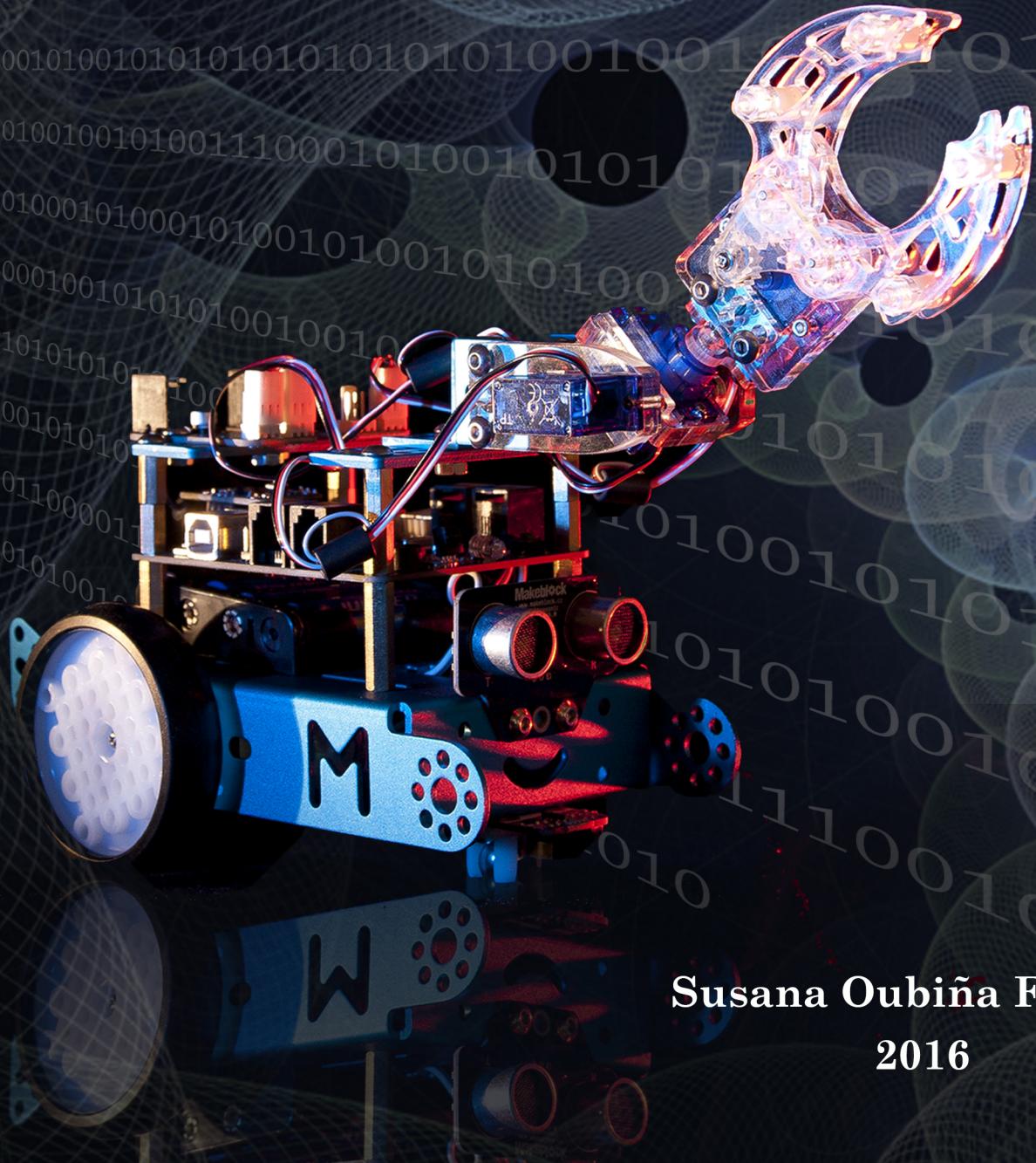


Divirtiéndome con mBot:

Guía de manejo y programación



**Susana Oubiña Falcón
2016**

Título original: Divirtiéndome con mBot: Guía de manejo y programación.

Autor: Susana Oubiña Falcón

Diseño de la portada: Sesé González García

Editor: Prodel S.A.

1^a Edición: junio, 2016



Susana Oubiña Falcón

ISBN-13: 978-84-608-9278-6



"Divirtiéndome con mBot: Guía de manejo y programación", por Susana Oubiña Falcón, es publicado bajo la licencia [Creative Commons Reconocimiento 4.0 Internacional License](#).

Me encanta la siguiente cita de *Albert Einstein*: “Si haces siempre lo mismo, no esperes resultados diferentes”. Para mí, en educación, sólo hay un camino: ¡Investiga, láñzate y prueba!

Índice

1. Introducción	3
1.1. Conectar mBot	3
1.2. Cargar el programa en mi mBot	6
2. Interface y Menú principal.....	11
2.1. Archivo	12
2.2. Editar.....	12
2.3. Conectar.....	13
2.4. Placas	15
2.5. Extensiones.....	16
2.6. Lenguaje	19
2.7. Ayuda.....	20
3. Comandos de los diferentes bloques en mBlock.....	21
3.1. Movimiento.....	21
3.2. Apariencia	22
3.3. Sonido.....	23
3.4. Lápiz	24
3.5. Datos y Bloques	25
3.6. Eventos	26
3.7. Control	27
3.8. Sensores.....	28
3.9. Operadores	29
3.10. Robots.....	30
4. Ejemplos de programación	34
4.1. Módulo de ultrasonidos	34
4.2. Motores	35
4.3. Motor y detección ultrasónica	37
4.4. Diodos RGB de la placa	38
4.5. Detector de línea	40
4.6. Matriz de LEDs 8x16.....	50
4.7. Robot sumo.....	57
4.8. Servomotor.....	66
4.9. Sensor de temperatura SD18B20.....	68

Divirtiéndome con mBot: Guía de manejo y programación

4.10. Sensor Me Temperatura y Me Humedad.....	74
4.11. Módulo PIR	76
4.12. Módulo Joystic.....	79
4.13. Módulo Me Touch o sensor táctil.....	80
4.14. Mini Gripper.....	81
4.15. Mini brazo articulado	84
4.16. Sensor de ángulo	87
4.17. Sensor de sonido	88
4.18. Potenciómetro	92
4.19. Módulo 4LEDs RGB	94
4.20. Módulo Display 7 segmentos.....	96
4.21. Módulo Brújula o Me Compass.....	100
4.22. Sensor de gas	101
4.23. Sensor de Luz	107
4.24. Módulo 4 botones.....	109
4.25. Pantalla TFT LCD 2.2.....	112
4.26. Módulo Micro Switch	116
4.27. Módulos controladores de motores.....	117
5. Arduino – Makeblock y viceversa	120
5.1. Makeblock – Arduino Uno	120
5.2. Arduino Uno - Makeblock	121
5.3. Otras placas con mBlock	129
5.3.1. Ejemplo 1: Semáforo con Arduino NANO trabajando con mBlock.....	130
5.3.2. Ejemplo 2: Naves invasoras del espacio con PicoBoard trabajando con mBlock.....	133
6. Placa Orion Base Board	138
7. Referencias de interés.....	141

1. Introducción

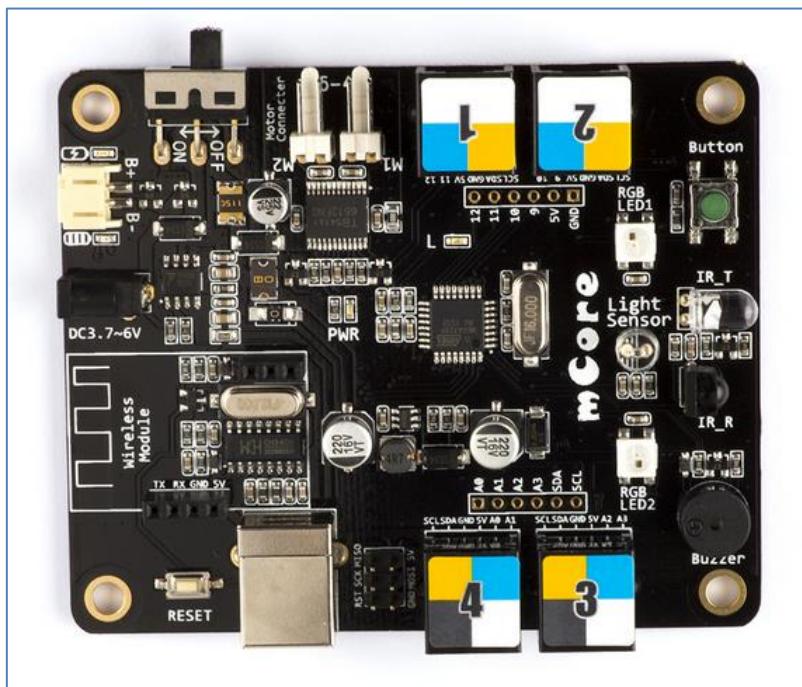
mBot es un robot de 400 gramos basado en Arduino Uno y creado por la casa Makeblock. Se puede alimentar por USB, por batería de litio o a través del conector del portapilas con 4 baterías AA.

En cuanto a su programación y control, hay 4 opciones: se puede programar desde *mBlock*, un software de programación gráfica muy similar a Scratch 2.0, y también se puede controlar mBot, directamente, sin necesidad de programación, desde una tablet o desde un Smartphone, gracias a su *app* (para Android o iOS) y a la posibilidad de acoplarle el módulo bluetooth. Además, el robot es muy versátil ya que podemos usar Arduino con mBot, programando mBot con el *IDE de Arduino* y no sólo con *mBlock*. En este último caso, necesitamos instalar las librerías de Makeblock en el IDE de Arduino. Librerías disponibles en la casa comercial para libre descarga a los usuarios. Finalmente, la última opción es programarlo con el software *mBlockly*.

Este documento se centrará en la programación del robot mBot desde el programa *mBlock*. Software que veo muy apropiado para el alumnado de la ESO ya que combina scratch con arduino en diferentes placas y robots (y no sólo con mBot). En muchas ocasiones, entraremos en Arduino y veremos que las librerías que proporciona la casa Makeblock son muy fáciles de manipular, modificar y entender, abriéndonos la puerta a usar estos robots para aprender arduino.

1.1. Conectar mBot

El robot mBot utiliza la placa mCore que puede verse en la imagen inferior. La placa, con un microcontrolador ATmega238, dispone de 4 puertos con conexiones RJ25 para conectar sensores y dos puertos para conectar motores. Además, mCore integra un interruptor de encendido, un botón, dos LEDs RGB, dos LEDs normales, un buzzer, un sensor de luminosidad y un sensor de infrarrojos receptor-emisor.



Placa mCore

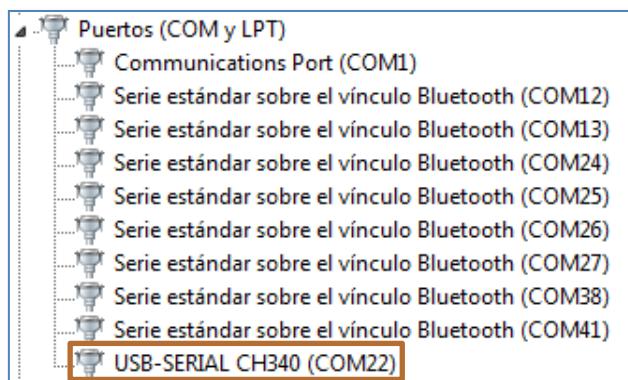
Divirtiéndome con mBot: Guía de manejo y programación

En la última versión de mBot, salida al mercado español en junio de 2016, la placa va protegida con una carcasa plástica.

La descarga del software mBlock, tanto para Windows como para iOS, puede hacerse desde el siguiente link: www.mblock.cc

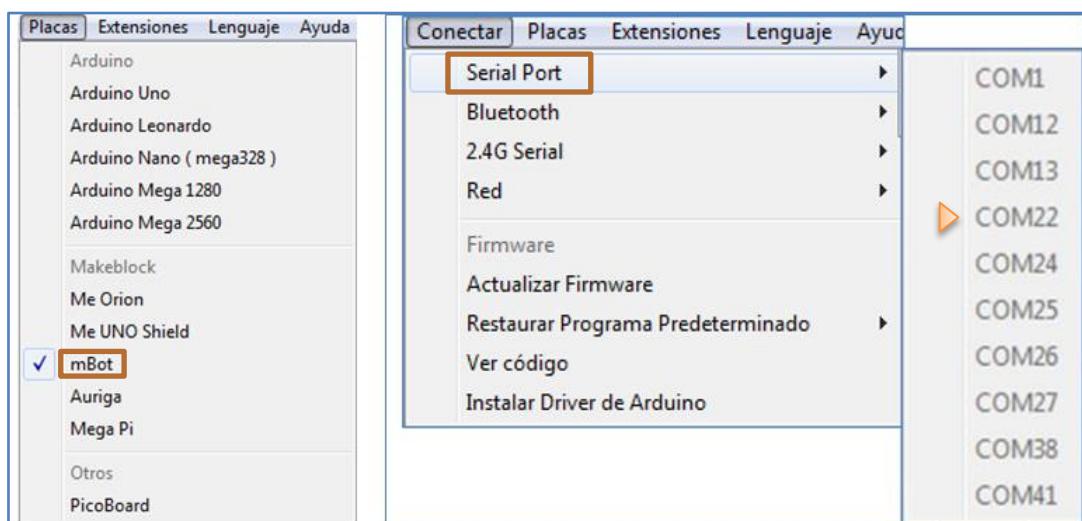
En cuanto a Linux¹, el programa funciona a través del Wine, software que nos permite instalar aplicaciones de Windows en Linux, dando permisos al usuario en el archivo ttyUSB0.

Tras abrir el programa mBlock, lo primero que debemos hacer es conectar el robot al software que lo controlará. Para ello, conectamos el cable USB desde el mBot al ordenador. En ese momento, si los drivers de mBot están instalados, se nos habilita un puerto COM de comunicación para realizar la conexión. En nuestro caso, nuestro puerto es el COM22, como puede verse en la siguiente imagen. Si queremos conocer el puerto que utiliza nuestro PC en ese momento, en Windows7 podemos acceder a *Equipo>Administrador de dispositivos*, en el apartado de Puertos (COM y LPT):



Puerto de conexión USB-SERIAL CH340

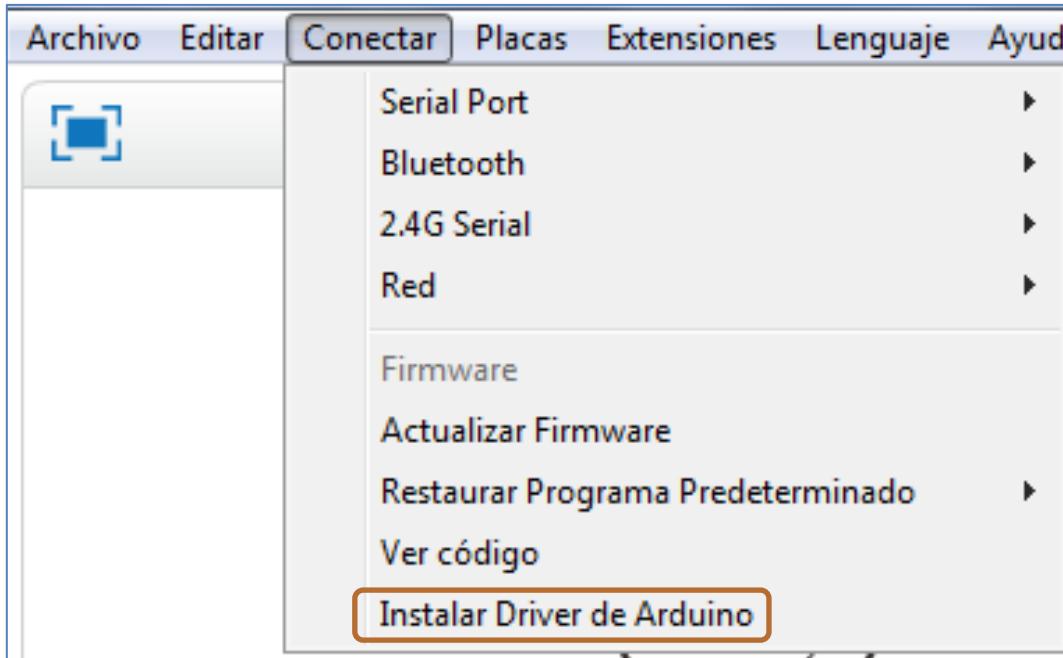
Ya en el programa *mBlock*, vamos a la pestaña “Placas” y escogemos *mBot* para finalmente, ir a la pestaña “Conectar” del menú principal y en su apartado *Serial Port*, seleccionar el puerto de conexión COM22:



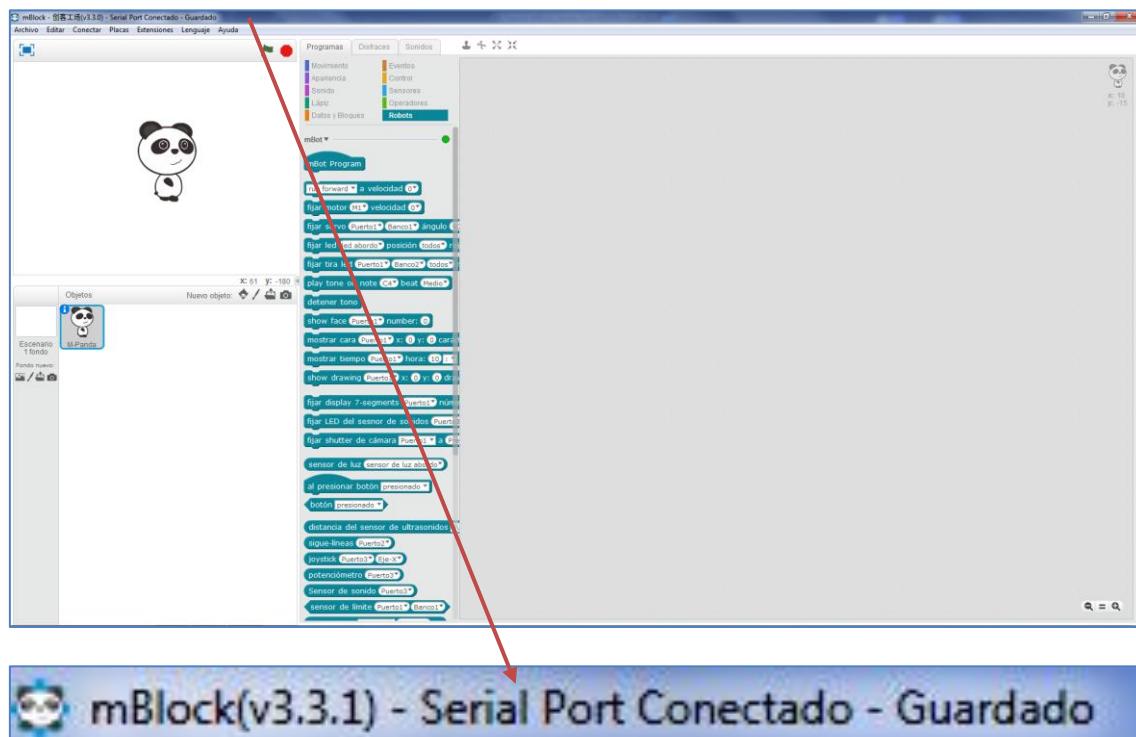
¹ Tutorial: <http://www.makeblock.es/foro/topic/25/mblock-con-linux/1#post-144>

Divirtiéndome con mBot: Guía de manejo y programación

En caso de no conectar, sólo debemos revisar si hemos instalado el driver de arduino que nos proporciona el puerto de conexión USB-SERIAL CH340. Este driver puede instalarse directamente desde el programa mBlock, a través de la pestaña *Conectar* usando la opción *Instalar Driver de Arduino*:



Tras estos pasos, el programa nos debe mostrar que mBot está conectado:



En el bloque **Robots** observamos que mBot está conectado (círculo en verde):



Nuestro trabajo ahora, será simplemente programar el robot utilizando los diferentes bloques que nos proporciona mBlock. El software mBlock nos permite conectarnos con el robot por puerto USB (que es lo que hemos hecho hasta ahora), por Bluetooth o por 2.4G.

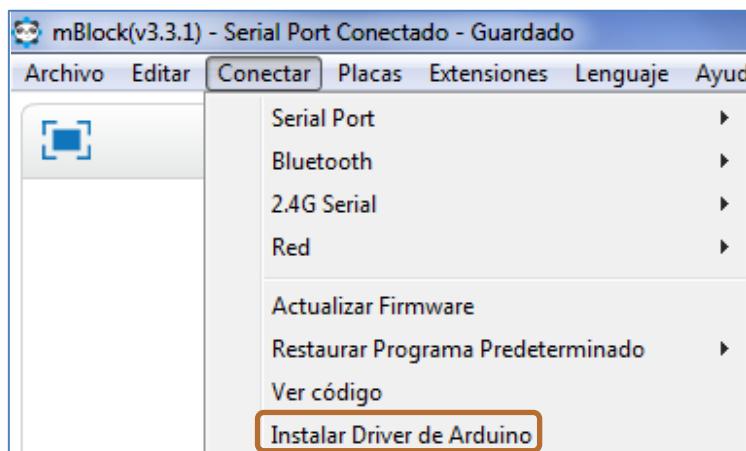
1.2. Cargar el programa en mi mBot

Podría interesarnos que nuestro programa funcione completamente **independiente** del ordenador (sin cable USB, Bluetooth o 2.4G). Para ello, una opción es grabarlo en el Arduino del mBot:

IMPORTANTE: Con esta opción de grabado, en nuestro programa no podemos utilizar ningún comando propio de Scratch. Por ejemplo, en el bloque Apariencia, "Decir..." correspondería al personaje "Panda" del programa mBlock y chocaría con la opción de grabado.

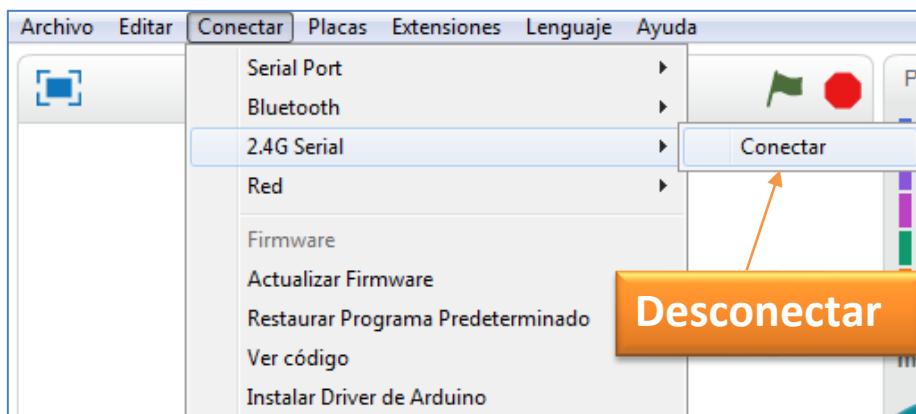
Los pasos a seguir para grabar el programa creado en el Arduino del mBot (mCore) son los siguientes:

- ❖ Paso 1: Debemos instalar (si procede) los drivers del mBot: *Conectar > Instalar Driver de Arduino*

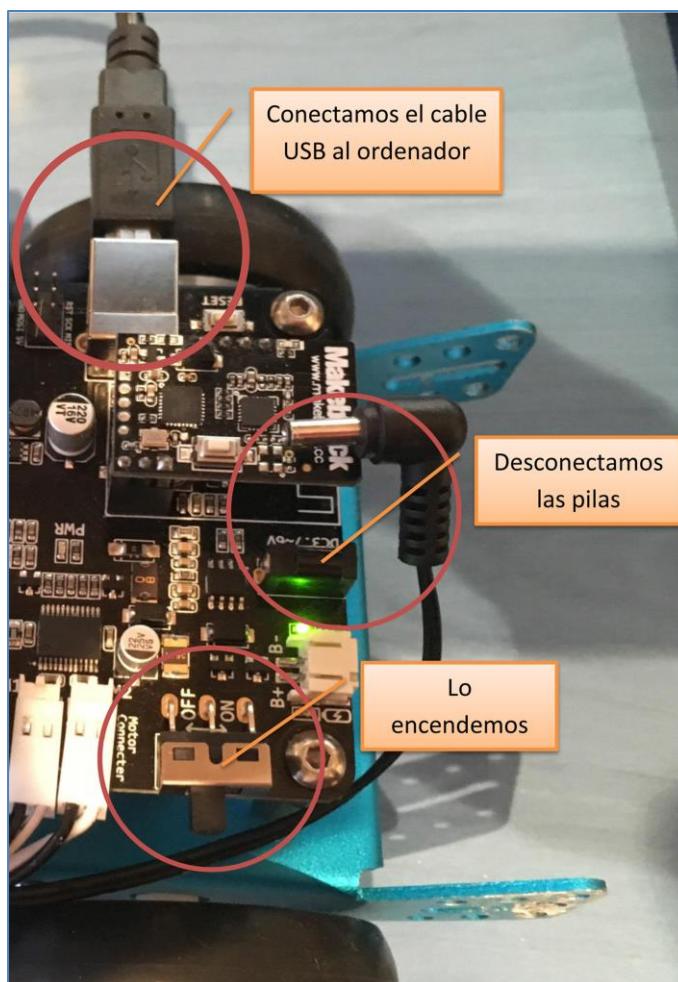


Divirtiéndome con mBot: Guía de manejo y programación

- ❖ Paso 2: Crear el programa que queremos grabar dentro de la placa mCore.
- ❖ Paso 3: Desconectamos nuestro mBot del ordenador ya que el robot podría estar conectado por Bluetooth o por 2.4GHz. En la siguiente imagen, se ve que desconectamos (no activa) la conexión 2.4G:

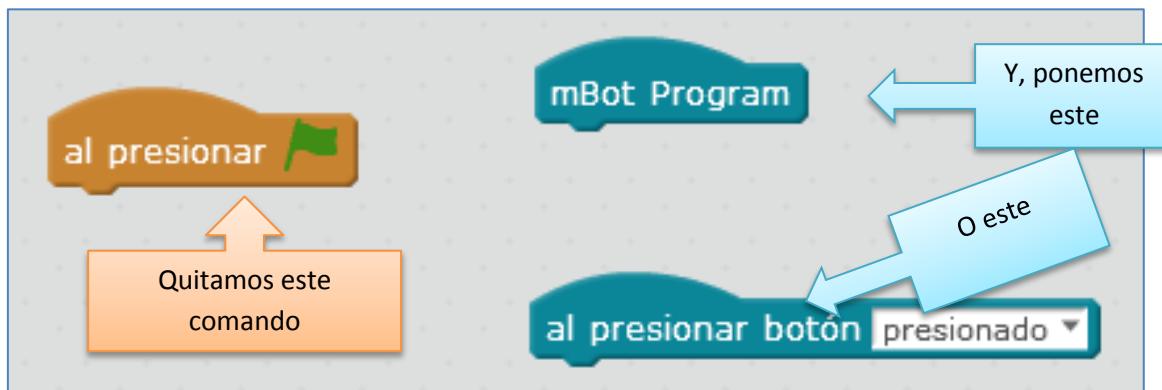


- ❖ Paso 4: Conectamos el mBot con el cable USB, desconectándolo de las baterías y lo encendemos:



Divirtiéndome con mBot: Guía de manejo y programación

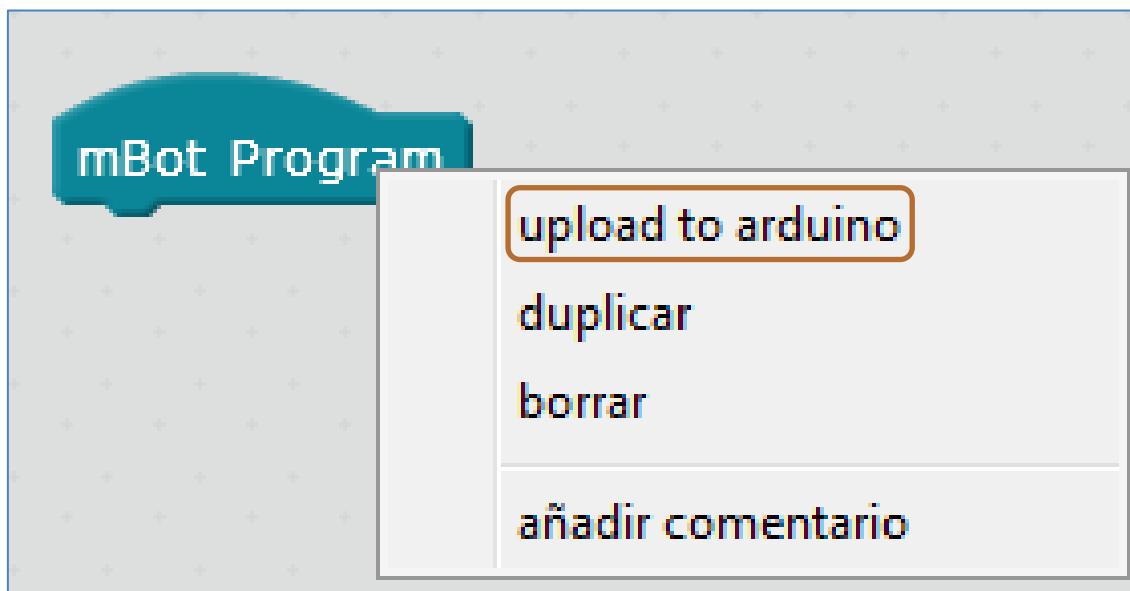
- ❖ Paso 5: Conectamos el mBlock con mBot por el puerto serie (Serial Port) COM X correspondiente:
- ❖ Paso 6: Cambiamos en nuestro programa el comienzo del programa (el de la bandera) por uno propio del robot “[mBot Program](#)”:



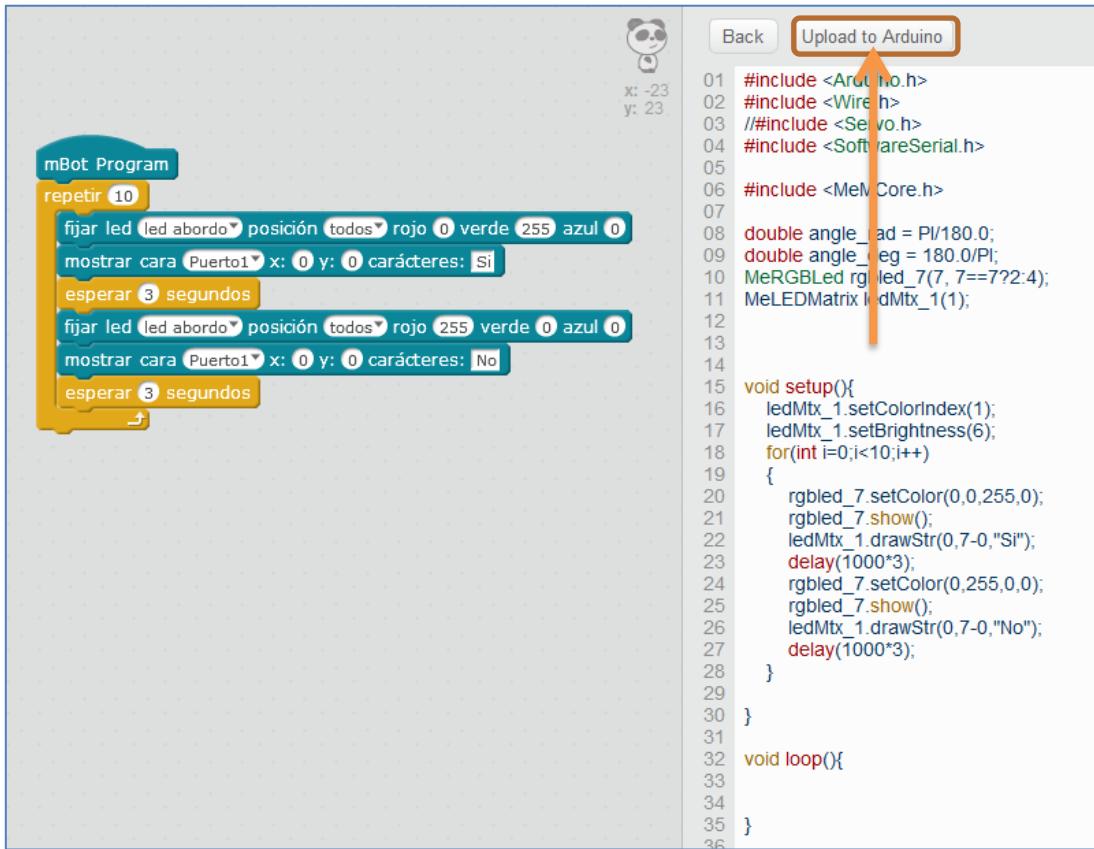
Usamos el comando “mBot Program”

IMPORTANTE: Sólo podemos cargar en la placa un único comando “[mBot Program](#)”.

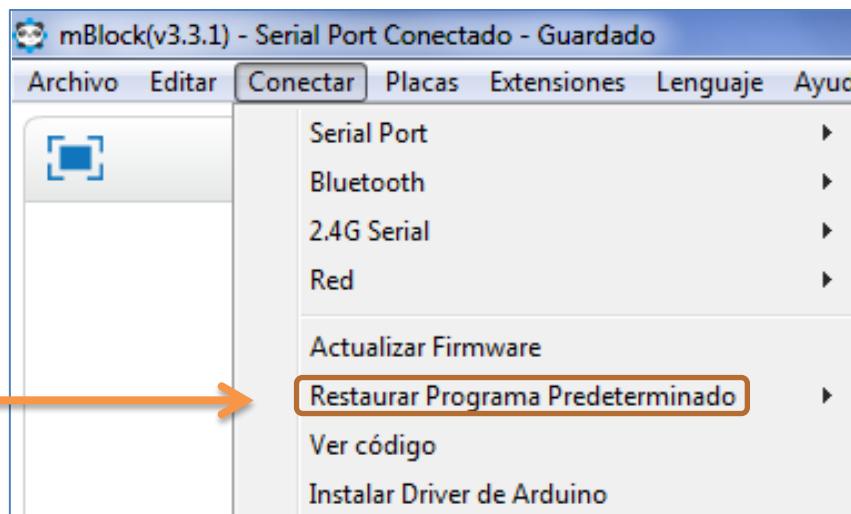
- ❖ Paso 7: Con el botón derecho, pinchamos en el comando [mBot Program](#) y elegimos *Upload to Arduino*. Otra opción es Editar > Modo Arduino > Upload to Arduino:



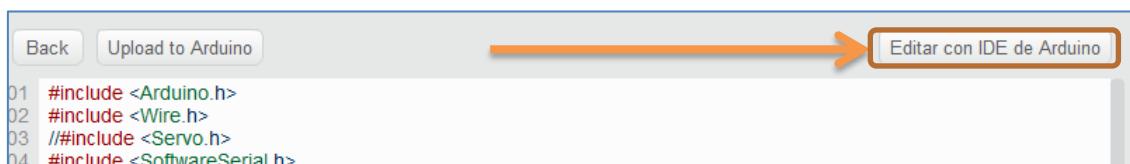
- ❖ Paso 8: Se nos abrirá una ventana con el código para grabarlo en el Arduino del mBot. Incluso, podemos modificar lo que queramos. Finalmente, hacemos clic en *Upload to Arduino* (ver siguiente imagen):



- ❖ Paso 9: Si no ha habido fallos, nos saldrá un mensaje que nos informa que el programa se ha grabado correctamente, pudiendo comenzar a disfrutar del programa introducido en el robot, sin el ordenador encendido. Para ello, debemos desconectar el cable USB y conectar las pilas o la batería de litio del robot. Veremos que nuestro mBot funciona de forma independiente.
- ❖ Paso 10: ¿Cómo se quita el programa que hemos instalado? Obviamente, en algún momento necesitaremos quitar el programa que hemos instalado para introducir otro nuevo o simplemente, seguir trabajando con el mBlock (sino, no puede volver a conectarse con el mBlock usando la bandera verde, por ejemplo). Para ello, sólo debemos seguir los siguientes pasos:
 - a. Tenemos que volver a desconectar las pilas o batería y conectar el cable USB del mBot al PC (ver paso 4).
 - b. Conectar mBot por el puerto serie correspondiente (ver paso 5). Si no aparece, debemos instalar otra vez los drivers.
 - c. Resetear el arduino del mBot. Este paso borrará nuestro programa y dejará el robot como estaba antes, con el programa de fábrica: *Conectar > Restaurar Programa Predeterminado*



Si nos fijamos, en el Paso 8 tenemos la opción de editar nuestro archivo con el IDE de Arduino y, por lo tanto, cargarlo desde ahí:



Resumiendo, mBot tiene dos formas de funcionar atendiendo a la dependencia o independencia de un PC:

- **Dependiente del ordenador:** hay tres formas de conectarlo. A saber, conexión inalámbrica 2.4G o conexión Bluetooth y mediante una conexión por cable USB.
- **Independiente del ordenador:** Sólo por Conexión por cable USB se carga el programa (Upload to Arduino). Luego, se desconecta y el robot ya puede ir sólo sin necesidad de un PC.

Es interesante el siguiente tutorial para trabajar con bluetooth:

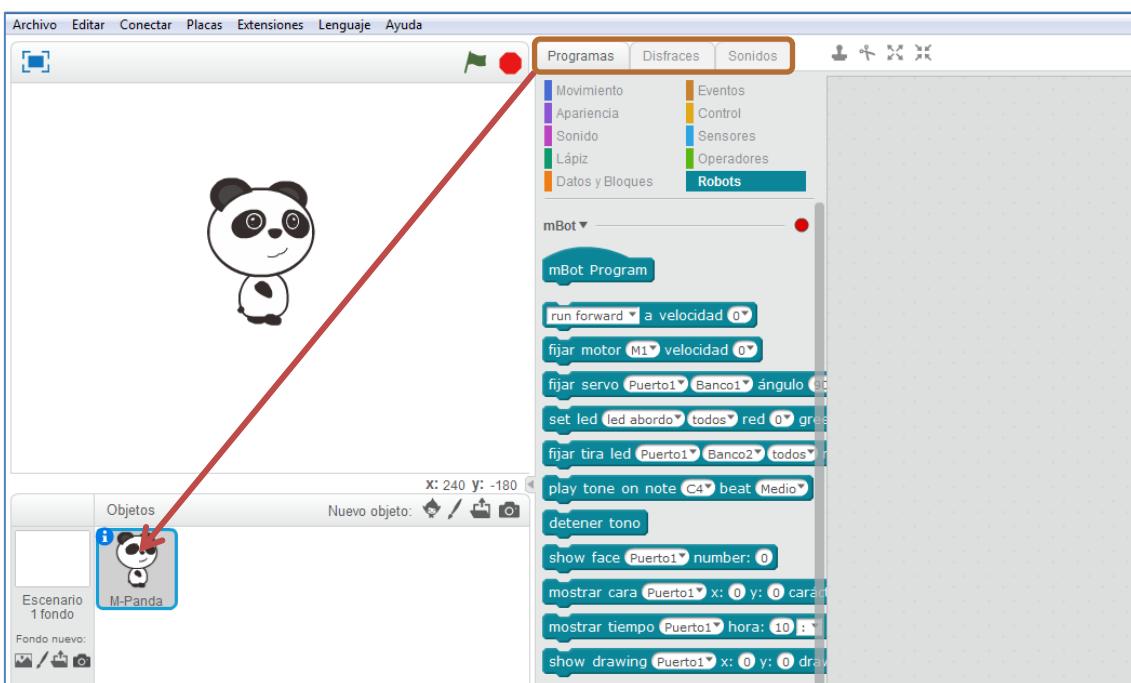
<https://www.youtube.com/watch?v=FxAsufUNcII>

2. Interface y Menú principal

El software mBlock se basa en scratch. De hecho, ambas interfaces (Scratch2.0 y mBlock) son muy parecidas.

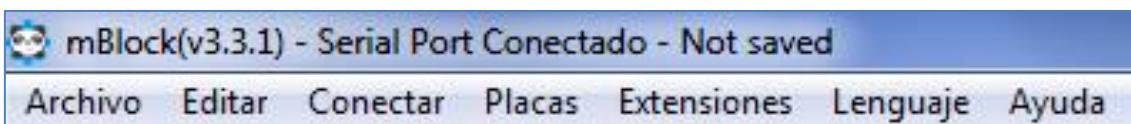
Scratch es un lenguaje de programación desarrollado por un equipo dirigido por Mitchell Resnick en el Media Lab del MIT. Sus orígenes nacen en el lenguaje LOGO, el cual fue pensado y creado para ser utilizado por niños (estudiantes), para que ellos desarrollaran habilidades matemáticas. Se programó bajo el lenguaje llamado squeak y éste, a su vez, a partir del lenguaje smalltalk. Ambos, lenguajes orientados a objetos. En consecuencia, Scratch, que es una evolución del LOGO, es *un lenguaje de programación que se caracteriza por estar totalmente orientado a objetos*. Es decir, los objetos se comunican entre ellos dentro de un mundo virtual. Además, se caracteriza por su simplicidad y por ser un lenguaje modular (utilizando bloques).

El software mBlock sigue su misma filosofía. En él, cada objeto y escenario presentan su propia pestaña *Programas*, *Disfraces* (o *Fondos* si es un escenario) y *Sonidos*, tal y como puede verse en la siguiente imagen:



Interface de mBlock

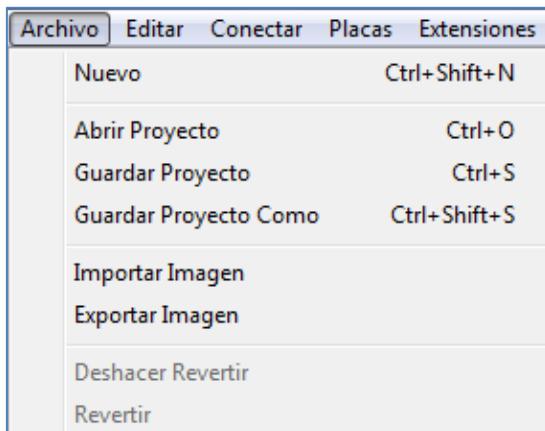
El menú principal se compone de siete pestañas, comenzando por la celda Archivo y finalizando en la celda de selección Ayuda. En su parte superior podemos ver la versión del programa que se está utilizando y si la placa que se pretende programar está o no conectada con el software por puerto serie.



A continuación se desarrollan las opciones de cada una de las pestañas del menú superior.

2.1. Archivo

Las opciones de la pestaña “Archivo” son:

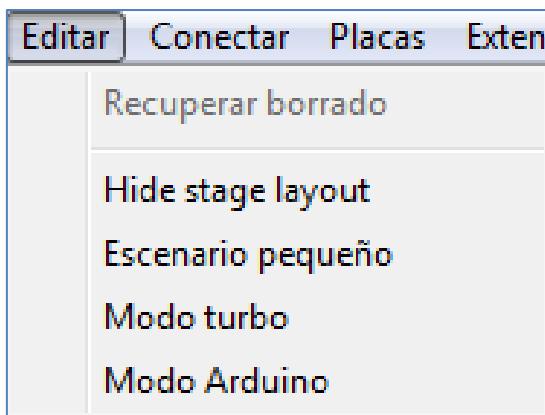


Opciones de la pestaña Archivo

- *Nuevo*: crea un nuevo proyecto
- *Abrir Proyecto*: Abre un proyecto existente
- *Guardar Proyecto*: Guarda el archivo
- *Guardar Proyecto Como*: Guarda el proyecto con un nombre cuya extensión es .sb2
- *Importar y exportar imagen*: introduce o exporta una imagen de un objeto o sprite.
- *Deshacer Revertir*: Deshace la acción revertir
- *Revertir*: Vuelve el programa al estado que tenía anteriormente

2.2. Editar

Las opciones de la pestaña “Editar” son:



Opciones de la pestaña Editar

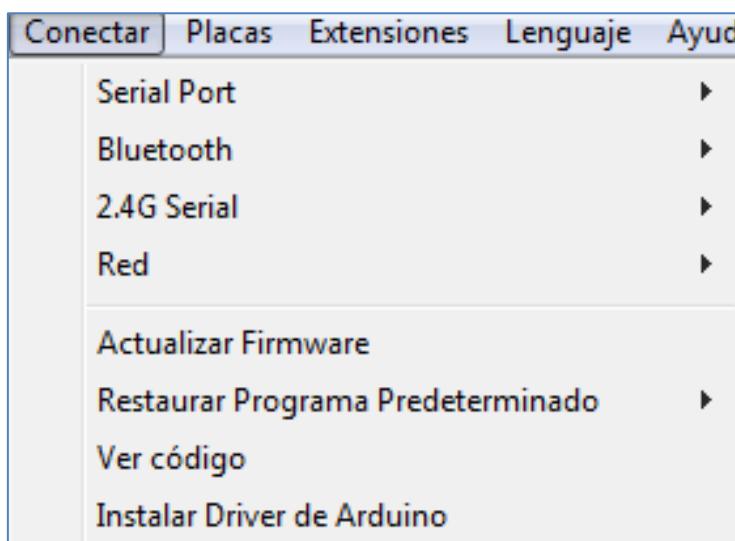
- *Recuperar borrado*: Restaura lo borrado anteriormente
- *Hide stage layout*: Oculta el escenario para hacer el área de edición de scripts más grande.
- *Escenario pequeño*: Hace el escenario más pequeño, aumentando el área de programación.

Divirtiéndome con mBot: Guía de manejo y programación

- *Modo turbo:* Acelera de velocidad de procesamiento del programa en la simulación scratch.
- *Modo Arduino:* Convertir el programa de mBlock a Arduino programa y lo sube a la placa principal Arduino para realizar la operación fuera de línea.

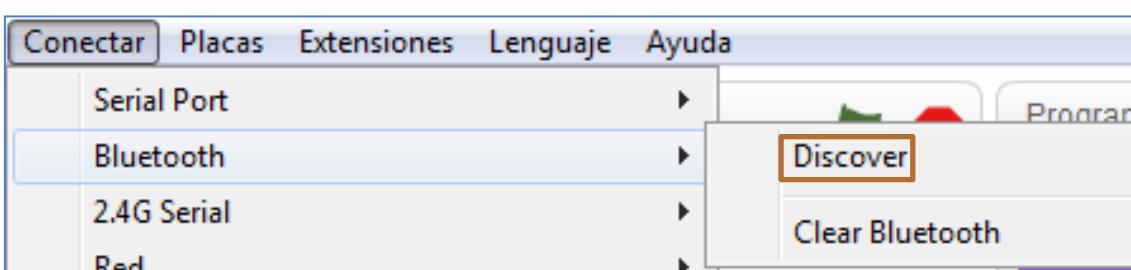
2.3. Conectar

El robot mBot soporta la programación inalámbrica, y eso significa que no tenemos que tener el robot conectado continuamente al PC para poderlo programar en Scratch. Con mBot disponemos de varias opciones de comunicación inalámbrica: Bluetooth o 2.4GHz y con cualquiera de ellas (no ambas a la vez) podemos programar el mBot. Con 2.4GHz, el robot se sincroniza automáticamente con el ordenador sin necesidad de "buscar dispositivo..." evitando de este modo interferencias con otros mBots, cuando se utilizan varios a la vez en un aula. También podemos descargar el programa al controlador directamente, por comunicación inalámbrica.



Las opciones de la pestaña “Conectar” son:

- *Serial Port:* Conecta el mBot al puerto COM donde está instalado el driver USB-SERIAL CH340
- *Bluetooth:* Descubre el Bluetooth para conectar y borra la conexión bluetooth. Se debe desconectar el puerto anterior. Primero debemos descubrir el modulo bluetooth que se llamará Makeblock y después conectarlo. Si el robot se conecta por bluetooth (dispone del módulo BT) puedes acoplarlo al mBot cada vez que queramos usarlo desde el móvil o tablet.



Divirtiéndome con mBot: Guía de manejo y programación

- **2.4G Serial:** Se conecta por 2.4G serial conectando primero el pendriver 2.4G en el PC y sincronizándolo después con el módulo mBot. Tras la vinculación con éxito, se hace clic en conectar. El 2.4G te permite usarlo remotamente desde el PC de una forma sencilla y en tiempo real mientras vas haciendo cambios en Scratch. También podrás usar esta versión del robot (2.4G) con el cable USB incorporado y con el mando de control remoto (incluido en ambas versiones) cuyas teclas se pueden programar también desde Scratch. La versión mBot 2.4G no lleva bluetooth y por lo tanto, con ella no podemos manejar el robot desde el móvil.

La tecnología 2.4G es la misma que utilizan los ratones y teclados inalámbricos para conectarse con el PC. Disponemos de dos modos 2.4G y, para cambiar entre un modo u otro, simplemente debemos pulsar el botón en el módulo de 2.4G:

Modo lento intermitente: Cuando el indicador 2.4G parpadea lentamente, significa que está en modo lento intermitente. En este modo, el módulo 2.4G tiene función de memoria y recuerda el adaptador 2.4G al que se ha conectado antes, de modo que, no se conectará a cualquier nuevo adaptador automáticamente.

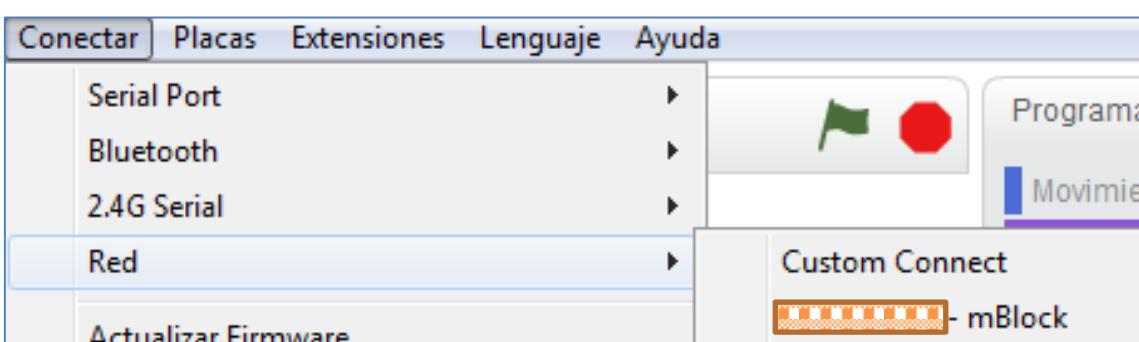
Modo rápido intermitente: Cuando el indicador parpadea 2.4G con velocidad rápida, significa que está en modo rápido de parpadeo. En este modo, el módulo 2.4G se conectará automáticamente al adaptador 2.4G que está encendido. Sin función de memoria en este modo.

La conexión inalámbrica del módulo 2.4G trae un pincho USB (no necesita ningún driver ya que el ordenador lo interpreta como un ratón inalámbrico). Cada pincho va asociado a un robot (o mejor dicho: La placa y el pincho 2.4G están emparejados y si el led de la mochila parpadea es que ha perdido conexión).

En caso de que el programa que realices con Scratch lo quieras dejar grabado permanentemente en la placa del robot, deberás usar el cable USB. Esto es así para que el Scratch pueda garbarlo en la placa que es un Arduino UNO "tuneado"

Importante: Ten en cuenta que el mBot sólo puede ser de 2 tipos. A saber: Bluetooth o 2.4G, pero nunca a la vez.

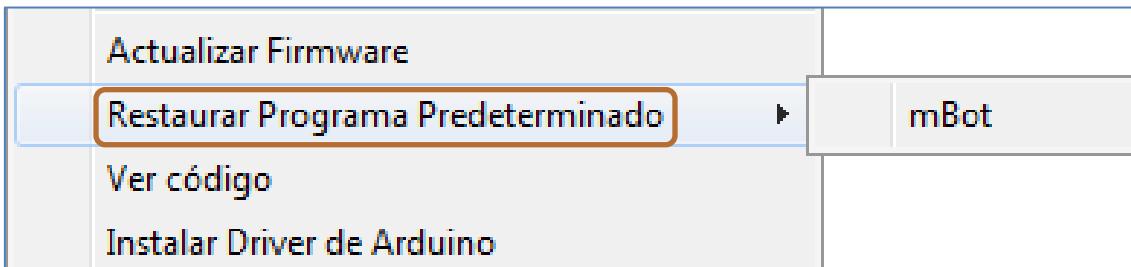
- **Red:** Descarga el programa por conexión directa inalámbrica.



Un *firmware* es un programa informático (con lógica de bajo nivel) que maneja el hardware del robot. Aquí, es un programa especial que se carga en la placa para

poder utilizar la App para controlar el robot desde iPhone o Android via bluetooth, o para poder usar mBlock.

- *Actualizar Firmware:* De esta forma nuestro robot se entendería perfectamente con mBlock. Se debe esperar a que la actualización finalice para no perjudicar la placa de mBot. Esta opción se usa con las placas o shields de Makeblock.
- *Restaurar Programa Predeterminado:* Restaura la configuración de fábrica de forma manual. Las opciones que se abren dependerán del robot conectado: mBot, Starter o mBot Ranger. Debemos esperar a que la restauración se complete ¿Cómo hacerlo?:



En primer lugar, asegúrese de que ha descargado la última versión del mBlock.

En segundo lugar, encienda el mBot con las baterías conectadas y conectar mBot al ordenador con el cable USB.

En tercer lugar, abra el software mBlock. A continuación, seleccione el puerto serie correcto en el menú "Conectar" y elija la placa mBot en el menú "Placas".

En cuarto lugar, haga clic en el "*Restaurar Programa Predeterminado*" en el menú "Conectar" y espere hasta que tenga éxito 100%.

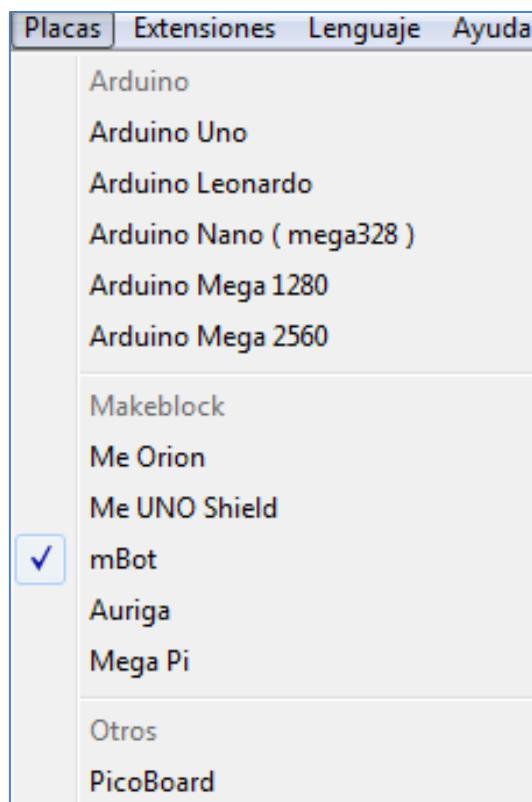
- *Ver código:* Abre una pantalla lateral derecha con el código del programa.
- *Instalar Driver de Arduino:* Instala los drivers de la placa.

2.4. Placas

El software mBlock no sólo es aplicable al robot mBot. Con él podemos programar otras placas electrónicas. Esta pestaña las clasifica en tres bloques: Arduino, Makeblock y Otros.

Dentro del conjunto arduino podemos conectar la placa Arduino Uno, Leonardo y Arduino Mega 1280 y 2560. En la opción de las placas de la casa Mackeblock nos encontramos con la placa mCore (mBot), Orion y Auriga, el shield para Arduino Uno y la Mega Pi. Finalmente, en el bloque “Otros” podemos trabajar con la conocida PicoBoard. Todas ellas son opciones que nos ofrece la pestaña “Placas”.

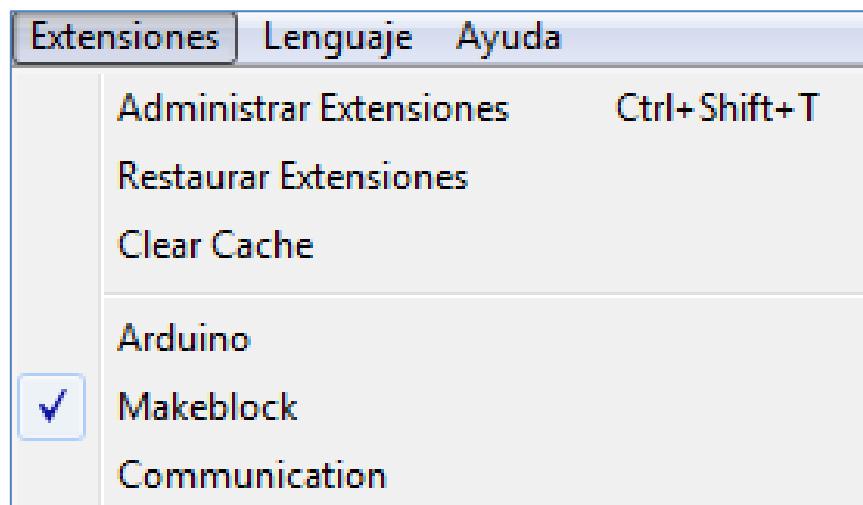
Obviamente, debemos elegir qué opción de las posibles programaremos. Esto se hace desde la pestaña Placas.



Pestaña “Placas” en la Versión 3.3.1 de mBlock

2.5. Extensiones

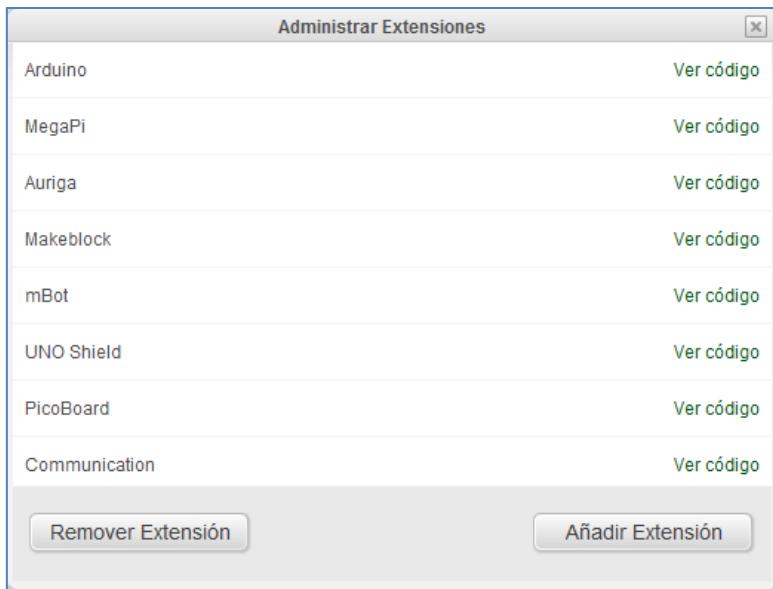
Las opciones de la pestaña “Extensiones” se muestran en la siguiente imagen:



Opciones de la pestaña Extensiones

- *Administrar Extensiones*: Nos muestra las extensiones que tenemos instaladas. Podemos sumar más extensiones o borrarlas. Para hacerlo, debemos seguir los pasos que se explican en el siguiente link:

<http://forum.makeblock.cc/t/how-to-add-an-extension-for-mblock/2280>

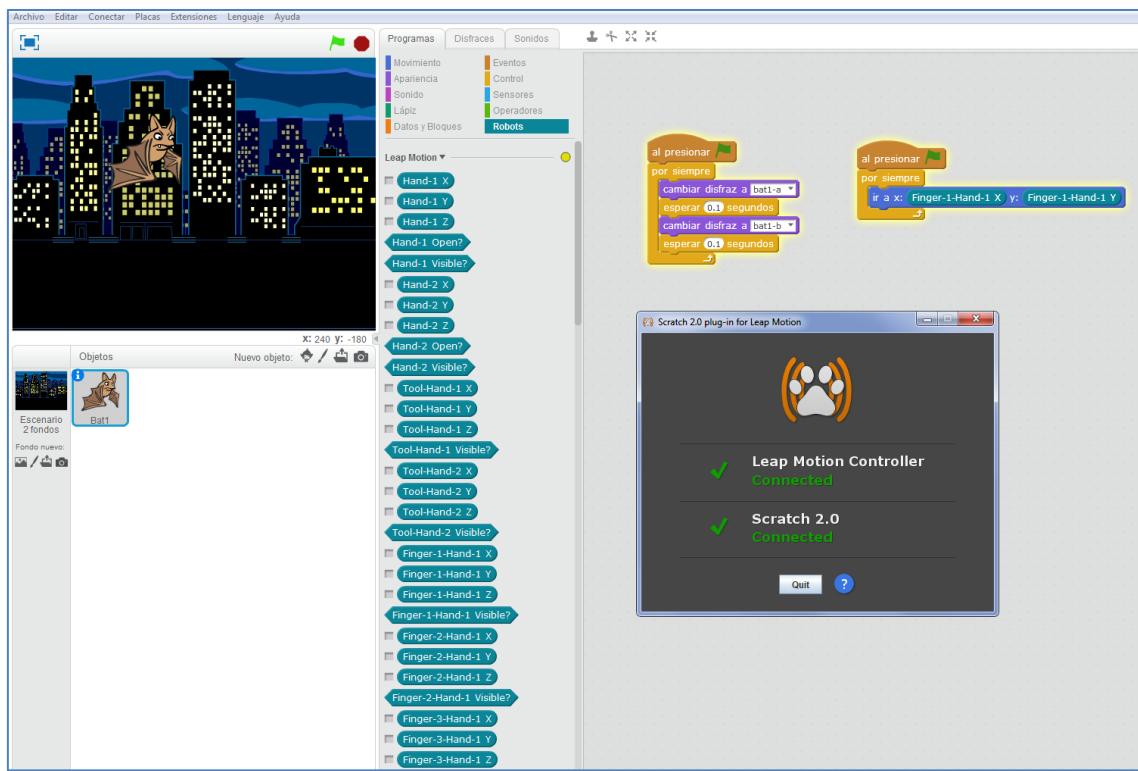


La opción que elija en la extensión afectará a los comandos mostrados en el bloque Robot del programa. Por lo tanto, cada opción mostrará sus comandos específicos en el bloque Robot del mBlock.

En el mercado hay dispositivos que presentan una extensión experimental con el software Scratch2.0 y que pueden trabajar con mBlock. Es el caso del dispositivo Leap Motion. Este dispositivo de tres pequeños infrarrojos requiere de un plugin, disponible en su casa App Home, que hace posible la detección del mismo en un entorno Scratch. Tras lanzarlo, sólo debemos añadir el archivo *LeapMotion.json* a las extensiones del software mBlock y conseguiremos que el programa reconozca los comandos del Leap Motion y podamos programarlo. En las siguientes imágenes podemos ver que se ha incluido la extensión del Leap Motion:

The screenshots demonstrate the configuration of the Leap Motion extension in mBlock. The first screenshot shows the 'Extensions' menu with 'Leap Motion' checked. The second screenshot shows the 'Administrador Extensiones' window with 'Leap Motion' highlighted. The third screenshot shows the expanded 'Robot' block palette, which includes specific blocks for controlling the Leap Motion device.

Divirtiéndome con mBot: Guía de manejo y programación



Ejemplo con Leap Motion

- *Arduino*: Incluye bloques compatibles con la plataforma Arduino



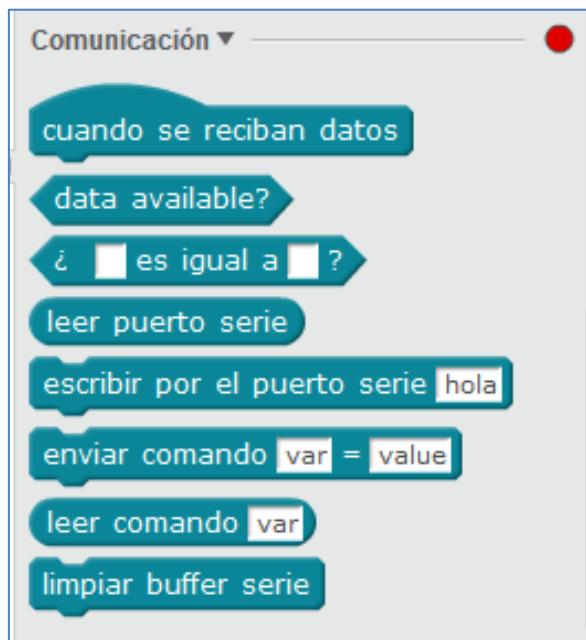
- *Makeblock*: *Me Orion*, *Mega Pi*, *Auriga*, *Me Uno Shield* y *mBot*; placas base específicas de Makeblock

Divirtiéndome con mBot: Guía de manejo y programación

- *PicoBoard:* Control y testeo de la placa PicoBoard desde scratch (deslizador, botón, sensor de luz, sensor de sonido, etc)



- *Comunicación:* Proporciona la función de la comunicación LAN.



- etc

2.6. Lenguaje

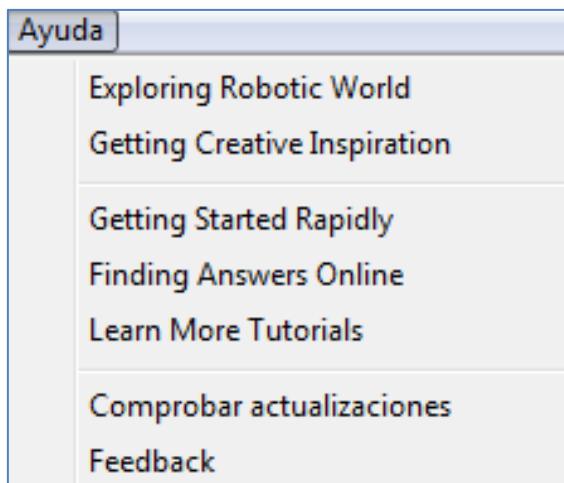
En esta pestaña podemos elegir el idioma y el tamaño de la letra o fuente del mBlock. En la siguiente imagen se ha seleccionado el español y tamaño 12:



Opciones de la pestaña Lenguaje

2.7. Ayuda

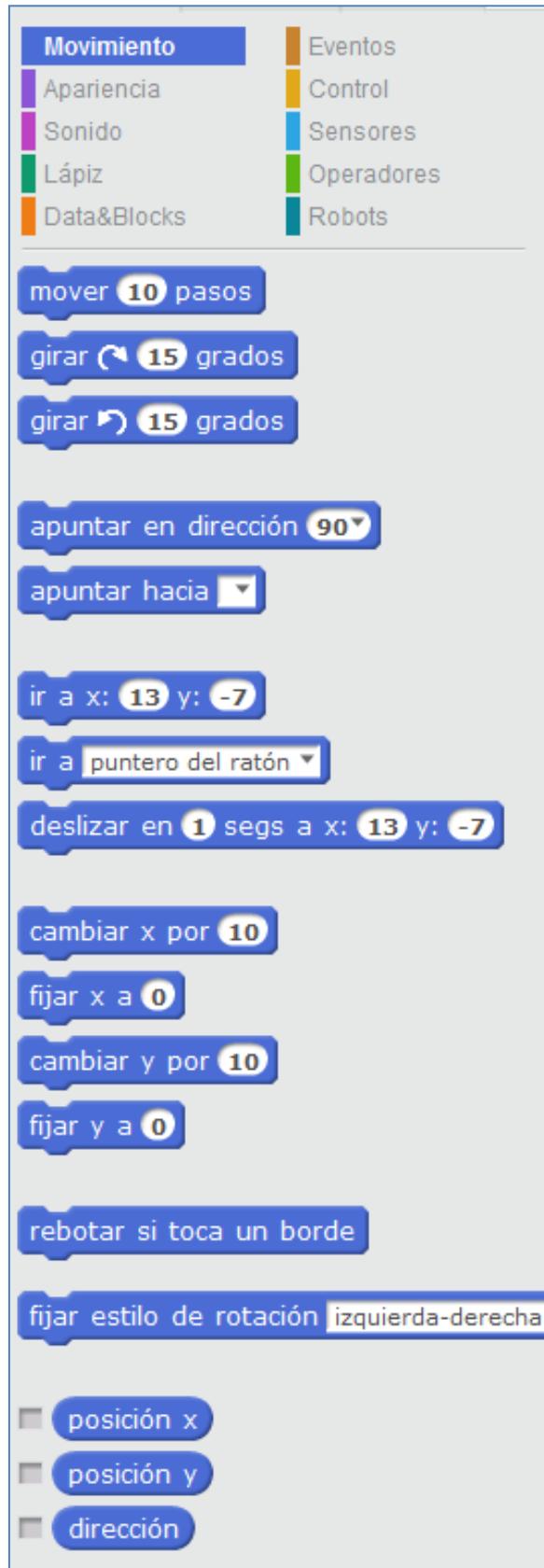
Con esta pestaña podemos informar de algún error o bug al equipo que actualiza y perfecciona el software mBlock. También podemos acceder al foro y tutoriales, así como, comprobar si hay actualizaciones del software.



Opciones de la pestaña Ayuda

3. Comandos de los diferentes bloques en mBlock

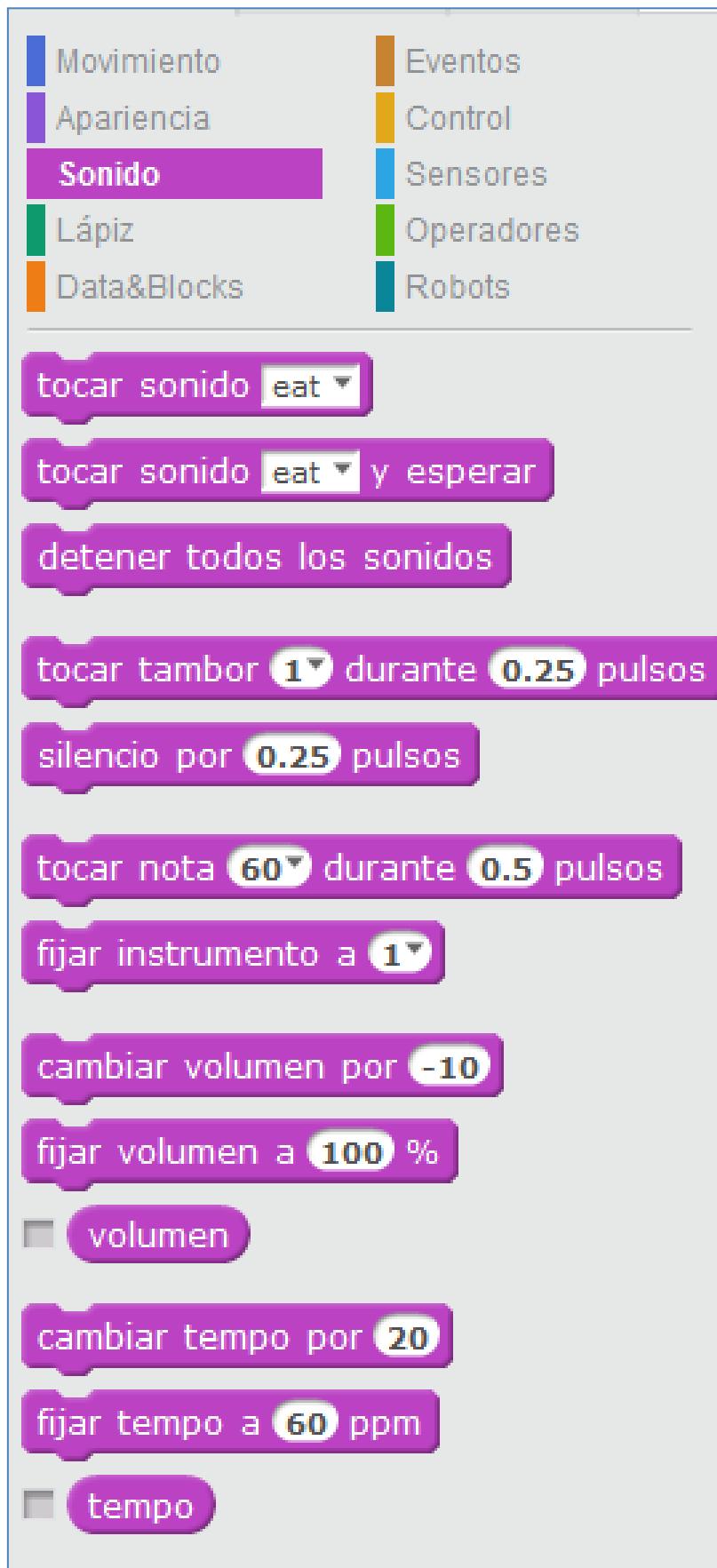
3.1. Movimiento



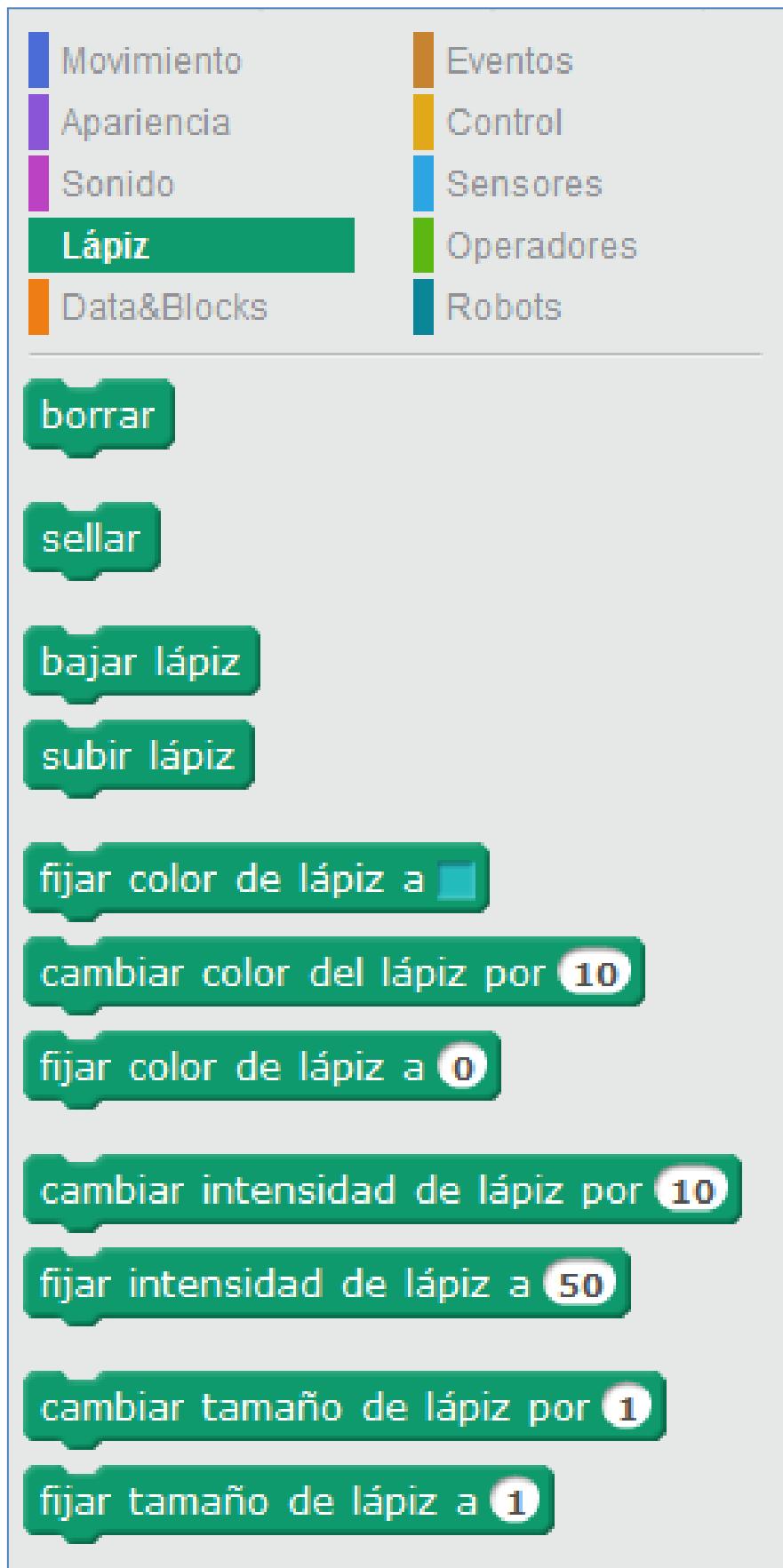
3.2. Apariencia



3.3. Sonido



3.4. Lápiz



3.5. Datos y Bloques

Puedes crear variables, listas de datos y bloques dentro del programa



Variables:



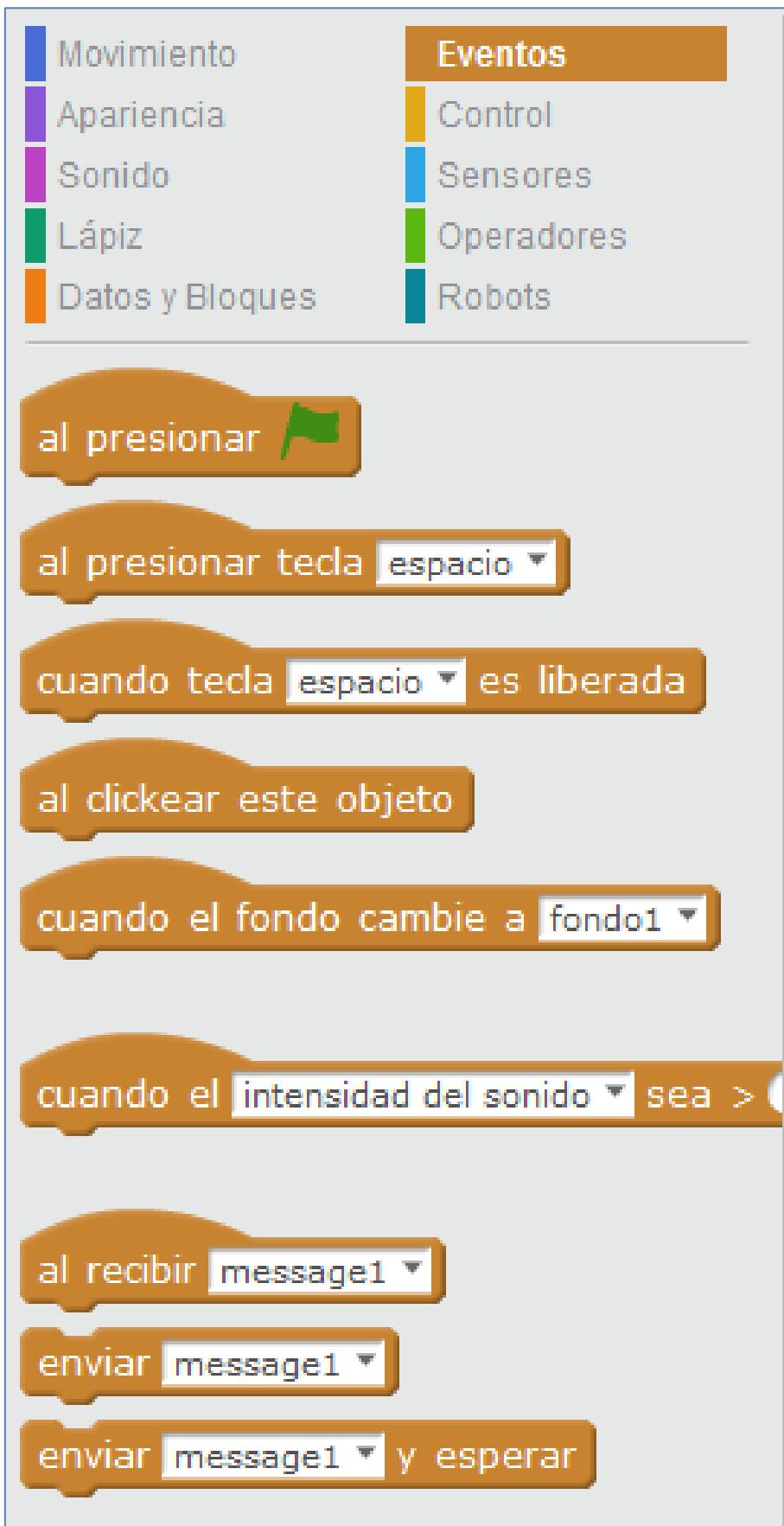
Listas:



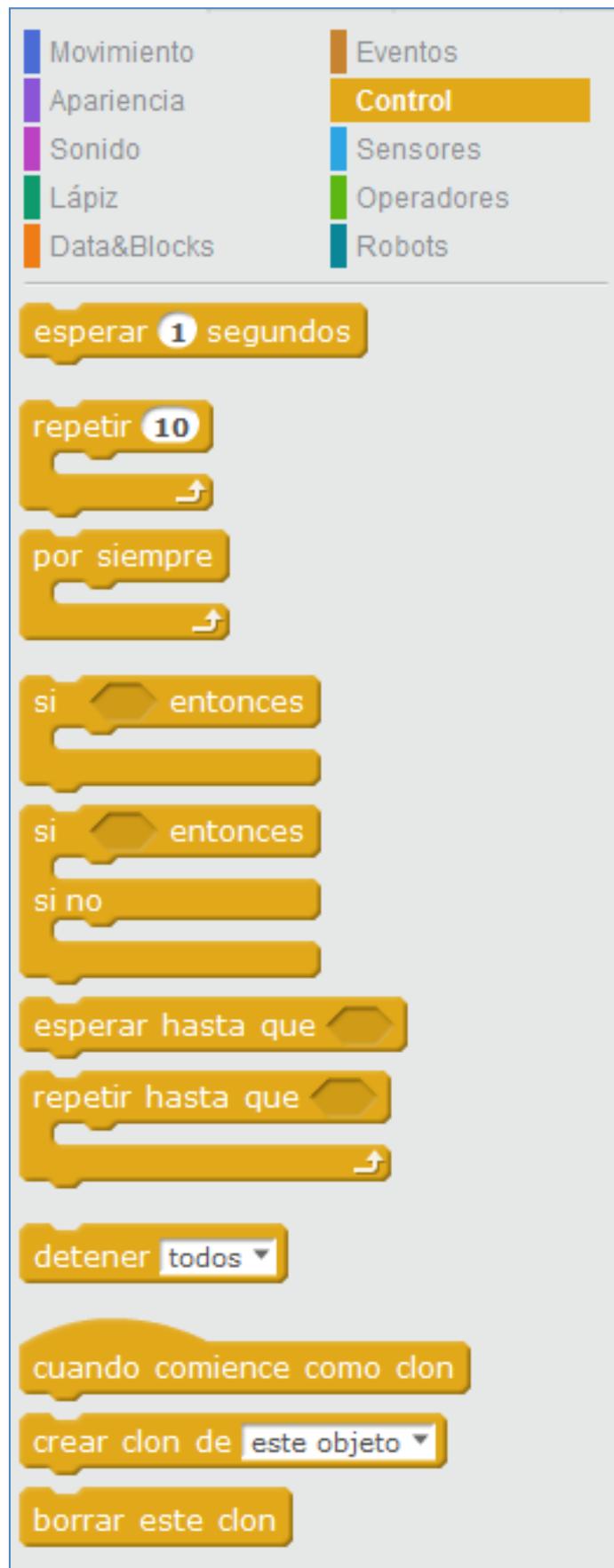
Bloques: primero creamos el bloque y después ya podemos definir qué realiza y lanzarlo cuando queramos.



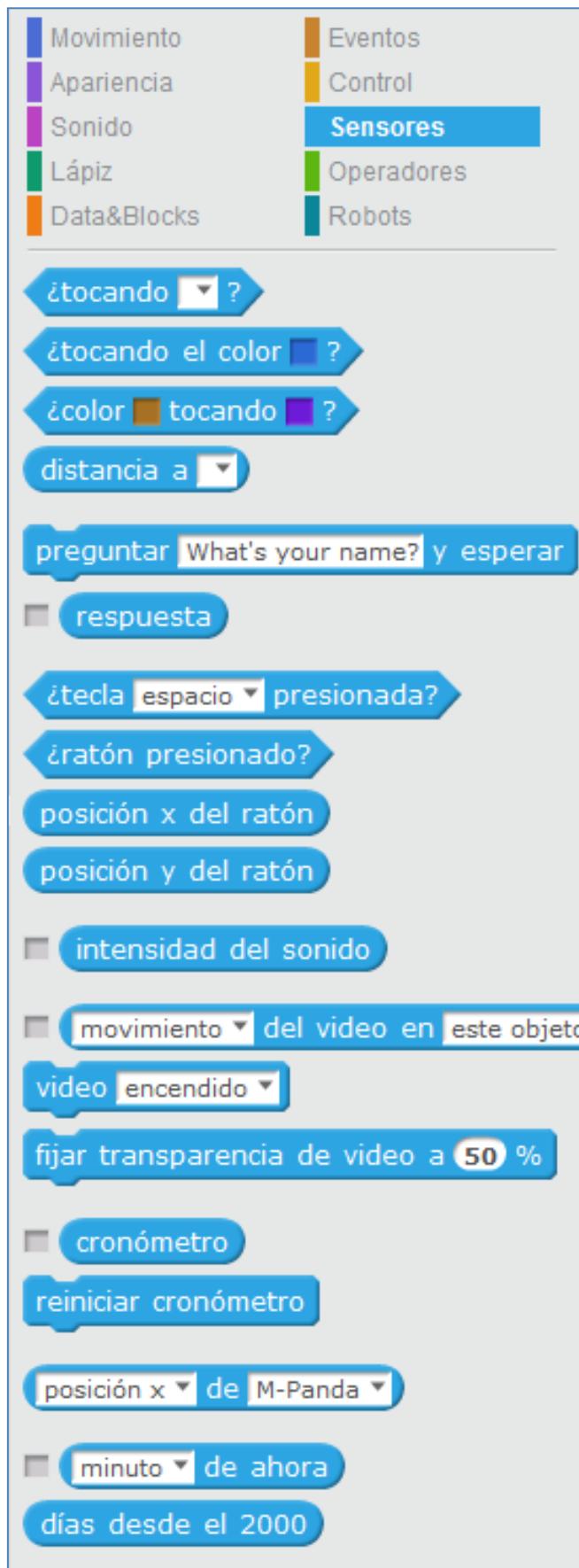
3.6. Eventos



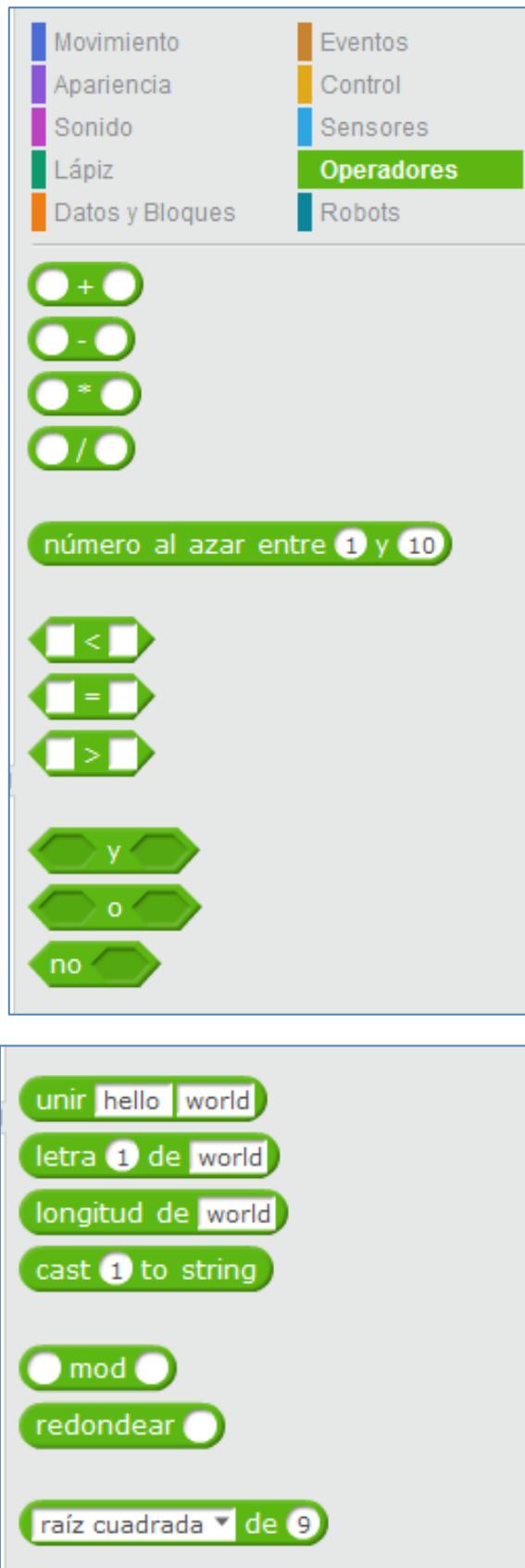
3.7. Control



3.8. Sensores



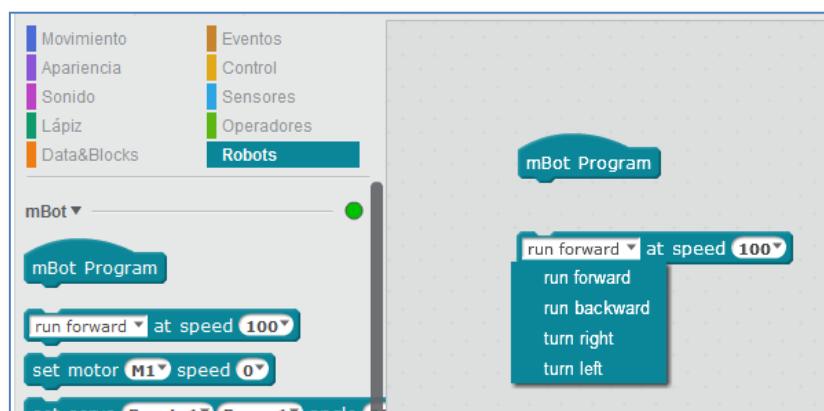
3.9. Operadores



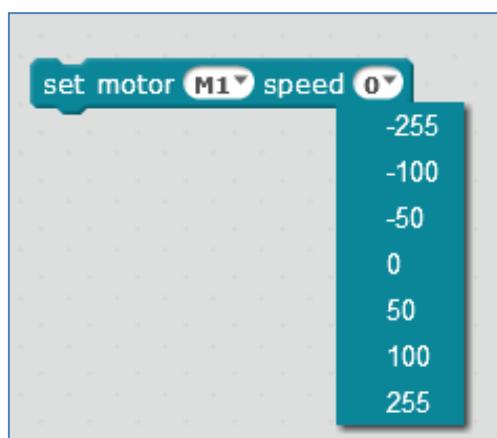
3.10. Robots

Bloque específico de comandos para las placas que se conectan con el software mBlock. Hablaremos de los comandos para el mBot, los cuales la casa Makeblock amplía y actualiza a menudo de cara a nuevos sensores.

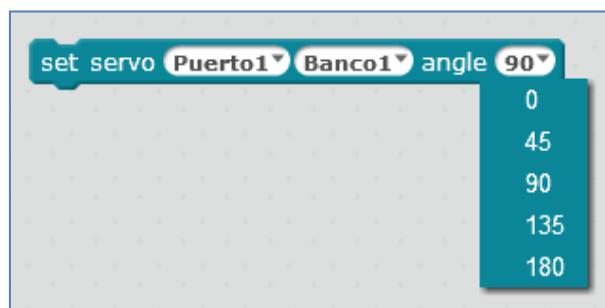
En este bloque nos interesa especialmente el comando "mBot Program". De hecho, un preámbulo para poder grabar un programa en la placa será cambiar el bloque de la banderita verde por la sentencia "mBot Program" y, de ese modo, todo lo que cuelgue de este comando, se ejecutará siempre que encendamos el mBot. Para grabarlo en la placa, debemos tener conectado el mBot mediante el cable USB, hacer clic con el botón derecho sobre este bloque (o *Editar > Modo Arduino*) y seleccionar "Upload to Arduino".



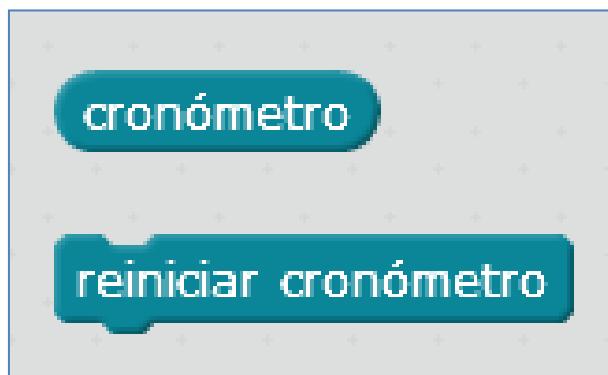
1. Dos motores M1 y M2 con diferentes velocidades:



2. Cuatro puertos con 2 Bancos y diferentes ángulos



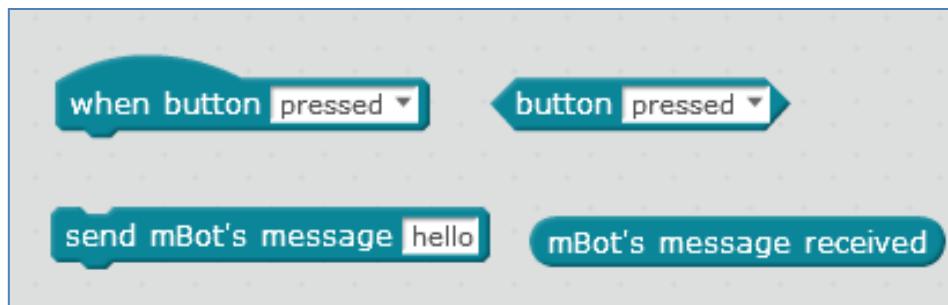
3. Cronómetro



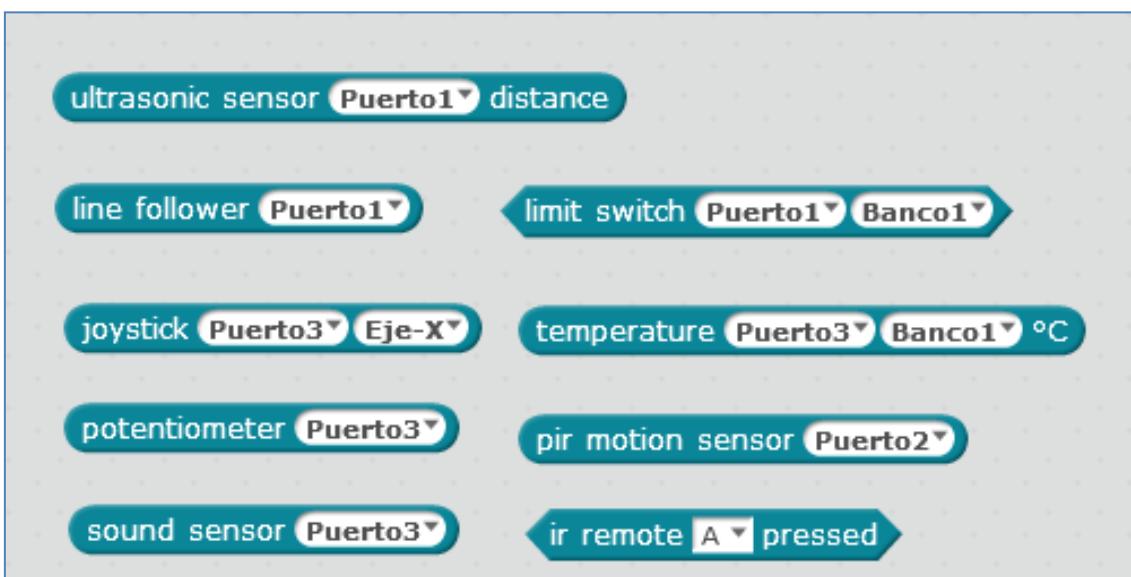
4. LED RGB



5. Botón y mensajes



6. Sensores y Control Remoto



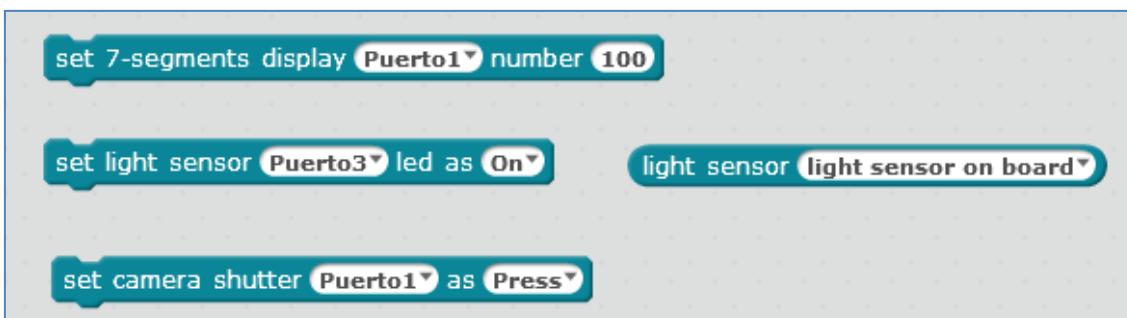
En la imagen anterior vemos que disponemos de comandos para los siguientes sensores: Sensor de ultrasonidos, seguidor de línea, sensor de temperatura, sensor de

Divirtiéndome con mBot: Guía de manejo y programación

sonido y sensor PIR. También podemos programar el pulsador, un potenciómetro, un joystick e incluso, programar el robot de forma remota con su mando especificando nosotros qué debe hacer cada tecla del mando a distancia.

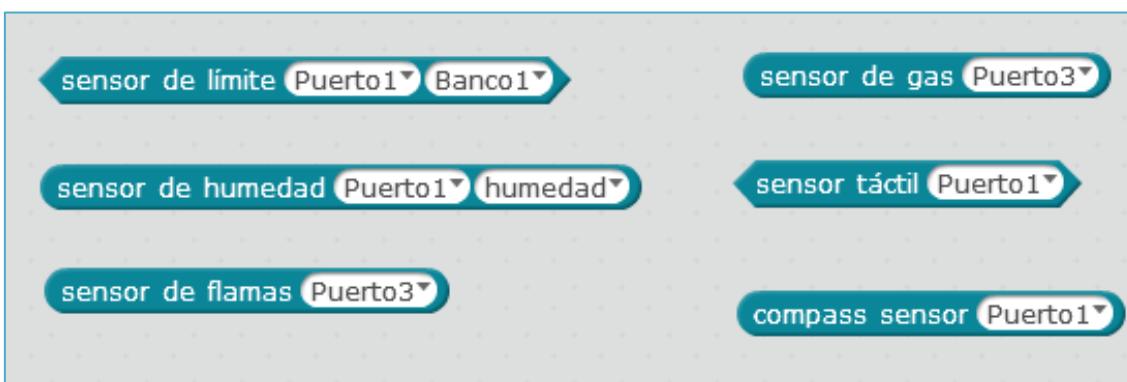
Los sensores PIR se utilizan para detectar el movimiento de los animales / humanos a una distancia aproximada de 6m. Por lo tanto, si alguien se mueve en ese rango, las salidas de los sensores serán HIGH sobre su pin SIG. Con un potenciómetro soldado en su placa de circuito, podemos ajustar su rango de detección para nuestras necesidades. Algunas ideas para su aplicación pueden ser: detección de movimiento, sistema ladrón-vigilancia, interruptor de automatización.

Nota: Es posible que debamos esperar 10 segundos hasta que el sensor comience a funcionar.

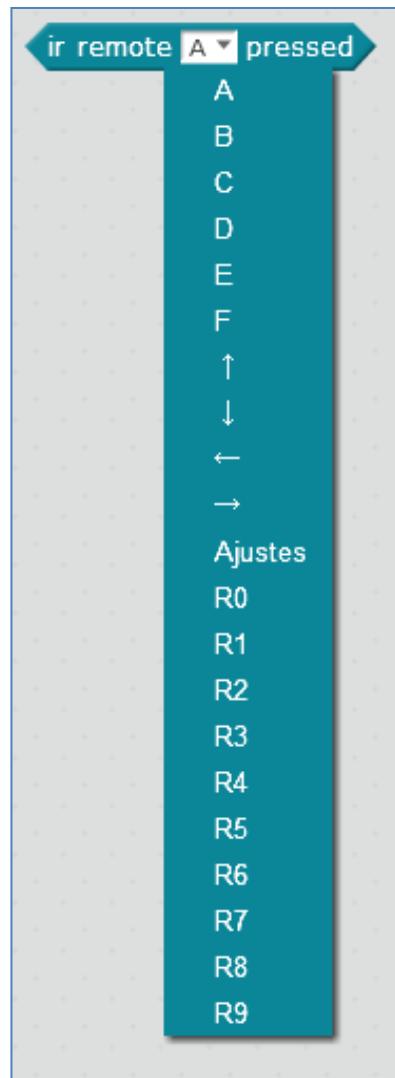


También podemos programar el sensor de luz, el sensor de video y el display 7 segmentos. Relativo al sensor de luz, que está situado al lado de los de IR, detecta la luz ambiente, pero, según la casa Makeblock, han experimentado que detecta la infrarroja, por lo que es sensible a la calefacción falseando su medida.

La casa Makeblock está incluyendo más comandos para otros sensores que va sacando al mercado. En la versión 3.3.1, que actualmente es la última ya estable, podemos utilizar los siguientes comandos para los sensores de gas, llamas, táctil, humedad, etc:



De forma remota, podemos controlar nuestro robot, programando a nuestro gusto cada una de las teclas del mando a distancia:



7. Música



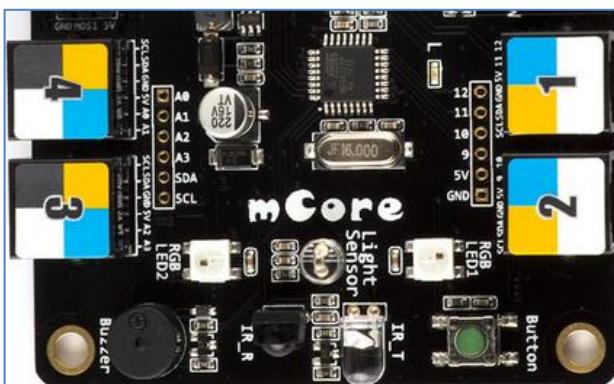
8. Mostrar



4. Ejemplos de programación

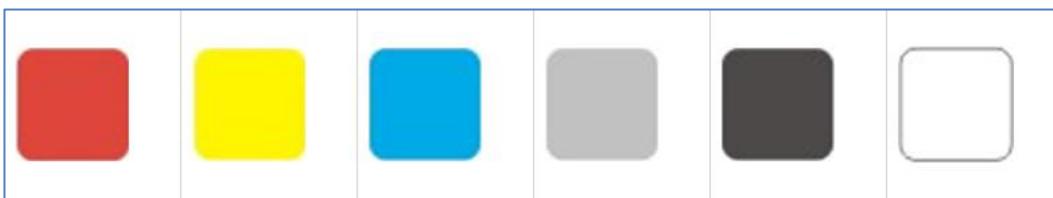
La placa mCore del mBot puede programarse utilizando diferentes lenguajes de programación. De hecho, no es más que una Arduino Uno y, por lo tanto, puede ser programada con Processing. Esto hace que podamos sacarle más rendimiento a la placa ya que, en teoría, tenemos un dos en uno. Aunque en este apartado usaremos muchas veces scratch para programar diferentes módulos, esos mismos componentes electrónicos podrían ser programados a través del IDE de Arduino y, a veces, usaremos esta posibilidad.

Los módulos que pretendemos conectar a la placa presentan y vienen clasificados por su color ID. Ese color debe corresponder con el color del puerto al cual pretendemos conectarlo. Por ejemplo, en la siguiente imagen vemos que el puerto 2 dispone de tres colores: amarillo, azul y blanco. Pues bien, a él podremos conectar cualquier módulo cuyo RJ25 disponga de, como mínimo, alguno de esos colores. Si el ID del módulo es negro, no podríamos conectarlo al puerto 2 (ni al 1), pero si al 3 o 4.



4 puertos de mCore

Los colores ID que podemos encontrarnos en los puertos de las diferentes placas de Makeblock son: Rojo (motores), Amarillo (interface digital), Azul (interface digital dual), Gris (Puerto serie, bluetooth), Negro (interface analógica y dual) y Blanco (Puerto I²C).



4.1. Módulo de ultrasonidos

Un módulo de ultrasonidos nos proporciona un dato numérico que se corresponde con la distancia entre el sensor y cualquier objeto que está en frente de él. Por lo tanto, se utiliza para medir distancias, logrando detectar objetos que se encuentran a 3 o 4cm del sensor. Su color ID es amarillo y eso significa que puedo conectarlo a cualquiera de los cuatro puertos de una placa mCore del mBot.

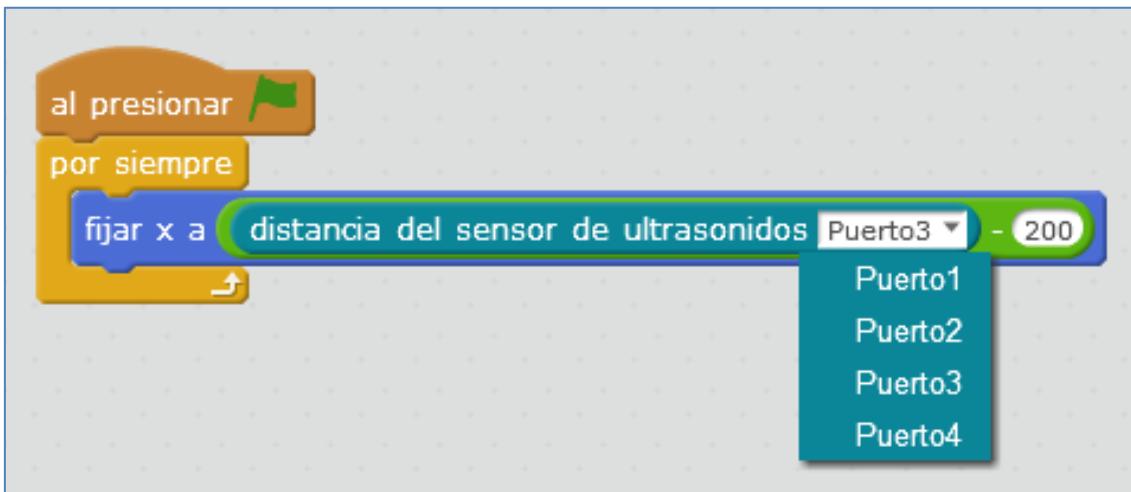


Módulo ultrasonidos

Supongamos que lo conectamos al puerto 1 de la placa mCore del robot. El siguiente script, nos mostrará la distancia que mide en el escenario del mBlock:



También podemos controlar el movimiento del ratón con el sensor de ultrasonidos. Una forma sería con el siguiente script:



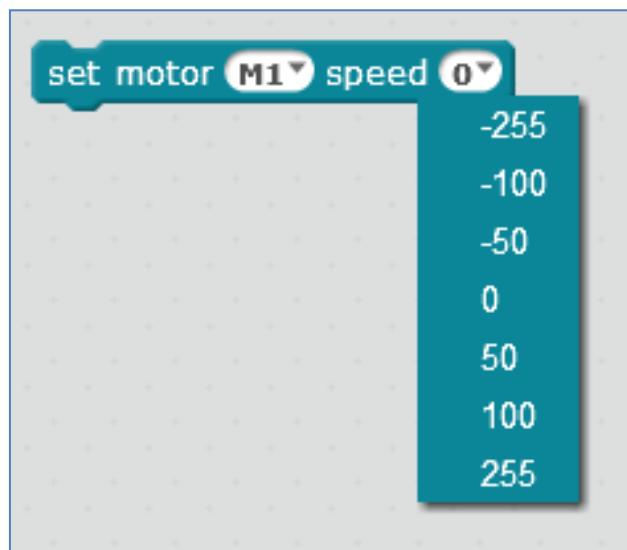
NOTA: Ojo, hay que ver en qué puerto de los 4 tenemos conectado nuestro sensor de ultrasonidos. El puerto típico para el mBot suele ser el puerto 3.

4.2. Motores

El robot mBot se compone de dos motores de continua de 6V a 200rpm. En la placa, estos motores se denotan por M1 y M2, pudiendo variar su velocidad, a la hora de programarlos, desde 0 a 255. El valor mínimo, 0, es la velocidad más lenta y 255 es la más rápida. Los valores negativos harán que gire en el otro sentido (rotación inversa).

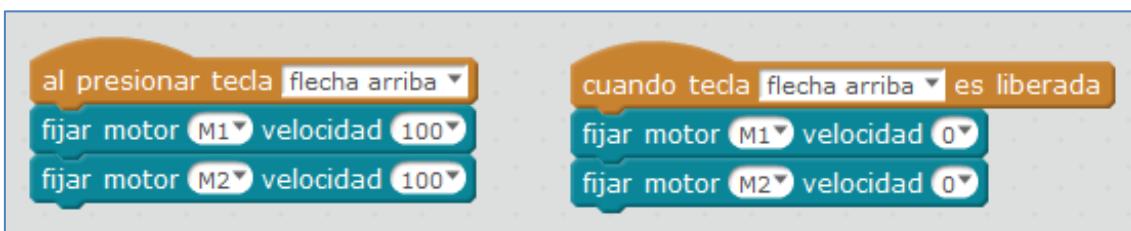


Motor DC 6V/200rpm



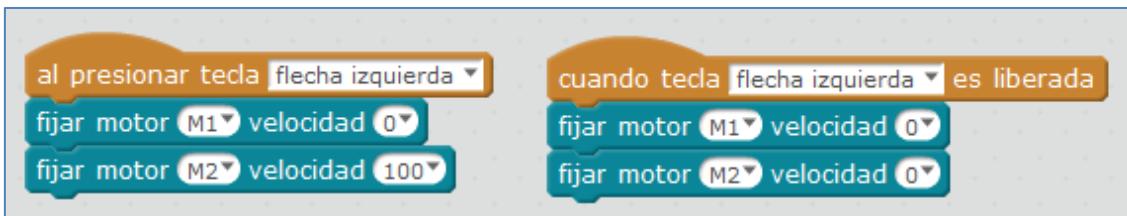
Podemos programar, por ejemplo, el movimiento de los motores a través del teclado del ordenador, al presionar o liberar una tecla. Debemos decidir qué ocurrirá cuando presiono la tecla “X” y qué ocurrirá cuando dejo de presionar la tecla “X”.

Por ejemplo: Cuando pulse la tecla “flecha arriba” los dos motores comienzan a funcionar hacia adelante, con una velocidad de 100, pero, cuando deje de presionar esa tecla, se pararán.

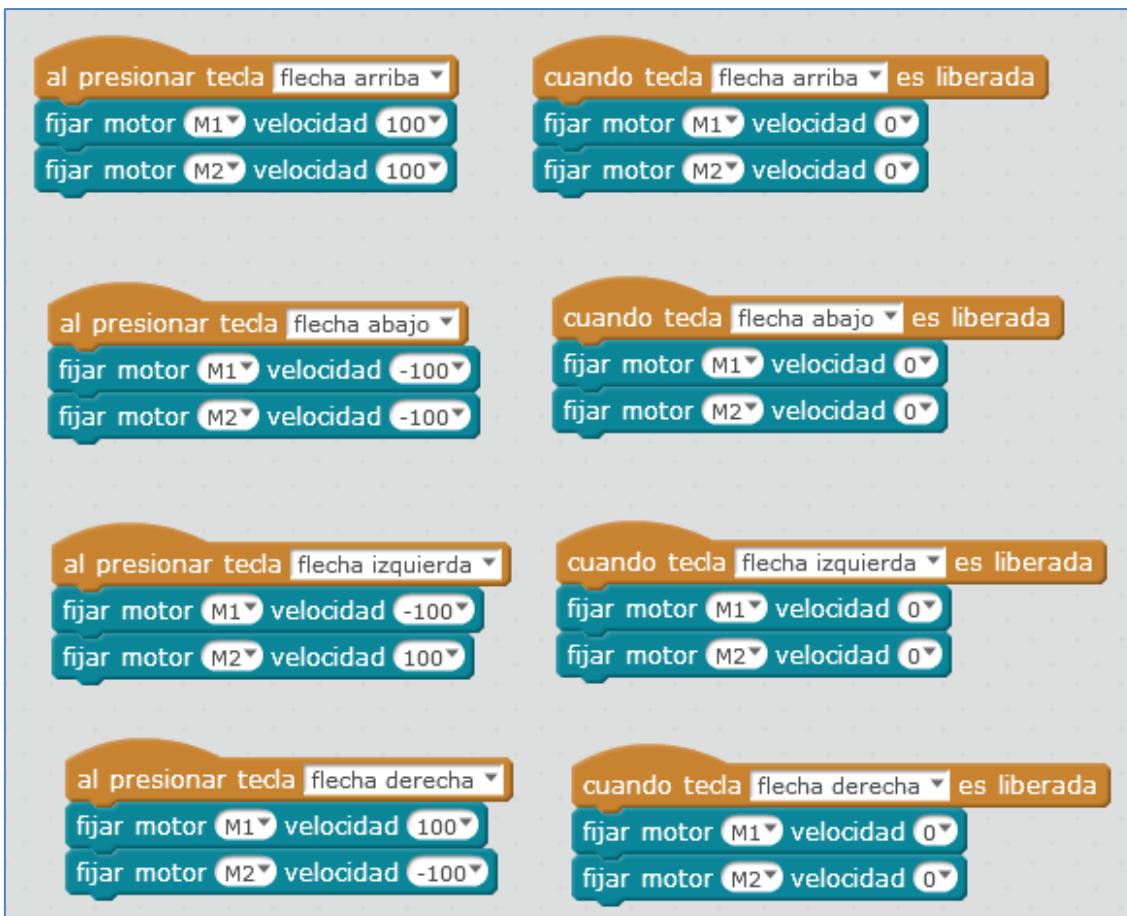


Para que gire hacia la izquierda, debemos poner el motor izquierdo a 0 (M1 o M2, dependiendo de cómo se conectarán²) y el derecho a una velocidad (M1 o M2) por ejemplo de 100. Al dejar de presionar, se debe parar. En el ejemplo que vemos en la siguiente figura, M1 es el motor izquierdo y M2 el derecho:

² En teoría, M1 es el izquierdo y M2 el derecho, pero podemos cambiarlos



Se puede mejorar el script anterior haciendo que M1 gire en el sentido contrario, con velocidad -100. El movimiento de control completo en un coche sería:



Script para el movimiento de los motores con el teclado

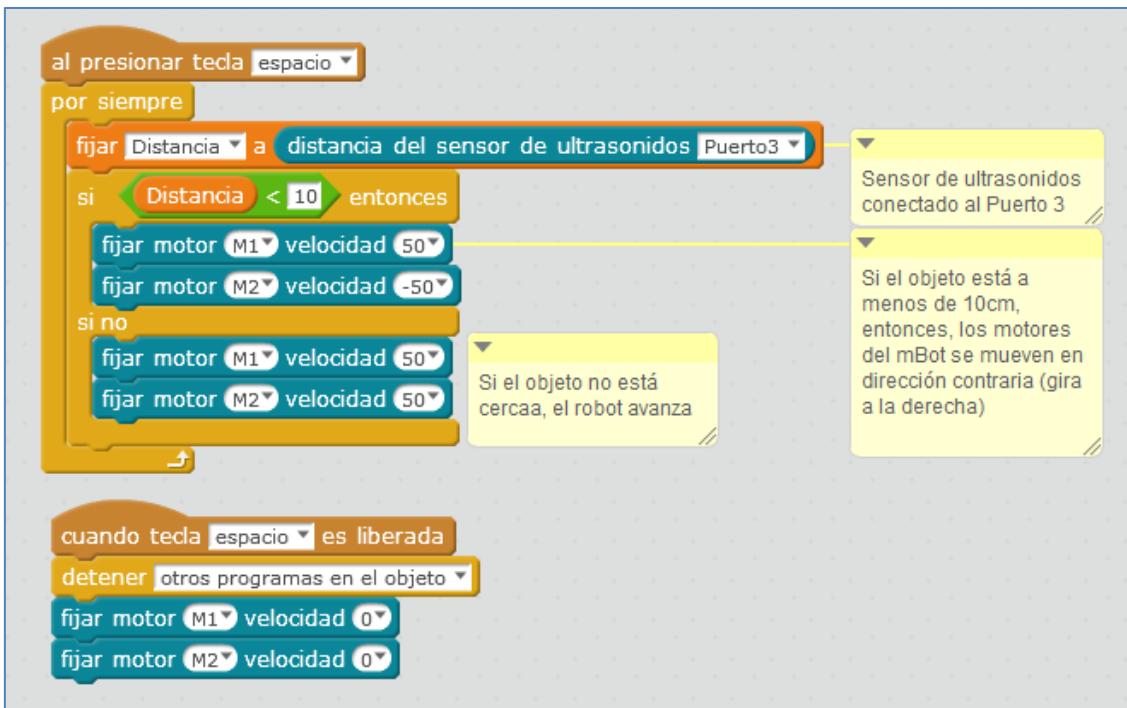
Con el tiempo y los avatares que vaya pasando el robot, quizás debamos reemplazar alguna pieza de un motor del mBot. El siguiente tutorial nos muestra cómo reemplazar esas posibles piezas rotas o defectuosas del motor:

<http://learn.makeblock.cc/mbot-motor-shaft-broken/>

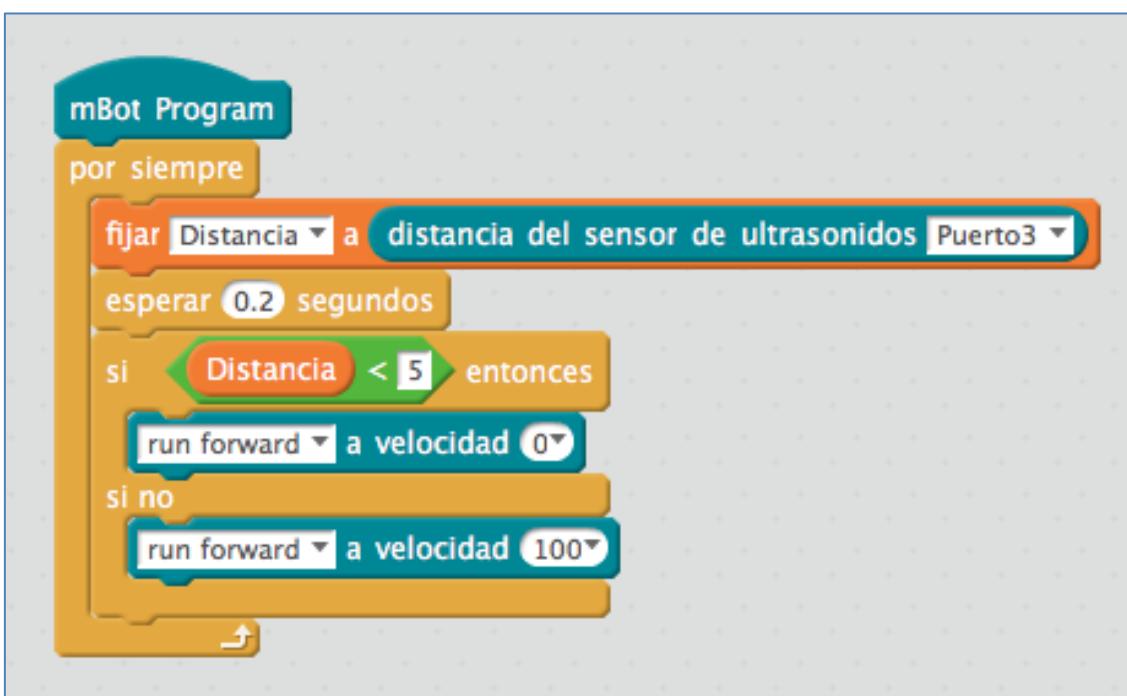
4.3. Motor y detección ultrasónica

Podemos unir los dos elementos que ya conocemos, sensor de ultrasonidos y motores, e intentar programar el siguiente reto:

Si el objeto se encuentra a menos de 10 cm del robot, debe girar. En caso contrario, debe avanzar.



En la imagen anterior, hemos trabajado sin cargar el programa en el robot, es decir, conectándolo por bluetooth o a través del módulo 2.4G. A veces, con el sensor de ultrasonidos, puede ocurrir que al cargar el programa, este no funcione correctamente. En ese caso, quizás se deba añadir un tiempo de espera en cada lectura para que logre captar la medida del sensor. En el siguiente ejemplo, he decidido 200ms en cada lectura:



4.4. Diodos RGB de la placa

La placa mCore dispone de dos RGB incrustados en la misma y que se denominan "led abordo". En el siguiente ejemplo, vamos a modificar el programa anterior

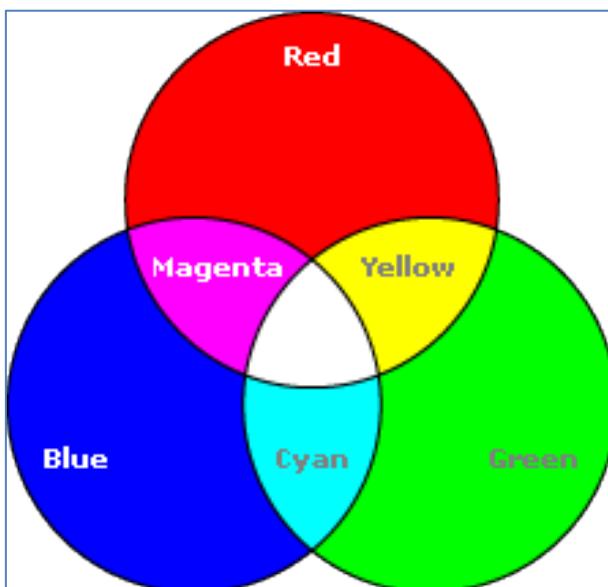
Divirtiéndome con mBot: Guía de manejo y programación

añadiendo: el encendido de los LEDs en rojo (precaución), verde (adelante) y apagado de los LEDs.



Los valores numéricos de cada color del LED RGB están comprendidos entre 0 y 255. Combinando los tres colores podemos crear cualquier color del arcoíris. Para conseguir el blanco, los tres colores (rojo, verde y azul), deben estar a 255.

Como se puede ver en la siguiente imagen, el amarillo lo conseguiríamos combinando el rojo con el verde (poniendo, por ejemplo, el rojo a 125, el verde a 125 y el azul a 0):



4.5. Detector de Línea

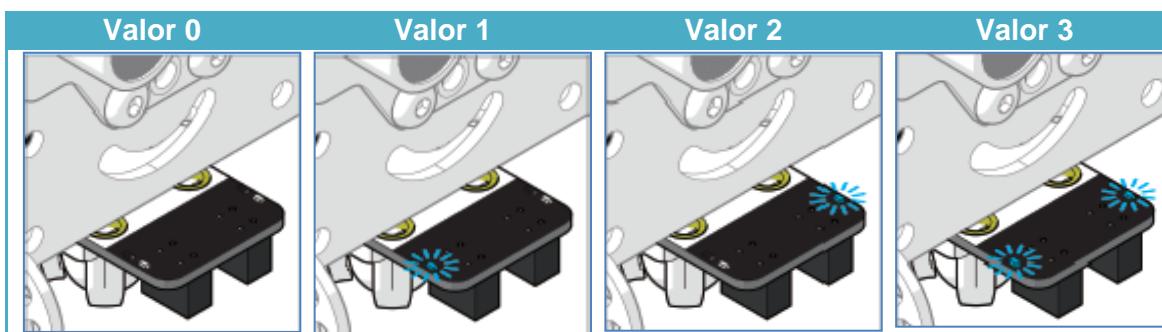
Este módulo sensor hace posible que nuestro robot se mueva a lo largo de una línea negra sobre el suelo. En el kit mBot se sitúa en su parte frontal funcionando *por reflexión*. En nuestro mBot, como el color de su ID es azul, lo podemos conectar a los puertos 1, 2, 3 y 4. En los ejemplos que siguen, usaremos el Puerto 2 del mBot.

El Módulo V2.2 está diseñado para detectar una línea negra. Cada módulo tiene dos partes: un *LED IR* que emite y un *fototransistor IR* que recibe la señal. Gracias a ellos podrás medir si estás sobre una línea negra y seguirla sobre un fondo blanco o viceversa. Los dos sensores de un módulo indican si están en "algo" negro o en "algo" blanco, y por lo tanto, se podrían programar para ser usados como seguidores de línea blanca. Ese "algo" blanco o "algo" negro no es, literalmente, el color blanco y negro, y me explico. El sensor, a todo color brillante lo detecta como blanco, así como, un color mate oscuro, lo detectará como negro.

A la hora de programar el sensor, se nos pueden dar 4 posibilidades. Eventualidades que el sensor de la casa Makeblock, tiene, implícitamente definidas con los siguientes valores numéricos: 0, 1, 2 y 3. Estos se describen como sigue:

- 0 es el valor numérico que representa que el sensor está totalmente encima de la línea negra.
- 3 es el valor numérico que representa que el sensor está totalmente encima de la línea blanca.
- 1 y 2 son los valores numéricos que se asocian cuando un lado del sensor está fuera de la línea negra, mientras el otro lado está en el color negro.

Las 4 posiciones pueden verse gráficamente en la siguiente imagen:

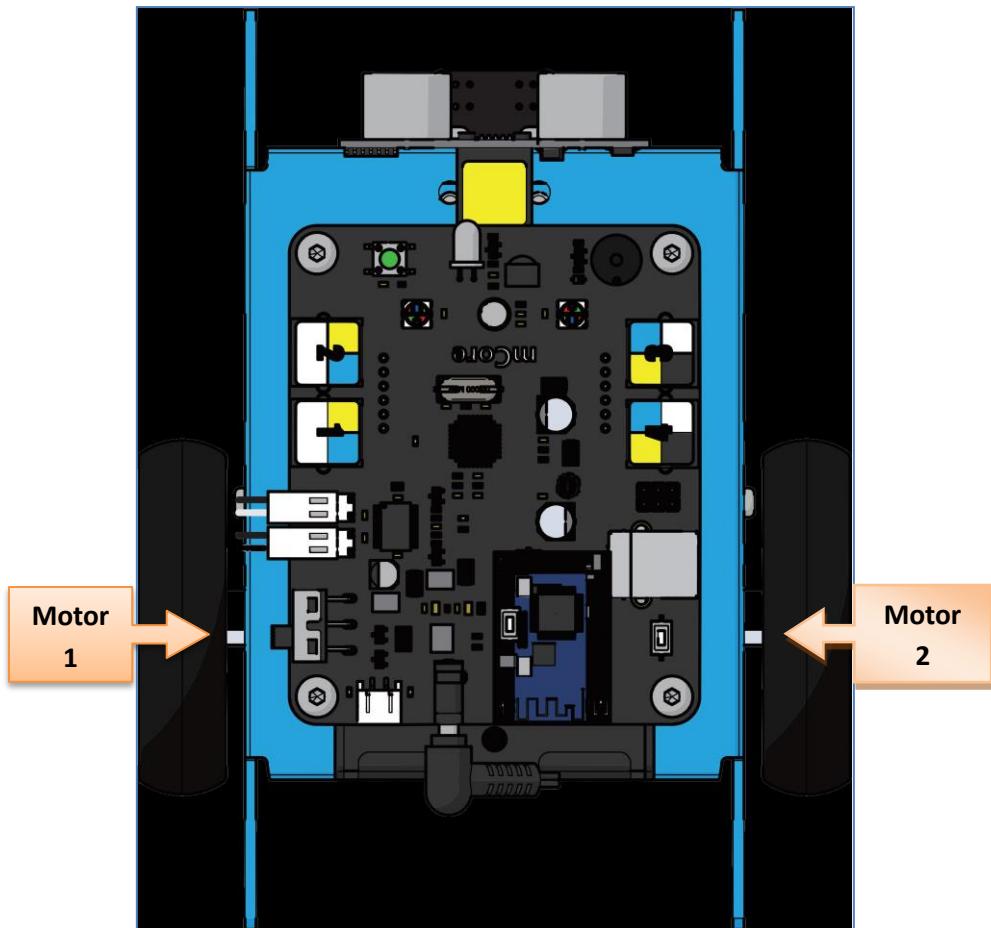


- A) Seguidor de línea negra: En el siguiente ejemplo, he querido que se accione al pulsar la tecla espacio. Tras presionar la tecla espacio, mBot ejecuta el siguiente programa: testea si mBot ve completamente la línea oscura y, si es así, se mueve recto. De lo contrario, testea si mBot está a la izquierda (a la derecha) y, si lo está, gira a la derecha (izquierda) y hace que mBot regrese a la línea de negra. Si mBot está lejos de la línea de negra, vuelve a la línea de color negro. Repite el ciclo hasta que deje de presionar la barra espaciadora.

En su programación, crearemos dos bloques (derecha e izquierda) con entradas numéricas que denoto por la variable "Velocidad":

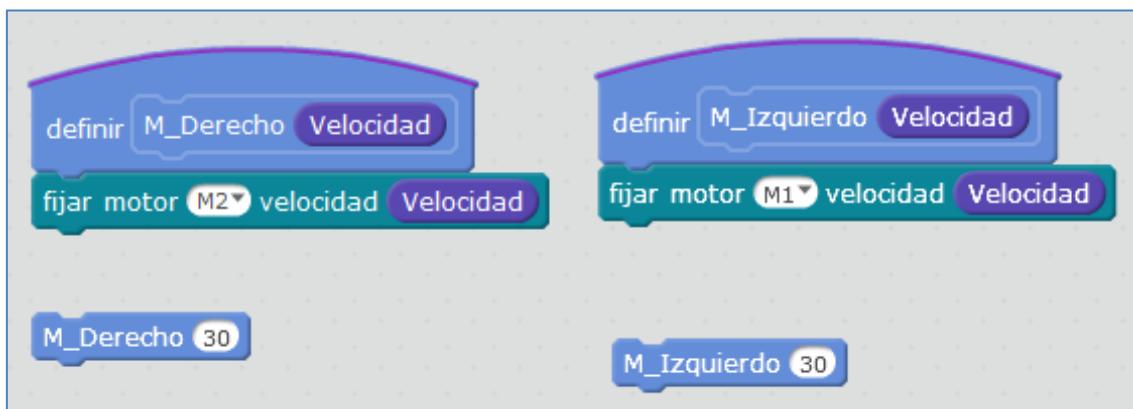
Divirtiéndome con mBot: Guía de manejo y programación

Viéndolo frontalmente: **Supongamos** (debemos comprobar las conexiones de los motores de nuestro mBot) que la rueda izquierda del mBot corresponde al motor 1 y la rueda derecha corresponde al motor 2, tal y como se muestra en la siguiente figura:



Robot mBot

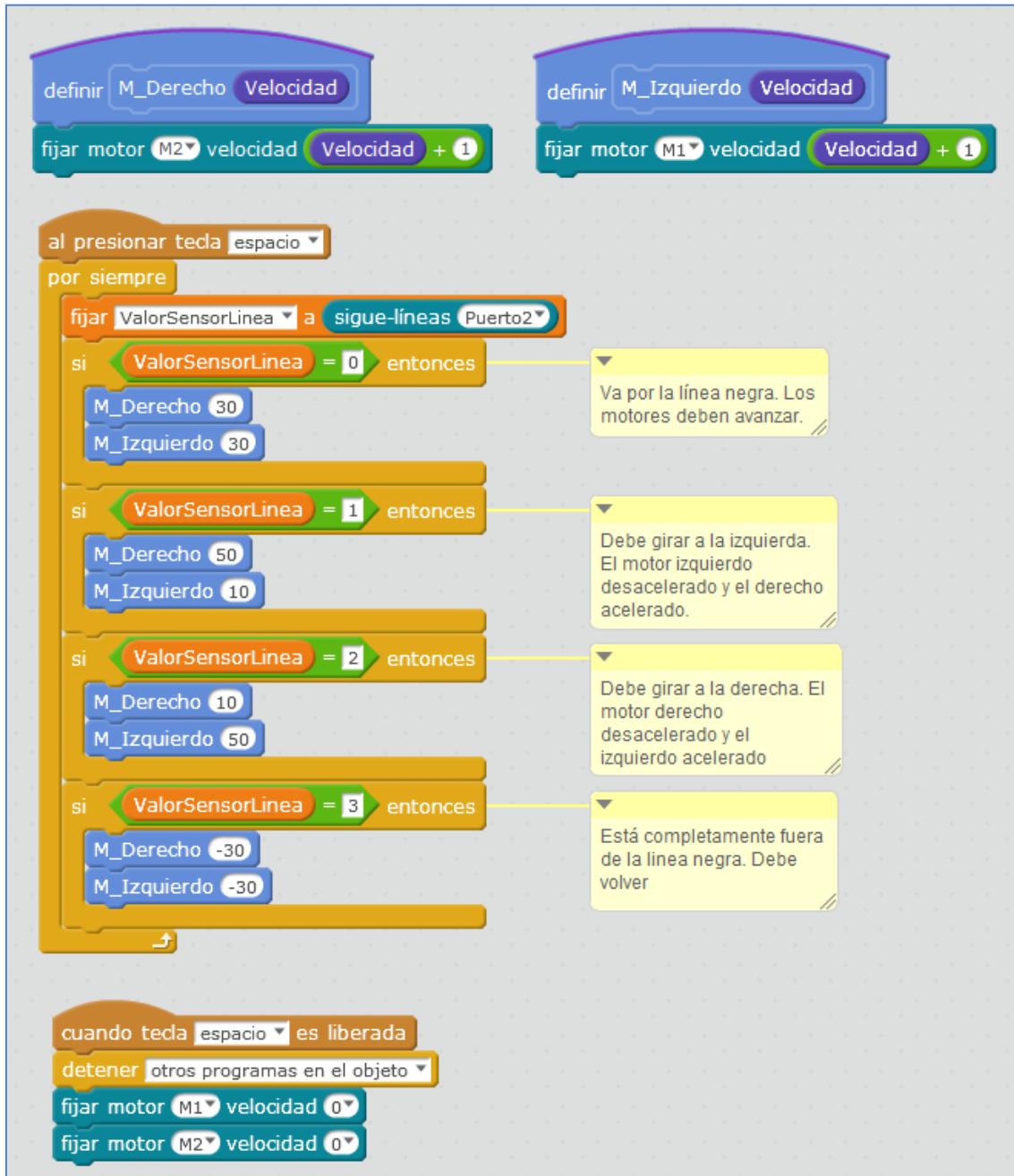
En estas condiciones, cuando ejecutamos el Bloque "M_Izquierdo", se ejecutan sus comandos. El valor 30 (-50, 0, 50, 100, 250³) en el bloque "M_Izquierdo" y "M_Derecho", reemplazará al valor "Velocidad". En este ejemplo, las velocidades de rotación del motor 2 y del motor 1 serán de 30:



³ En teoría, no debemos sobrepasar estas velocidades para no forzar los motores DC

Divirtiéndome con mBot: Guía de manejo y programación

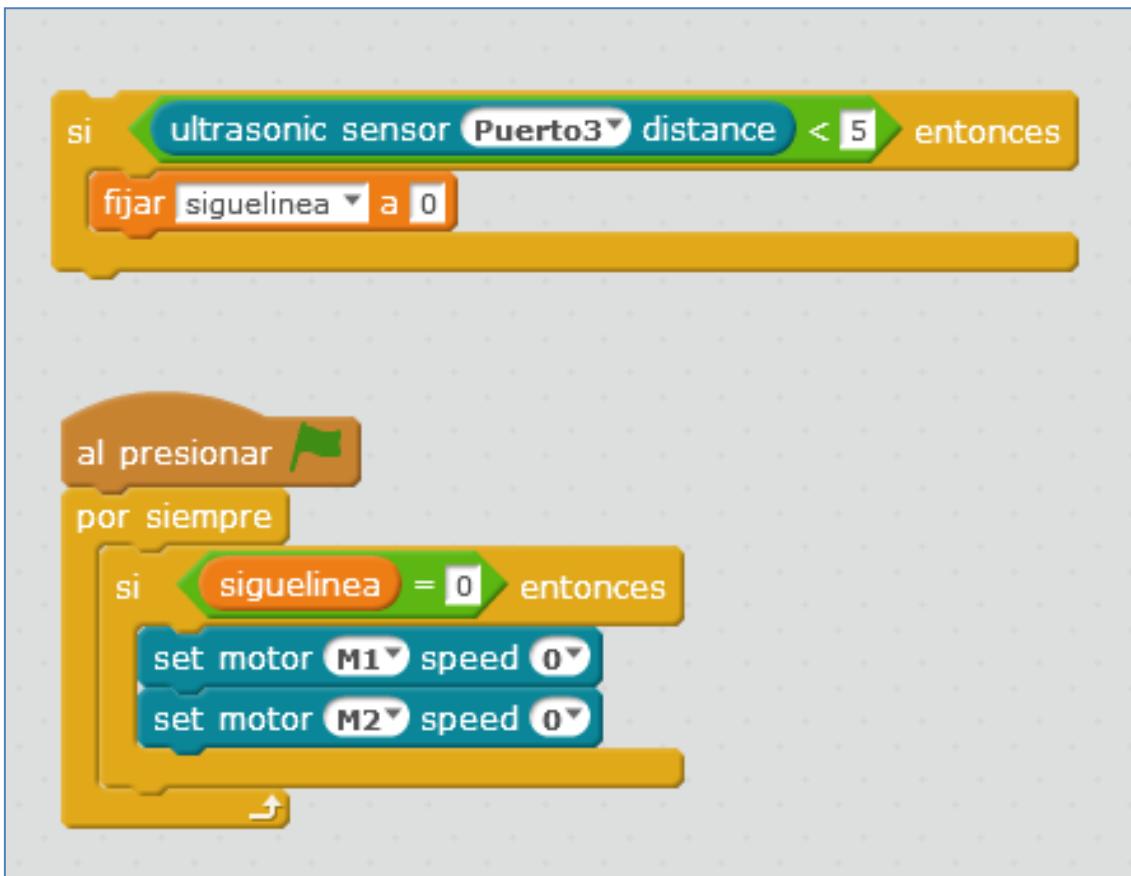
Finalmente, el seguidor de línea mostraría el siguiente script: Al pulsar la barra espaciadora (mantenla pulsada), mBot comienza a juzgar el estado de los sensores de patrullaje de la línea (0, 1, 2, 3). La rotación del motor depende de la desviación, que facilita el ajuste de la dirección de mBot. Podemos incluso incluir comandos de operación, como se ve en el siguiente script:



Ten en cuenta en el sigue-líneas que: si devuelve un 0 es que va por buen camino, si devuelve un 1 habría que girar hacia la izquierda, si devuelve un 2 habría que girar hacia la derecha y si devuelve un 3 es que se ha ido de la línea negra, y en ese caso, "lo mejor" es que des marcha atrás.

Podríamos mejorarlo con el sensor de ultrasonidos, incluyendo otro condicional “Si...entonces”, dentro del comando “Por siempre” anterior que pare el mBot antes de

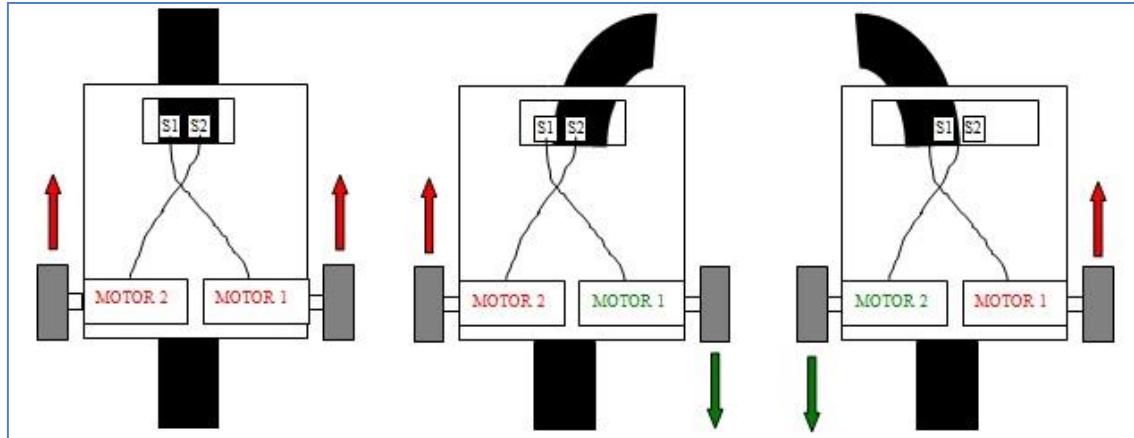
que caiga. Para ello podemos usar una variable “siguelinea” de modo que, si está a cero, se parará:



Podríamos mejorar el programa anterior *añadiendo pequeños tiempos de espera* en cada movimiento. Es decir, pequeños retardos de 0.1 segundos en cada movimiento para que no se escape de la línea si esta es cerrada y le dé tiempo a girar. De esa forma, conseguiríamos que el robot se desviara menos veces de la línea que queremos que siga.

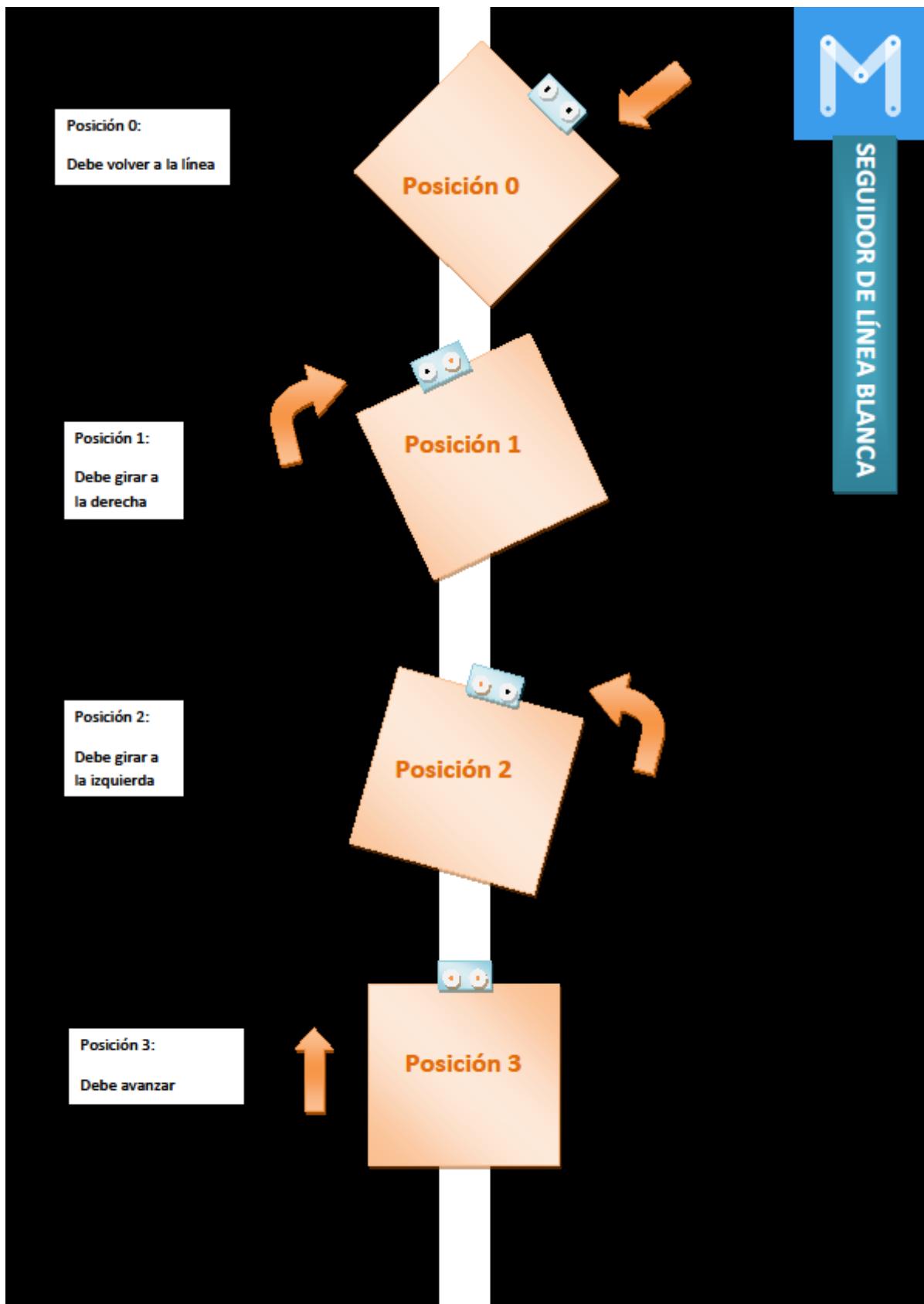
En las imágenes siguientes se explica el funcionamiento de un pequeño robot capaz de seguir una línea negra sobre una base blanca. Nuestro robot consta de dos motores para mover, de manera independiente, las ruedas traseras y de dos sensores de línea que suministran un voltaje próximo a 0v (LOW) si el color es negro y cercano a 5v (HIGH) si es blanco.

Si los dos sensores están sobre la línea negra los dos motores giran hacia delante y el coche se mueve en línea recta	Si el sensor 1 se sale de la línea negra, el motor 1 se mueve hacia atrás y el vehículo gira hacia la derecha.	Si el sensor 2 se sale de la línea negra, el motor 2 se mueve hacia atrás y el vehículo gira hacia la izquierda.
---	--	--



Falta tener en cuenta una última posibilidad, que el robot esté totalmente fuera de la línea negra. La tarjeta mCore debe leer el estado de los sensores de línea y dar las órdenes oportunas a los dos motores.

- B) A la hora de programar que mBot siga la línea blanca, debemos tener en cuenta qué posición nos devuelve el sensor de línea cuando detecta que está dentro o fuera de la línea blanca. Estas posiciones, pueden verse representadas en la siguiente imagen:



Con pequeñas modificaciones en el programa “seguidor de línea negra”, ya tendríamos el seguidor de línea blanca.

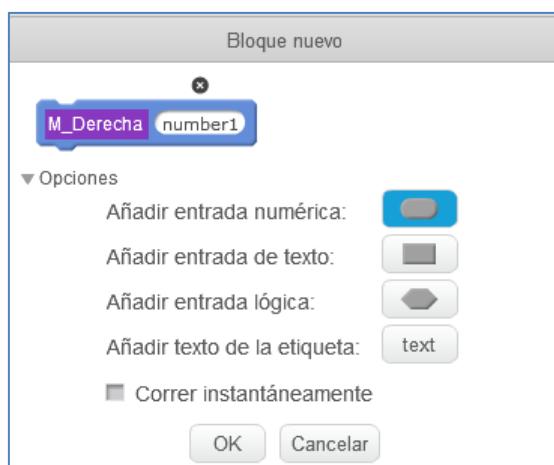
Divirtiéndome con mBot: Guía de manejo y programación

Supongamos que las conexiones de los motores M1 y M2 de nuestro mBot son las siguientes.

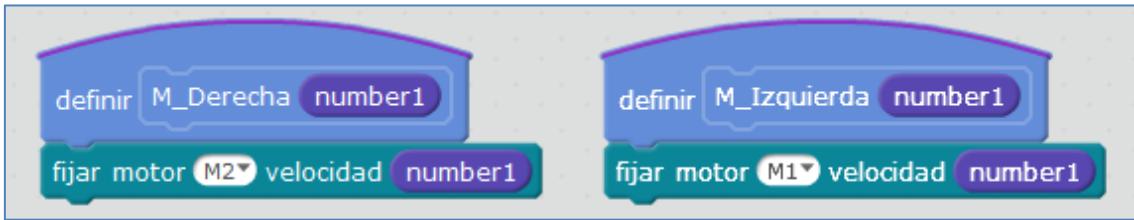
Visto de forma frontal: M1 es el Motor Izquierdo y M2 es el Motor Derecho.



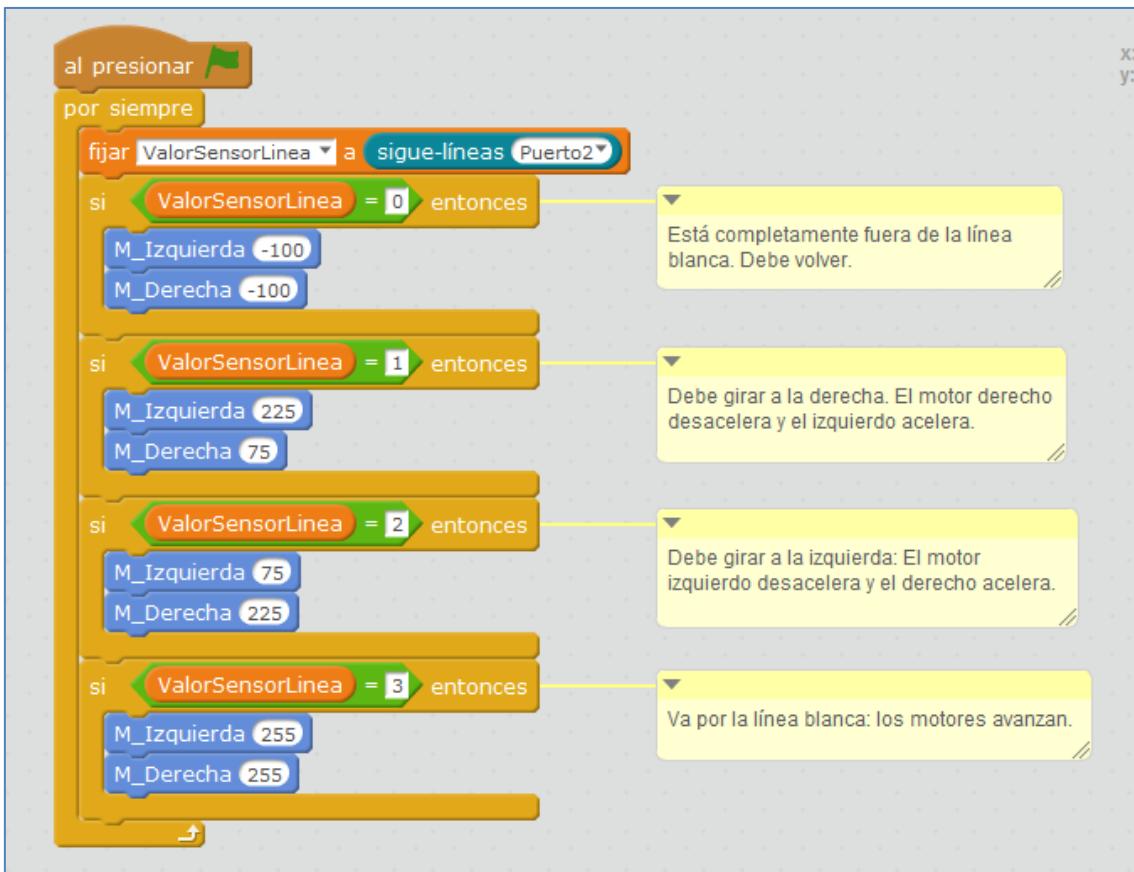
He decidido construir bloques para cada movimiento de ambos motores: Motor derecho con M2 y motor izquierdo con M1. Cada bloque depende de una variable numérica llamada “number1”:



Crear un bloque con entrada numérica (Datos y Bloques)



El programa final, que podemos utilizar para nuestro mBot (en este caso, se ha creado sin añadir los tiempos de espera o retardos en cada movimiento) es el siguiente:



Este script controla cada una de las posiciones u opciones que se pueden dar. Gráficamente, estas posiciones se han simulado en las siguientes imágenes:



Posición 0 (fuera de la línea blanca)



Posición 1



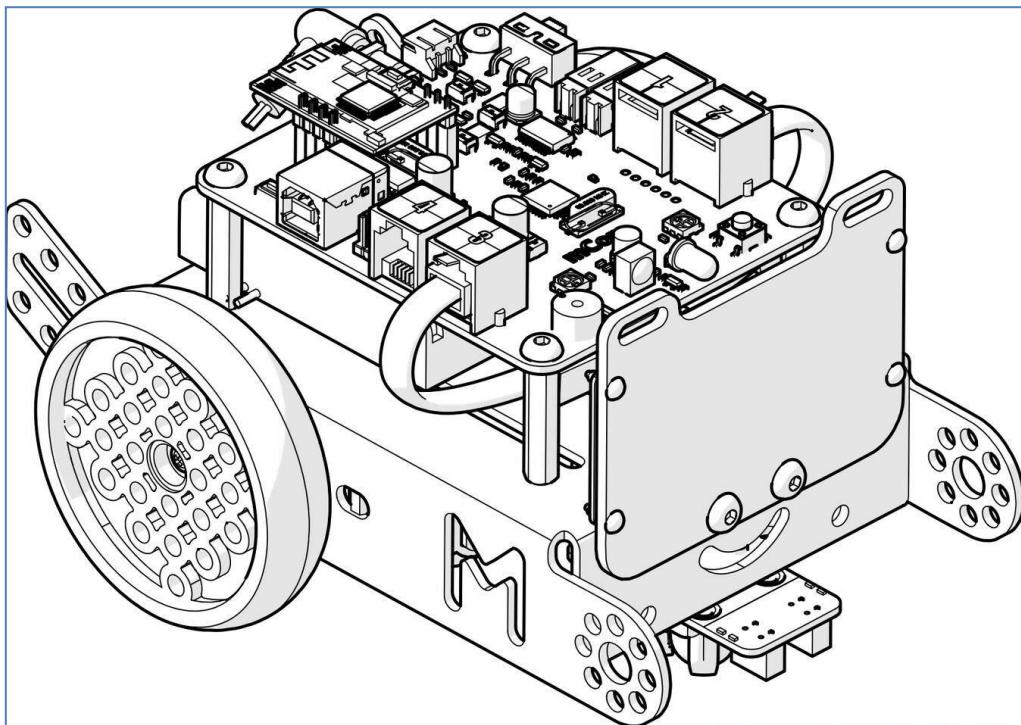
Posición 2



Posición 3 (Detecta blanco)

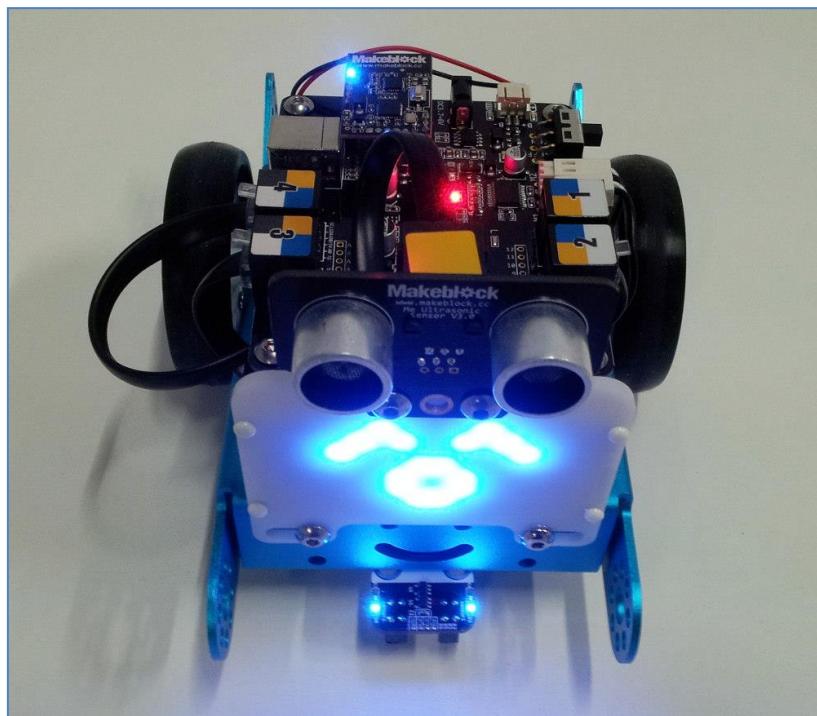
4.6. Matriz de LEDs 8x16

Antes de nada, debemos montar la matriz de LEDs en el robot mBot. Su posición correcta de montaje es la que se muestra en la siguiente figura, pero, si se desea, podemos variar el montaje y este cambio debe tenerse en cuenta a la hora de programarla:



Matriz de LED en posición correcta para su fácil programación

Otra forma de montaje (invertida) que varía su programación es el siguiente:

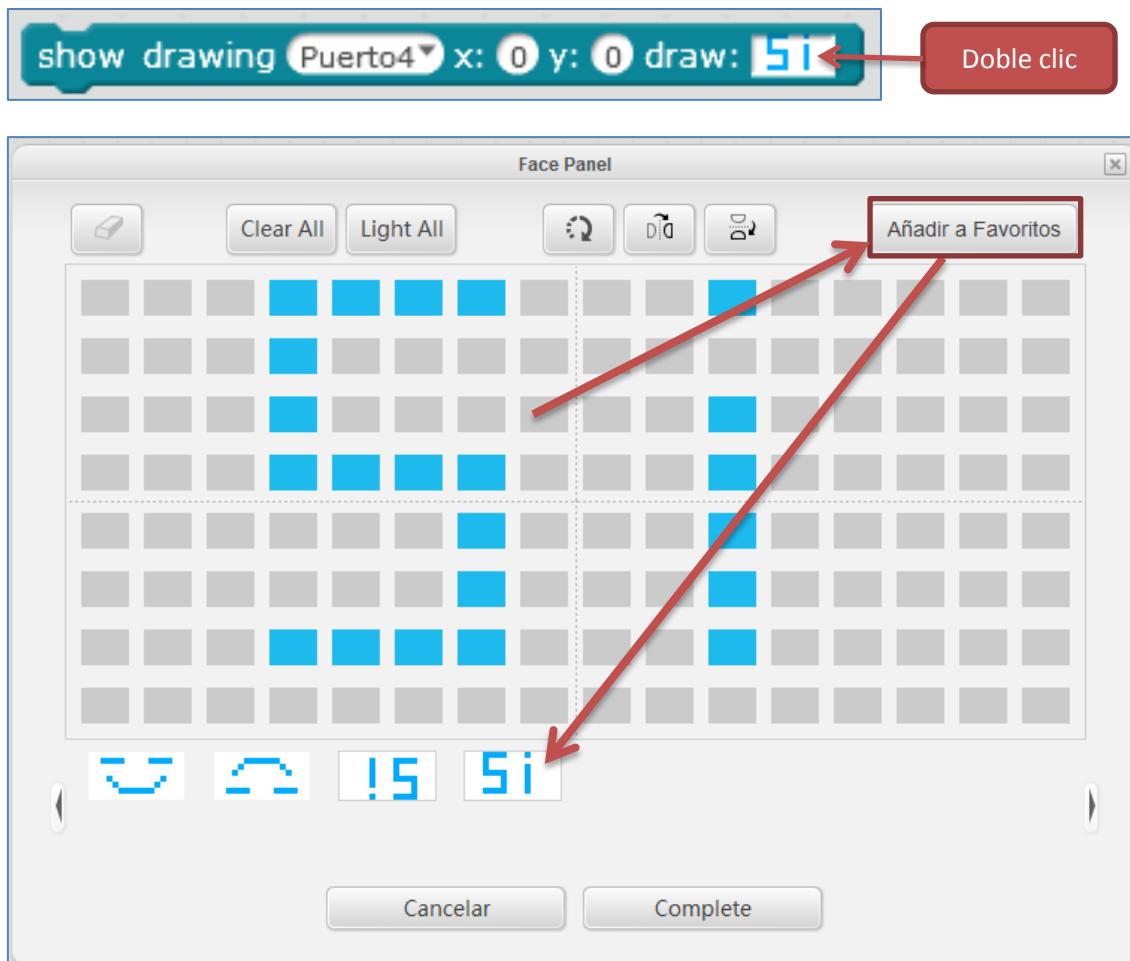


Después de implementarla debemos seguir los siguientes pasos:

1. Conectar el mBot al ordenador mediante un cable USB y actualizar el firmware (mBot) en mBot. (*Conectar > Actualizar Firmware*)
2. Encender el mBot y seleccionar el puerto serie y placa correctos.
3. Escribir los programas y ejecutarlos con doble clic y bandera verde.

La instrucción o comando que más se usa al trabajar con la matriz de LEDs es **show drawing**. Si hacemos *doble clic*, nos permite poner el gráfico que queramos, e incluso guardarlo en favoritos!

(OJO! El gráfico saldría en el montaje correcto porque, si la montamos invertida, su gráfico también debemos crearlo invertido. Esto se explica mejor en los siguientes ejemplos).



Veamos varios ejemplos de programación con mBlock:

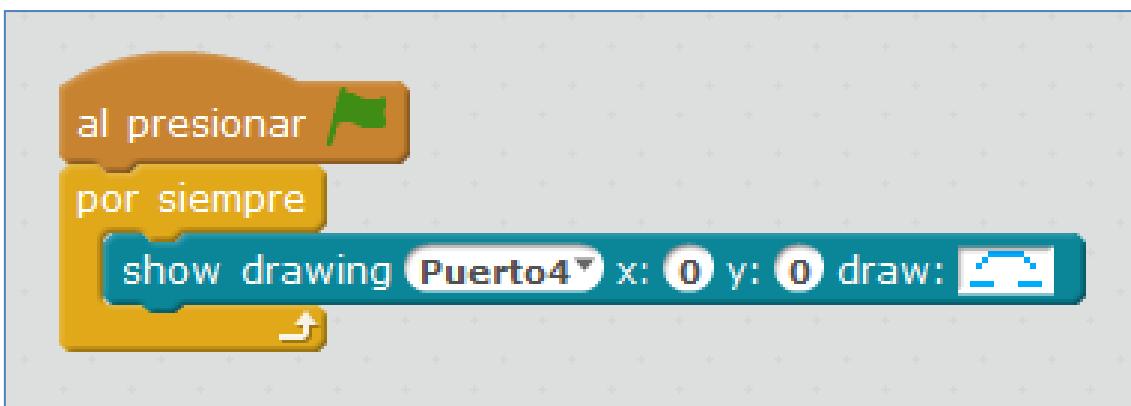
Divirtiéndome con mBot: Guía de manejo y programación

Ejemplo 1: Queremos que en la matriz se refleje una cara alegre:

Con la matriz correctamente montada, crearíamos el siguiente programa:



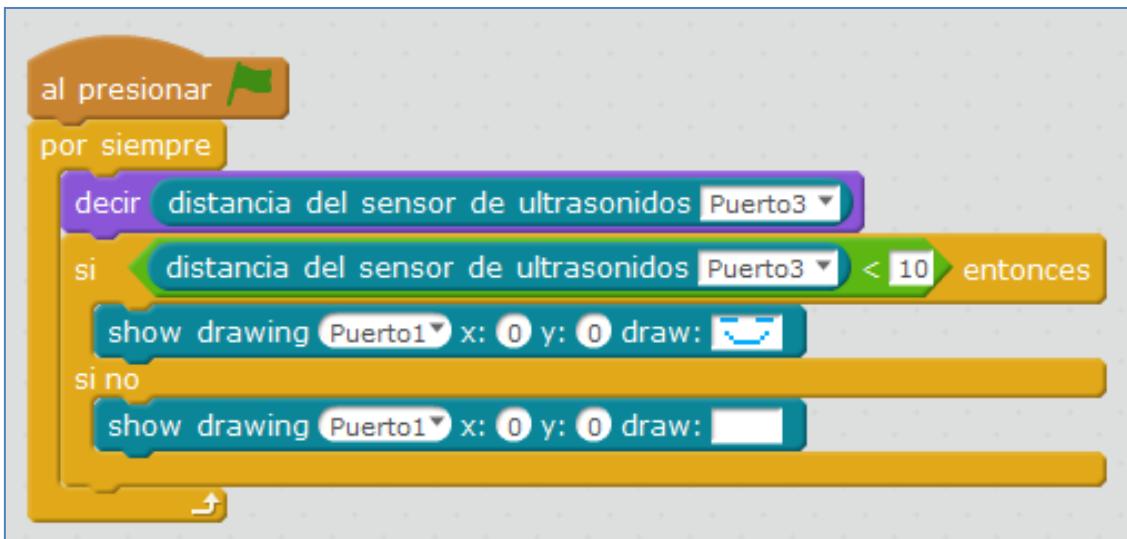
Pero, con la matriz en montaje invertido, deberíamos enviarle al robot el siguiente programa:



Ejemplo 2: Queremos mostrar un pequeño texto permanentemente. Por ejemplo, la palabra "Si", en la configuración de la matriz invertida. Lo programaríamos de la siguiente forma:



Ejemplo 3: Combinando la matriz con el sensor de ultrasonidos:



Ejemplo 4: Podemos enviar un texto, con la instrucción **show face**

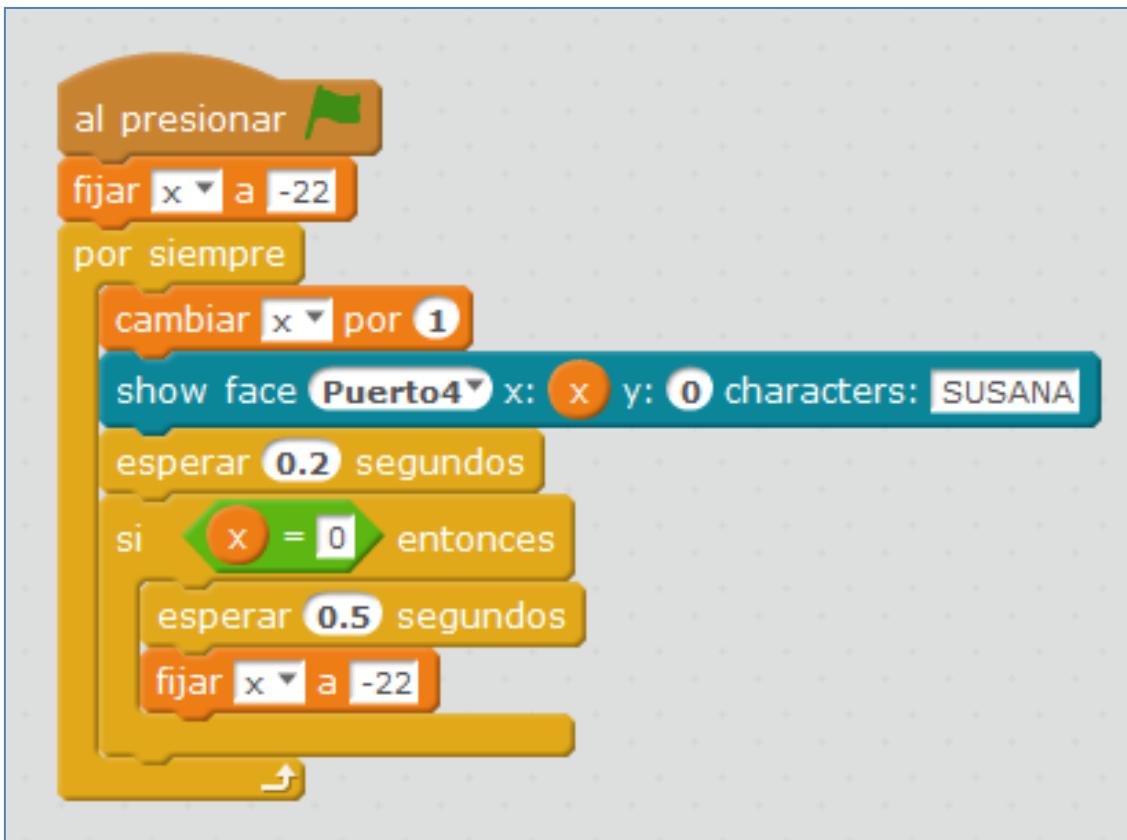
OJO, si está invertida, el texto también hay que invertido

```
show face [Puerto1 v] x: 0 y: 0 characters: [Hello]
```

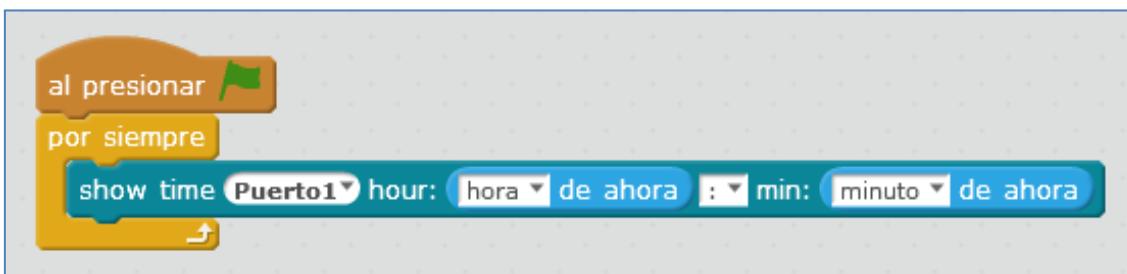
El siguiente programa envía a las coordenadas (0,0) la palabra SUSANA pero, para que coja, debemos programarlo como sigue:



Si queremos que se repita continuamente el texto:



Ejemplo 5: También podemos enviar la hora. Un script que nos serviría para ese fin es el siguiente:



Ejemplo 6: En el siguiente ejemplo usamos el módulo de ultrasonidos, y dos botones del mando de mando de IR del robot (las teclas R8 y R9).

El sensor de IR puede recibir del mando números, letras A-D, flechas y el botón configuración.

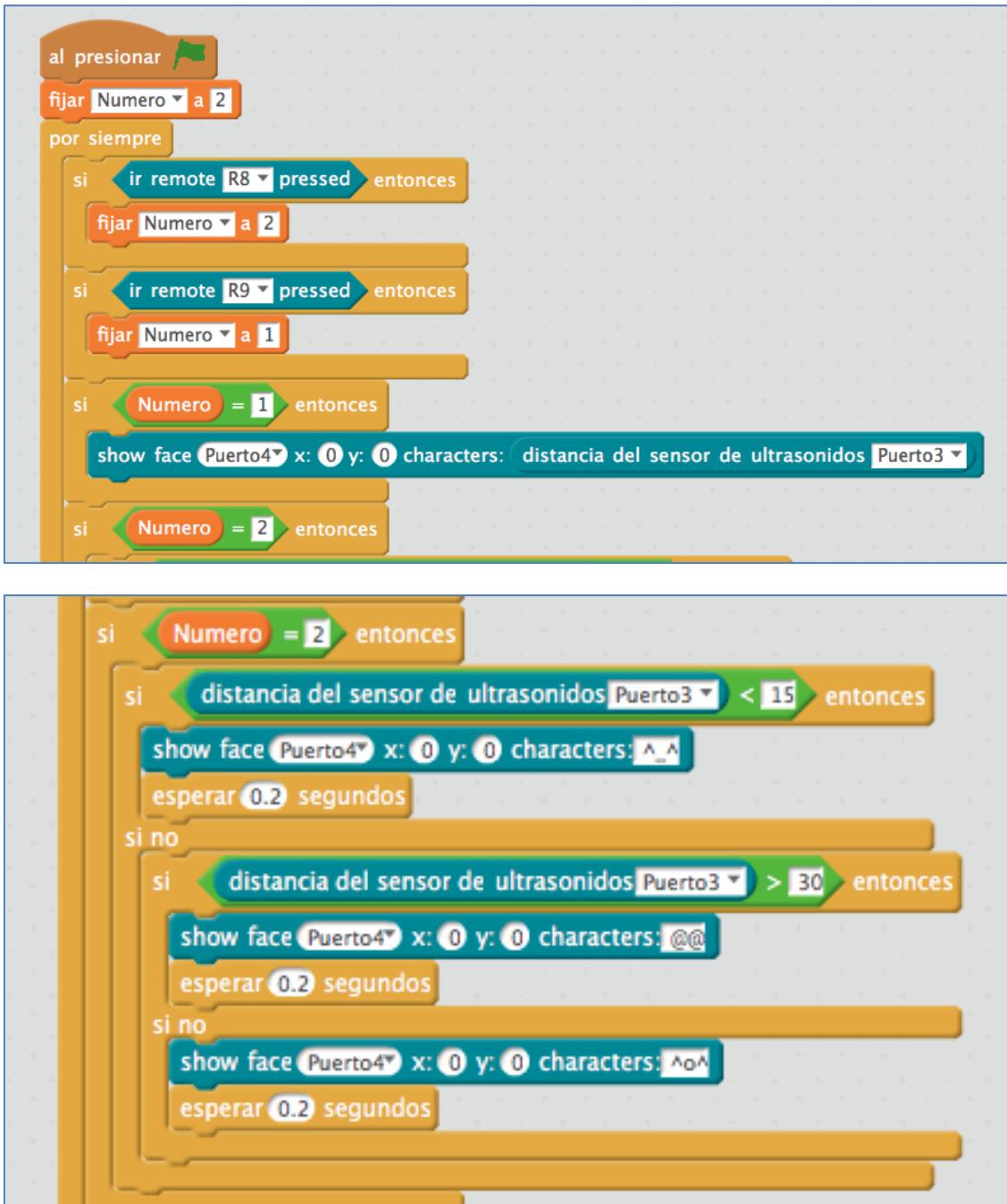
Ojo: *El mando no es exclusivo de un robot, es decir, los demás robots del aula reciben la misma información por lo que es importante apuntar el mando a los sensores del robot y no a otro robot del aula.*

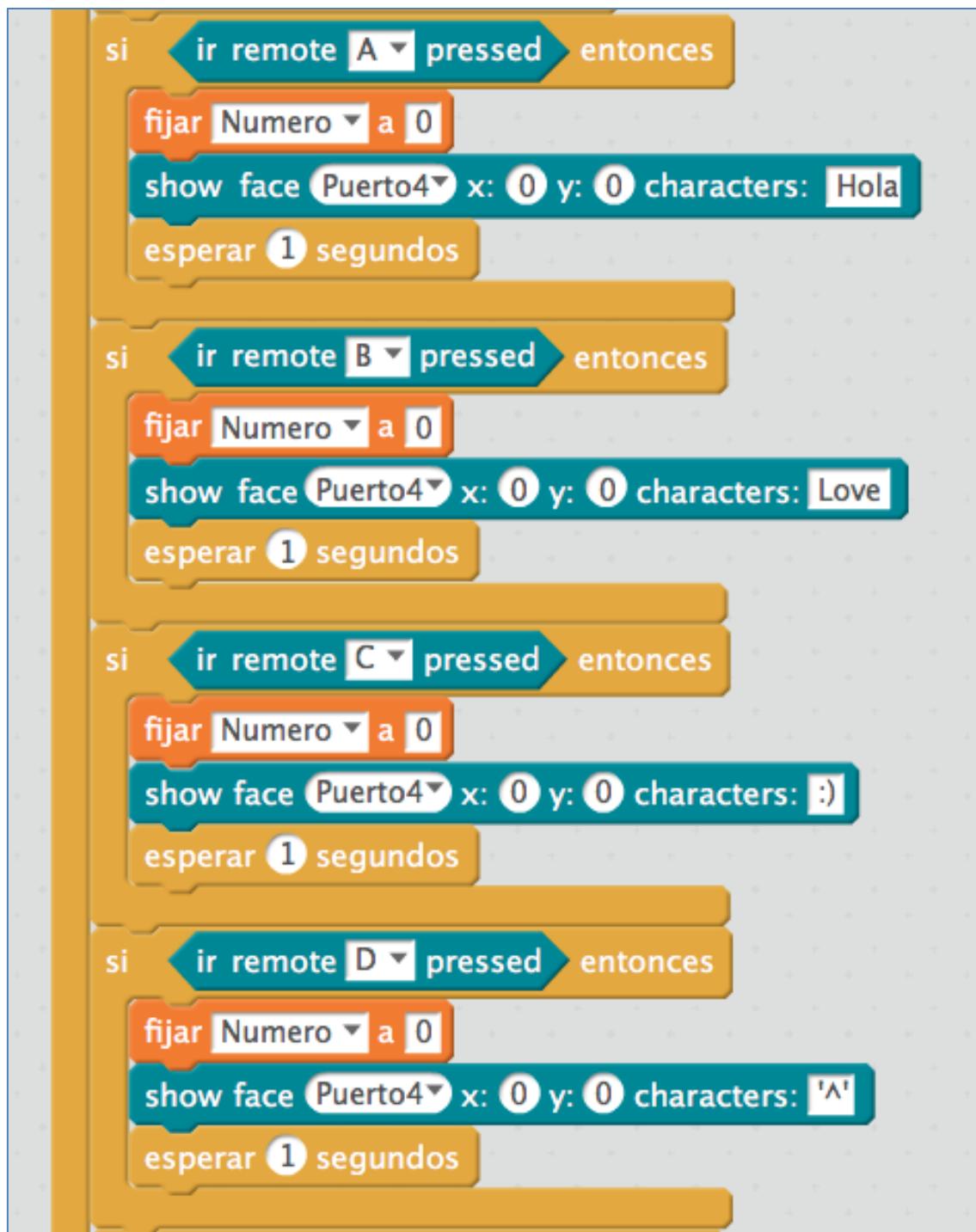
Realizar programas con el sensor de IR tiene una dificultad: no se puede hacer a través del ordenador. Es decir, no se puede realizar como lo hemos hecho hasta ahora: Bandera verde y comunicación entre PC y mBot

Divirtiéndome con mBot: Guía de manejo y programación

¿Por qué? porque el mBot tiene instalado en esta configuración el programa por defecto *Firmware* de leer el mando, y no podemos saltarlo.

En el siguiente ejemplo, gracias al módulo de ultrasonidos, el robot testea la distancia a la que está un objeto (si es menor que 15 o mayor que 30) y, según las diferentes distancias, muestra unas expresiones en la matriz de LED del robot mBot. El script es el siguiente:

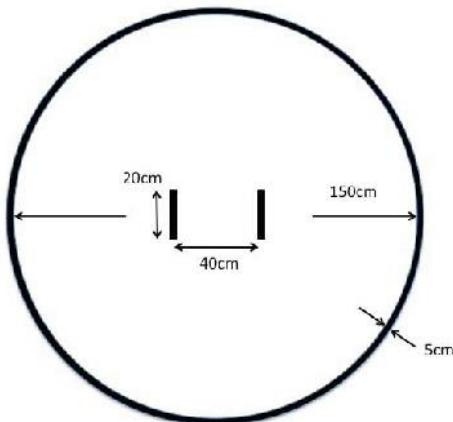






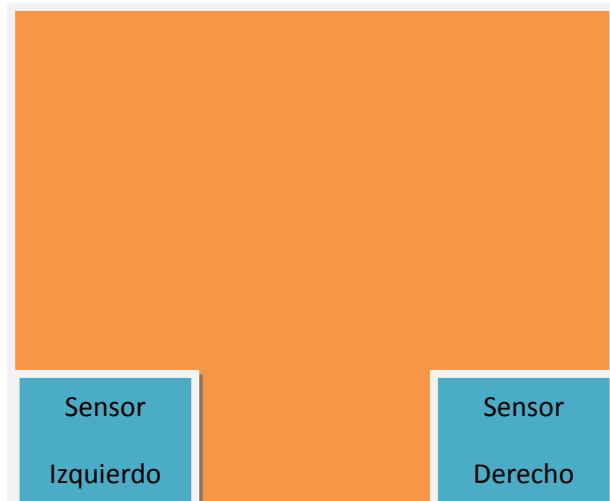
4.7. Robot sumo

La programación para una prueba *sumo* varía drásticamente dependiendo del número de sensores que le pongas a tu robot y de donde los acomodes. La prueba *sumo* se realizará en una pista circular. Dentro de esa pista, el robot debe atacar y saber buscar a los contrincantes, pero, al mismo tiempo, no debe salir de ese recinto, es decir, debe ser capaz de detectar el color del borde de la pista y, en ese momento, volver a dentro.



Un sensor que nos servirá para detectar al enemigo es el sensor de ultrasonidos. Nuestro robot puede disponer de uno o de dos, y en el caso de que sean dos, estos pueden estar situados en el lugar que nos parezca mejor (uno delante y otro atrás, los dos delante a ambos lados del robot, etc.).

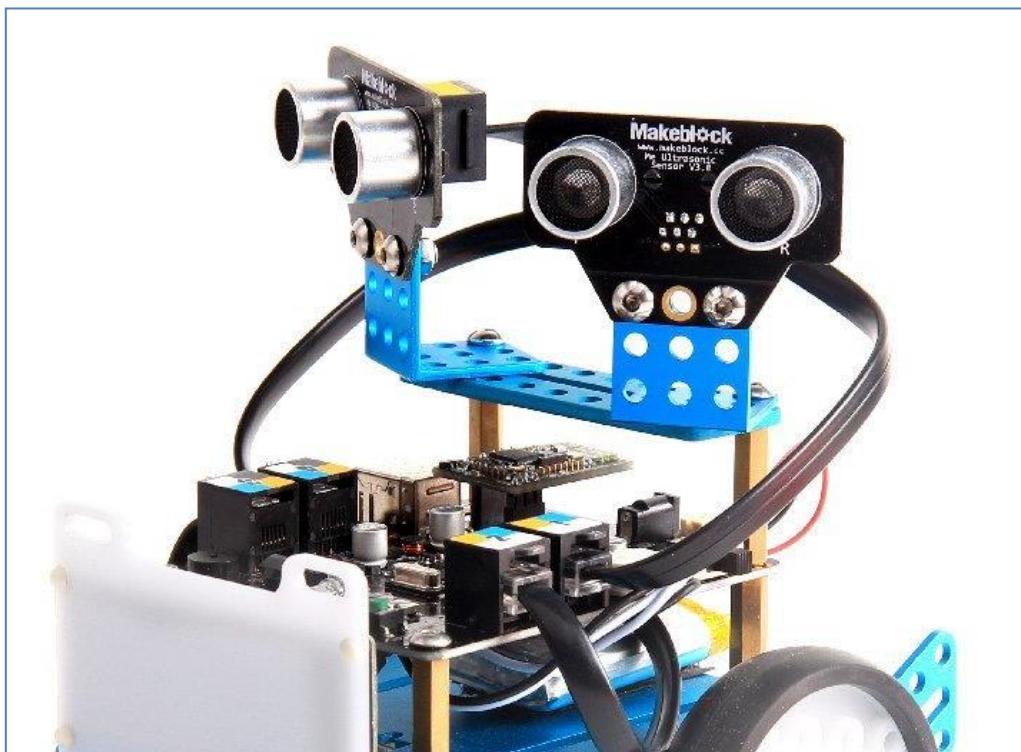
Tras nuestra elección, la programación es sencilla si tenemos claro cómo queremos que funcione nuestro robot. Por ejemplo, imaginemos que nos hemos decantado por dos sensores de ultrasonidos a ambos lados del robot en la parte frontal (después incluiremos los sensores de línea). Vamos a llamarlos sensor derecho y sensor izquierdo. En la siguiente imagen puede verse su situación esquemática en un bloque:



Parte frontal del robot

Podemos tener la estrategia de ir siempre hacia adelante o de girar a la derecha o izquierda si ningún sensor ha detectado nada, pero, si algún sensor detecta al oponente, entonces debe dirigirse a él para atacar. Esta estrategia se resume en la siguiente tabla de verdad donde 0 significa que no detecta y 1 que detecta:

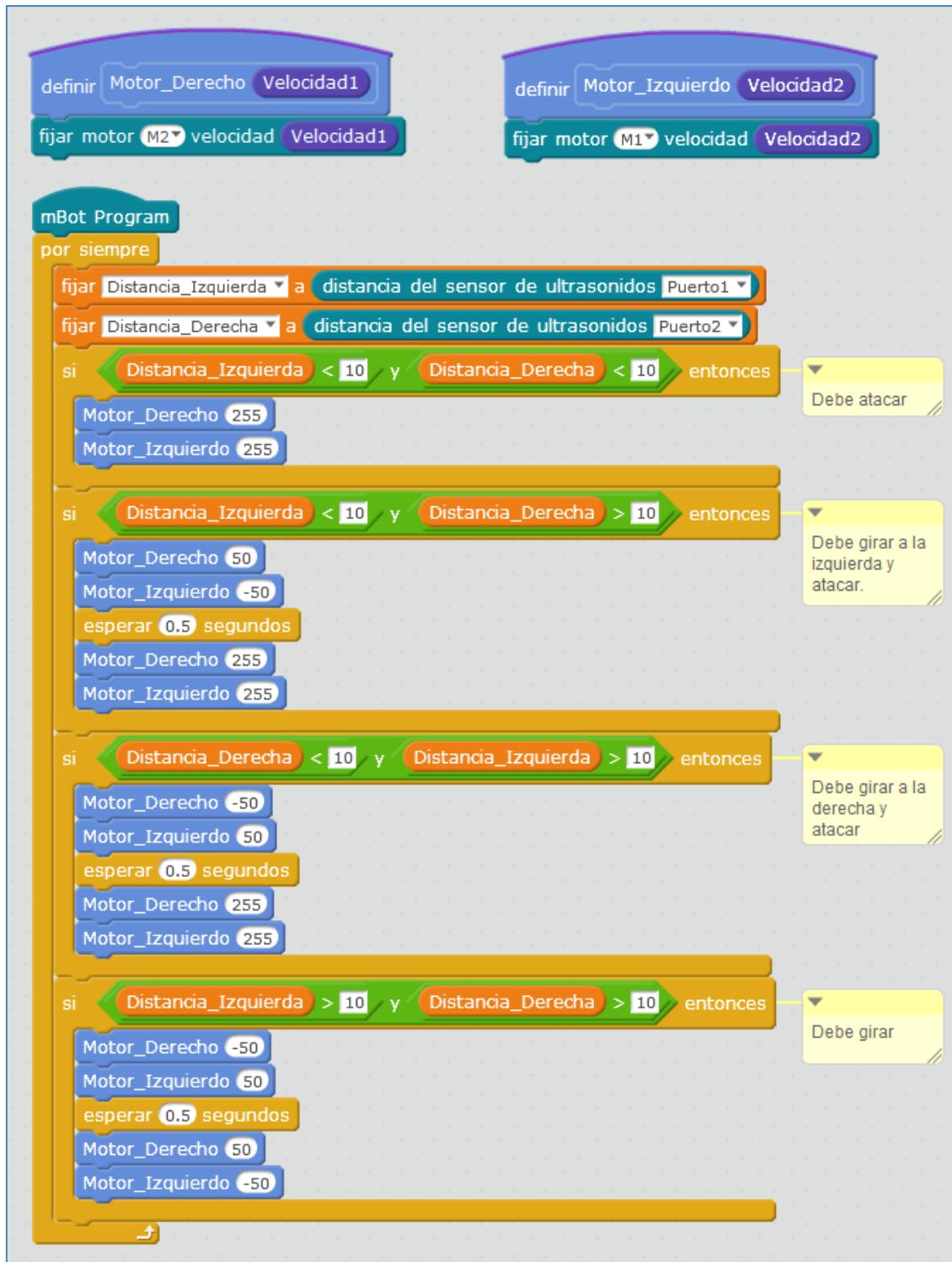
Sensor Izquierdo	Sensor Derecho	Enemigo	Acción
0	0	No	Girar o seguir adelante
0	1	Si	Girar a la derecha
1	0	Si	Girar a la izquierda
1	1	Si	Ir de frente



mBot con 2 sensores de ultrasonidos

Divirtiéndome con mBot: Guía de manejo y programación

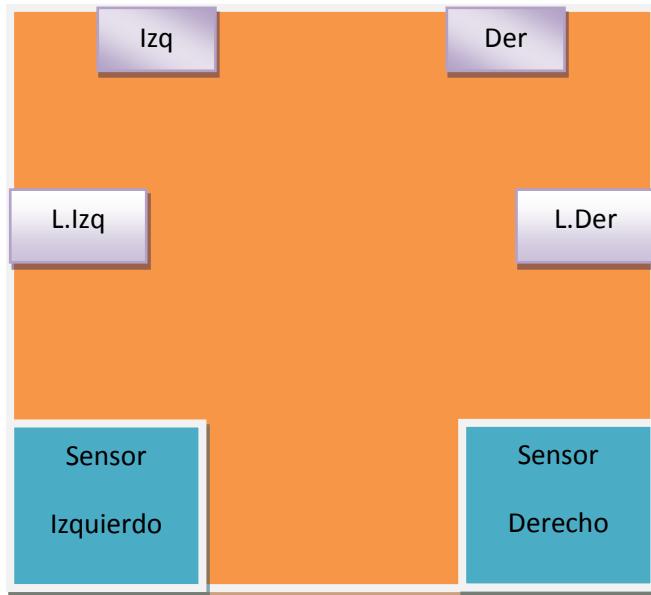
Relativo a la distancia, en el sentido de cuál debemos usar como límite de detección, esta dependerá de la situación exacta de nuestros sensores en el robot y de nuestros deseos en el programa. En este ejemplo, se ha decidido que sean 10cm:



Primera parte del programa Sumo con dos ultrasonidos

Divirtiéndome con mBot: Guía de manejo y programación

Personalmente, lo ideal es que el robot siempre ataque de frente y al mismo tiempo, que no salga de la zona de lucha, limitada por un color determinado de línea. Para conseguir esto último, podemos usar sensores detectores de línea y programarlos para que, cuando el sensor detecte la línea, el robot se aleje de ella. Supongamos que usamos 4 y los situamos de la siguiente forma:



Pensando en nuestra estrategia, una posible tabla de verdad, considerando 0 como no detección y 1 como detección, podría ser la siguiente:

L.Izq	Izq	Der	L.Der	Acción a programar
0	0	0	0	Buscar
0	0	0	1	Derecha Rápido: La rueda izquierda gira hacia adelante y la derecha hacia atrás
0	0	1	0	Derecha Suave: sólo gira la rueda izquierda y la rueda derecha está quieta
0	0	1	1	Nada
0	1	0	0	Izquierda Suave
0	1	0	1	Nada
0	1	1	0	De frente
0	1	1	1	Nada
1	0	0	0	Izquierda Rápido
1	0	0	1	Nada
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

En ella, basta con programar las filas resaltadas en lila para que nuestro robot sumo ataque y evite salir del recinto de lucha.

Divirtiéndome con mBot: Guía de manejo y programación

A la hora de programar ambos sensores a la vez, ultrasonidos y sensores de línea, “todo” es libre hacia los intereses del programador. Me explico, puedo considerar que lo más importante es que mi robot priorice en escapar de la línea blanca tomando como secundario su ataque o, cualquier otra alternativa a esta estrategia. En mi caso, para mí, lo más importante es salvarme, por lo que, en los siguientes ejemplos particularizados al mBot, se verá esta defensa inicial.

Ejemplo 1 de programa sumo

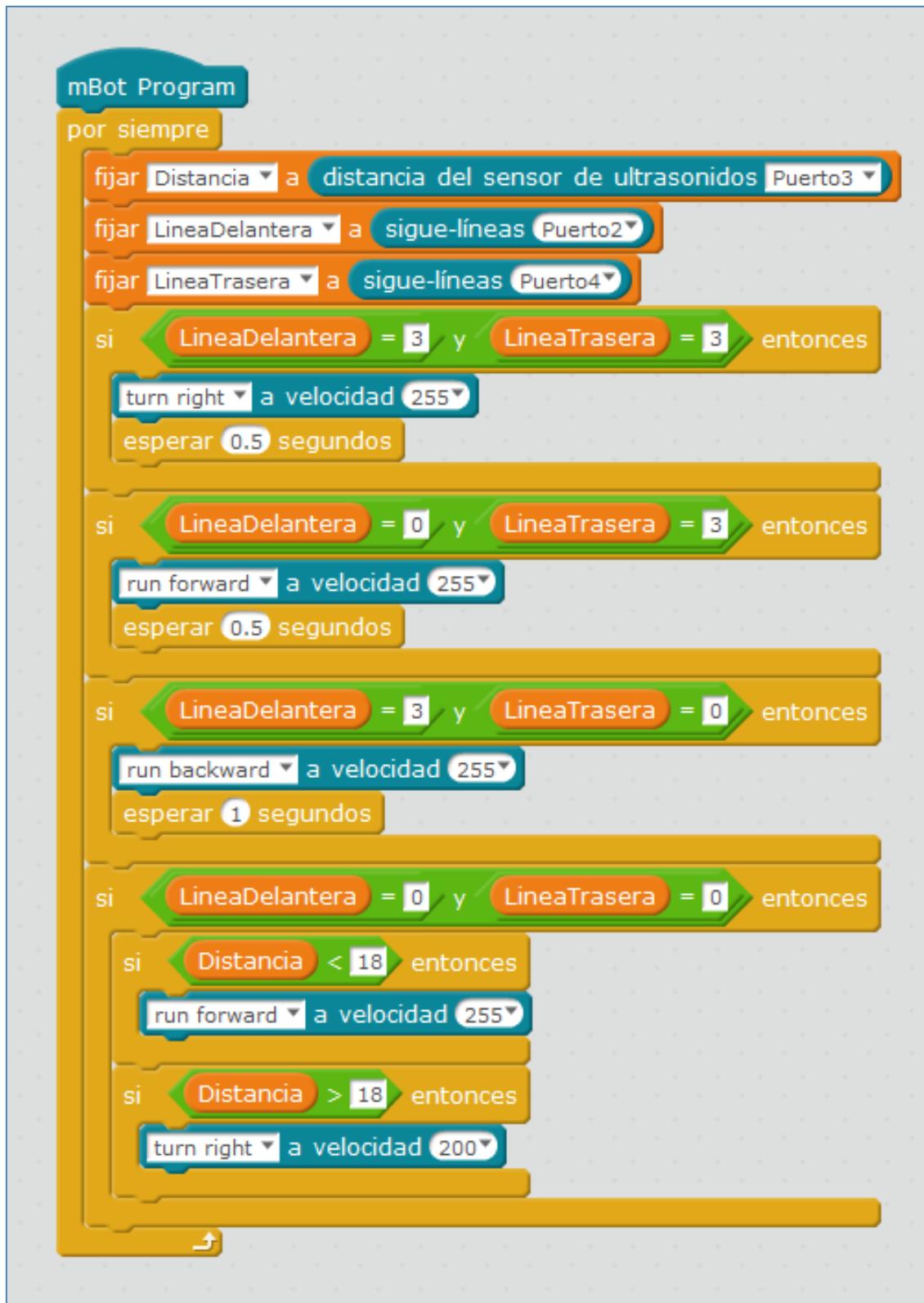
En este caso, usaremos un sensor de ultrasonidos para la parte delantera del mBot y dos sensores de línea (uno delantero y otro trasero). El programa está pensado para luchar en un circuito limitado por una línea blanca.

Disponemos de tres variables: *Distancia* (que es la que medirá el sensor de ultrasonidos y que se estima en 18 cm para la detección), *LíneaDelantera* (medirá negro o blanco en el sensor delantero) y *LíneaTrasera* (medirá negro o blanco en el sensor trasero):



Comenzaré con mi prioridad: estar en el recinto. Los sensores de línea, como ya vimos, presentan 4 posiciones y verán blanco en la posición 3. En este caso, debemos intentar que nuestro robot no salga del circuito de lucha. Por lo tanto: Si ve línea blanca el delantero, debemos ir hacia atrás, y si ve blanca el trasero, debemos ir hacia adelante. Ya conseguido esto, también debe atacar, y lo hará si está en el recinto (ve negro) y detecta a un oponente. También debe buscar, y lo hará si está en el recinto y no detecta a un oponente.

Un posible programa sumo que afronta las consideraciones anteriores es el siguiente:

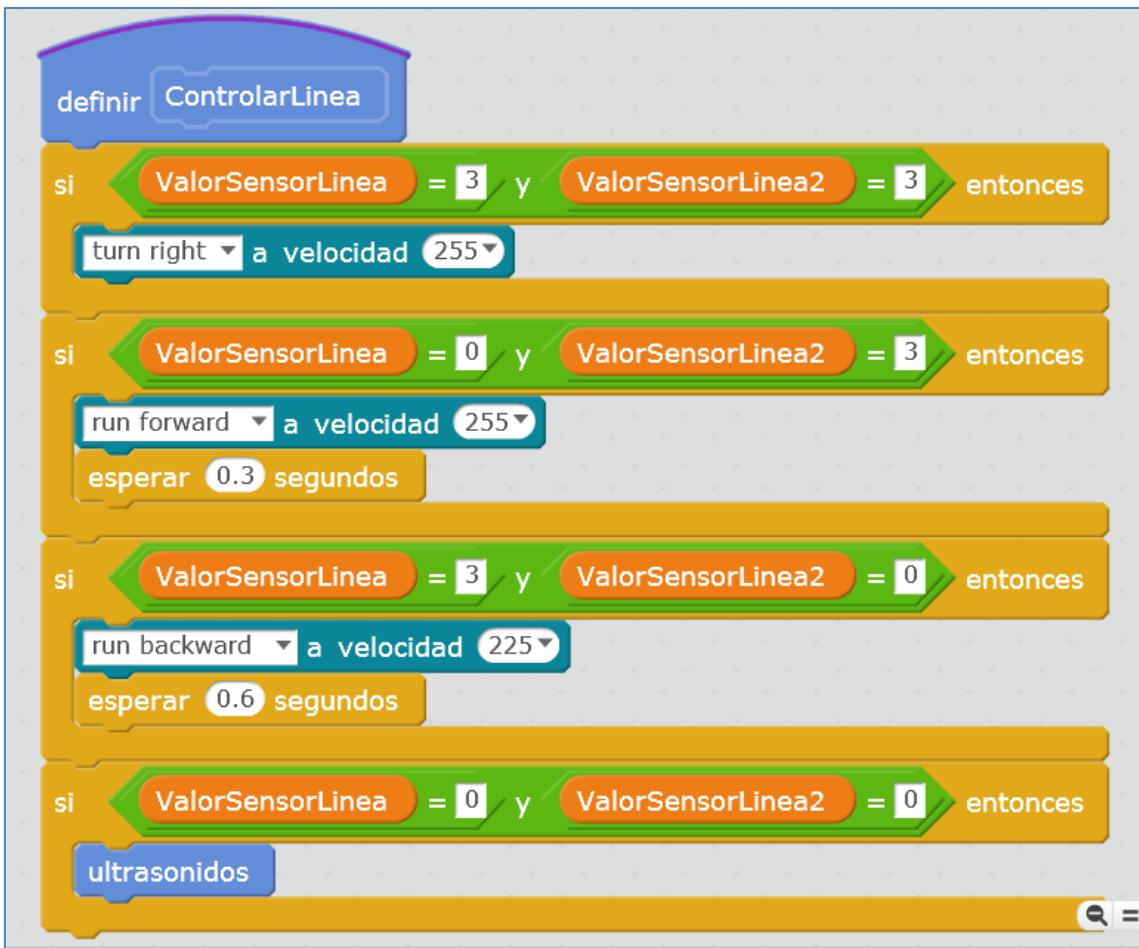


Ejemplo 2 de programa sumo:

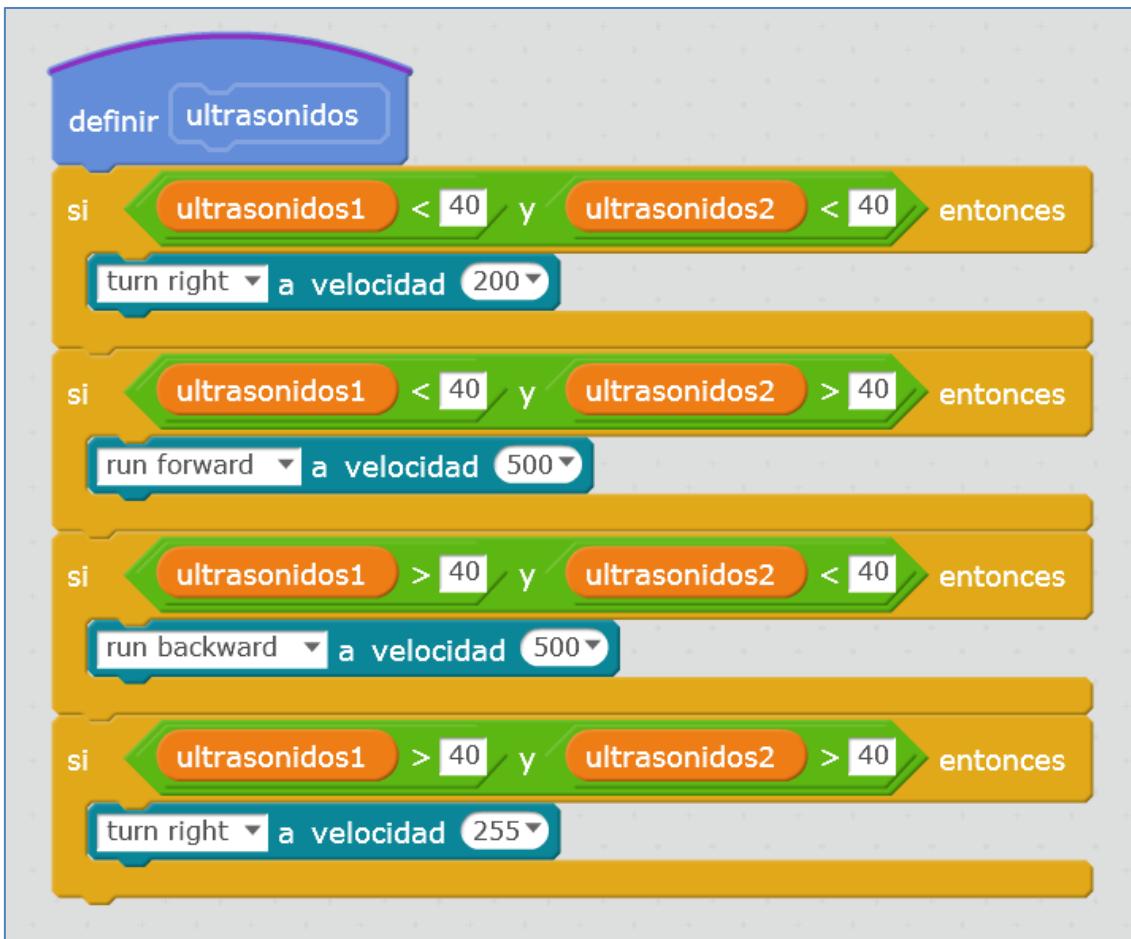
Supongamos ahora que queremos usar dos sensores de ultrasonidos y que los situamos, uno delante y otro detrás. Además, a nuestro robot le añadimos dos sensores de línea y de la misma forma que antes, uno estará delante y otro detrás. Para su control, necesitamos definir 4 variables y estas pueden ser las siguientes: *ValorSensorLinea* (para delante), *ValorSensorLinea2* (para atrás), *ultrasonidos1* (para delante) y *ultrasonidos2* (para atrás):



Si mi prioridad es la misma; estar en el recinto, comenzaré con esa línea de programación. Los sensores de línea, como ya vimos, presentan 4 posiciones y verán blanco en la posición 3. En este caso, debemos intentar que nuestro robot no salga del circuito de lucha. Por lo tanto: Si ve línea blanca el delantero, debemos ir hacia atrás, y si ve blanca el trasero, debemos ir hacia adelante. En este ejemplo, he programado estas posibilidades en un bloque que he llamado *ControlarLinea* y que es el siguiente:

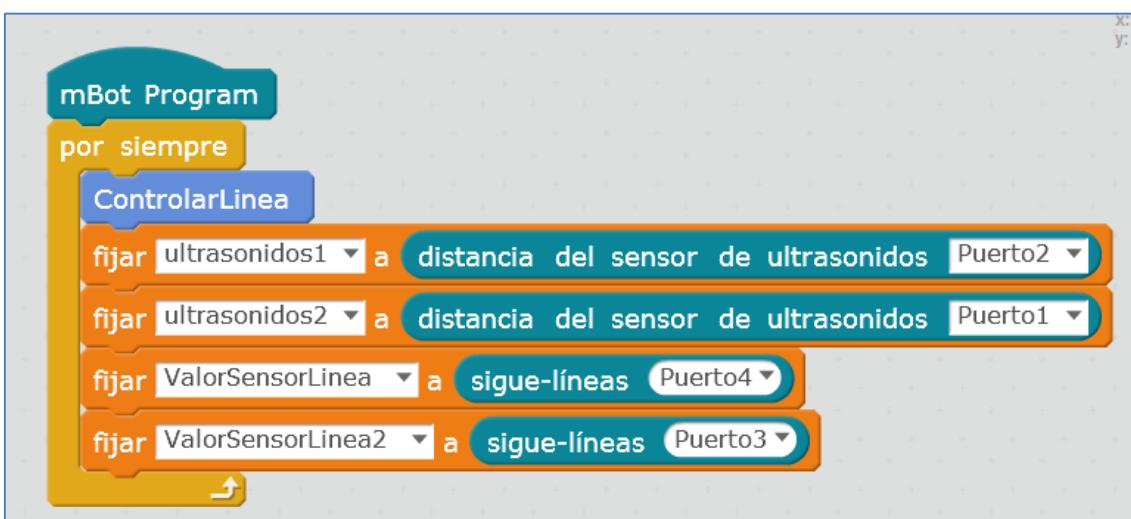


Como podéis ver, en el caso de que el robot esté en el recinto, vea negro, debe comportarse como un buen sumo y atacar o buscar. Esto lo consigo usando sus sensores de ultrasonidos. Sensores que he programado en el bloque definido como “ultrasonidos” y cuya programación, limitada en este caso por el valor numérico de detección 40 cm, es la siguiente:



El bloque de ultrasonidos sólo lo ejecutará cuando el robot esté en el recinto, atacando o buscando en la forma que desee el programador.

Hasta ahora, hemos definido los bloques, pero no el programa de ejecución que los llama y que es el siguiente:



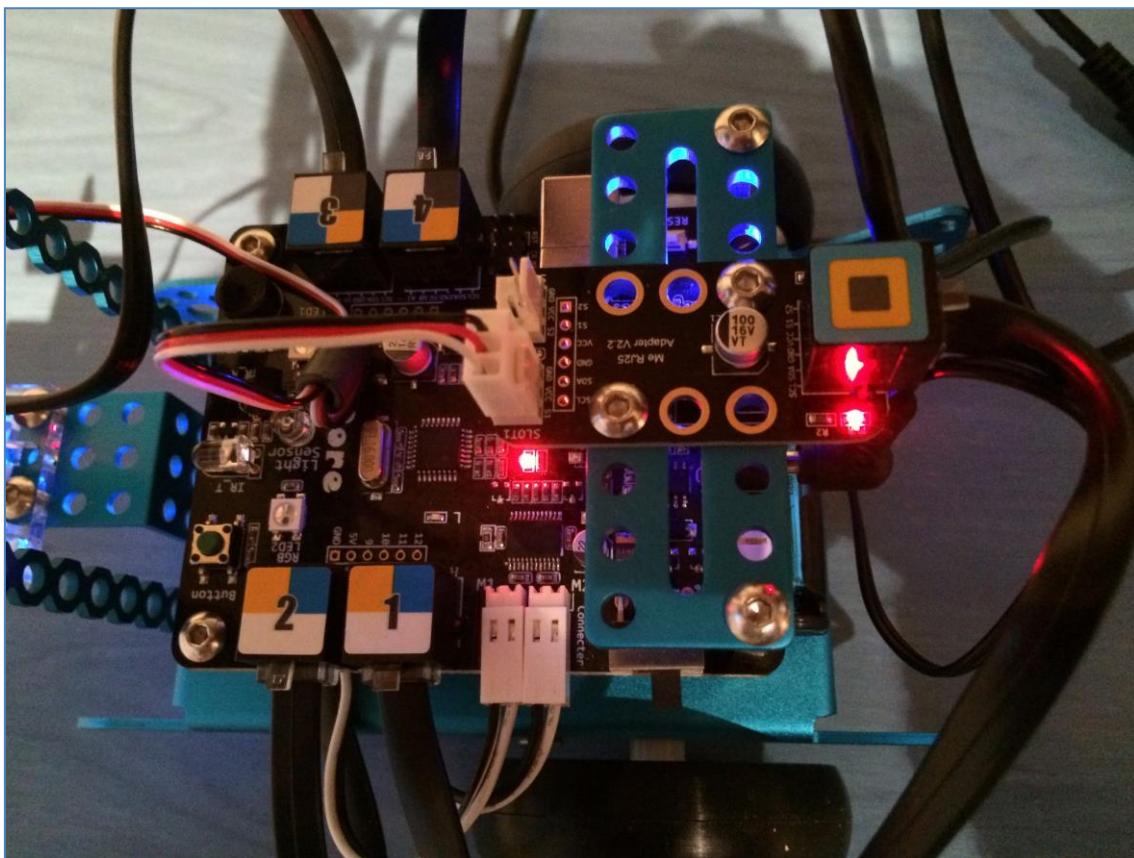
4.8. Servomotor

Un servomotor se puede considerar como un tipo especial de motor de continua que se caracteriza por tener la capacidad de girar un ángulo determinado dentro de un intervalo de operación.



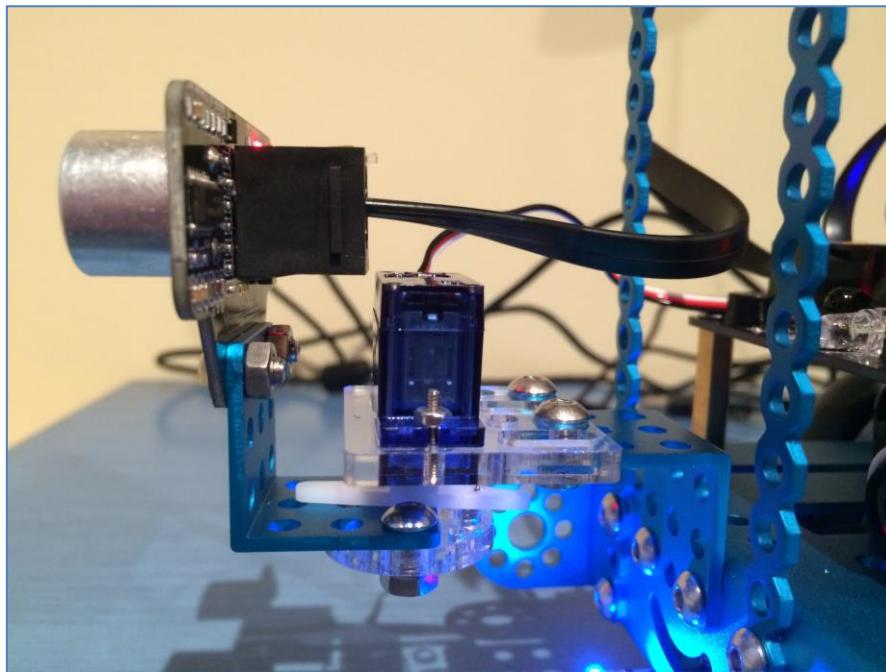
Micro Servo 9g (servomotor)

El servomotor de la casa Makeblock necesita de un adaptador RJ25, como puede verse en el montaje de la siguiente imagen. Todo adaptador RJ25 dispone de 2 "Bancos" o "Slots" o conectores, para poder conectar en el mismo módulo un sensor de temperatura y un servo, por ejemplo. El adaptador necesita de un puerto y puede conectarse a cualquiera de los puertos: 1, 2, 3 o 4, ya que sus colores ID son negro, amarillo y azul. En nuestro caso, lo hemos conectado al puerto 1:



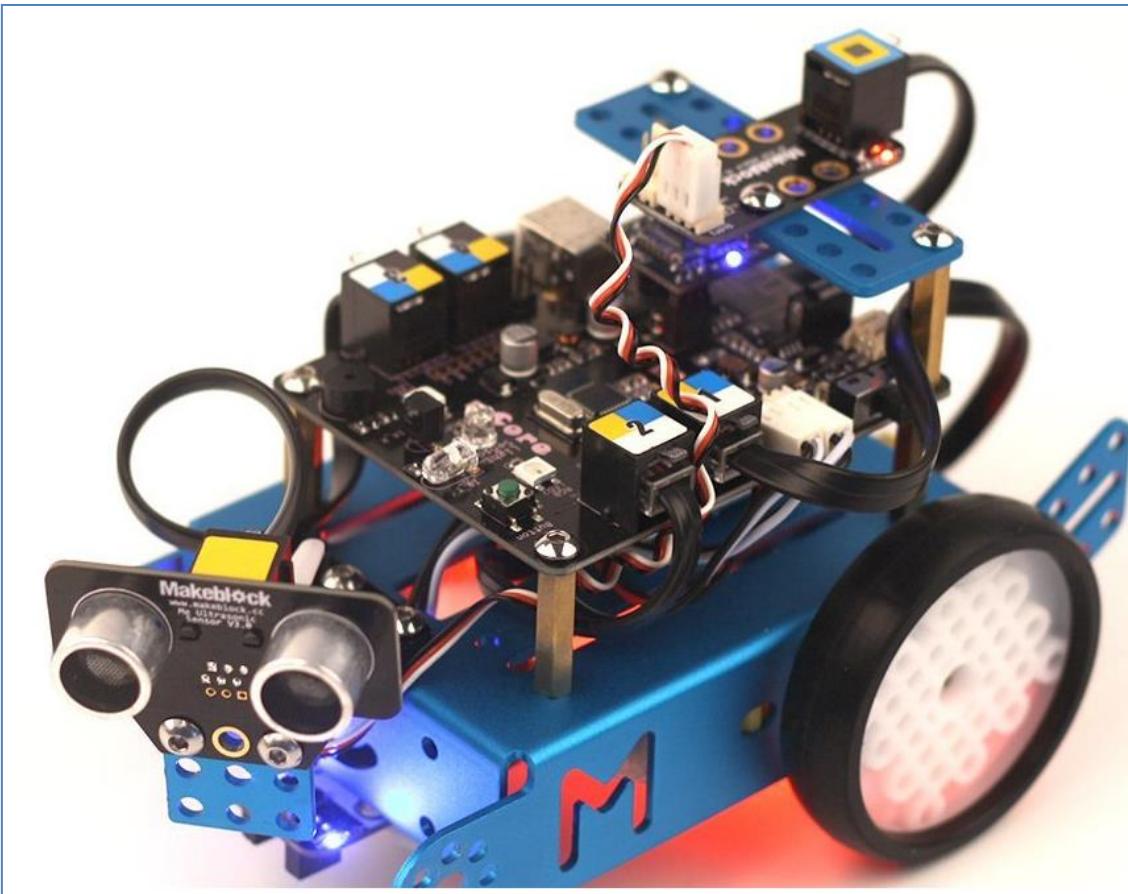
Divirtiéndome con mBot: Guía de manejo y programación

A mayores, en nuestro ejemplo, el servo se ha unido al sensor de ultrasonidos, de modo que, consiga moverlo en diferentes posiciones angulares del servo:



Un script para mover el servo en 5 diferentes posiciones angulares (0, 45, 90, 135 y 180) sería el siguiente: Se puede elegir el “Banco1” (un banco es un slot o un conector) porque no tenemos el adaptador conectado a más de un componente.

```
al presionar [bandera]
fijar angulo a [0]
repetir (2)
  set servo [Puerto1 v] [Banco1 v] angle [angulo]
  esperar (1) segundos
  cambiar angulo por (45)
  set servo [Puerto1 v] [Banco1 v] angle [angulo]
  esperar (1) segundos
  cambiar angulo por (45)
  set servo [Puerto1 v] [Banco1 v] angle [angulo]
  esperar (1) segundos
  cambiar angulo por (45)
  set servo [Puerto1 v] [Banco1 v] angle [angulo]
  esperar (1) segundos
fijar angulo a [0]
```



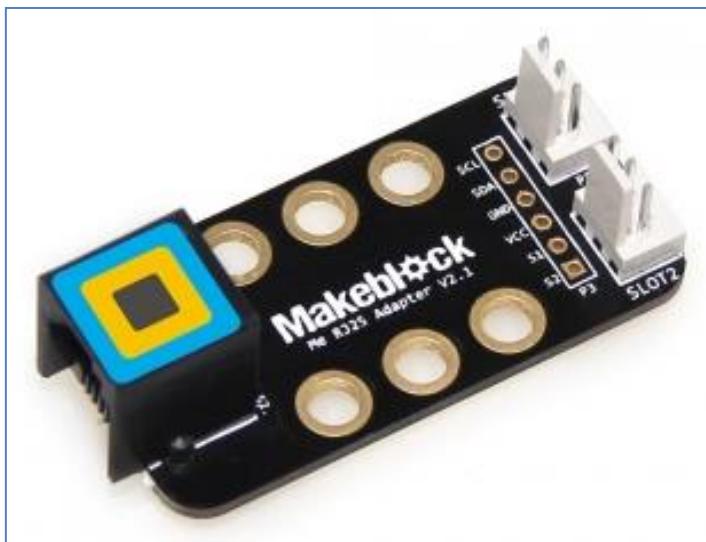
4.9. Sensor de temperatura SD18B20

El sensor de temperatura sumergible de la casa Makeblock es el sensor digital SD18B20. Presenta una precisión de $\pm 0.5^{\circ}\text{C}$ (de -10°C a $+85^{\circ}\text{C}$), siendo, su tiempo de captura de la medida, inferior a 1 segundo. Gracias a su envoltorio estanco sellado podemos sumergirlo en un líquido y usarlo para que nos proporcione la medida de la temperatura de una sustancia líquida. Al ser digital, el valor de temperatura que nos proporcionará no se ve alterado por la distancia del cableado del sensor.

Este sensor se puede programar con arduino o con mBlock, usando en scratch el correspondiente comando:



Para conectarlo a la placa del mBot, necesita de un módulo adaptador RJ25 como el que se muestra en la imagen y que ya hemos utilizado con el módulo servomotor:



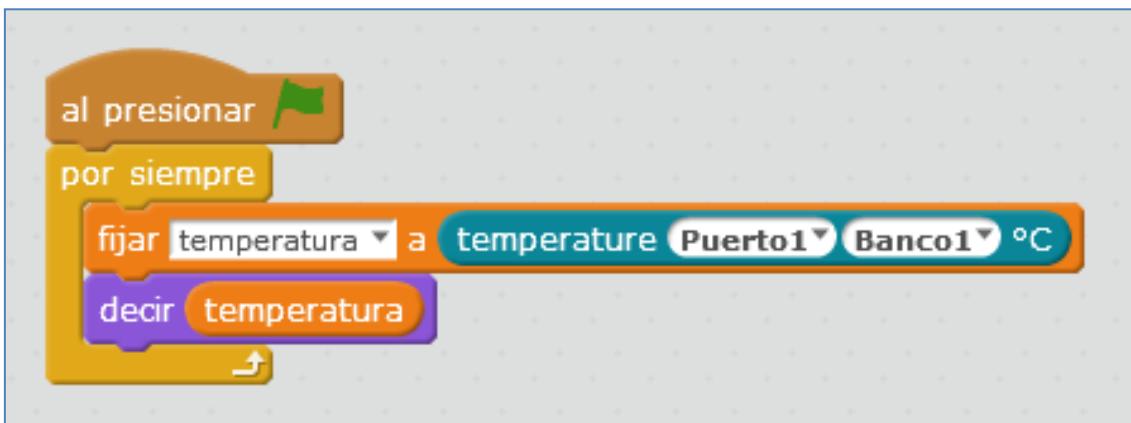
Módulo adaptador RJ25

El sensor de sonda sumergible, cuyo rango de medida es de -55°C a 125°C, puede verse en la siguiente figura:



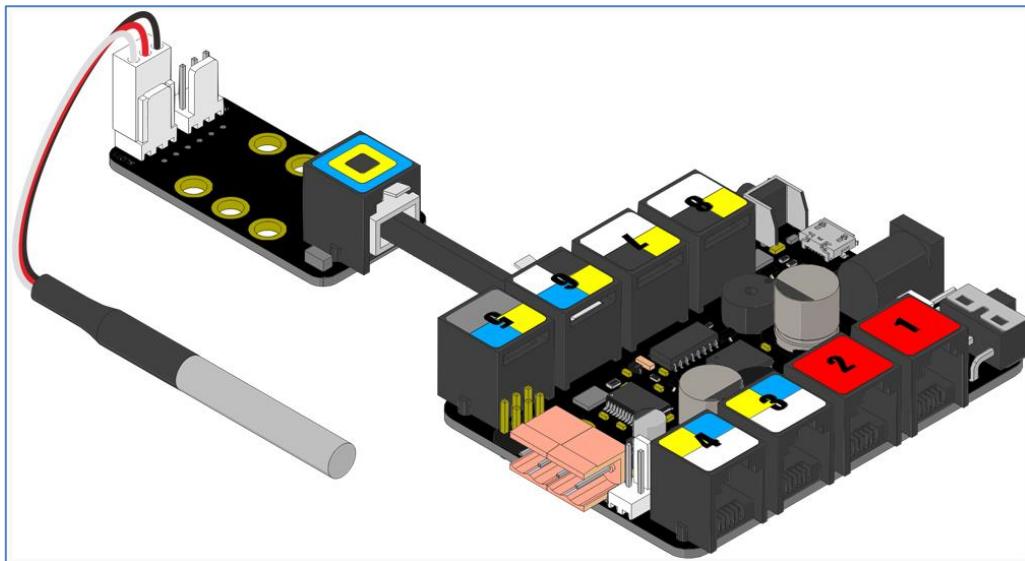
Módulo sensor de temperatura sumergible

Un programa sencillo consistiría en medir la temperatura de la sonda del sensor y mostrarla en el escritorio. El script, estando el sensor conectado al *Banco1* y el adaptador al *Puerto 1*, sería el siguiente:



Divirtiéndome con mBot: Guía de manejo y programación

El conexionado, en este caso usando una placa Orion conectando el sensor por medio del adaptador al puerto 6, sería el siguiente:



En el siguiente ejemplo se ha conectado a la placa mCore en el *banco 2* del *puerto 4*. Se ha creado un programa para que el sensor mida la temperatura de una sustancia que testeó y me muestre su estado (sólido, líquido o gas), simulando el movimiento de sus moléculas. Para ello, utilizo un único objeto, llamado "Molécula", y 4 escenarios o fondos. En las siguientes imágenes se observa el objeto molécula en el escenario inicial y los 4 escenarios del programa:



Escenarios del programa

Cada escenario se llama mediante mensajes. Es decir, al recibir el mensaje indicado (sólido, líquido o gas), cambia el fondo del escenario al que se ha programado:



Script del escenario

El objeto molécula es el que lleva el peso del programa:

Inicialmente no busco que se muestre la temperatura que está midiendo el sensor de temperatura sumergible conectado al *Banco2* de un adaptador RJ25 que se une al *Puerto 4* de la placa del mBot, pero si quiero que tome la primera muestra para que sus medidas se vayan regulando:



Sensor de temperatura conectado al Puerto 4

Tras hacer clic en el objeto “Molécula” (ya han pasado 4 segundos), se muestra el valor numérico de la variable temperatura y el programa testea si se encuentra en estado sólido, líquido o gas, a través de diferentes intervalos de temperatura. Encontrado su rango de temperatura, envía el mensaje correspondiente (mensaje que recoge el escenario). Si fuéramos estrictos y para el “agua”, sus temperaturas serían 0°C y 100°C y no 10° y 100°, que es lo que he utilizado:



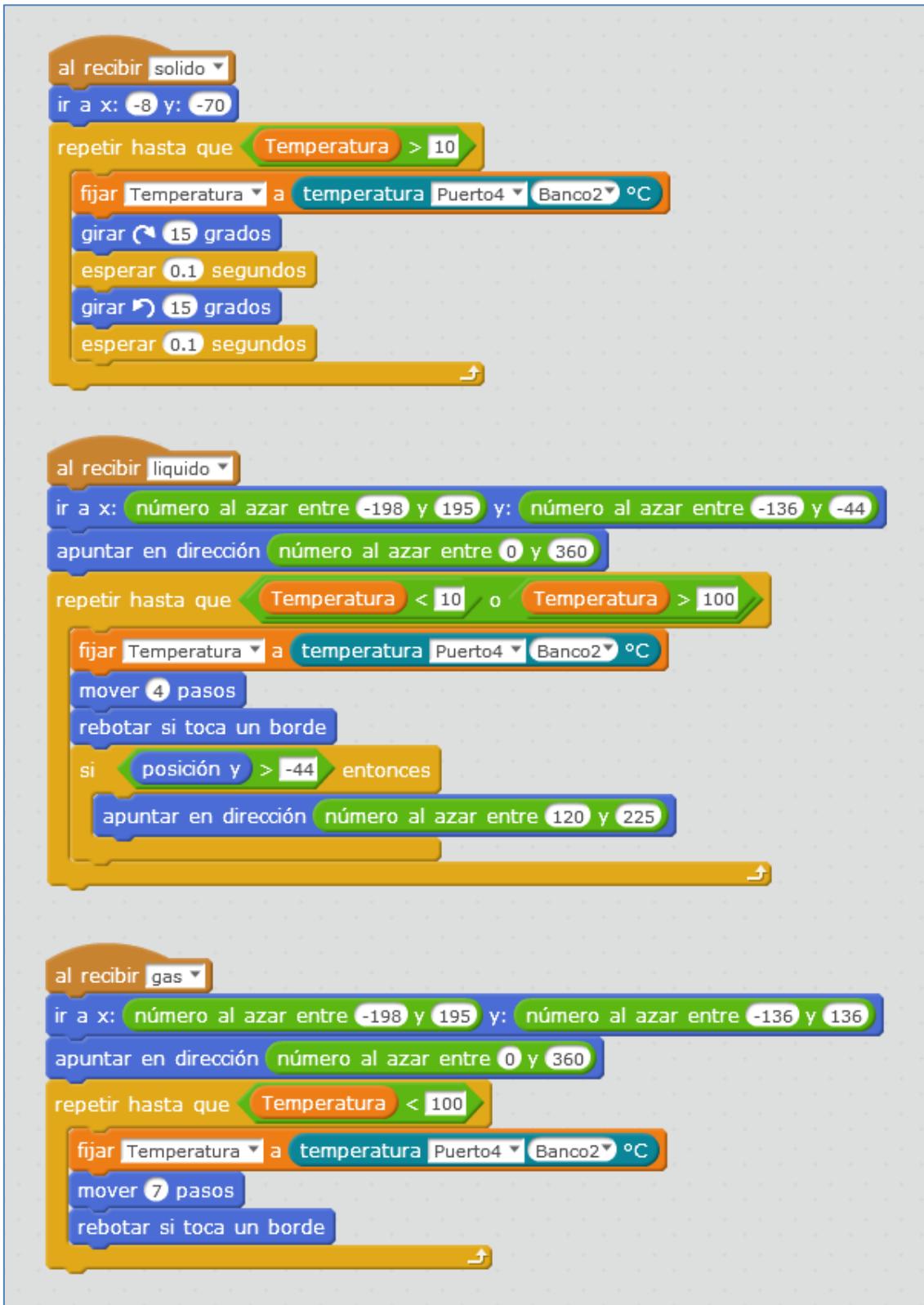
Los mensajes “sólido, líquido y gas” realizan la simulación en el movimiento de la molécula.



Situación “Líquido”

Divirtiéndome con mBot: Guía de manejo y programación

En todo momento, se testean cambios en la variable temperatura, y de este modo nos aseguramos que realiza la animación de su “estado” correspondiente:



La animación “sólido” simula que la molécula vibra, moviéndose 15° a la derecha e izquierda. La animación “líquido”, hace que la molécula se mueva a una determinada

velocidad (4 pasos) en la zona del líquido (sin sobrepassar el límite del dibujo del agua) y rebotando si toca el borde:

Finalmente, la animación “Gas”, hace que la molécula se mueva más rápido y por todo el escenario, rebotando si toca un borde del mismo.

4.10. Sensor Me Temperatura y Me Humedad

La casa Makeblock dispone del sensor DHT11, que es capaz de medir la temperatura (NTC) y humedad (resistiva) ambiente. Sus especificaciones son las siguientes:

- 5V DC
- Rango temperatura: 0°C a 50°C ± 2°C (sólo T^a positiva)
- Rango Humedad: 20% a 90% ±5%
- Precisión: 1% humedad, 1° Temperatura
- Compatible con cualquier placa Arduino



Sensor de temperatura y humedad

Este sensor nos proporciona una salida digital calibrada. Su color amarillo en el RJ25 significa que tiene un puerto simple digital y necesita estar conectado al puerto con ID amarillo en Makeblock, aunque, también puede conectarse a una placa Arduino Uno a través de sus tres conexiones GND, 5V y Data, siendo Data un pin digital como por ejemplo el pin3.

En el siguiente ejemplo voy a usar el shield de Makeblock para una Arduino Uno, conectando el sensor de humedad y temperatura al puerto7. En las librerías disponemos de 2 archivos para el sensor de humedad. Usaré el *test 1*, que es el que nos proporciona la humedad y temperatura ambiente en °C (el *test2* nos lo aporta en diferentes magnitudes para la temperatura). En el *test 1* realicé unas pequeñas modificaciones relacionadas con la placa y el puerto:

Divirtiéndome con mBot: Guía de manejo y programación

```
MeHumitureSensorTest1 Arduino 1.6.8
Archivo Editar Programa Herramientas Ayuda
MeHumitureSensorTest1 §

/*
 * \par Copyright (C), 2012-2016, MakeBlock
 * @file    MeHumitureSensorTest1.ino
 * @author  MakeBlock
 * @version V1.0.0
 * @date    2015/09/07
 * @brief   Description: this file is sample code for humidity sensor device.
 *
 * Function List:
 * 1. void MeHumiture::update(void)
 * 2. uint8_t MeHumiture::getHumidity(void)
 * 3. uint8_t MeHumiture::getTemperature(void)
 *
 * \par History:
 * <pre>
 * <Author>      <Time>        <Version>        <Descr>
 * Mark Yan       2015/09/07     1.0.0          rebuild the old lib
 * </pre>
 */
#include "MeShield.h" ← Cambio MeOrion.h por MeShield.h

MeHumiture humiture(PORT_7); ← En mi caso, el
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  humiture.update();
  Serial.print("Humidity (% ) =");
  Serial.print( humiture.getHumidity() );
  Serial.print(", Temperature (oC) =");
  Serial.println( humiture.getTemperature() );
  Serial.println("########################################");
  delay(1000);
}
```

En la siguiente imagen, referida al Monitor Serie, podemos ver que nos aporta la temperatura y la humedad ambiente:

Divirtiéndome con mBot: Guía de manejo y programación

The screenshot shows the Arduino IDE interface. On the left, the code for 'MeHumitureSensorTest1' is displayed, which includes comments about the library version and usage. It defines a 'MeHumiture' object and prints humidity and temperature values to the serial port. On the right, the 'Serial Monitor' window titled 'COM10 (Arduino/Genuino Uno)' shows the output of the code, with data lines appearing every few seconds. The bottom status bar indicates the connection is 'Arduino/Genuino Uno en COM10'.

```
MeHumitureSensorTest1:6.0.8
Archivo Editar Programa Herramientas Ayuda
MeHumitureSensorTest1.ino
Autor: MaxBlock
Versión: 1.0.0
Estado: 2013/09/07
Resumen: Descripción: this file is sample code for humidity sensor device.

* Function List:
* 1. void MeHumiture::update(void)
* 2. uint8_t MeHumiture::getHumidity(void)
* 3. uint8_t MeHumiture::getTemperature(void)
*
* * User History:
* <prev>
* <author> Cline <version> <desc>
* Mark Tan 2013/09/07 1.0.0 remains the old lib
* </prev>
*
* #include "MeShield1.h"
MeHumiture humiture(PORT_7);
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  humiture.update();
  Serial.print("Humidity: (");
  Serial.print(humiture.getHumidity());
  Serial.print(",");
  Serial.print(" Temperature (");
  Serial.print(humiture.getTemperature());
  Serial.print(") ");
  Serial.println("*****");
  delay(1000);
}

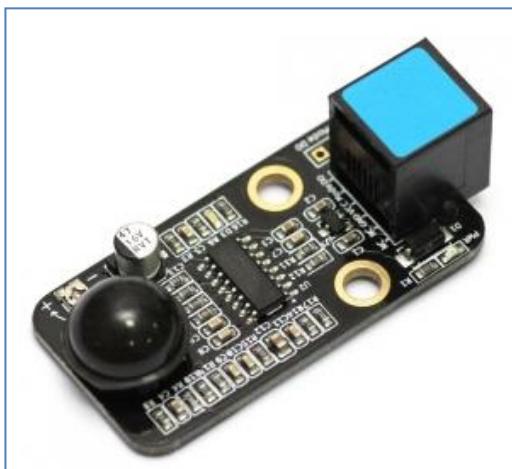
Sobreto
Opciones de compilación cambiadas, reconstruyendo todo
El sketch usa 4.151 bytes (1%) del espacio de almacenamiento de programa. El máximo es 32.256 bytes.
Las variables globales usan 741 bytes (3%) de la memoria dinámica, dejando 1.107 bytes para las variables locales. El máximo es 2.048 bytes.
15:33
09/05/2016
```

4.11. Módulo PIR

Es un sensor pasivo que capta la luz infrarroja del ambiente y reacciona a sus cambios. Nos servirá para detectar el movimiento y básicamente lo que hace es crear y transmitir un campo de luz infrarroja que choca con los objetos de la habitación logrando detectar los cambios en el eco que recibe. Al alimentarlo eléctricamente proporciona una salida digital (cero o uno), que puede ser regulada en sensibilidad mediante un potenciómetro. Su ángulo de visión es de 90º y su radio de alcance se sitúa entre 6 y 9 metros.

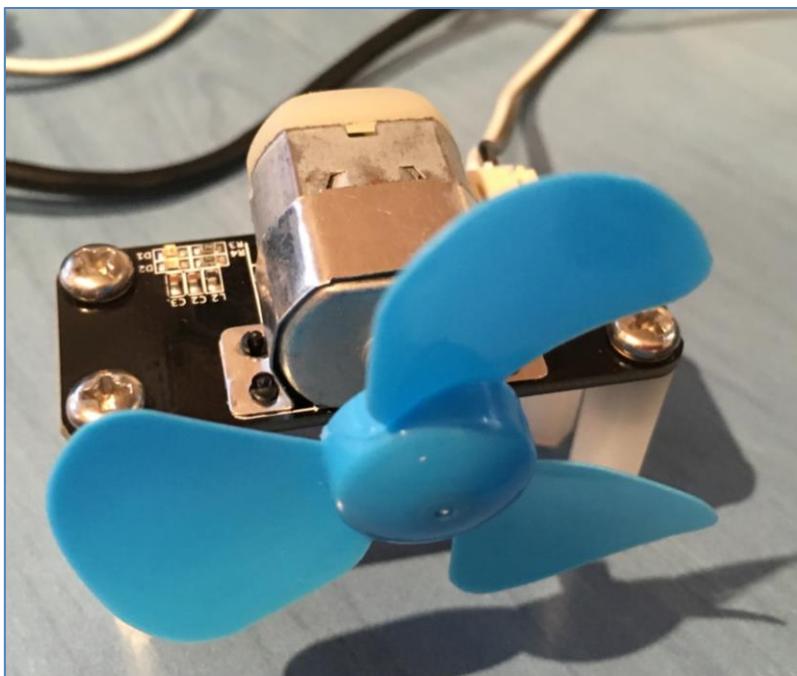
El sensor PIR de movimiento de la casa Makeblock se usa para detectar personas o animales en un rango de hasta 6m. Si se mueve algo dentro de ese rango de distancia, el sensor activa la salida digital SIG a alto. Mediante un potenciómetro, soldado en el módulo, podremos ajustar el rango de detección.

Importante: Después de alimentarlo, hay que esperar unos 10 segundos a que el sensor se inicialice.



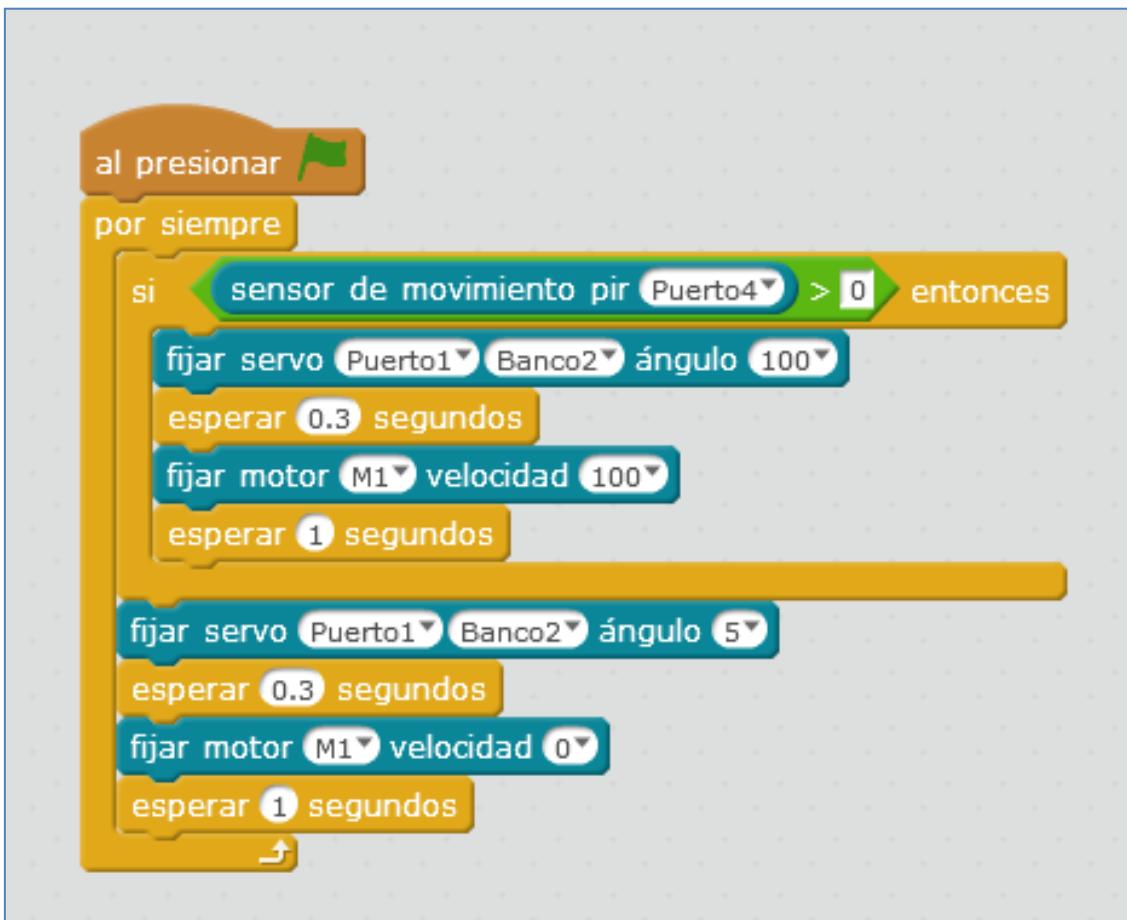
Sensor Me PIR

El siguiente ejemplo se implementa sobre una placa Makeblock Orion utilizando un módulo sensor Me PIR, un módulo servo motor con el módulo adaptador RJ25 y un módulo 130 DC motor con hélice y que en la siguiente programación se conecta a M1. El módulo PIR se conecta al puerto 4 y el servo, mediante el "Banco2" del adaptador, al puerto 1.



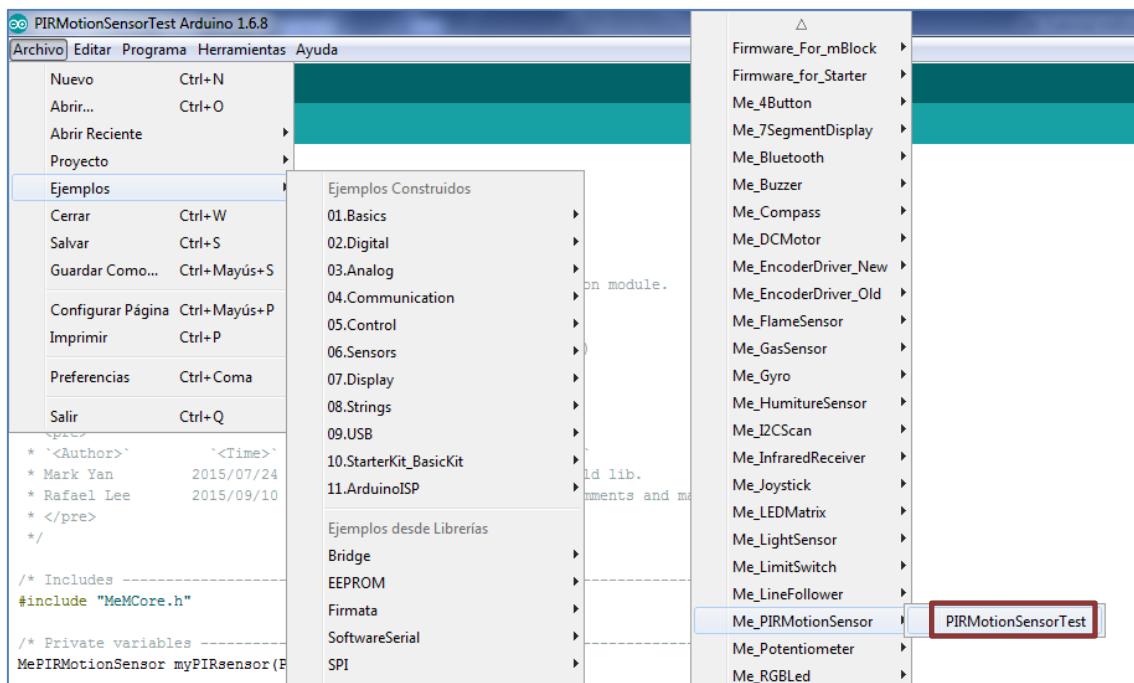
Módulo 130 DC con hélice

Cuando el sensor PIR detecta el movimiento de un cuerpo, hace girar el servo y el motor M1. En caso contrario, ambos se paran:

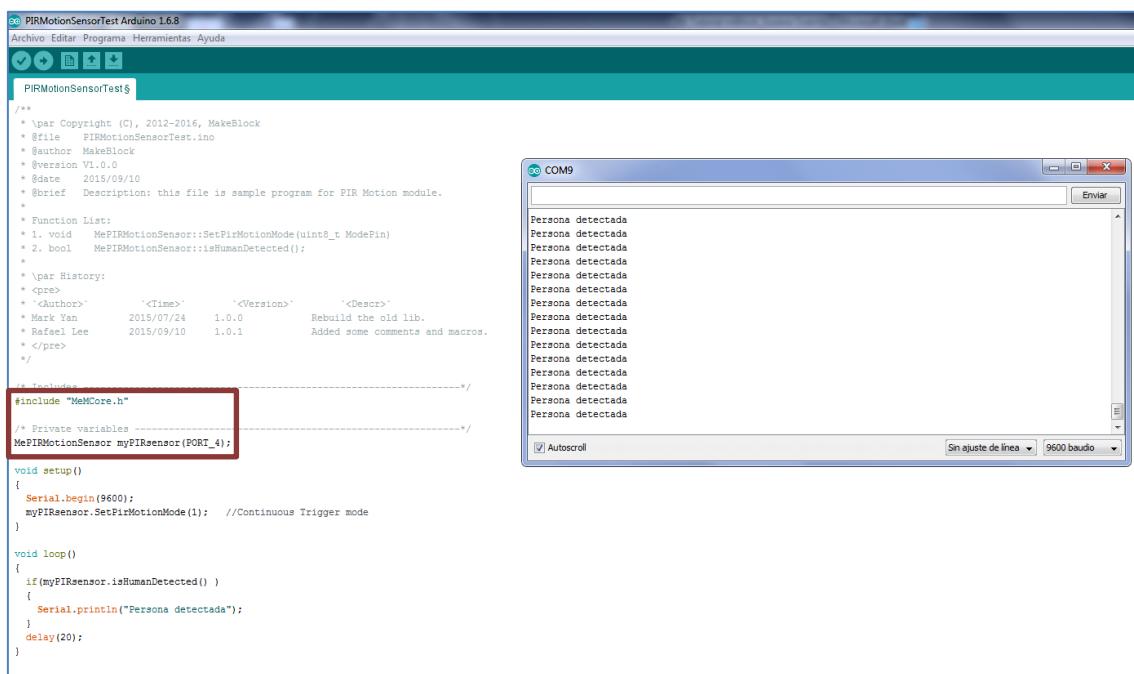


Divirtiéndome con mBot: Guía de manejo y programación

Dentro de los ejemplos que nos proporciona la casa Makeblock en sus librerías de Arduino nos encontramos con el archivo *PIRMotionSensorTest*. Este archivo se ha creado para que el sensor nos informe, mediante el puerto Serial Begin, sobre la existencia de una persona en la habitación:



Para comprobarlo utilizaré la placa mCore conectando el sensor PIR al Puerto 4 (ID azul) de la misma. Después cargaré el programa a la placa desde el IDE de Arduino. El resultado es obvio, habrá detección porque yo estoy presente, tal y como se muestra en la siguiente imagen:



4.12. Módulo Joystic

El módulo joystic puede usarse para, por ejemplo, simular el movimiento de un objeto por el escenario. De hecho, no es más que una palanca de control de mando. En el siguiente ejemplo, se han creado 2 variables, referidas a las coordenadas X e Y que nos ofrece el módulo joystic.



Se observa que, inicialmente, el módulo ofrece las cantidades numéricas -2 y -18 para los ejes X e Y, respectivamente. También sabemos que, en el escenario scratch, nuestra X varía entre -240 y 240 y la Y, entre -180 y 180. Así pues, tomando los siguientes puntos para ambos ejes, podemos calcular los parámetros m y n de la función a programar:

X	Y=mx+n	X	Y=mx+n
-2	0	-18	0
240	490	490	180
Eje X	Eje Y		

Con estos puntos, los valores de los parámetros “m” y “n” en el eje X y en el eje Y son, respectivamente: 0,492 y 0,984 para el eje X y, 0,354 y 6,378 para el eje Y. Finalmente, el script para programar el joystic que está conectado al puerto 7 que consiga que el sprite se mueva por el escenario, es el siguiente:



4.13. Módulo Me Touch o sensor táctil

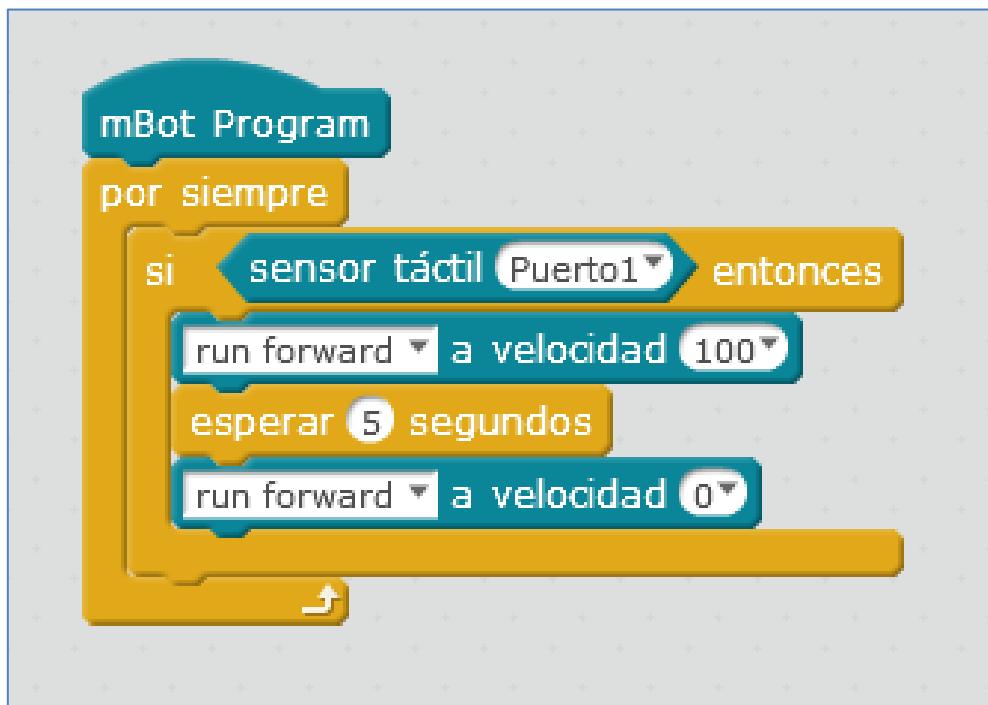
Un sensor táctil es un dispositivo que recibe y responde a una señal o estímulo que se asocia con la fuerza que recibe. El sensor Me Touch es un sensor táctil de tipo capacitivo, de modo que, si un dedo (o cualquier otro objeto con propiedades capacitivas) se aproxima a este sensor táctil, hace que el sensor funcione como un condensador. La naturaleza dieléctrica del sensor hace variar la capacidad del sensor para detectar el contacto táctil.



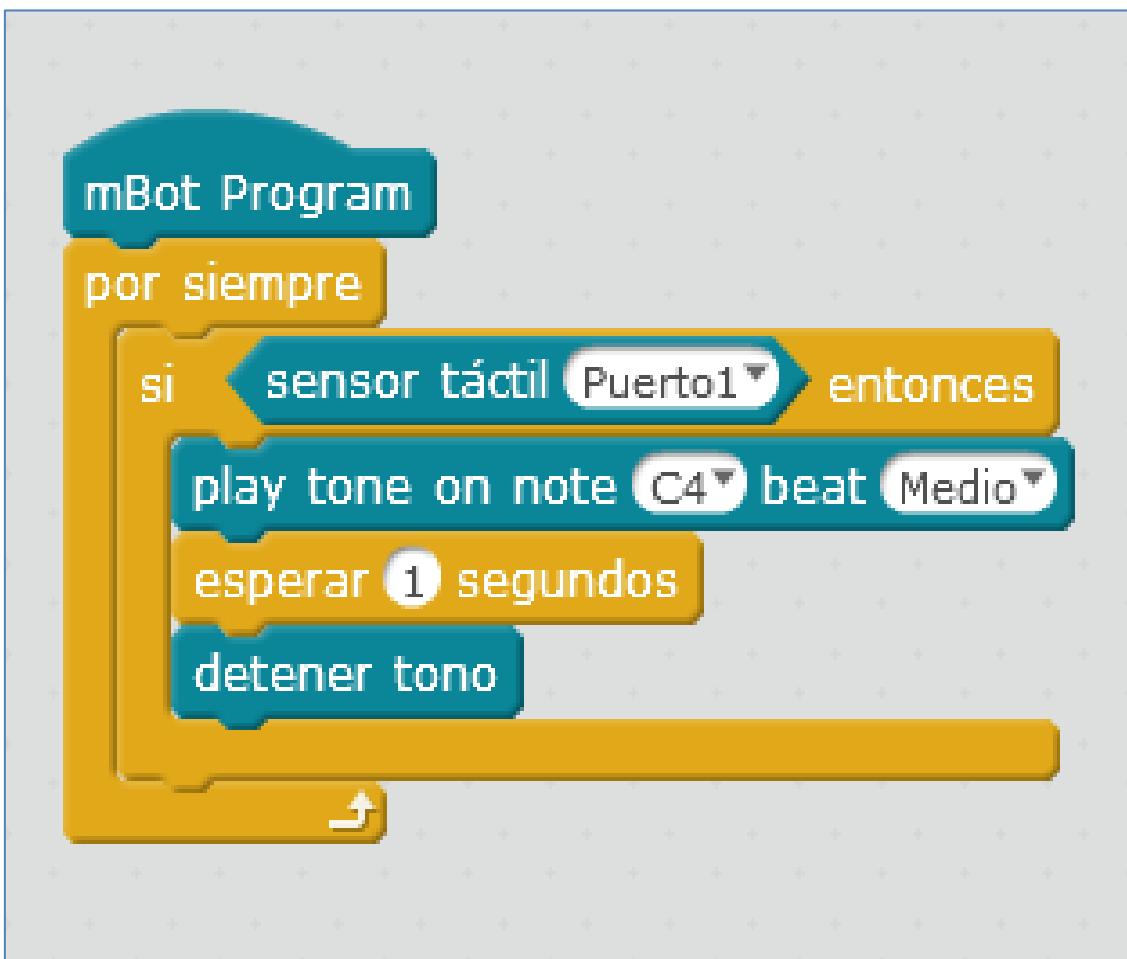
Sensor Me Touch

En cuanto a su programación, es importante saber que el sensor debe ejecutar el programa cargándolo a la placa al mBot. Por lo tanto, no podemos usar “Al presionar la bandera verde”. En su lugar, usaremos el comando del robot “[mBot Program](#)”.

En ejemplo podría ser el siguiente: cuando toquemos el sensor, el mBot comenzará a andar durante 5 segundos, para después pararse:



También podría sonar la nota C4 durante 1 segundo:



4.14. Mini Gripper

Podemos implementar y programar una garra robótica en el mBot. La que vemos en el siguiente link está construida con piezas de LEGO:

[Garra robótica con Lego](#)

Por cuestiones de peso, la garra no puede tener un gran tamaño si queremos, obviamente, incluirla al mBot. La casa Makeblock ha sacado al mercado dos garras robóticas: Una enfocada hacia el kit Starter y otra para el mBot. Ambas se diferencian en tamaño y en potencia de la placa para la que han sido diseñadas. Vamos a comentarlas:

La garra robótica que se ve en la siguiente imagen funciona como cualquier motor DC convencional, por lo que, para programarla, sólo hay que hacer avanzar o retroceder el motor a la velocidad deseada. En principio, en la placa mCore no se puede acoplar porque se supone que tenemos utilizada la conexión para los motores de sus ruedas. Para poderla utilizar deberíamos cambiar esta placa por una superior, que disponga de conexión para más motores. En cuanto a la conexión, la placa del mBot sólo tiene canales para 2 motores y éstos están ocupados por los motores de las ruedas. Se

Divirtiéndome con mBot: Guía de manejo y programación

podría “hacer un apaño”, por así decirlo, sacando tensión por uno de los pines de la placa para usarlo con algún driver de motor estándar de Arduino.



Garra robótica para Starter Robot Kit

En cambio, la garra robótica de la siguiente figura si puede acoplarse al mBot, por potencia y por peso. Se denomina Mini Gripper y tiene incorporado un servomotor. Por lo tanto, necesita de un adaptador RJ25 que lo conecte a la placa del mBot.



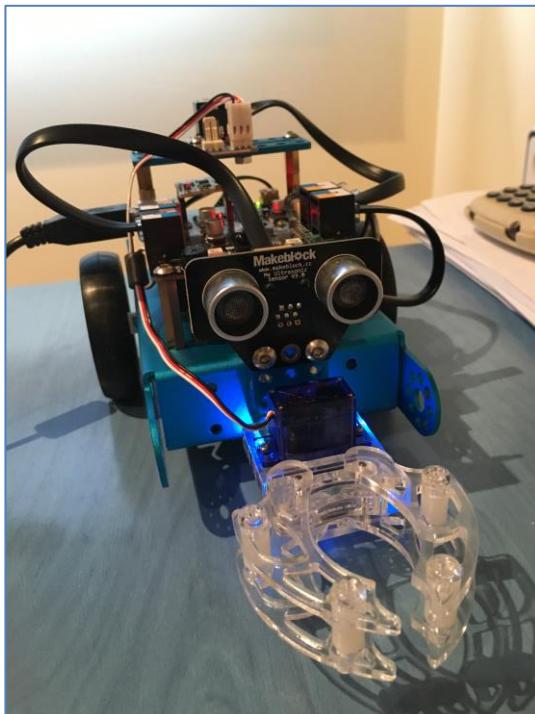
Mini Gripped o garra robótica para mBot

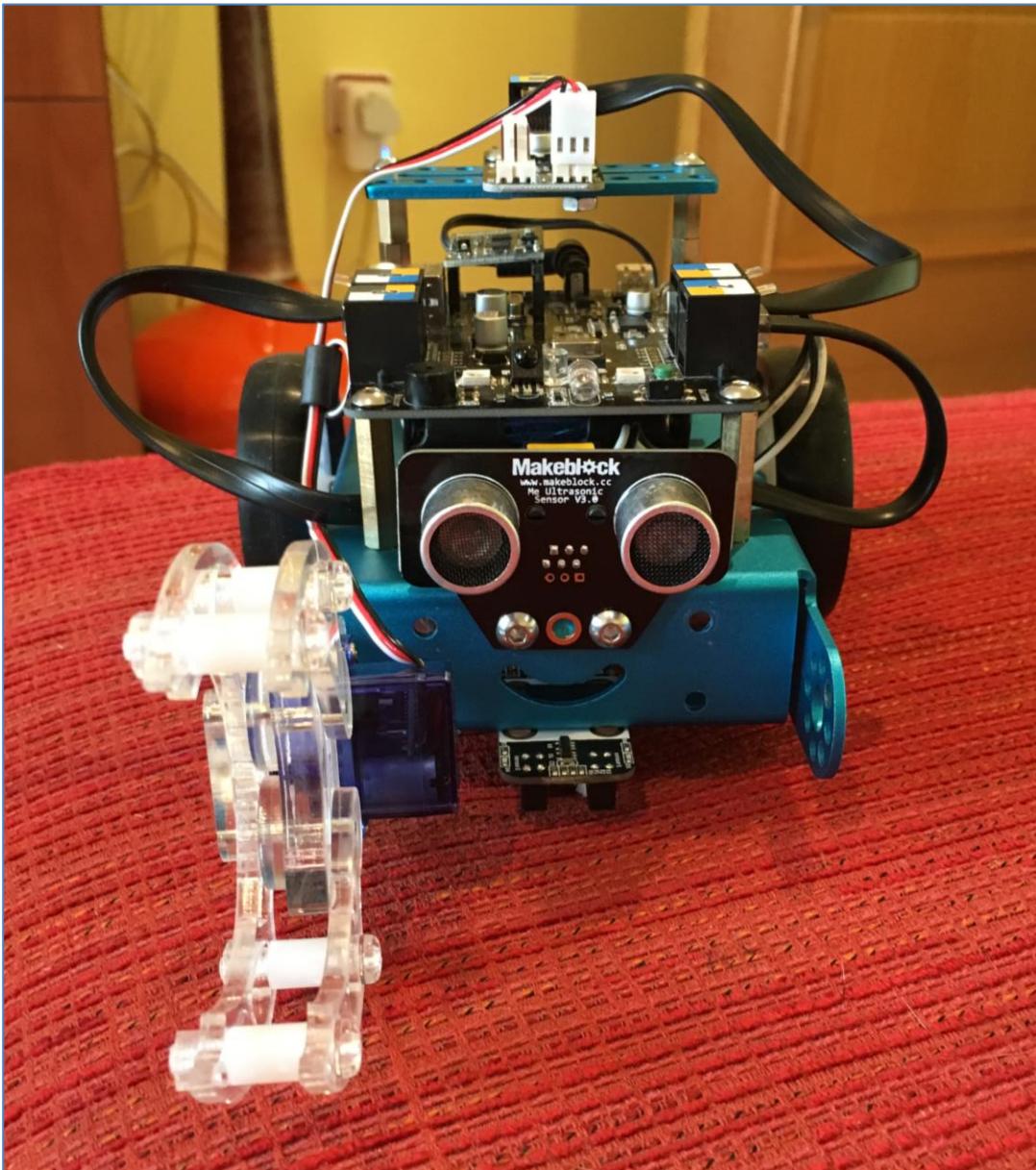
Divirtiéndome con mBot: Guía de manejo y programación

Su programación es muy sencilla en el entorno mBlock ya que sólo debemos usar el comando “[Fijar servo_Puerto correspondiente_Banco correspondiente_con el ángulo de giro deseado](#)”. Por ejemplo, para abrir la garra fijaremos el ángulo a 0º y para cerrarla, le indicaremos que lo gire a 90º. Los ángulos posibles del servo varian entre 0 y 180º:



Algunas posibilidades de montaje con el mBot pueden verse en las siguientes imágenes:

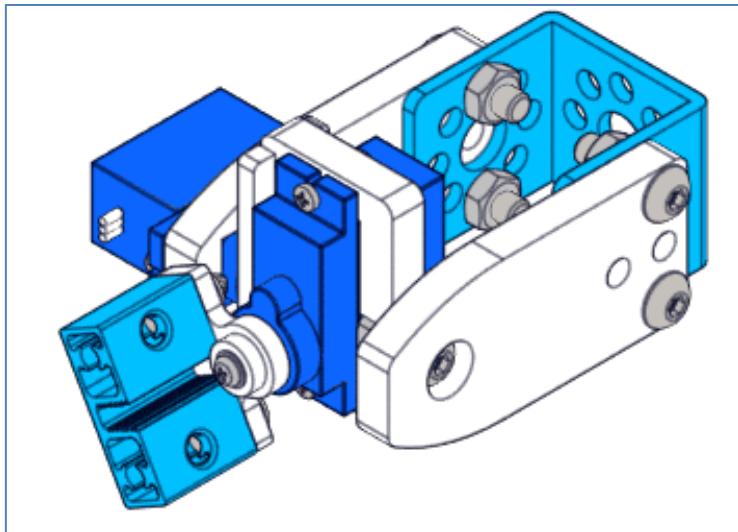




En este último montaje hay que tener cuidado con el sensor de ultrasonidos. Su detección no es lineal; es más bien cónica. Por esta razón, hay que hacer pruebas para situar la garra fuera del cono de detección del sensor de ultrasonidos para que no la detecte y detecte a otro objeto que se sitúe en el frontal del mBot.

4.15. Mini brazo articulado

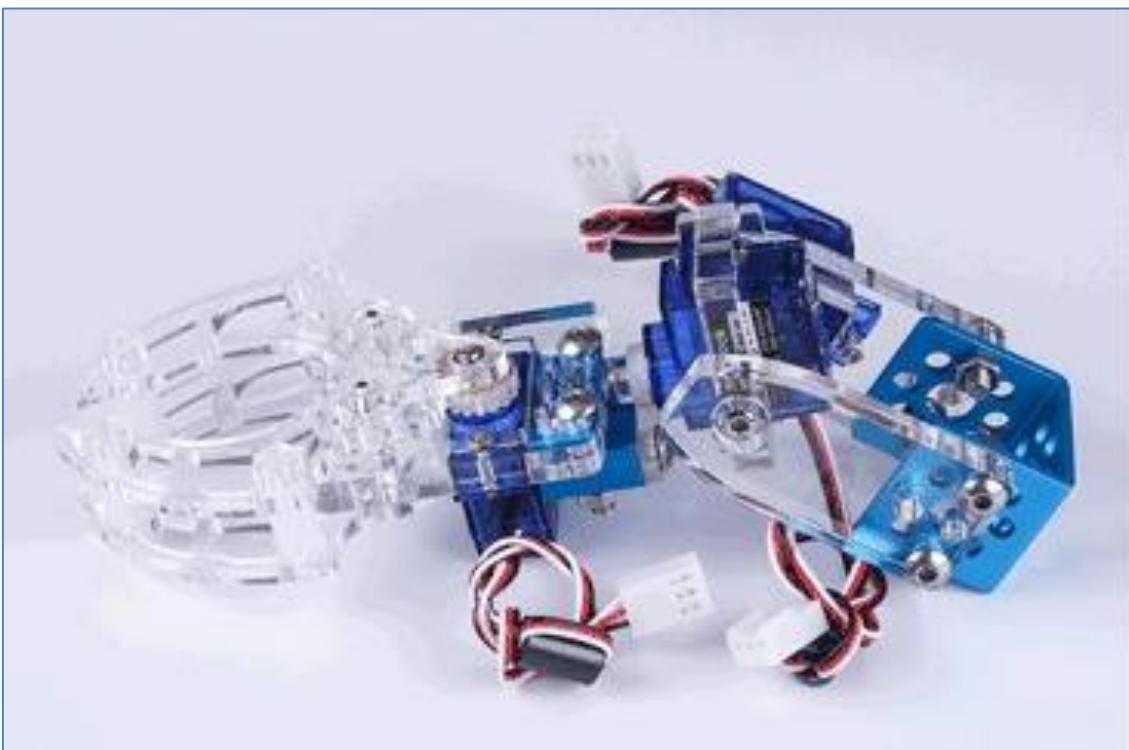
El mini brazo robótico articulado está formado por dos microservos 9g que nos permiten rotar la estructura 180 grados, tanto en vertical como en horizontal. Está pensado para ser usado con el mBot, el Starter Kit, el mBot Ranger o cualquier otro proyecto sobre una placa Arduino, pudiendo combinarse con el módulo Mini Garra del punto anterior o incluso con un sensor de ultrasonidos.



Mini brazo articulado

Podemos programar los dos servos del mini brazo articulado de forma sencilla con Scratch o con el IDE de Arduino. Con mBlock cada servo utiliza un “Banco” del módulo Adaptador RJ25.

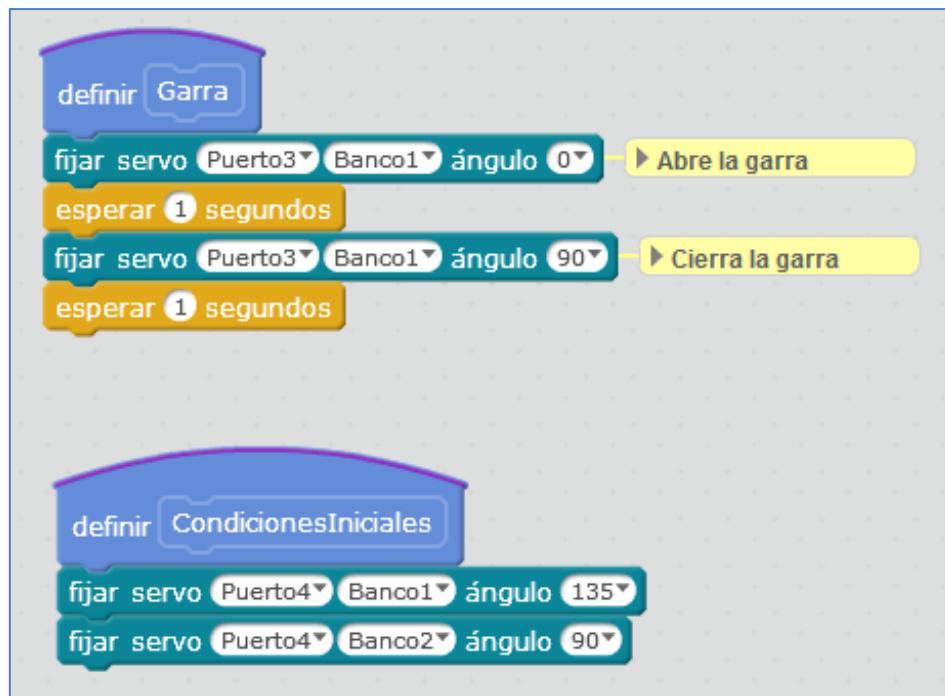
Si a mayores le incluimos la mini garra robótica, lo que conseguimos es el montaje de la siguiente imagen. Simplemente, estamos añadiendo un servo más, por lo que necesitamos echar mano de otro Adaptador RJ25:



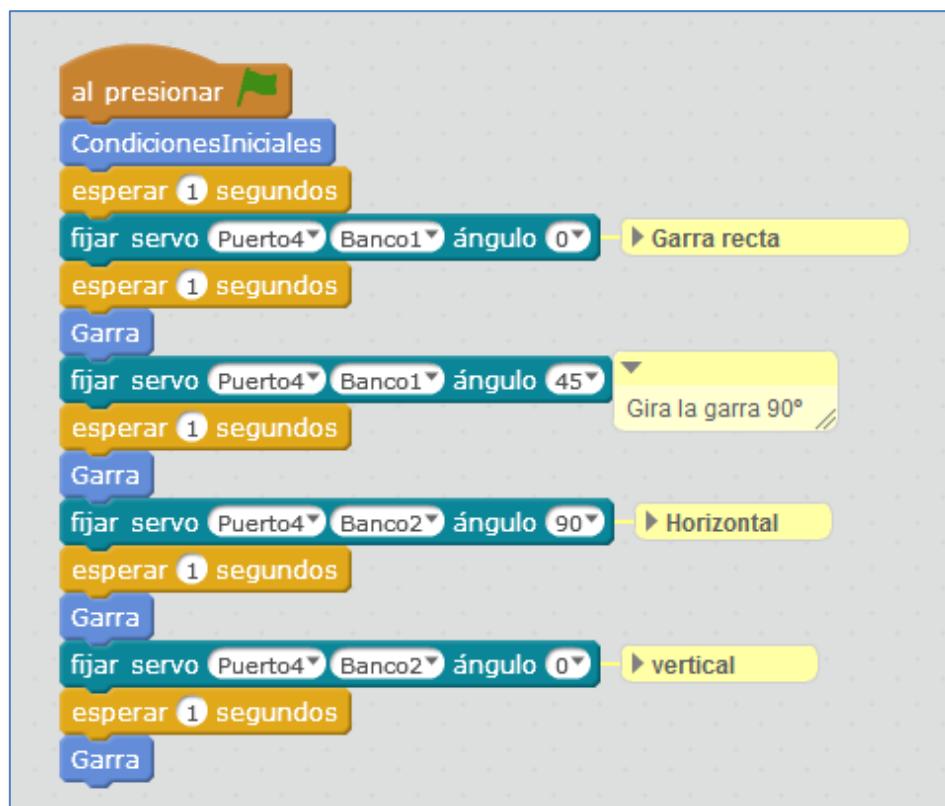
Un ejemplo para su programación con mBlock podría ser el siguiente: Notar que en el ejemplo, el servo de la garra está conectado al *Banco 1* del adaptador que se asocia al puerto 3 y los otros dos servos cubren ambos *Bancos* del adaptador conectado al puerto 4 de la placa mBot.

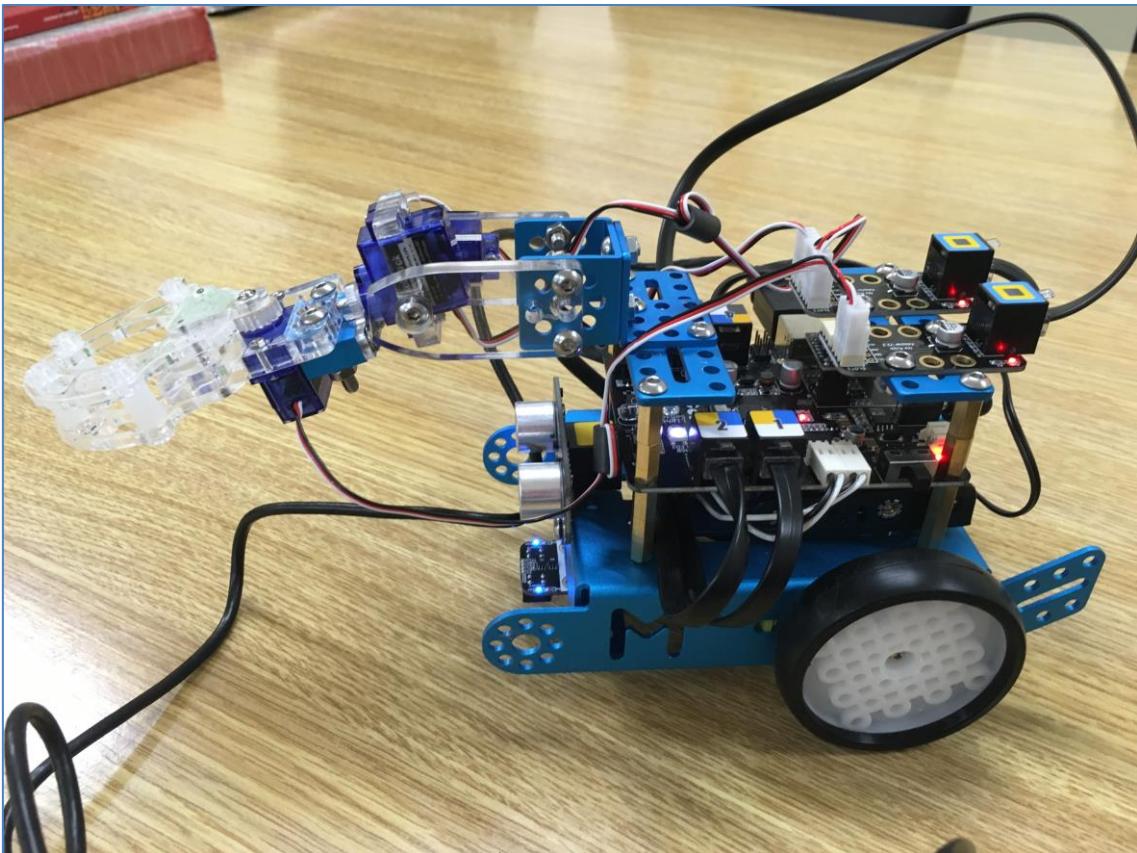
Divirtiéndome con mBot: Guía de manejo y programación

En el ejemplo, creo dos bloques. Uno para el movimiento de la garra, y que defino con el bloque “Garra”, y otro para las condiciones iniciales de comienzo en los otros dos servos que controlan el mini brazo giratorio, y que denoto por “*CondicionesIniciales*”:



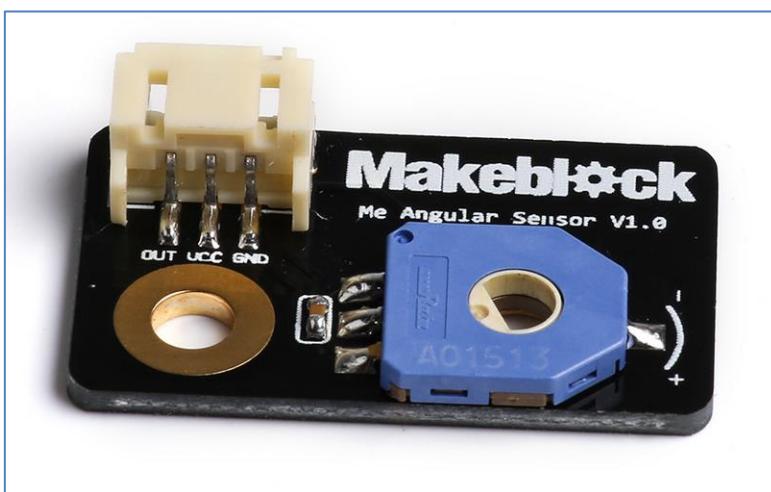
Finalmente, sólo resta hacer el programa en si. El mio es muy sencillo y sólo muestra algunos movimientos de ambos servos del puerto 4. En cada giro programado, se abre y cierra la garra:





4.16. Sensor de ángulo

Un sensor de ángulo está diseñado para detectar el ángulo de inclinación de un mecanismo que esté unido a él. Nuestro sensor de ángulo necesita de un módulo adaptador RJ25.



Sensor de ángulo Makeblock

El sensor devuelve un valor analógico, al igual que el sensor de temperatura, así que, ambos podemos programarlos de la misma forma. Este programa, en este caso para una placa Orion, nos valdría para cualquier sensor de Arduino que nos proporcione un valor analógico:

```
#include "MeOrion.h"

// Adaptador RJ25 conectado al puerto 6
MePort sensor_angular(PORT_6);

float valor;
void setup()
{
    Serial.begin(9600);
}

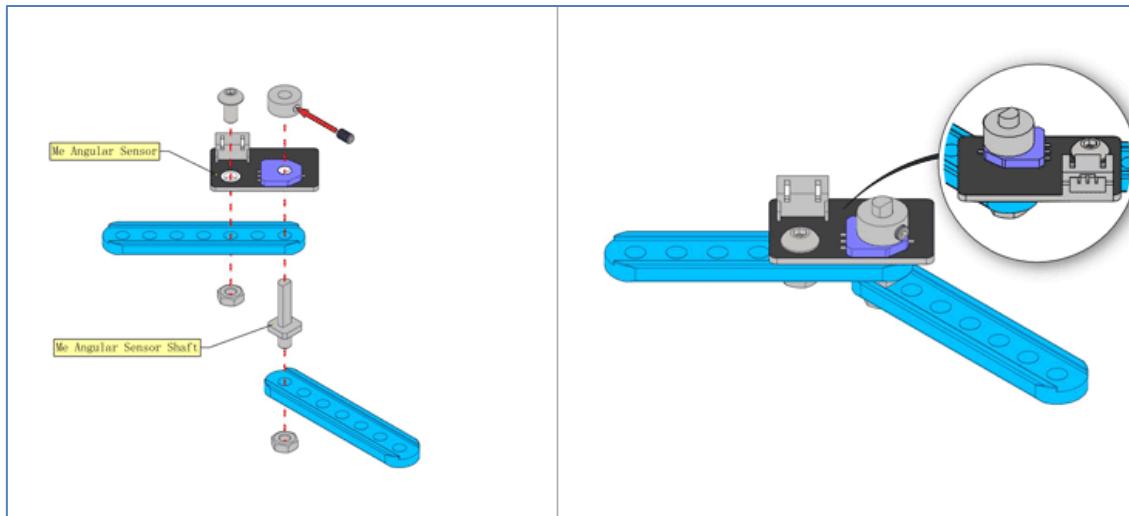
void loop()
{
    // Valor analógico del slot2 del Adaptador RJ25
    valor = sensor_angular.aRead2();

    Serial.print("Angulo=");
    Serial.println(valor);

    delay(1000);
}
```

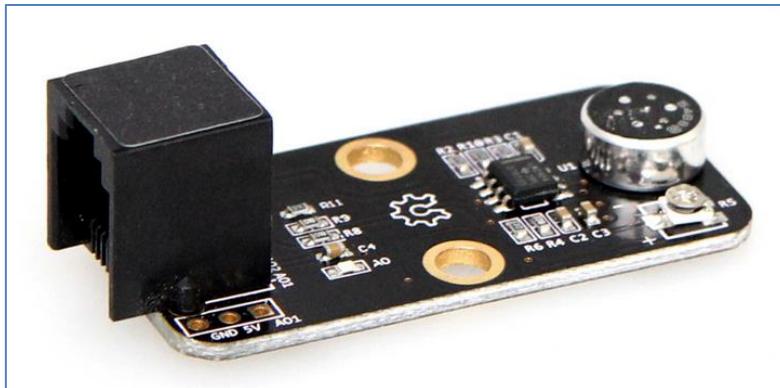
Para una placa Arduino Uno con el shield de makeblock, cambiaríamos “MeOrion.h” por “MeShield.h”.

Una forma de conectarlo es a través de un adaptador Shaft t (ver siguiente imagen):



4.17. Sensor de sonido

El sensor de sonido está destinado a detectar la intensidad de sonido de su entorno. Podemos utilizarlo para realizar proyectos que requieran que interactuemos con la voz. El sensor dispone de un amplificador LM386 e incorpora un potenciómetro de ajuste de sensibilidad, proporcionándonos valores analógicos entre 0 y 1023.



Sensor de sonido

El color ID negro del sensor en su RJ25 nos informa que podemos conectarlo a cualquier RJ25 de las placas o shield con este color. Es decir, en la mCore nos valdría el puerto 3 y 4 y en el shield de Arduino Uno el puerto 7 u 8. Además, podremos conectarlo directamente a una placa Arduino Uno usando sus conexiones laterales GND, 5V y AO1 siendo esta última una salida analógica de la placa Arduino Uno, como por ejemplo el pin A0.

El archivo *SoundSensorTest* de la Librería de Makeblock nos muestra cómo podemos programarlo de forma simple en Arduino. Usando el shield de Makeblock para una Arduino Uno, situando el sensor en el puerto 7, podemos observar que en el puerto serial se nos muestra un valor numérico elevado cuando alzo la voz y este valor baja cuando estoy callada:

```
/*
 * Copyright (C) 2012-2016, MakeBlock
 * File: SoundSensorTest.ino
 * Author: MakeBlock
 * Version: V1.0.0
 * Status: 2015/09/01
 * Note: Descripción: this file is sample code for the sound sensor device.
 *
 * Function List:
 * 1. void_t_<soundSensor>(strength)
 *
 * History:
 * <original> <version> <desc>
 * Mark Tan 2015/09/01 1.0.0 rebuild the old lib
 * </pre>
 */
#include "MBlock.h"

MeSoundSensor mySound(PORT_7);

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.print("value=");
    Serial.println(mySound.strength());
    delay(100);
}

Options de compilación cambiadas, reconstruyendo todo
El Sketch usa 3.748 bytes (1%) del espacio de almacenamiento de programa. El máximo es 32.256 bytes.
Las variables Globales usan 685 bytes (3%) de la memoria dinámica, dejando 1.365 bytes para las variables locales. El máximo es 2.048 bytes.
*/
```

value=129
value=131
value=132
value=129
value=130
value=129
value=130
value=129
value=130
value=129
value=132
value=130
value=130
value=132
value=131
value=550
value=541

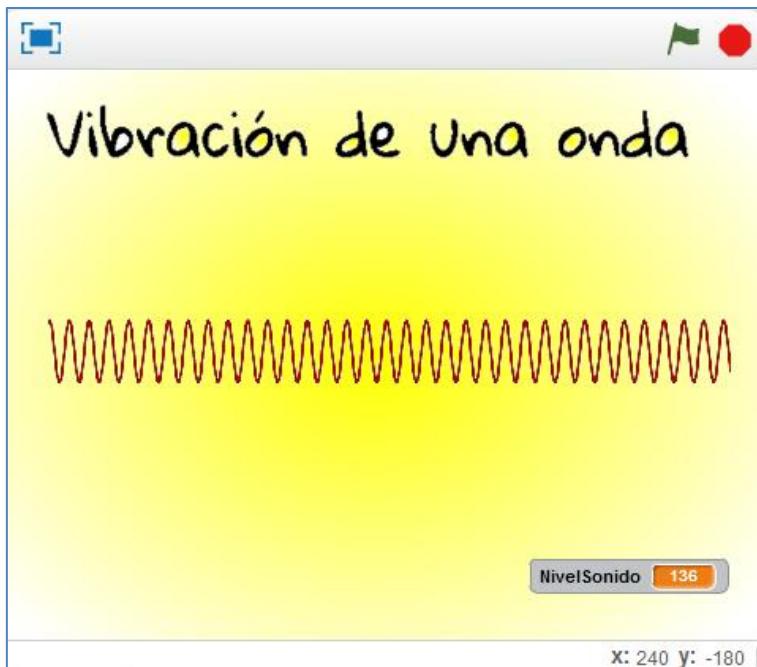


The screenshot shows the Arduino IDE interface with the title bar "SoundSensorTest Arduino 1.6.8". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". Below the menu is a toolbar with icons for file operations. The main code editor window contains the following C++ code:

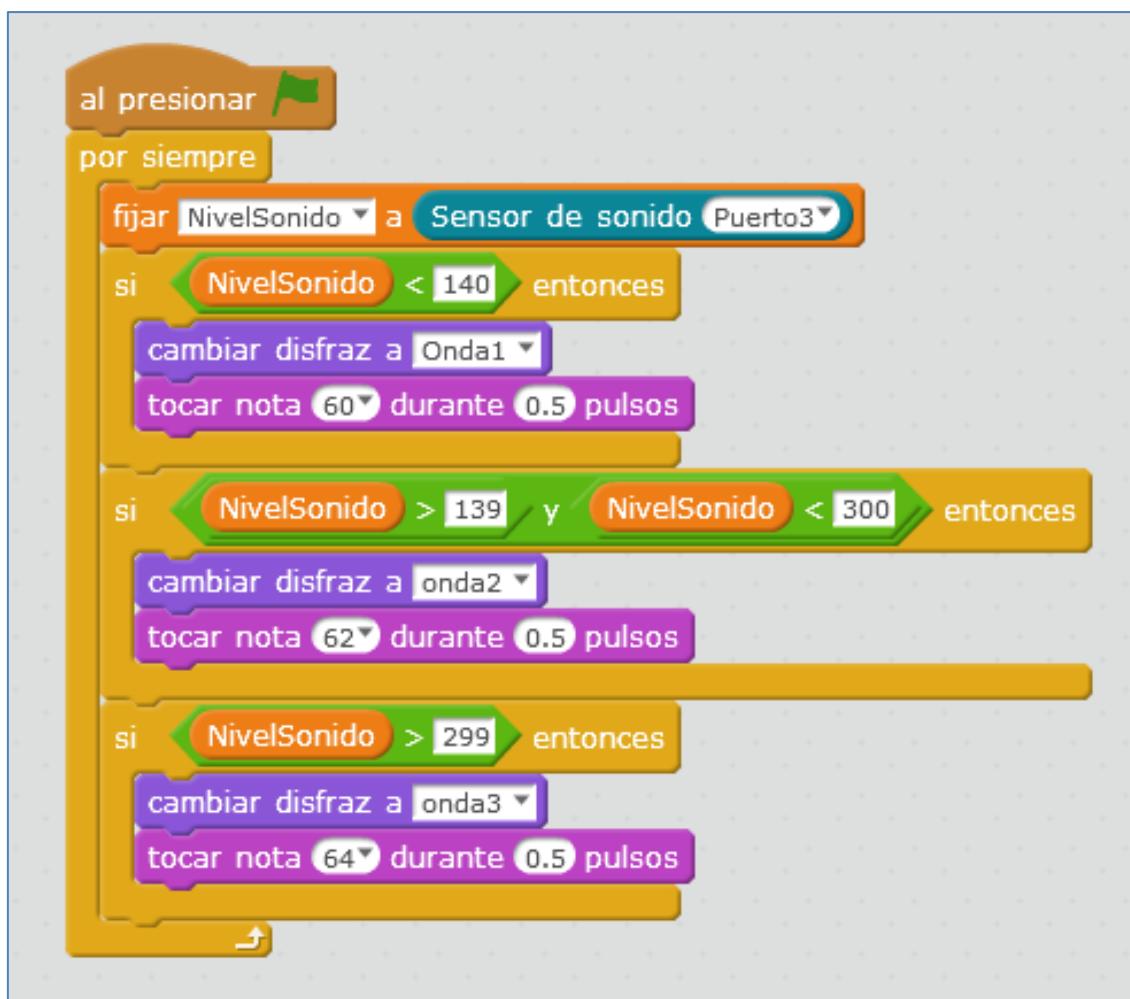
```
/**\n * \par Copyright (C), 2012-2016, MakeBlock\n * @file SoundSensorTest.ino\n * @author MakeBlock\n * @version V1.0.0\n * @date 2015/09/01\n * @brief Description: this file is sample code for Me sound sensor device.\n *\n * Function List:\n * 1. int16_t MeSoundSensor::strength()\n *\n * \par History:\n * <pre>\n * <Author>      <Time>      <Version>      <Descr>\n * Mark Yan      2015/09/01    1.0.0          rebuild the old lib\n * </pre>\n */\n#include "MeShield.h"\n\nMeSoundSensor mySound(PORT_7);\n\nvoid setup()\n{\n    Serial.begin(9600);\n}\n\nvoid loop()\n{\n    Serial.print("value=");\n    Serial.println(mySound.strength());\n    delay(100);\n}
```

Este sensor también podemos programarlo con el software mBlock, conectando el sensor de sonido a los puertos 3 o 4, que son los habilitados para este sensor (ID negro del sensor).

Supongamos que quiero simular el sonido con la vibración de una onda y aprovecharla para hacer sonar las notas musicales DO, RE y MI. En el siguiente escenario se muestra la onda sinusoidal perfecta y, sin ruido. En él, el sensor de sonido marca un valor en la variable *Nivel/Sonido* de 136 y el objeto se encuentra en el disfraz “ondas1”.

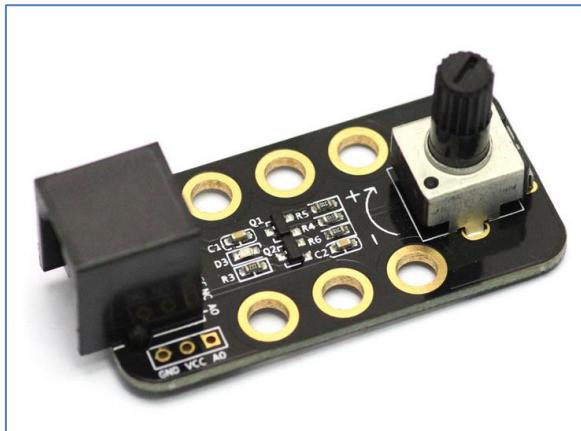


Si el nivel de sonido es menor que 140, debemos ver la onda1 y sonará la nota DO. Si el nivel de sonido entá entre 140 y 299, debemos ver la onda2 y sonará la nota RE y, finalmente, si el sonido supera 299, veremos el disfraz de ondas3 y sonará la nota MI.



4.18. Potenciómetro

El potenciómetro lineal de Makeblock es, según nos informa la casa, de $50\text{K}\Omega$. Es de 270 grados, siendo el mínimo valor hacia la izquierda y el máximo valor al tope de la derecha. Su color de ID negro nos indica que sólo podrá conectarse a los puertos 3 y 4 de la placa mCore y a los puertos 7 y 8 del shield de arduino.



Módulo Potenciómetro

Al igual que en otros módulos, si no queremos usar el shield de arduino y pretendemos conectarlo a la placa Arduino Uno, disponemos de 3 terminales: GND, VCC y AO; siendo esta última una de las salidas analógicas de la placa Arduino Uno.

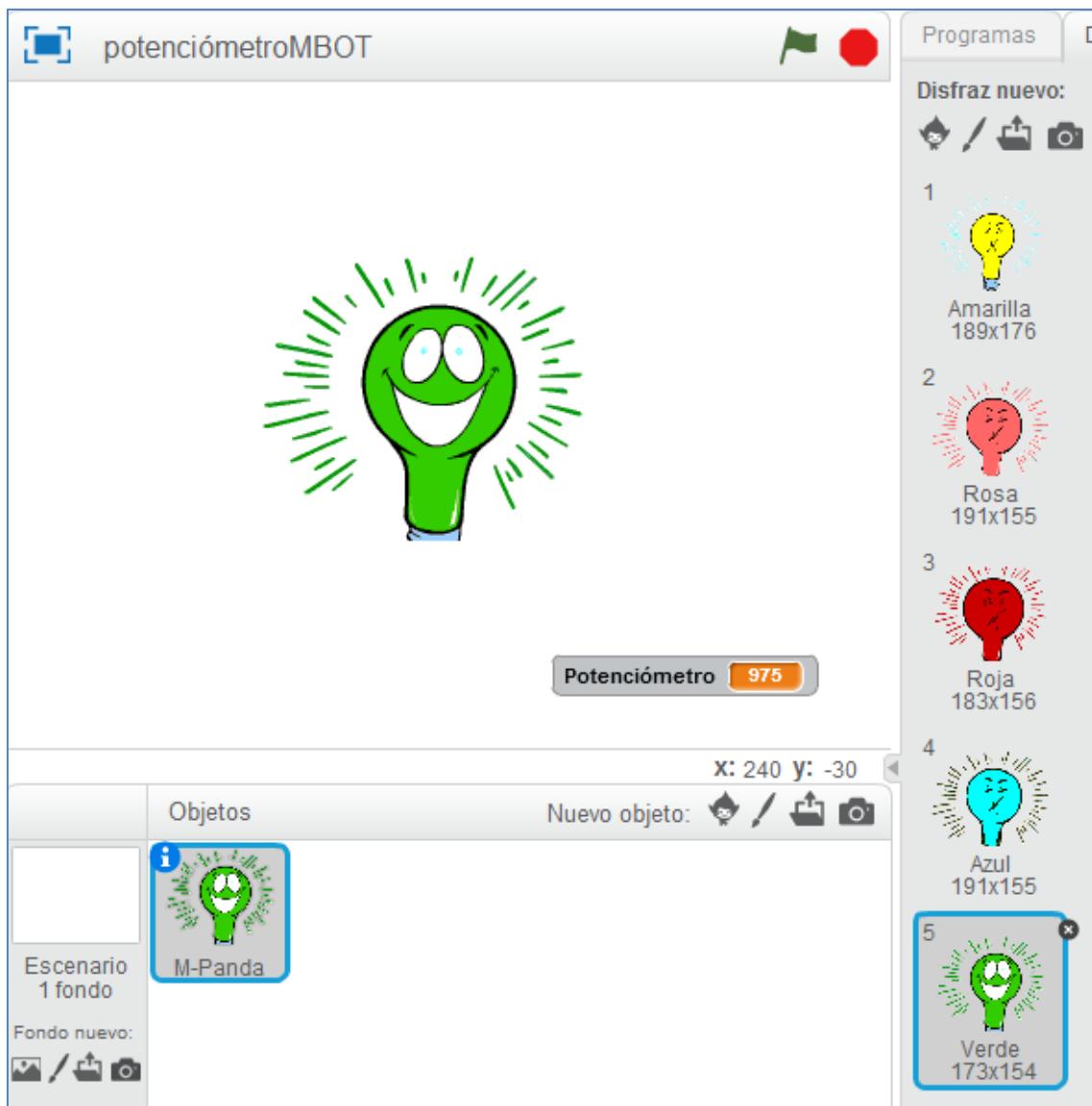
Un ejemplo que nos ayudará a programarlo en arduino lo encontramos en su librería, en el archivo *PotentiometerTest*. Simplemente, debemos modificar el nombre de la placa que usamos y el puerto en el que conectamos el potenciómetro.

También podemos programarlo con mBlock. En el siguiente ejemplo utilizo el valor numérico que me proporciona el potenciómetro para modificar el color de los dos led RGB del mBot. La siguiente imagen muestra el escenario del programa con el valor máximo del potenciómetro (girado 270 grados hacia la derecha), que resulta ser de 975.



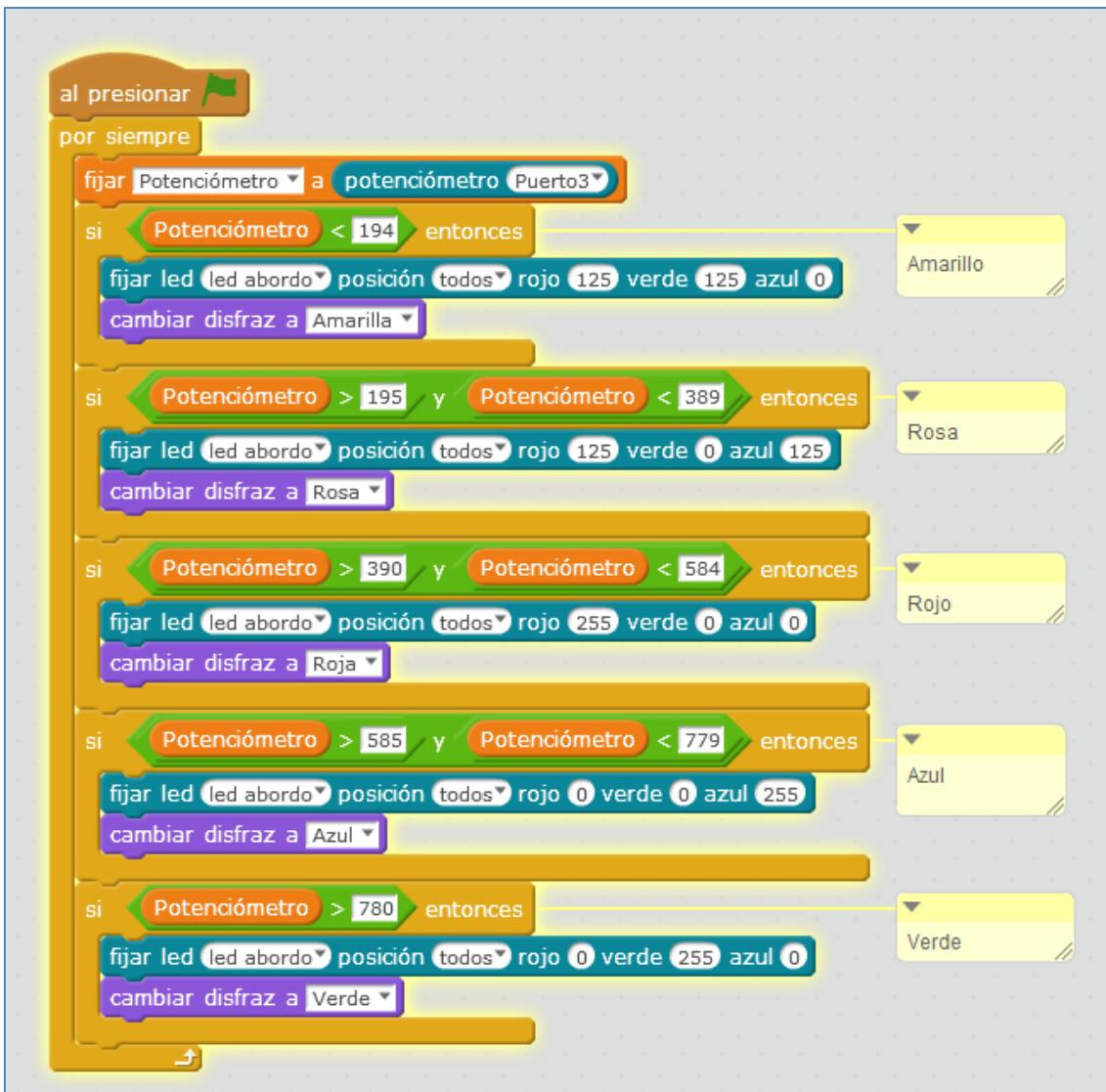
Divirtiéndome con mBot: Guía de manejo y programación

Como puede verse en la siguiente imagen, dispongo de 5 disfraces para recorrer los 975 puntos numéricos del potenciómetro. He decidido hacer intervalos “simétricos” de módulo 195 (975:5=195) y, a cada intervalo asociarle un color de LED RGB correspondiente con el color del disfraz:



Disfraces del objeto

El programa, para una placa mCore del mBot, estando el potenciómetro conectado al puerto 3, es el siguiente:



4.19. Módulo 4LEDs RGB

Con este módulo que incluye 4 LEDs RGB, podremos controlar el brillo y el color de cada uno de sus leds de forma independiente.



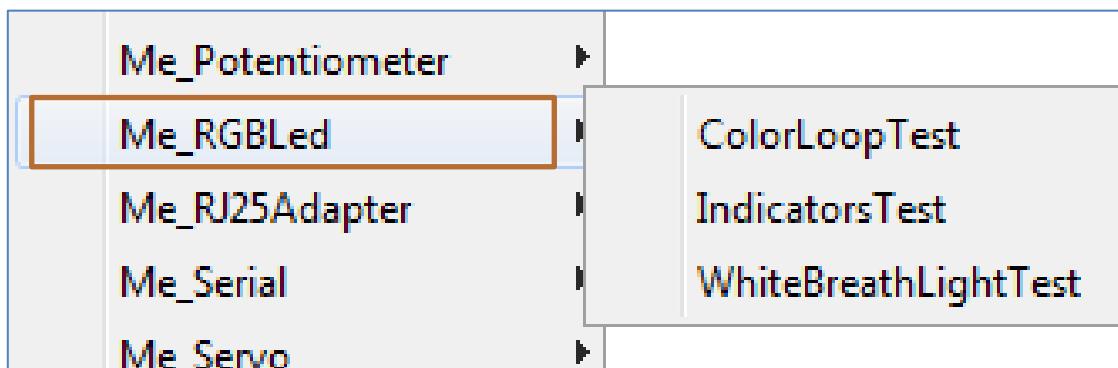
Módulo 4 LEDs RGB

Divirtiéndome con mBot: Guía de manejo y programación

Su conector es de color amarillo, así que, sólo lo podremos conectar a los puertos que dispongan de este color. En una placa mCore nos vale cualquiera de sus 4 puertos.

Al igual que en otros componentes, podemos usar este módulo directamente en una Arduino Uno, sin shield. En este caso, las tres conexiones se nos muestran en un lateral: Vcc, GND y SIG, siendo la señal un pin digital de la placa Arduino Uno.

En las librerías de Makeblok disponemos de tres programas de ejemplo que nos ayudarán a programarlos desde el IDE de Arduino:



Desde mBlock podemos programar cada led de forma independiente. Por ejemplo, si el módulo está conectado al puerto 3 (color amarillo), y queremos que el primer led tenga el color rojo, el segundo el color verde, el tercero el color azul y el cuarto el color blanco, el programa que realizaría nuestros deseos sería el siguiente:



El programa anterior es muy simple y con mBlock podemos hacer mucho más. Por ejemplo, simular un arcoíris usando los leds de abordo y los del módulo 4 LEDs RGB. Este reto lo ejecuta el siguiente script:

En él se crean 3 variables: LedAleatorio (que escoge aleatoriamente un LED de los 4 posibles del módulo 4LEDs), n (que por defecto toma el valor 1 de los leds de a bordo de la placa y, gracias a n+1, selecciona el LED RGB 2 de la placa mCore) y la variable color (que tomará un valor al azar entre 0 y 255).

Divirtiéndome con mBot: Guía de manejo y programación

Todo este abanico de colores permanecerá fijo en la placa mCore (abordo) y en el módulo 4 LEDs RGB (conectado al puerto 4) durante 1 segundo. Pasado ese tiempo, se produce una nueva aleatoriedad de color y elección de led principal:



4.20. Módulo Display 7 segmentos

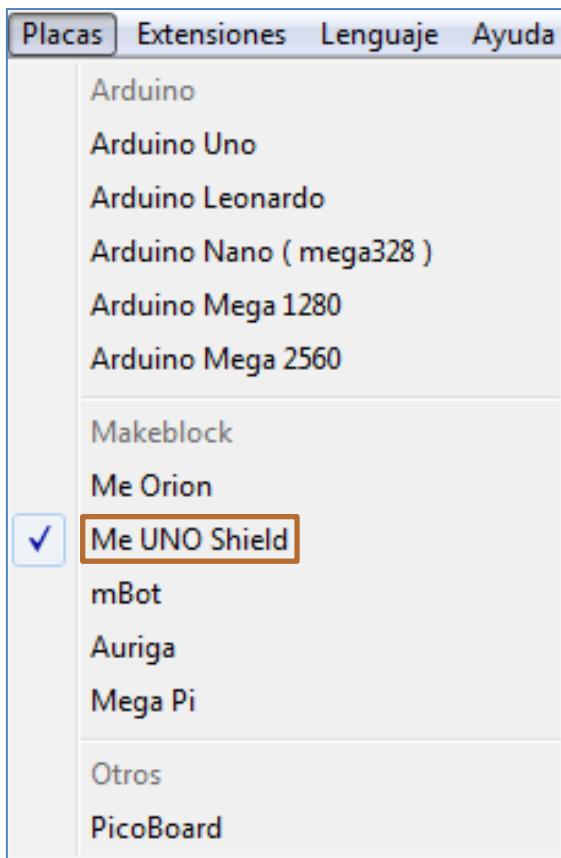
El módulo display 7 segmentos tiene su ID azul (puerto de señal digital doble). Color que nos indica que puede programarse con la placa mCore del robot mBot. Otras opciones son la placa Orion de Makeblock y la placa arduino uno, por ejemplo. A esta última, si se desea, se le puede acoplar el shield para arduino de Makeblock. Este display de 4 dígitos de ánodo común se basa en el chip TM1637 y con él podemos controlar cada segmento de su correspondiente dígito, así como, su punto decimal.



Módulo Display 7 segmentos

Divirtiéndome con mBot: Guía de manejo y programación

Si usamos una placa Arduino Uno y queremos programarla desde mBlock, debemos escoger la placa, en este caso *UNO Shield*, y el puerto de conexión correspondiente para nuestro Arduino Uno, tal y como se muestra en la siguiente imagen:



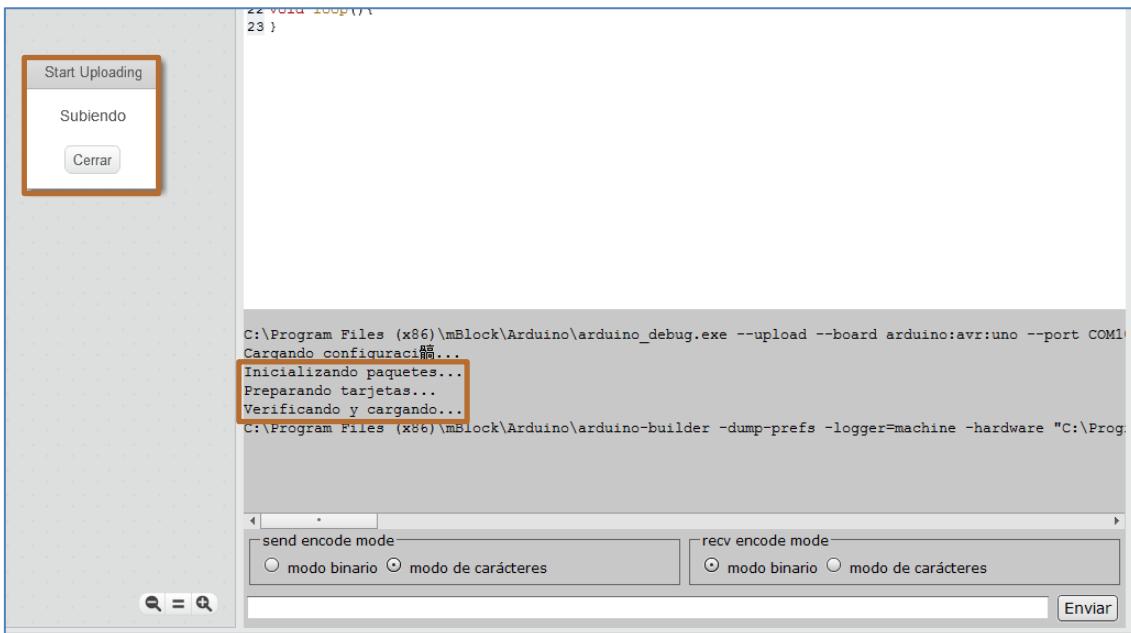
Supongamos que queremos visualizar un contador ascendente de 100 a 110, usando un display 7 segmentos conectado al puerto 7 del shield de arduino. El programa en mBlock sería el siguiente:



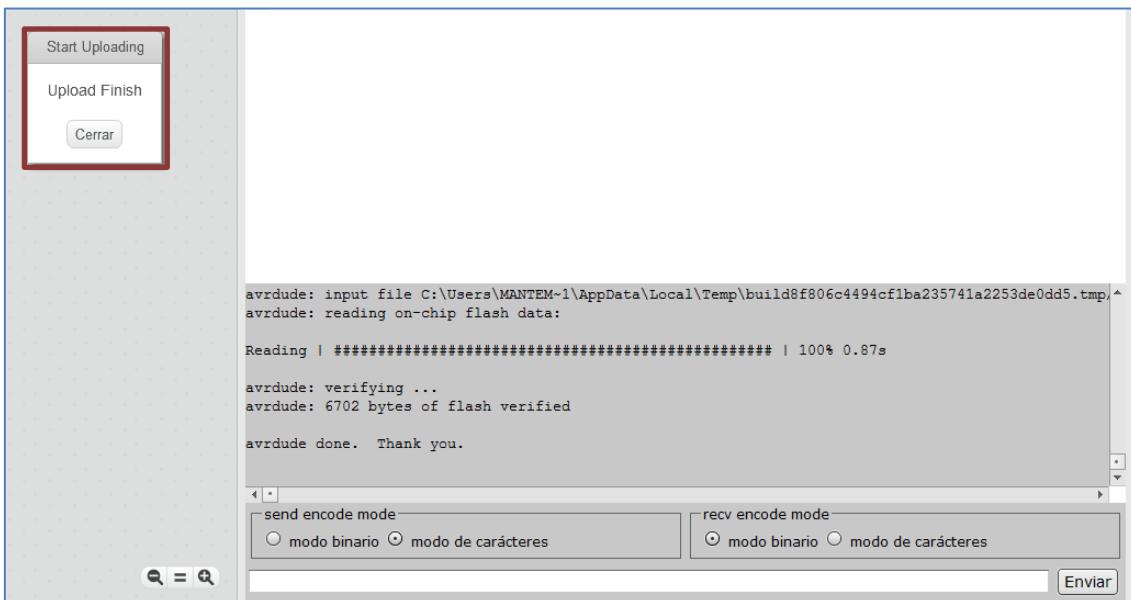
Para cargar el programa anterior en la placa arduino, debemos hacer clic en el comando azul *UNO Shield Program*. El programa nos informa que está subiendo el

Divirtiéndome con mBot: Guía de manejo y programación

programa: está cargando la nueva configuración, inicializando los paquetes necesarios, preparando la tarjeta y verificando y cargando:



Cuando finaliza la carga del programa a la placa y, si no ha habido fallos, el mensaje que nos muestra es el de carga finalizada:

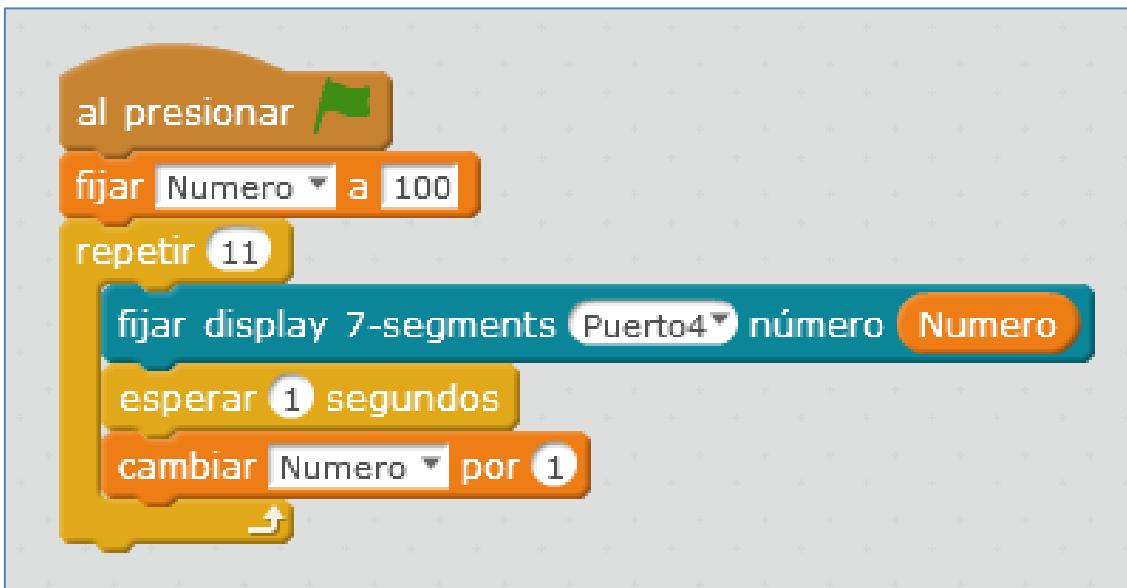


Si nos fijamos, el código arduino es el siguiente:

```
Back Upload to Arduino Editar con IDE de Arduino  
01 #include <Arduino.h>  
02 #include <Wire.h>  
03 //#include <Servo.h>  
04 #include <SoftwareSerial.h>  
05  
06 #include <MeShield.h>  
07  
08 double angle_rad = PI/180.0;  
09 double angle_deg = 180.0/PI;  
10 double Numero;  
11 Me7SegmentDisplay seg7_7(7);  
12  
13  
14 void setup(){  
15   Numero = 100;  
16   for(int i=0;i<11;i++)  
17   {  
18     seg7_7.display(Numero);  
19     delay(1000*1);  
20     Numero += 1;  
21   }  
22 }  
23  
24 }  
25  
26 void loop(){  
27  
28 }  
29 }  
30  
31 }
```

En nuestro display, veremos que van cambiando los números desde 100 a 110 en un intervalo de 1segundo cada uno.

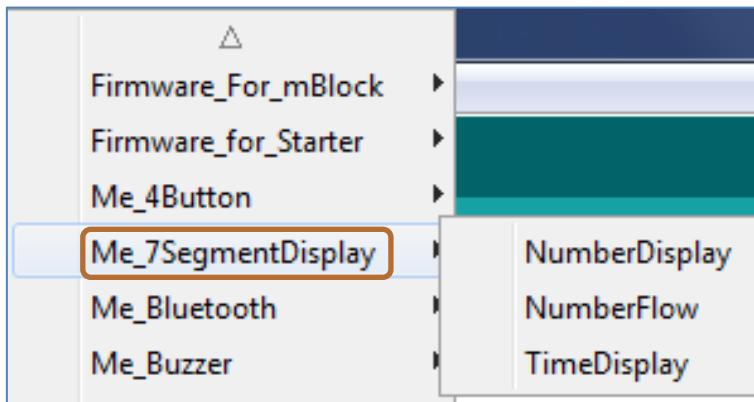
Si quisiéramos usar la placa mCore, sin cargar el programa a la misma, el script que debemos ejecutar con la bandera verde es el siguiente: (notar que el display está conectado al puerto 4 de la placa mCore)



Como en otros módulos, también podemos conectar directamente el módulo display7 segmentos a la placa arduino. Para ello, debemos conectar las 4 conexiones de su lateral a la placa Arduino Uno. Estas son: CLK (forma cuadrada de la señal de reloj),

DIO (línea de datos), 5V y GND; siendo las señales CLK y DIO pines digitales de la placa Arduino Uno.

Si preferimos utilizar código arduino, podemos partir de los ejemplos que se nos proporciona en las librerías de la casa:



4.21. Módulo Brújula o Me Compass

El sensor magnético Brújula o Me Compass es una brújula digital de 3 ejes. Su chip principal es Honeywell HMC5883L. Su campo de acción está entre -8 y 8 gauss con una resolución de 5mili-gauss. Puede autocalibrarse, y se programa a través de las librerías de arduino.

Del siguiente link podemos descargar los pasos que debemos seguir para calibrarlo: learn.makeblock.cc/wp-content/uploads/2015/06/Me_Compass_User_Manual.doc



Módulo Me Compass o módulo brújula

El color de su ID es blanco, y por lo tanto, puede conectarse a cualquier puerto de las diferentes placas que dispongan de este color.

En un lateral observamos 6 conexiones, las cuales utilizaremos si queremos conectar este módulo a una placa arduino uno sin pasar por el shield de makeblock. Estas 6 conexiones, mostradas como 2 y 4, son, respectivamente: RDY, KEY (ambas de conectarán a pines digitales de la placa Arduino Uno) y GND, 5V, SDA y SCL (estos dos últimas conexiones irán a 2 pines analógicos de salida de la placa Arduino Uno).

Este módulo es bastante sensible a los cambios del campo magnético en el ambiente. Es importante calibrar el módulo para obtener el valor angular correcto en la circunstancia actual.

4.22. Sensor de gas

El sensor de gas se compone de un sensor de humo del tipo MQ2, que se caracteriza por tener un corto tiempo de respuesta y por ser muy estable. A menudo se utiliza como un dispositivo de monitoreo de fugas de gas en el hogar o en las fábricas, logrando detectar el gas natural, butano, propano, metano, alcohol, hidrógeno, el humo, etc.

Su ID negro nos indica que tiene un puerto analógico y necesita ser conectado al puerto con ID negro en Makeblock Orion, Auriga, mCore o Shield de Arduino Uno.



Módulo Sensor de gas

Este sensor de gas dispone de una resistencia ajustable para adecuar la sensibilidad del sensor hacia el humo. La concentración de humo varía según la distancia entre el sensor y la fuente de humo, de modo que, a menor distancia, mayor será la concentración de humo y viceversa. Por lo tanto, es necesario establecer un valor de tensión que se corresponda con el umbral de una concentración de humo apropiada.

El dispositivo del tipo MQ-2 es un sensor de humo que utiliza óxido de estaño como material sensible de gas.

Como en otros muchos módulos, puede usarse este sensor para la detección de humos conectándolo directamente a una placa Arduino Uno. Para ello, debemos tener en cuenta sus tres conexiones dispuestas a un lateral del sensor y que son: GND, Vcc y DO; siendo DO un pin de salida digital de la placa Arduino Uno.

Vamos a ejemplificar diferentes opciones de programación para este sensor:

Podemos programarlo partiendo del ejemplo de sus librerías, haciendo el cambio de la placa correspondiente que utilizamos en su programación y su puerto de conexión. En mi caso, el shield de Arduino Uno y el puerto 8.

Divirtiéndome con mBot: Guía de manejo y programación



The screenshot shows the Arduino IDE interface with the title bar "MeGasSensorTest Arduino 1.6.8". The menu bar includes Archivo, Editar, Programa, Herramientas, and Ayuda. Below the menu is a toolbar with icons for upload, download, and other functions. The main code editor window contains the following C++ code:

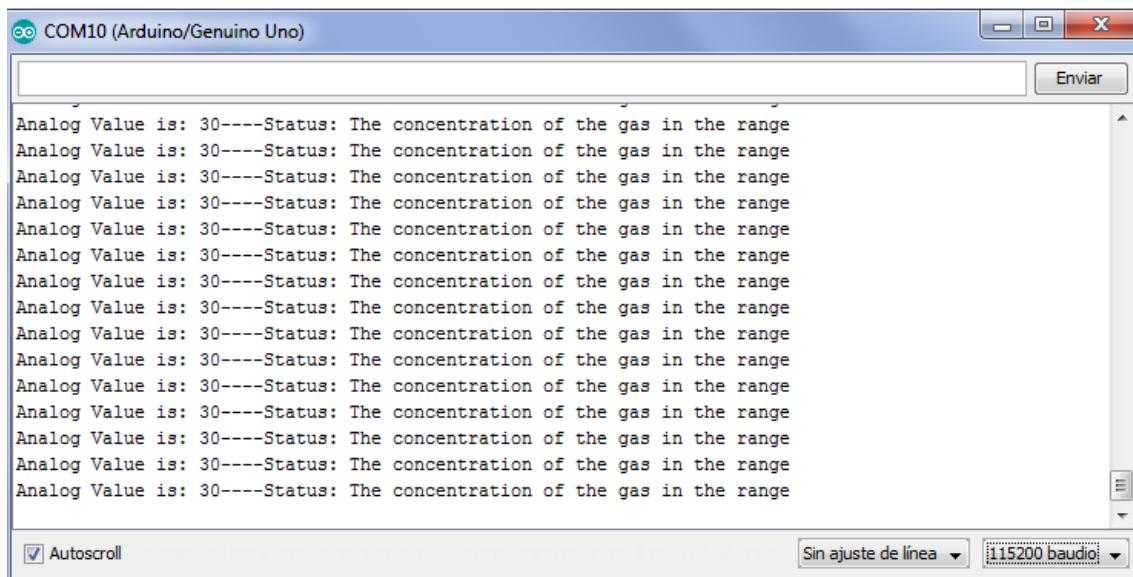
```
/*
 * \par Copyright (C), 2012-2016, MakeBlock
 * @file    MeGasSensorTest.ino
 * @author  MakeBlock
 * @version V1.0.0
 * @date    2015/09/09
 * @brief   Description: this file is sample code for Me gas sensor device.
 *
 * Function List:
 *   1. uint8_t MeGasSensor::readDigital(void)
 *   2. MeGasSensor::readAnalog(void)
 *
 * \par History:
 * <pre>
 * <Author>      <Time>        <Version>      <Descr>
 * Mark Yan      2015/09/09    1.0.0          rebuild the old lib
 * </pre>
 */
#include "MeShield.h"

MeGasSensor GasSensor1(PORT_8);

void setup()
{
  Serial.begin(115200);
}

void loop()
{
  Serial.print("Analog Value is: ");
  Serial.print(GasSensor1.readAnalog());
  Serial.print("----Status: ");
  if(GasSensor1.readDigital() == Gas_Exceeded)
  {
    Serial.println("The concentration exceeds");
  }
  else if(GasSensor1.readDigital() == Gas_not_Exceeded)
  {
    Serial.println("The concentration of the gas in the range");
  }
  delay(200);
}
```

Si aproximamos un mechero al sensor pero no abrimos el gas, el puerto serial a la frecuencia indicada por el programa 115200 baudios, nos informa, en la ventana de diálogo, que la concentración de gas está en el rango permitido.



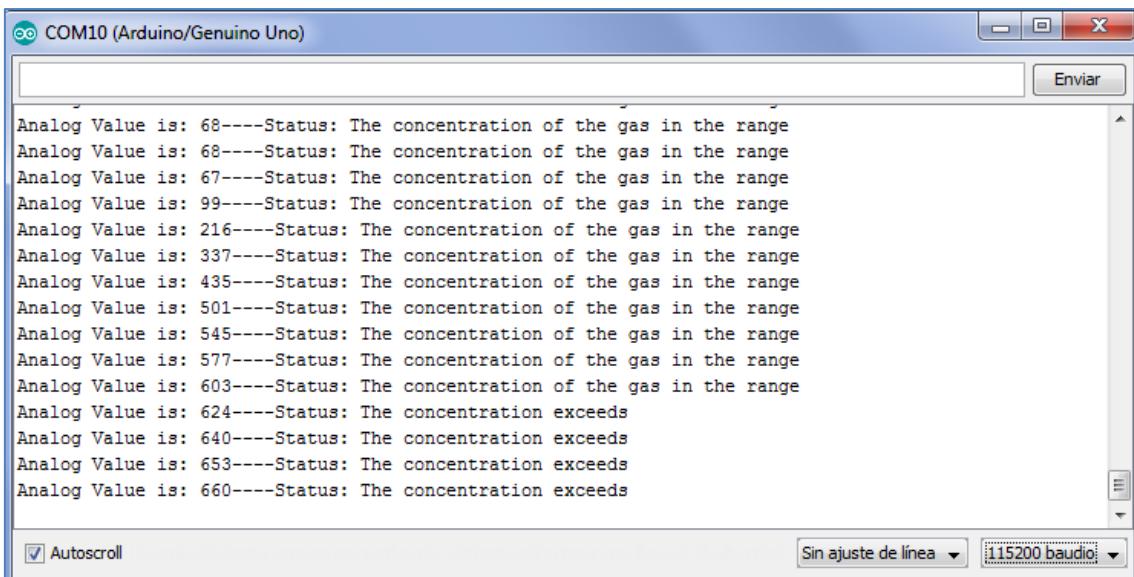
The screenshot shows the Arduino Serial Monitor window titled "COM10 (Arduino/Genuino Uno)". The window has a red close button and a blue minimize/maximize button. The text area displays repeated messages from the serial port:

```
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
Analog Value is: 30----Status: The concentration of the gas in the range
```

At the bottom of the window, there are several status indicators: "Autoscroll" (checked), "Sin ajuste de línea" (dropdown menu), and "115200 baudio" (dropdown menu).

Divirtiéndome con mBot: Guía de manejo y programación

Pero, si dejamos escapar el gas del mechero, el valor analógico cambia, aumentando considerablemente. En el puerto podemos leer el mensaje de alerta informándonos que la concentración de gas es excesiva.

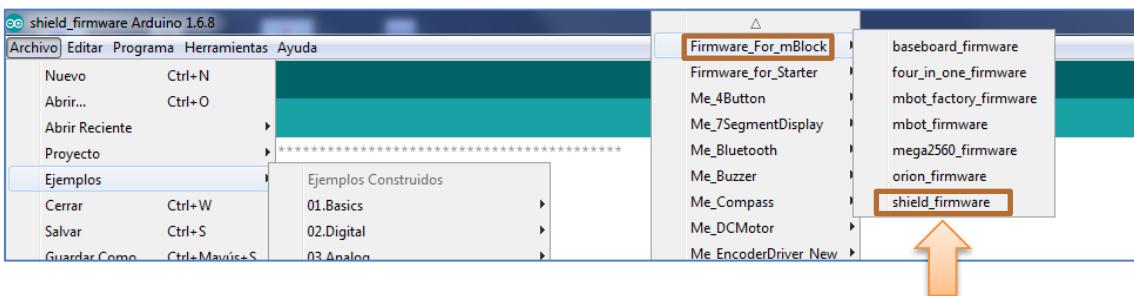


The screenshot shows the Arduino Serial Monitor window titled "COM10 (Arduino/Genuino Uno)". The text area displays a series of messages from the Arduino code, indicating analog values and gas status. The messages are as follows:

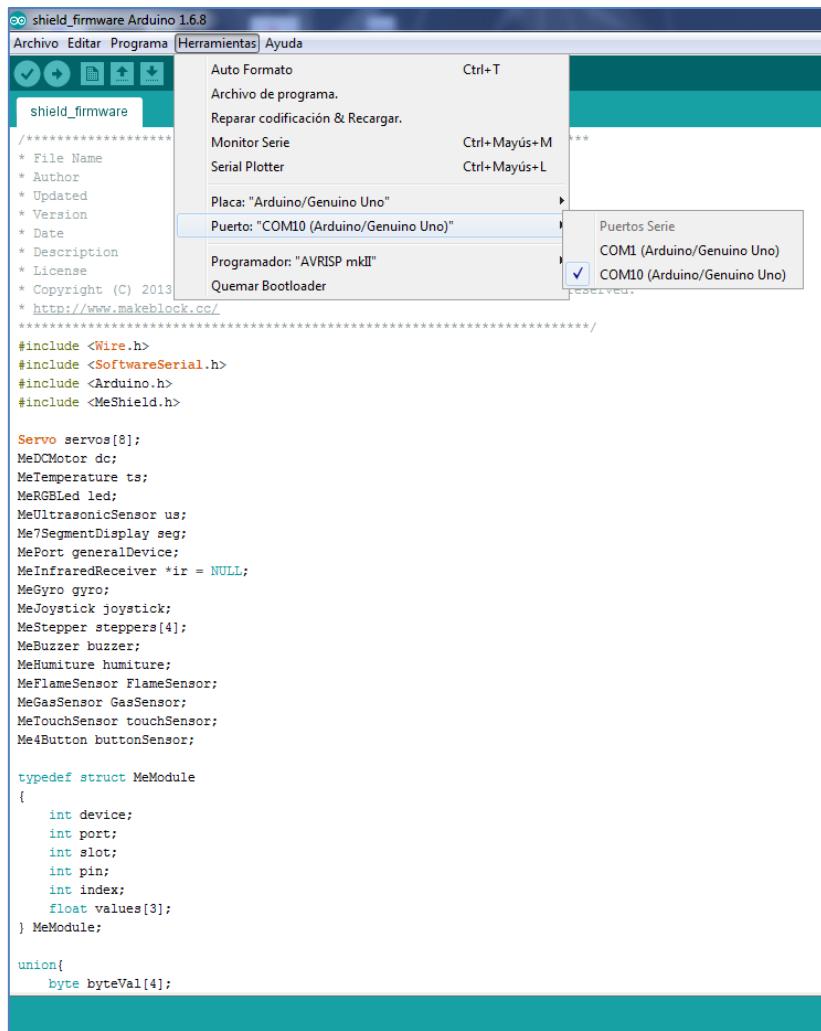
```
Analog Value is: 68----Status: The concentration of the gas in the range  
Analog Value is: 68----Status: The concentration of the gas in the range  
Analog Value is: 67----Status: The concentration of the gas in the range  
Analog Value is: 99----Status: The concentration of the gas in the range  
Analog Value is: 216----Status: The concentration of the gas in the range  
Analog Value is: 337----Status: The concentration of the gas in the range  
Analog Value is: 435----Status: The concentration of the gas in the range  
Analog Value is: 501----Status: The concentration of the gas in the range  
Analog Value is: 545----Status: The concentration of the gas in the range  
Analog Value is: 577----Status: The concentration of the gas in the range  
Analog Value is: 603----Status: The concentration of the gas in the range  
Analog Value is: 624----Status: The concentration exceeds  
Analog Value is: 640----Status: The concentration exceeds  
Analog Value is: 653----Status: The concentration exceeds  
Analog Value is: 660----Status: The concentration exceeds
```

At the bottom of the window, there are checkboxes for "Autoscroll" and "Sin ajuste de línea" (No line wrap), and a dropdown for "115200 baudios".

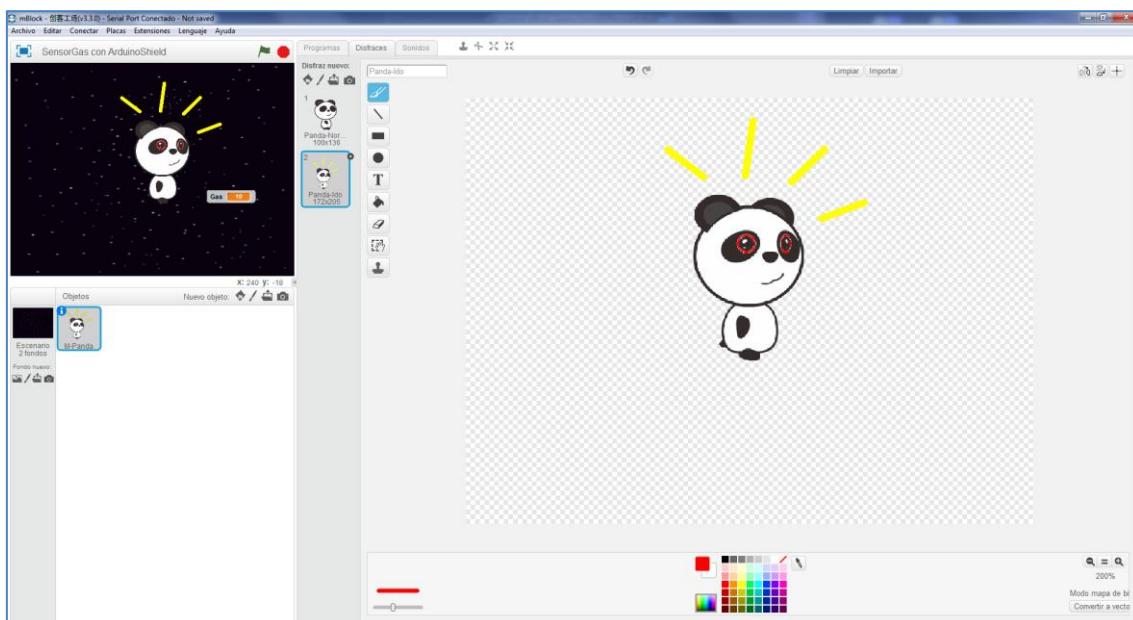
También podemos programar la placa Arduino Uno dentro de mBlock. Para ello debemos cargarle el firmware determinado, llamado “shield_firmware” a la placa Arduino Uno desde el IDE de Arduino:



Divirtiéndome con mBot: Guía de manejo y programación



Podemos crear una simulación sencilla. Por ejemplo, la detección de la fuga de gas de un mechero. Relativo al objeto de mi simulación, voy a usar el personaje “M-Panda” del mBlock y “tunear” su segundo disfraz:



Divirtiéndome con mBot: Guía de manejo y programación

El pequeño script que vamos a ejecutar es el siguiente:



Al abrir el gas del mechero y acercárselo al sensor, si la concentración de gas que detecta es menor que 60, el objeto “M-Panda” dirá “Nivel normal” por 2 segundos. En caso contrario, cambia el disfraz y nos informa que está “gaseado”. Es decir, el nivel de gas es elevado:



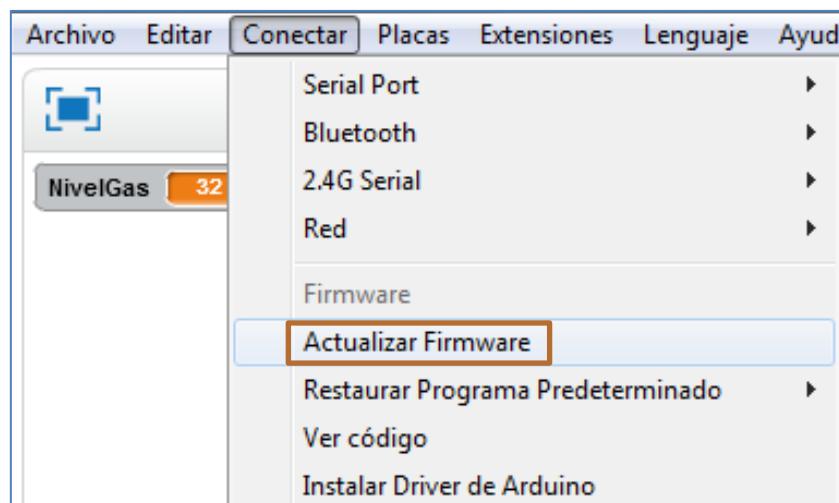
Concentración normal de gas



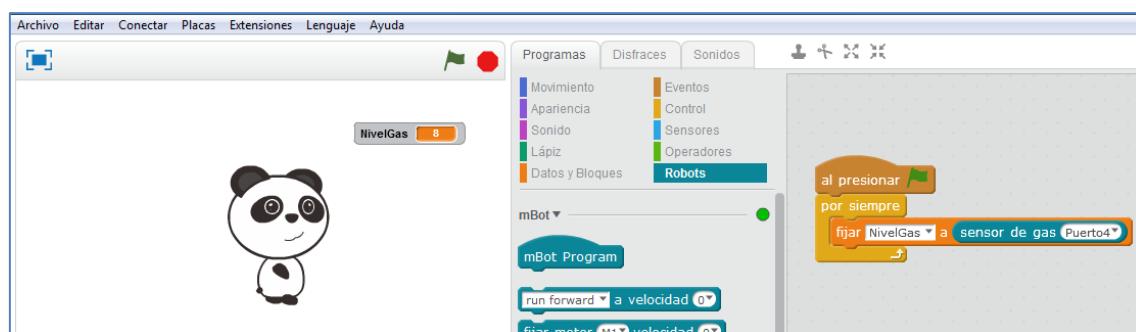
Concentración elevada de gas

Finalmente, podemos programarlo con scratch desde el programa mBlock cargando el firmware desde el propio programa.

Primero, tras conectar la placa del mBot al puerto serie, debemos actualizar su firmware: *Conecrtar > Actualizar Firmware*



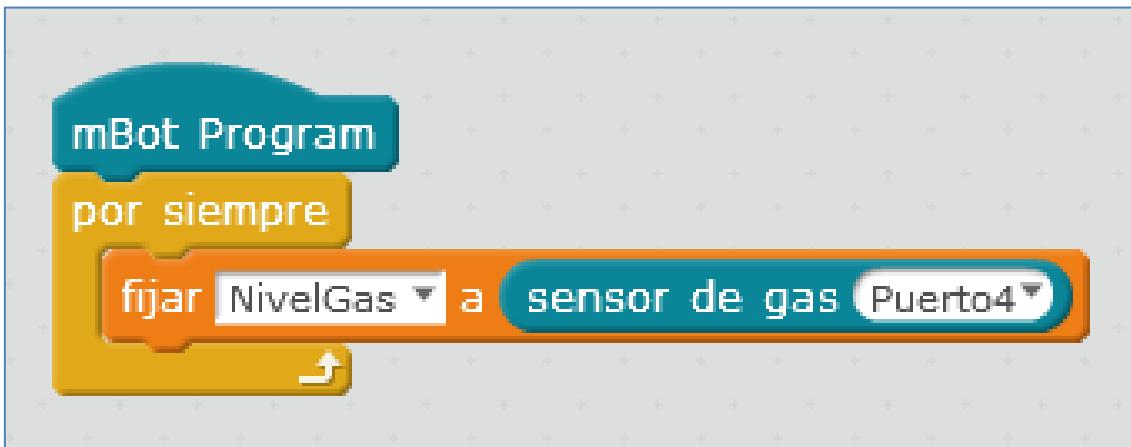
Ahora, simplemente, nos queda programar. Podemos, por ejemplo, crear el siguiente script: (notar que usamos el mBot y conectamos el sensor de gas a su puerto 4)



Divirtiéndome con mBot: Guía de manejo y programación

En los niveles normales de gas, tras una primera comprobación, el sensor, en su medida, nos ofrece el valor numérico 8 (ver imagen anterior). Pero, si dejamos correr el gas del mechero, este valor numérico se dispara.

También podríamos haber hecho clic en comando mBot Program:



4.23. Sensor de Luz

Desarrollado sobre la base del principio de efecto fotoeléctrico en semiconductores, este sensor de luz puede utilizarse para detectar la intensidad de la luz ambiente. Es decir, detecta fácilmente si tenemos o no la luz encendida o apagada. Su ID negro significa que tiene un puerto de señal analógica y necesita ser conectado al puerto con ID negro en Makeblock Orion, Auriga, mCore o Shield de Arduino Uno.



Módulo Sensor de Luz

En el caso de la mCore, podemos usar el puerto 3 o 4. Si conectamos el sensor de luminosidad al puerto 3 y creamos un programa simple que nos muestre la luminosidad numérica de la habitación, con o sin luz encendida, nos encontraremos con el siguiente resultado:



Luminosidad con la luz apagada



Luminosidad con la luz encendida

El rango de valores del sensor varía entre 0 y 980, de modo que, a mayor valor, mayor luminosidad y a menor valor, mayor oscuridad.

El script que activa el sensor es el siguiente:

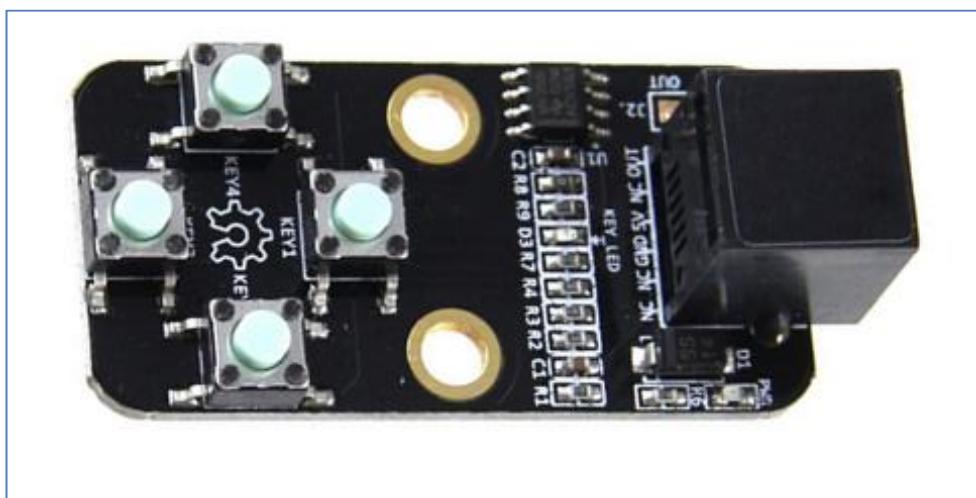


Como otros sensores, este componente también se podría conectar directamente a la placa Arduino Uno teniendo en cuenta sus 3 conexiones del lateral: GND, Vcc y AO (soldadura cuadrada). Siendo la señal AO un pin analógico de salida de la placa Arduino Uno. Finalmente, sólo decir que en la librería de Makeblock disponemos de tres programas que nos ejemplifican su programación desde el IDE de Arduino:

Me_LightSensor	MeLightSensorTest
Me_LimitSwitch	MeLightSensorTestResetPort
Me_LineFollower	MeLightSensorTestWithLEDOn

4.24. Módulo 4 botones

Como su nombre indica, dispone de 4 pulsadores que, tienen especial funcionalidad o relevancia en la programación de juegos o en el movimiento de un personaje en el mBlock. Como se observa en la siguiente imagen, cada pulsador está numéricamente serigrafiado, en el sentido de las agujas del reloj, por las letras “key 1, key 2, key 3 y key4”:



Módulo de 4 pulsadores

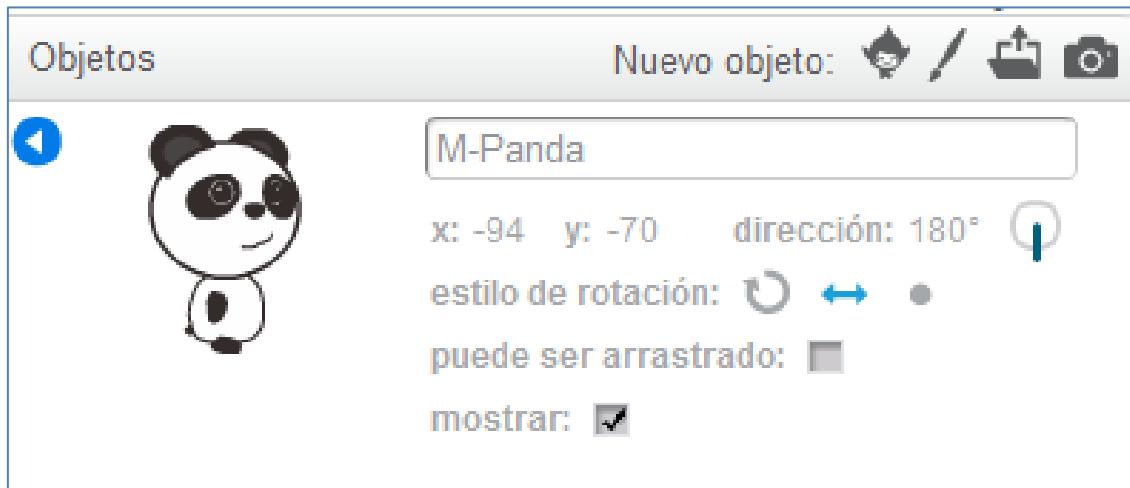
Su ID de color negro significa que tiene un puerto de señal analógica y necesita ser conectado al puerto con ID negro en Makeblock Orion, Auriga, mCore o Shield de

Divirtiéndome con mBot: Guía de manejo y programación

Arduino Uno. En el caso de la placa mCore, este módulo puede ir conectado a los puertos 3 y 4.

Supongamos que queremos mover el objeto “M-Panda” por el escenario del mBlock utilizando este módulo, de modo que:

Si pulsamos el pulsador “key1”, el objeto “M-Panda”, que presenta estilo de rotación derecha-izquierda, se moverá 10 pasos hacia la derecha. Al pulsar “key3” se moverá 10 pasos hacia la izquierda. Y, al pulsar “key4” o “key2” se desplazará, respectivamente, 10 pasos verticalmente hacia arriba o hacia abajo.



Estilo de rotación del objeto “M-Panda”

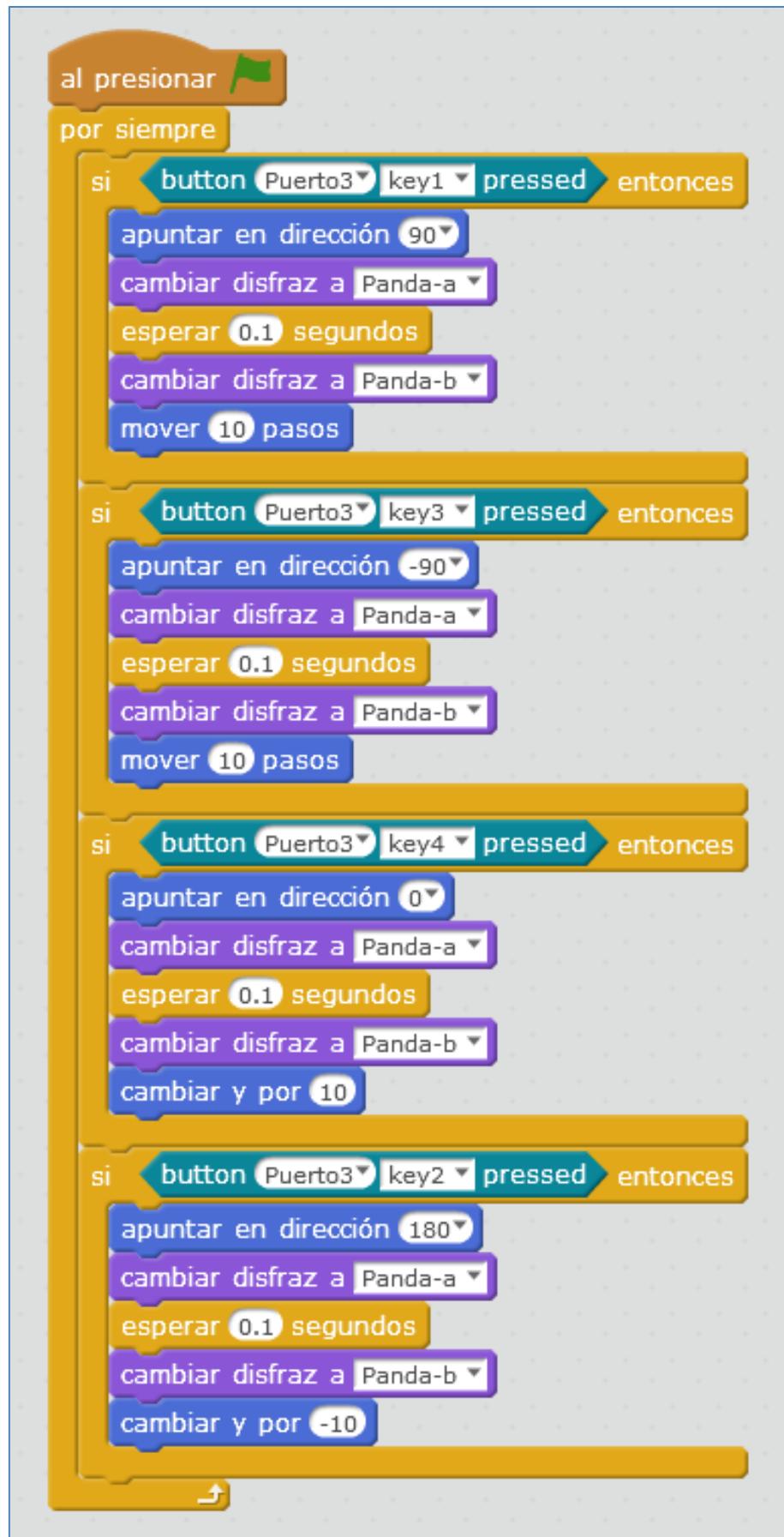
mBlock presenta el comando:

“button_PuertoCorrespondiente_PulsadorCorrespondiente_pressed”, que nos recuerda al comando azul claro del bloque sensores del scratch relativo a la presión de una tecla del teclado. Ambos, podemos verlos en la imagen inferior.

Como nuestro deseo es usar este módulo 4 botones, debemos ceñirnos al propio comando del bloque de robots del mBlock:



El script, para nuestro sensor conectado al puerto 3 de la placa mCore del mBot, y que implementa el programa de movimiento requerido sobre el objeto “M-Panda”, sería el siguiente:



4.25. Pantalla TFT LCD 2.2

La pantalla TFT LCD tiene un tamaño de 2.2 pulgadas y una resolución de 220x176. Almacena hasta 2M, pudiéndole enviar comandos vía serie a diferentes escalas de baudios (entre 2400 y 256000), aunque por defecto funciona a 9600 baudios. En ella podemos mostrar texto y figuras geométricas, permitiendo la regulación de luz de fondo entre 0 y 100.

Su color de identificador es Gris-Azul y eso significa que sólo la puedo conectar a los puertos de las placas que dispongan de alguno de estos colores, como es el caso del puerto 5 de la placa Orion de Makeblock o el puerto 5 del shield de arduino.



Parte trasera de la pantalla TFT LCD



Parte frontal de la pantalla TFT LCD

Divirtiéndome con mBot: Guía de manejo y programación

Podemos utilizar directamente una placa Arduino Uno. En este caso, las conexiones que debemos realizar se especifican en la siguiente imagen y tabla (entre TX y RX sus contactos son cruzados):



Pantalla TFT LCD	Arduino Uno
TX	RX (pin digital 0)
RX	TX (pin digital 1)
Vcc	5V
GND	GND

NOTA: La casa Makeblok dispone del *cable Dupont* que facilita la conexión de un RJ25 a pines hembra para la placa Arduino Uno.

Si utilizamos el archivo de ejemplo (*Me TFT*) de las librerías de la casa, podemos entender qué comandos utiliza para los diferentes colores, cómo lograr dibujar cuadrados, círculos, escribir texto, etc:

Divirtiéndome con mBot: Guía de manejo y programación

The screenshot shows the MeTFT Arduino 1.6.8 IDE interface. The code editor contains the `MeTFT.ino` sketch. The code includes comments about the copyright (2012-2016, MakeBlock), file information, and a brief description of the sample program for TFT LCD. It defines a `MeSerial` object and includes headers for `MeShield.h` and `SoftwareSerial.h`. The code then sets up 64 colors and defines screen coordinates. A detailed command description table follows, listing commands like `CLC(c)`, `SCC(c,n)`, `SBC(c)`, `PS(x,y,c)`, `PL(x1,y1,x2,y2,c)`, `BOX(x1,y1,x2,y2,c)`, `BOXF(x1,y1,x2,y2,c)`, `PIC(x,y,n)`, `DS12(x,y,'String',c)`, and `DS16(x,y,'String',c)` along with their descriptions and sample code. Below the code editor, a status bar shows "Subido" and "Opciones de compilación cambiadas, reconstruyendo todo". A message at the bottom indicates the sketch uses 12.612 bytes of program memory and 1.226 bytes of dynamic memory.

```
/*
 * \par Copyright (C), 2012-2016, MakeBlock
 * @file    MeTFT.ino
 * @author  MakeBlock
 * @version V1.0.0
 * @date    2015/11/10
 * @brief   Description: this file is sample program for TFT LCD.
 *
 * Function List:
 *
 * \par History:
 * <pre>
 * <Author>      <Time>      <Version>      <Descr>
 * Mark Yan      2015/11/10    1.0.0
 * </pre>
 */
#include "MeShield.h"
#include <SoftwareSerial.h>

MeSerial mySerial(PORT_5);

//----- 64 colors of c -----
// These 64 colors are 'c' value of command
// with this sample code, you can see the color in your LCD.

//----- Screen coordinates -----
// The upper left corner is coordinate point(0,0)
// Screen resolution is 320 * 240
// transverse direction is the X direction,
// the vertical direction is the Y direction;

//----- Command Description -----
// Command          Description           sample code
// CLC(c);         clear screen with c color    CLS(0);
// SCC(c,n)        customizing colors for c    SCC(15,65535);
// SBC(c)          Set Background Color       SBC(1)
// PS(x,y,c)       draw a point at (x, y) with color c  PS(100,100,1);
// PL(x1,y1,x2,y2,c);  draw line from (x1,y1) to (x2,y2)  PL(0,0,100,0,2);
// BOX(x1,y1,x2,y2,c);  draw rectangle,Left corner(x1,y1)  BOX(0,0,100,100,2);
// BOXF(x1,y1,x2,y2,c);  draw Solid rectangle,Left corner(x1,y1)  BOXF(0,0,100,100,2);
// PIC(x,y,n)      painted figure n at(x,y)      PIC(0,0,1);
// DS12(x,y,'String',c);  display character 12 dot matrix characters
// DS16(x,y,'String',c);  display character 16 dot matrix characters

Subido
Opciones de compilación cambiadas, reconstruyendo todo

El Sketch usa 12.612 bytes (39%) del espacio de almacenamiento de programa. El máximo es 32.256 bytes.
Las variables Globales usan 1.226 bytes (59%) de la memoria dinámica, dejando 822 bytes para las variables locales.
```

Algunos comandos o instrucciones de operación general son los siguientes:

- ❖ `CLS(C);` // Limpia la pantalla con el color “C”.
 - Por ejemplo: `Serial.print("CLS(0);")`; limpiaría la pantalla con el color negro.
- ❖ `PS(X,Y,C);` // Dibuja un punto con el color “C” en la posición (X,Y)
- ❖ `PL(X1,Y1,X2,Y2,C);` // Dibuja una línea recta con el color “C” desde el punto (X1,Y1) al punto (X2,Y2)

- ❖ BOX(X1,Y1,X2,Y2,C); // Dibuja un cuadrado o rectángulo con el color “C”, siendo su esquina superior derecha el punto (X1,Y1) hasta su esquina inferior en el punto (X2,Y2).
- ❖ BOXF(X1,Y1,X2,Y2,C); // Dibuja un cuadrado o rectángulo relleno con el color “C”, siendo su esquina superior derecha el punto (X1,Y1) hasta su esquina inferior en el punto (X2,Y2).
- ❖ CIR(X,Y,R,C); // Dibuja un círculo de radio “R” con color “C” siendo su centro el punto (X,Y).
- ❖ DS12(X,Y,’Content’,C); // Muestra una línea de **12** caracteres (contenido) de matriz de puntos con color “C” en la posición (X, Y). También permite mostrar **16, 24 y 32** caracteres.
 - Por ejemplo: `Serial.print("DS32(150,150,'hola mundo',4);");`

Otros comandos avanzados son los siguientes:

- ❖ SBC(C); // Configurar el color de fondo C y mostrar el color de relleno sin la matriz de puntos mientras se observan otros caracteres.
- ❖ SCC(C,N); // Personalizar el color “C” utilizando la variable “N”.
- ❖ BS12(X1,Y1,X2,LW,’Content’,C); // Visualizar una cadena de **12 (16, 24 o 32)** caracteres desde la posición (X1,Y1), automáticamente hasta la posición X2, con el espacio de línea LW y color C.
- ❖ DRn; // Configurar el sentido de visualización de la pantalla, siendo “n” el número 0, 1, 2 o 3.
 - Por ejemplo, de forma vertical sería `Serial.print("DR1");` y de forma invertida sería DR2.
- ❖ SEBL(n); // Configurar el brillo de la pantalla, de 0 a 100, siendo 100 su máximo valor.

Relativo al color “C”, los códigos son los siguientes:

0	Black
1	Red
2	Green
3	Blue
4	Yellow
5	Cyan
6	Pink
7	White

4.26. Módulo Micro Switch

Este módulo no es más que un interruptor físico que puede ser usado, entre otras opciones, como un final de carrera para, por ejemplo, desactivar la actuación de motores.



Micro Switch B

Su conexión, en las placas Makeblock, requieren un Adaptador RJ25 y su utilidad dependerá de la creatividad del programador.

En el siguiente ejemplo actúa como simple contador en el escenario del mBlock, a través de la variable “Número”. La placa a la que se ha conectado el Micro Switch (a través del adaptador) es la mCore del mBot, usando el *Banco1* del adaptador y estando éste conectado al *Puerto4* de la placa.

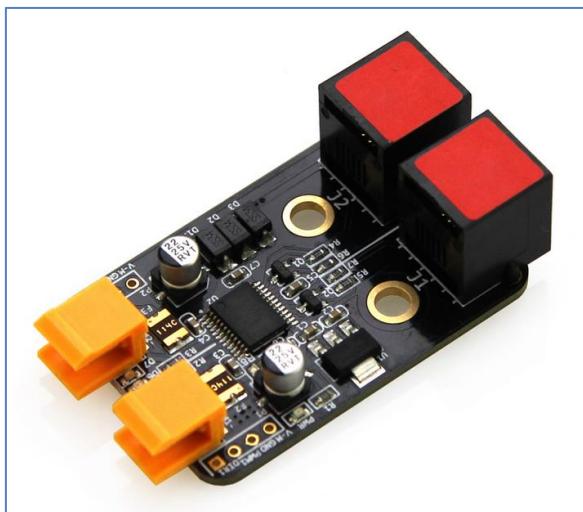
Al hacer contacto el final de carrera, este envía una señal que hará que el contador aumente en una unidad y suene la nota C4:



4.27. Módulos controladores de motores

La controladora de motores DC dual de las series Me está diseñada para que sea fácil de programar y de cablear. Además, podemos usar su librería de Arduino para un manejo sencillo.

Cada controladora puede controlar hasta 2 motores DC a través de su respectivo conector RJ25 que contiene las líneas de alimentación (6V a 12V) y de señal. También viene con pines de 2.54mm para que se pueda utilizar con jumpers. Presenta la función de regulación de velocidad PWM y soporta una corriente máxima de 1A por cada motor. Su ID es rojo y eso significa que debe estar conectada un puerto de color rojo de la placa Orion de Makeblock (entre otras).



Controladora de motores DC Dual

En sus laterales podemos observar 4 agujeros (para 4 pines) para cada motor. En la imagen se ve los correspondientes para el motor 2:



V-M	Potencia del motor (6V-12V)
GND	Tierra
PWM	Modulación por ancho de pulso (regulación de velocidad)
DIR	Control de dirección

Conectarlo a una placa Orion es muy sencillo: necesitamos tantos RJ25 como motores (como máximo 2) queramos conectar.

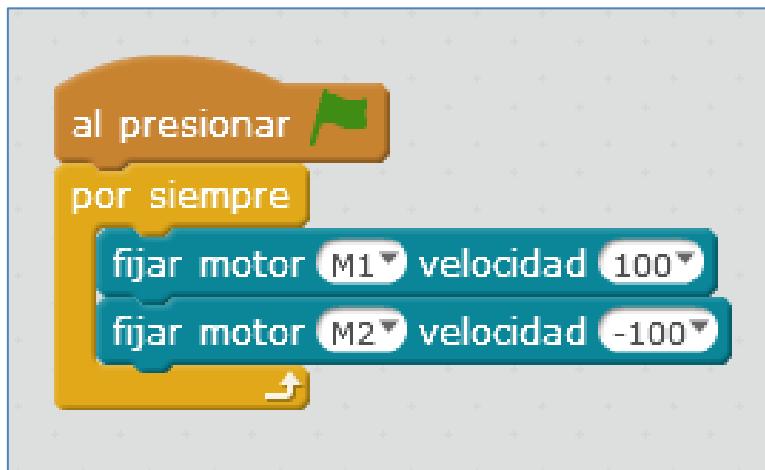
Si queremos usarlo con una placa Arduino Uno, las conexiones podemos hacerlas a través de un conector RJ25 a Dupont (en cada motor) o soldando cables en los

Divirtiéndome con mBot: Guía de manejo y programación

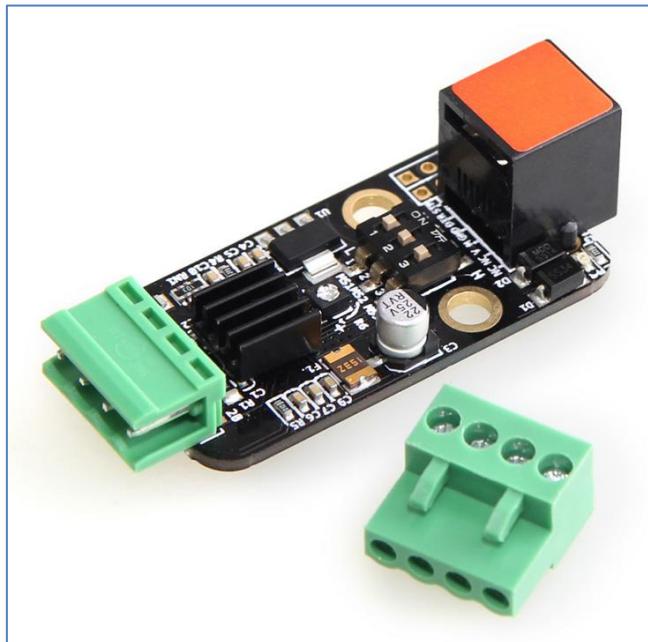
agujeros de la tabla anterior. La conexión final del controlador con una Arduino Uno se muestra en la siguiente tabla:

V-M	Vin
GND	Tierra GND
PWM	Pin PWM como el Pin9
DIR	Pin 8

Para programar los dos motores con mBlock, con sentidos de giro contrarios (por ejemplo), deberíamos activar el siguiente script:



En el caso de un motor paso a paso, disponemos de un módulo controlador específico para él:



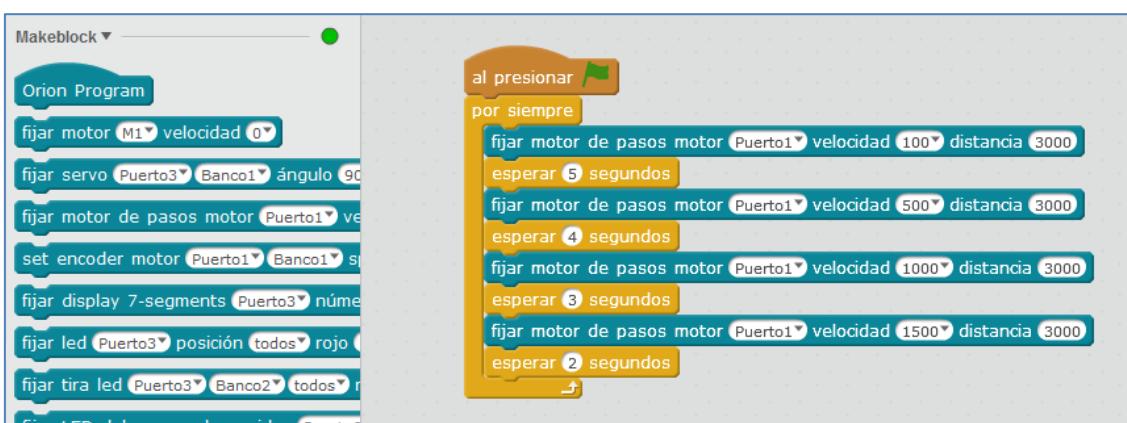
Controladora de motor paso a paso

Los motores paso a paso son ideales para mecanismos que requieren movimientos muy precisos, transformando una señal de pulso en un desplazamiento lineal o

Divirtiéndome con mBot: Guía de manejo y programación

angular. Para cada señal de pulsos lo hacemos girar un cierto ángulo, pudiéndolos mover un paso a la vez por cada pulso que se aplique. Este paso puede variar desde 90° hasta pequeños movimientos de tan solo 1.8°: es decir, que se necesitarán 4 pasos en el primer caso (90°) y 200 para el segundo caso (1.8°), para completar un giro completo de 360°. Si se requiere una mayor precisión, podemos elegir otro modo. Por ejemplo, elegimos el modo paso a paso 1/4 (es decir, 0,45° por paso), y en este caso, el motor debe girar 800 micro pasos para completar la vuelta.

Este controlador, al tener ID rojo, debe utilizarse con la placa Orion o Auriga (o con una Arduino Uno). En la siguiente imagen se controla un motor paso a paso desde scratch, haciendo que su velocidad de rotación vaya aumentando, de menor a mayor velocidad de forma continua.



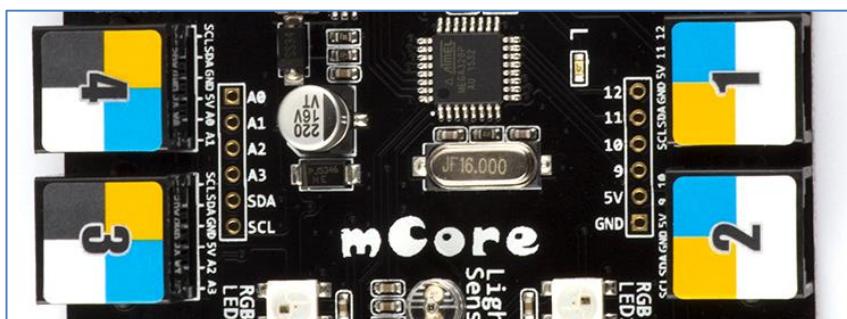
5. Arduino – Makeblock y viceversa

Al inicio del punto cuarto de este documento presentaba la placa mCore como una placa Arduino Uno. En algunos ejemplos del punto anterior se han programado sensores basándonos en las librerías de Makeblock cargadas a partir del IDE de Arduino, seleccionando la placa como una Arduino Uno.

También hemos hecho ejemplos⁴ partiendo de una placa Arduino Uno, a la que le añadíamos el shield de Makeblock para facilitar las conexiones de los sensores y componentes de la casa. En cuanto a su programación, esta podía hacerse con scratch, a través de mBlock, o con el IDE de arduino.

5.1. Makeblock – Arduino Uno

Si observamos la placa mCore, vemos serigrafiados los pines analógicos y digitales que nos recuerdan una placa Arduino Uno y que, mCore, implementa en cada puerto. En la siguiente imagen podemos visualizar los pines A0, A1, A2 y A3, así como, los D9, D10, D11 y D12, 5V y GND:



Arduino Uno en mCore

Por lo tanto, podemos afirmar que una placa mCore es una placa Arduino Uno que usa diferentes conectores. Me explico, la placa mCore dispone de conectores RJ25 y una Arduino Uno carece de ellos. La casa Makeblock sacó al mercado el conector RJ25 a Dupont que es un conector que en un extremo presenta un enganche RJ25 y, al otro extremo, 6 conectores hembra para unirlos a diferentes sensores y actuadores que no sean los de la propia casa.



Conecotor RJ25 a Dupont

Si no queremos usar estos conectores Dupont, sólo nos queda la opción de soldar cables en el tipo de agujeros que vemos en la siguiente imagen de los 4 puertos de la placa:

⁴ Ejemplos: 4.10, 4.17, 4.22 y 4.24



Entre las dos opciones, cables Dupont y soldar, particularmente, prefiero la primera.

Como la placa mCore es una Arduino Uno, podríamos cargar cualquier programa desde el IDE de Arduino a la placa mCore, seleccionando la opción Arduino Uno con el puerto COM correspondiente.

Por ejemplo, podríamos cargar el siguiente programa para el pin 12 de nuestra placa, haciendo que muestre su estado en el puerto *serial*:

```
int pin = 12;

void setup()
{
    Serial.begin(9600);
    pinMode(pin, INPUT);
}

void loop()
{
    // Leer el valor del pin
    int buttonState = digitalRead(pin);

    // mostrarlo por el monitor de puerto serie
    Serial.println(buttonState);
    delay(1000);
}
```

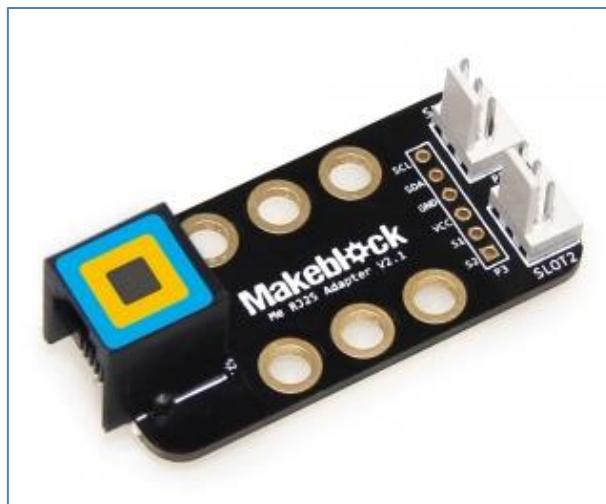
Obviamente, la placa mCore, al incluir componentes electrónicos propios ya soldados en ella, utiliza los restantes pines de la placa Arduino Uno que no vemos serigrafiados en esos elementos. Pines que tiene implementados en RGBs, Buzzer, etc. Si queremos usarlos en el IDE de Arduino, podemos echar mano de las librerías de Makeblock e incluir al inicio de nuestros programas, en la cabecera, el comando `#include <MeMCore.h>`. Con esta explicación queda claro que, una placa mCore podemos usarla como lo que es, como una placa Arduino Uno.

5.2. Arduino Uno - Makeblock

En Tecnología, en nuestros talleres, muchos disponemos de placas Arduino Uno. Pues bien, la pregunta del millón es si podemos usarlas con mBlock y programar una gran cantidad de sensores y actuadores con scratch. La respuesta es sí; podemos hacerlo y me explico:

Divirtiéndome con mBot: Guía de manejo y programación

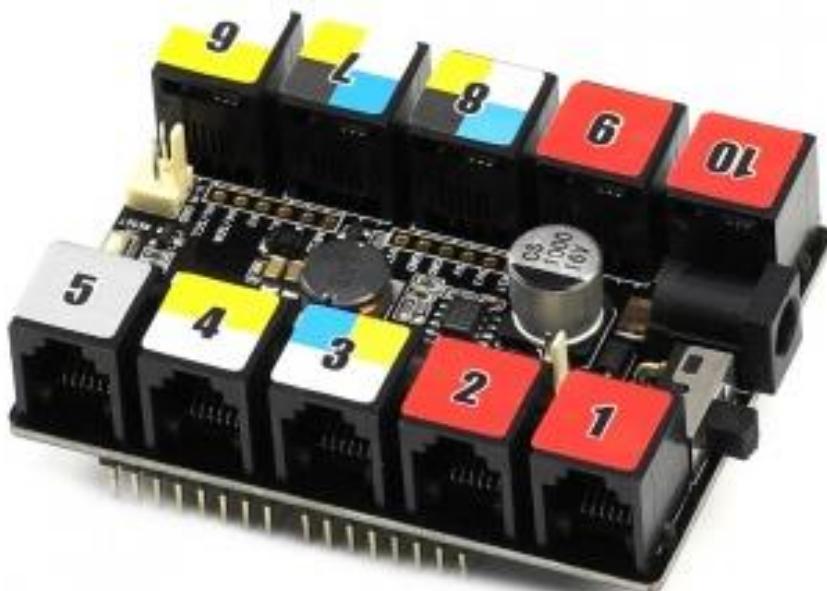
Los sensores y actuadores de la casa Makeblock se conectan mediante conectores RJ25. Conectores que no dispone la placa Arduino Uno. Una forma de usarlos es mediante un shield que disponga de estos adaptadores RJ25. Como se vio en el punto cuarto de este documento, algunos componentes de la casa requieren del módulo adaptador RJ25 y, por ese motivo, cuando proceda, combinaremos ese adaptador con el shield para Arduino Uno.



Adaptador RJ25

La casa Makeblock ha sacado al mercado un Shield que convierte la placa Arduino Uno en una placa controladora Makeblock de 10 conectores RJ25.

Como la placa sigue siendo Arduino Uno, la programación sencilla desde el IDE de Arduino requiere de la instalación de las librerías Makeblock para Arduino.

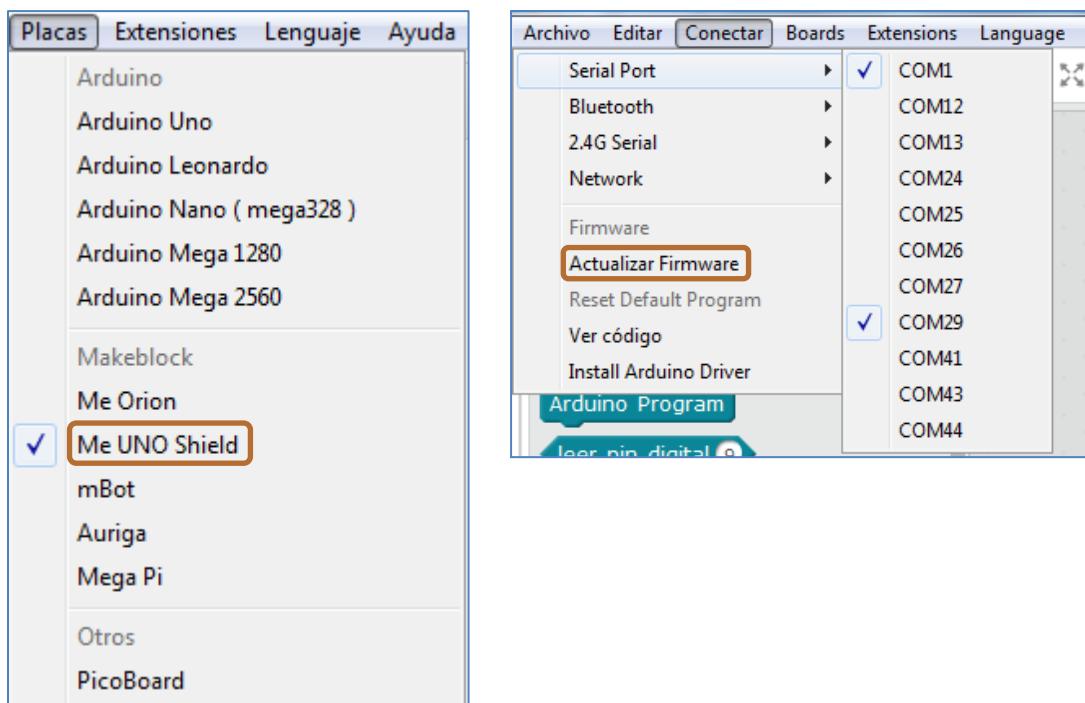


Shield Arduino Uno

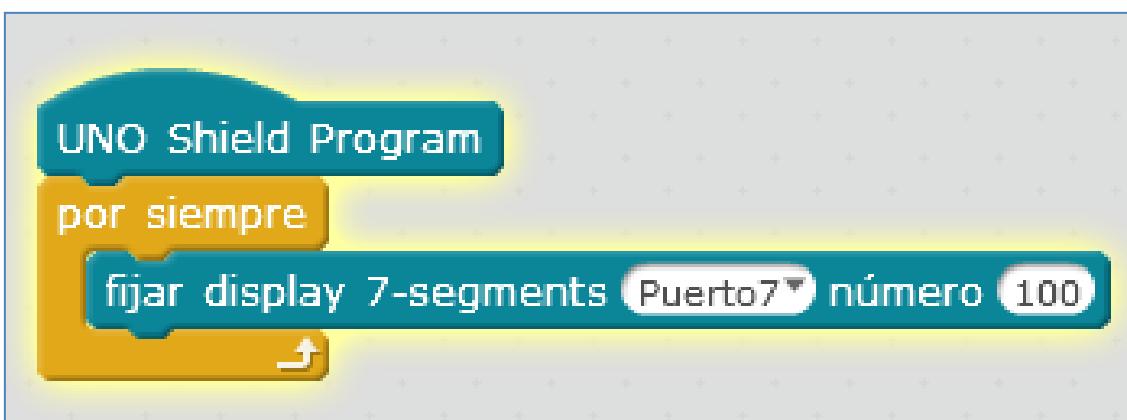
Divirtiéndome con mBot: Guía de manejo y programación

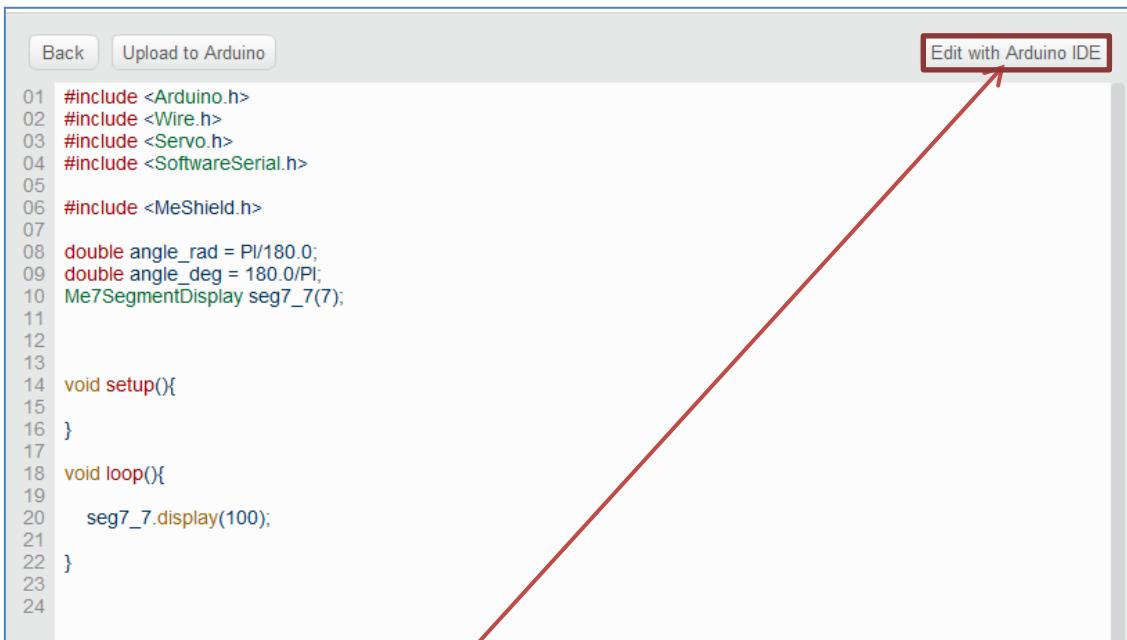
Podemos usar la placa *Arduino Uno* junto con el *shield de Arduino Uno* desde scratch, es decir, desde mBlock. Para ello debemos cargar un firmware en la placa Arduino Uno. Este firmware se puede cargar desde el propio programa mBlock de la siguiente forma: Primero seleccionamos la placa “*Me UNO Shield*” desde la pestaña *Placas*, luego el puerto y finalmente, en el menú conectar hacemos clic en “Actualizar Firmware”. O, simplemente, si lo preferimos, cargamos el código desde el IDE de Arduino como ya hemos hecho en el algún ejemplo en el punto anterior.

NOTA: Si se quiere programar la placa Arduino Uno directamente desde el IDE de arduino, programándola en C++, obviamente, no hace falta instalar ningún firmware.



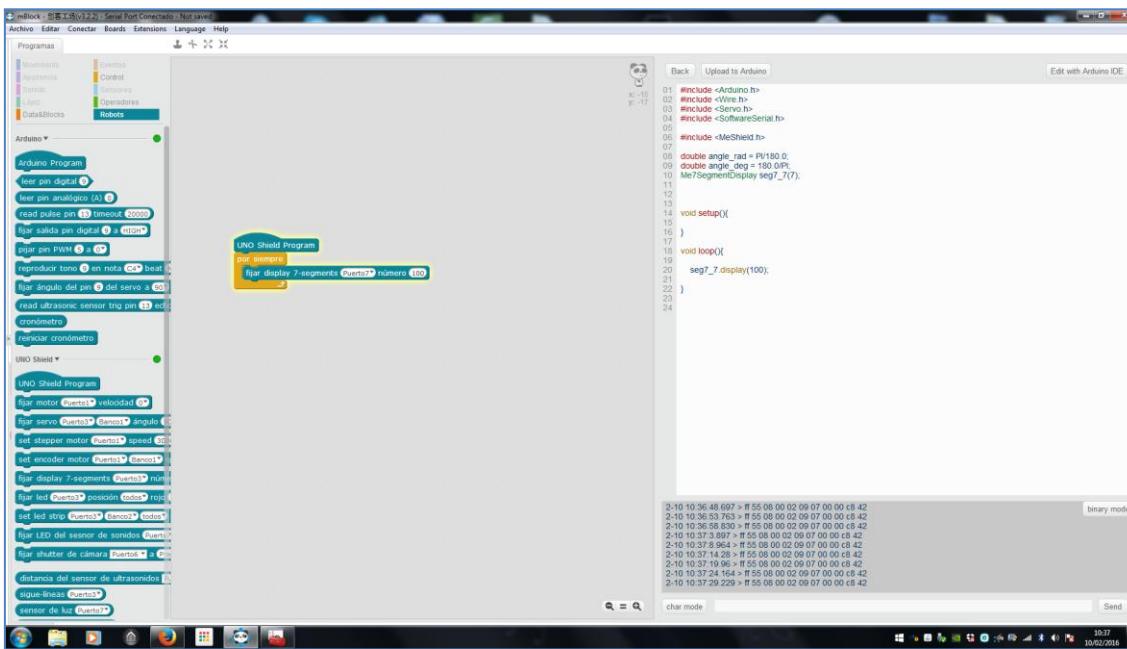
En la siguiente imagen, a modo de ejemplo, se ha conectado el módulo *display 7 segmentos* al puerto 7 del shield de arduino. En el programa se le dice que, simplemente, escriba el número 100. Después, en el *Modo Arduino* se hace clic en *Upload to Arduino*:



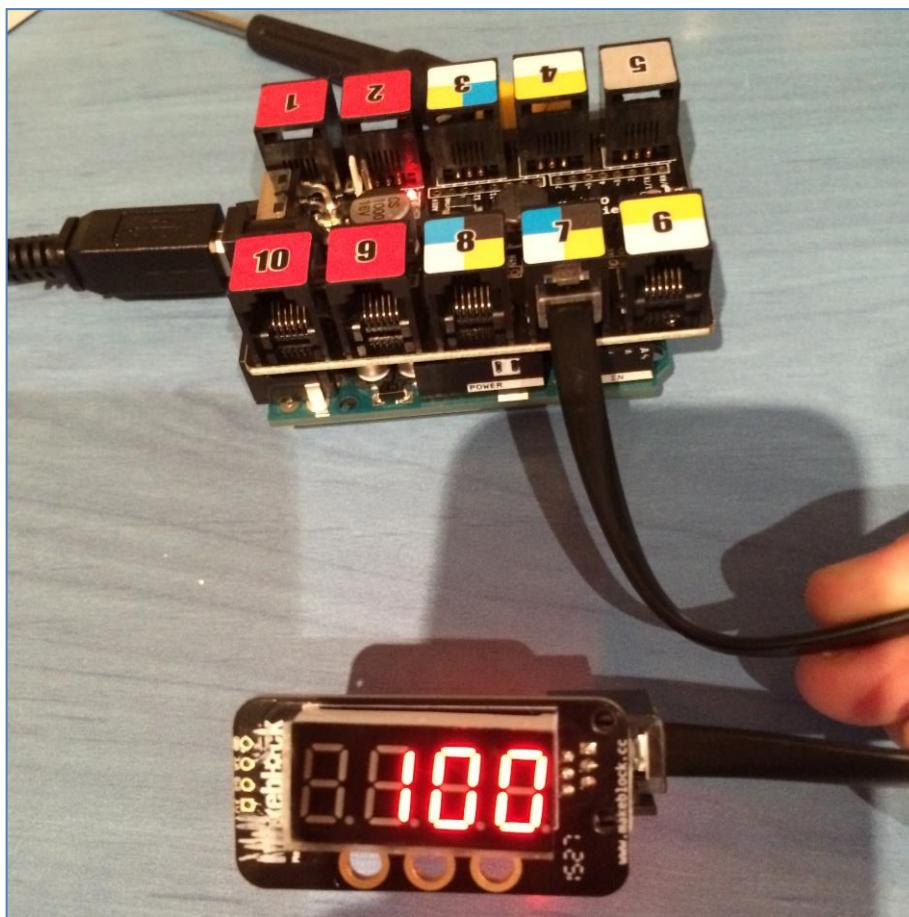


```
01 #include <Arduino.h>
02 #include <Wire.h>
03 #include <Servo.h>
04 #include <SoftwareSerial.h>
05
06 #include <MeShield.h>
07
08 double angle_rad = PI/180.0;
09 double angle_deg = 180.0/PI;
10 Me7SegmentDisplay seg7_7(7);
11
12
13
14 void setup(){
15 }
16
17 void loop(){
18     seg7_7.display(100);
19 }
20
21
22 }
23
24
```

Incluso, si quisiéramos, podríamos editar del código arduino desde el IDE de Arduino haciendo clic en *Edit with Arduino IDE*.

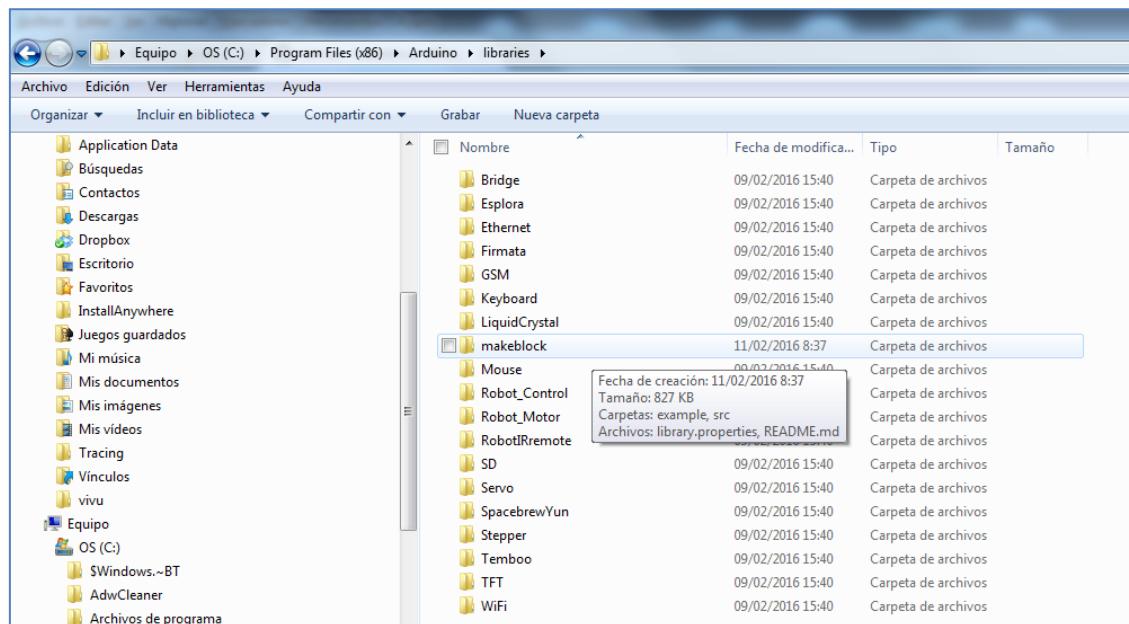


Ahora, ya con el programa cargado, podemos ver el resultado esperado en la siguiente imagen:



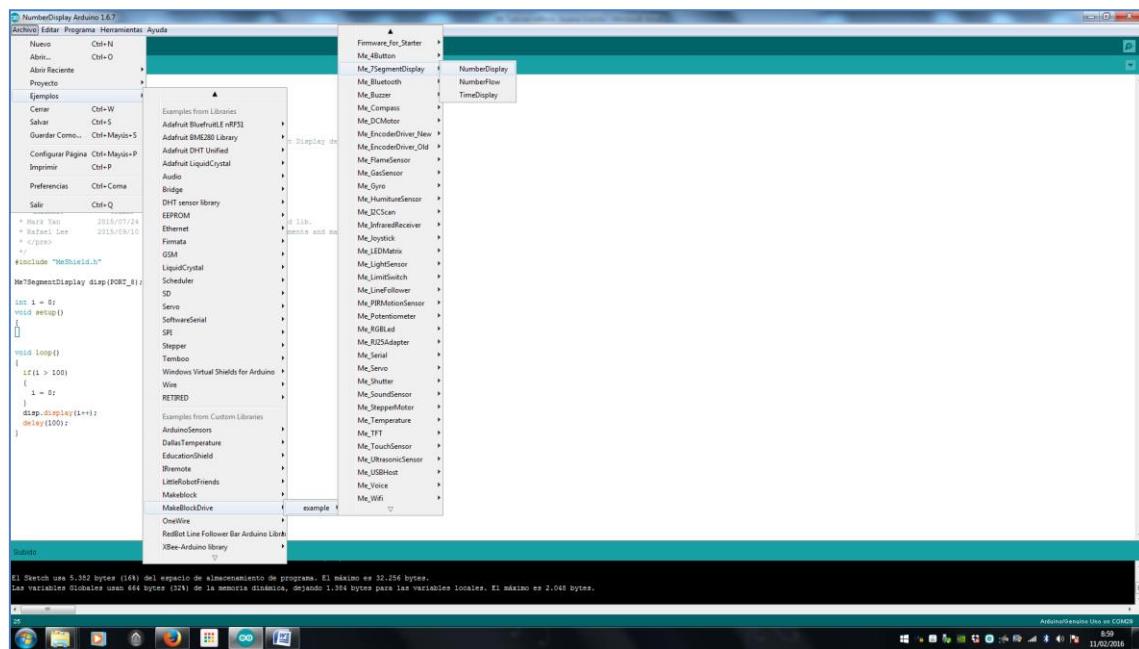
Para trabajar con el Arduino IDE debemos instalar las librerías. Podemos descargarlas del siguiente link: <https://github.com/Makeblock-official/Makeblock-Libraries>

Para instalarlas, copiamos la carpeta “makeblock” en la carpeta “libraries” de Arduino:



Divirtiéndome con mBot: Guía de manejo y programación

En el Arduino IDE, en ejemplos (dentro de la pestaña Archivo), podemos abrir algún archivo de esta carpeta. Por ejemplo, el archivo “NumberDisplay” para nuestro display 7 segmentos, que lo que hará será mostrarnos los dígitos del 1 al 100:



Tal y como hemos hecho otras veces en este documento, como vamos a usar el shield de arduino uno, debemos modificar la línea `#include "MeOrion.h"` por `#include "MeShield.h"` (ver siguiente imagen):

```
#include "MeShield.h"

Me7SegmentDisplay disp(PORT_8);

int i = 0;
void setup()
{
}

void loop()
{
    if(i > 100)
    {
        i = 0;
    }
    disp.display(i++);
    delay(100);
}
```

Divirtiéndome con mBot: Guía de manejo y programación

Tras cargar el programa a la placa Arduino Uno, vemos que nuestro display 7 segmentos, conectado al puerto 8, muestra los dígitos del 1 al 100.

Si en lugar del shield de Arduino Uno usamos otra placa de Makeblock, los cambios del comando o sentencia *include* serían los que se muestran en el punto 5 de la siguiente imagen:

Makeblock Library v3.23

Revision of history: Author Time Version Descr Mark Yan 2015/07/24 3.0.0 Rebuild the old lib. Rafael Lee 2015/09/02 3.1.0 Added some comments and macros. Lawrence 2015/09/09 3.2.0 Include some Arduino's official headfiles which path specified. Mark Yan 2015/11/02 3.2.1 fix bug on MACOS. Mark Yan 2016/01/21 3.2.2 fix some library bugs. Mark Yan 2016/05/17 3.2.3 add support for MegaPi and Auriga Board.

Arduino Library for Makeblock Electronic Modules

This library is v3.23, we've made a lot of modifications in the code structure. There is now a very big difference from v2.0. We strongly recommend that all Makeblock customers use the new version.

How to use:

1. Download the source from the git <https://github.com/Makeblock-official/Makeblock-Libraries>
2. copy the makeblock folder to your arduino default library. Your Arduino library folder should now look like this (on Windows): [arduino installation directory]\libraries\makeblock\src (On MACOS): [arduino Package Contents]\contents \Java\libraries\makeblock\src
3. Open the Arduino Application. (If it's already open, you will need to restart it to see changes.)
4. Click "File-> Examples". Here are some test programs in MakeBlockDrive-> example
5. Depending on the type of board you're using, you need to modify the header file to match. For example, if you're using a mCore. You should change "#include " to "#include " Corresponding boards and their header file are:

Orion <-----> MeOrion.h
BaseBoard <-----> MeBaseBoard.h
mCore <-----> MeMCore.h
Shield <-----> MeShield.h
Auriga <-----> MeAuriga.h
MegaPi <-----> MeMegaPi.h

A veces, nos interesa incorporar nuevas librerías para controlar otros dispositivos (como por ejemplo una LCD) desde mBlock en una placa Arduino Uno. Esto no es sencillo, pero puede hacerse.

Para hacerlo, *según el foro de la casa Makeblock*, debemos modificar el archivo de extensiones de mBlock. Los pasos a seguir para conseguirlo, son los siguientes:

1. Ir a la carpeta donde tengamos instalado el mBlock.

Y dentro de ésta, al directorio /ext/libraries/arduino

2. Añadir nuevos bloques

Divirtiéndome con mBot: Guía de manejo y programación

Editamos el archivo *Arduino.s2e* y veremos las especificaciones de bloques.

3. Ejemplo: Vamos a crear un bloque de ejemplo llamado **Comentario**. El bloque de ejemplo hace lo siguiente:

- Incluimos una nueva librería en la cabecera del programa
- Añadimos un comentario con el texto que le pasemos en Scratch a la sección Setup y Loop
- Hacemos una llamada a una función de la nueva librería

```
["w", "Comentario%n", "", "Comentario",
 {"encode": "{s0}", "setup": "", "inc": "#include
<nueva_libreria.h>", "def": "", "work": "//{0} en
setup;\n", "loop": "//{0} en loop;\n nueva_libreria();"}],
```

OJO con la sintaxis en JSON, debemos respetar la coma final.

Este bloque lo podemos añadir, por ejemplo, después del bloque de **read Analog Pin**, quedando así:

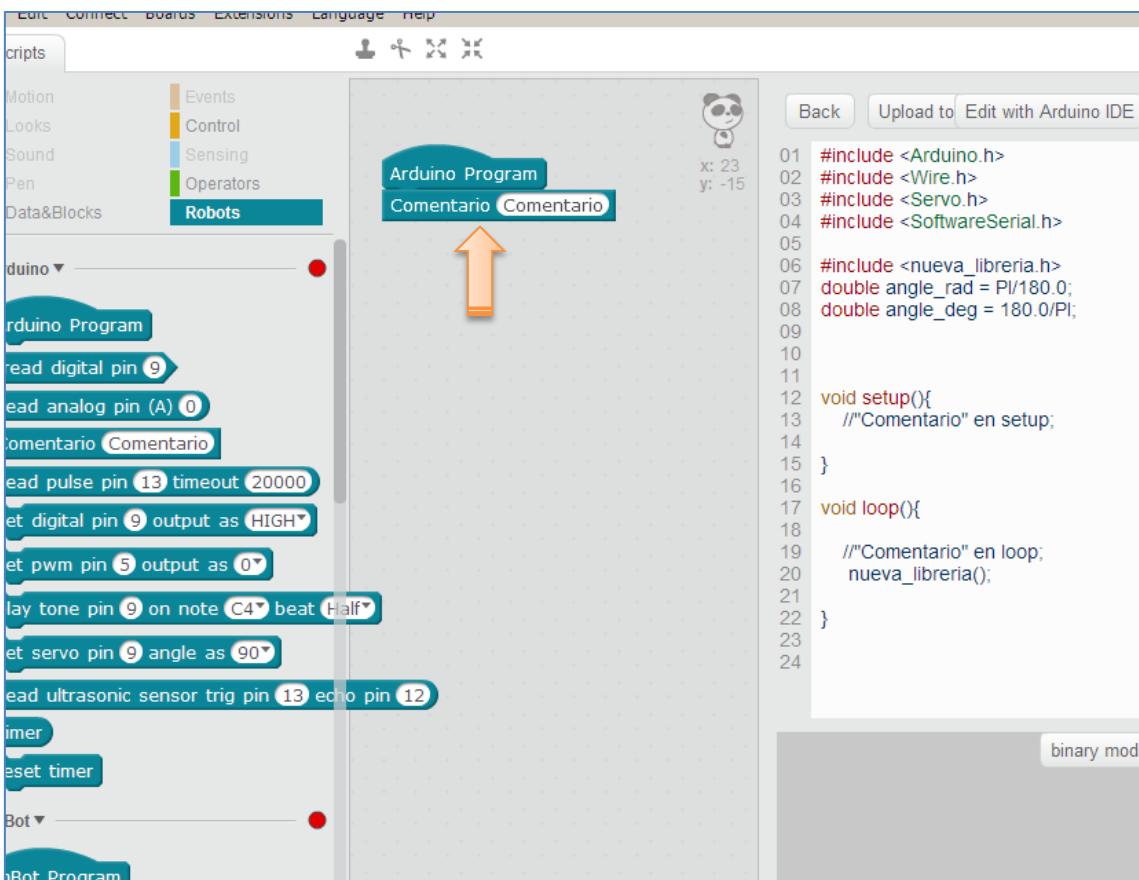
```
["R", "read analog pin (A)%n", "getAnalog", "0",
 {"encode": "{d0}", "setup": "pinMode(A0+{0}, INPUT); \n", "inc": "", "de
f": "", "work": "analogRead(A0+{0})", "loop": ""}],
```

```
["w", "Comentario%n", "", "Comentario",
 {"encode": "{s0}", "setup": "", "inc": "#include
<nueva_libreria.h>", "def": "", "work": "//{0} en
setup;\n", "loop": "//{0} en loop;\n nueva_libreria();"}],
```

...

Se observa cómo están construidos los demás bloques, tanto de Arduino como de mBot, y para qué sirve cada parámetro. El resultado es el siguiente:

Divirtiéndome con mBot: Guía de manejo y programación



A la hora de probarlo, nos puede ocurrir alguna de las siguientes incidencias:

- **Lo he cambiado y en el Scratch no aparece el bloque**

Eso es porque mBlock crea una copia de los directorios más importantes dentro de tu sistema. Por ejemplo, en Windows 7 la crea en:

[Usuario]/AppData/Roaming/com.makeblock.mBlock.v3.3.0

- Cierra el mBlock
- Elimina esa carpeta cada vez que modifiques el mBlock
- Abre nuevamente el mBlock

- **No veo los bloques de Arduino, sólo los de mBot**

Activa la extensión de Arduino desde el menú "extensiones"

5.3. Otras placas con mBlock

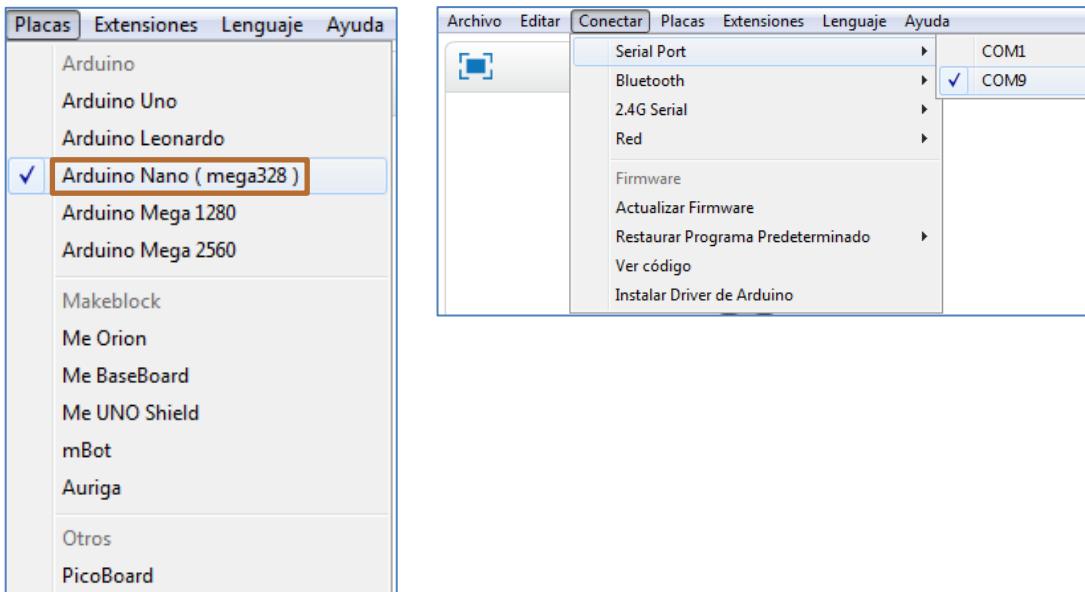
Si nos fijamos en las opciones de la pestaña *Placas* del software mBlock, podemos programar una gran variedad de placas con scratch. Dentro del universo arduino tenemos: Arduino Uno (ya ejemplificada en muchas partes de este documento), Leonardo, Nano y Mega. Pero, a mayores, podemos programar la conocida tarjeta de sensores Picoboard.

A continuación mostraremos sólo dos ejemplos: uno con la tarjeta Nano y otro con la PicoBoard.

5.3.1. Ejemplo 1: Semáforo con Arduino NANO trabajando con mBlock

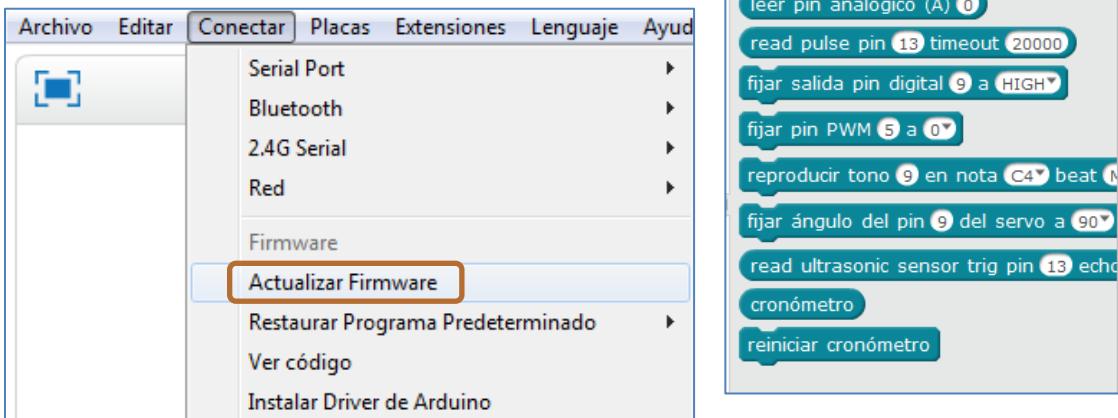
Desde mBlock podemos programar con scratch un gran abanico de placas arduino. A saber: Arduino Uno, Leonardo, Nano y Mega. El ejemplo que vamos a desarrollar podríamos haberlo implementado con una placa Arduino Uno pero, en este caso, he decidido hacerlo con una Arduino NANO.

Tras abrir el software mBlock escogemos la placa Arduino Nano (en la pestaña Placas) y la conectamos a su puerto COM correspondiente (Conectar > Serial Port):



Si vamos al bloque *Robots*, nos encontramos con los comandos de la extensión Arduino en activo (círculo verde):

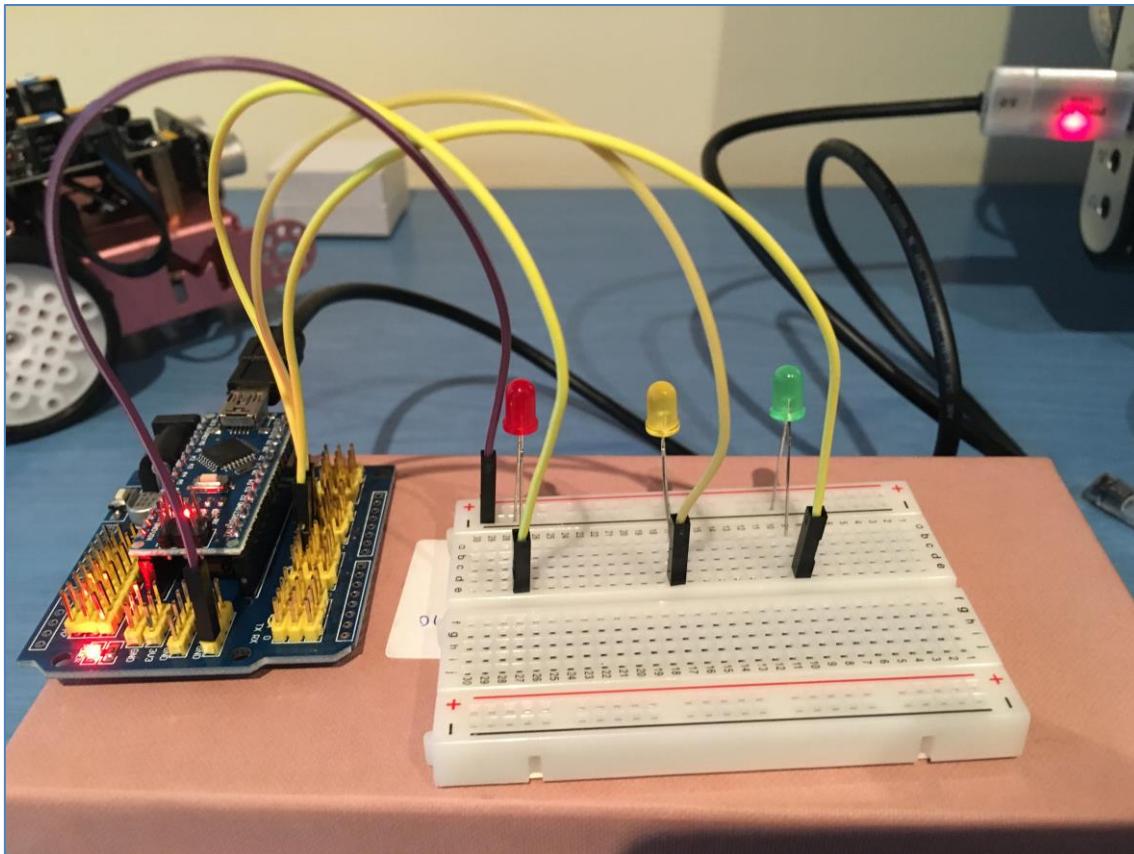
Antes de comenzar a programar la placa, debo introducirle el firmware adecuado a la misma. Esto podemos hacerlo desde mBlock, en la pestaña Conectar > Actualizar Firmware:



Divirtiéndome con mBot: Guía de manejo y programación

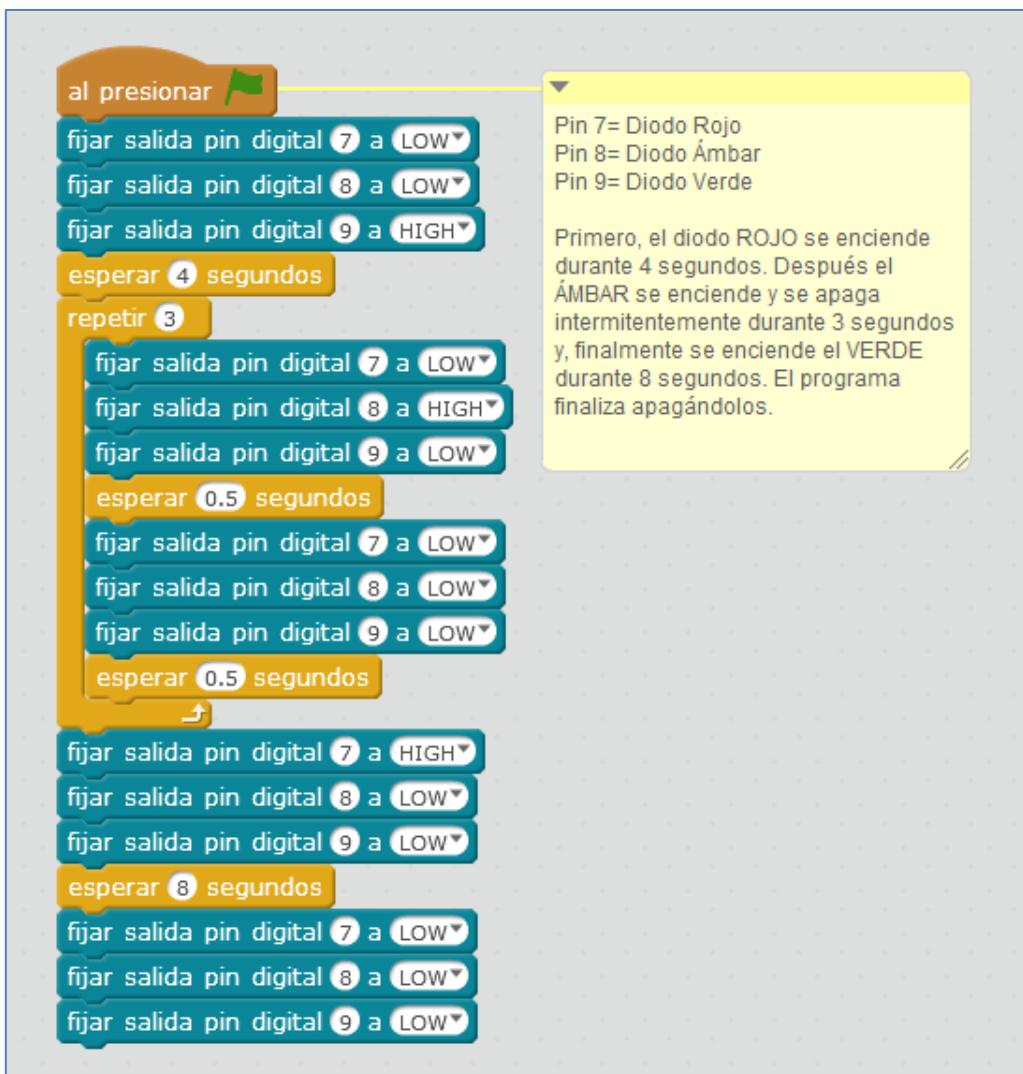
Tras la actualización del firmware nuestro programa mBlock se encuentra preparado para programar la tarjeta NANO. A continuación, implementamos el circuito eléctrico de nuestro semáforo:

He decidido conectar los pines 7, 8 y 9, respectivamente, a los diodos LEDs rojo, ámbar y verde que simularán las luces de nuestro semáforo. Por comodidad utilizo el shield de arduino NANO de la figura de abajo. En él, los pines digitales presentan tres opciones GVS, siendo “S” la señal y que representará al ánodo de cada diodo. Los cátodos de los diodos LEDs van dirigidos a tierra a través del cable lila:



Al ser este nuestro primer ejemplo, el semáforo será sencillo. Lo programaremos con mBlock, de modo que:

- Primero se encienda el LED rojo durante 4 segundos.
- Despues se de paso al LED ámbar, que debe actuar intermitentemente (ON-OFF) durante 3 segundos y continúe el LED verde, que se encenderá durante 8 segundos.
- Finalmente, los 3 LEDs deben apagarse.

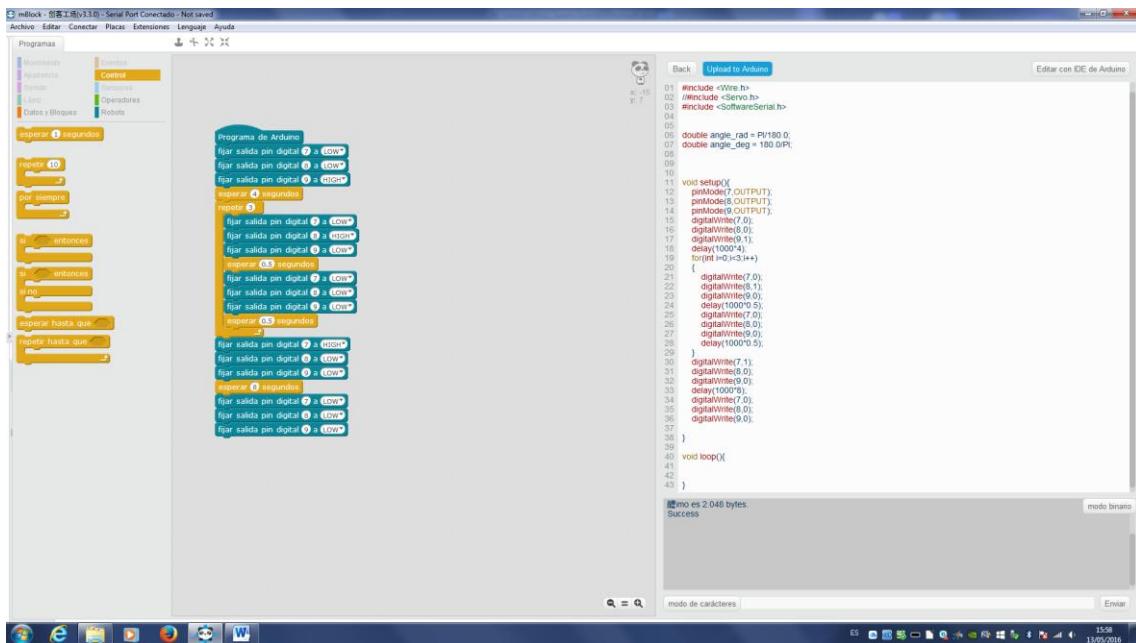


Tras probar el circuito y ver que funciona, podemos cargarle el programa a la placa y así evitar el cable USB en nuestro proyecto. Esto lo conseguimos en dos pasos: Primero cambiaremos el comando de la bandera verde del bloque **Eventos** por el comando **Programa de Arduino** del bloque **Robots**:



Divirtiéndome con mBot: Guía de manejo y programación

Y por último, vamos a la pestaña *Editar > Modo Arduino* y hacemos clic en *Upload to Arduino*. El software mBlock comienza a llamar a sus librerías y a procesar los comandos que hemos usado. Sólo nos queda esperar a que nos informe, mediante un mensaje, que se ha cargado exitosamente el programa:



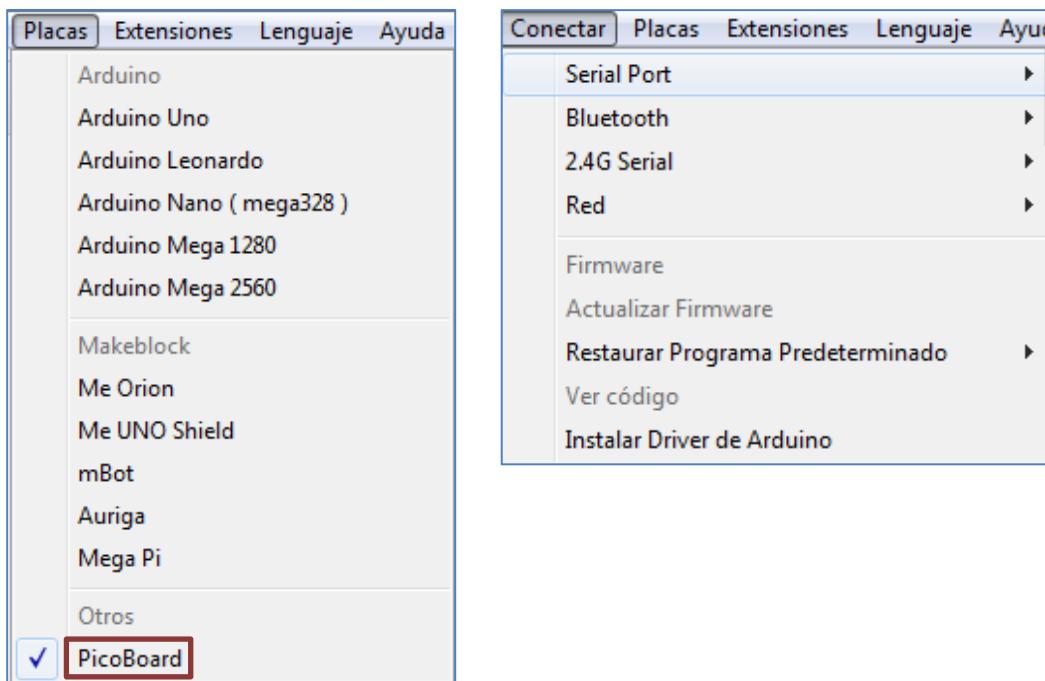
Finalmente, ya podemos eliminar el cable USB de la placa y alimentar la placa con las baterías adecuadas.

5.3.2. Ejemplo 2: Naves invasoras del espacio con PicoBoard trabajando con mBlock.

La PicoBoard es una tarjeta de sensores que presenta los siguientes componentes incrustados en ella: el deslizador o slider, el pulsador, el sensor de sonido y el sensor de luz. A mayores posibilita incluir cuatro componentes electrónicos usando sus 4 conectores por medio de pinzas de cocodrilo. Esta tarjeta nos permite recoger información (datos) del mundo real. Al conectarla al mBlock se nos abre un abanico de posibilidades ya que hace posible interpretar estas mediciones externas, consiguiendo programar y crear proyectos que respondan a esos efectos instantáneos del mundo físico que se están sensorizando.

Antes de programarla debemos conectar la PicoBoard. Para ello seleccionamos la tarjeta PicoBoard en la pestaña *Placas* del programa mBlock, así como, el puerto de conexión correspondiente en la pestaña *Conectar*.

Divirtiéndome con mBot: Guía de manejo y programación

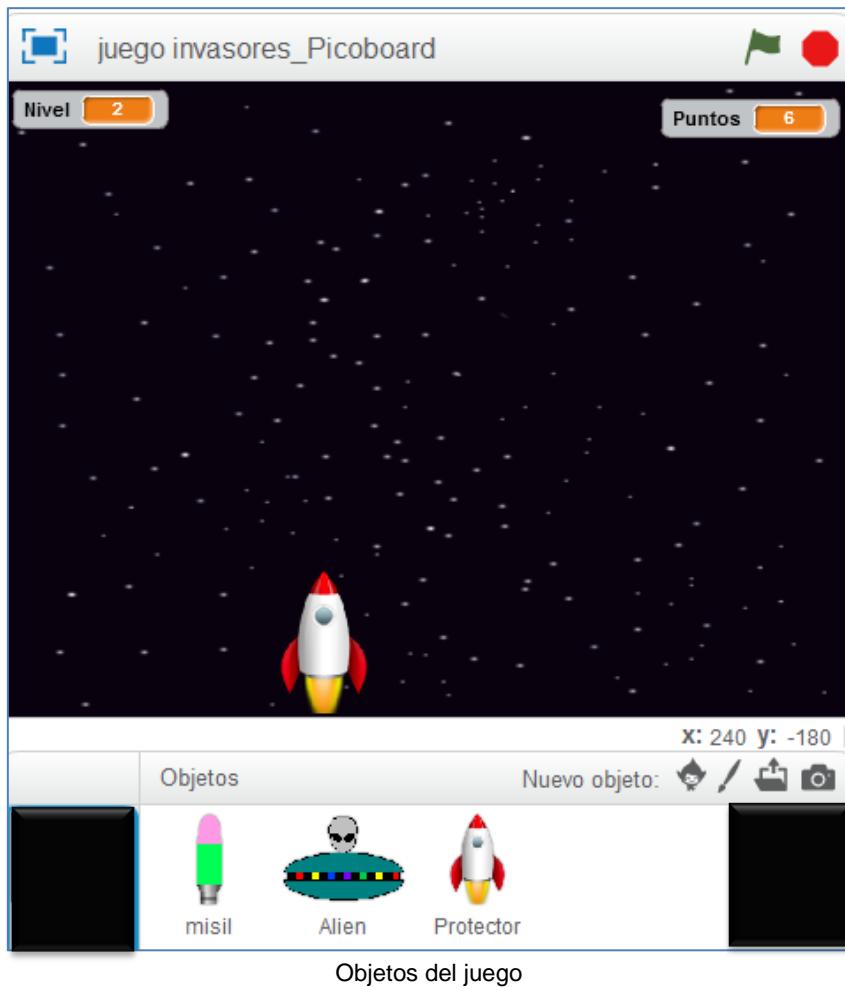


En el mercado actual existen diferentes modelos de picoboard. En la imagen inferior podemos ver el modelo más antiguo, modelo que soporta el software mBlock. También es compatible con el programa la siguiente versión de la picoboard. Sólo se necesita instalar el driver de cada respectiva tarjeta de sensores.



PicoBoard (Scratch Sensor Board)

Vamos a ejemplificar su uso con un juego. El programa “Naves invasoras del espacio” dispone de 3 objetos: Misil, Alien y la Nave Protectora. Lo que nos interesa, para este apartado, es la programación del slider, del botón y del sensor sonido de la PicoBoard en diferentes partes del juego.



La nave protetora se mueve por el escenario en el sentido horizontal gracias al desplazamiento del slider o deslizador de la Picoboard. Su script de movimiento es el siguiente:



Movimiento de un objeto en el eje X con el slider

Divirtiéndome con mBot: Guía de manejo y programación

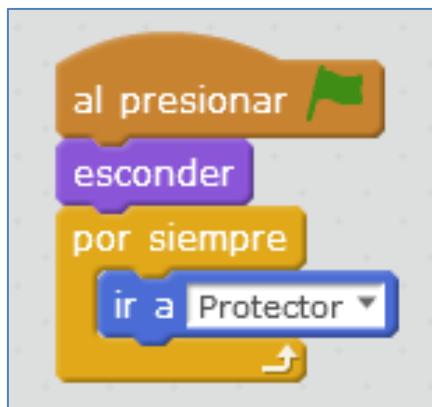
La explicación del por qué del factor de corrección 3,5 es el siguiente:

Si fijo la X al valor del deslizador, sólo se moverá 100 píxeles, desde la abscisa $x=0$ a $x=100$. Pero me interesa que recorra todo el escenario. Para que el objeto se vea completo y no cortado, decido que mi variable X del escenario, con el deslizador, no puede superar 175 píxeles, ni puede ser menor que -175. De esa forma, calculo el factor de corrección:

$$(0 - 50) \cdot X = -175 \text{ (Para el valor mínimo)}$$
$$(100 - 50) \cdot X = 175 \text{ (Para el valor máximo)}$$

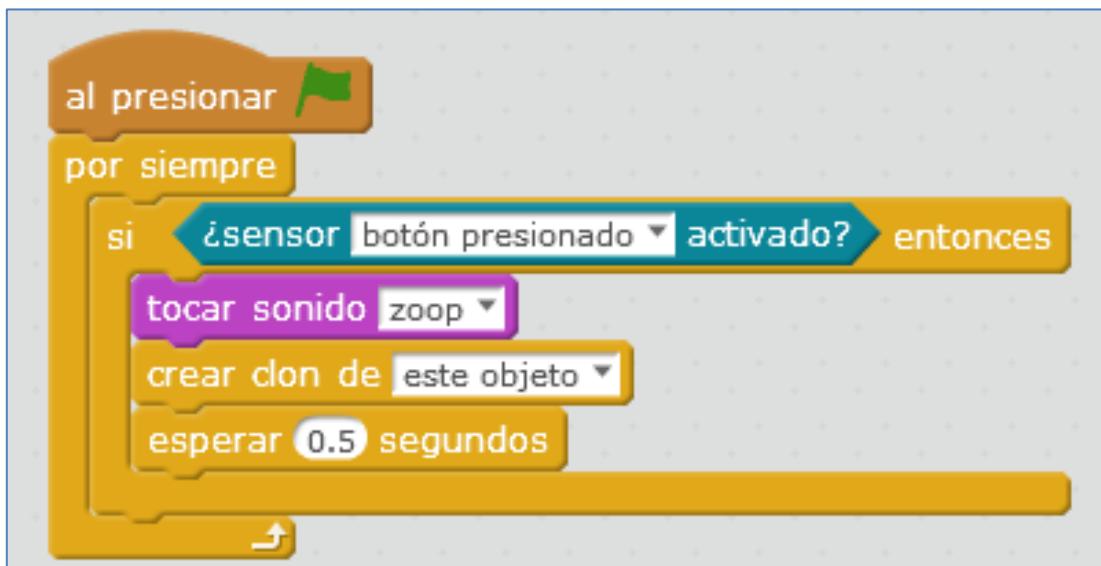
Con estas ecuaciones, el factor de corrección (X) es el mismo, y pasa a tomar el valor 3,5.

El objeto “misil” se dispara desde la nave protectora. De hecho, al presionar la bandera verde, el comando azul “ir a Protector” hace que el misil se sitúe y se esconda en la nave Protector:



Parte del script del objeto “Misil”

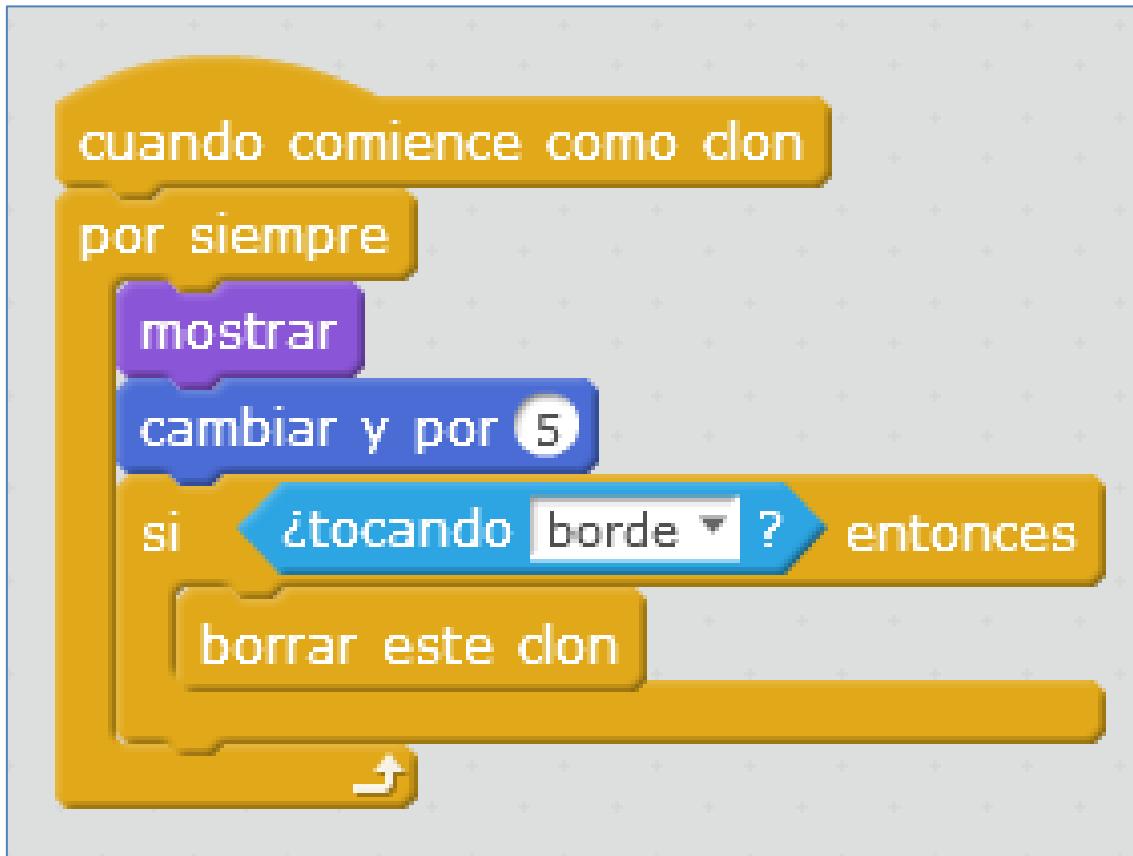
Cada vez que se presione el botón de la PicoBoard, se crea un clon del objeto misil cada 0,5 segundos:



Parte del script del objeto “Misil” con el botón de la PicoBoard

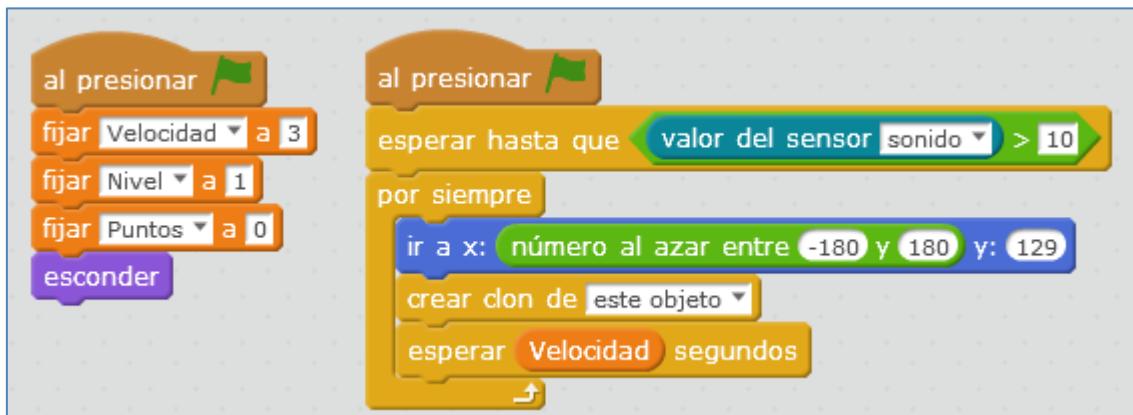
Divirtiéndome con mBot: Guía de manejo y programación

Estos clones están escondidos porque el objeto principal, misil, tambien lo estaba. Por lo tanto, cuando comiencen como clon, deben mostrarse y realizar el movimiento que necesitemos. En nuestro caso, subir 5 pasos verticalmente y desaparecer cuando lleguen al borde superior del escenario:



Parte del script del objeto "Misil"

Referente al sensor sonido, podemos hacer que el juego, como juego que se precie, se juegue con una música o sonido determinado. En este caso, he decidido que las naves alienígenas comiencen a clonarse cuando el sonido de la canción supere el valor de 10:



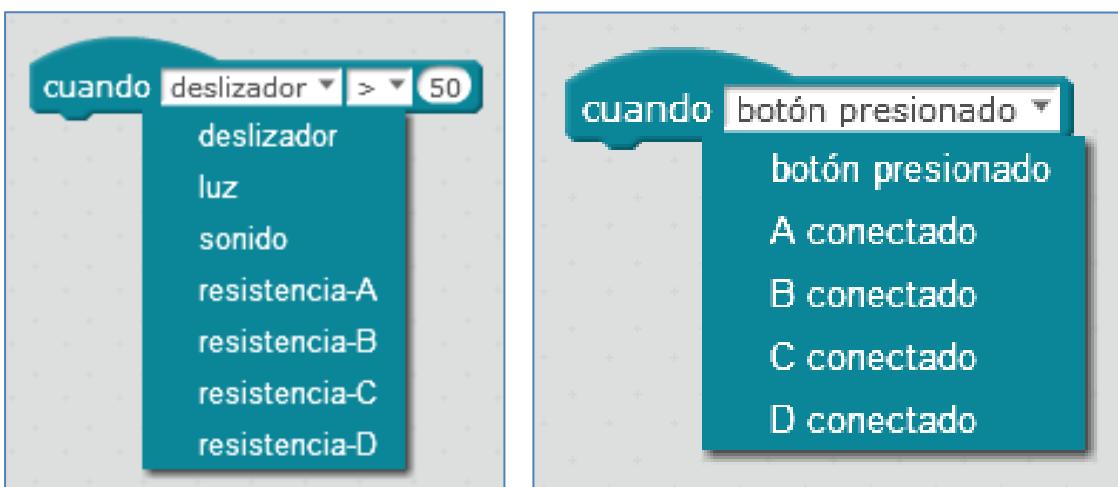
Parte del script del objeto "Alien" con el sensor de sonido

El sonido que detecta, si estamos callados, lo recogerá de la canción y que, en este caso, se ha programado en el escenario:



Script del escenario

Podríamos haber programado el sensor de luz y otros sensores que conectáramos a los conectores A, B, C y D (funcionando o no como resistencias), utilizando los comandos que se ven a continuación, pero para este juego no nos han hecho falta:

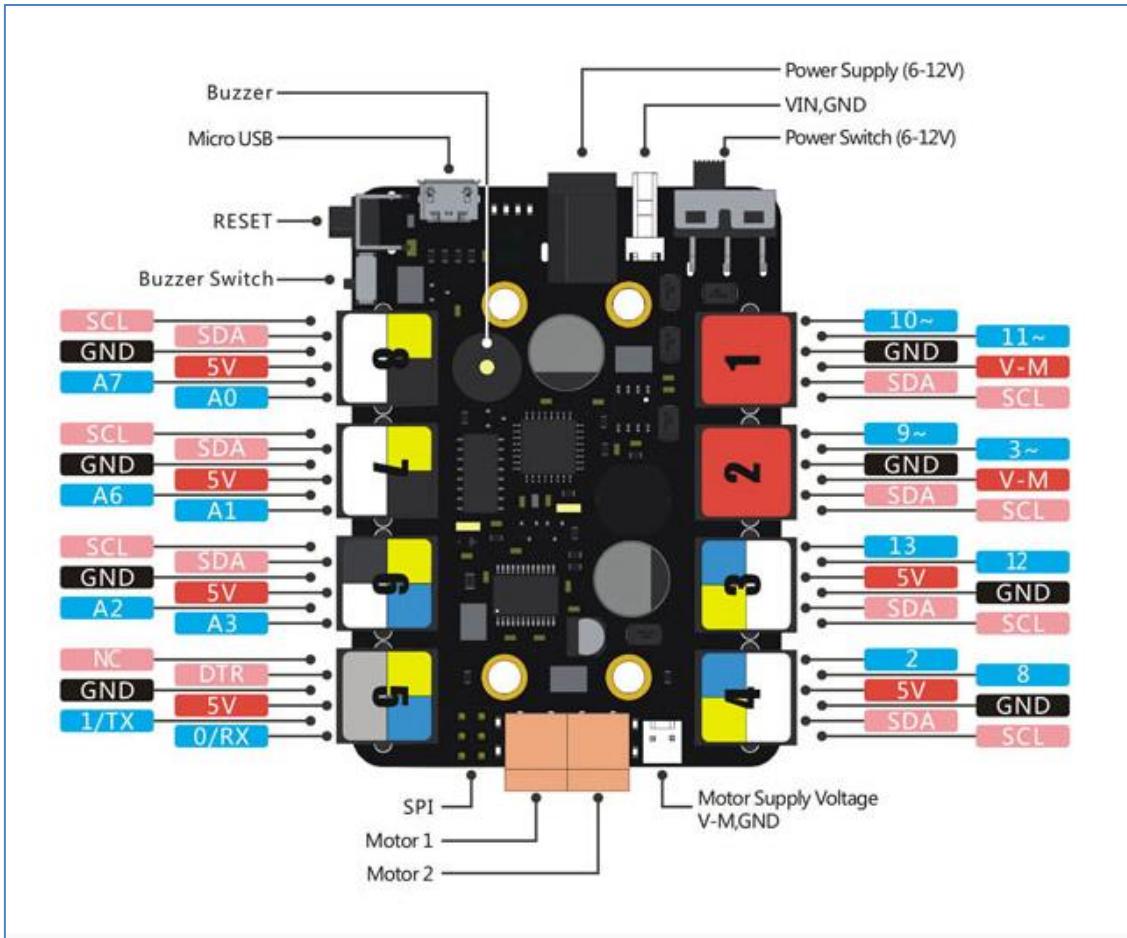


6. Placa Orion Base Board

La placa base Orion de la casa Makeblock también es una placa basada en Arduino UNO. En ella encontramos 8 puertos RJ11⁵ para 8 conectores RJ25 y 2 drivers de motores DC (conectores naranjas), alimentándose con una tensión de 6 a 12V.

Esta placa es la que utiliza el robot Starter de la casa Makeblock y puede ser programada con scratch, desde mBlock, o con código arduino desde el IDE de Arduino.

⁵ Ahora llamados RJ25



Placa Orion

La descarga de las librerías Makeblock para esta placa, es accesible en el siguiente link:

<https://github.com/Makeblock-official/Makeblock-Libraries>

http://www.makeblock.es/tutoriales/librerias_makeblock_arduino/

Algunos ejemplos de programación se pueden ver en el siguiente enlace:
http://www.makeblock.es/tutoriales/ejemplos_programacion_arduino/

- Motores DC: Los motores DC de Makeblock pueden conectarse tanto a los puertos M1 y M2 de la propia placa, o a los puertos 1 y 2 a través de una controladora de motores DC.

```
#include "makeblock.h"
#include "arduino.h"
#include "softwareserial.h"
#include "wire.h"

MeDCMotor motorDriver1(M1);      // Un motor directamente a la placa
MeDCMotor motorDriver2(PORT_2);  // Otro motor al puerto 2

uint8_t motorSpeed = 100;

void setup()
{
}

void loop()
{
    motorDriver1.run(motorSpeed); // valor entre -255 y 255.
    motorDriver2.run(motorSpeed); // valor entre -255 y 255.
    delay(2000);
    motorDriver1.run(-motorSpeed); // Cambio de sentido de giro
    motorDriver2.run(-motorSpeed);
    delay(2000);
    motorDriver1.stop();         // Parar motores
    motorDriver2.stop();
    delay(2000);
}
```

- Servomotores: Los motores servo se pueden conectar a la placa base a través de un adaptador RJ25 capaz de mover 2 servos.

```
#include "Makeblock.h"
#include "Arduino.h"
#include "SoftwareSerial.h"
#include "Wire.h"

MeServo servoDriver(PORT_2, 1); // La controladora al puerto 2, y el servo al puerto 1 de la controladora

void setup()
{
    servoDriver.begin();
}

void loop()
{
    servoDriver.begin();
    servoDriver.write(0);
    delay(1000); // Esperamos a que el servo alcance su posición
    servoDriver.write(180);
    delay(1000); // Esperamos a que el servo alcance su posición
    servoDriver.detach(); // Soltamos el servo y no bloqueamos su posición
    delay(1000);
}
```

7. Referencias de interés

Makeblock España: <https://www.makeblock.es/>

Central Makeblock: <http://makeblock.com/>

