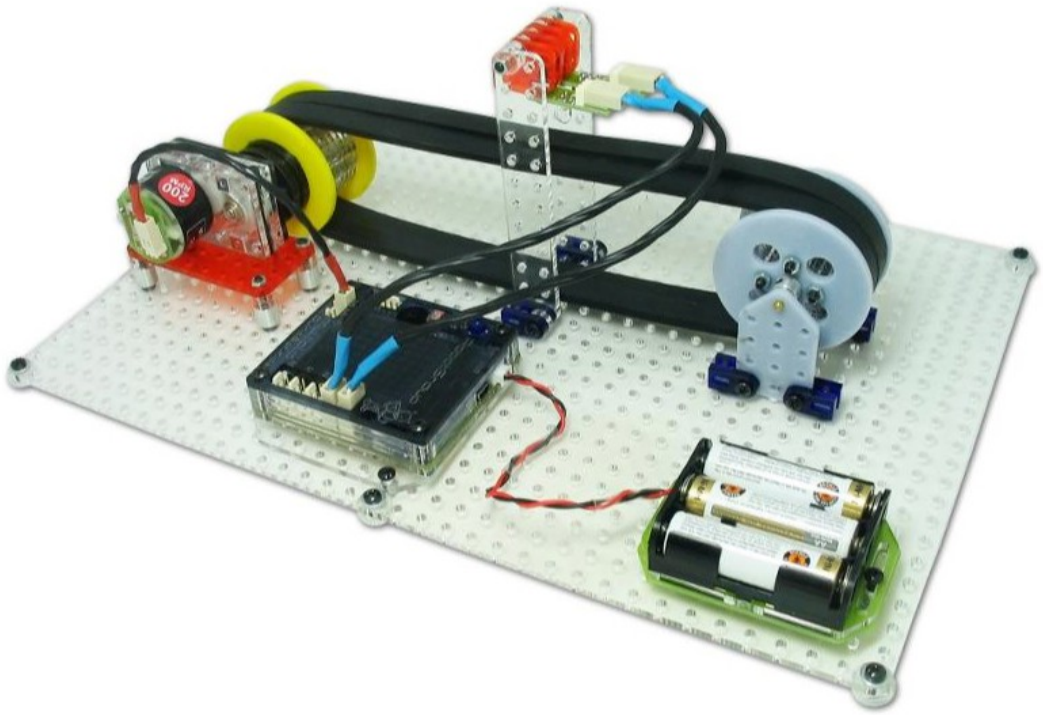
ATMega32U4, con USB integrado y equipa a los robots Multiplo.N6:



Minibloq genera código C/C++ nativo para estos controladores, compilándolo con WinAVR, que es el toolchain completo (compilador, librerías y software de bootloading) de GNU C/C++ (o GCC) para procesadores AVR. El sistema de bootloading utiliza el puerto USB y envía el archivo binario (.hex) a la memoria flash del ATMega32U4. Existen dos variantes de este controlador: la estándar, que se comunica por el puerto USB emulando un puerto serie (CDC) y la HID, que también utiliza el puerto USB pero a través de una conexión HID. Éste último tiene algunas ventajas, sobre todo la de no necesitar en absoluto de la instalación de drivers bajo Windows. La conexión USB es también más estable y rápida bajo Windows. En Linux requiere de un pequeño script de instalación, el cual debe ser ejecutado sólo una vez, con permisos administrativos.

## 5.2. DuinoBot.Kids

La controladora DuinoBot.Kids es similar a la placa DuinoBot en cuanto a su estructura interna, difiriendo en que incluye un display LED de matriz de puntos (8x8) y no tiene los conectores de expansión estándar de Arduino. Está orientada sobre todo a alumnos de escuelas primarias. Este controlador equipa a los kits para enseñanza de tecnología que RobotGroup comenzó a fabricar a mediados de 2011. La caja incluye además varios sensores (infrarrojos, luz visible, sonido, de choque) y unas 400 piezas mecánicas. Con ellos es posible realizar aproximadamente 70 armados, y todas sus funciones pueden ser controladas con Minibloq. A continuación se ve una cinta transportadora armada con este kit:

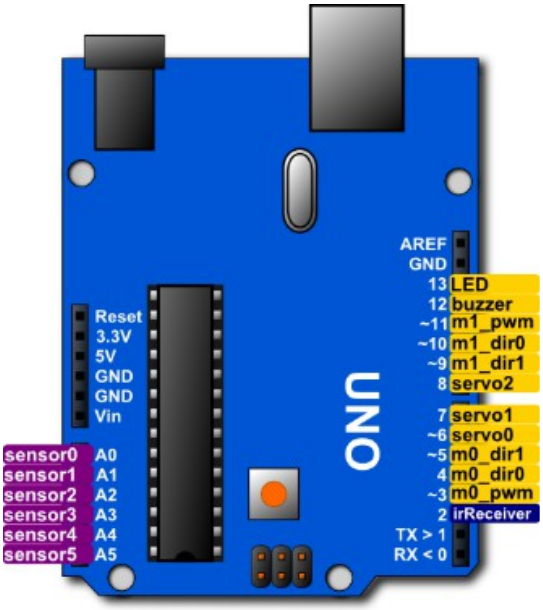


## 5.3. Arduino (y variantes)

### 5.3.1. Arduino Uno

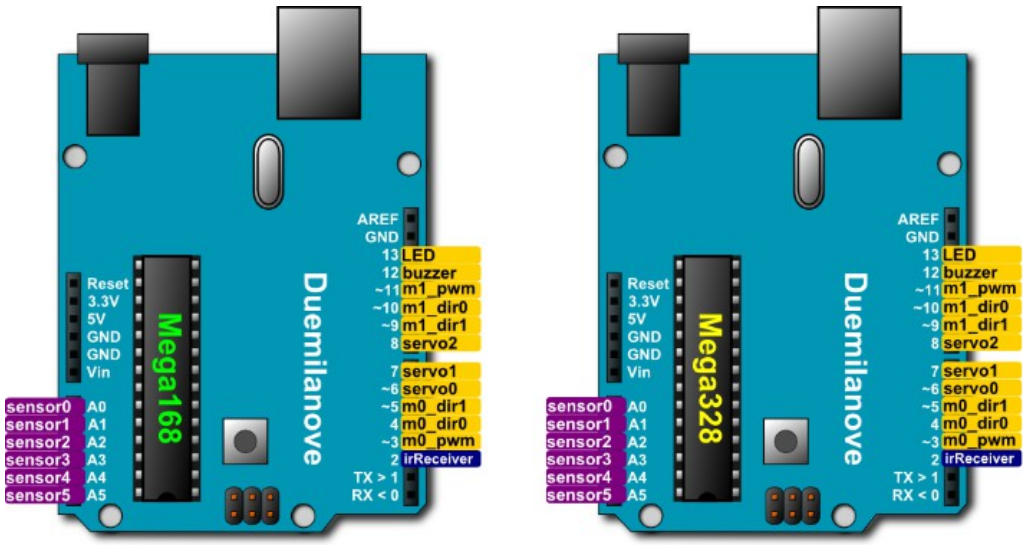
La placa de desarrollo Arduino es la plataforma de desarrollo open source más extendida del mundo. Con ella es posible prototipar rápidamente proyectos con microcontroladores, etc..

Como el entorno de programación original (Arduino IDE, también open source) sólo soporta sintaxis C/C++, Minibloq facilita la programación a aquellas personas que realmente quieren comenzar a programar microcontroladores y no tienen ninguna experiencia, especialmente los más jóvenes. La última iteración de Arduino es el modelo Arduino Uno. Su núcleo es un microcontrolador AVR ATMega328, y se comunica por USB con otro microcontrolador AVR secundario dedicado (ATMega8U2). A continuación se muestra la imagen conceptual que se creó específicamente para Minibloq, donde se indica con etiquetas dónde debe ser conectado cierto tipo de hardware que está asociado a ciertos pines (por cuestiones relativas a los periféricos internos del MCU, como los pines de salida de los contadores -timers-, etc.). Cada placa controladora soportada en el entorno tiene asociada una imagen de este tipo con sus respectivas etiquetas:



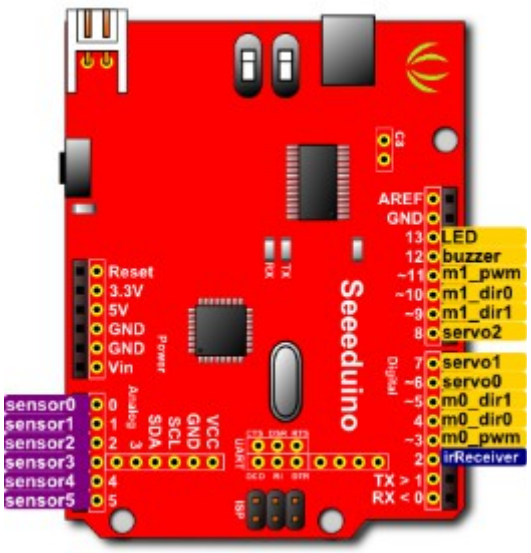

### 5.3.2. Arduino Duemilanove Mega328 y Mega168

La versión anterior a la placa Arduino Uno es Arduino Duemilanove. Ésta venía con 2 versiones, donde sólo difería el MCU (ATMega328 o ATMega168). La gran base instalada de estas placas en todo el mundo la convierte en una plataforma importante. Minibloq la soporta también:



### 5.3.3. Seeeduino v2.2 Mega328

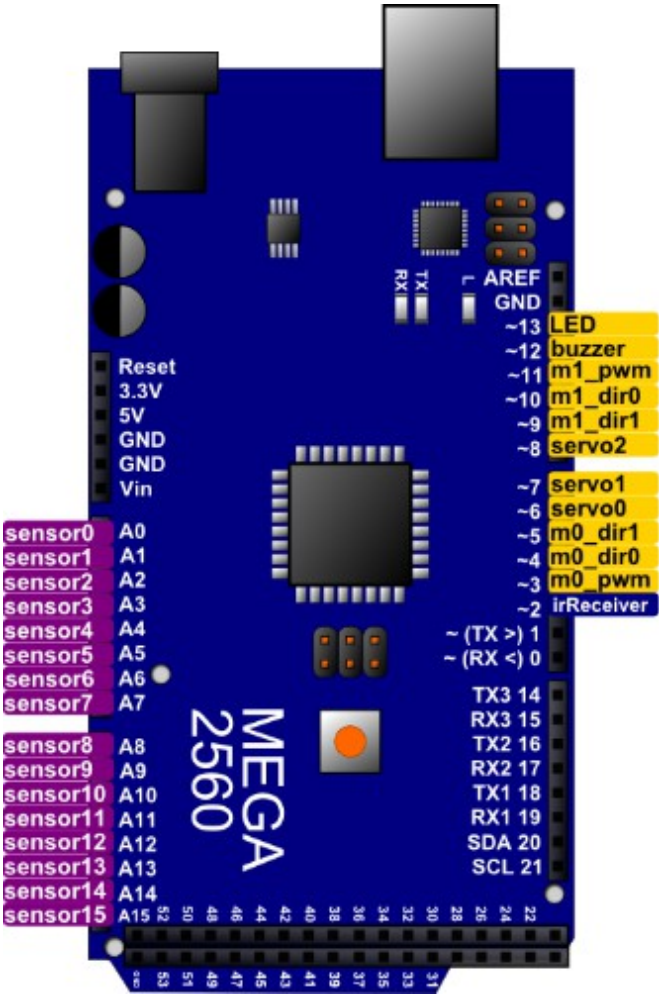
Al tratarse de una plataforma open souce, existen clones de Arduino, algunos de los cuales también están muy extendidos, por lo que se incluyeron en Minibloq. Uno de los más populares es el controlador Seeeduino v2.2 (también equipado con un ATMega328 como MCU):




## 5.4. Arduino Mega (y variantes)

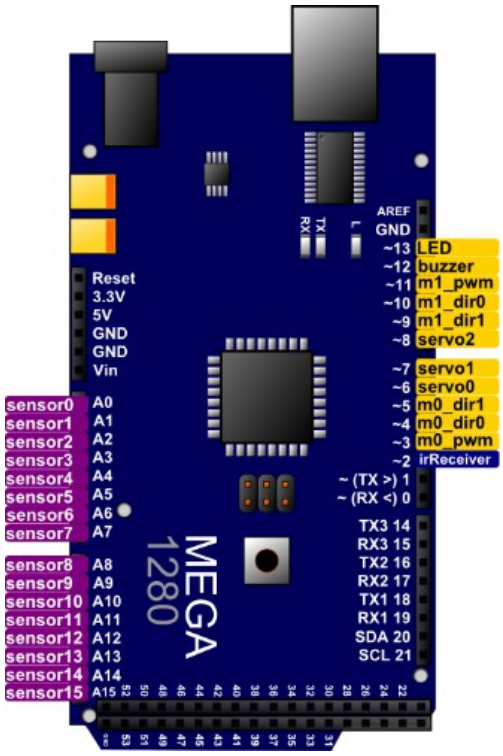
### 5.4.1. Arduino Mega 2560

El modelo más potente de Arduino es el Mega2560. Como su nombre lo indica cuenta con un MCU ATMega2560, el cual tiene 256 KBytes de memoria de programa (flash), muchos más pines de entrada salida (I/O) y gran cantidad de periféricos internos (tales como timers con salidas con modulación ancho de pulso -PWM-). Minibloq soporta este tipo de placa, con el mismo toolchain mencionado antes (WinAVR).



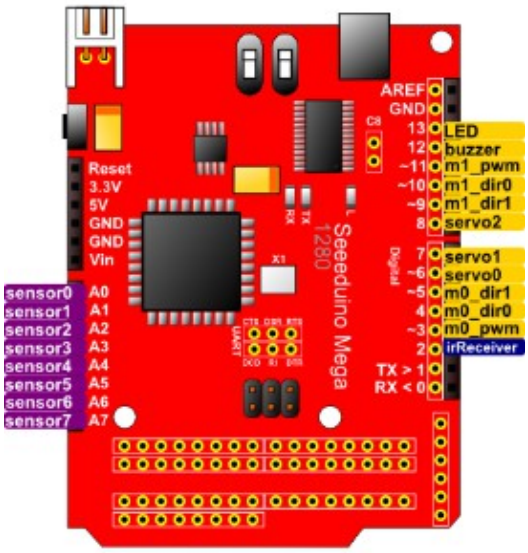
### 5.4.2. Arduino Mega 1280

Tal como ocurre con Arduino Uno y Duemilanove, el modelo anterior de Arduino Mega era el Mega 1280, de 128 KBytes de RAM. Difería también en el chip de comunicaciones USB, utilizando un FTDI en vez de un ATMega8U2. Esta placa está también muy extendida en el mundo y es posible programarla con Minibloq.




### 5.4.3. Seeeduino Mega 1280

La placa Seeeduino Mega 1280 es un popular clon del Arduino Mega 1280, más económico, y también fue incluido el soporte en Minibloq:

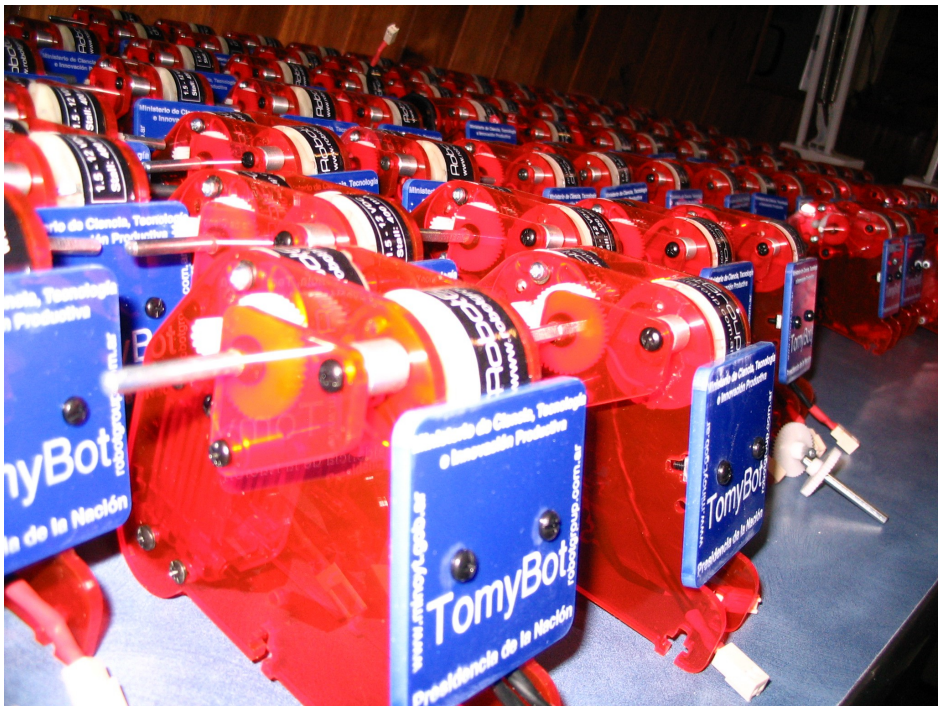



## 5.5. Maple (ARM Cortex M3)

Una de las características más interesantes de Minibloq es que no está acotado a una única familia de microcontroladores. Siempre fue la idea del proyecto soportar procesadores ARM. Originalmente se pensaba en soportar los ARM7TDMI, pero éstos ya no tienen prácticamente cuota de mercado en los nuevos desarrollos embebidos de 32 bits, sobre todo frente a los nuevos ARM Cortex M3. Esta familia de procesadores se está extendiendo rápidamente, e incluso en muchos casos está reemplazando con éxito a microcontroladores de 8 bits, dados su bajo costo y su bajo consumo. Uno de los kits de desarrollo más extendidos en esta familia de MCUs, y compatible con Arduino es Maple. Minibloq soporta en su versión actual los modelos de Maple entre las versiones Rev 3 y Rev 5, aunque de forma aún preliminar. Para esto, se utiliza también un toolchain de código abierto basado en el GNU C/C++, llamado ARM_EABI y mantenido (aunque 100% open source) por la empresa Codesourcery, Inc. (hoy de Mentor Graphics http://www.mentor.com). Como particularidad, se puede mencionar también que los microcontroladores ARM Cortex M3 pueden correr código tanto de memoria Flash como de Ram, siendo posible en Minibloq seleccionar también a qué memoria se envía el programa (bajar el código a RAM tiene la ventaja de que no desgasta la memoria Flash, la cual tiene un número limitado –aunque alto- de grabaciones, pero el espacio de programa en RAM es bastante menor, y por otro lado, volátil).

## 5.6. ATTiny25/45/85

Además de las placas y kits comerciales, se ha incluido en Minibloq la posibilidad de que el usuario programe hardware basado en los pequeños microcontroladores AVR ATTiny25, 45 y 85. Estos MCUs de tan sólo 8 pines son ideales para pequeños proyectos donde se requiere poca capacidad de entrada/salida a la vez que cierta capacidad procesamiento, ya que cuentan con un núcleo AVR no muy diferente al de los mayores ATMega328 de Arduino. RobotGroup ha fabricado este año un pequeño nuevo robot equipado con dichos procesadores, el TomyBot, cuyo chasis se puede ver a continuación. Éste también puede ser programado con Minibloq (si se conecta a un programador externo de los soportados, como el USBasp por ejemplo):

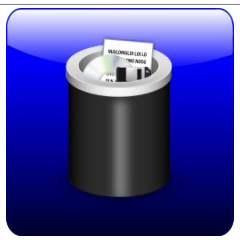
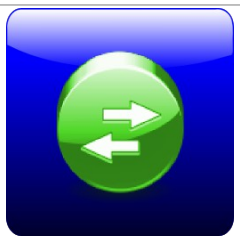


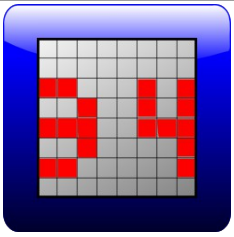
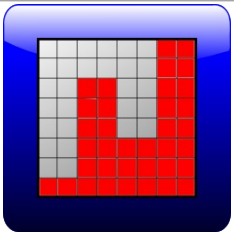
# 6. Bloques

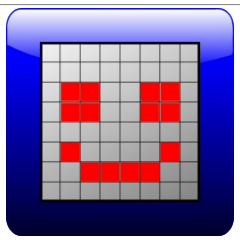
A continuación se describen brevemente los bloques de Minibloq y sus parámetros. En algunos casos se ha optado por no traducir el nombre del bloque, sobre todo en aquellos ligados a hardware (como por ejemplo "IOPin").

## 6.1. Selector de acciones

| Imagen | Nombre y descripción | Parámetros |
|---|---|---|
|  | **comienzo (de componente)**<br>Este bloque indica el comienzo del programa (o "Componente").<br><br>**Note:** Este bloque no se encuentra en el selector de acciones, pero es una acción de todos modos. Minibloq lo agrega automáticamente a cada nuevo componente. | ninguno |
|  | **mientras**<br>Este bloque es un inicio de ciclo "mientras" (while). Todos los bloques entre él y el próximo bloque de fin de ciclo se repetirán mientras se cumpla la condición especificada aquí. | **condición:** booleano |
|  | **repite**<br>Este bloque es un inicio de ciclo "repite" (repeat). Todos los bloques entre él y el próximo bloque de fin de ciclo se repetirán tantas veces como indique el parámetro numérico. | **cantidad de repeticiones:** número |
|  | **si (condicional)**<br>Este bloque permite tomar decisiones, en base a la condición lógica especificada. | **condición:** booleano |

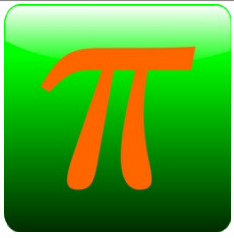
| | | |
|---|---|---|
|  | **esperar**<br>Este bloque hace que el programa espere durante el tiempo especificado (en milisegundos). | **intervalo (ms):** número |
|  | **variable (crear)**<br>Este bloque permite crear e inicializar una variable, para almacenar en ella un número o el resultado de una expresión, y de este modo poder utilizar dicho valor en otras partes del programa. | **valor inicial:** número |
|  | **variable (escritura)**<br>Este bloque permite asignar un valor a la variable seleccionada. | **valor:** número |
|  | **IOpin (escritura)**<br>Este bloque permite establecer el estado de un pin de salida digital del controlador. | **valor:** booleano |
|  | **AnalogWrite (escritura analógica)**<br>Este bloque permite controlar una salida analógica (PWM). | **valor:** número |
|  | **SerialNumber (escritura)**<br>Este bloque permite enviar datos (valores numéricos) por el puerto USB (o por un puerto serie, dependiendo del modelo de controlador) del controlador a la computadora. | **valor:** número |

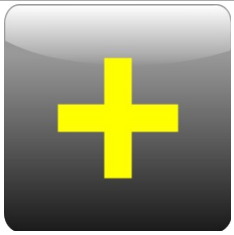
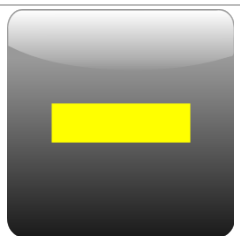
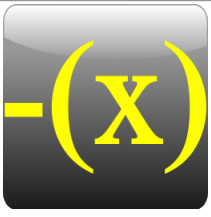
| | | |
|---|---|---|
|  | ## SerialText (escritura)<br>Este bloque permite enviar datos (texto) por el puerto USB (o por un puerto serie, dependiendo del modelo de controlador) del controlador a la computadora. | **valor:** texto |
|  | ## Buzzer (escritura)<br>Este bloque permite emitir sonidos. El primer parámetro determina la frecuencia (nota) y el segundo la duración. | **nota (Hz):** número<br>**duración(ms):** número |
|  | ## Motor (escritura)<br>Este bloque permite controlar un motor eléctrico conectado a una de las salidas de motor del controlador. | **potencia (-100 a 100):** número |
|  | ## ServoRC (escritura)<br>Este bloque permite establecer la posición (en grados) de un servo R/C conectado a una de las salidas digitales del controlador. | **ángulo:** número |
|  | ## ScreenNumber (escritura)<br>Este bloque permite mostrar números de -99 a +99 en la pantalla del controlador. | **valor (-99 a 99):** número |
|  | ## ScreenBars (escritura)<br>Este bloque permite mostrar 4 valores numéricos (que pueden ir de 0 a 100) en forma de barras en la pantalla del controlador. | **barra0 (0 a 100):** número<br>**barra1 (0 a 100):** número<br>**barra2 (0 a 100):** número<br>**barra3 (0 a 100):** número |

| | | |
|---|---|---|
|  | **ScreenSprite (escritura)**<br>Este bloque permite mostrar en la pantalla del controlador uno de los dibujos preinstalados. | **value:** dibujo |
|  | **comentario**<br>Este bloque permite agregar un comentario al programa. | ninguno |

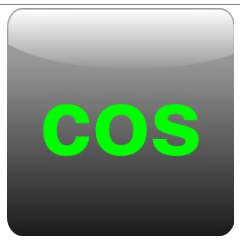
## 6.2. Selector contextual numérico

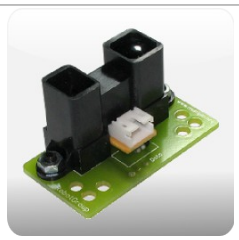
| Imagen | Nombre y descripción | Parámetros |
|---|---|---|
|  | **variable (lectura)**<br>Este bloque devuelve el valor almacenado en la variable seleccionada. | ninguno |
|  | **Motor (lectura)**<br>Este bloque devuelve el estado de un motor eléctrico conectado a una de las salidas de motor del controlador. | ninguno |
|  | **ServoRC (lectura)**<br>Este bloque devuelve la posición (en grados) de un servo R/C conectado a una de las salidas digitales del controlador. | ninguno |
|  | **AnalogRead**<br>Este bloque devuelve el valor de la entrada analógica (o "entrada de sensor") seleccionada. | ninguno |

| | | |
|---|---|---|
|  | **PulseIn**<br>Este bloque devuelve el tiempo (en microsegundos) que dura un pulso en la entrada digital seleccionada. Si el primer parámetro es verdadero (true), el bloque esperará hasta que la entrada tome dicho valor, y medirá el tiempo hasta que cambie a falso (false). Si fuera falso (false), esperará a que sea falso y luego a que vuelva a ser verdadero. El segundo parámetro especifica el tiempo máximo que el bloque quedará en espera para realizar la medición. | ninguno |
|  | **timeStamp**<br>Este bloque devuelve la cantidad de milisegundos desde que el programa comenzó. | ninguno |
|  | **número pseudoaleatorio**<br>Este bloque devuelve un número pseudoaleatorio entre 0 y 100. | ninguno |
|  | **buzzerNote (constante)**<br>Este bloque devuelve la frecuencia correspondiente a la nota seleccionada. | ninguno |
|  | **número (constante)**<br>Este bloque devuelve un número constante. | ninguno |
|  | **pi (constante)**<br>Este bloque devuelve la constante pi con un número limitado de decimales (3.14159265358979323846). | ninguno |

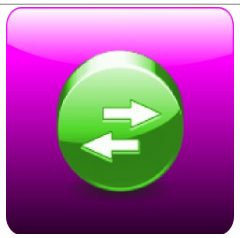
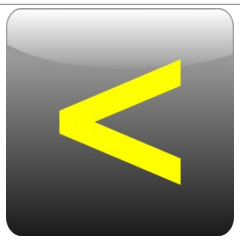
| | | |
|---|---|---|
|  | **e (constante)**<br>Este bloque devuelve la constante e (o "número de Euler") con un número limitado de decimales (2.7182818284590452354). | ninguno |
|  | **sumar**<br>Este bloque permite sumar números, variables y otras expresiones numéricas. | **valor1:** número<br>**valor2:** número<br>Nota: Más operandos (valores) pueden ser agregados con el botón de ***agregado de parámetros***:<br>  |
|  | **sustraer**<br>Este bloque permite sustraer números, variables y otras expresiones numéricas. | **valor1:** número<br>**valor2:** número |
|  | **multiplicar**<br>Este bloque permite multiplicar números, variables y otras expresiones numéricas. | **valor1:** número<br>**valor2:** número<br>Nota: Más operandos (valores) pueden ser agregados con el botón de ***agregado de parámetros***:<br>  |
|  | **dividir**<br>Este bloque permite dividir números, variables y otras expresiones numéricas. También se puede utilizar para expresar números como fracciones. | **numerador:** número<br>**denominador:** número |
|  | **negativo (signo negativo unario)**<br>Este bloque hace negativo al número o a la expresión numérica a su izquierda. | **valor:** número |

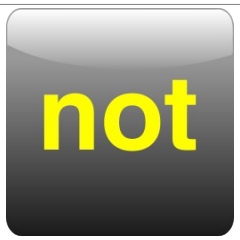
| | | |
|---|---|---|
|  | **potencia**<br>Este bloque permite elevar el primer parámetro (base) a la potencia dada por el segundo parámetro (exponente). | **base:** número<br>**exponente:** número |
|  | **valor absoluto**<br>Este bloque retorna el valor absoluto del número o de la expresión numérica a su izquierda. | **valor:** número |
|  | **resto (de la división)**<br>Este bloque devuelve el resto de la división entre sus 2 parámetros. | **valor1:** número<br>**valor2:** número |
|  | **mínimo**<br>Este bloque retorna el mínimo de dos números, variables u otras expresiones numéricas. | **valor1:** número<br>**valor2:** número |
|  | **máximo**<br>Este bloque retorna el máximo de dos números, variables u otras expresiones numéricas. | **valor1:** número<br>**valor2:** número |
|  | **mapear**<br>Este bloque permite mapear linealmente (realizando una regla de tres) un valor numérico desde un rango de valores hacia otro rango. | **x:** número<br>**fromLow:** número<br>**fromHigh:** número<br>**toLow:** número<br>**toHigh:** número |

| | | |
|---|---|---|
|  | **limitar**<br>Este bloque permite limitar un número, una variable u otra expresión numérica a un valor entre un mínimo y un máximo. | **x:** número<br>**a:** número<br>**b:** número |
|  | **seno**<br>Este bloque retorna el valor del seno del ángulo dado por el número o la expresión numérica a su izquierda (en radianes). | **valor:** número |
|  | **conseno**<br>Este bloque retorna el valor del coseno del ángulo dado por el número o la expresión numérica a su izquierda (en radianes). | **valor:** número |
|  | **tangente**<br>Este bloque retorna el valor de la tangente del ángulo dado por el número o la expresión numérica a su izquierda (en radianes). | **valor:** número |
|  | **arcoseno**<br>Este bloque retorna el ángulo (en radianes) cuyo seno es está dado por el número o la expresión numérica a su izquierda. | **valor:** número |
|  | **arcocoseno**<br>Este bloque retorna el ángulo (en radianes) cuyo coseno es está dado por el número o la expresión numérica a su izquierda. | **valor:** número |
|  | **arcotangente**<br>Este bloque retorna el ángulo (en radianes) cuya tangente es está dada por el número o la expresión numérica a su izquierda. | **valor:** número |

| | | |
|---|---|---|
|  | **IRRemote**<br>Este bloque devuelve el número obtenido de un sensor de control remoto infrarrojo. | ninguno |
|  | **Ping**<br>Este bloque devuelve la distancia aproximada (en cm) medida con un sensor ultrasónico Parallax's PING))) [TM], Seeed Studio SEN136B5Bm, Seeed Studio Grove Ultrasonic Ranger, o similar. | ninguno |
|  | **IRRanger (10 a 80 cm)**<br>Este bloque devuelve la distancia aproximada (en cm) medida con un sensor infrarrojo de distancia Sharp GP2Y0A21YK0F (10 a 80 cm aprox.). | ninguno |
|  | **IRRanger (20 a 150 cm)**<br>Este bloque devuelve la distancia aproximada (en cm) medida con un sensor infrarrojo de distancia Sharp GP2Y0A02YK0F (20 a 150 cm aprox.). | ninguno |

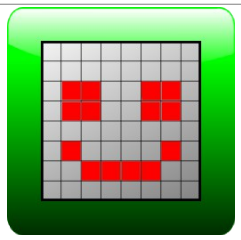
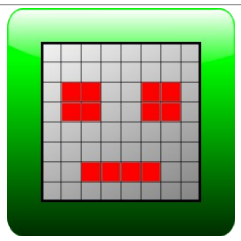
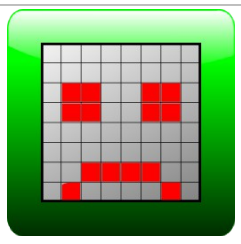
## 6.3. Selector contextual booleano

| Imagen | Nombre y descripción | Parámetros |
|---|---|---|
|  | **IOPin (lectura)**<br>Este bloque devuelve el estado de un pin digital del controlador. | ninguno |
|  | **verdadero (constante)**<br>Este bloque devuelve siempre "verdadero" (true). | ninguno |
|  | **falso (constante)**<br>Este bloque devuelve siempre "falso" (false). | ninguno |
|  | **igual**<br>Este bloque permite comparar números, y devuelve "verdadero" (true) si son iguales. | **valor1:** número<br>**valor2:** número |
|  | **distinto**<br>Este bloque permite comparar números, y devuelve "verdadero" (true) si son diferentes. | **valor1:** número<br>**valor2:** número |
|  | **menor que**<br>Este bloque permite comparar números, y devuelve "verdadero" (true) si el primer parámetro es menor al segundo parámetro. | **valor1:** número<br>**valor2:** número |

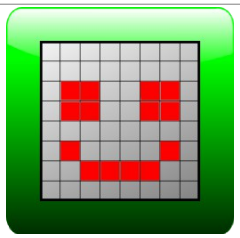
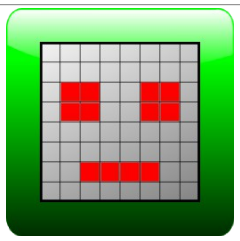
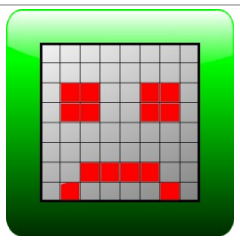
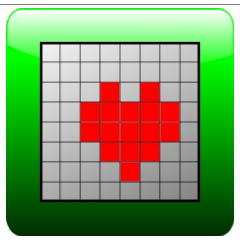
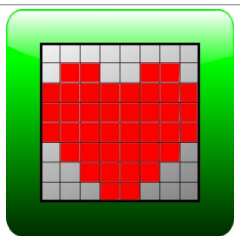
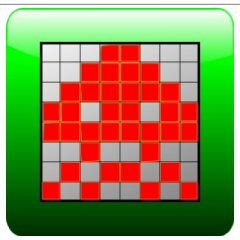
| | | |
|---|---|---|
|  | **mayor que**<br>Este bloque permite comparar números, y devuelve "verdadero" (true) si el primer parámetro es mayor al segundo parámetro. | **valor1:** número<br>**valor2:** número |
|  | **menor o igual que**<br>Este bloque permite comparar números, y devuelve "verdadero" (true) si el primer parámetro es menor o igual al segundo parámetro. | **valor1:** número<br>**valor2:** número |
|  | **mayor o igual que**<br>Este bloque permite comparar números, y devuelve "verdadero" (true) si el primer parámetro es mayor o igual al segundo parámetro. | **valor1:** número<br>**valor2:** número |
|  | **negación lógica**<br>Este bloque permite negar expresiones lógicas (booleanas). | **valor:** booleano |
|  | **y (función lógica)**<br>Este bloque realiza un "Y" lógico (booleano) y devuelve el resultado. | **valor1:** booleano<br>**valor2:** booleano<br>Nota: Más operandos (valores) pueden ser agregados con el botón de ***agregado de parámetros***:<br>  |
|  | **o (función lógica)**<br>Este bloque realiza un "O" lógico (booleano) y devuelve el resultado. | **valor1:** booleano<br>**valor2:** booleano<br>Nota: Más operandos (valores) pueden ser agregados con el botón de ***agregado de parámetros***:<br>  |

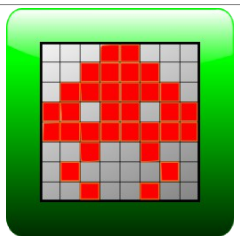
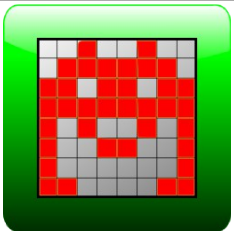
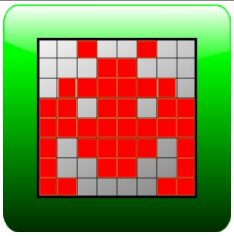
| | | |
|---|---|---|
|  | o-exclusivo (función lógica)<br>Este bloque realiza un "O-Exclusivo" lógico (booleano) y devuelve el resultado. | **valor1:** booleano<br>**valor2:** booleano |

### 6.4. Selector contextual de constantes de texto

| Imagen | Nombre y descripción | Parámetros |
|---|---|---|
|  | emoticonSmile (constante)<br>Este bloque devuelve el emoticon de una cara sonriente. | ninguno |
|  | emoticonWhatever (constante)<br>Este bloque devuelve el emoticon de una cara indiferente. | ninguno |
|  | emoticonAngry (constante)<br>Este bloque devuelve el emoticon de una cara enojada. | ninguno |
|  | texto (constante)<br>Este bloque devuelve texto (constante). | ninguno |

## 6.5. Selector contextual de gráficos

| Imagen | Nombre y descripción | Parámetros |
|---|---|---|
|  | spriteSmile (constante)<br>Este bloque devuelve el dibujo de una cara sonriente. | ninguno |
|  | spriteWhatever (constante)<br>Este bloque devuelve el dibujo de una cara indiferente. | ninguno |
|  | spriteAngry (constante)<br>Este bloque devuelve el dibujo de una cara enojada. | ninguno |
|  | spriteHeartSmall (constante)<br>Este bloque devuelve el dibujo de un corazón grande. | ninguno |
|  | spriteHeartBig (constante)<br>Este bloque devuelve el dibujo de un corazón pequeño. | ninguno |
|  | spriteInvader0 (constante)<br>Este bloque devuelve el dibujo del "Invasor 0". | ninguno |

| | | |
|---|---|---|
|  | **spriteInvader1 (constante)**<br>Este bloque devuelve el dibujo del "Invasor 1". | ninguno |
|  | **spriteInvader2 (constante)**<br>Este bloque devuelve el dibujo del "Invasor 2". | ninguno |
|  | **spriteInvader3 (constante)**<br>Este bloque devuelve el dibujo del "Invasor 3". | ninguno |