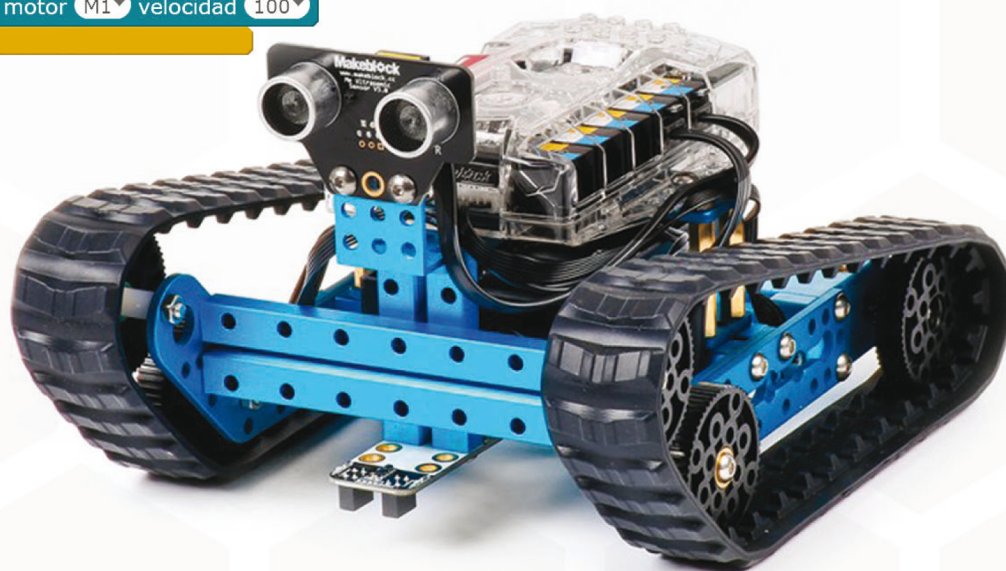


Divirtiéndome con Ranger



Título original: Divirtiéndome con mBot Ranger.

Autor: Susana Oubiña Falcón

Editor: Makeblock España.

1ª Edición: agosto, 2016

 Susana Oubiña Falcón

ISBN-13: 978-84-617-4422-0



"Divirtiéndome con mBot Ranger ", por Susana Oubiña Falcón, es publicado bajo la licencia [Creative Commons Reconocimiento 4.0 Internacional License](https://creativecommons.org/licenses/by/4.0/).

Índice

1. Introducción	3
2. mBlock.....	5
2.1. Menú principal	8
2.1.1. Archivo	8
2.1.2. Editar	8
2.1.3. Conectar	9
2.1.4. Placas.....	12
2.1.5. Extensiones.....	13
2.1.6. Lenguaje.....	16
2.1.7. Ayuda	17
2.2. Comandos de los bloques del mBlock	17
2.2.1. Datos y Bloques	18
2.2.2. Robots.....	19
2.2.3. Crear nuevas librerías desde mBlock	22
3. Modos de trabajo con el Kit mBot Ranger en mBlock	24
3.1. Robot.....	24
3.2. Controladora de juegos y creaciones.....	25
3.3. Placa de pruebas y laboratorio de electrónica	26
4. Ejemplos de programación con los sensores y componentes de Me Auriga.....	27
4.1. Anillo de LEDs RGB	27
4.2. Sensor de temperatura	29
4.3. Sensores de luminosidad.....	31
4.4. Sensor giróscopo.....	36
4.5. Sensor de sonido.....	38
4.6. Zumbador	39
5. Ejemplos de programación con diversos módulos	41
5.1. Módulo de ultrasonidos.....	42
5.2. Motores del robot.....	43
5.3. Motor y detección ultrasónica	48
5.4. Detector de línea	49
5.5. Matriz de LEDs 8×16	62
5.6. Robot sumo	68

5.7.	Servomotor	76
5.8.	Sensor de temperatura SD18B20	79
5.9.	Sensor de Me Temperatura y Me Humedad	84
5.10.	Módulo PIR	87
5.11.	Módulo Joystic	90
5.12.	Módulo Me Touch o sensor táctil.....	91
5.13.	Garras: Mini Gripper y Gripped	93
5.14.	Mini brazo articulado	101
5.15.	Sensor de ángulo	103
5.16.	Sensor de sonido	105
5.17.	Potenciómetro	108
5.18.	Potenciómetro slider	110
5.19.	Módulo 4LEDs RGB.....	114
5.20.	Tira de LEDs RGB	117
5.21.	Módulo Display 7 segmentos	118
5.22.	Módulo Brújula o Me Compass	122
5.23.	Sensor de gas.....	125
5.24.	Sensor de llama	134
5.25.	Sensor de luz	139
5.26.	Módulo 4 botones.....	142
5.27.	Pantalla TFT LCD 2.2.....	144
5.28.	Módulo Micro Switch	148
5.29.	Módulos controladores de motores	149
6.	Otras placas con mBlock	154
6.1.	Ejemplo 1: Semáforo con Arduino Nano sobre mBlock.....	156
6.2.	Ejemplo 2: Naves invasoras del espacio con PicoBoard bajo mBlock.....	159
6.3.	Ejemplo 3: Control de encendido/apagado de un LED por botón con Arduino Mega 2560 bajo mBlock.....	164
7.	Referencias	169

Divirtiéndome con mBot Ranger

1. Introducción

El robot mBot Ranger es el kit que la casa Makeblock define como su robot STEM (*Science, Technology, Engineering y Mathematics*). Un robot pensado para que alumnos/as de edades de 12 años en adelante exploren el mundo de la robótica. Es un kit resistente (chasis de aluminio) y versátil, en el sentido de que permite la construcción de tres montajes diferentes: robot tanque, robot de carreras de 3 ruedas y robot balancín.

Relativo a su alimentación, mBot Ranger necesita de 6 pilas o baterías de 1,5V del tipo AA.

Su placa controladora, protegida por una carcasa de plástico transparente, es una Me Auriga que acomoda 7 sensores diferentes. A saber: módulo con 12 LEDs RGB y dos sensores de luz, comunicación inalámbrica vía bluetooth/2.4G, sensor de sonido, sensor de temperatura, zumbador y giróscopo.

Esta placa, basada en el microprocesador open-source Arduino Mega 2560, dispone de 10 puertos de expansión con conectores RJ25. Entre ellos, 4 son específicos para motores de 6-12V, uno para conexiones de componentes por puerto serie y 5 para diferentes componentes que requieran conexiones digitales, analógicas o I²C. En estos últimos se pueden conectar el módulo sigue-líneas o el sensor de ultrasonidos que se incluyen en el kit del robot.

Es relevante conocer el significado de los colores ID que podemos encontrarnos en los puertos de las diferentes placas de Makeblock. Estos son:

Rojo (motores), Amarillo (interface digital), Azul (interface digital dual), Gris (Puerto serie, bluetooth), Negro (interface analógica y dual) y Blanco (Puerto I²C).



Colores ID de los puertos de las placas Makeblock

Particularizados a los diez puertos de la placa Me Auriga, 4 presentan el color rojo, uno el gris y 5 la siguiente combinación de cuatro colores, blanco-amarillo-azul-negro.

En la siguiente imagen (Fig1) podemos ver la placa Auriga del robot: En ella observamos que se le han acoplado (no soldado) el módulo bluetooth (que puede cambiarse por un módulo 2.4G) y el anillo de 12 LEDs RGB (módulo que incluye dos sensores de luz). En esta imagen frontal, distinguimos el sensor de sonido y el buzzer o zumbador incrustados en la placa.

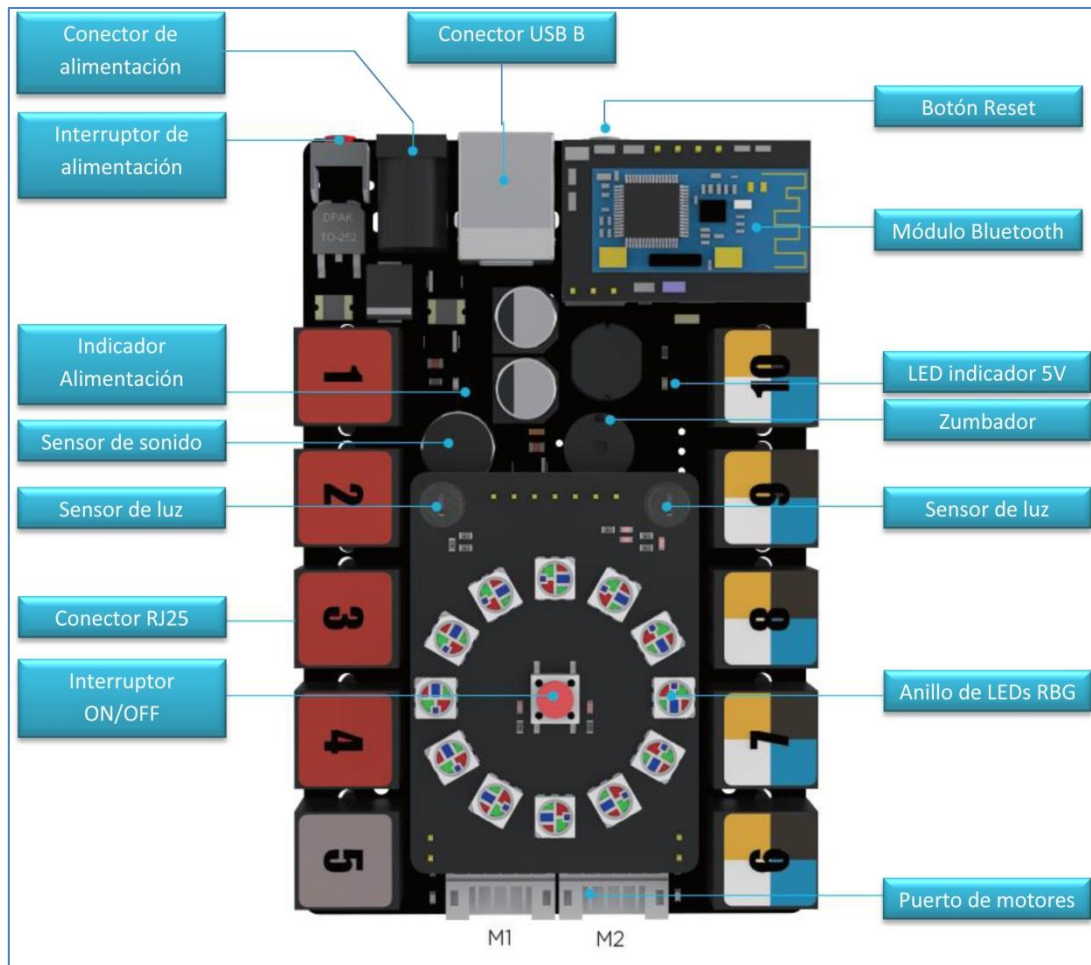
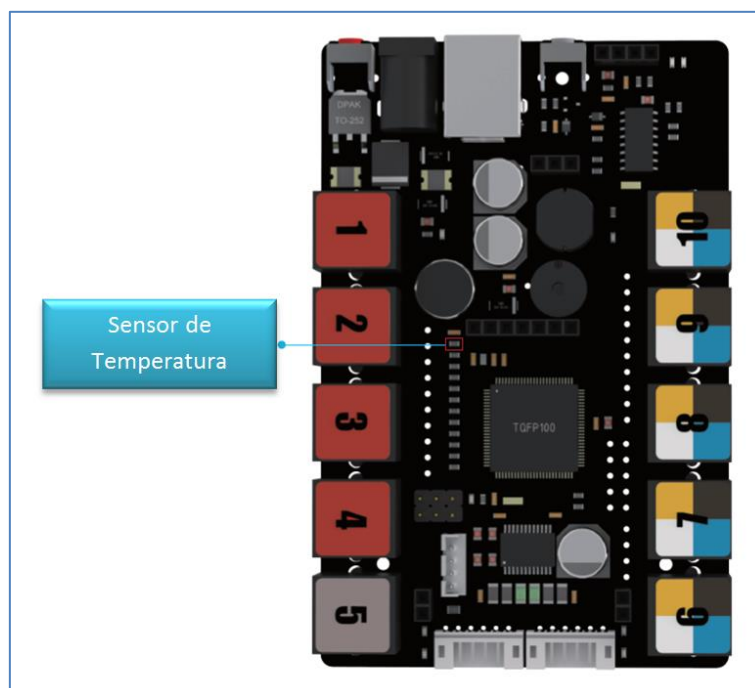
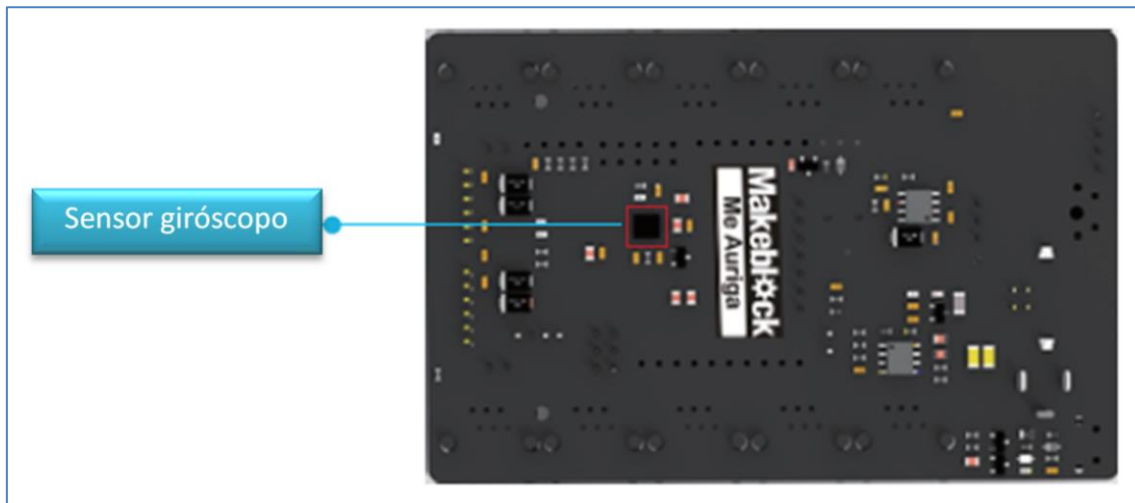


Fig1: Placa Me Auriga

Al extraer el módulo del anillo de LEDs RGB de la placa, localizamos el sensor de temperatura ambiente (termistor NTC):



Y al voltear la placa, en su parte trasera podemos ver el sensor giróscopo basado en el módulo MPU 6050, y que va incluido en la placa:



Me Auriga se basa en el micro-controlador ATmega2560, disponiendo de una memoria flash de 256Kb, y esto la hace ideal a la hora de necesitar implementar con ella proyectos más complejos.

Concerniente a su programación y control, el robot mBot Ranger presenta tres caminos o vías: con la APP Makeblock HD 2.0 (de libre descarga en App Store o Google Play), a través del software mBlock (Windows y MAC OSX) o con el IDE de Arduino (Windows, MAC OSX y Linux).

Este documento se centrará en la programación del robot mBot Ranger desde el programa mBlock. Software que veo muy apropiado para el alumnado de la ESO porque combina scratch con arduino en diferentes placas y robots (y no sólo con el mBot Ranger). A mayores, pretendo explicar y ejemplificar la utilidad de un gran abanico de accesorios, sensores y actuadores, algunos muy comunes en el campo de la robótica educativa y que nos servirán para llevar a cabo proyectos o retos más complejos, abarcando el desarrollo de prototipos reales más sofisticados. 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.

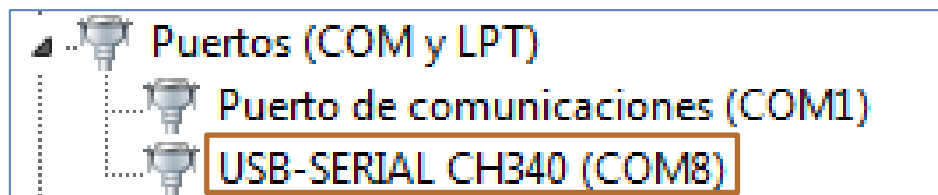
2. mBlock

mBlock es el software estrella de la casa Makeblock. Está basado en el editor Scratch 2.0 y permite programar el robot de forma inalámbrica, mediante un entorno sencillo de arrastrar y soltar comandos. Este entorno gráfico fusiona scratch y arduino, pudiendo ser utilizado no sólo para programar sus placas sino también para programar una gran cantidad de tarjetas del universo arduino, así como, la tarjeta de sensores PicoBoard que se asocia con scratch. La descarga del software mBlock, tanto para Windows como para iOS, puede hacerse desde el siguiente link: www.mblock.cc

Divirtiéndome con mBot Ranger

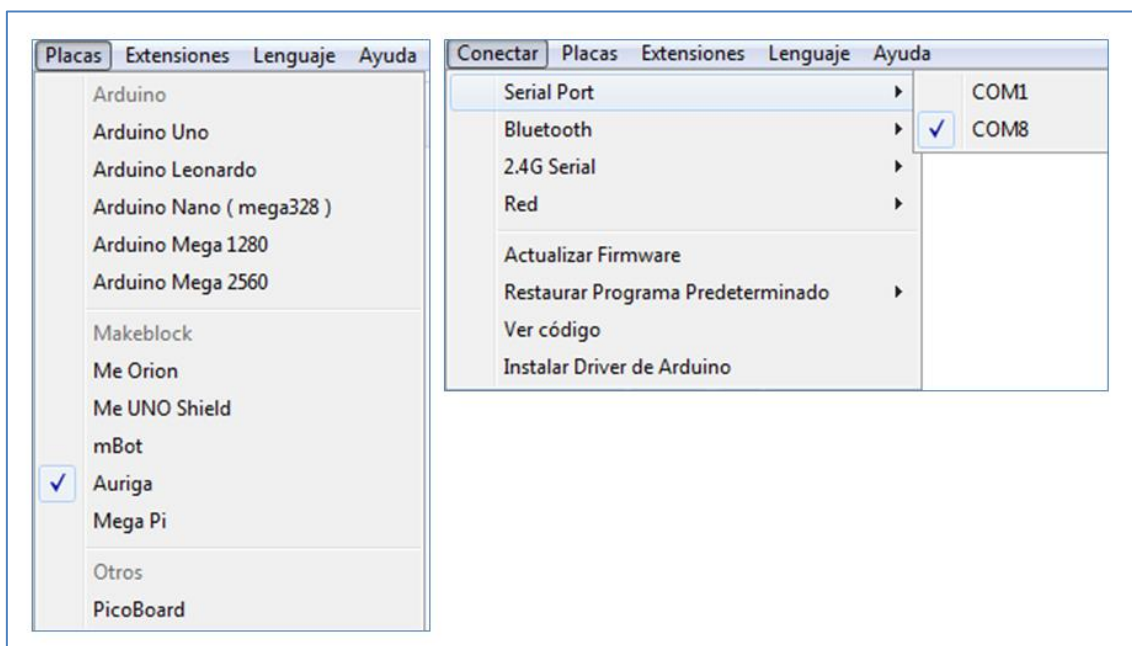
En cuanto a Linux¹, el programa funciona a través del Wine, software que nos permite instalar aplicaciones de Windows en Linux, dando permisos al usuario en el archivo ttyUSB0.

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 Ranger al ordenador. En ese momento, si los drivers de la placa están instalados, se nos habilita un puerto COM de comunicación para realizar la conexión. En nuestro caso, nuestro puerto es el COM8, como puede verse en la siguiente imagen. Si queremos conocer el puerto que utiliza nuestro PC en ese momento, en Windows podemos acceder a *Equipo>Administrador de dispositivos*, en el apartado de Puertos (COM y LPT):



Puerto de conexión USB-SERIAL CH340

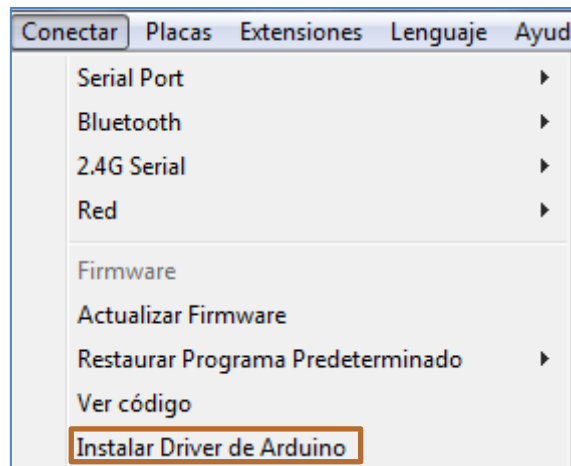
Ya con el programa mBlock abierto, vamos a la pestaña “Placas” y escogemos *Auriga* para finalmente, ir a la pestaña “Conectar” del menú principal y en su apartado *Serial Port*, seleccionar el puerto de conexión COM8:



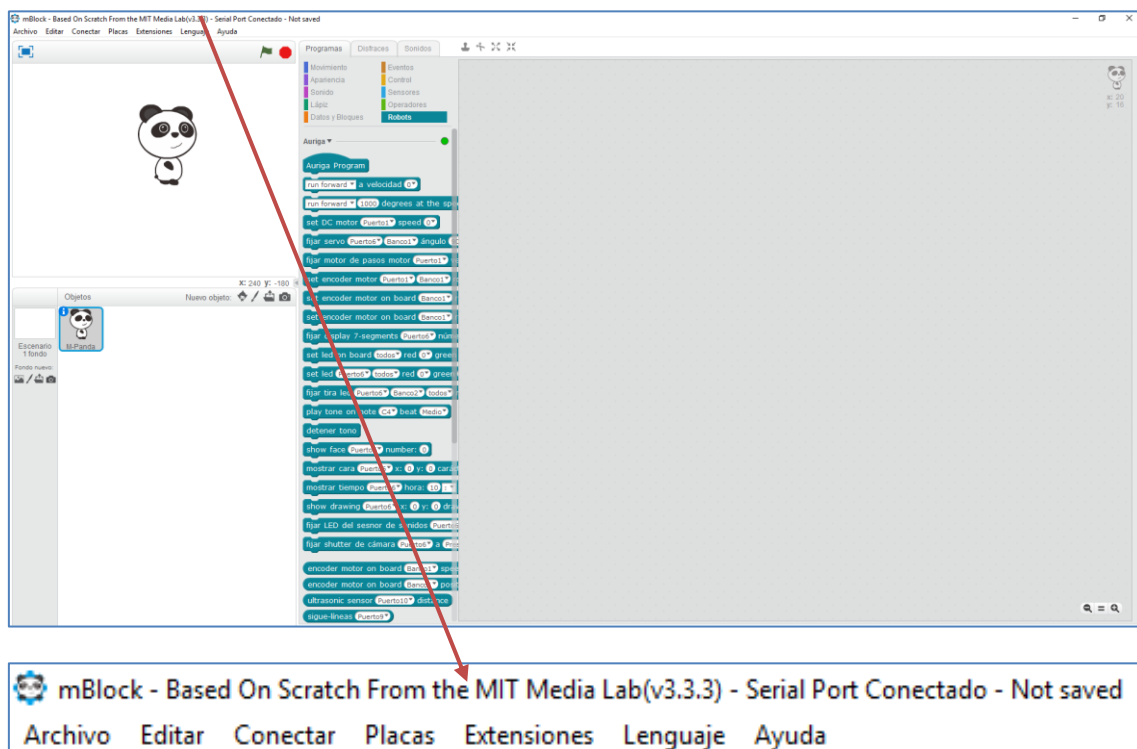
En caso de no conectar, sólo debemos revisar si hemos instalado el driver de arduino que nos proporciona el puerto de conexión USB-SERIAL CH340. Este driver puede instalarse directamente desde el programa mBlock, a través de la pestaña *Conectar* usando la opción *Instalar Driver de Arduino*, tal y como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

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

Divirtiéndome con mBot Ranger



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



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

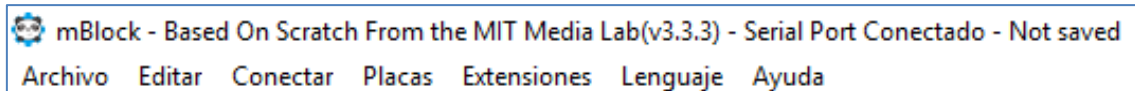


Divirtiéndome con mBot Ranger

Nuestro trabajo ahora consistirá en 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.

2.1. Menú principal

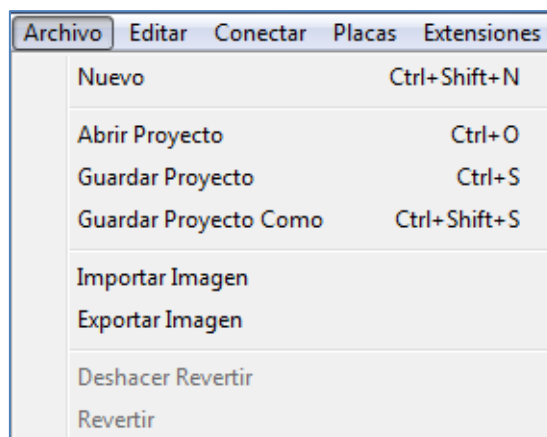
El menú principal se compone de siete pestañas, comenzando por la celda Archivo y finalizando en la celda de selección Ayuda. En su parte superior podemos ver la versión del programa que se está utilizando y si la placa que se pretende programar está o no conectada con el software.



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

2.1.1. Archivo

Las opciones de la pestaña “Archivo” son:

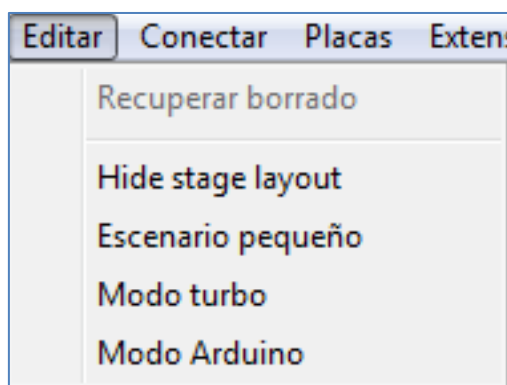


Opciones de la pestaña Archivo

- *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

2.1.2. Editar

Las opciones de la pestaña “Editar” son:

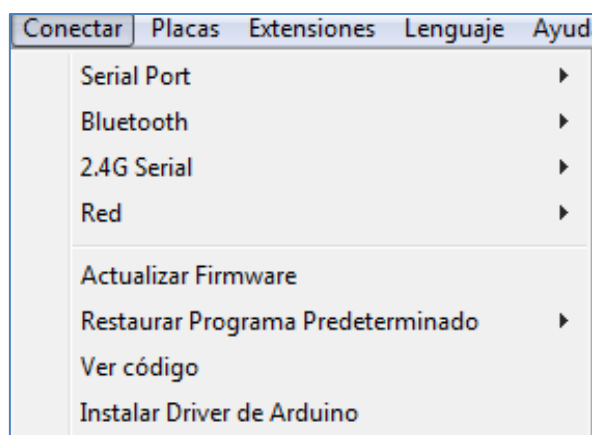


Opciones de la pestaña Editar

- *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.
- *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.

2.1.3. Conectar

El robot mBot Ranger 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 Ranger disponemos de varias opciones de comunicación inalámbrica: Bluetooth o 2.4GHz y con cualquiera de ellas (no ambas a la vez) podemos programar el robot. Con 2.4GHz, el robot se sincroniza automáticamente con el ordenador sin necesidad de "buscar dispositivo..." evitando de este modo interferencias con otros robots, cuando se utilizan varios a la vez en un aula. También podemos descargar el programa al controlador directamente, por comunicación inalámbrica.

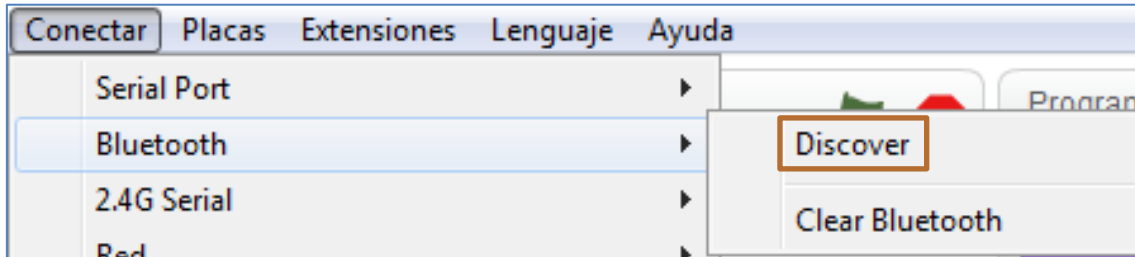


Opciones de la pestaña Conectar

Divirtiéndome con mBot Ranger

Las opciones de la pestaña “Conectar” son:

- *Serial Port*: Conecta el robot 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 Ranger cada vez que queramos usarlo desde el móvil o tablet.



- *2.4G Serial*: Se conecta por 2.4G serial conectando primero el pendriver 2.4G en el PC y sincronizándolo después con el robot mBot Ranger. 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 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.

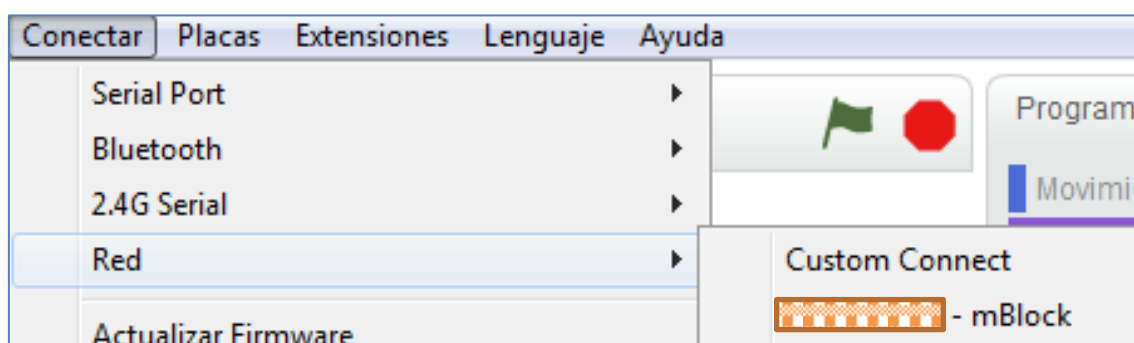
Modo rápido intermitente: Cuando el indicador parpadea 2.4G con velocidad rápida, significa que está en modo rápido de parpadeo. En este modo, el módulo 2.4G se conectará automáticamente al adaptador 2.4G que está encendido. Sin función de memoria en este modo.

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 quieres dejar grabado permanentemente en la placa del robot, deberás usar el cable USB. Esto es así para que el Scratch pueda grabarlo en la placa Auriga que no es más que una placa Arduino Mega.

Divirtiéndome con mBot Ranger

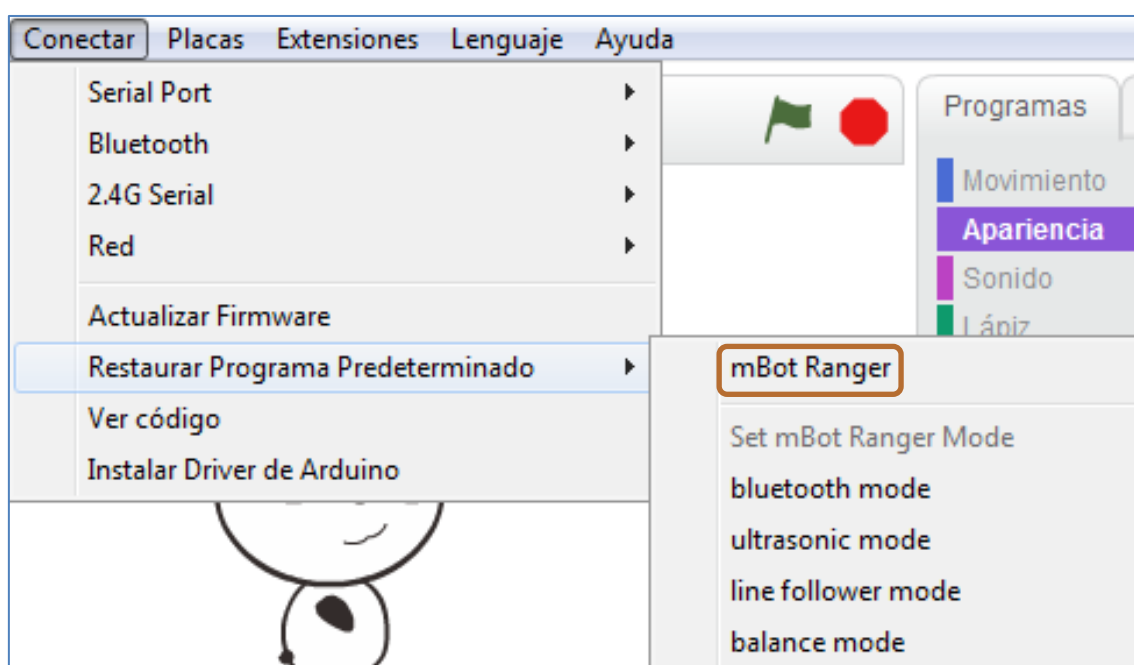
- *Red*: Descarga el programa por conexión directa inalámbrica.



- *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. En este caso, el firmware es un programa que permite la comunicación entre la placa Me Auriga y el ordenador dentro del software.

NOTA: Tras cargar un programa de Arduino a Me Auriga, si queremos conectar la placa de nuevo con el mBlock, debemos instalar otra vez este firmware. Es decir, debemos *Actualizar Firmware*.

- *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.

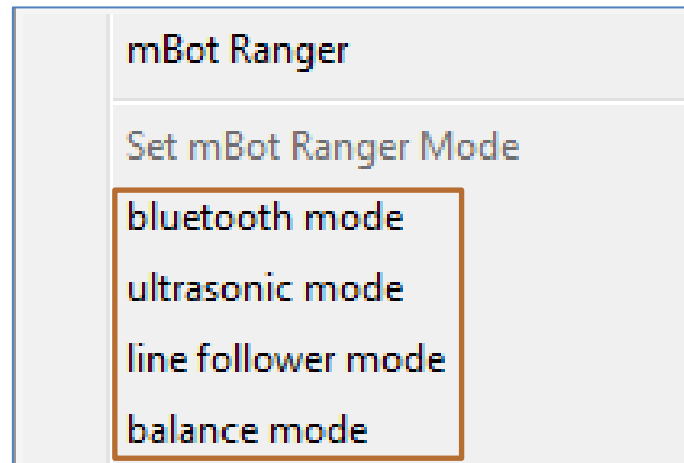
En segundo lugar, encienda el robot y conéctelo al ordenador con el cable USB.

Divirtiéndome con mBot Ranger

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

En cuarto lugar, haga clic en "mBot Ranger", opción de "Restaurar Programa Predeterminado" del menú "Conectar" y espere hasta que tenga éxito 100%.

A veces nos interesa restaurar partes del programa de fábrica (cuando trabajemos con su APP, con el móvil o simplemente cargar un programa de forma autónoma). Estas partes se conocen como "modos": modo bluetooth, modo ultrasonidos, modo seguidor de línea o modo balancín.

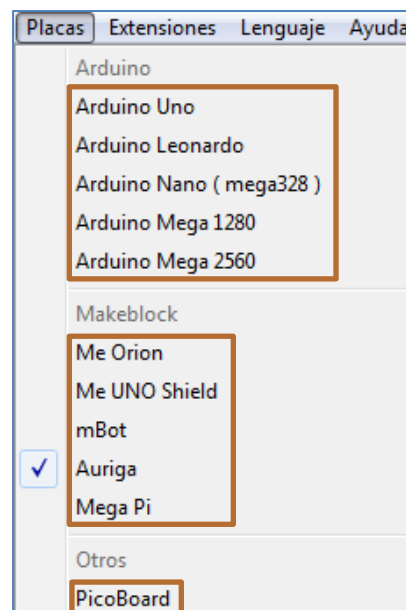


- *Ver código:* Abre una pantalla lateral derecha con el código del programa.
- *Instalar Driver de Arduino:* Instala los drivers de la placa.

2.1.4. Placas

El software mBlock no sólo es aplicable al robot mBot Ranger. Con él podemos programar otras placas electrónicas. Placas que el programa subdivide en tres bloques: Arduino, Makeblock y Otros. Dentro del universo arduino podemos conectar la placa Arduino Uno, Leonardo y Otros. Dentro del universo arduino podemos conectar la placa Arduino Uno, Leonardo y Arduino Mega 1280 y 2560. En el bloque Mackeblock disponemos de la placa Orion, mBot (mCore) y Auriga, así como, el shield para Arduino Uno y la Mega Pi. Finalmente, en el conjunto "Otros" podemos utilizar con el programa la conocida PicoBoard. Todas ellas son opciones que nos ofrece la pestaña "Placas".

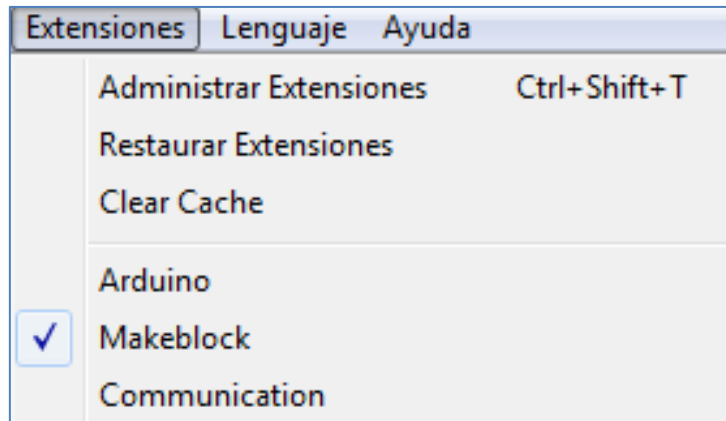
Obviamente, debemos elegir qué opción de las posibles programaremos. Esto se hace desde la pestaña Placas.



Pestaña "Placas" en la Versión 3.3.3 de mBlock

2.1.5. Extensiones

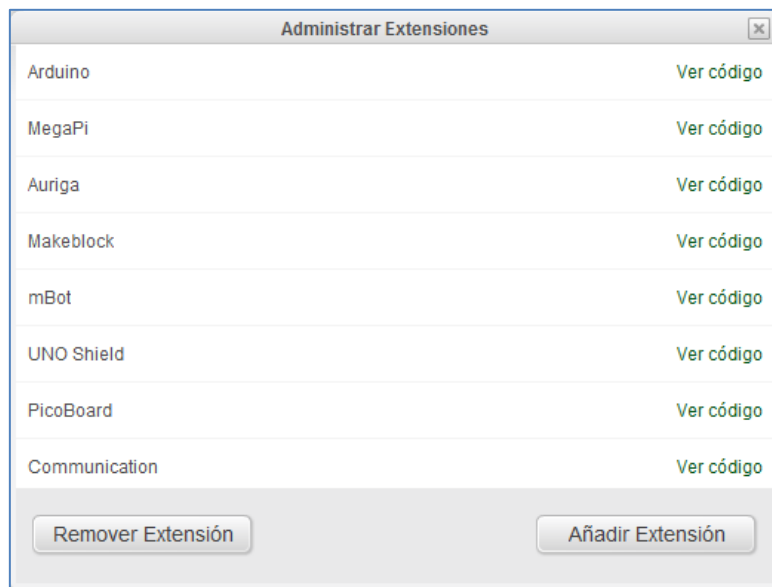
Las opciones de la pestaña “Extensiones” se muestran en la siguiente imagen:



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>

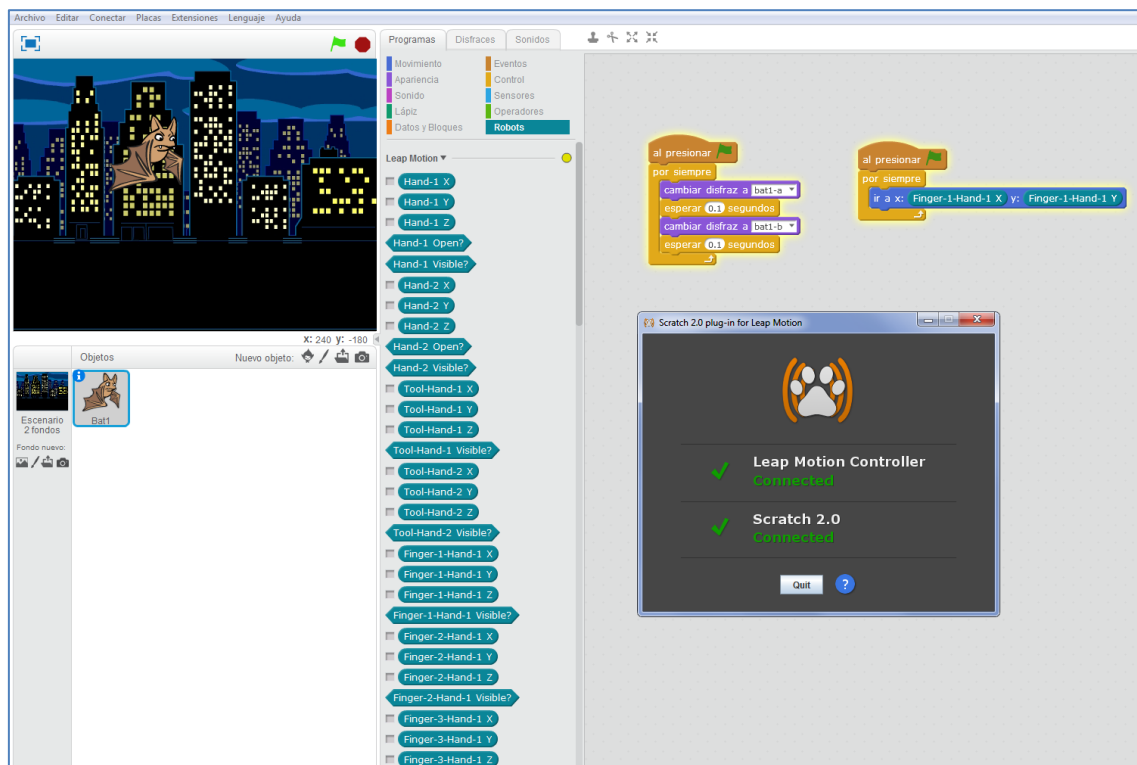
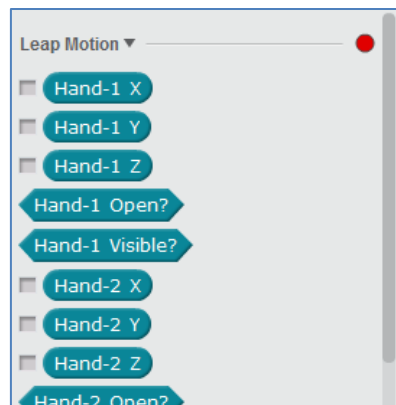
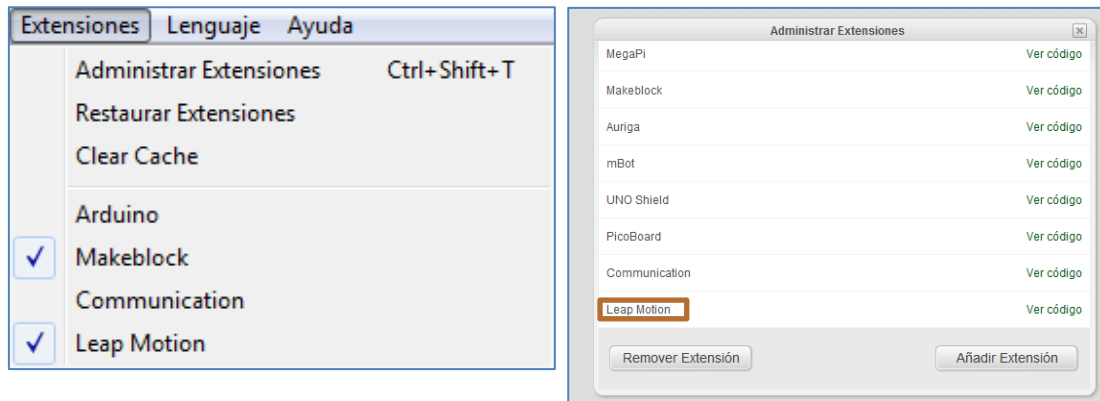


La opción que elija en la extensión afectará a los comandos mostrados en el bloque *Robot* del programa. Por lo tanto, cada opción mostrará sus comandos específicos en el bloque *Robot* del mBlock.

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

Divirtiéndome con mBot Ranger

comandos del Leap Motion y podamos programarlo. En las siguientes imágenes podemos ver que se ha incluido la extensión del Leap Motion:



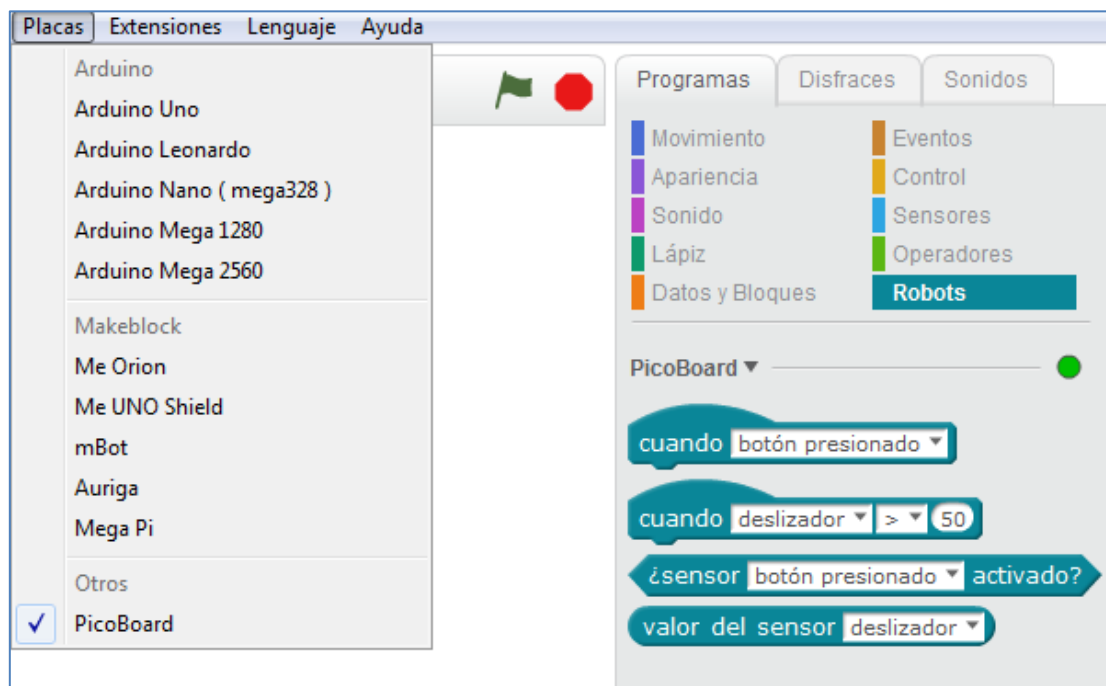
Ejemplo con Leap Motion

Divirtiéndome con mBot Ranger

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

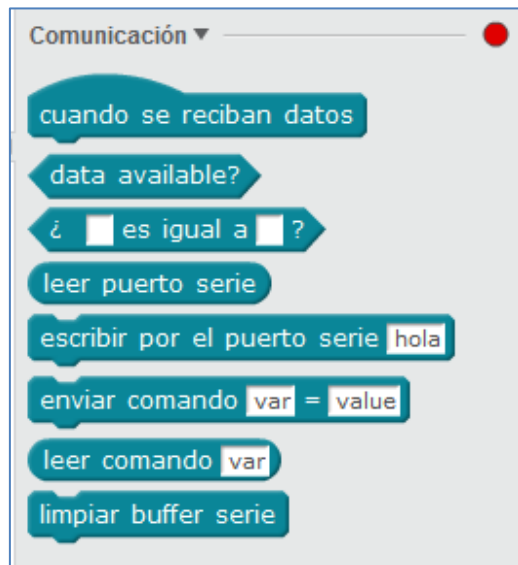


- *Makeblock*: Me Orion, Mega Pi, Auriga, Me Uno Shield y mBot; placas base específicas de Makeblock
- *PicoBoard*: Control y testeo de la placa PicoBoard desde scratch (deslizador, botón, sensor de luz, sensor de sonido, etc)



Divirtiéndome con mBot Ranger

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



- etc

2.1.6. Lenguaje

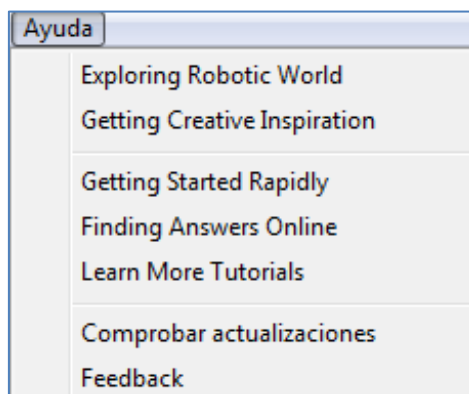
En esta pestaña podemos elegir el idioma y el tamaño de la letra o fuente del mBlock. En la siguiente imagen se ha seleccionado el español y tamaño 12:



Opciones de la pestaña Lenguaje

2.1.7. 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

2.2. Comandos de los bloques del mBlock

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 presenta su propia pestaña *Programas*, *Disfraces* (o *Fondos* si es un escenario) y *Sonidos*.

Para programar un objeto debemos arrastrar los comandos deseados a la zona de *Programas*; comandos que se organizan o clasifican en bloques. En el software mBlock, los bloques son muy similares a los que podemos encontrarnos en el editor Scratch 2.0. Muy similares, pero no idénticos. A continuación se explican sólo las diferencias.

Existen dos diferencias: La primera radica en que mBlock junta los bloques *Datos* y *Más Bloques* del scratch 2.0 en uno sólo que llama *Datos y Bloques* y la segunda se centra en la inclusión de un nuevo bloque que llama *Robots*, con comandos específicos para programar cada diferente placa que se conecte con el software. La siguiente imagen ejemplifica visualmente estas diferencias:



Bloques de Scratch2.0 versus Bloques de mBlock

2.2.1. Datos y Bloques

Nos deja crear variables, listas de datos y bloques dentro del programa mBlock:

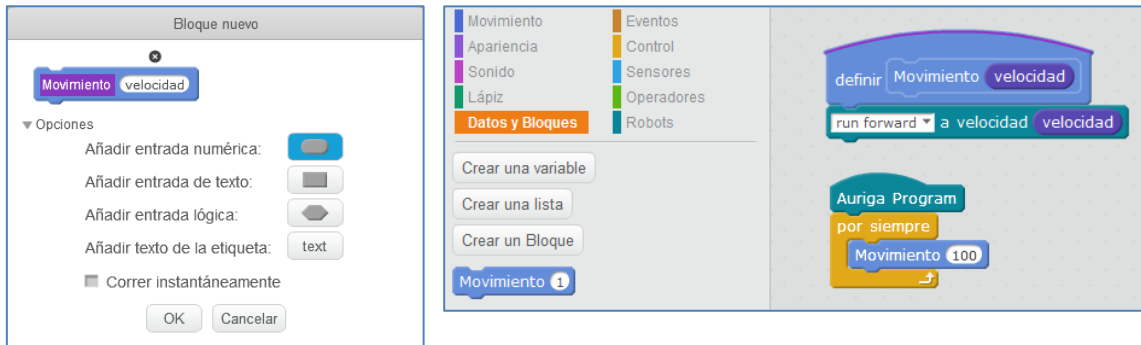


Variables y Listas



Divirtiéndome con mBot Ranger

Bloques: Tras hacer clic en “Crear un Bloque” definimos qué debe realizar, para lanzarlo cuando queramos.



En nuestro ejemplo se ha definido un bloque *Movimiento* que depende de una variable numérica que denomino “velocidad”.

2.2.2. Robots

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

Estos comandos pueden ser específicos o no para la placa en cuestión que estemos programando. La razón es obvia, algunos han sido diseñados concretamente para la potencia de esa placa y por eso no aparecen en el bloque **Robots** de otra placa.

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



Disponemos de una gran diversidad de comandos que clasifico de la siguiente forma:

- Sentencias específicas para el control de motores y servos (Fig 1) en la placa Me Auriga.
- Sentencias para programar los diferentes modelos de LEDs (Fig 2) que soporta la placa.
- Sentencias para mostrar el tiempo y programar la matriz de LEDs (Fig 3)
- Sentencias que nos ayudarán a programar el sensor de sonido (Fig 4)

Divirtiéndome con mBot Ranger

- Comandos específicos para una gran diversidad de módulos que se pueden conectar a la placa Me Auriga o que van incrustados en ella (sensor de temperatura y giróscopo) (Fig 5).

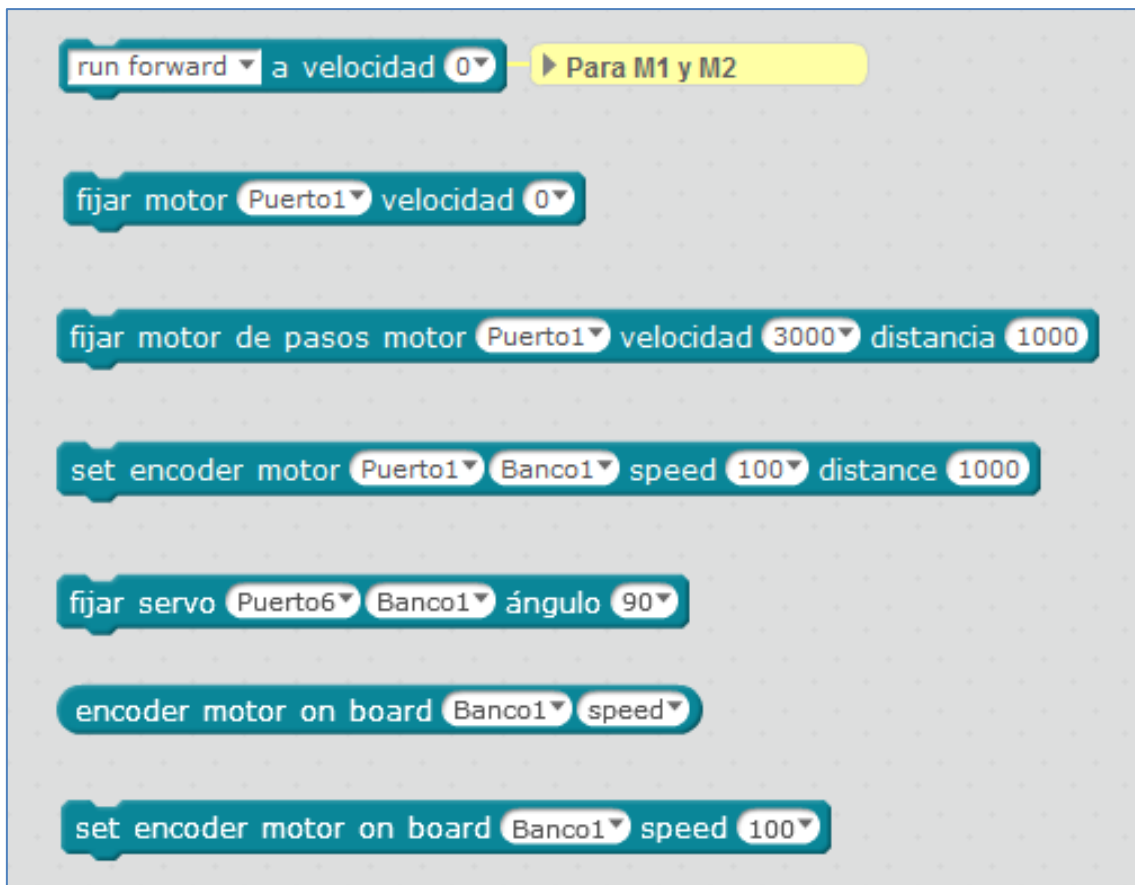


Fig 1: Comandos para motores y servos



Fig 2: Comandos para diferentes modelos de LEDs

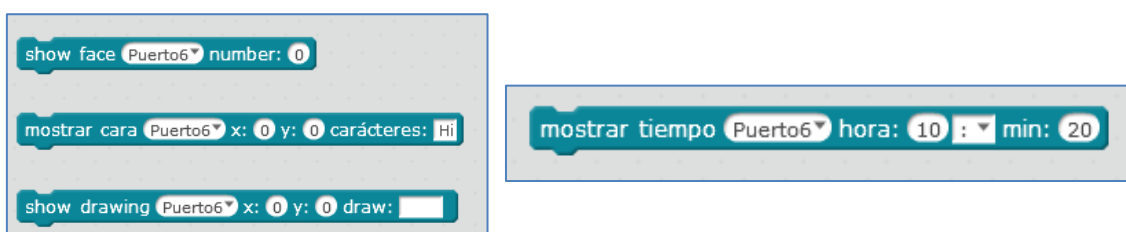


Fig 3: Comandos para la Matriz de LEDs y para mostrar el tiempo

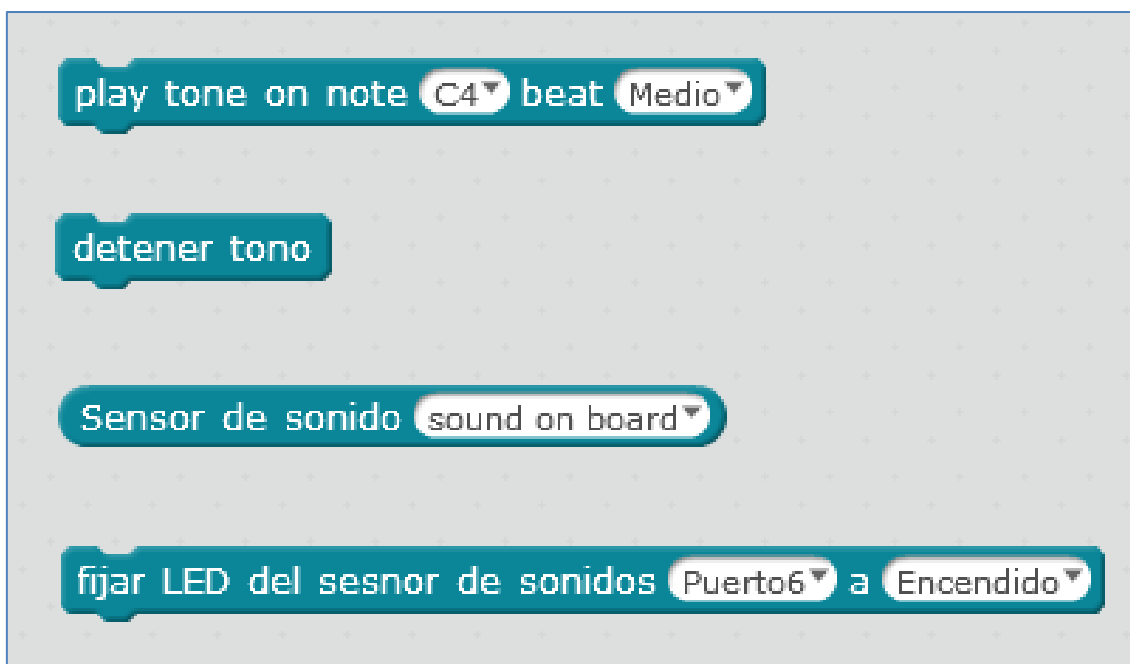


Fig 4: Comandos asociados con el sensor de sonido



Fig 5: Comandos específicos para multitud de sensores

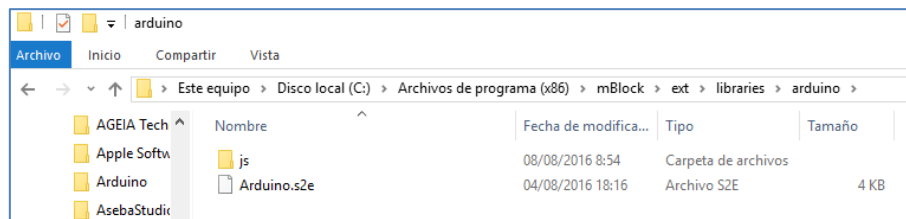
Divirtiéndome con mBot Ranger

2.2.3. Crear nuevas librerías desde mBlock

A veces puede interesarnos 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: Para ello 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": "", "def": "", "work": "analogRead(A0+{0})", "loop": ""}],
```

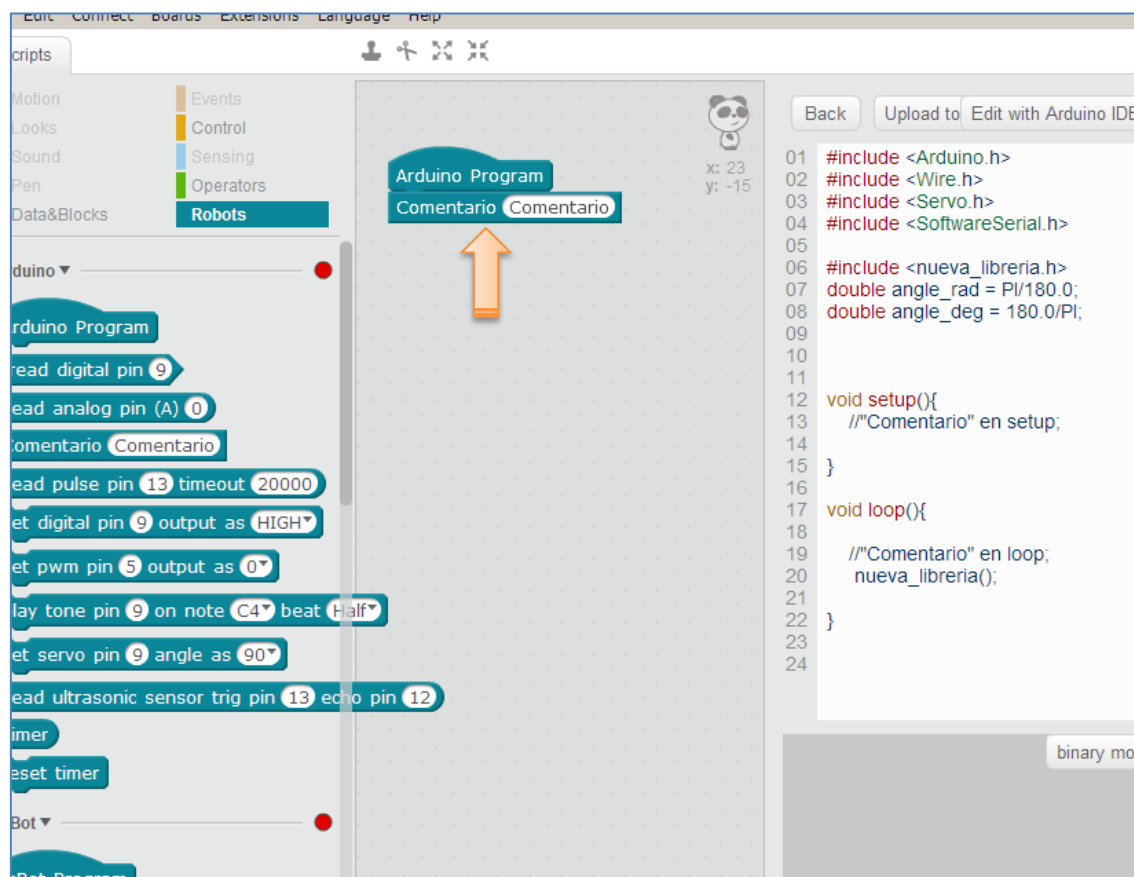
```
[{"w", "Comentario%n", "", "Comentario", {"encode": "{s0}", "setup": "", "inc": "#include <nueva_libreria.h>", "def": "", "work": "//{0} en setup;\n", "loop": "//{0} en loop;\n nueva_libreria();"}],  
...
```


Divirtiéndome con mBot Ranger

Si nos fijamos en sus librerías, incluso podemos ver cómo están contruidos los demás bloques, tanto de Arduino como de otras placas (Auriga, mbot, mega_pi, etc), y para qué sirve cada parámetro.

Este equipo > Disco local (C:) > Archivos de programa (x86) > mBlock > ext > libraries >				
Nombre	Fecha de modifica...	Tipo	Tamaño	
arduino	08/08/2016 8:54	Carpeta de archivos		
Auriga	08/08/2016 8:54	Carpeta de archivos		
mbot	08/08/2016 8:54	Carpeta de archivos		
mega_pi	08/08/2016 8:54	Carpeta de archivos		
orion	08/08/2016 8:54	Carpeta de archivos		
picoboard	08/08/2016 8:54	Carpeta de archivos		
serial	08/08/2016 8:54	Carpeta de archivos		
uno_shield	08/08/2016 8:54	Carpeta de archivos		

En nuestro ejemplo, el resultado es el siguiente:



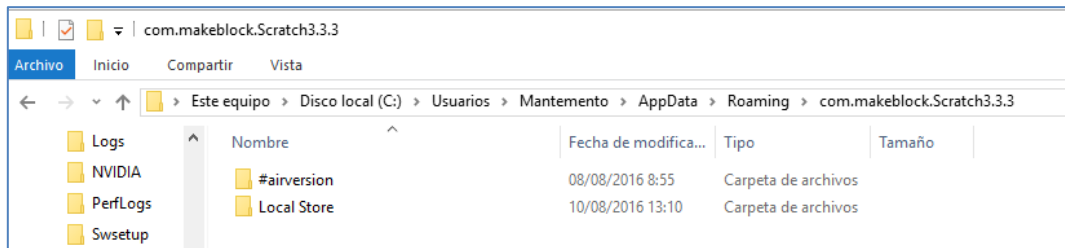
A la hora de probarlo, nos puede ocurrir alguna de las siguientes incidencias:

- Lo he cambiado y en el Scratch (mBlock) no aparece el bloque

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

Divirtiéndome con mBot Ranger

C:\Users\[Tu Usuario]\AppData\Roaming\com.makeblock.Scratch3.3.3



Como solución:

- 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, etc.
Activa la extensión de Arduino desde el menú "extensiones"

3. Modos de trabajo con el Kit mBot Ranger en mBlock

Cada docente diseñará diferentes formas de trabajo en sus aulas con el kit mBot Ranger bajo mBlock, buscando que sus alumnos/as desarrollen las competencias y habilidades que los lleven a resolver problemas de su mundo real. Para mí, en todas ellas, el profesor debe asumir el rol de guía y el niño aprender haciendo, con pruebas y errores, aprendiendo a su ritmo de ellos.

La robótica, como bien sabemos, presenta un carácter interdisciplinar. En consecuencia, debe usarse para abordar proyectos y retos que estén presentes en las materias de inglés, tecnología, matemáticas y demás. Con ella se puede aprender mecánica, física, matemáticas, informática, arte, y un largo etcétera donde incluyo, obviamente, la tecnología. En la programación y concretamente en la robótica existen varias líneas pedagógicas que la defienden y argumentan con evidencias científicas. Entre sus muchas virtudes cito algunas: desarrollo de la creatividad, asimilación significativa de contenidos, aprendizaje tecnológico, trabajo en equipo, aumento de autoestima y confianza en sí mismos, potenciadora de emociones, trabajo del pensamiento analítico, toma de decisiones para dar solución a una necesidad, etc.

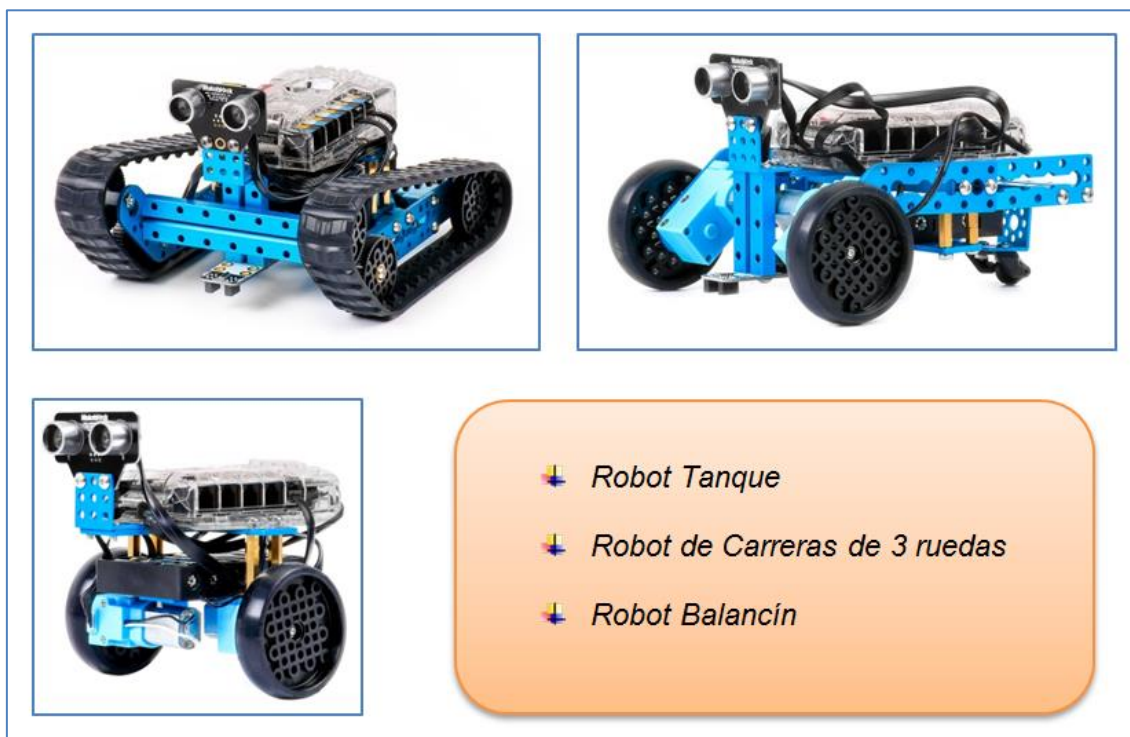
No quiero a hablar de metodologías. En este apartado sólo se mencionarán aplicaciones o usos genéricos. Particularmente, para el Kit mBot Ranger, a mí se me ocurren tres y serán las que brevemente describo:

3.1. Robot

Sin lugar a dudas, la mejor forma de experimentar y aprender robótica es a través de un robot. El kit mBot Ranger incluye las estructuras y componentes necesarios para

Divirtiéndome con mBot Ranger

montar tres diferentes modelos con los que el alumno/a podrá interactuar con el mundo físico:



Como la placa Me Auriga deja libres bastantes conectores RJ25 y su potencia no es baja (pensar que es una Arduino Mega), a ella se le pueden añadir más motores, sensores y actuadores de modo que abarquen los deseos de las diferentes creaciones de nuestro alumnado, hacia mayores funcionalidades en cada uno de los tres modelos que inicialmente pueden montarse.

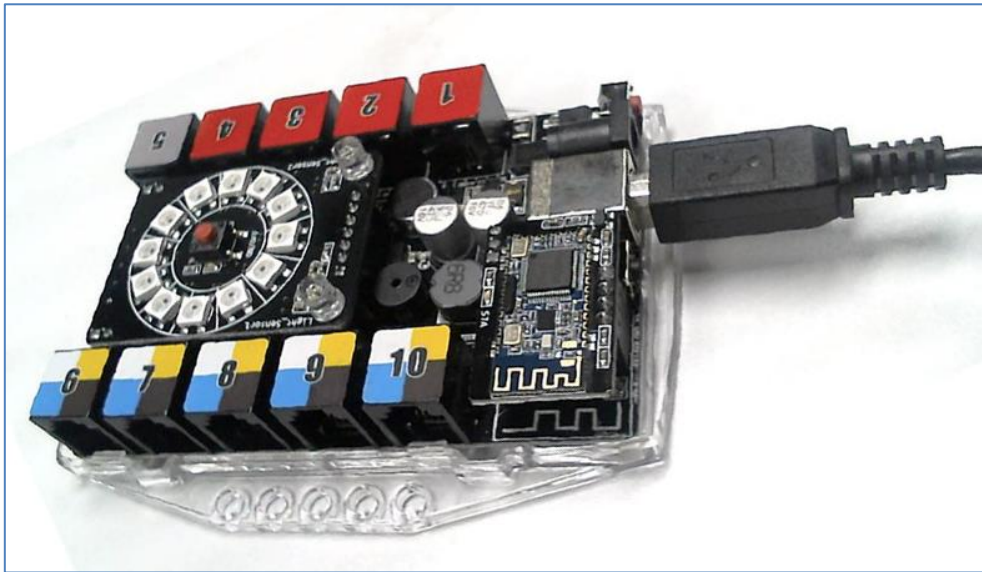
Programando en Scratch bajo la herramienta mBlock (o a través del IDE de Arduino) podemos sensorizar el entorno físico. Es decir, conseguir leer datos de sensores como el de luz, sonido, temperatura o humedad, entre otros, y utilizarlos para que los objetos que hemos diseñado o creado en un programa que se visualiza gráficamente como scratch, respondan a diferentes estímulos del mundo físico real que ejecutará el robot. Esta funcionalidad se plasmará en lograr captar, a tiempo real, datos del entorno, aprender conceptos de forma más dinámica experimentando físicamente con los elementos y conseguir que el alumno/a interactúe de forma directa, aprendiendo haciendo. El alumno/a, al sentirse atraído por la novedad y potencial de estos dispositivos, se sentirá motivado y deseará experimentar, crear y aprender.

3.2. Controladora de juegos y creaciones

Lo que más me gusta del scratch, este entorno de programación libre, es que los usuarios pueden desarrollar juegos y simulaciones, expresar emociones y sentimientos, así como, reflejar pensamientos e ideas de forma interactiva y creativa. Todo esto se puede hacer con mBlock. A mayores, su potencial se amplifica gracias a la conexión y programación de diferentes placas con el programa. La utilidad de la

Divirtiéndome con mBot Ranger

placa del robot, Me Auriga, como una tarjeta controladora de sus juegos y creaciones, es una herramienta muy seductora en las aulas.



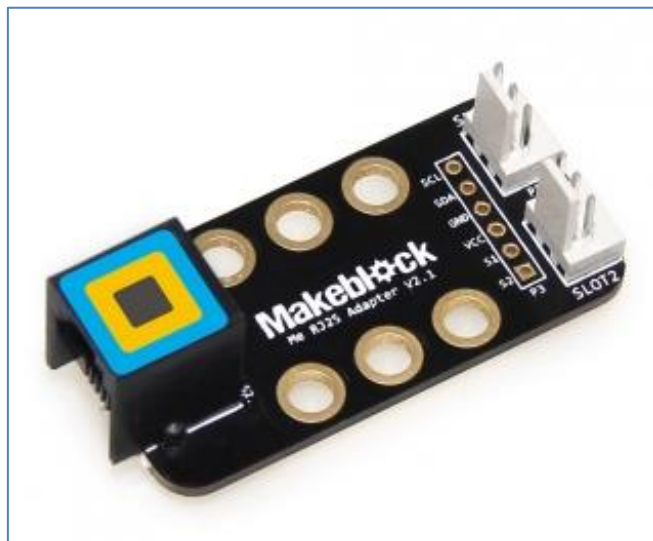
3.3. Placa de pruebas y laboratorio de electrónica

La placa Me Auriga es una placa Arduino Mega2560 que lleva incrustados unos determinados sensores y que presenta conectores RJ25. Este corazón del robot incluye componentes, módulos y puertos RJ25 suficientes para facilitar el desarrollo de diferentes prototipos.

Si queremos darle uso como placa de pruebas o laboratorio de electrónica necesitaremos utilizar un amplio espectro de sensores y actuadores. Estos módulos que se le acoplen, cuando lo requiera el proyecto o simplemente para comprender cómo trabaja un determinado componente, no tienen por qué ser los de la propia casa (que ya disponen de conectores RJ25 y vienen preparados para “conectar y listo”). La casa Makeblock da opción a utilizar cualquier sensor o actuador del mercado, proporcionándonos soluciones para su conexión: los adaptadores RJ25 y los conectores RJ25 a *Dupont*. Estos últimos son conectores que en un extremo presentan 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.



Conector RJ25 a Dupont



Adaptador RJ25

En fin, que si tenemos una necesidad de conectar algo: ¡adelante!, sólo debemos conectarlo y codificarlo. Es una manera simple y ordenada para empezar con la programación de una idea y creación de prototipos de hardware bajo el lenguaje de programación Scratch (con mBlock) y, como no, bajo Arduino (IDE de Arduino).

4. Ejemplos de programación con los sensores y componentes de Me Auriga

Tal y como se ha descrito en la introducción de este documento, la placa Me Auriga incorpora unos sensores y componentes, elementos que ejemplificaremos con mBlock en este apartado.

4.1. Anillo de LEDs RGB

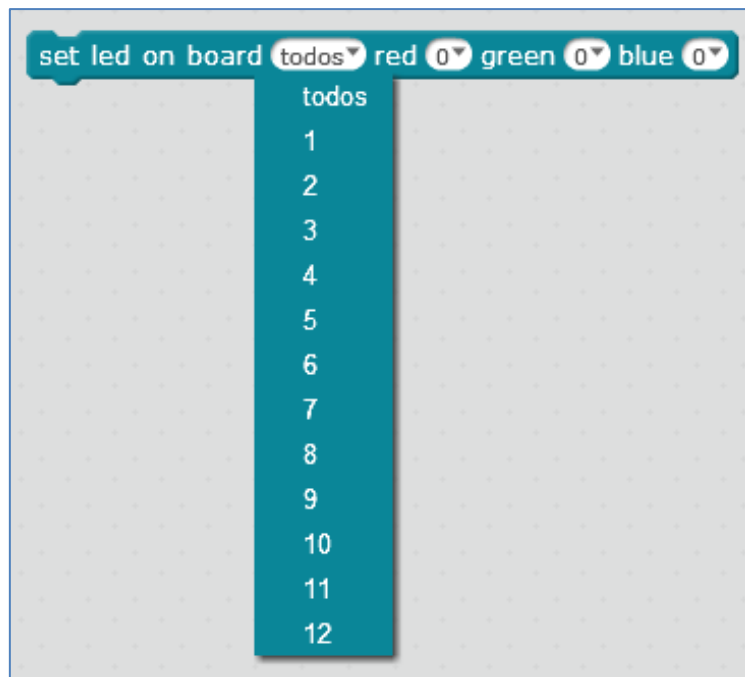
El robot mBot Ranger dispone de un módulo en forma de anillo de 12 LEDs RGB que viene acoplado, no incrustado, a la placa Me Auriga:



Anillo de LEDs RB

Divirtiéndome con mBot Ranger

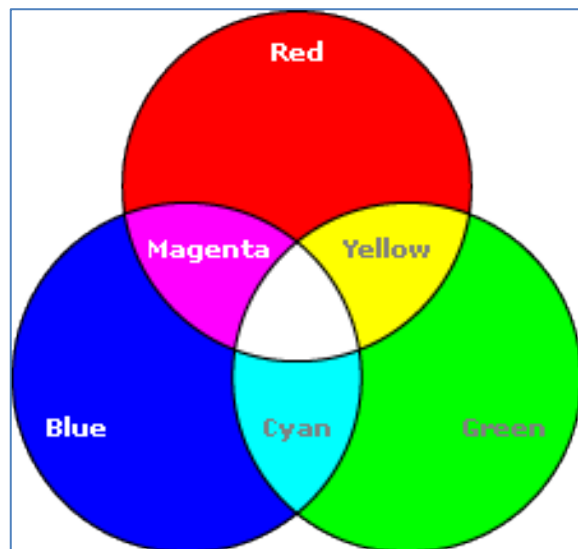
El comando que se utiliza para programar cada uno de los LEDs de forma independiente, o todos a la vez, en sus diferentes colores o variaciones, es el siguiente:



Comando para programar el anillo de 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.

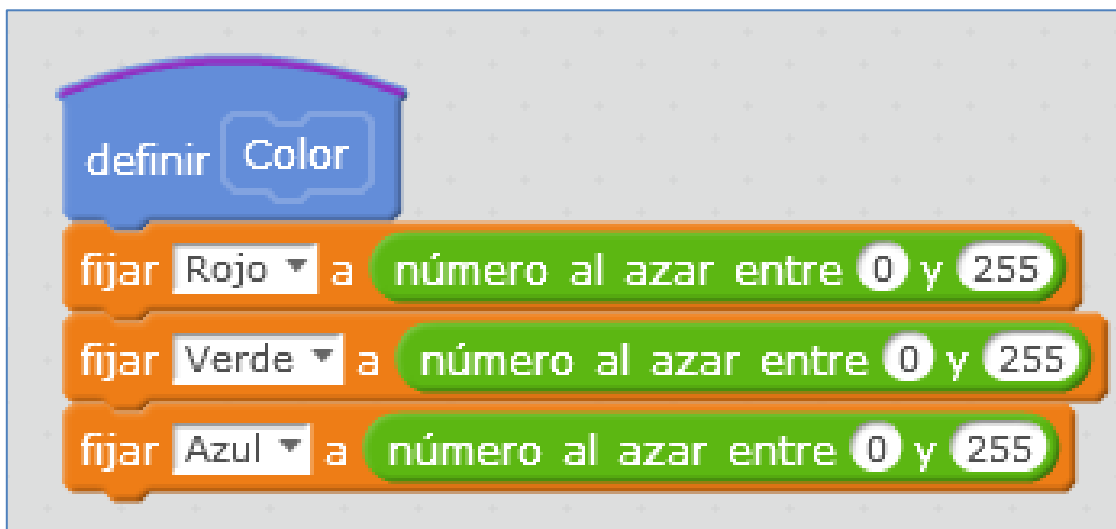
Como se puede ver en la imagen derecha, el amarillo lo conseguiríamos combinando el rojo con el verde (poniendo, por ejemplo, el rojo a 125, el verde a 125 y el azul a 0).



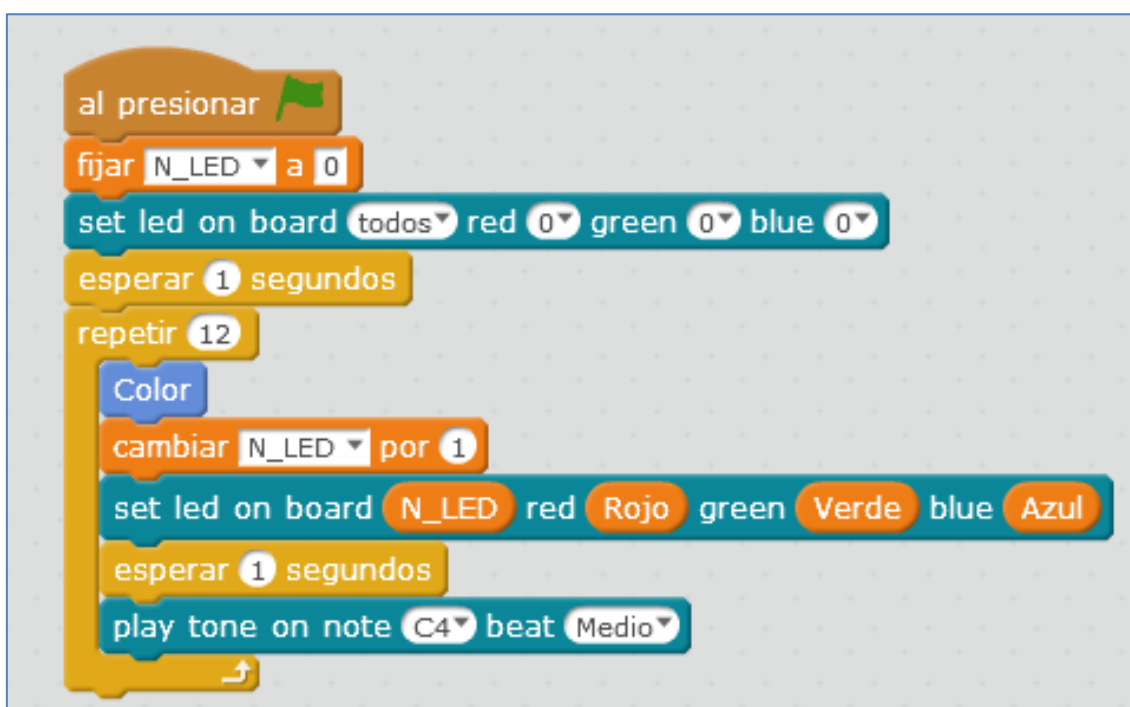
Como ejemplo vamos a diseñar un programa que envíe a cada LED RGB del anillo una combinación de sus tres colores básicos calculada al azar entre todas sus posibilidades, y que esta combinación se visualice en cada LED. Para ello necesitamos crear 4 variables: *N_LED* (que denota cada LED del anillo, de 1 a 12), *Rojo*, *Verde* y *Azul*. Las variables denotadas por los nombres de los colores básicos asumirán un número aleatorio entre el valor numérico 0 y el 255, tal y como se observa

Divirtiéndome con mBot Ranger

en el siguiente bloque que he definido por el nombre “Color” y que debe ejecutarse en cada LED RGB:



Inicialmente pongo todos los LEDs sin color y a continuación creo un simple contador para la variable *N_LED* que se repetirá 12 veces (recorriendo cada uno de los doce LEDs de forma ordenada). En cada repetición se llama al bloque “Color” consiguiendo que fije unos valores nuevos aleatorios para los colores Rojo, Verde y Azul y que estos sean los que se le apliquen al LED en cuestión:



A mayores, tras cada visualización de color al azar, sonará la nota musical Do (C4).

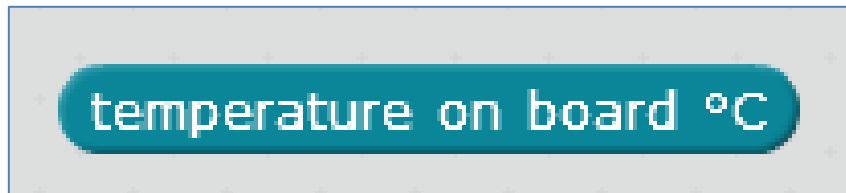
4.2. Sensor de temperatura

Si extraemos el módulo Anillo de LEDs RGB de la placa Me Auriga, nos encontramos con un sensor de temperatura analógico que nos ofrece la medida de la temperatura

Divirtiéndome con mBot Ranger

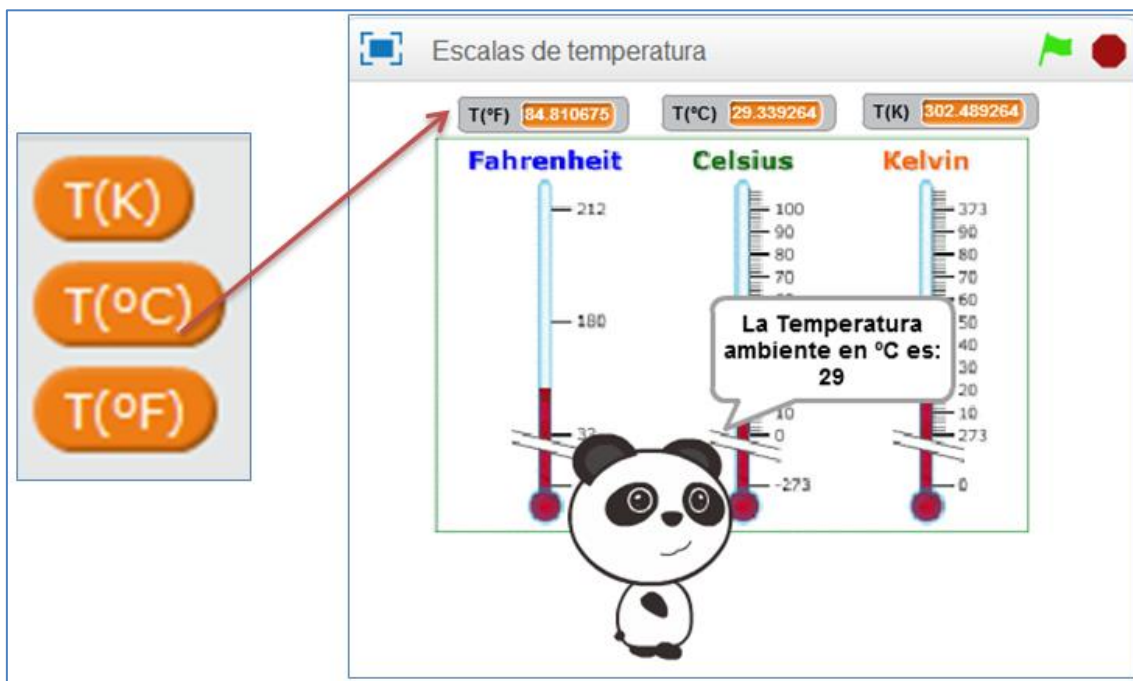
ambiente en un rango aproximado de -40°C a $+125^{\circ}\text{C}$. Este sensor es un termistor NTC. Todo termistor o sensor de temperatura resistivo basa su funcionamiento en la variación que sufre la resistencia del material semiconductor cuando se produce un cambio en la temperatura. Al ser del tipo NTC (*Negative Temperature Coefficient*), esta variación afecta de la siguiente forma: la resistencia del semiconductor disminuye al aumentar la temperatura ambiente; de ahí que el coeficiente sea negativo.

Programar el sensor de temperatura en mBlock es muy sencillo. Debemos hacer uso del siguiente comando que nos otorgará la medida de la temperatura ambiente en $^{\circ}\text{C}$:



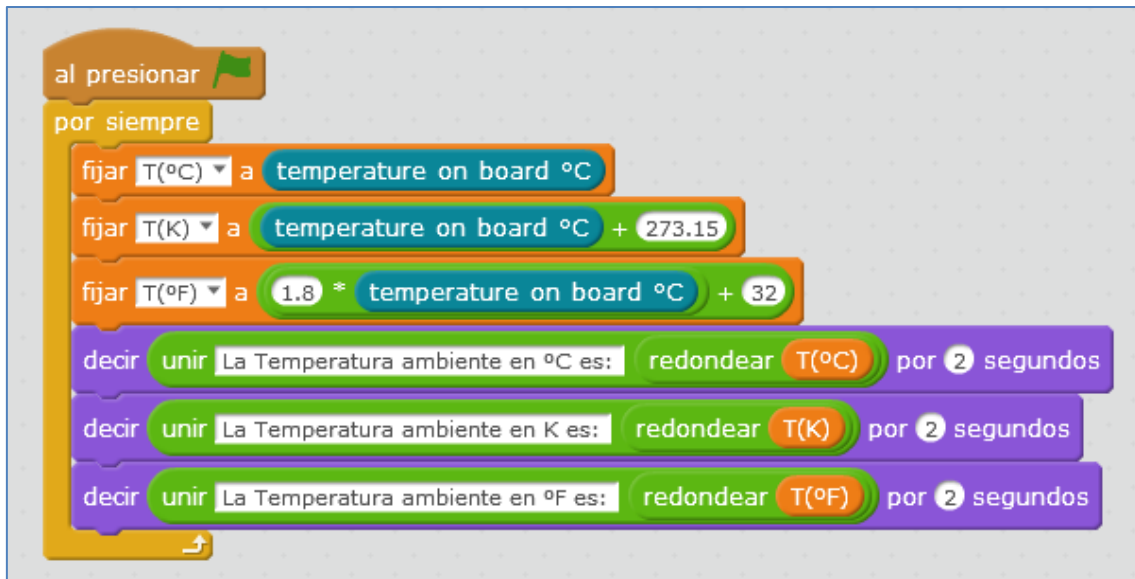
En el siguiente ejemplo vamos a mostrar la temperatura ambiente en tres escalas de temperatura diferentes. A saber: Fahrenheit, Celsius y Kelvin.

Para ello, creo tres variables que se asocian a cada una de las escalas y cuyo cálculo ($^{\circ}\text{F}$ y K) o medición ($^{\circ}\text{C}$), será visible, sin aproximación o redondeo, en el escenario del programa:



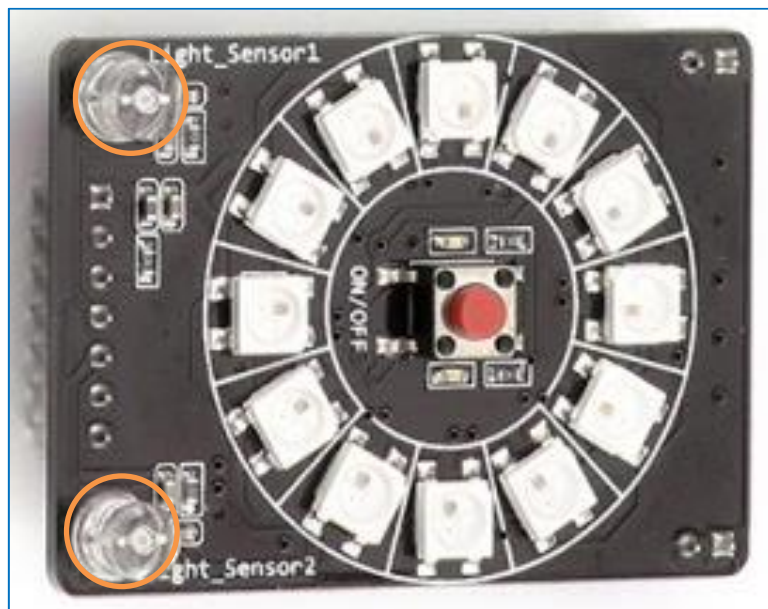
Escenario con escalas de Temperatura y variables

El objeto *M-Panda* nos dirá, cada 2 segundos, el valor numérico de la temperatura (valor aproximado y para ello uso la función redondeo) en las tres escalas. Es decir, cada 6 segundos mostrará las variaciones que ha sufrido la magnitud en cuestión en su respectiva escala. El script que nos ofrece la temperatura ambiente en $^{\circ}\text{C}$, $^{\circ}\text{F}$ y K , con y sin redondeo, es el siguiente:



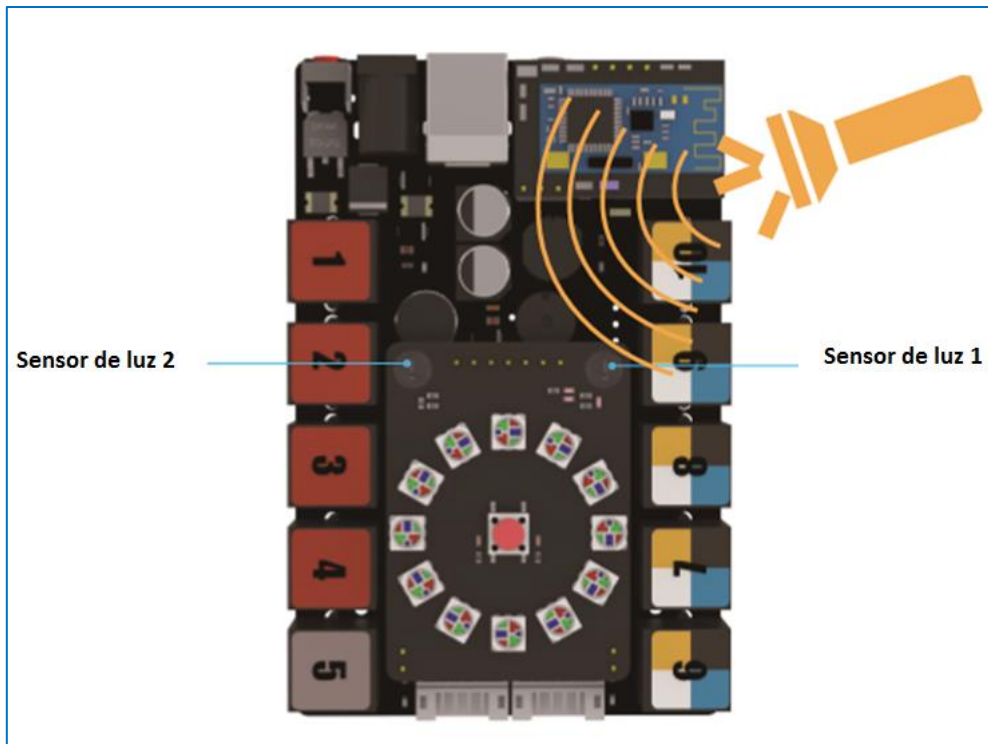
4.3. Sensores de luminosidad

Ya sabemos que a la placa del robot Ranger se le puede acoplar un módulo anillo de LEDs RGB (Ver 4.1). Este módulo no viene soldado a la placa y se incluye en el kit del robot. Como se observa en la siguiente imagen, a mayores, incorpora dos sensores de luz muy bien situados, a ambas esquinas, ideales para llevar a cabo un robot seguidor de luz.



Módulo Anillo de LEDs

En la figura anterior se observa que ambos sensores están serigrafiados en el anillo de LEDs. De esta forma podemos conocer cuál de ellos se reconoce en el programa mBlock como 1 y cuál como 2 (*on board 1* y *on board 2*).



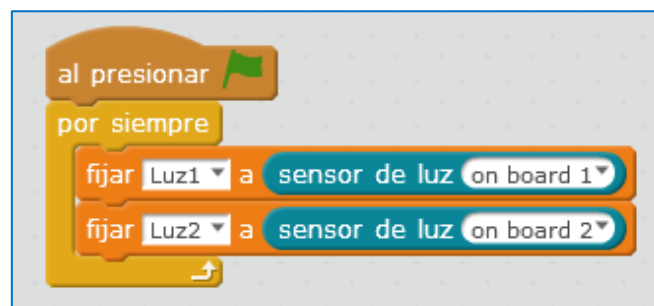
Sensores de luz en la placa Me Auriga

El comando que se utiliza para programar un sensor de luz, esté o no incrustado en el anillo de LEDs, es el mismo (ver imagen derecha). En el desplegable del comando podemos escoger si queremos programar los sensores de la placa (*on board 1* y *on board 2*) o si queremos programar el módulo sensor de la casa Makeblock (apartado 5.25), estando éste conectado a un determinado puerto de la placa Me Auriga (del 6 al 10).



Como reto, vamos a diseñar *un robot seguidor de luz* utilizando los dos sensores de luz que vienen incrustados en el anillo de LEDs RGB de la placa.

Antes de nada debemos testear la medida de los sensores en la luz ambiente de la habitación y cuando acercamos una fuente de luz, como puede ser una linterna. Para ello uso las variables Luz1 y Luz2:





Con iluminación ambiente



En oscuridad



Con una linterna de frente

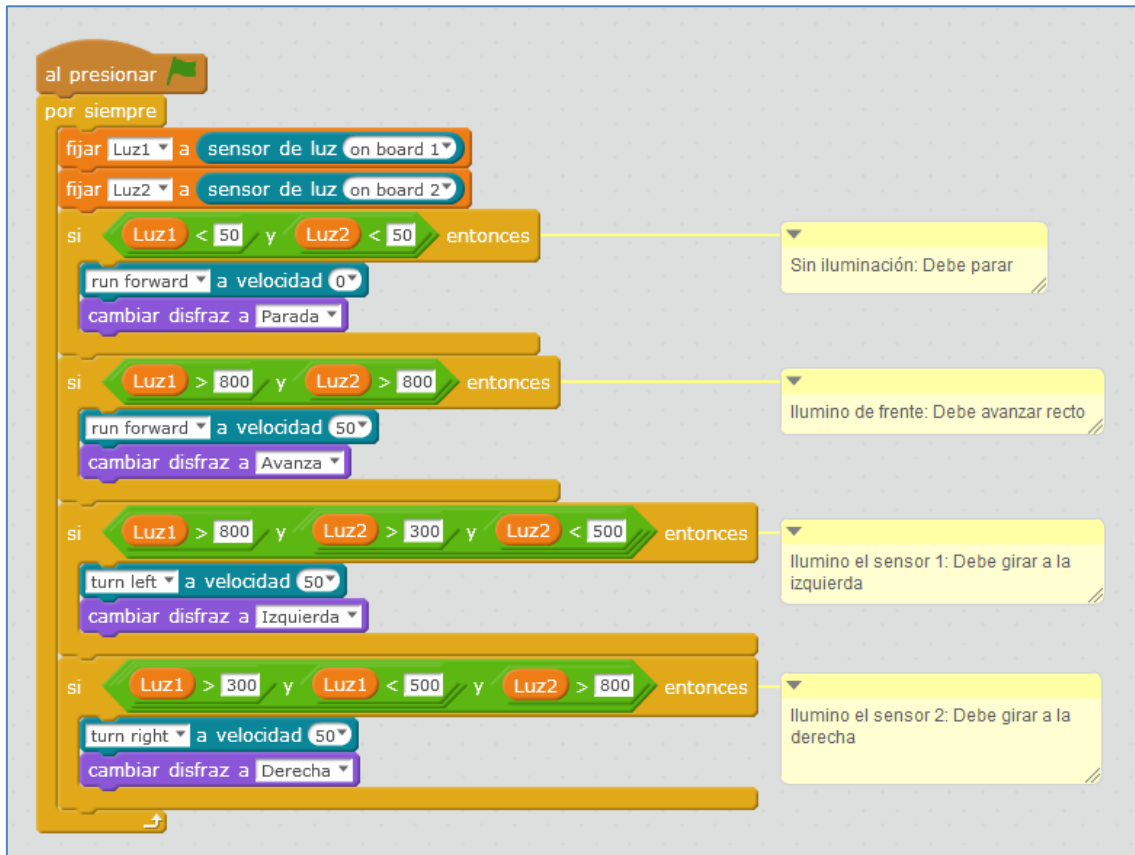
Divirtiéndome con mBot Ranger

Tras un testeo inicial, las acciones que debo programar se resumen en la siguiente tabla:

Imagen	Sensor de Luz 1	Sensor de Luz 2	Acción
	20	20	Sin iluminación: Debe parar.
	800	900	Iluminación frontal: Debe avanzar recto
	900	300	Ilumino el sensor 1: Debe girar a la izquierda
	300	900	Ilumino el sensor 2: Debe girar a la derecha

Divirtiéndome con mBot Ranger

Así pues, el programa seguidor de luz, que incorpora una simulación del mismo en el escenario del mBlock, sería el siguiente:



Obviamente, el programa deberíamos ajustarlo hacia la fuente de luz que nosotros vayamos a usar. También, lo correcto es cargarlo en la placa y así evitar el cable de conexión del robot con el PC (aunque, por supuesto, también evitaríamos el cable usando la mochila 2.4G en lugar del módulo bluetooth que trae incorporado el kit del robot mBot Ranger).

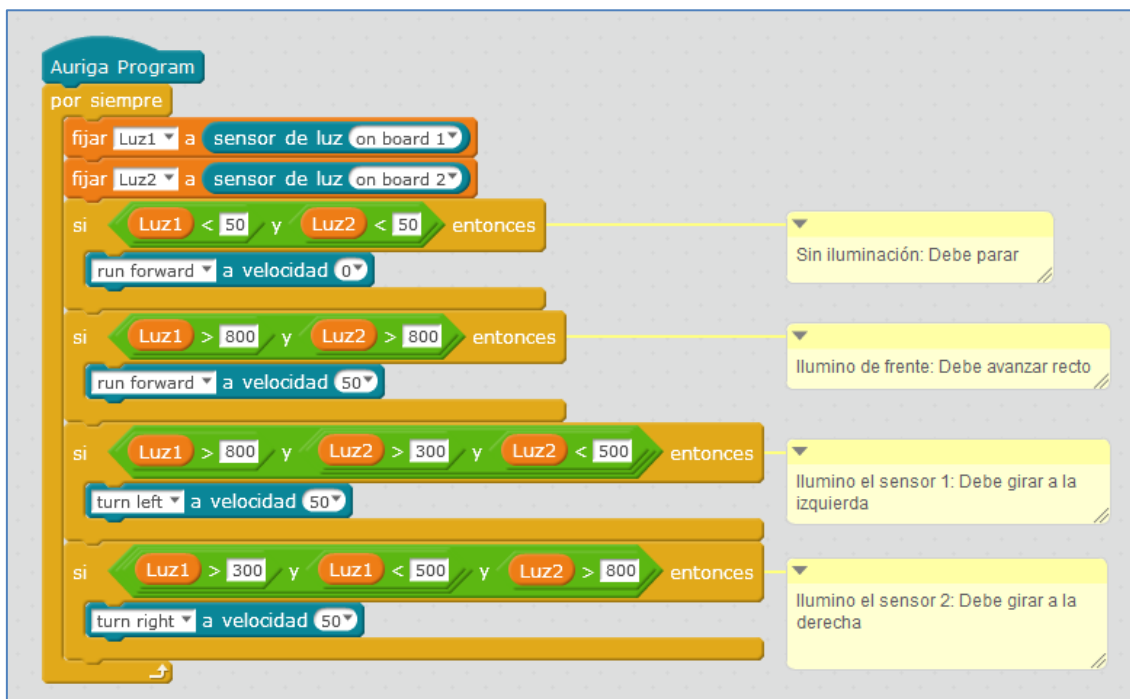


Mochila 2.4G

Para cargar el programa a la placa debemos cambiar el comando de inicio de la bandera verde por el comando **Auriga Program**, así como, eliminar los comandos lila relativos a las llamadas de disfraces del objeto programado.

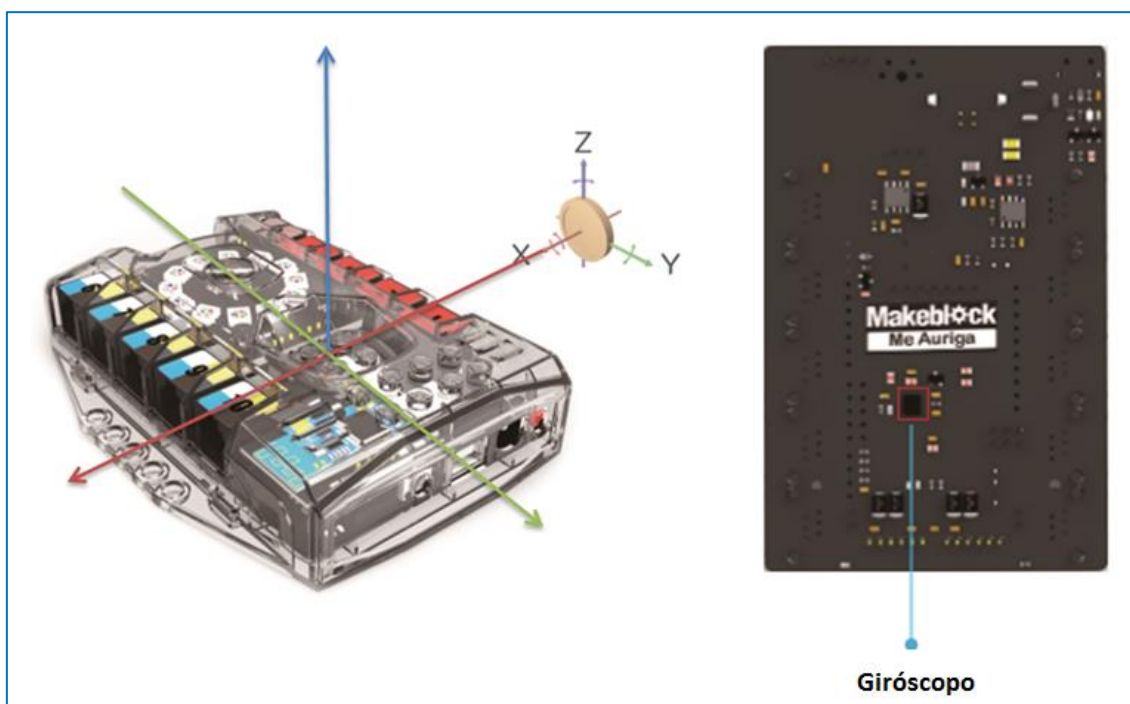
Divirtiéndome con mBot Ranger

El programa final, preparado para cargar en la placa, quedaría como se ve en la siguiente imagen:



4.4. Sensor giróscopo

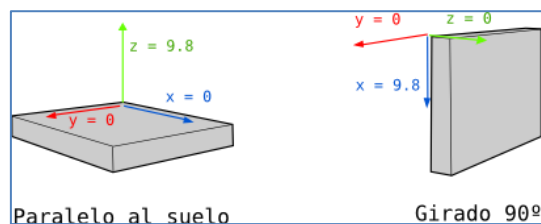
La propia placa Me Auriga lleva incluida (soldada) un sensor giróscopo basado en el chip MPU-6050 y que está formado por un acelerómetro de tres ejes (uno en cada eje), un medidor de la velocidad angular (también de tres ejes) y un procesador digital de movimiento. Este sensor se sitúa en la parte trasera de la placa Auriga:



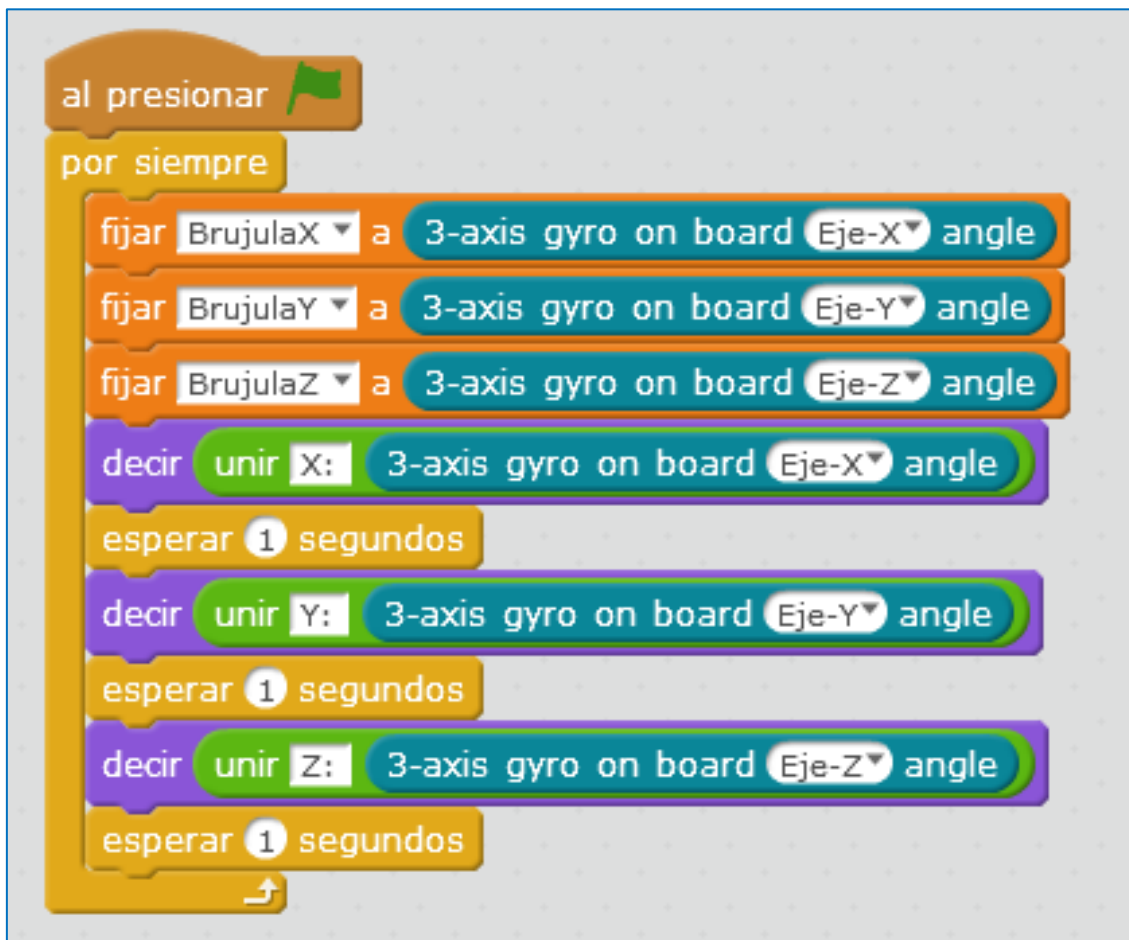
Divirtiéndome con mBot Ranger

La Unidad de Procesamiento de Movimiento MPU-6050 posee 6 grados de libertad, también llamados DOF (*Degrees Of Freedom*) y esto significa que nos aporta 6 valores de salida. Los DOF provienen de los 3 ejes de un acelerómetro y de 3 ejes de un giróscopo. Para interconectarse con la placa utiliza el protocolo de comunicación de datos I²C. El acelerómetro mide aceleraciones en las tres dimensiones del espacio y el giróscopo mide la velocidad angular, que no es más que el número de grados que gira en cada segundo.

El chip MPU detecta la aceleración de la gravedad terrestre (aceleración perpendicular al suelo). Si se encuentra perfectamente alineado con el suelo, entonces, el eje Z marcaría 9,8, y los otros dos ejes (X e Y) nos aportarían el valor nulo. Pero, si giramos el módulo sensor 90° (ver imagen siguiente), resulta que el eje perpendicular al suelo es el eje X, y es el que marcaría 9,8. Y, en este caso, los ejes Y y Z nos aportarían el valor nulo.



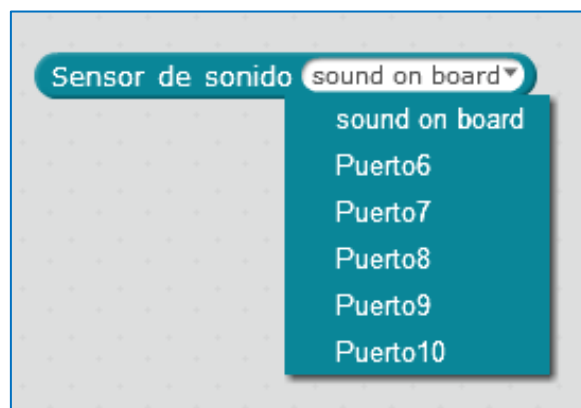
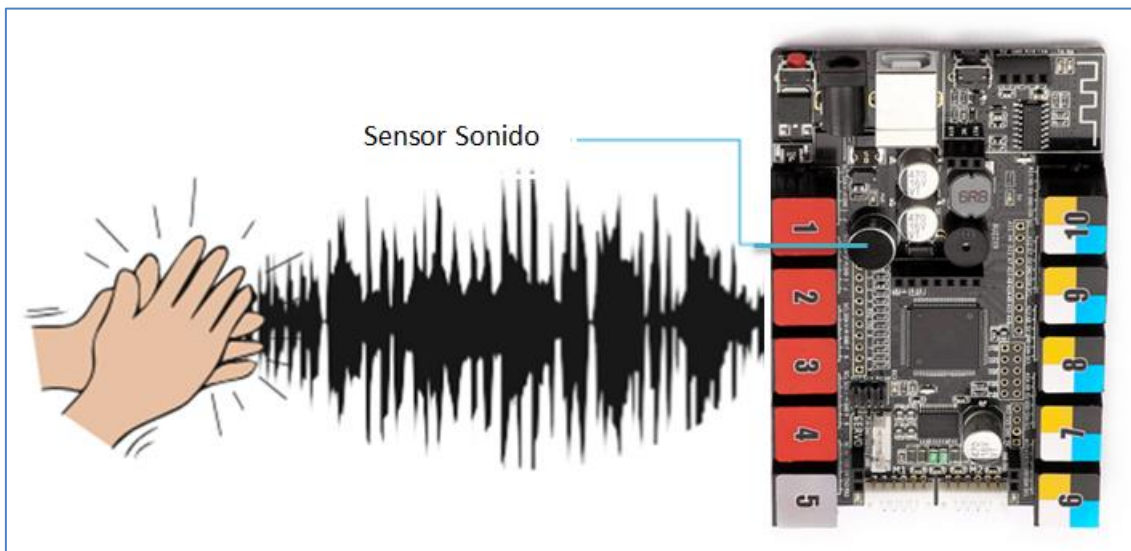
Con este script el objeto *M-Panda* nos mostrará el valor que le proporciona el acelerómetro de la placa Auriga en los ejes X, Y y Z:



Divirtiéndome con mBot Ranger

4.5. Sensor de sonido

Este sensor ya va incluido (soldado) en la placa Auriga del robot Ranger. El comando para su control bajo mBlock es el mismo que usaríamos si quisiéramos programar el módulo de sonido (Ver 5.16):



Comando para el sensor sonido

Conseguir que nos muestre el valor sonido del ambiente en el escenario, en todo momento, es muy sencillo. Sólo debemos ejecutar en siguiente script:

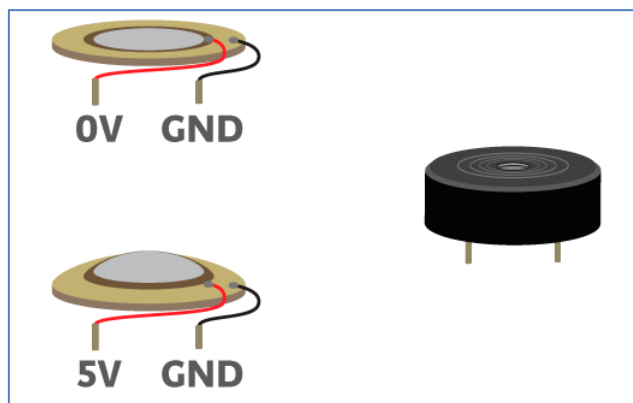


Ejemplo de control del sensor sonido en la placa Me Auriga

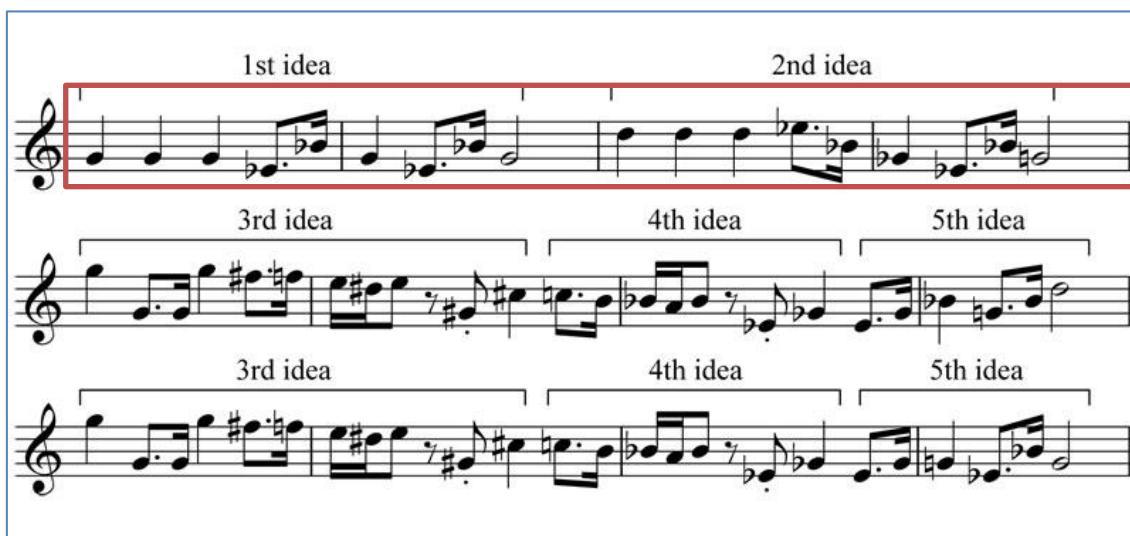
Divirtiéndome con mBot Ranger

4.6. Zumbador

Un zumbador o buzzer es un componente electrónico que es capaz de transformar energía eléctrica en acústica. Es decir, en sonido. Nuestro buzzer se compone de dos discos (uno metálico y otro cerámico) que tienen propiedades piezoeléctricas. Cuando se le aplica una tensión (V) al elemento, los materiales se repelen y como resultado se obtiene una onda sonora. Y una vez que se le deja de aplicar ese voltaje, se revierte este efecto piezoeléctrico y los materiales retoman su posición inicial, emitiendo de nuevo el sonido.

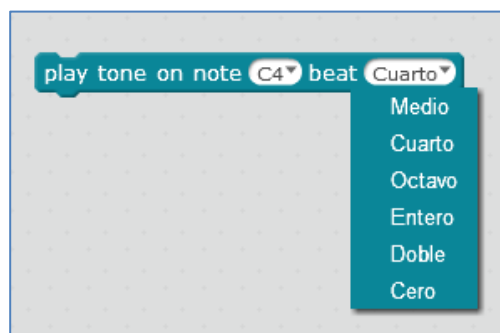


Como reto podemos hacer que suene parte de una canción. Por ejemplo, los cuatro primeros compases de la canción “*Imperial March*” de la película *Star Wars*.

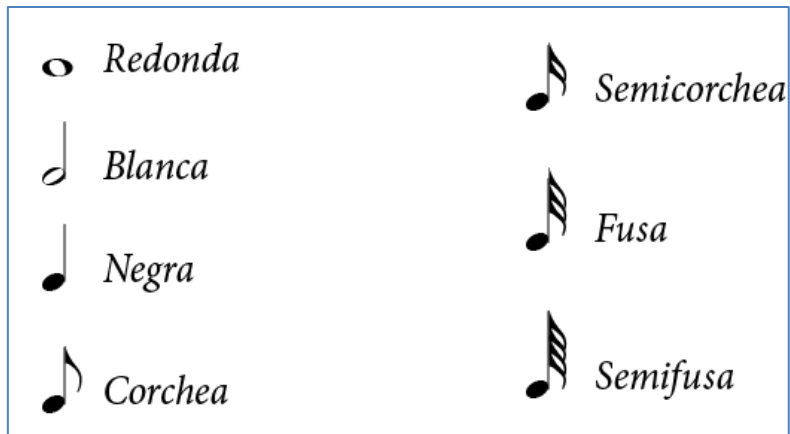


Las notas musicales en mBlock se presentan en nomenclatura inglesa. El comando que debemos utilizar para hacer sonar una determinada nota musical, durante un tiempo concreto se observa en la imagen derecha.

El desplegable de los tiempos de duración del sonido se asocia a cada tipo de nota: blanca, negra, etc.



Divirtiéndome con mBot Ranger



El cifrado equivalente a cada nota del pentagrama en la nomenclatura inglesa se muestra en la siguiente imagen:

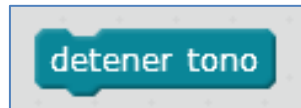
Do	→	C
Re	→	D
Mi	→	E
Fa	→	F
Sol	→	G
La	→	A
Si	→	B

Finalmente, reproducir los 4 primeros compases de la canción implicaría crear el siguiente script:



Divirtiéndome con mBot Ranger

Con el siguiente comando se detiene el tono:



5. Ejemplos de programación con diversos módulos

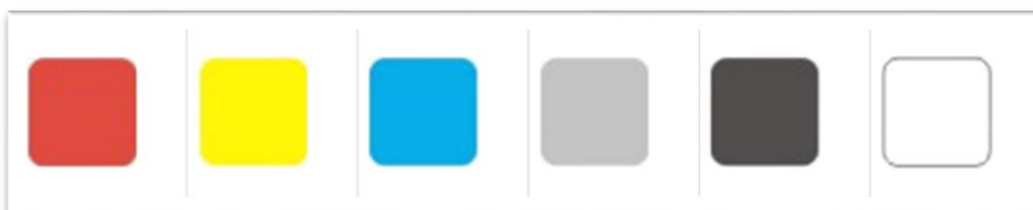
La placa Me Auriga del mBot Ranger puede programarse utilizando diferentes lenguajes de programación. Además de scratch, al ser una Arduino Mega, puede ser programada con Processing. Esto hace que podamos sacarle más rendimiento a la placa ya que, en teoría, tenemos un dos en uno. Aunque en este apartado 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 6 combina cuatro colores: blanco, amarillo, azul y negro. Pues bien, a él podremos conectar cualquier módulo cuyo RJ25 disponga de, como mínimo, alguno de esos colores. Si el ID del módulo es rojo, no podríamos conectarlo al puerto 6, pero si a cualquiera de los puertos numerados del 1 al 4. Si el ID del componente es gris, por ejemplo, sólo podríamos conectarlo al puerto 5 de la placa Auriga:



Puertos RJ25 de Me Auriga

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).



Divirtiéndome con mBot Ranger

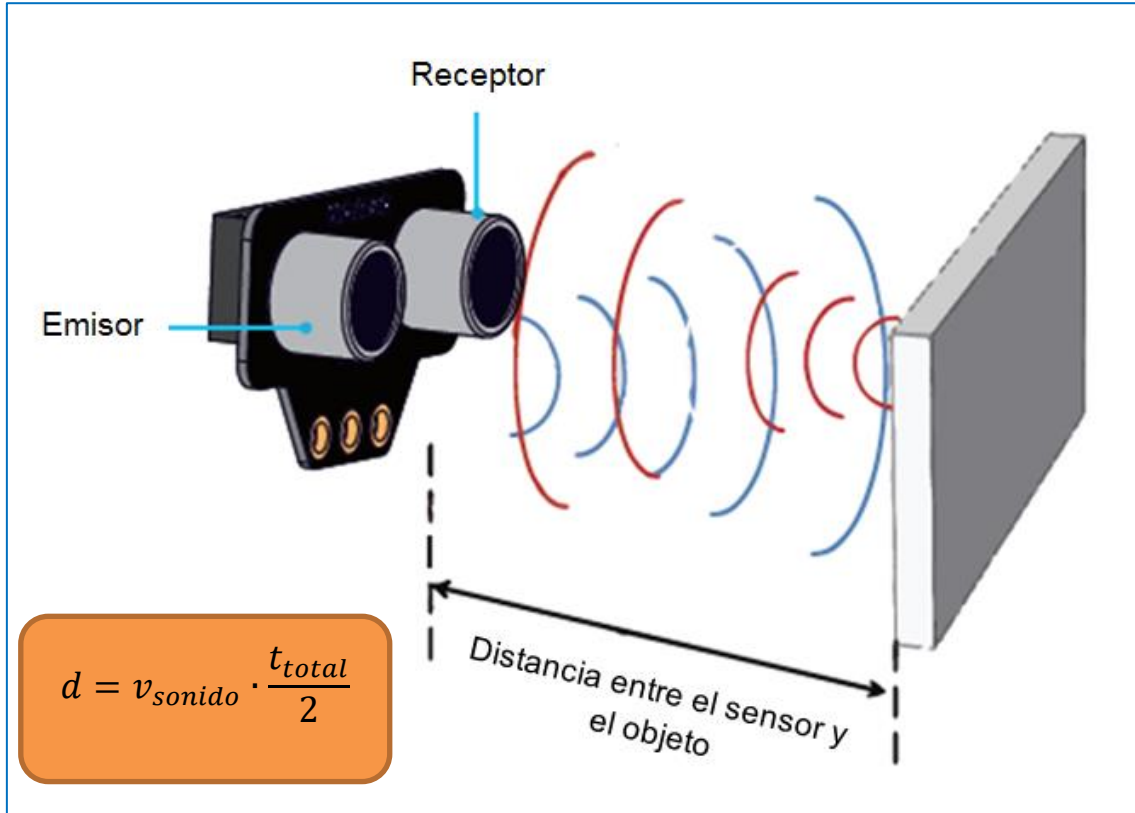
5.1. Módulo de ultrasonidos

Un módulo de ultrasonidos nos proporciona un dato numérico que se corresponde con la distancia entre el sensor y cualquier objeto que esté 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 cinco puertos de la placa Auriga numerados del 6 al 10.



Módulo ultrasonidos

El sensor se compone de dos elementos: un emisor de ondas de ultrasonidos y un receptor de las mismas. Como sabemos, el sonido se mueve a una velocidad constante conocida. Por lo tanto, la distancia nos la aporta logrando medir el tiempo que tarda la onda en salir del emisor y en llegar al receptor (ese tiempo se divide entre dos para obtener la distancia entre el objeto y el sensor):



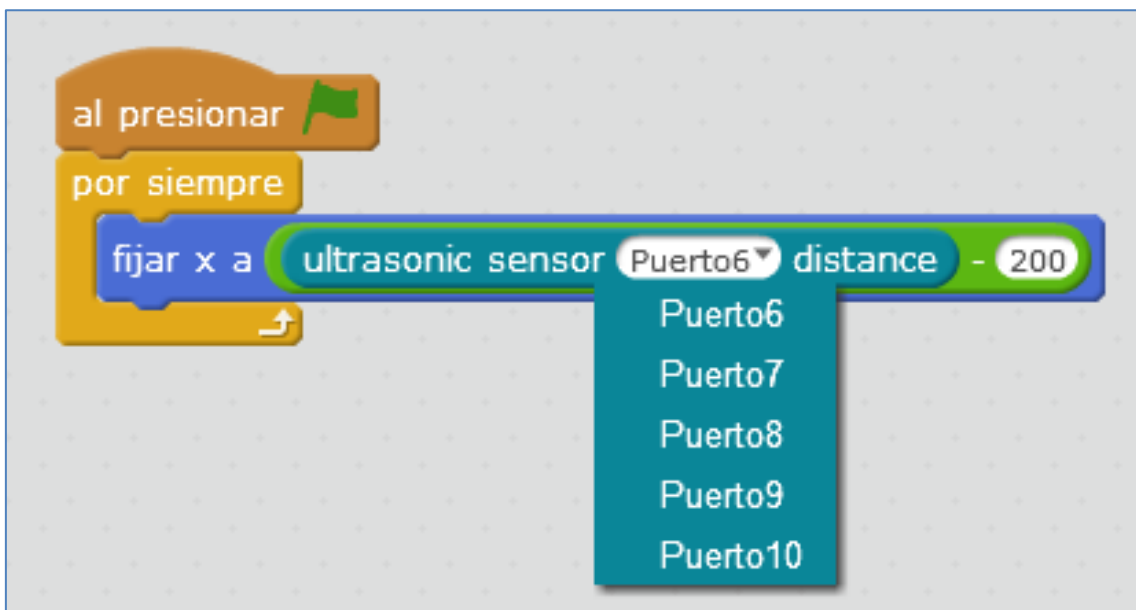
Funcionamiento del módulo ultrasonidos

Divirtiéndome con mBot Ranger

Supongamos que lo conectamos al puerto 6 de la placa Me Auriga 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 (acercando o alejando un cuerpo a este sensor). Una forma sería con el siguiente script:



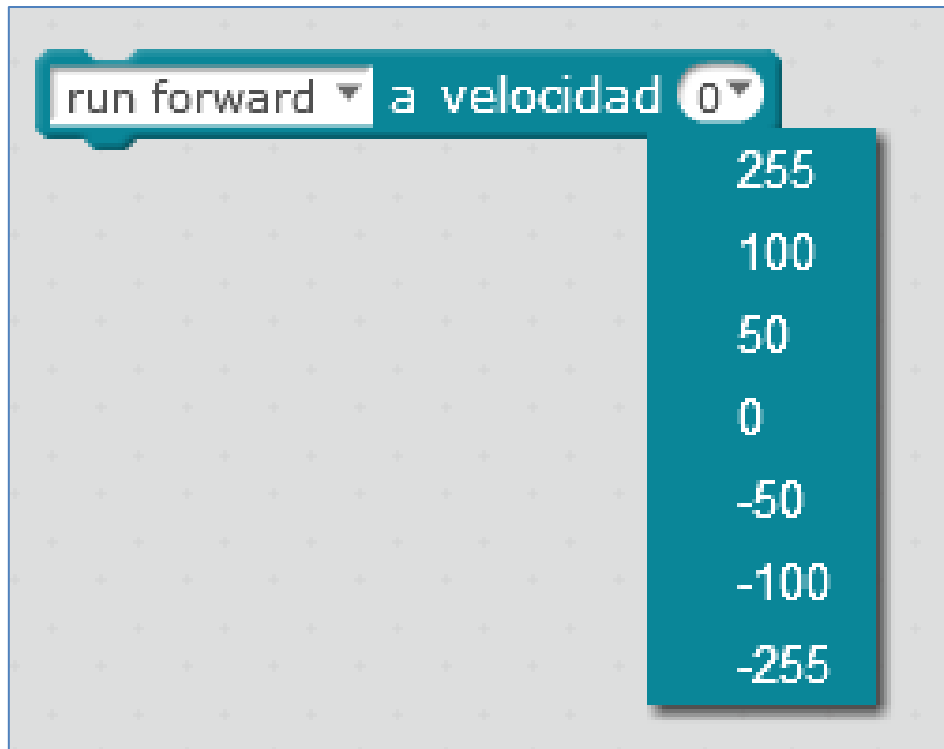
NOTA: Ojo, hay que fijarse en qué puerto de los 5 posibles tenemos conectado nuestro sensor de ultrasonidos.

5.2. Motores del robot

El robot mBot Ranger se compone de dos motores de continua de 7.4V a 180rpm. 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 sentido contrario (rotación inversa).

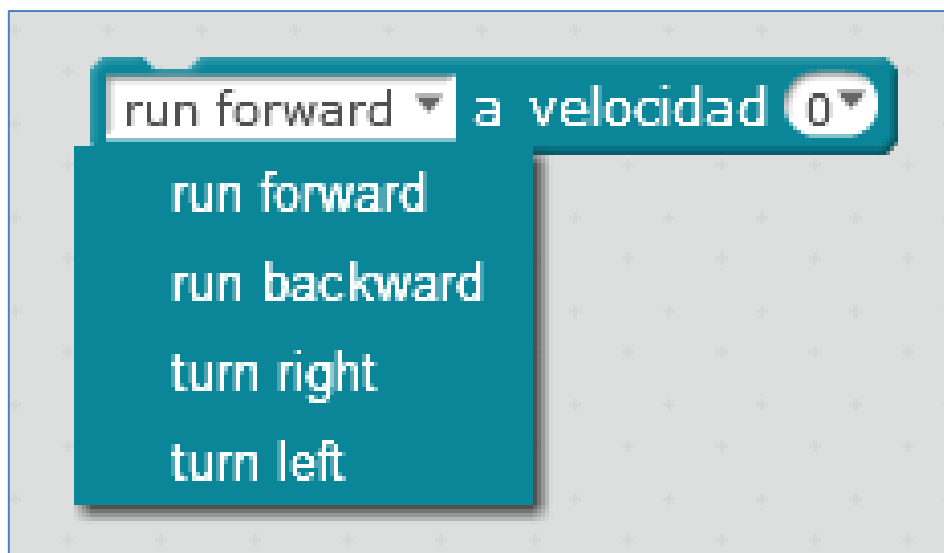


Motor DC 7.4V/180rpm

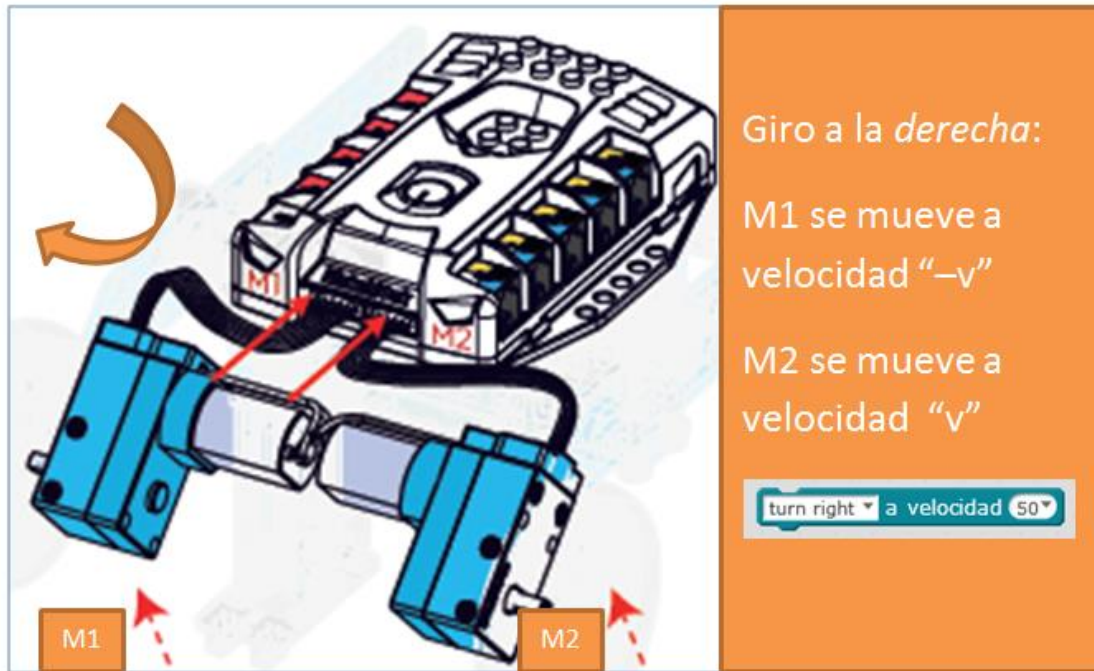


Aunque las velocidades que nos muestra su desplegable en la imagen anterior son 0, 50, 100 y 255; en su lugar podemos escribir otro valor numérico (en teoría, entre 0 y ± 255).

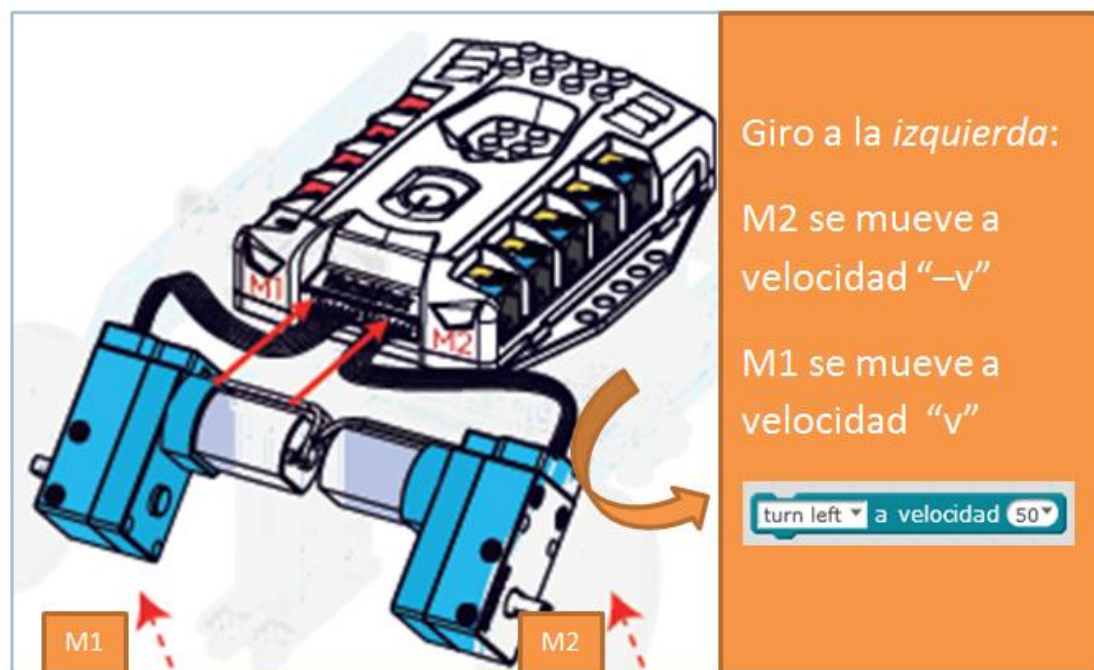
Con este comando podemos decidir no sólo su velocidad, sino también el sentido de giro de los dos motores del robot: si queremos que se mueva hacia adelante (run forward), hacia atrás (run backward), girar a la derecha (turn right) o girar a la izquierda (turn left):



Cuando elegimos la opción “run forward” (o “run backward”) con una velocidad determinada, ambos motores M1 y M2 giran en el mismo sentido a esa velocidad, bien hacia adelante o hacia atrás. En el caso de “turn right” (o “turn left”) a una velocidad “v”, el motor M1 (o M2) gira a “-v” y el motor M2 (o M1) se mueve a esa velocidad “v”:



Giro hacia la derecha

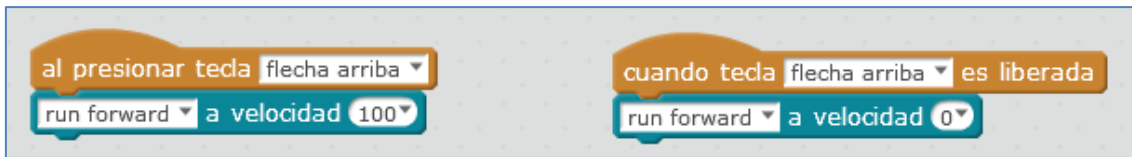


Giro hacia la izquierda

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 o libero la presión en la tecla “X”.

Divirtiéndome con mBot Ranger

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, ambos se pararán.



Podemos completar el movimiento de nuestro robot de modo que se mueva en todas las direcciones (hacia adelante, hacia atrás, hacia la derecha y hacia la izquierda) presionando y liberando diferentes teclas del teclado.

El script que consigue incorporar un movimiento de control completo para el robot mBot Ranger sería el siguiente:



Script para el movimiento de los motores con el teclado

Con el robot mBot podíamos controlar la velocidad de cada motor de forma independiente, de modo que, por ejemplo, uno se moviera a 100 y otro a 80. En el robot Ranger también es posible mover cada motor de forma independiente. La diferencia entre ambos robots radica en los comandos que debemos usar para ejecutar este movimiento de velocidad y sentido independiente para cada motor. Si queremos que el motor M1 se mueva a una velocidad de 100 y el motor M2 a una velocidad diferente de 80 (ambos hacia adelante), con el robot mBot usaríamos el script de la Fig1 y con el robot Ranger deberíamos usar el script de la Fig2:

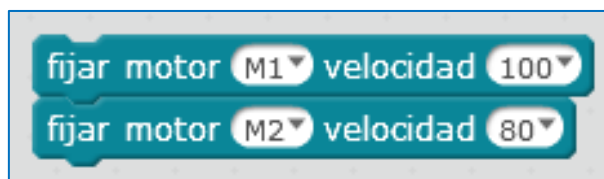


Fig1. Control de motores del mBot

Divirtiéndome con mBot Ranger

Como se observa, el motor M1 se asocia al *Banco1* y el M2 al *Banco2*. Es curioso el signo que denota el sentido de movimiento de cada rotor del motor. En M1, ha de ser negativo para que avance hacia adelante y positivo para que gire hacia atrás. Estos signos producen el efecto contrario en el motor M2:

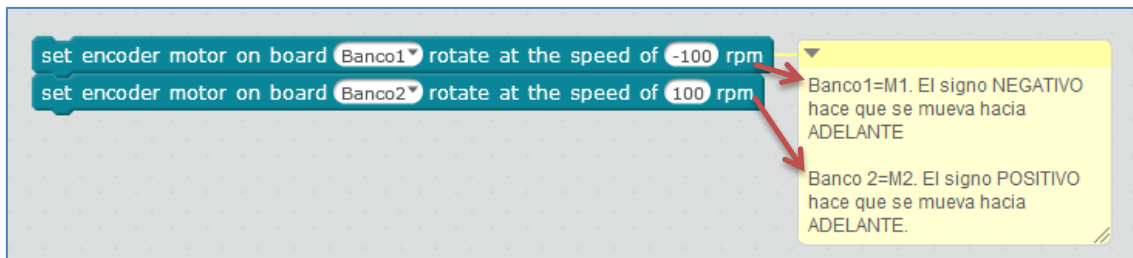
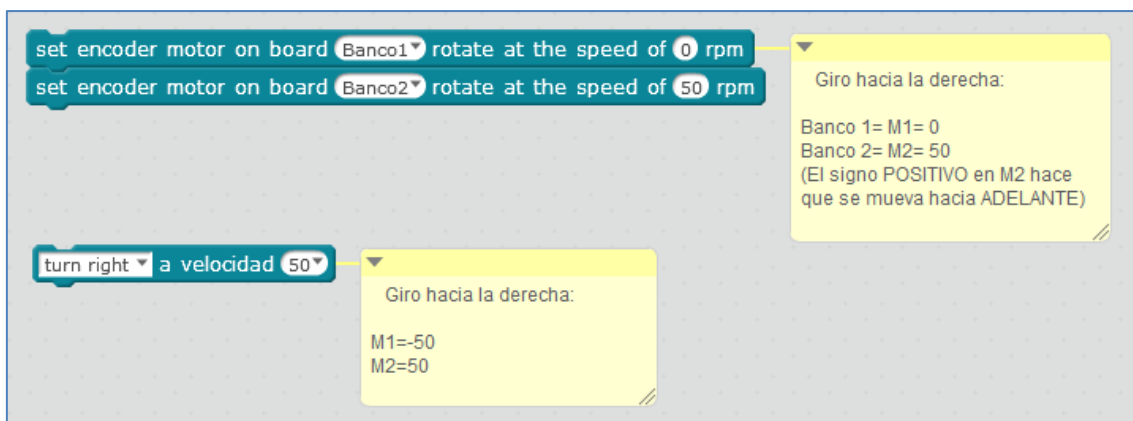


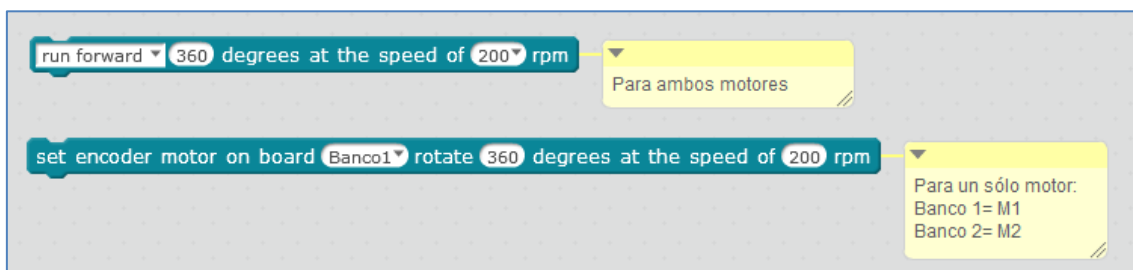
Fig2. Control de motores del Ranger

Una posible aplicación con estos comandos podría llevarse a cabo con el seguidor de línea. Cuando el robot se escapa de la línea, este debe volver a ella, bien sea girando hacia la derecha, izquierda o yendo hacia atrás. A veces nos interesa usar giros más precisos y más lentos y otras veces, giros más rápidos y bruscos:



Giro hacia la derecha

A mayores, podemos incluso controlar el número de vueltas a las que queremos que giren ambas o una rueda en particular del robot, pudiendo escoger una determinada velocidad (rpm) para ellas o incluso un sentido de giro (hacia adelante, atrás, derecha e izquierda). Para conseguirlo, usaremos los siguientes comandos:



Finalmente, también podemos testear la cantidad de grados a los que gira cada rueda, así como, su velocidad:

encoder motor on board Banco1 position/degrees

encoder motor on board Banco1 speed/rpm

Hasta ahora, en este apartado, hemos hablado del control de los motores de las ruedas del robot mBot Ranger. En su placa Auriga vemos que disponemos de 4 conectores rojos específicos para motores. Además, en los conectores del 6 al 10 podemos conectar servos. Para su programación (motores DC, paso a paso o servos) debemos usar los siguientes comandos:

set DC motor Puerto1 speed 0

fijar motor de pasos motor Puerto1 velocidad 3000 distancia 1000

set encoder motor Puerto1 Banco1 rotate 1000 degrees at the speed of 180 rpm

fijar servo Puerto6 Banco1 ángulo 90

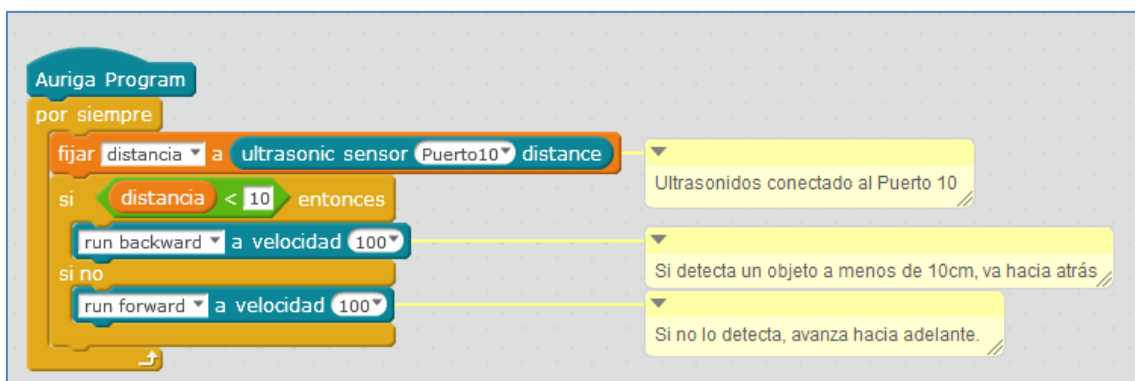
Comandos para: motor DC, paso a paso y servo

5.3. Motor y detección ultrasónica

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

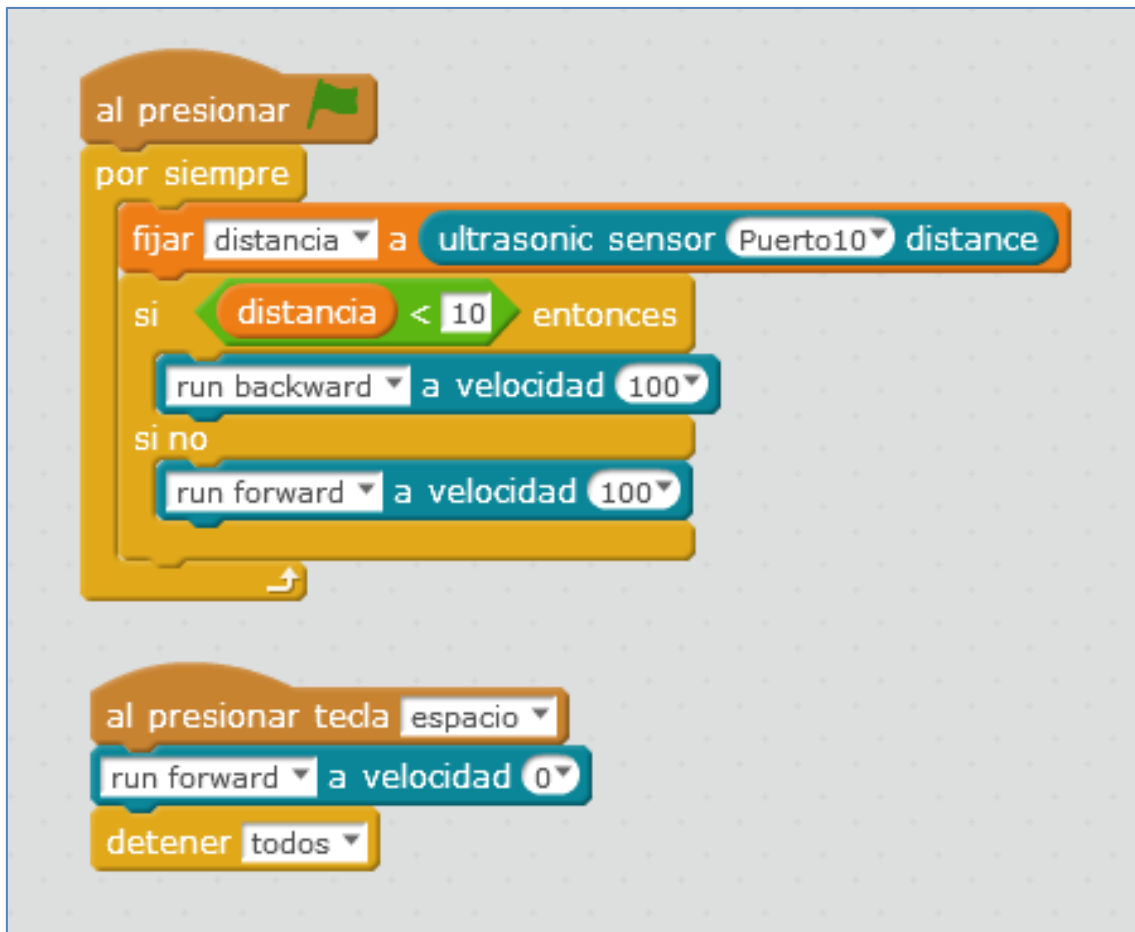
Si el objeto que detecta se encuentra a menos de 10 cm del robot, debe ir hacia atrás. En caso contrario, debe avanzar.

El script que ejecuta el reto anterior es el siguiente:



Divirtiéndome con mBot Ranger

Si no queremos cargar el programa en la placa Auriga sólo debemos cambiar el comando azul **Auriga Program** por el comando típico de la bandera verde. A mayores, he incluido la parada de los motores. Esta se realizará cuando presione la tecla “espacio” del teclado:

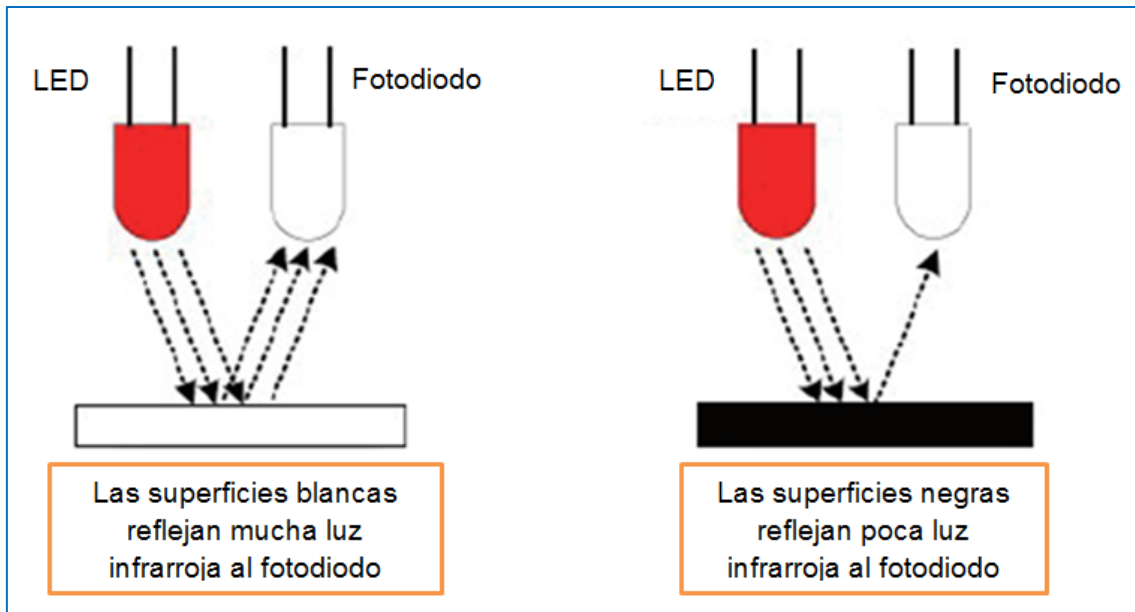


5.4. Detector de línea

Este módulo sensor hace posible que nuestro robot se mueva a lo largo de una línea negra sobre el suelo. En el kit mBot Ranger se sitúa en su parte frontal funcionando *por reflexión*. En nuestro robot, como el color de su ID es azul, lo podemos conectar a cualquiera de los puertos numerados del 6 al 10. En los ejemplos que siguen, usaremos el Puerto 9 de la placa Me Auriga.

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* (o fotodiodo) 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.

Divirtiéndome con mBot Ranger

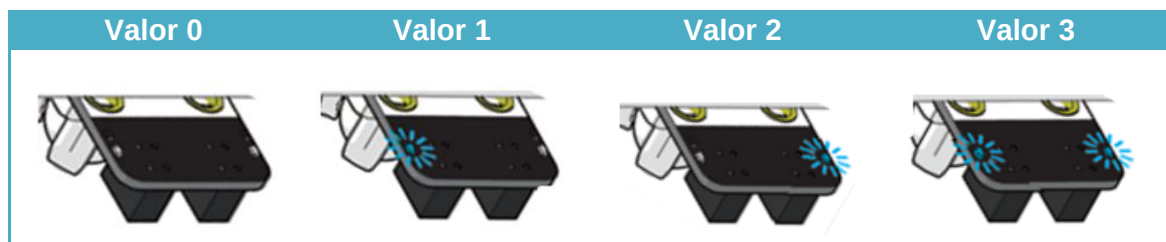


Funcionamiento del módulo detector de línea

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.

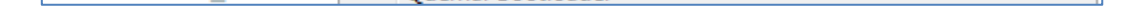
Las 4 posiciones pueden verse gráficamente en la siguiente imagen:



Podemos testear el buen funcionamiento del módulo seguidor de línea desde el IDE de Arduino, cargando en la placa Me Auriga el ejemplo "*LineFollowerTest*". Este firmware se incluye en las librerías MakeblockDrive² que nos proporciona la casa:

Primero abrimos el ejemplo en el IDE de Arduino:

² Descarga: <http://learn.makeblock.com/en/Makeblock-library-for-Arduino/>



Divirtiéndome con mBot Ranger

Antes de cargar el programa a nuestra placa Me Auriga, debemos realizar unos pequeños cambios en él. Estas modificaciones son mínimas; sólo debemos referenciar la placa en la cual se va a cargar este firmware y el puerto RJ25 al que se conectará el módulo seguidor de línea que queremos probar.



Con estos cambios, nuestro firmware quedaría como se ve en la siguiente imagen:

```
LineFollowerTest Arduino 1.6.8
Archivo Editar Programa Herramientas Ayuda

LineFollowerTest$

/**
 * \par Copyright (C), 2012-2016, MakeBlock
 * @file LineFollowerTest.ino
 * @author MakeBlock
 * @version V1.0.0
 * @date 2015/09/09
 * @brief Description: this file is sample code for Me line follower module.
 *
 * Function List:
 * 1. uint8_t MeLineFollower::readSensors(void)
 *
 * \par History:
 * <pre>
 * <Author> <Time> <Version> <Descr>
 * Mark Yan 2015/09/09 1.0.0 Rebuild the old lib.
 * </pre>
 */

#include "MeAuriga.h"

MeLineFollower lineFinder(PORT_9); // Line Finder module can only be connected to PORT_3, PORT_4, PORT_5, PORT_6 of base shield. */

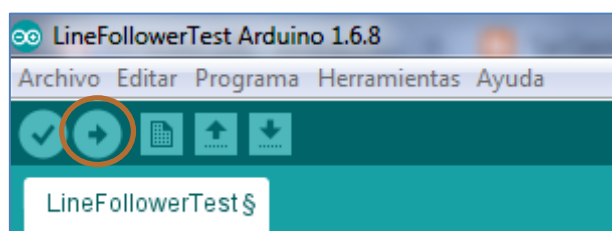
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int sensorState = lineFinder.readSensors();
  switch(sensorState)
  {
    case S1_IN_S2_IN: Serial.println("El sensor 1 y 2 están dentro de la línea negra"); break;
    case S1_IN_S2_OUT: Serial.println("El sensor 2 está fuera de la línea negra"); break;
    case S1_OUT_S2_IN: Serial.println("El sensor 1 está fuera de la línea negra"); break;
    case S1_OUT_S2_OUT: Serial.println("El sensor 1 y 2 están fuera de la línea negra"); break;
    default: break;
  }
  delay(200);
}

Subido

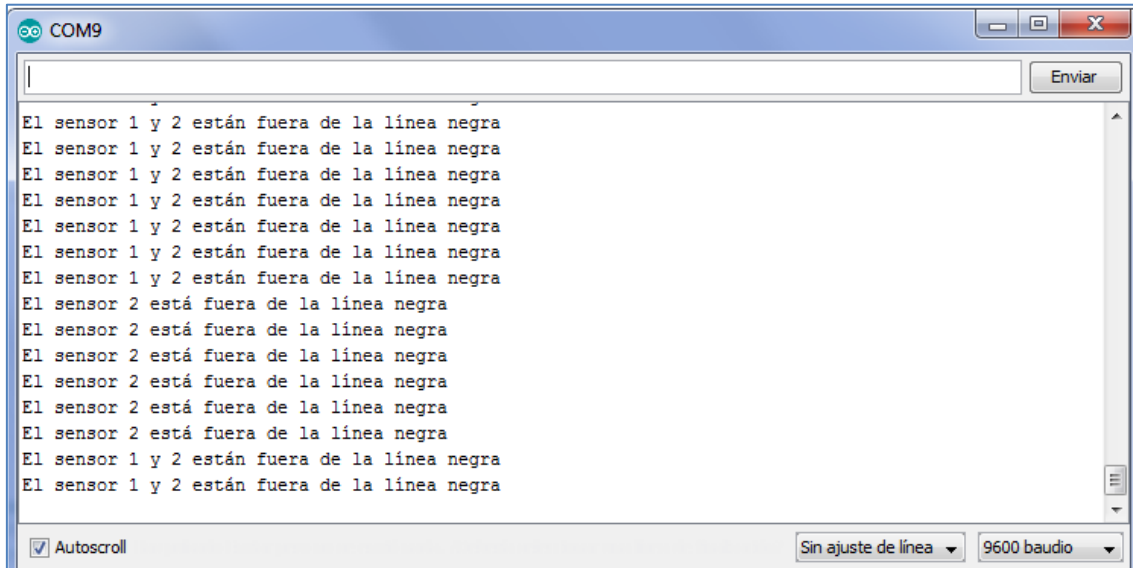
El Sketch usa 5.056 bytes (1%) del espacio de almacenamiento de programa. El máximo es 253.952 bytes.
Las variables Globales usan 1.118 bytes (13%) de la memoria dinámica, dejando 7.074 bytes para las variables locales. El máximo es 8.19
```

Finalmente, cargamos el programa en la placa haciendo clic en el icono flecha derecha debajo del menú superior:



Divirtiéndome con mBot Ranger

Si ahora visionamos la respuesta en el puerto serial del IDE de Arduino, al situar bajo el módulo seguidor de línea un objeto “blanco”, se nos mostrará un mensaje que nos dice que ambos sensores están fuera de la línea negra. Pero, si sólo situamos este objeto “blanco” bajo el sensor 2, el mensaje cambia y podemos leer que sólo el sensor 2 está fuera de la línea negra:

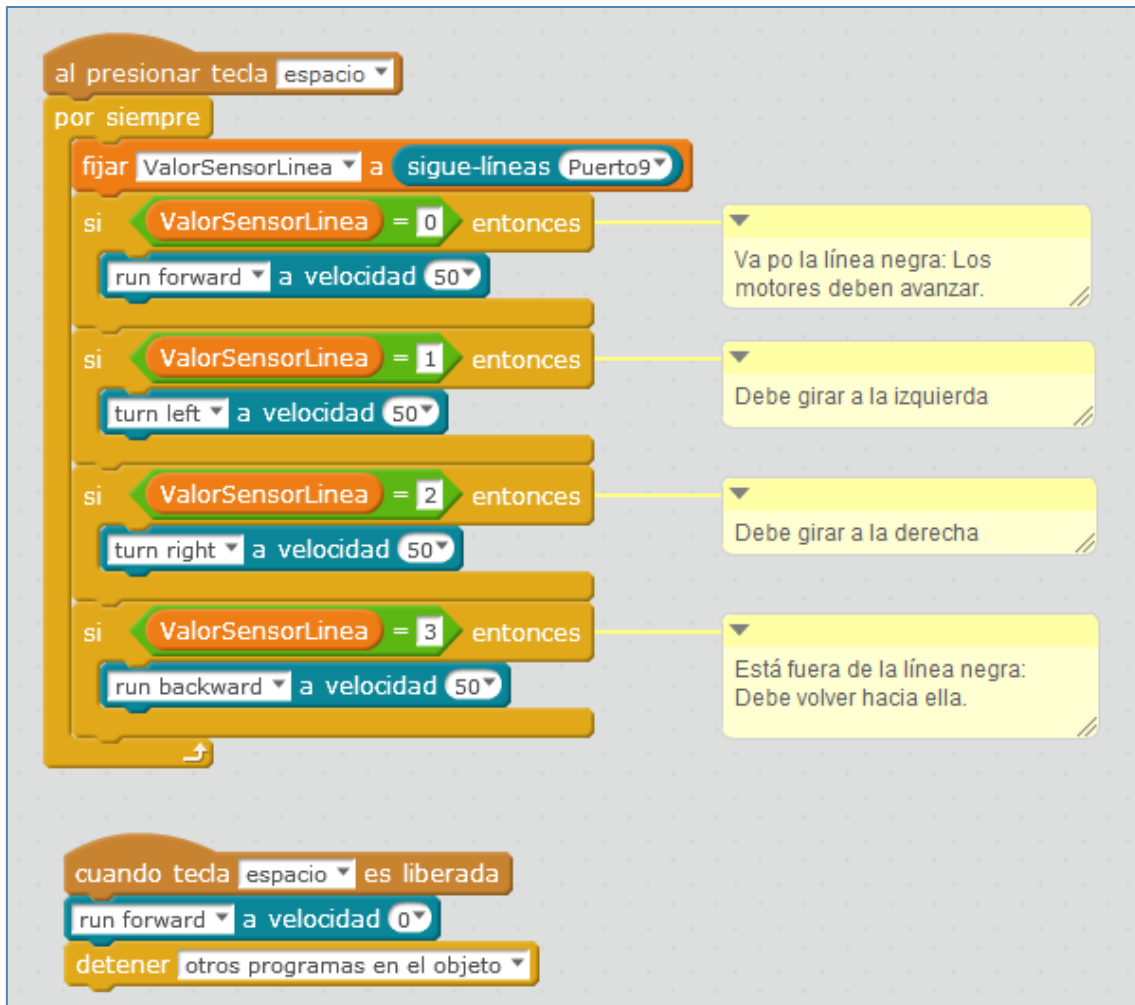


Un reto típico que podemos programar con este módulo es conseguir que nuestro robot siga una línea negra o una línea blanca. Vamos a hacerlo con los comandos del mBlock.

Seguidor de línea negra

En el siguiente ejemplo no he cargado el programa en la placa y he querido que se accione al pulsar la tecla espacio. Para evitar el cable de conexión de la placa Me Auriga con el ordenador, cambiaremos su módulo Bluetooth por el módulo 2.4G y conectaremos su pendriver al PC. El módulo seguidor de línea de ID azul se ha conectado al puerto 9 de la placa Me Auriga (podríamos usar cualquiera de los puertos numerados entre el 6 y el 10).

El programa seguidor de línea negra creado se explica como sigue: Tras presionar la tecla espacio, mBot Ranger ejecuta un bucle continuo testeando los 4 posibles valores numéricos que puede detectar el sensor de línea. Por lo tanto, testea si el robot ve completamente la línea oscura y, si es así, se mueve recto. De lo contrario, testea si el robot está a la izquierda (a la derecha) y, si lo está, gira a la derecha (izquierda) y hace que el robot regrese a la línea negra. Finalmente, testea si mBot Ranger está lejos de la línea de negra y, en ese caso, hace que retome la línea de color negro. Se repite el ciclo hasta que deje de presionar la barra espaciadora.



Script seguidor de Línea Negra

En su programación he creado la variable “ValorSensorLinea”. En ella se guardarán los diferentes valores numéricos del estado de patrullaje de la línea (0, 1, 2, 3). La rotación del motor depende de la desviación que se produzca, facilitando el ajuste de la dirección del robot. El sensor de línea, con ID azul, se ha conectado al puerto 9 de la placa Me Auriga.

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 utilizar el módulo bluetooth que viene en el kit del robot Ranger y cargar el programa a la placa. Los cambios que se producirían en el programa anterior son mínimos, tal y como puede observarse en el siguiente script seguidor de línea negra:

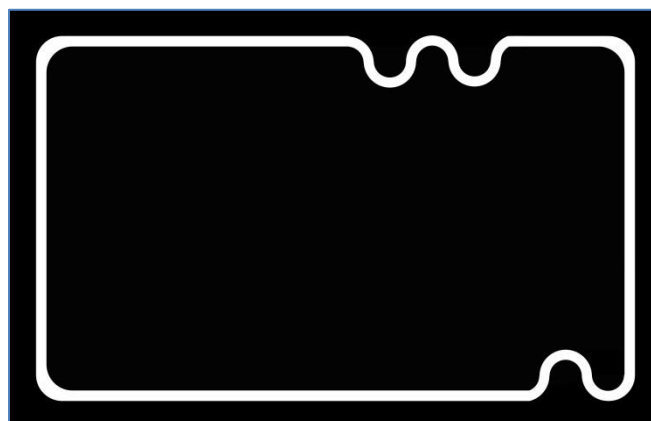


Script seguidor de línea negra

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.

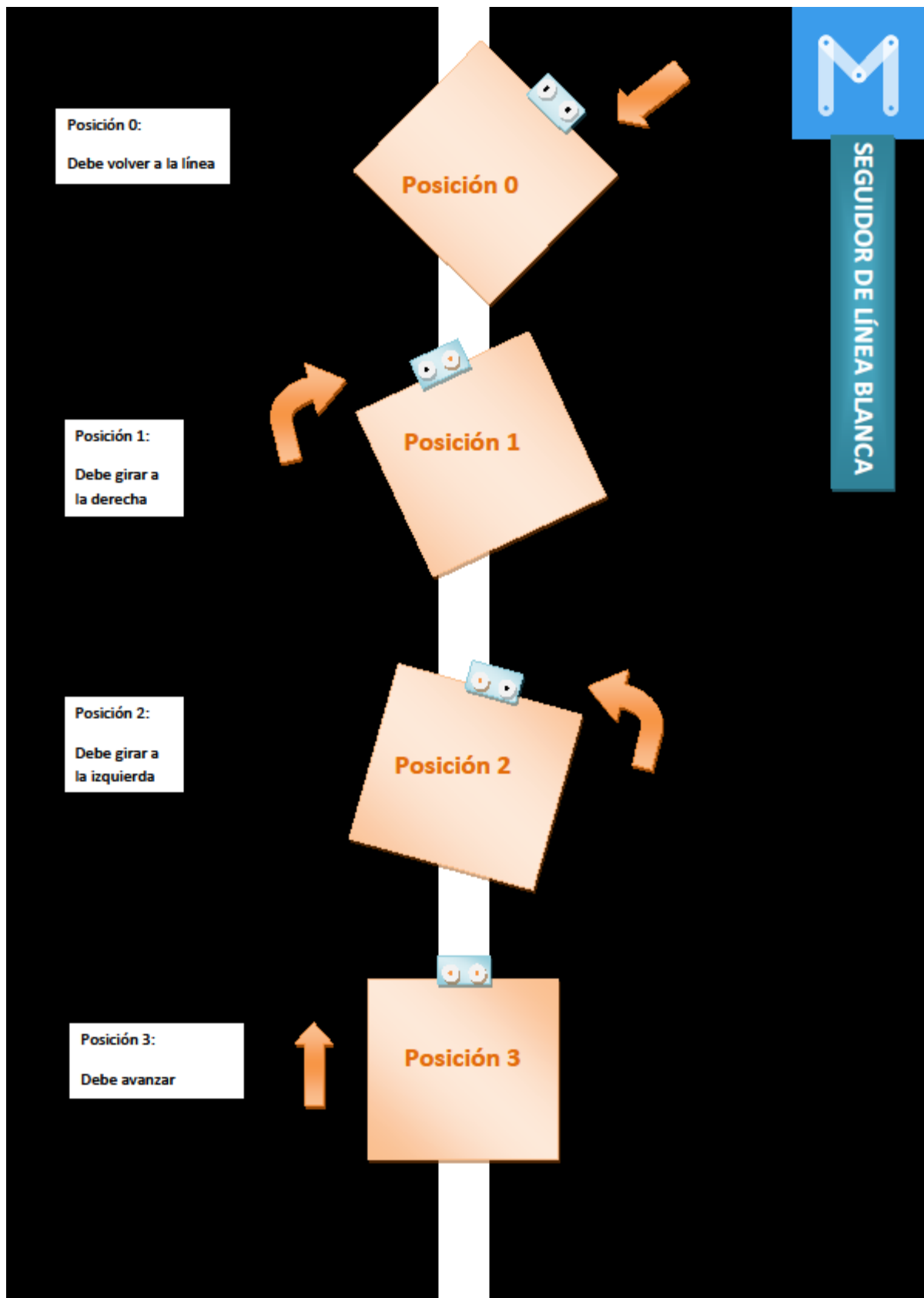
Seguidor de línea blanca

A la hora de programar nuestro robot mBot Ranger para que siga la línea blanca, debemos tener en cuenta qué posición nos devuelve el sensor de línea cuando detecta que está dentro o fuera de esa línea blanca.



Divirtiéndome con mBot Ranger

Estas posiciones, pueden verse representadas en la siguiente imagen:



Divirtiéndome con mBot Ranger

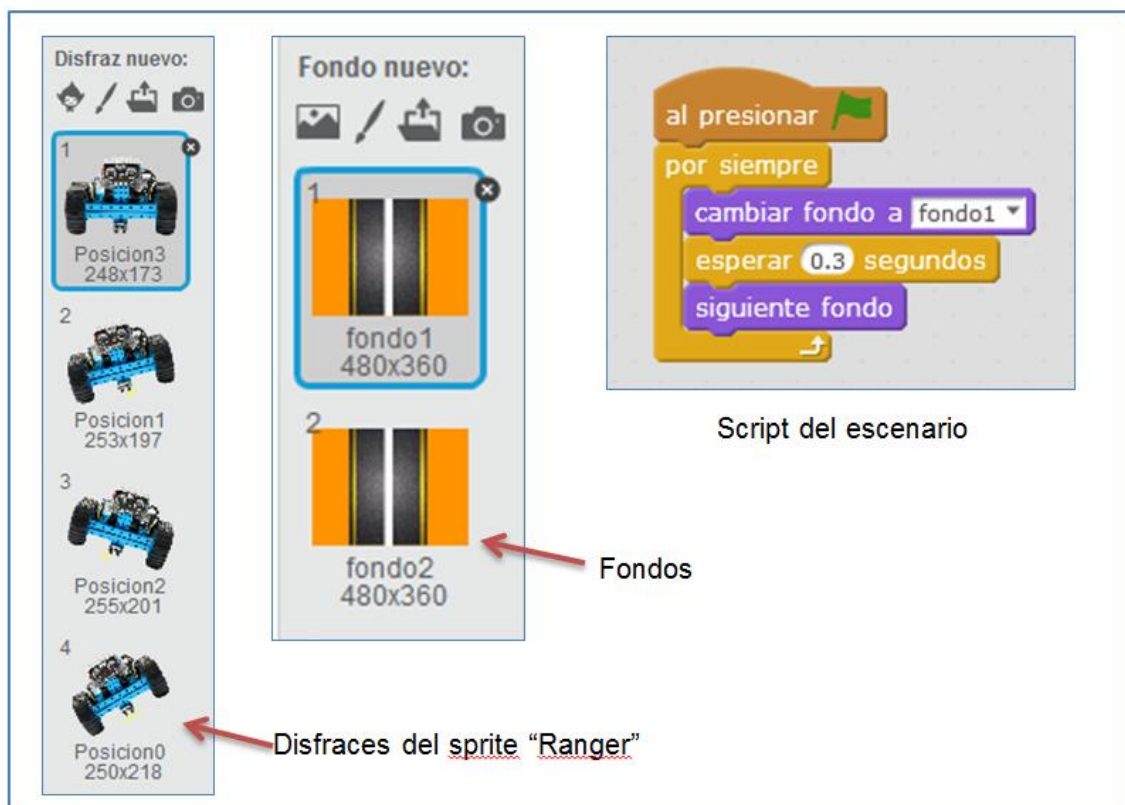
Por lo tanto, con pequeñas modificaciones en el programa “seguidor de línea negra”, ya tendríamos el script para un robot seguidor de línea blanca.

Como mejora, podemos crear un programa que simule en el escenario del mBlock las acciones que sigue el robot en una pista real. En este caso, al ser un seguidor de línea, cambiaríamos el módulo bluetooth de la placa Me Auriga por el módulo 2.4G, consiguiendo con ello evitar el cable de conexión de la placa con el PC.

Por lo tanto, nuestro reto no es sólo programar el robot para que logre seguir la línea blanca. A mayores, buscamos lograr simular sus acciones en el escenario del mBlock.

Comenzamos diseñando los objetos y fondos que nos lleven a poder implementar correctamente este reto:

Nuestro programa dispondrá de un objeto llamado *Ranger* con 4 posibles disfraces, cada uno de los cuales simulará una posición del sensor de línea. En cuanto al escenario, se han creado dos pistas, que llamo “*fondo1*” y “*fondo2*”, pretendiendo con ellas simular el movimiento del robot por la carretera que se visualiza en el escenario del software mBlock. Esta percepción del movimiento se consigue forzando un cambio continuo de fondos, de uno a otro en un tiempo pequeño (0,3 segundos):



Disfraces del objeto y fondo. Script del escenario.

En la siguiente imagen se observa como se vería el objeto sobre uno de los fondos del escenario:



El script para el objeto *Ranger* presenta dos partes. Por un lado se programa el movimiento que debe realizar el robot para seguir la línea blanca (Fig1) y por otro lado se programa la simulación que realizará en el escenario (Fig2). De este modo, el programa final para el objeto *Ranger*, que podemos utilizar para nuestro mBot Ranger (en este caso, se ha creado sin añadir los tiempos de espera o retardos en cada movimiento) es el siguiente:

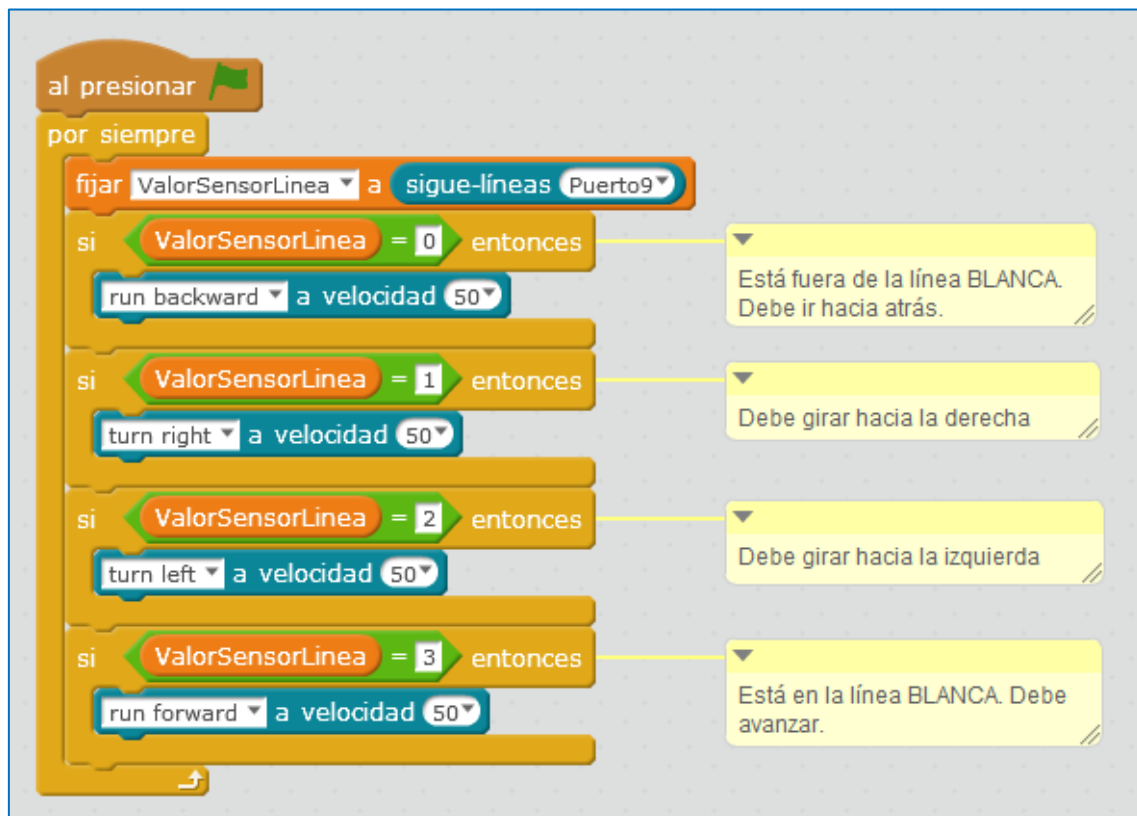


Fig1. Script Seguidor de línea blanca



Fig2. Script Simulación del seguidor de línea blanca.

Con el programa se 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)

Si quisiéramos cargar el programa en la placa, eliminaríamos el script de la simulación (Fig2) y cambiaríamos el comando de la bandera verde de la Fig1 por el comando del bloque de Robots **Auriga Program**:

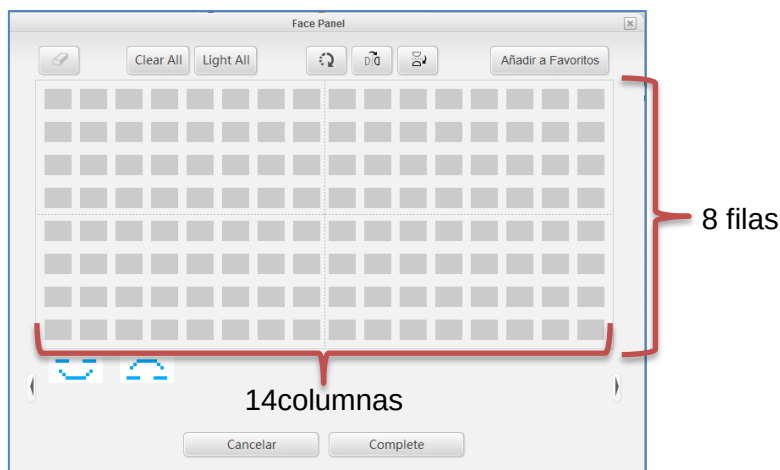


Divirtiéndome con mBot Ranger

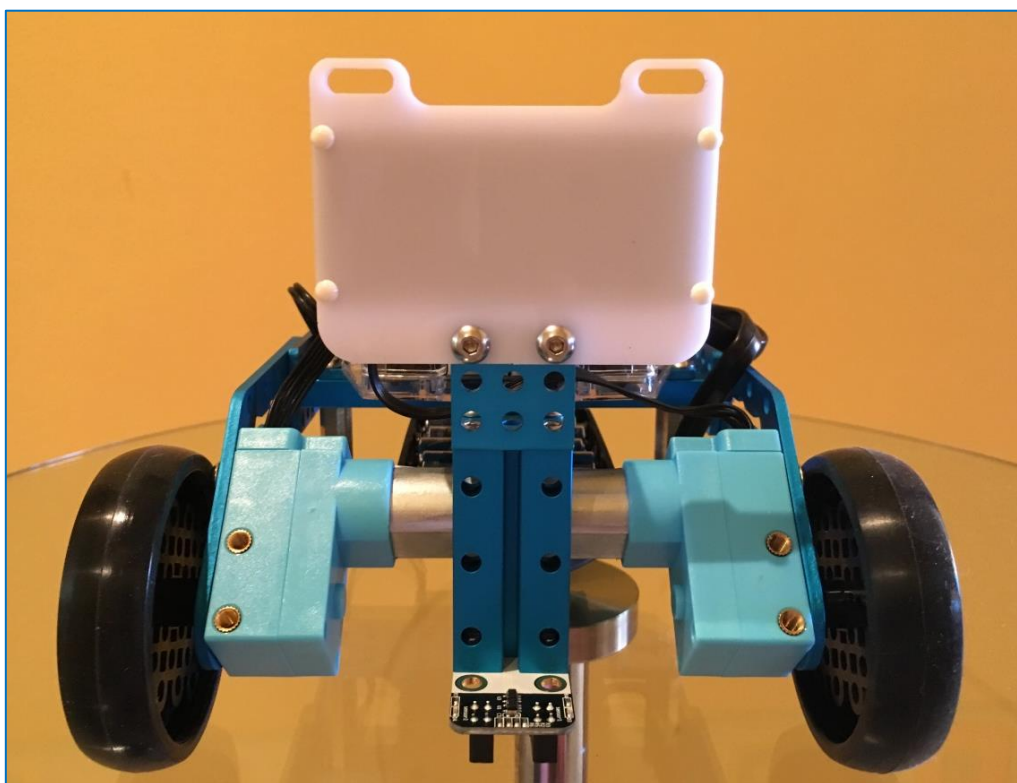
Finalmente iríamos a la pestaña Editar > Modo Arduino > Upload to Arduino. Y, tras un tiempo de espera, el programa se cargaría exitosamente en la placa del robot.

5.5. Matriz de LEDs 8×16

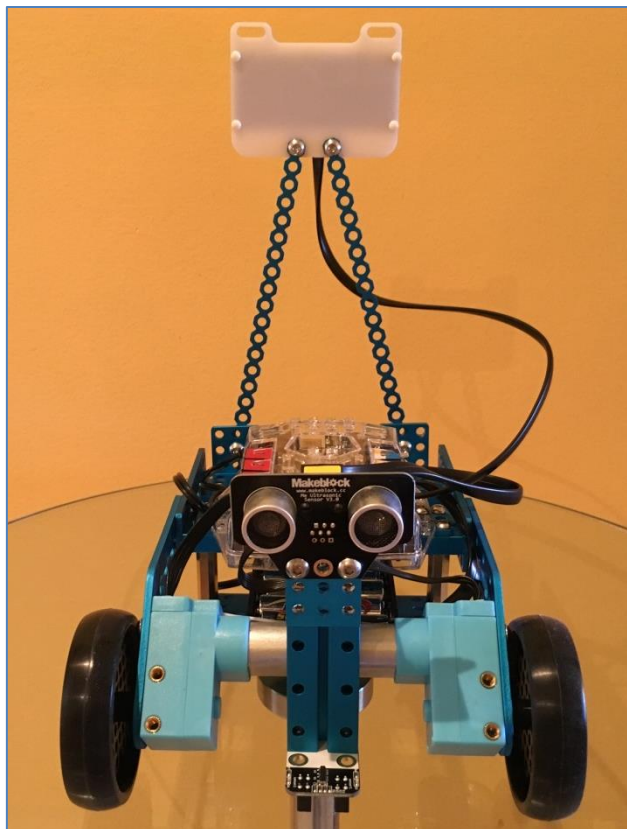
Nuestra matriz de LEDs es un conjunto de arrays de LEDs formado por 8 filas y 16 columnas que pueden ser encendidos y apagados de forma individual.



Antes de nada, debemos montar la matriz de LEDs en nuestro robot. Como su ID es azul podemos conectarla a cualquier puerto de la placa Me Auriga numerado entre 6 y 10. 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:

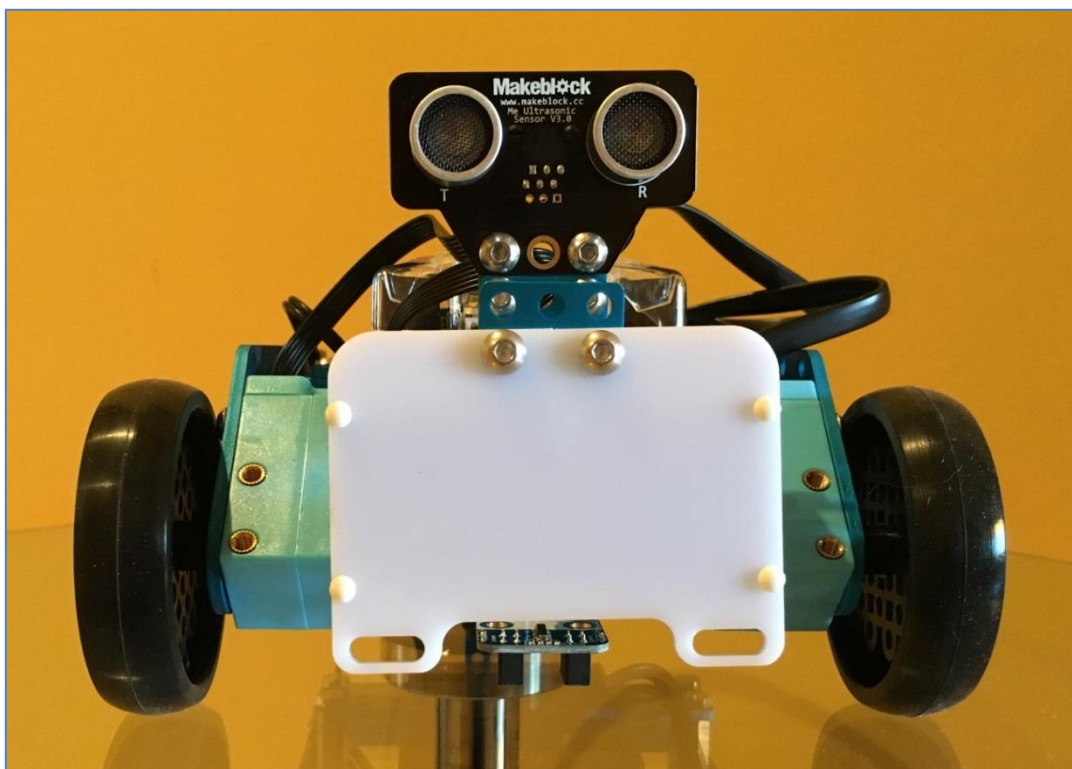


Matriz de LED en posición correcta para su fácil programación



Matriz de LED en posición correcta para su fácil programación

Otra posibilidad forma de montaje (invertida) que varía su programación, puede verse en la siguiente figura:



Matriz de LED invertida

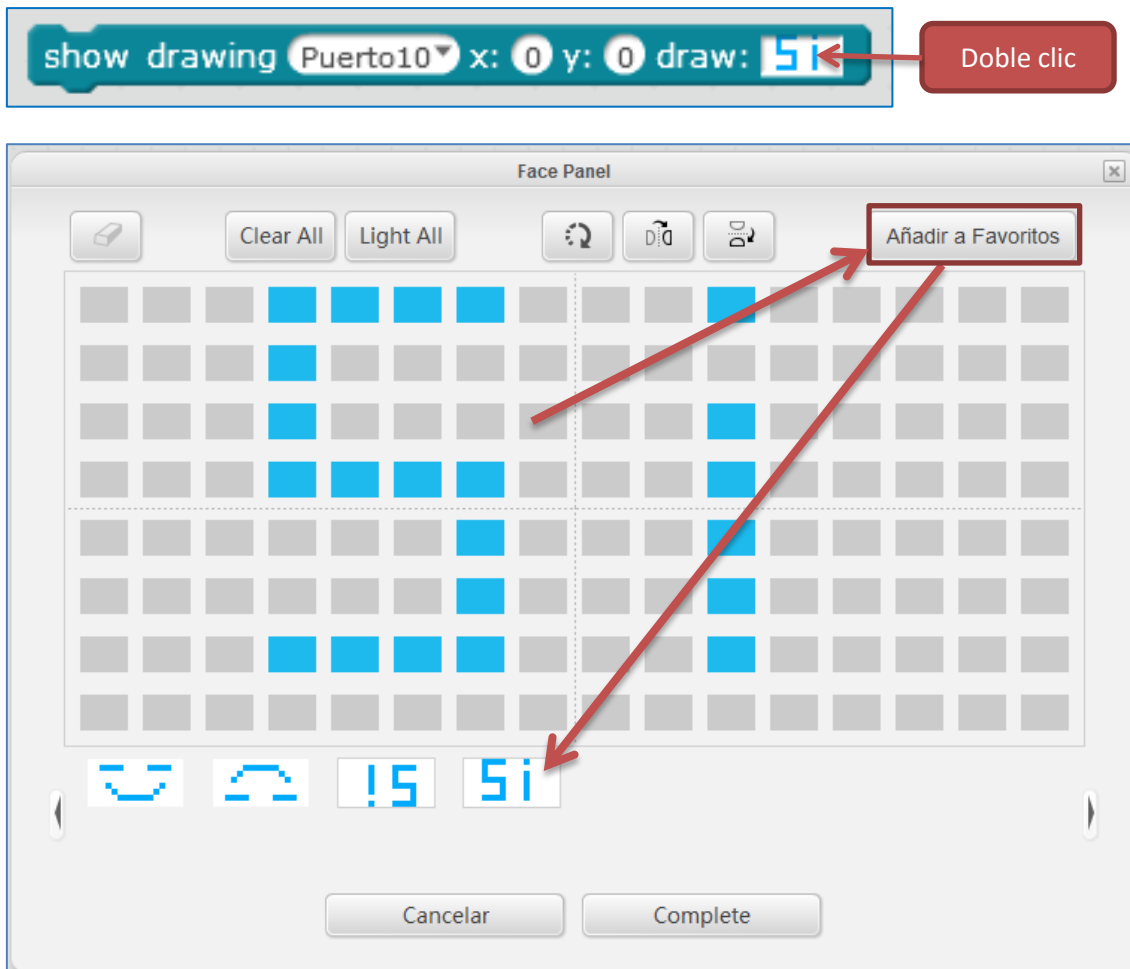
Divirtiéndome con mBot Ranger

Después de implementarla debemos seguir los siguientes pasos:

1. Conectar el mBot Ranger al ordenador mediante un cable USB y actualizar el firmware (*Conectar > Actualizar Firmware*)
2. Encender el robot y seleccionar el puerto serie y placa correspondientes.
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 en el gráfico de la matriz de LEDs, e incluso guardarlo en favoritos!

(OJO! El gráfico se visionaría 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 de la matriz con mBlock. En los siguientes ejemplos se ha conectado la matriz de LEDs a los puertos 8 o 10 de la placa Me Auriga:

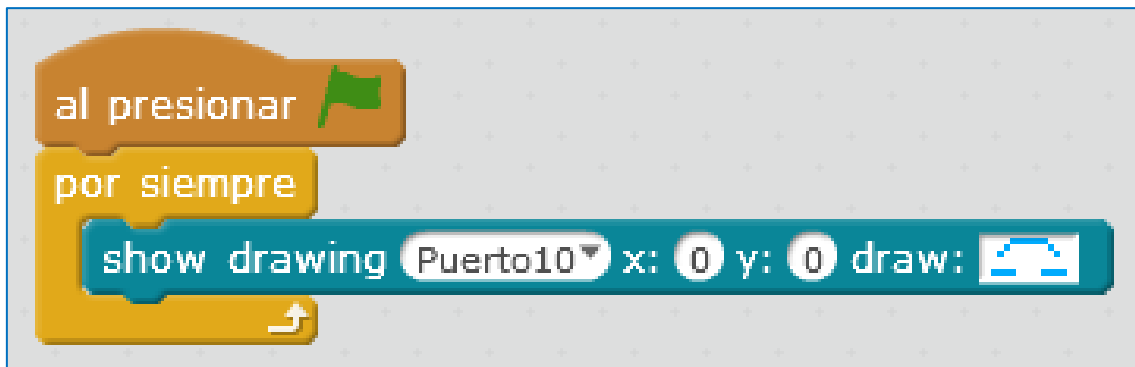
Divirtiéndome con mBot Ranger

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

Con la matriz correctamente montada, crearíamos el siguiente programa:



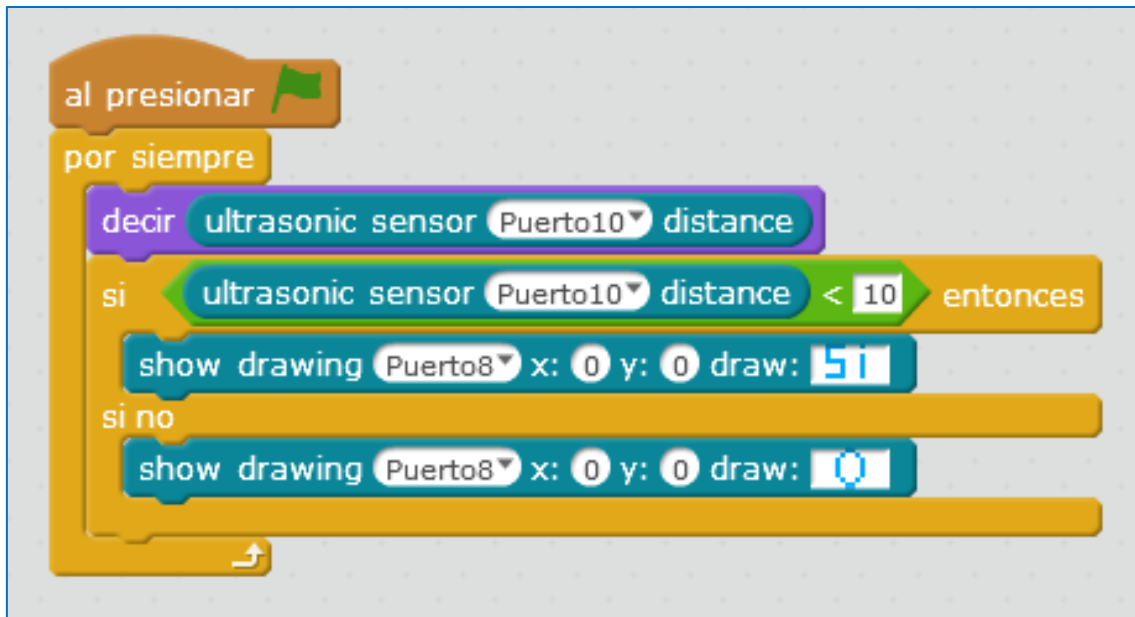
Pero, con la matriz en montaje invertido, deberíamos enviarle al robot el siguiente programa:



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:

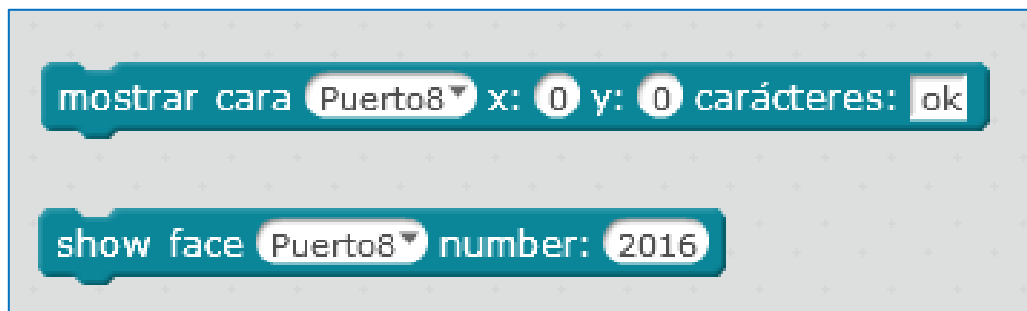


Ejemplo 3: Combinando la matriz (Puerto 8) con el sensor de ultrasonidos (Puerto 10), podríamos crear un programa que nos avise, mostrando la palabra “Si” en la matriz de LEDs, si un objeto se encuentra a una distancia menor de 10 cm del robot. En caso contrario debe ejecutar otro gráfico.

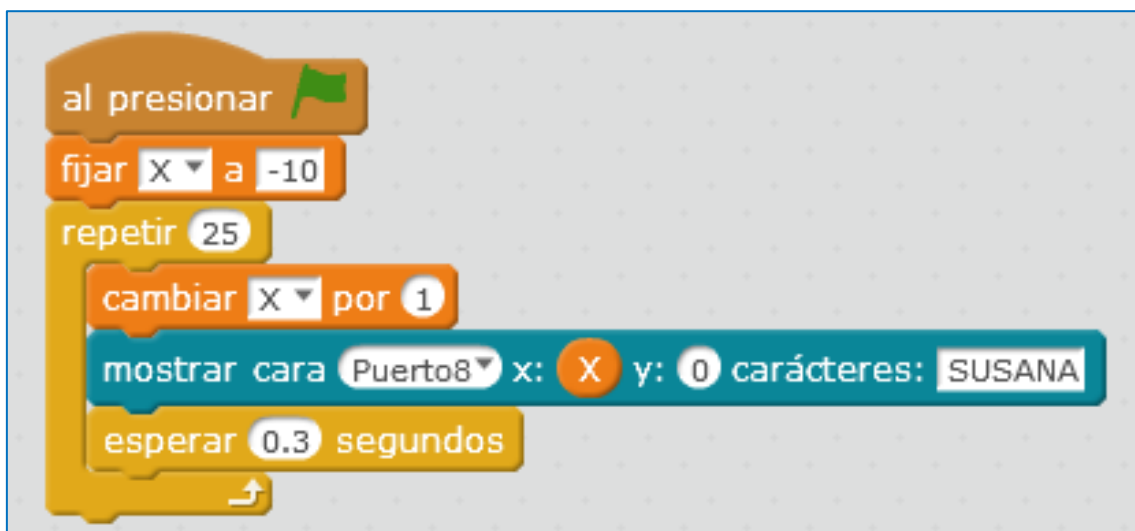


Ejemplo 4: Podemos enviar un número a la matriz con la instrucción **show face**, así como, un texto con la instrucción **mostrar cara**.

OJO, si está invertida, el texto y el número también hay que invertirlos



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



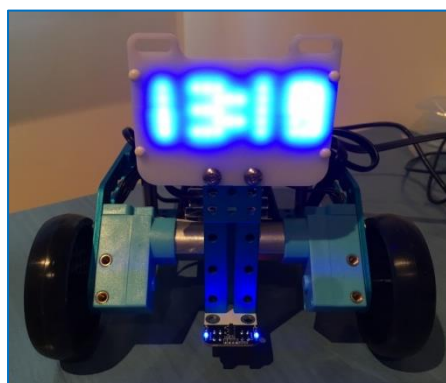
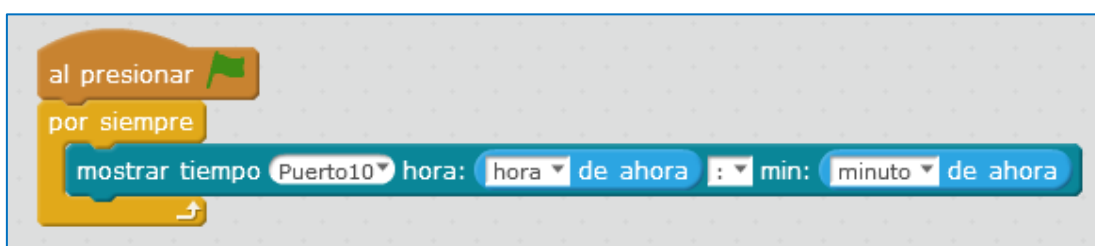
Divirtiéndome con mBot Ranger

Se ha decidido una espera de 0.3 segundos y de esta forma nos da tiempo a leer el texto que va corriendo por la matriz de LEDs.

Podemos desear que en la matriz se visiones, de forma continua, el texto anterior. Esto hace necesario incluir un bucle “Por siempre” e inicializar la coordenada X a -22 cuando esta llegue al valor cero. El script sería el siguiente:



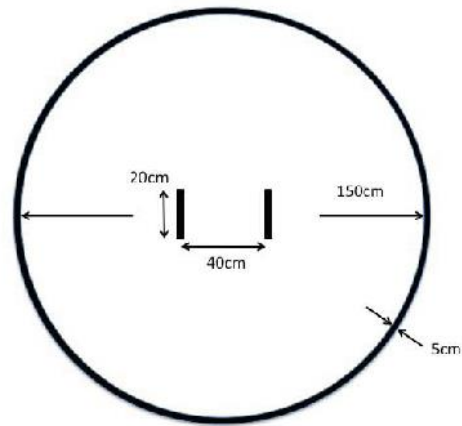
Ejemplo 5: También podemos enviar la hora y minutos y que esta se vaya actualizando. Un script que nos serviría para ese fin se muestra en la siguiente figura:



Divirtiéndome con mBot Ranger

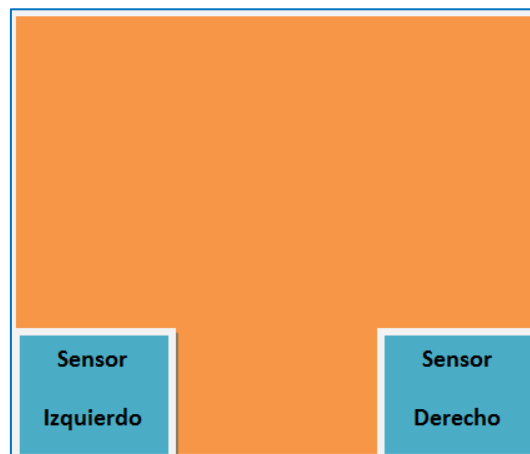
5.6. Robot sumo

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:



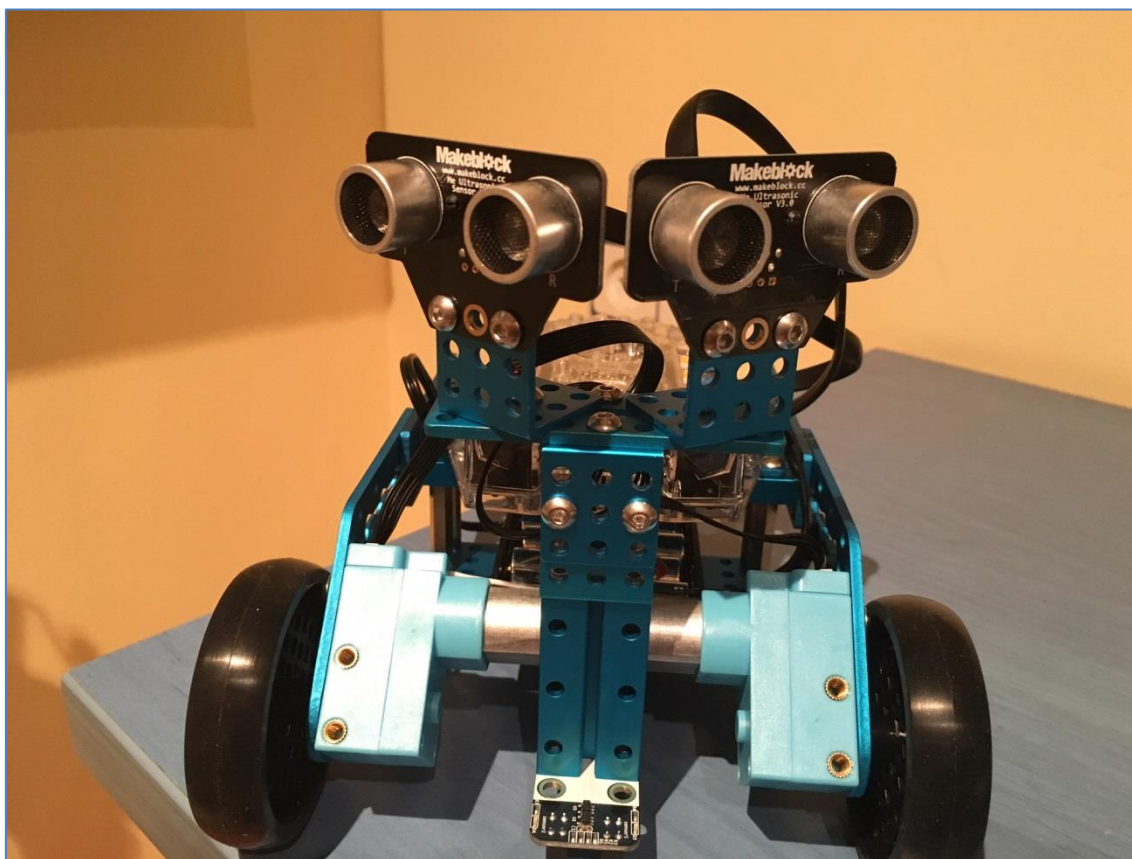
Esquema parte frontal del robot

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



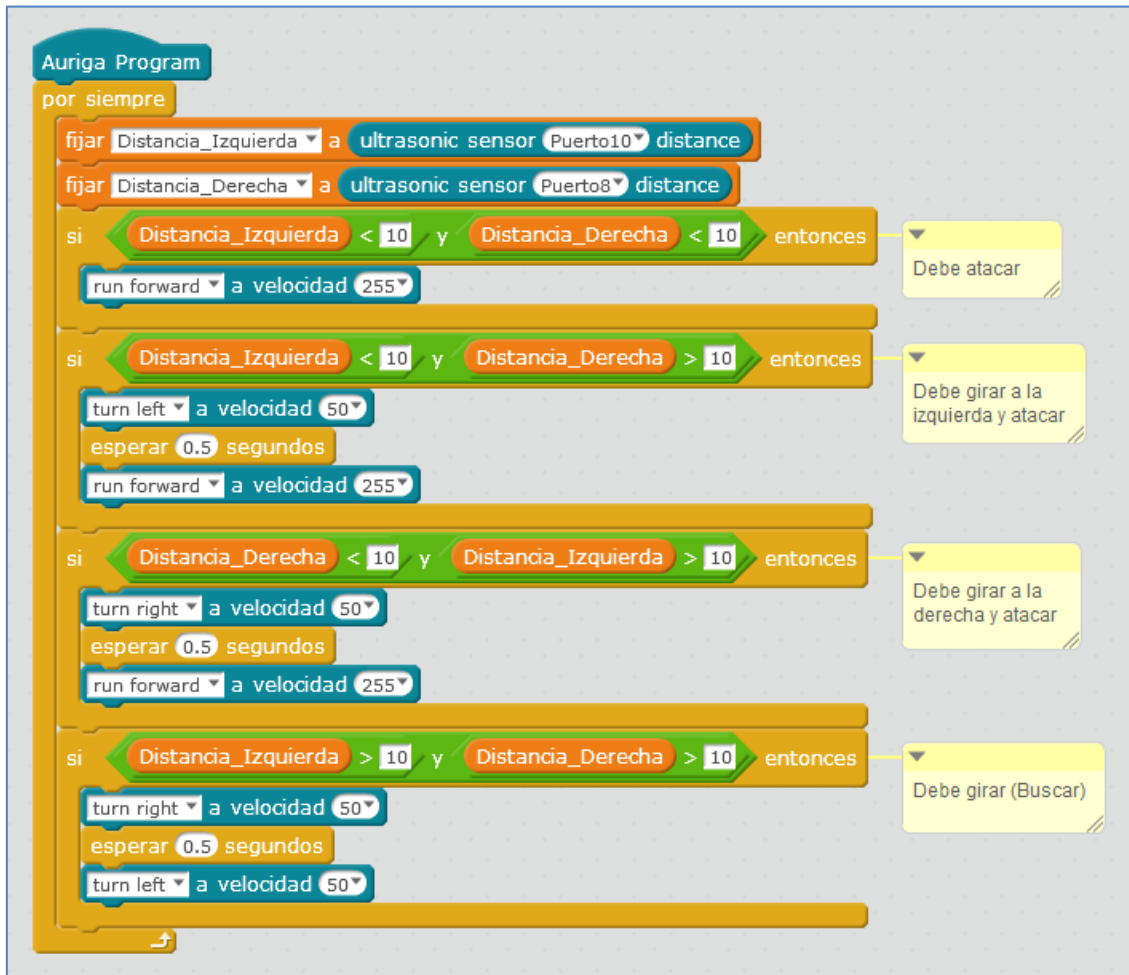
mBot Ranger con 2 sensores de ultrasonidos.



Divirtiéndome con mBot Ranger

Relativo a la distancia, en el sentido de cuál debemos usar como límite de detección, esta dependerá de la situación exacta de nuestros sen

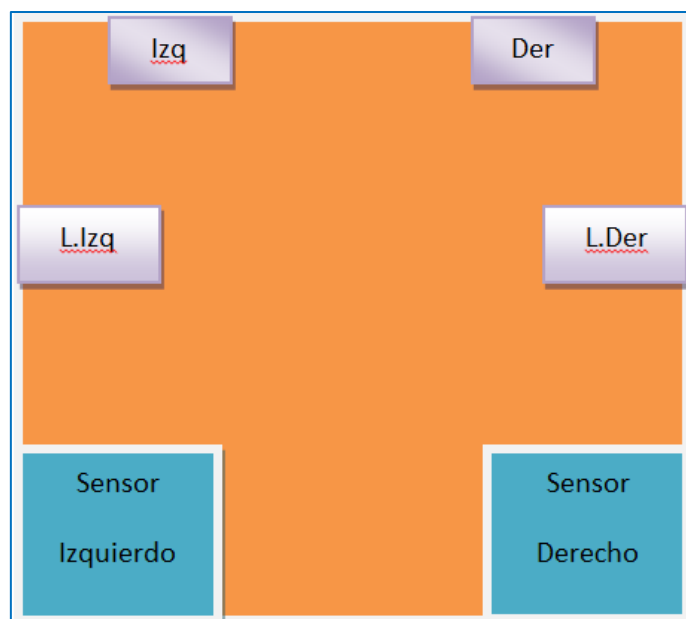
sores en el robot y de nuestros deseos en el programa. En este ejemplo, se ha decidido que ese umbral sea 10cm:



Primera parte del programa Sumo con dos ultrasonidos

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:

Divirtiéndome con mBot Ranger



Boceto Robot Sumo

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.

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

Divirtiéndome con mBot Ranger

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 Ranger, se verá esta defensa inicial.

Ejemplo 1 de programa sumo

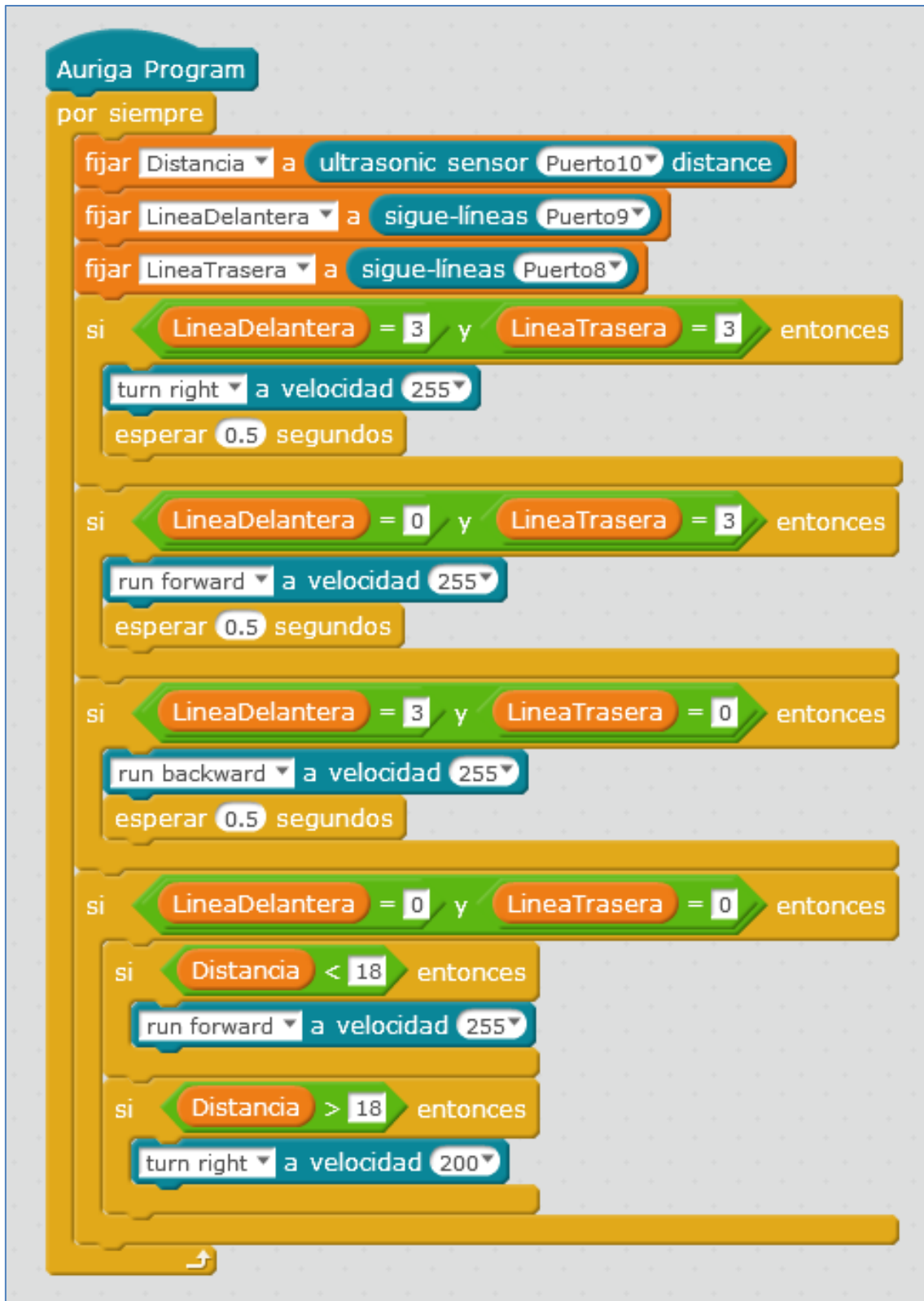
En este caso, usaremos un sensor de ultrasonidos para la parte delantera del robot y dos sensores de línea (uno delantero y otro trasero). El programa está pensado para luchar en un circuito limitado por una línea blanca.

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 *LineaTrasera* (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:

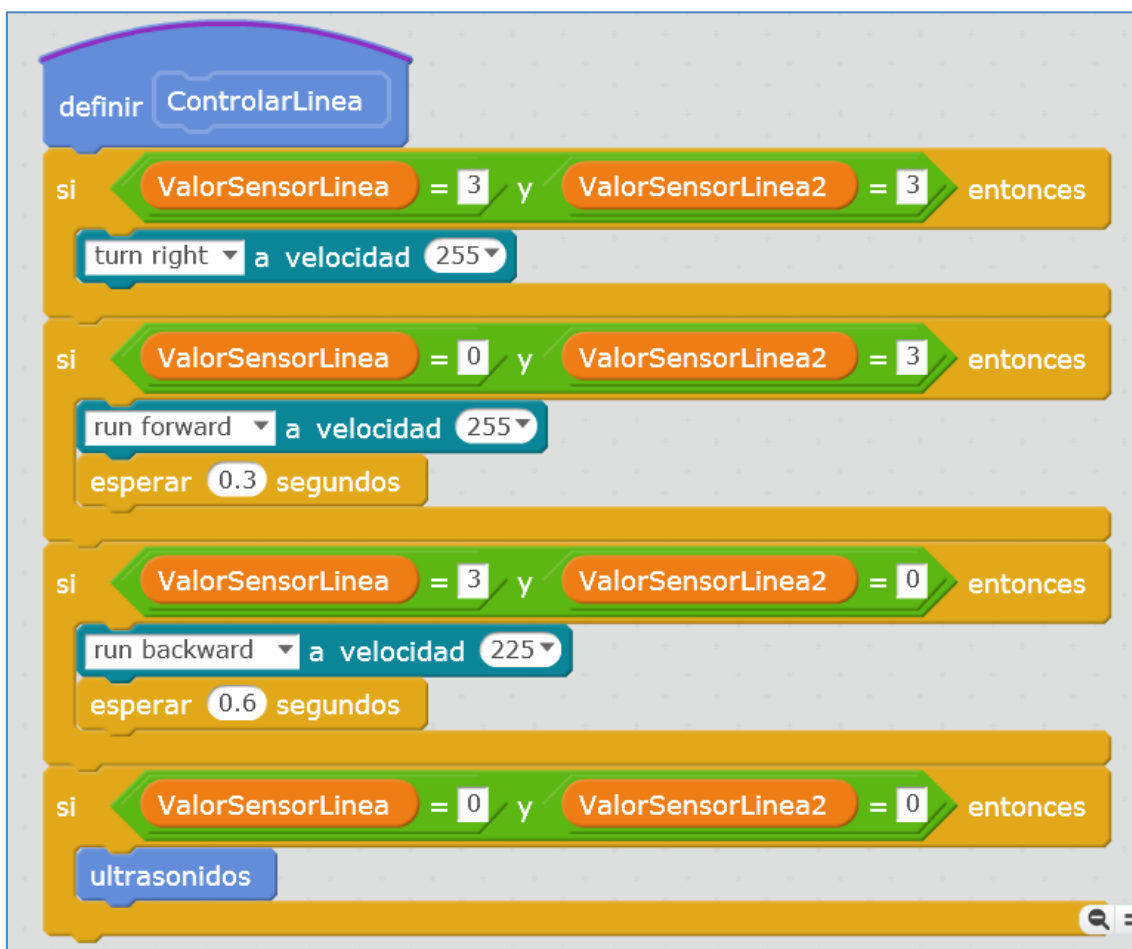
Supongamos ahora que queremos usar dos sensores de ultrasonidos y que los situamos, uno delante y otro detrás. Además, a nuestro robot le añadimos dos

Divirtiéndome con mBot Ranger

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):

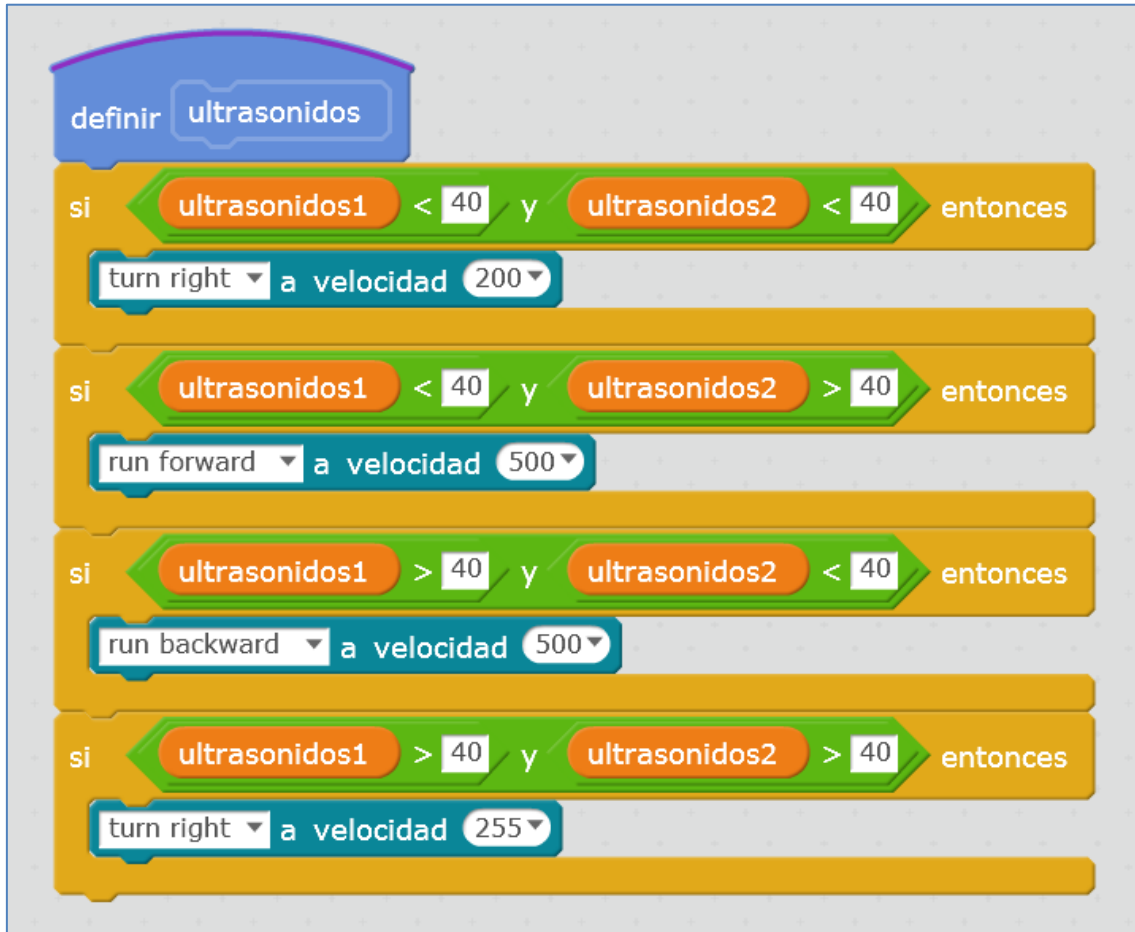


Si mi prioridad es la misma; estar en el recinto, comenzaré con esa línea de programación. Los sensores de línea, como ya vimos, presentan 4 posiciones y verán blanco en la posición 3. En este caso, debemos intentar que nuestro robot no salga del circuito de lucha. Por lo tanto: Si ve línea blanca el 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:



Divirtiéndome con mBot Ranger

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:

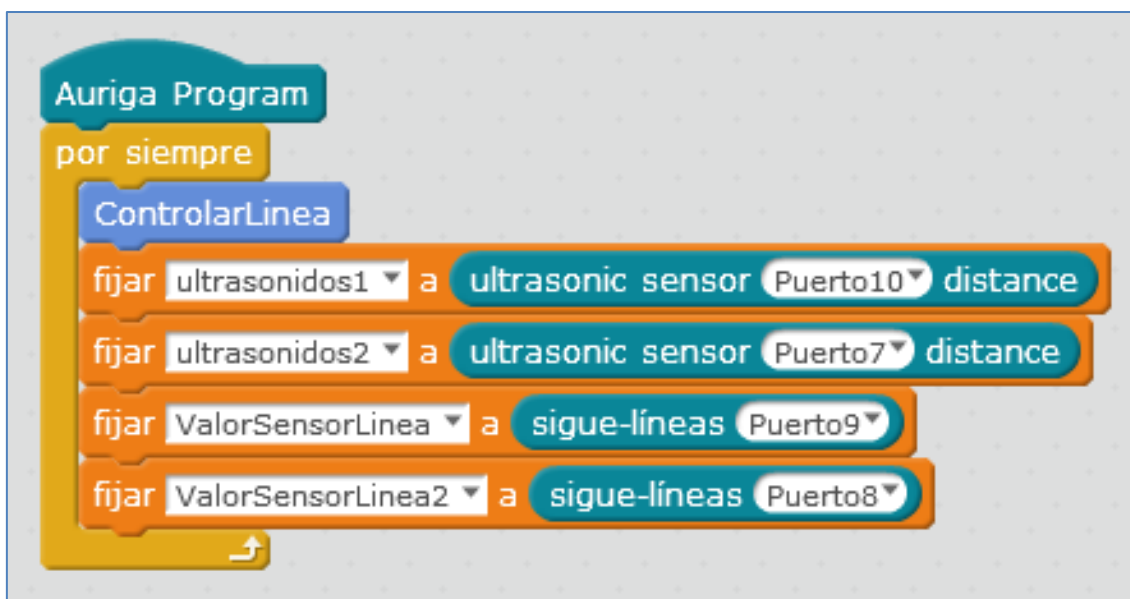


Bloque de ccontrol para los módulos de ultrasonidos

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

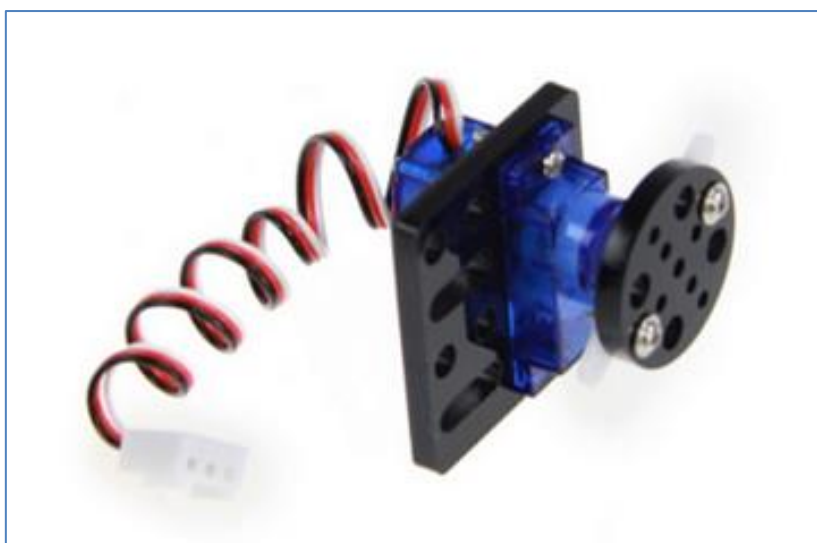
Hasta ahora, hemos definido los bloques, pero no el programa de ejecución que los llama. En él fijamos los puertos de conexión de los diferentes módulos. Como puede verse, los sensores de ultrasonidos, de ID amarillo, se han ligado a los puertos 7 y 10 y los siguelíneas, de ID azul, a los puertos 8 y 9.

El script de mando que ejecuta el sumo en el robot es el siguiente:



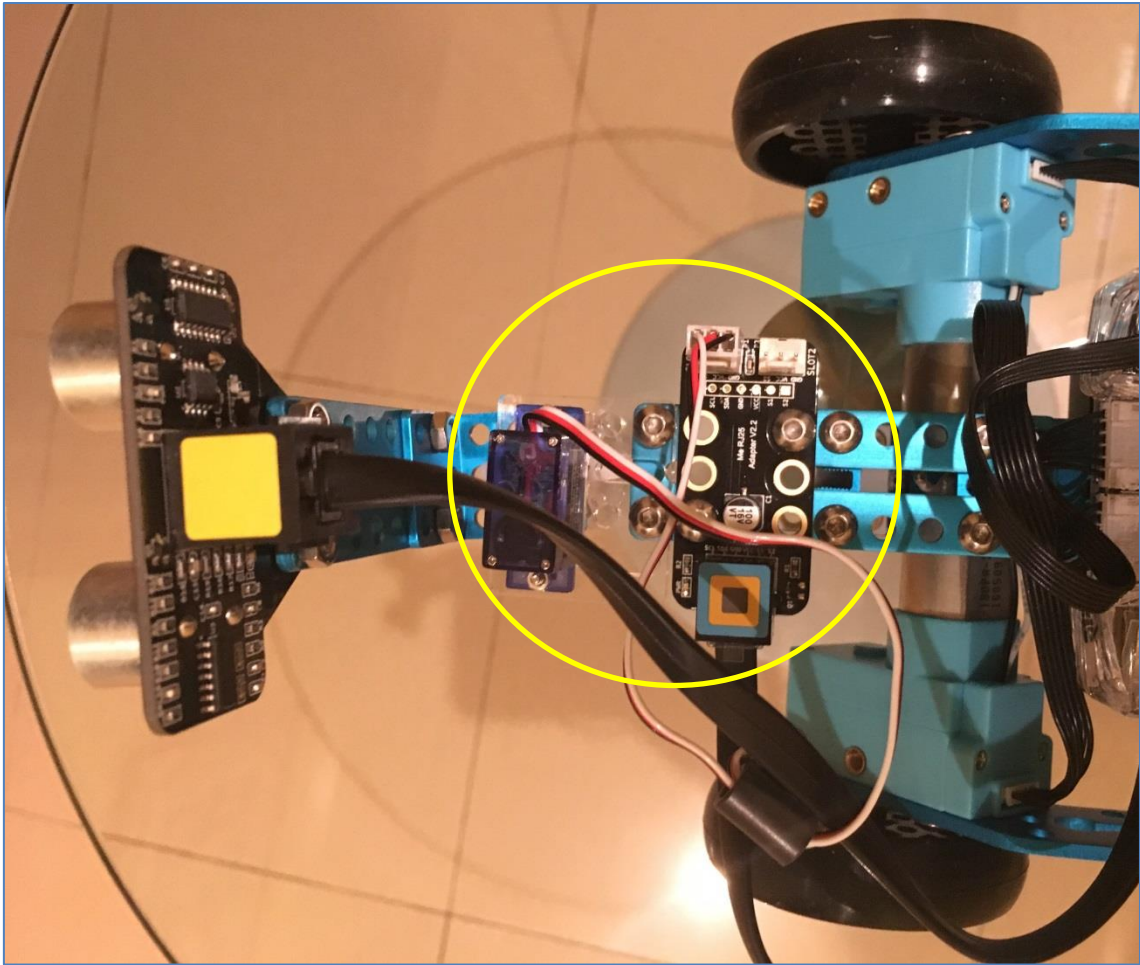
5.7. Servomotor

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.

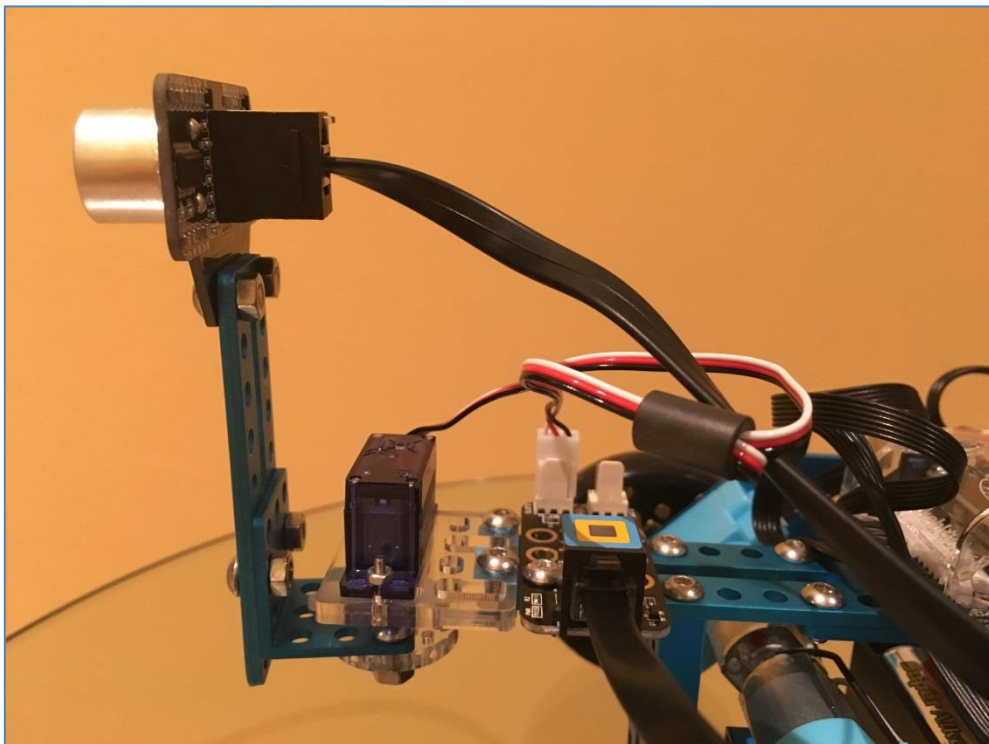


Micro Servo 9g (servomotor)

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: 6, 7, 8, 9 o 10, ya que sus colores ID son negro, amarillo y azul. En nuestro caso, lo hemos conectado al puerto 6:

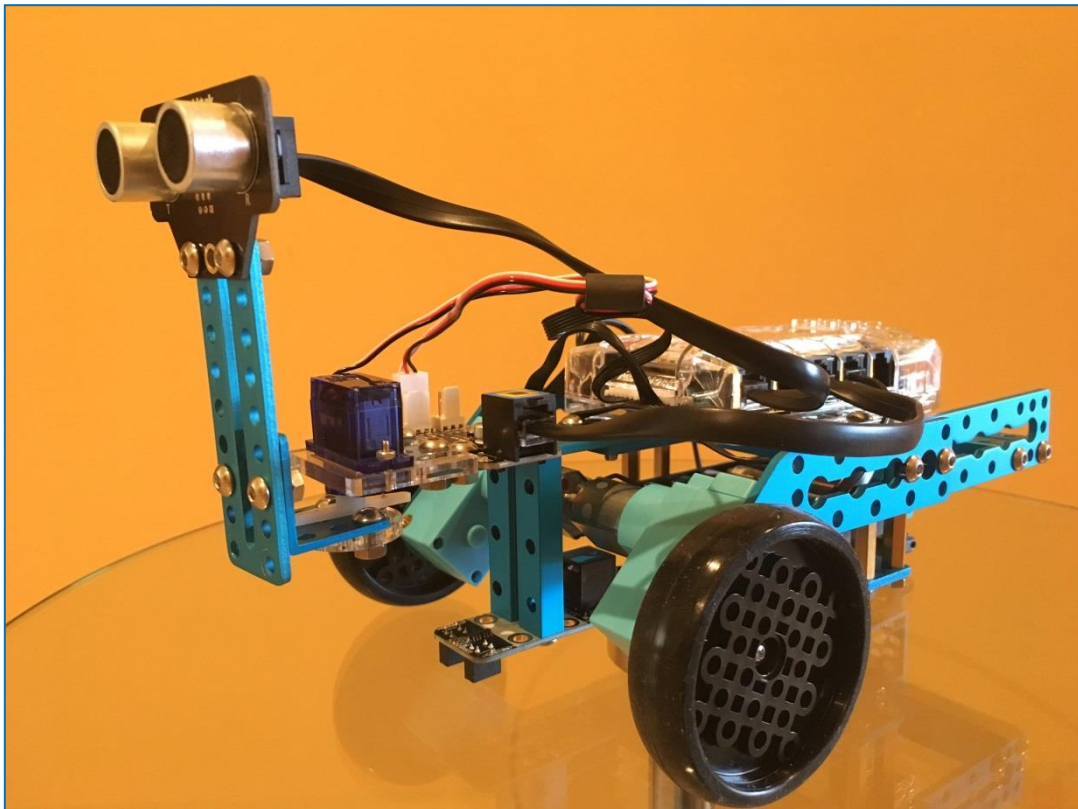


A mayores, en nuestro ejemplo, el servo se ha unido al sensor de ultrasonidos, de modo que, consiga moverlo en diferentes posiciones angulares del servo:



Divirtiéndome con mBot Ranger

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.

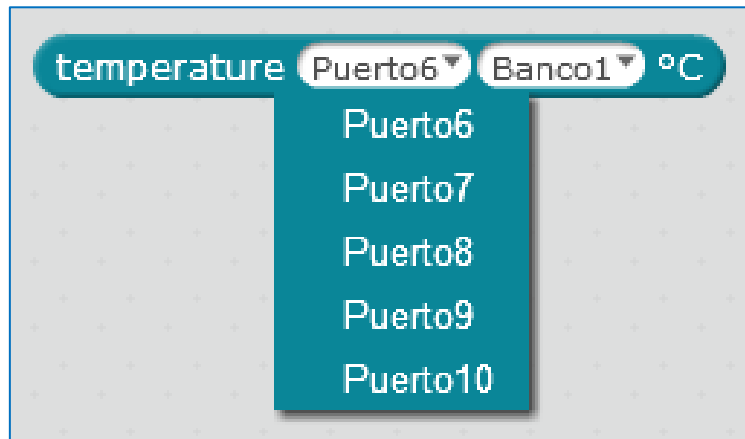


Divirtiéndome con mBot Ranger

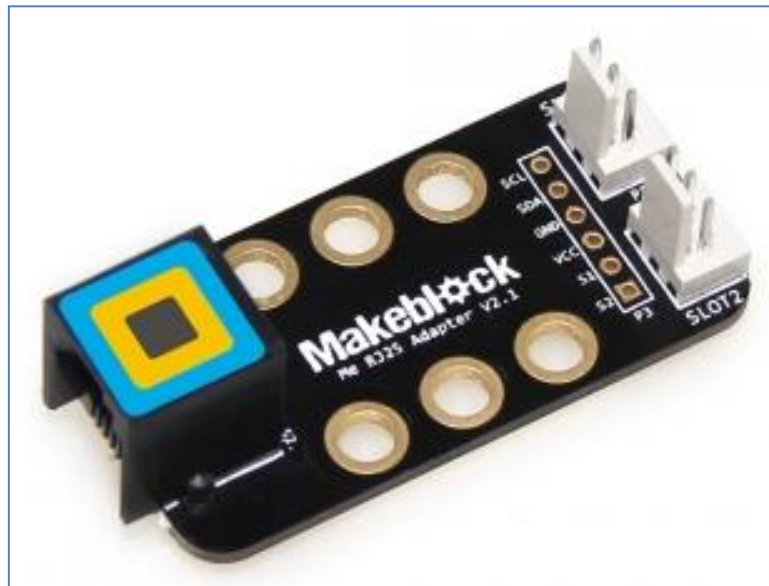
5.8. Sensor de temperatura SD18B20

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 Ranger, necesita de un módulo adaptador RJ25 como el que se muestra en la imagen y que ya hemos utilizado con el módulo servomotor:



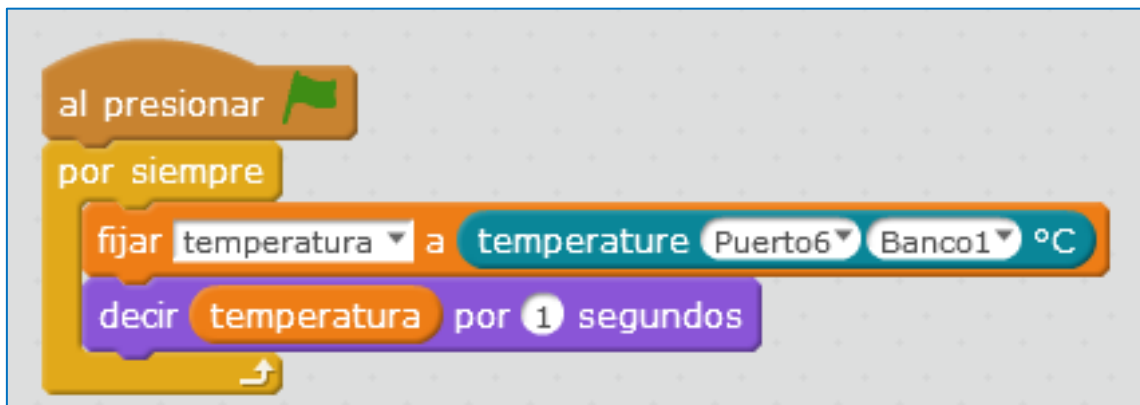
Módulo adaptador RJ25

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

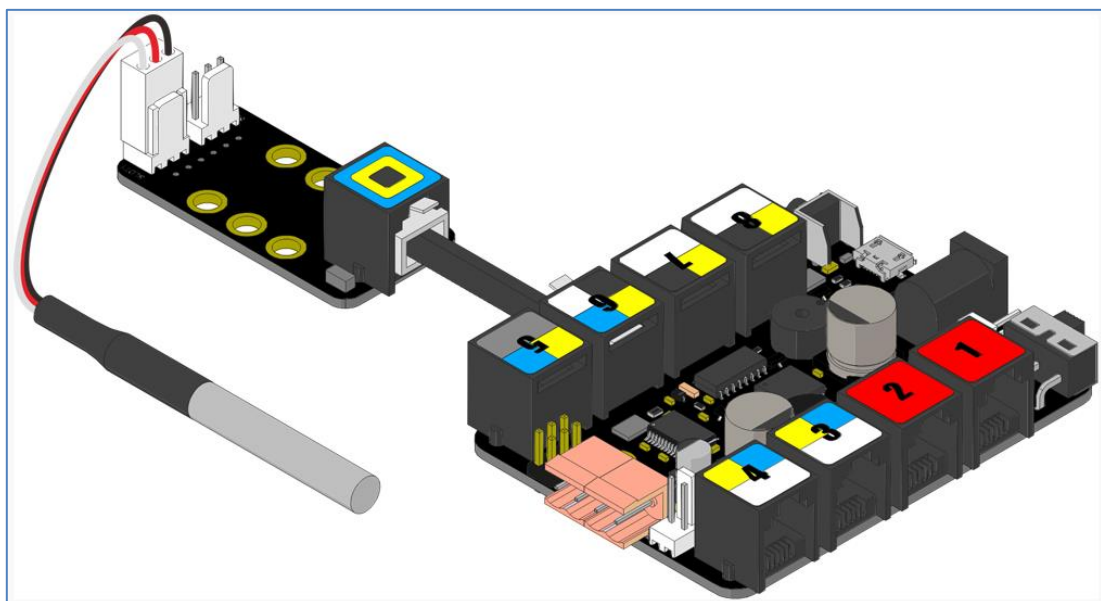


Módulo sensor de temperatura sumergible

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 6*, sería el siguiente:



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



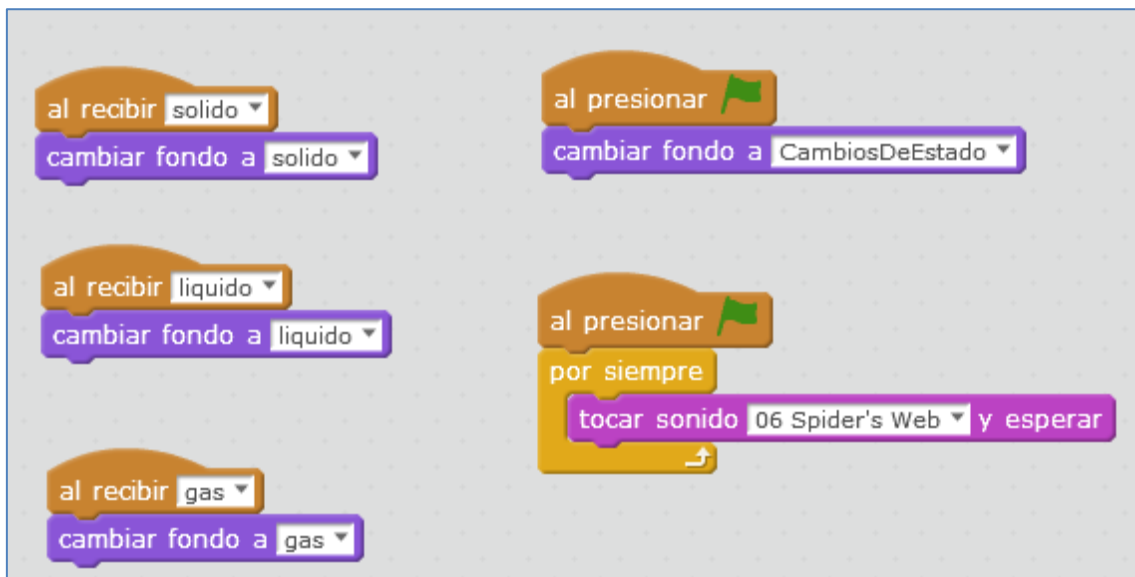
Divirtiéndome con mBot Ranger

En el siguiente ejemplo se ha conectado a la placa Me Auriga en el *banco 2* del *puerto 6*. Se ha creado un programa para que el sensor mida la temperatura de una sustancia que testeo 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:

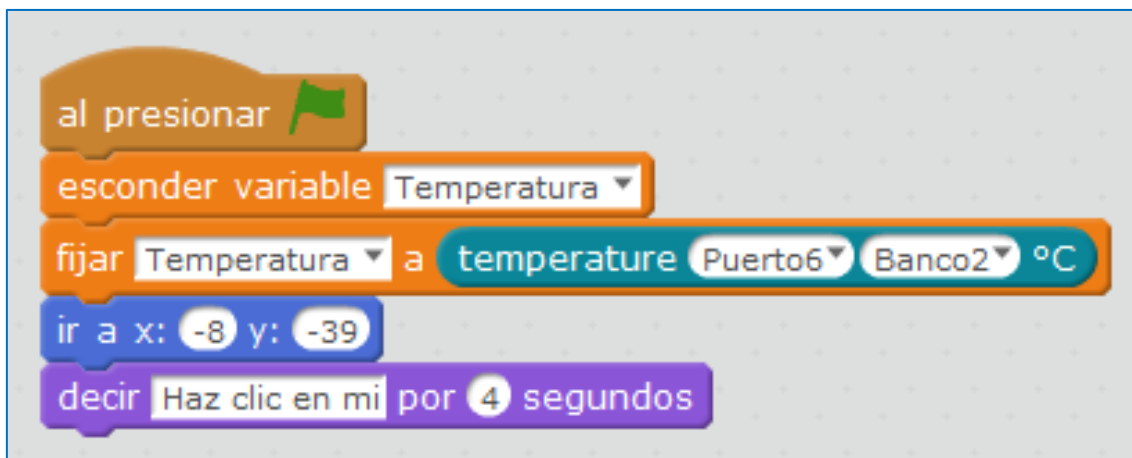


Script del escenario

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

Divirtiéndome con mBot Ranger

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 6* de la placa del mBot Ranger, pero si quiero que tome la primera muestra para que sus medidas se vayan regulando:



Sensor de temperatura conectado al Puerto 6

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:



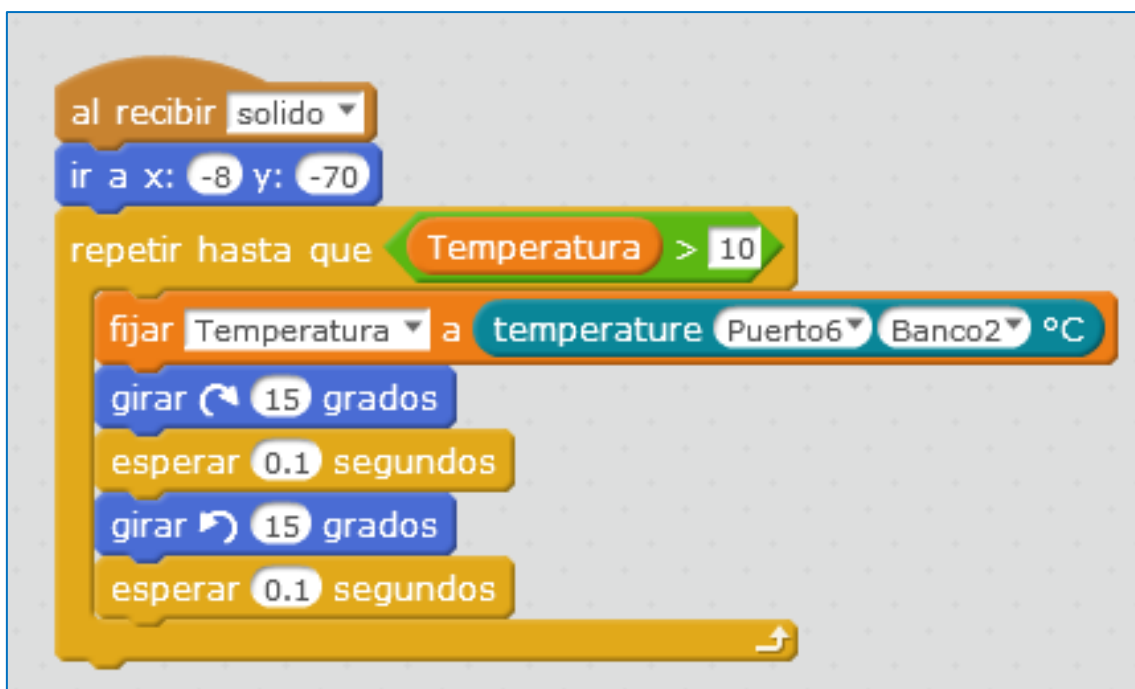
Divirtiéndome con mBot Ranger

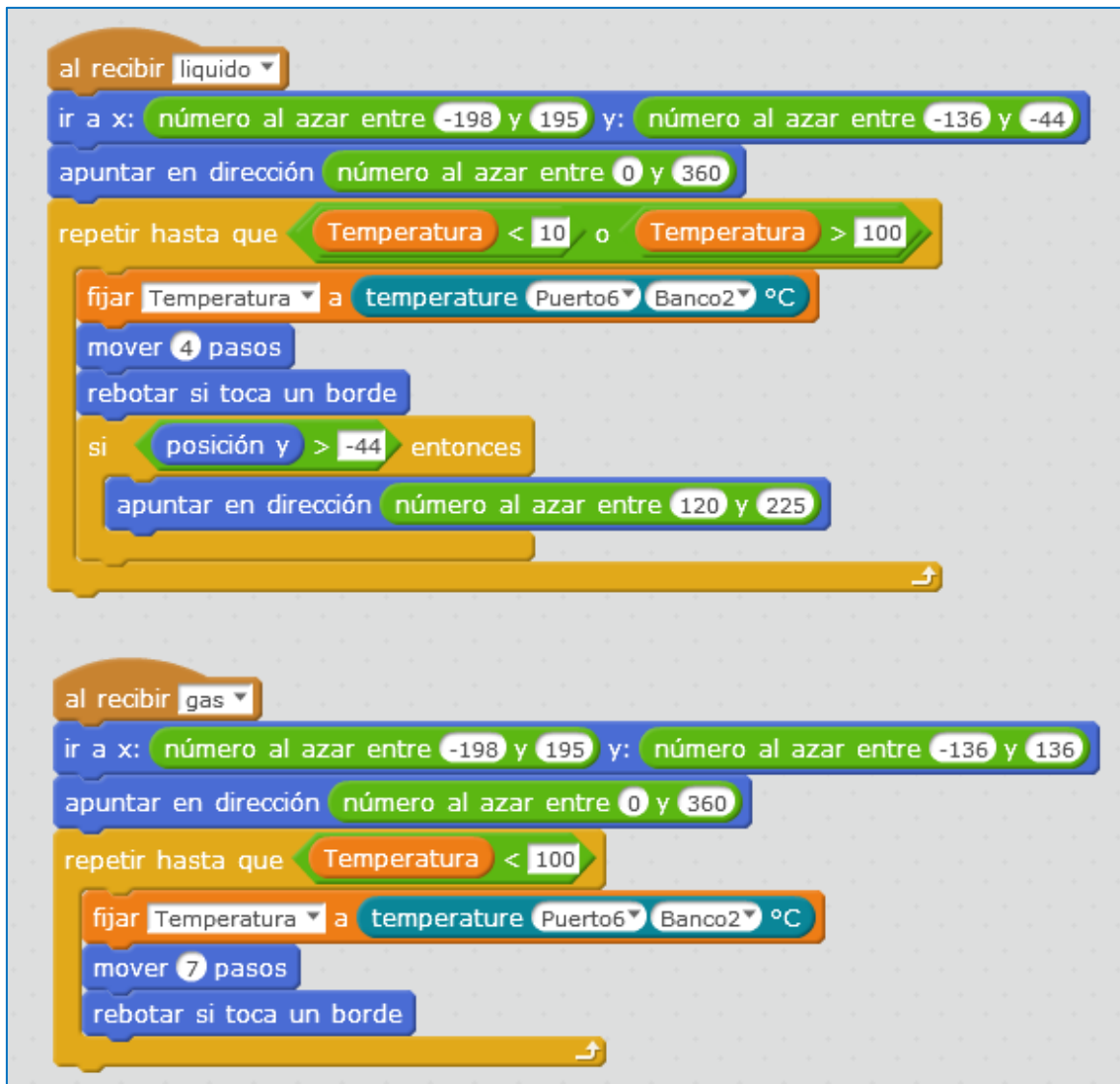
Los mensajes “sólido, líquido y gas” realizan la simulación en el movimiento de la molécula.



Situación “Líquido”

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 “*solido*” simula que la molécula vibra, moviéndose 15° a la derecha e izquierda. La animación “*liquido*”, hace que la molécula se mueva a una determinada velocidad (4 pasos) en la zona del líquido (sin sobrepasar 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.

5.9. Sensor de Me Temperatura y Me Humedad

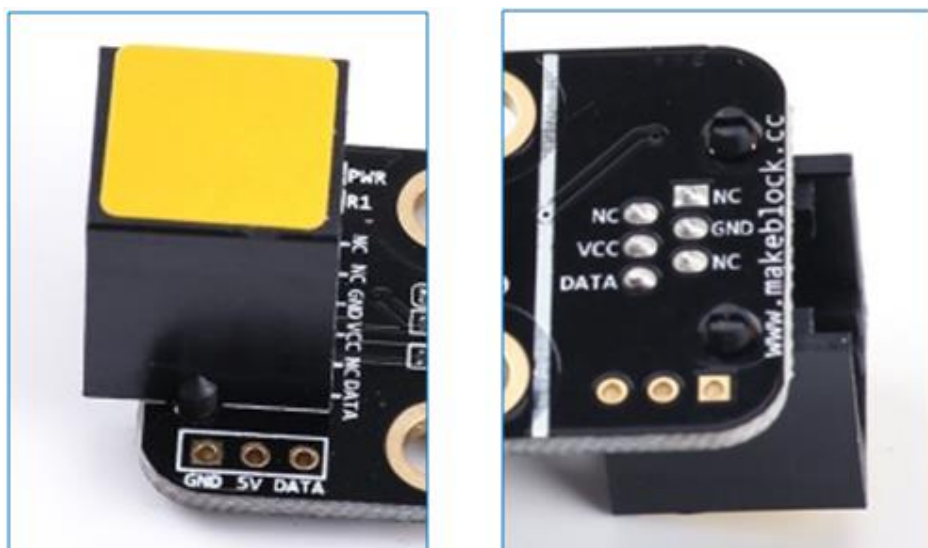
La casa Makeblock dispone del sensor DHT11, que es capaz de medir la temperatura (NTC) y humedad (resistiva) ambiente. Sus especificaciones son las siguientes:

- 5V DC
- Rango temperatura: 0°C a 50°C ± 2°C (sólo Tª 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 en sus diferentes modelos³ a través de sus tres conexiones GND, 5V y Data, siendo Data un pin digital como por ejemplo el pin3.

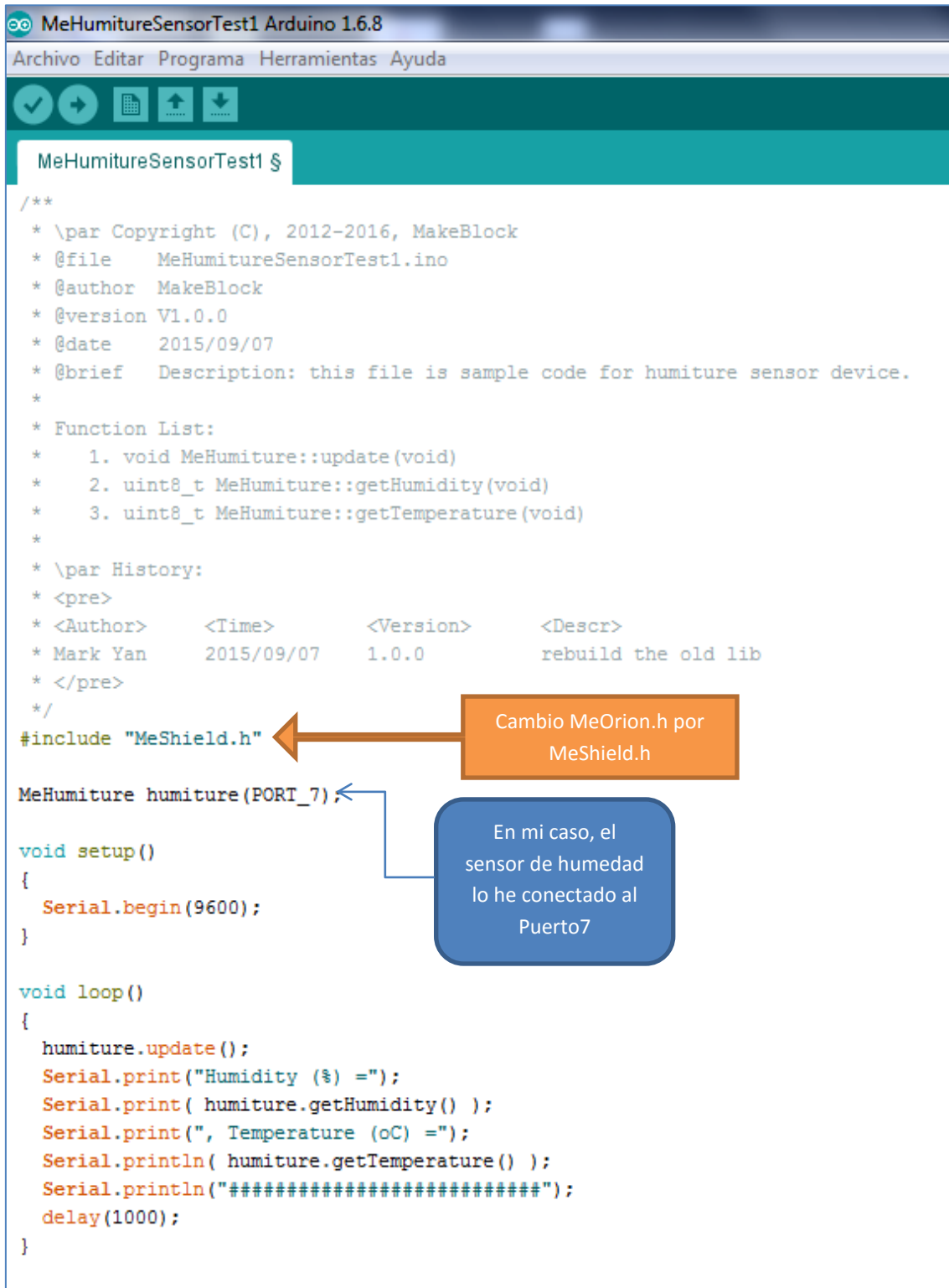


En el siguiente ejemplo voy a usar el shield de Makeblock para una Arduino Uno, conectando el sensor de humedad y temperatura al Puerto 7. En las librerías que nos proporciona la casa Makeblock disponemos de 2 archivos para el sensor de humedad. Usaré el *test 1*, que es el que nos proporciona la humedad y temperatura ambiente en °C (el *test2* nos lo aporta en diferentes magnitudes para la temperatura). En el *test 1*

³ Arduino Uno, Arduino Nano, Arduino Leonardo y Arduino Mega

Divirtiéndome con mBot Ranger

realizo unas pequeñas modificaciones relacionadas con la placa y el puerto de conexión del módulo sensor en la misma:



```
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 humiture 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"

MeHumiture humiture(PORT_7);

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

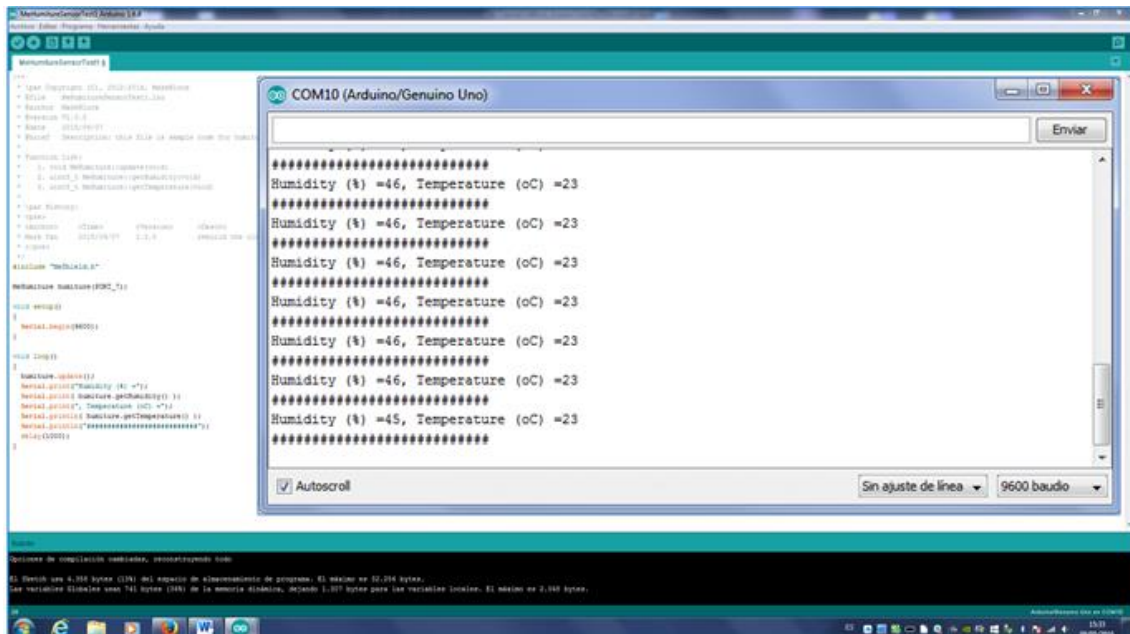
void loop()
{
  humiture.update();
  Serial.print("Humidity (%) =");
  Serial.print( humiture.getHumidity() );
  Serial.print(", Temperature (oC) =");
  Serial.println( humiture.getTemperature() );
  Serial.println("#####");
  delay(1000);
}
```

Cambio MeOrion.h por MeShield.h

En mi caso, el sensor de humedad lo he conectado al Puerto7

En la siguiente imagen, referida al *Monitor Serie*, podemos leer la temperatura y la humedad ambiente que nos aporta en cada segundo:

Divirtiéndome con mBot Ranger

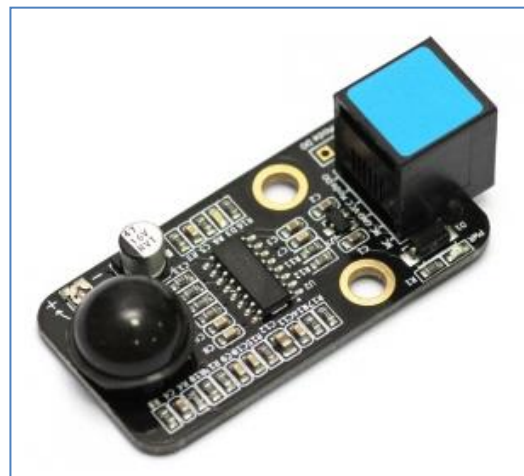


5.10. Módulo PIR

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 90° y su radio de alcance se sitúa entre 6 y 9 metros.

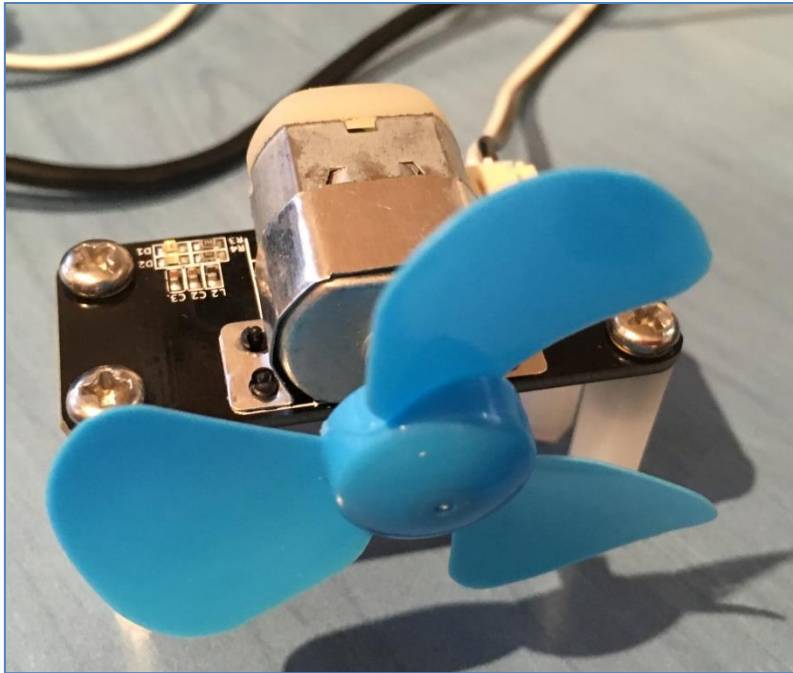
El sensor PIR de movimiento de la casa Makeblock se usa para detectar personas o animales en un rango de hasta 6m. Si se mueve algo dentro de ese rango de distancia, el sensor activa la salida digital SIG a alto. Mediante un potenciómetro, soldado en el módulo, podremos ajustar el rango de detección.

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



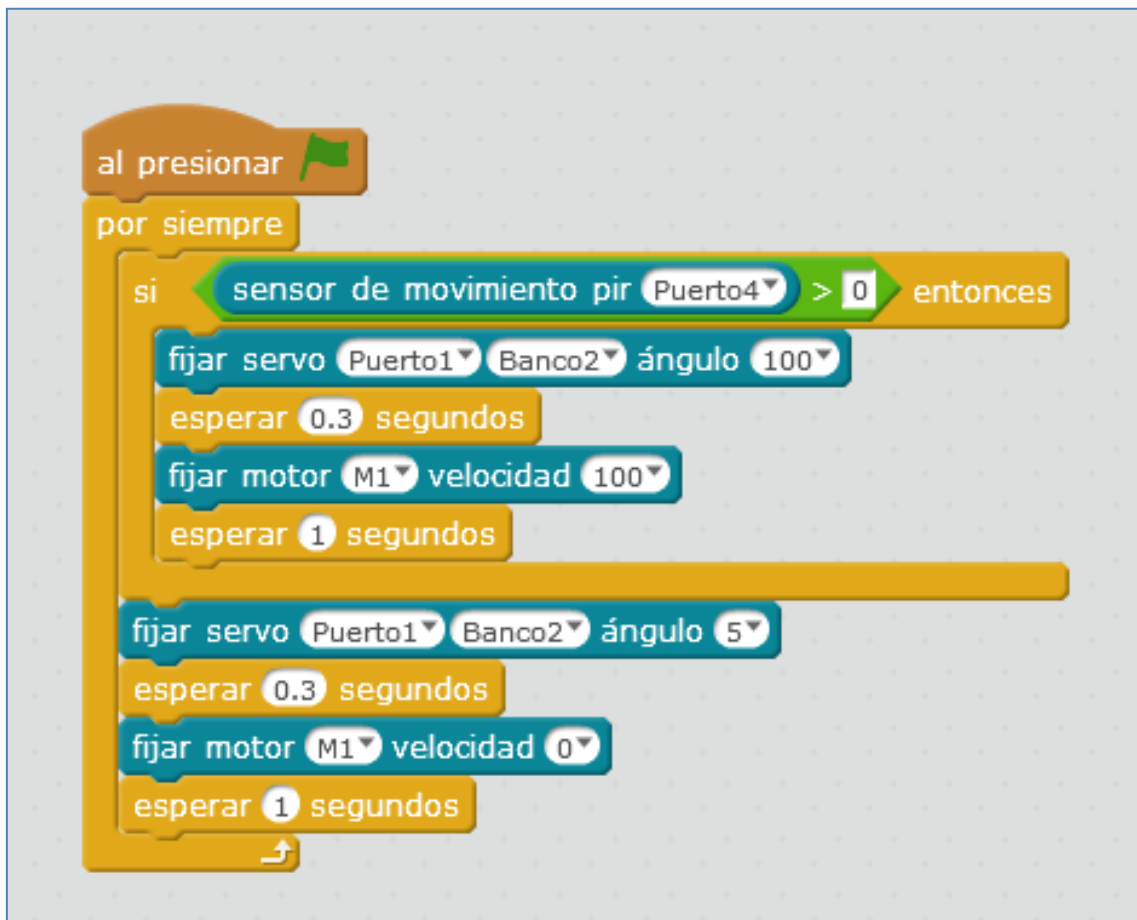
Sensor Me PIR

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.



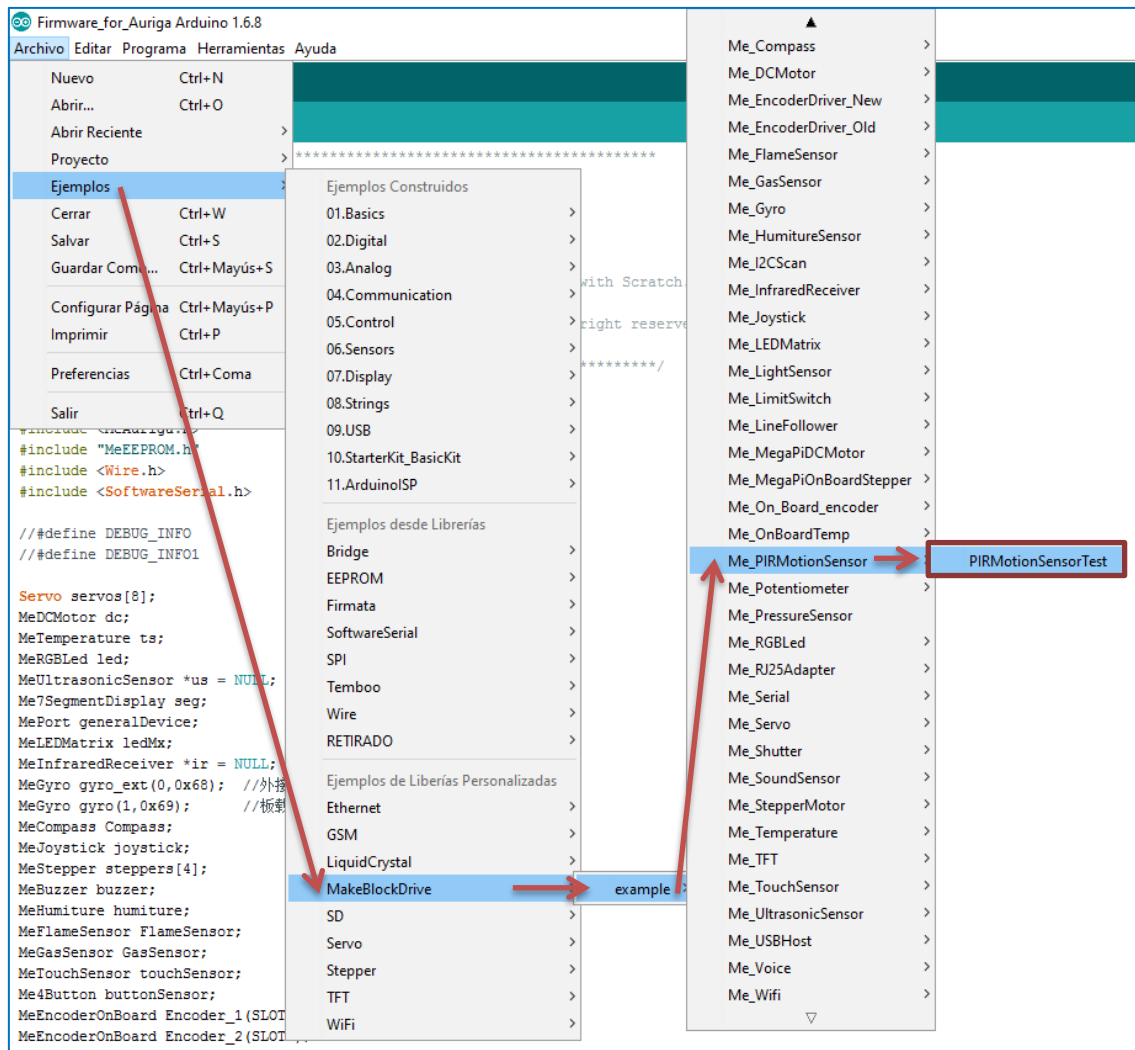
Módulo 130 DC con hélice

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

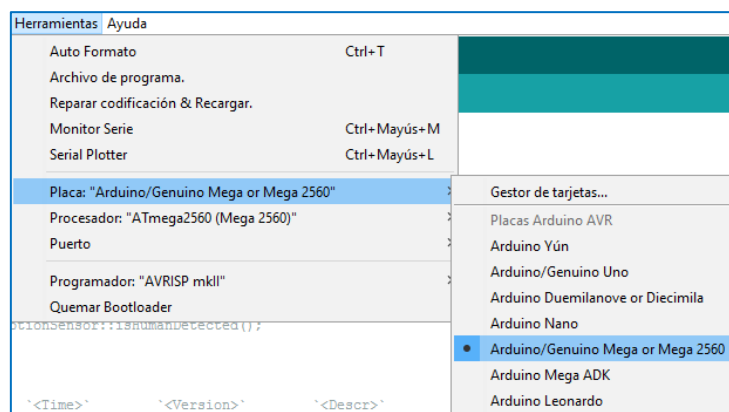


Divirtiéndome con mBot Ranger

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 Me Auriga conectando el sensor PIR al Puerto 6 (ID azul) de la misma. Después cargaré el programa a la placa desde el IDE de Arduino, seleccionando la placa *Arduino/Genuino Mega or Mega 2560*.



El resultado es obvio, habrá detección porque yo estoy presente, tal y como se muestra en la siguiente imagen:

El módulo joystick 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 joystick.



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 para las funciones 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 joystick que está conectado al puerto 7 que consiga que el sprite se mueva por el escenario, es el siguiente:



5.12. 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

Divirtiéndome con mBot Ranger

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.



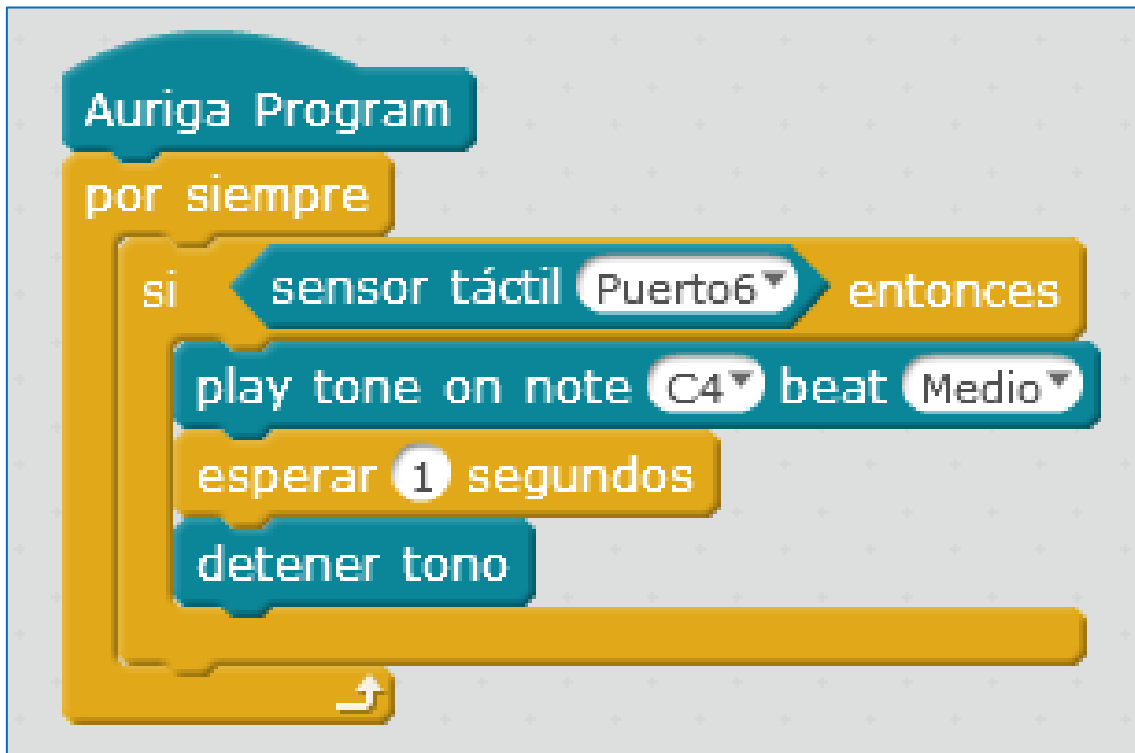
Sensor Me Touch

En cuanto a su programación, es importante saber que el sensor debe ejecutar el programa cargándolo a la placa al mBot o del Ranger. Por lo tanto, no podemos usar “Al presionar la bandera verde”. En su lugar, usaremos el comando del robot [Auriga Program](#). Tras cargar el programa en el Ranger desconectamos el cable USB y encendemos el robot. Veremos que comienza a ejecutarlo.

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



También podría sonar la nota C4 durante 1 segundo:



5.13. Garras: Mini Gripper y Gripped

Podemos implementar y programar una garra robótica para los robots de la casa Makeblock. La que vemos en el siguiente link está construida con piezas de LEGO y fue diseñada para el robot mBot:

[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 que podemos usar con el Ranger 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, como la placa Orion o la placa Me Auriga, 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 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. Otra opción sería usar la garra robótica en la placa en una de las conexiones de los motores del mBot.



Garra robótica para Starter Robot Kit

En cambio, la garra robótica de la siguiente figura si puede acoplarse al mBot, por potencia (sin complicarnos con los motores de sus ruedas) 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 robot (Orion, mCore o Auriga).



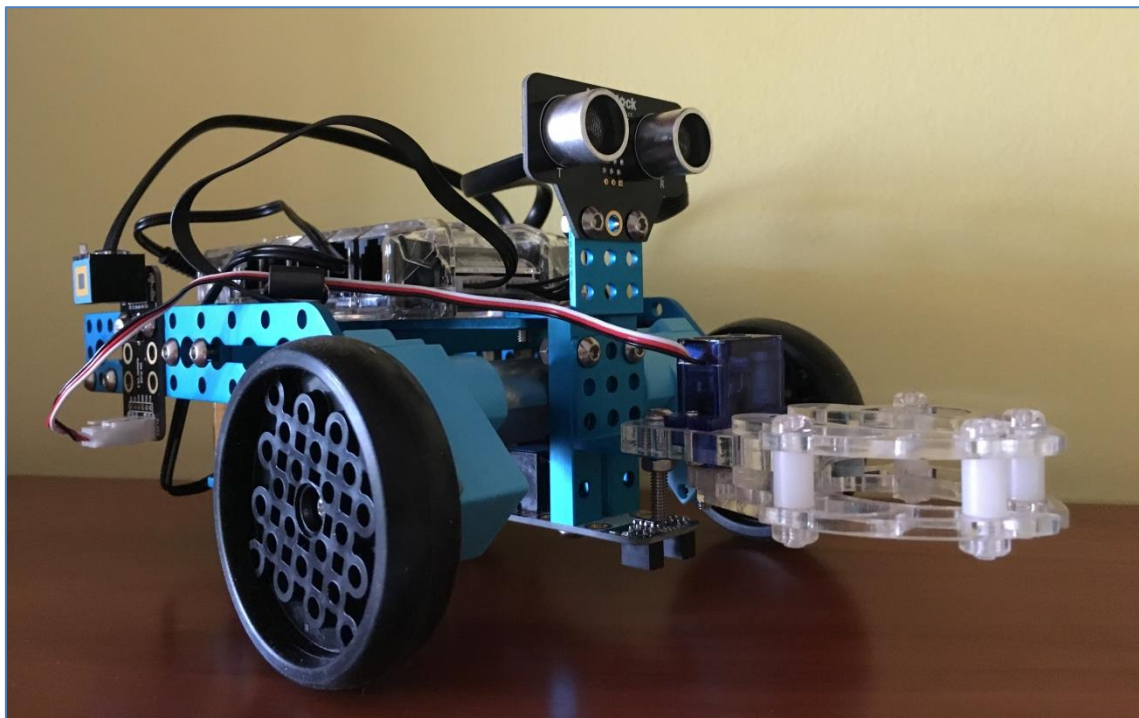
Mini Gripper o garra robótica para mBot

Ambas, garra robótica o Gripped y Mini Gripped, pueden adaptarse y programarse en el robot mBot Ranger.

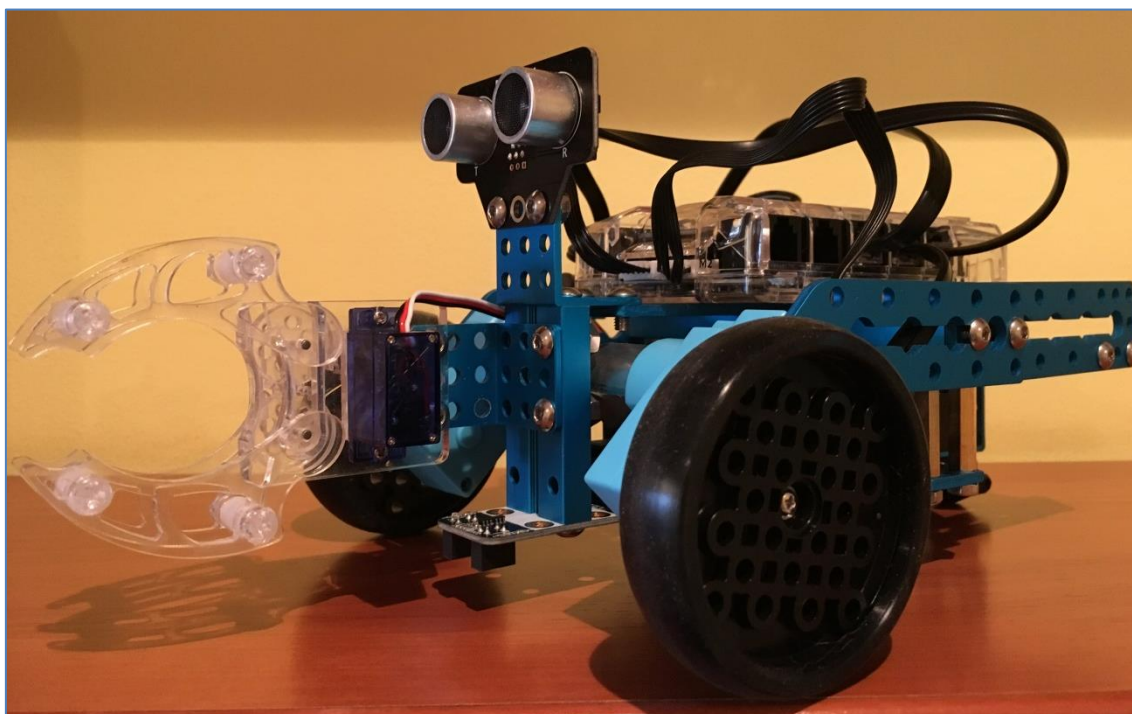
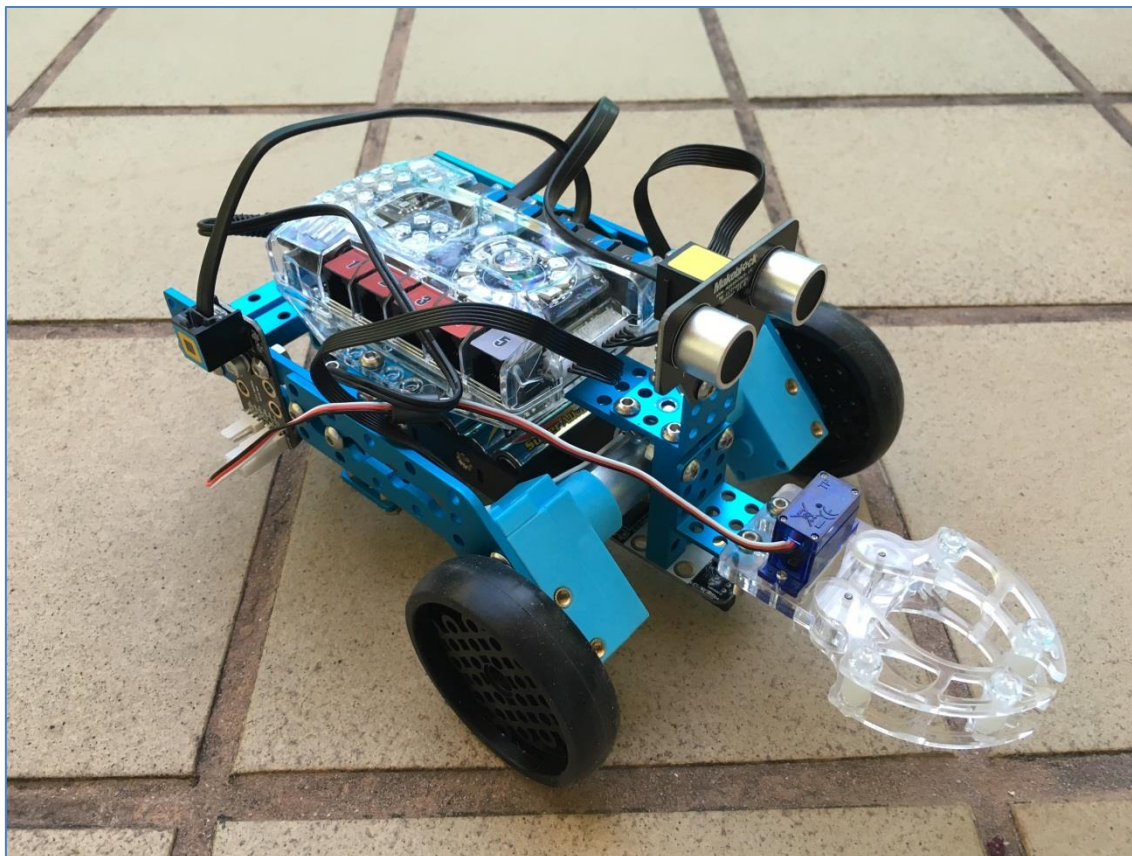
Divirtiéndome con mBot Ranger

Mini Gripped

Para la placa Me Auriga: 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 0° y para cerrarla, le indicaremos que lo gire a 90°. Los ángulos posibles del servo varían entre 0 y 180°:



Mini Gripped en el Ranger

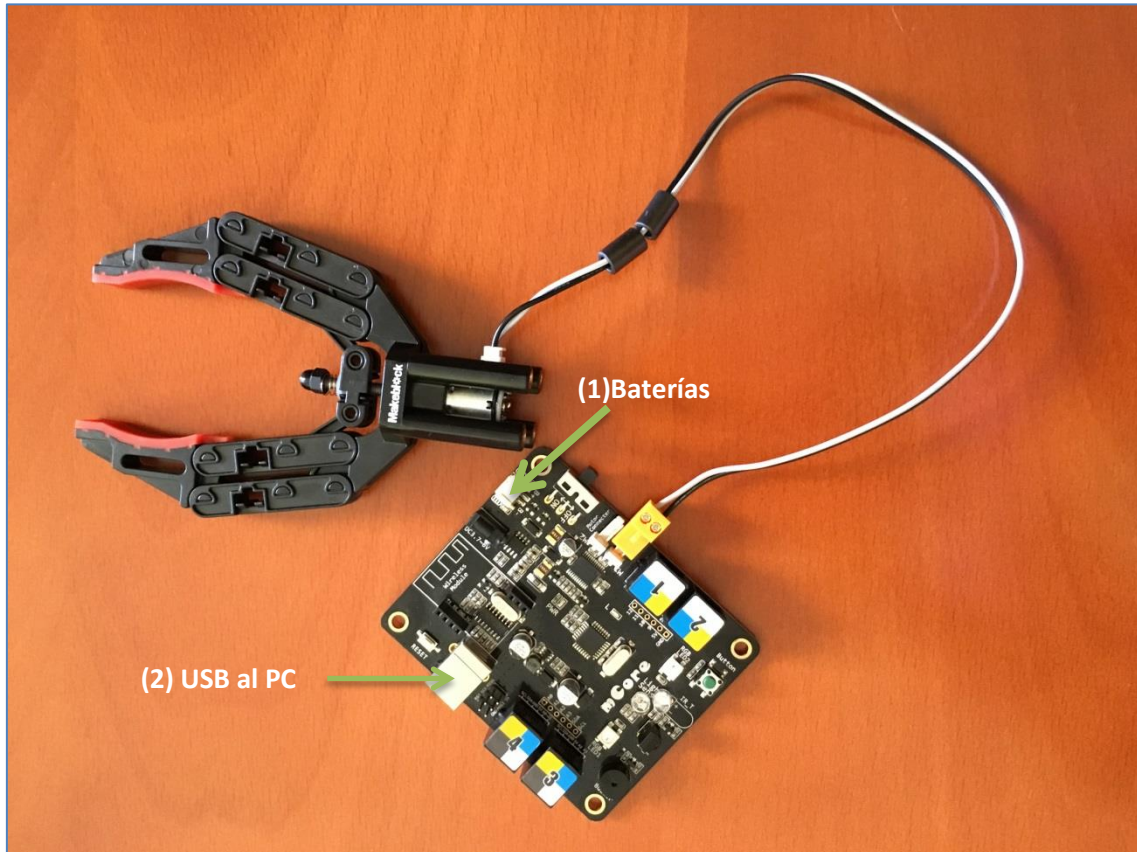


Ejemplos de montaje Mini Gripped

Divirtiéndome con mBot Ranger

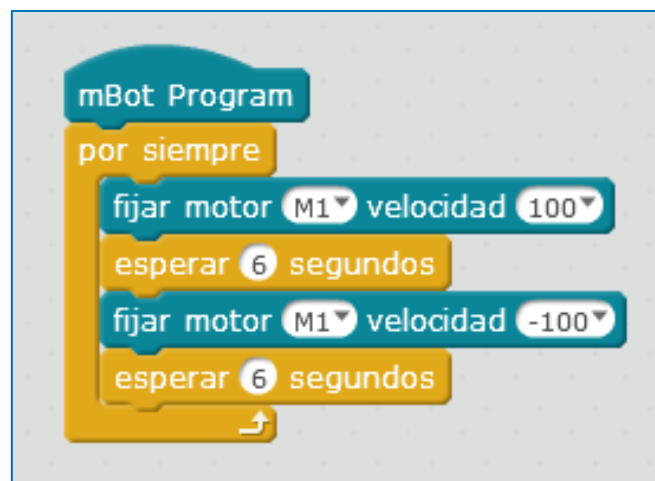
Gripped o garra robótica

Aunque no fue pensada para la placa mCore, podemos acoplársela a ella de forma muy sencilla si desconectamos los motores del robot mBot. En la siguiente imagen se ha conectado al motor M1 de la placa mCore. En ella vemos que nos faltaría alimentar la placa, bien sea por baterías (1) o por el puerto USB tipo B (2) al PC:



Gripped con mCore

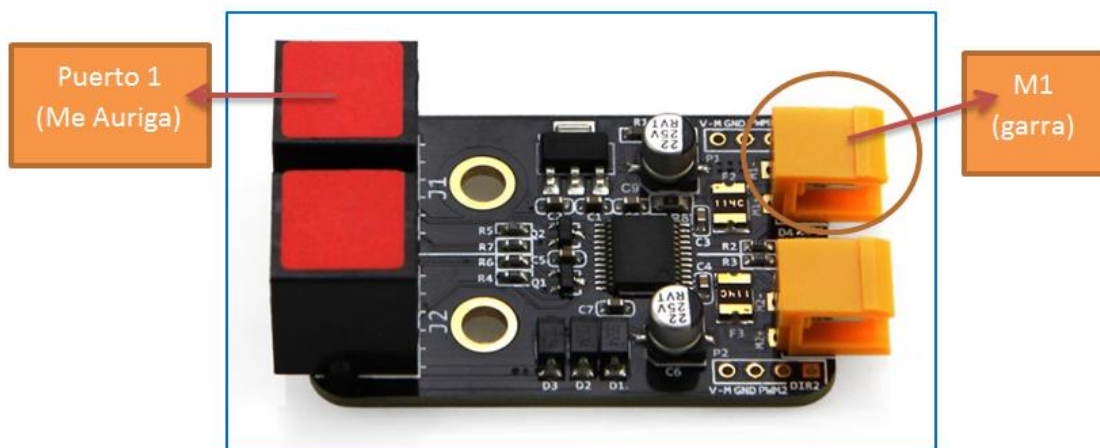
El ejemplo más simple de control de la garra robótica consistiría en hacer que se abriera y que se cerrara. El script que nos permite realizar este deseo es el siguiente (obviamente, debemos asegurarnos que el interruptor de la placa esté accionado):



Divirtiéndome con mBot Ranger

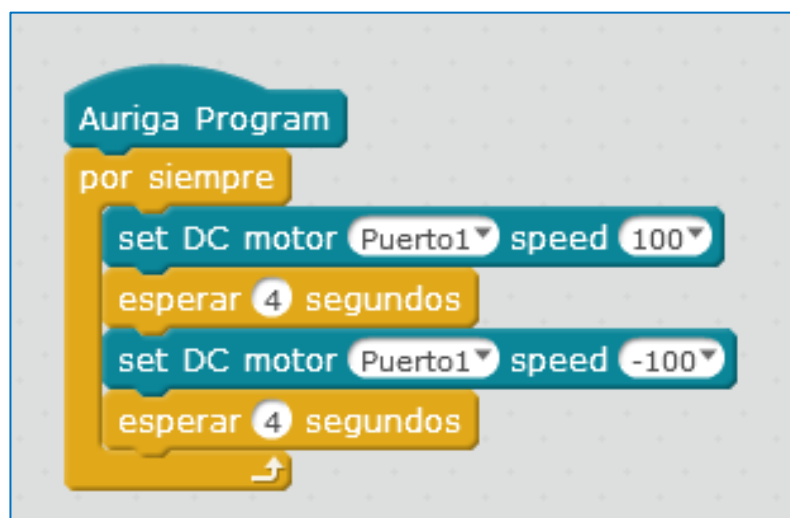
Relativo a la placa Auriga, es importante conocer lo siguiente. En esta placa, una vez conectada al mBlock, vemos que tanto los sensores como los LEDs funcionan sin problema, pero, a veces da la impresión que los motores (tanto los propios como los que podemos conectar a los puertos con ID rojo) no funcionan. El motivo de esta aparente “anomalía” es simple: la placa Auriga necesita estar encendida para poder alimentar a los motores. Por lo tanto, debemos asegurarnos que la luz “PWR” esté encendida para poder usar los motores. Esta luz se enciende pulsando cualquiera de los botones ON/OFF de la placa (la placa se apaga presionando el botón de forma interrumpida durante 5 segundos).

Supongamos que queremos conectar nuestra garra al Puerto 1 de la placa Me Auriga. Necesitaremos utilizar un controlador de motores de continua (ver punto 5.29 de este documento). He decidido usar el controlador de motores DC dual, que nos serviría para dos motores. Como la garra robótica sólo dispone de un motor, he elegido usar el puerto J1 del controlador de motores, de modo que, un cable RJ25 vaya desde el Puerto1 de la placa Auriga al puerto J1 del controlador. Usar J1 implica, necesariamente, que la garra robótica se conecte a M1, tal y como podemos ver en la siguiente imagen:



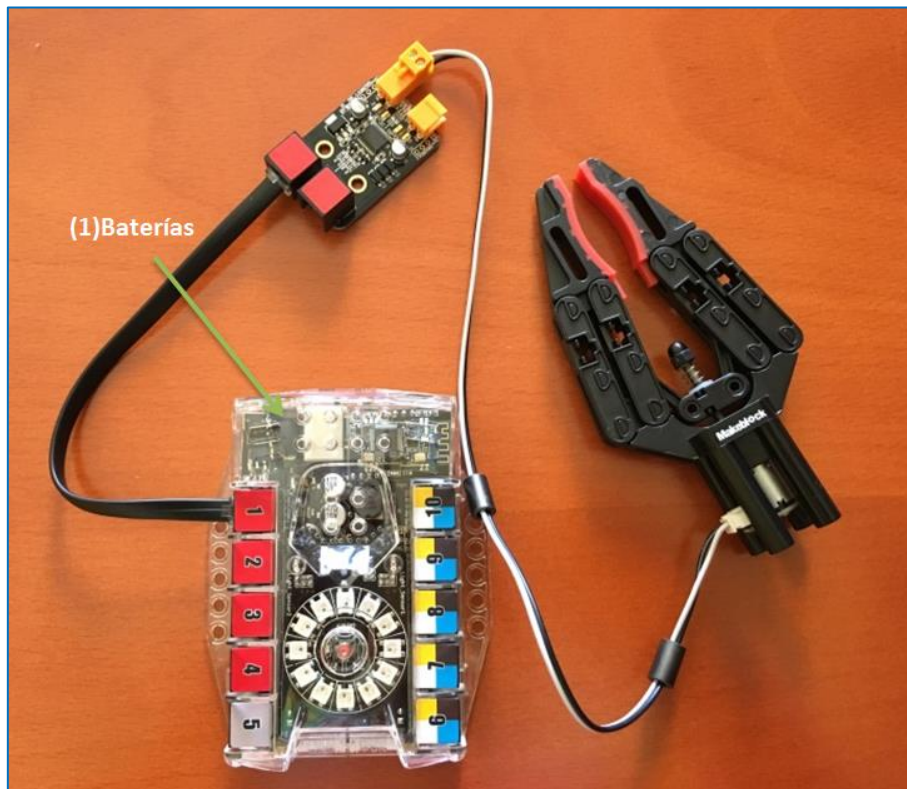
Controladora de motores DC Dual

Sólo resta programarla. Vamos a desarrollar el mismo ejemplo, visto en la placa mCore: conseguir que la garra robótica se abra y se cierre, de forma continua. El script que nos permite realizar este deseo, cargándolo a la placa Auriga y conectando después las baterías, es el siguiente:



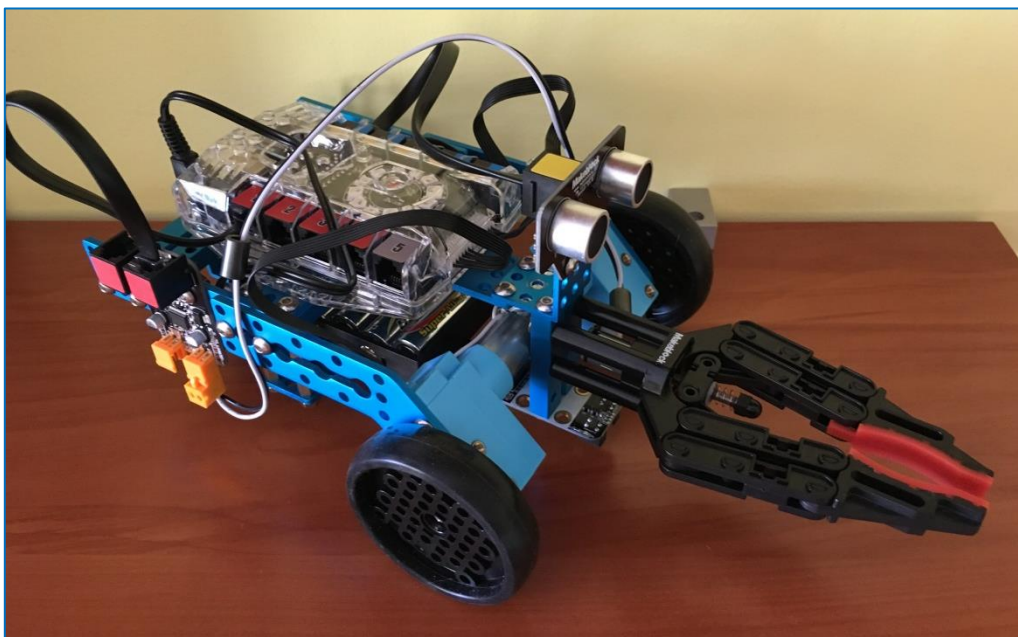
Divirtiéndome con mBot Ranger

El montaje de la garra robótica quedaría como se observa en la siguiente imagen:



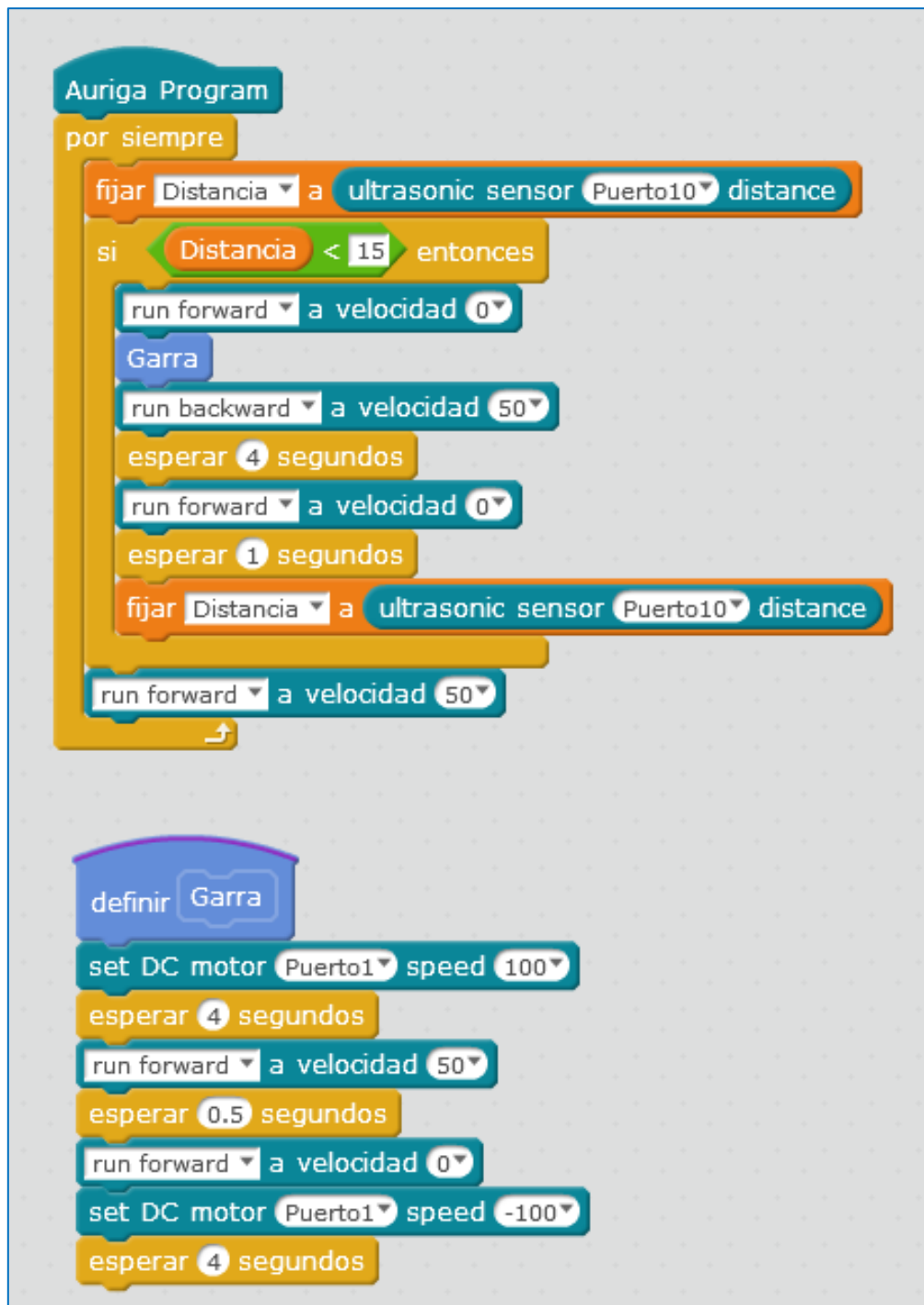
Gripped con Auriga

Un bonito reto consistiría en lograr combinar la garra robótica con el sensor de ultrasonidos, de modo que, por ejemplo, cuando al moverse detecte de forma frontal un objeto, se pare, abra la garra, se mueva hasta que el objeto penetre en la abertura de la garra, despues se cierre y finalmente, se dirija al lugar de partida. Como montaje físico, de nuestro robot, para llevar a cabo este reto, podría valernos el que se ve en esta imagen:



Divirtiéndome con mBot Ranger

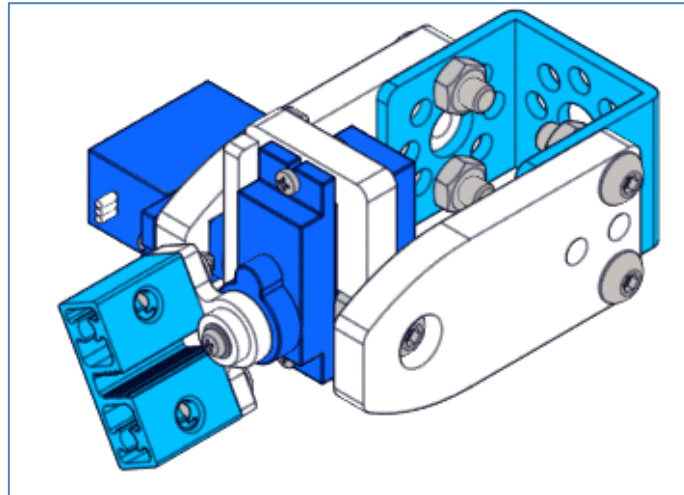
El programa que ejecuta nuestro reto, estando el sensor de ultrasonidos conectado al Puerto 10 de Auriga y la garra robótica al Puerto 1 gracias al controlador de motores DC dual, es el siguiente:



En el script anterior se observa que hemos definido una variable “Distancia” que testeará el valor numérico del módulo de ultrasonidos, así como, el bloque azul “Garra” que se ejecuta sólo cuando el sensor detecte un objeto en frente del robot a una distancia inferior a 15cm. Tras cargarlo, debemos desconectar el cable USB de la placa y conectar las baterías a la misma, para finalmente, encender el robot y ver su ejecución.

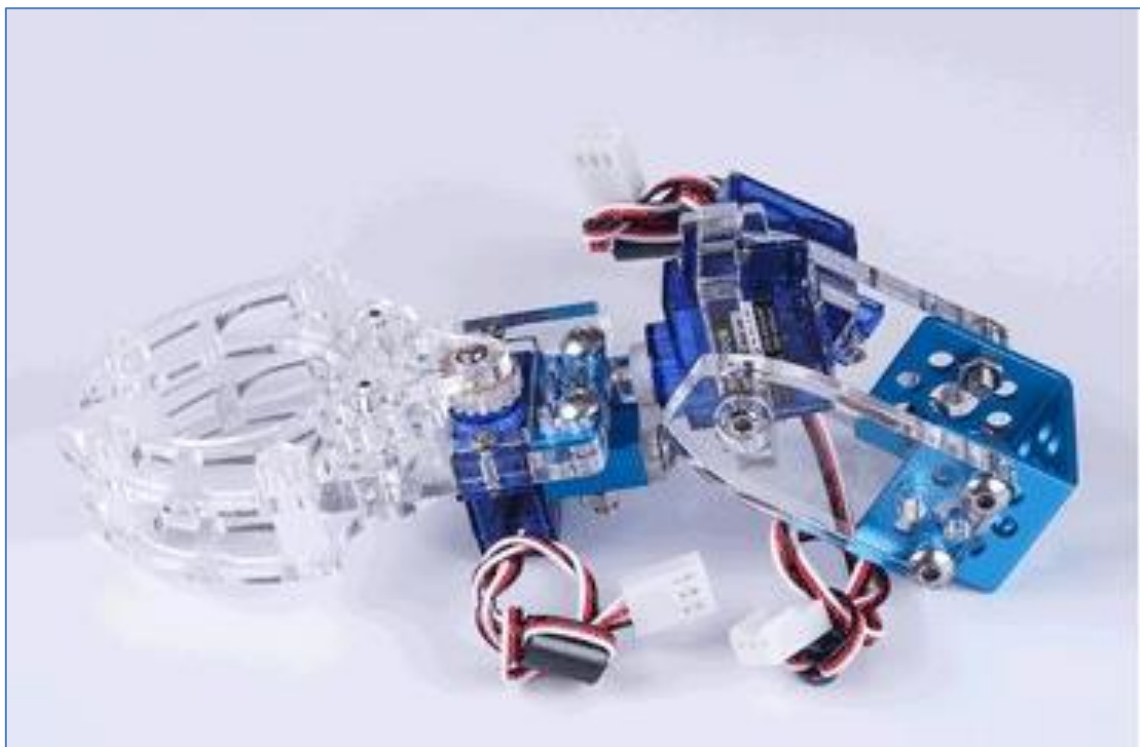
5.14. Mini brazo articulado

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 con otros módulos como por ejemplo 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. Debemos tener presente que, cada servo utiliza un “Banco” del módulo Adaptador RJ25. Por lo tanto, si a mayores le incluimos la mini garra robótica (ver siguiente imagen), estamos añadiendo un servo más, por lo que necesitamos echar mano de otro Adaptador RJ25:



Divirtiéndome con mBot Ranger

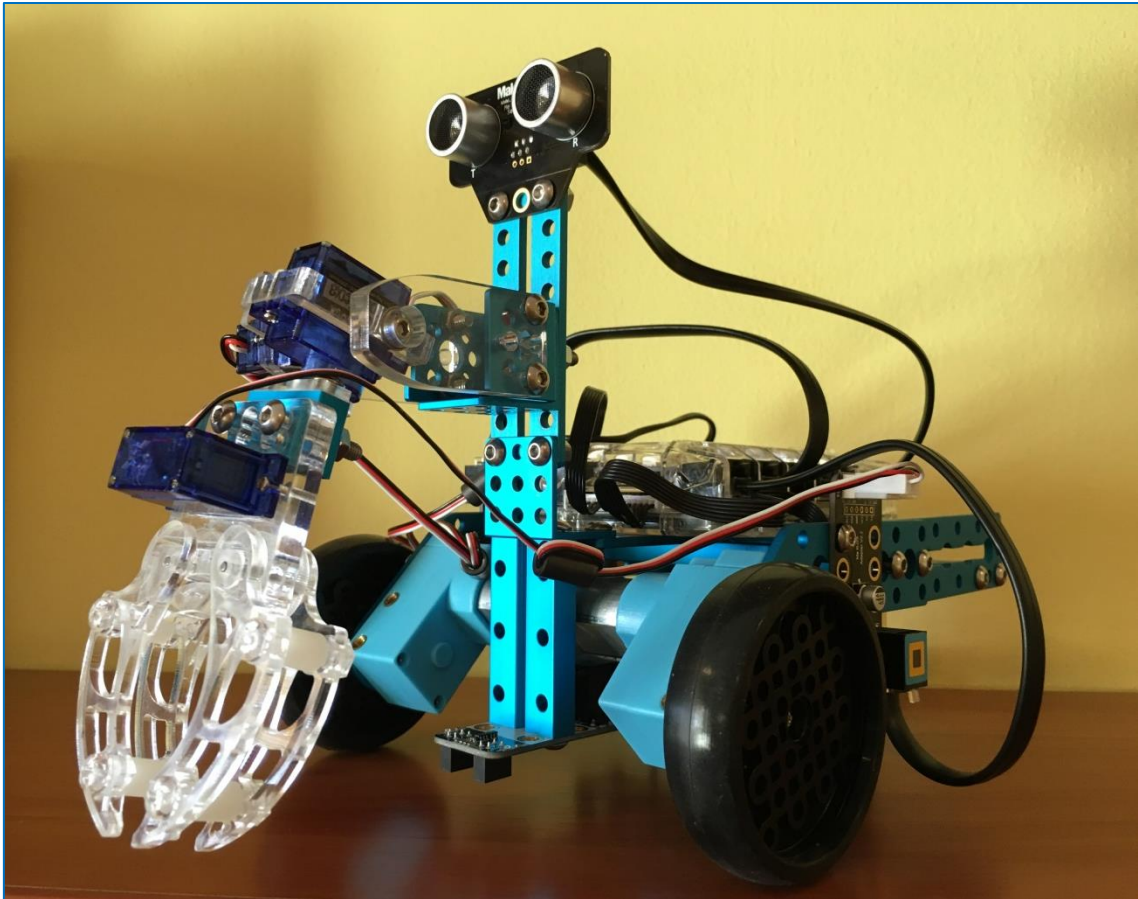
Un ejemplo para su programación con mBlock, usando la placa Auriga, podría ser el siguiente: Notar que en el ejemplo, el servo de la garra está conectado al *Banco1* del adaptador que se asocia al puerto 6 y los otros dos servos cubren ambos *Bancos* del adaptador conectado al puerto 7 de la placa Me Auriga.

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*”:



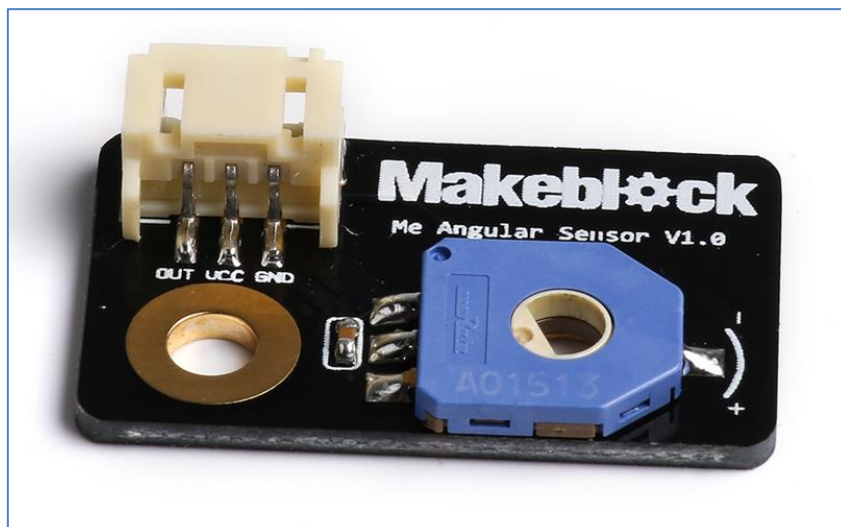
Finalmente, sólo resta hacer el programa en si. El mio es muy sencillo y sólo muestra algunos movimientos de ambos servos del puerto 7. En cada giro programado, se abre y cierra la garra:





5.15. 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

Divirtiéndome con mBot Ranger

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);
}

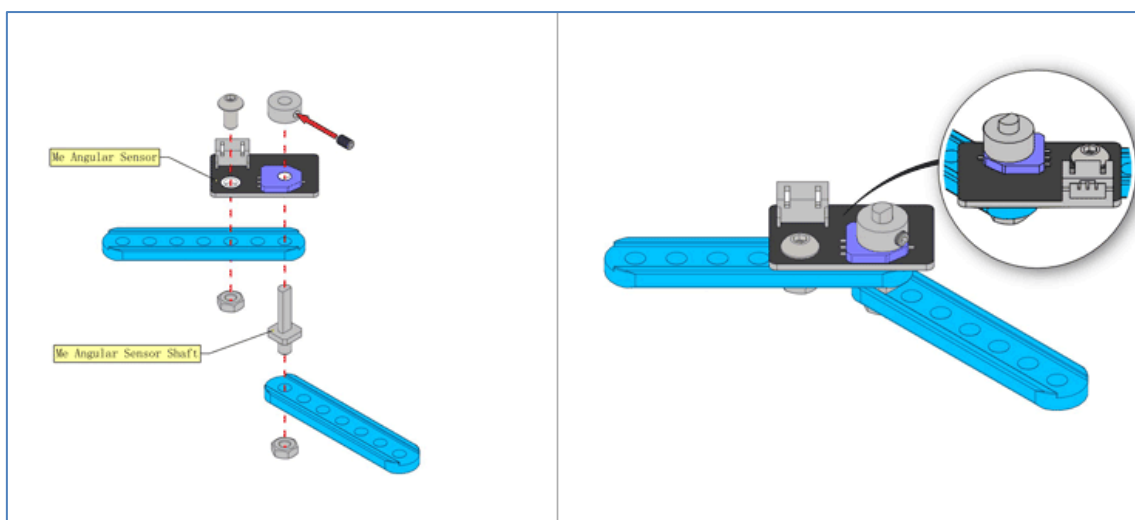
void loop()
{
    //- Valor analógico del slot2 del Adaptador RJ25
    valor = sensor_angular.aRead2();

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

    delay(1000);
}
```

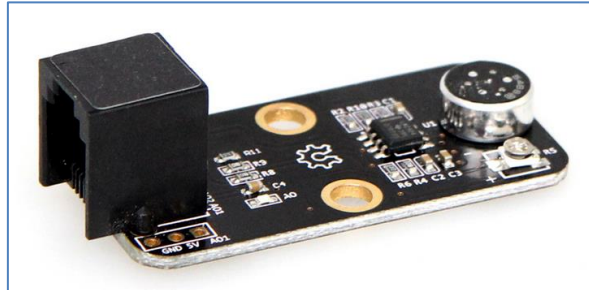
Para una placa Arduino Uno con el shield de makeblock, cambiaríamos "MeOrion.h" por "MeShield.h". Y para una placa Me Auriga como la de nuestro robot, cambiaríamos "MeOrion.h" por "MeAuriga.h" conectando el sensor al puerto 6 de la placa a través del adaptador RJ25.

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



5.16. Sensor de sonido

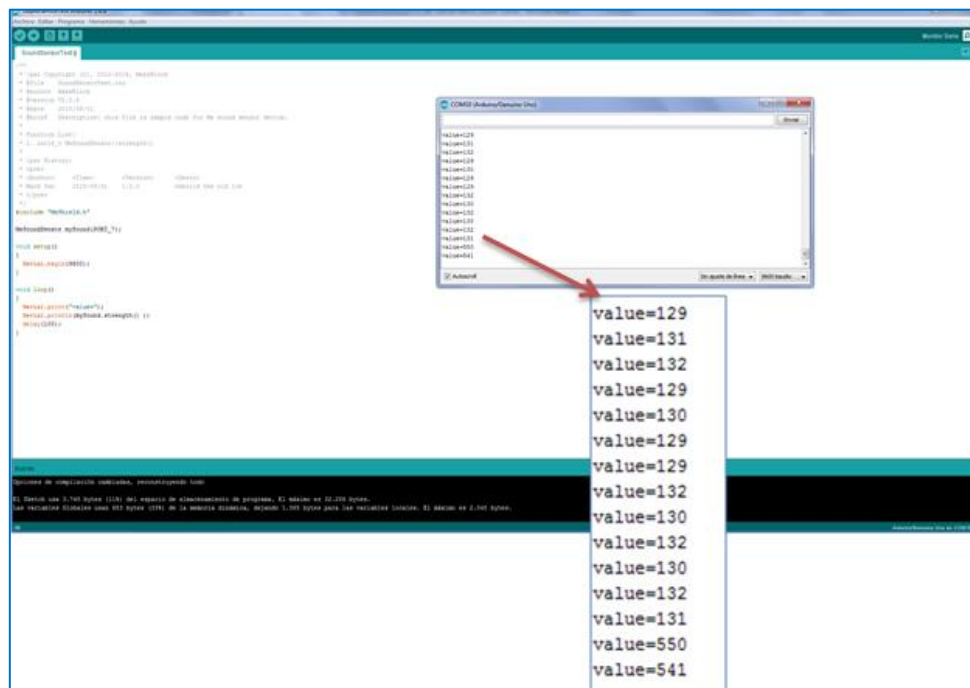
El sensor de sonido está destinado a detectar la intensidad de sonido de su entorno. Podemos utilizarlo para realizar proyectos que requieran que interactuemos con la voz. El sensor dispone de un amplificador LM386 e incorpora un potenciómetro de ajuste de sensibilidad, proporcionándonos valores analógicos 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 o 4, en la Me Auriga del puerto 6 al 10 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 y especificado la placa que vamos a utilizar (MeShield.h), podemos observar que en el puerto *serial begin* se nos muestra un valor numérico elevado cuando alzo la voz y ese valor disminuye cuando estoy en silencio:

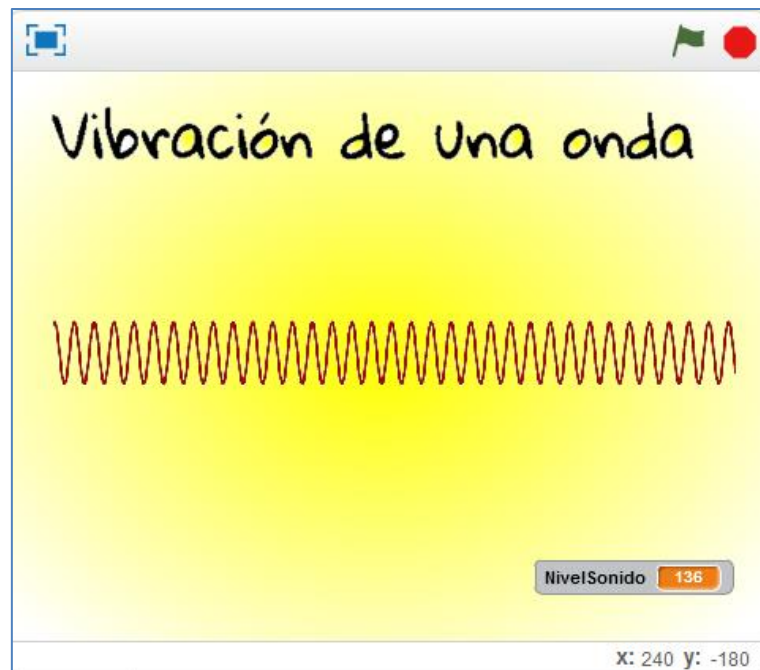




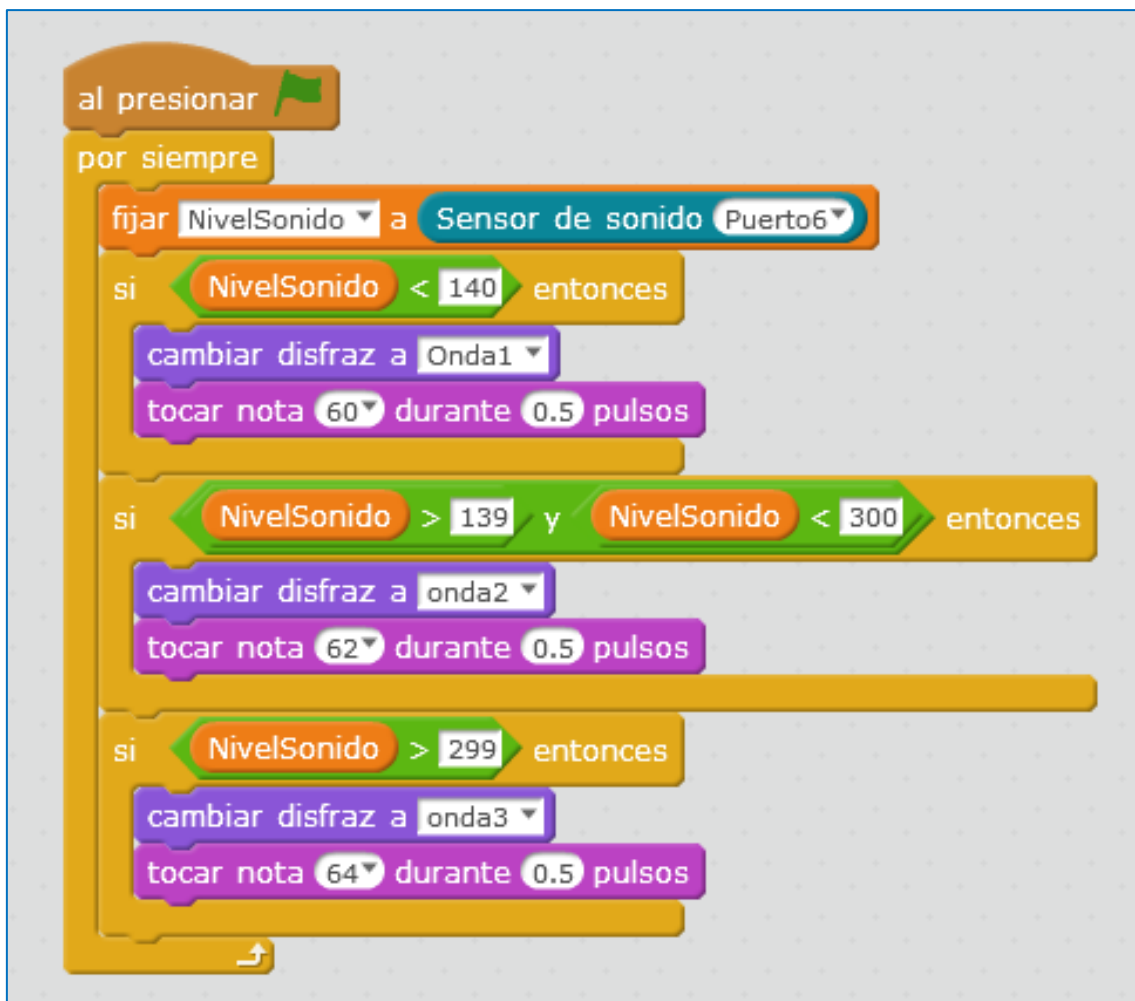
```
SoundSensorTest$  
  
/**  
 * \par Copyright (C), 2012-2016, MakeBlock  
 * @file    SoundSensorTest.ino  
 * @author  MakeBlock  
 * @version V1.0.0  
 * @date    2015/09/01  
 * @brief   Description: this file is sample code for Me sound sensor device.  
 *  
 * Function List:  
 * 1. int16_t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

Como se comentaba anteriormente, este sensor también podemos programarlo con el software mBlock, conectando el sensor de sonido a cualquiera de los puertos numerados del 6 al 10 de la placa Me Auriga, 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 *NivelSonido* de 136 y el objeto se encuentra en el disfraz "ondas1".

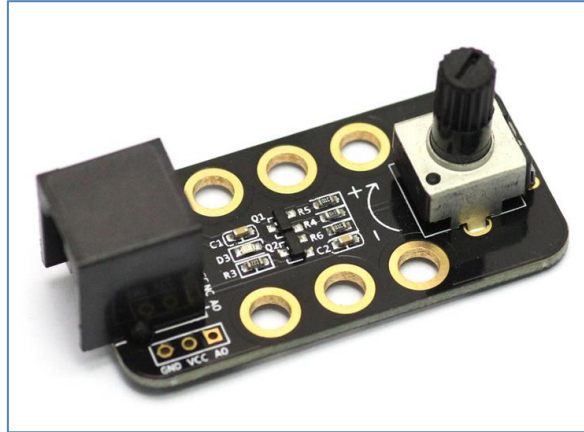


Si el nivel de sonido es menor que 140, debemos ver la onda1 y sonará la nota DO. Si el nivel de sonido está entre 140 y 299, debemos ver la onda2 y sonará la nota RE y, finalmente, si el sonido supera 299, veremos el disfraz de ondas3 y sonará la nota MI.



5.17. Potenciómetro

El potenciómetro lineal de Makeblock es, según nos informa la casa, de 50K Ω . Gira 270 grados, siendo el mínimo valor hacia la izquierda y el máximo valor el 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, a los puertos 7 y 8 del shield de arduino y a los puertos del 6 al 10 de la placa Auriga.



Módulo Potenciómetro

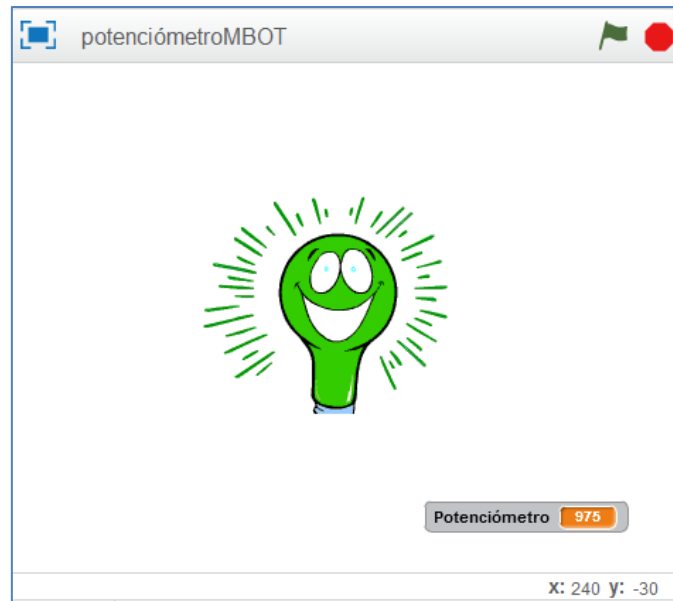
Al igual que en otros módulos, si no queremos usar el shield de arduino y pretendemos conectarlo a la placa Arduino Uno, disponemos de 3 terminales: GND, VCC y AO; siendo esta última una de las salidas analógicas de la placa Arduino Uno.



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 diodos LED RGB del anillo de LEDs del robot. 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 Ranger



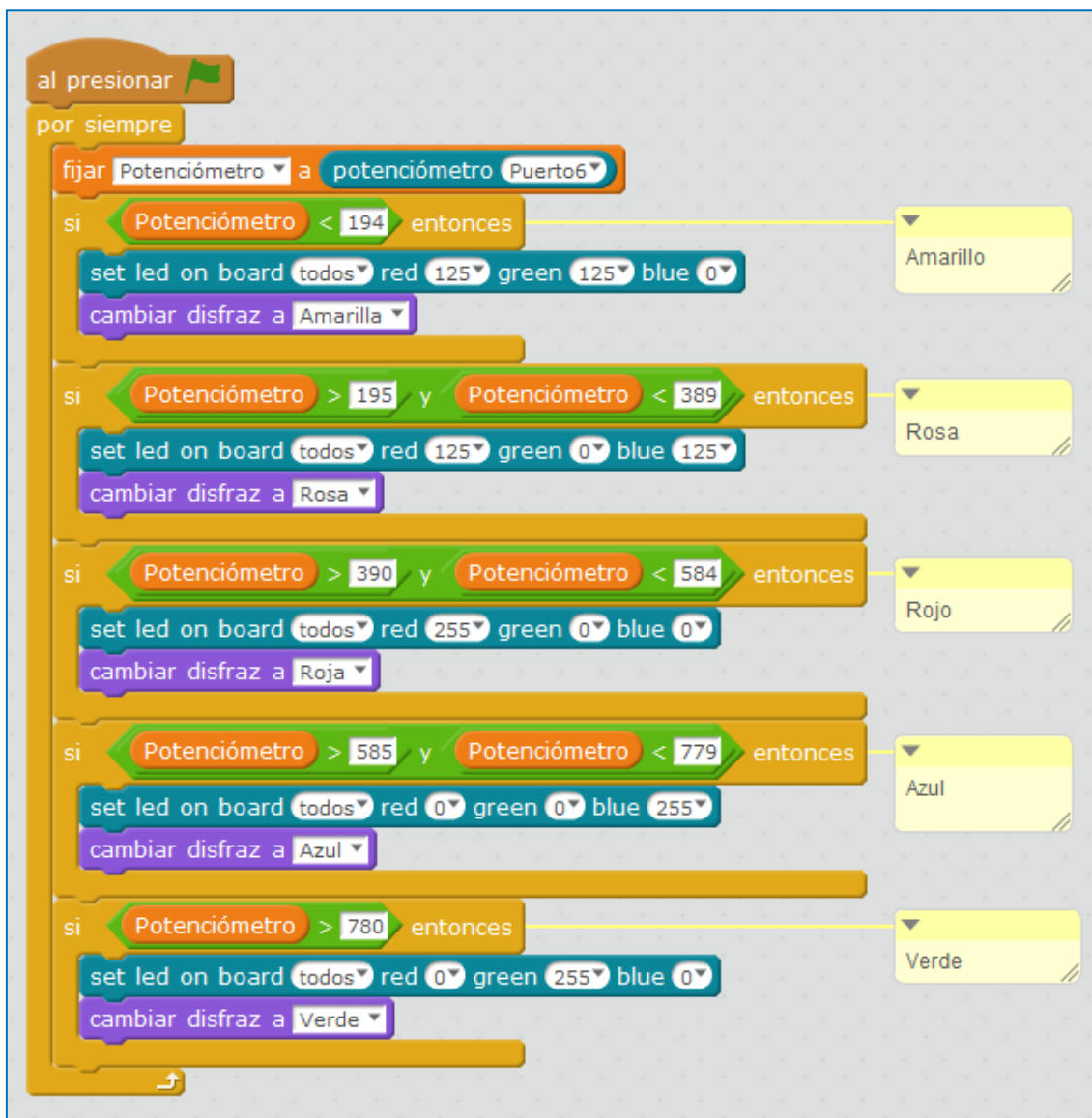
Como puede verse en la siguiente imagen, dispongo de 5 disfraces para recorrer los 975 puntos numéricos del potenciómetro. He decidido hacer intervalos “simétricos” de módulo 195 ($975:5=195$) y, a cada intervalo asociarle un color de LED RGB correspondiente con el color del disfraz:



Disfraces del objeto

Divirtiéndome con mBot Ranger

El programa, para una placa Auriga del mBot Ranger, estando el potenciómetro conectado al puerto 6, es el siguiente:

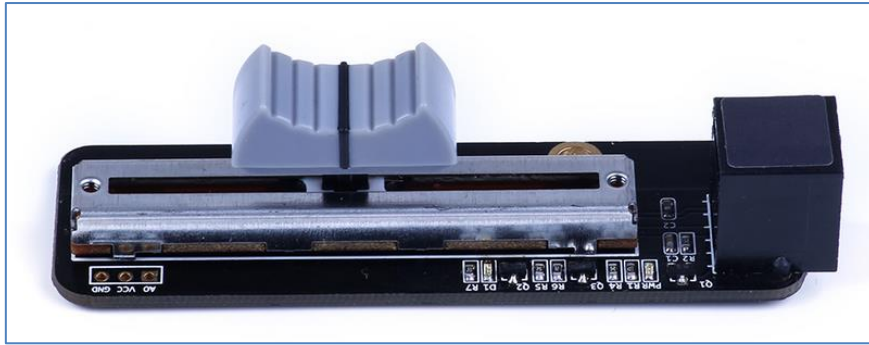


5.18. Potenciómetro slider

El potenciómetro slider o deslizando de Makeblock dispone de una salida analógica que ofrece valores entre 0 y 1023 y que se reflejan en el escenario del mBlock en el siguiente rango de valores: entre -490 y +533. Su ID es negro y esto hace que podamos conectarlo, por medio de un cable RJ25, a cualquiera de los puertos de la placa Me Auriga numerados del 6 al 10.

Se puede utilizar en proyectos interactivos, con juegos, para controlar el movimiento de un sprite por el escenario del mBlock, o para controlar otras variables como el volumen, la intensidad de luz, la velocidad de un motor, etc.

Divirtiéndome con mBot Ranger



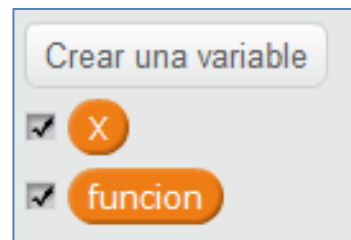
Potenciómetro Slider

Ejemplo 1:

Un reto que podemos crear sería mover un objeto por el escenario en el eje X.

Para ello necesito calcular la función de movimiento lineal cambiando su escala de [-490, 533] a [-210, 210] (El escenario del mBlock toma valores en el eje x desde -240 a 240. Hemos decidido -210 y 210 para que el objeto se vea totalmente en el escenario).

Creo dos variables en el bloque de Datos y Bloques: la variable *X*, que será el valor numérico que nos ofrece el sensor y la variable *funcion* que no es más que una función lineal que diseñaré para realizar mi cambio de escala deseado.

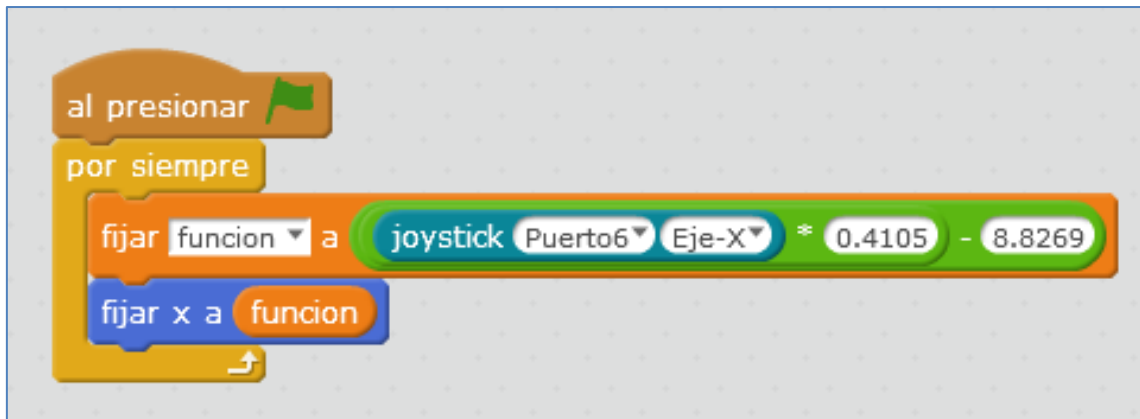


Para calcular los parámetros de la pendiente y de la ordenada en el origen de nuestra variable *funcion*, necesitamos dos puntos. Esos puntos son (-490, -210) y (533, 210). Así pues, sustituyéndolos en la ecuación de una recta lineal, llegamos a un sistema de dos ecuaciones con dos incógnitas cuya solución es:

$$f(x) = 0,4105x - 8,8269$$

m	0.4105
n	-8.8269

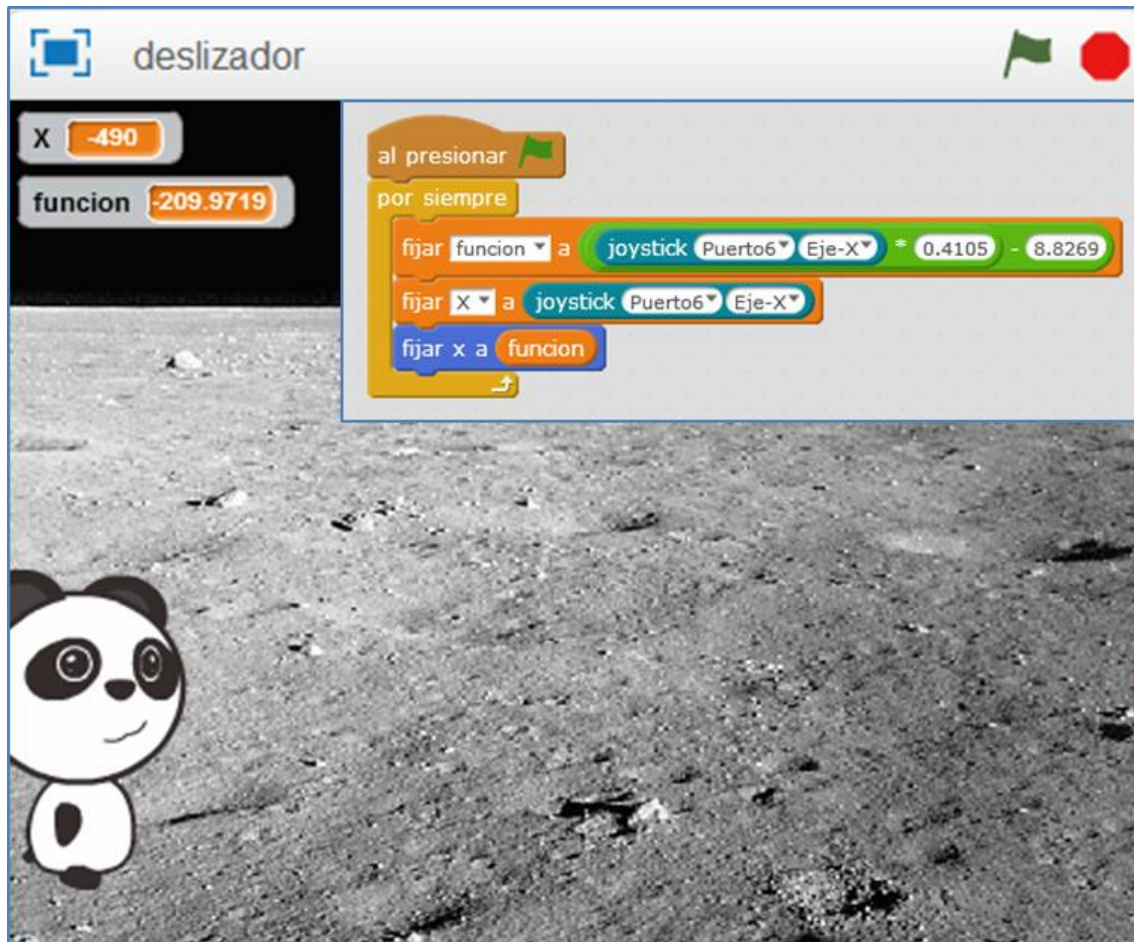
Calculada la función, podemos programar el objeto. Con el siguiente script conseguimos que nuestro sprite se mueva por el escenario en el sentido del eje X al deslizar el slider del potenciómetro. El slider se encuentra conectado al puerto 6 de la placa Me Auriga:



Si ampliamos el programa anterior con la línea de la siguiente imagen, podemos comprobar los valores mínimo y máximo de la variable X del slider:



Posición máxima en el eje X



Posición mínima en el eje X

Actuaríamos de la misma forma si en lugar de moverlo por el eje X quisiéramos desplazar el objeto por el escenario en el eje Y. Buscaríamos los puntos para poder calcular la nueva función lineal y cambiaríamos el comando del joystick al eje Y (pesar que el eje Y varía en el escenario desde -180 a 180).

Ejemplo 2:

Podríamos controlar la intensidad de luz del anillo de LEDs RGB.

Cada color RGB toma valores entre 0 y 255 y esto significa que debemos crear una función que nos convierta el valor máximo y mínimo del slider en los valores máximo y mínimo de cada color RGB.

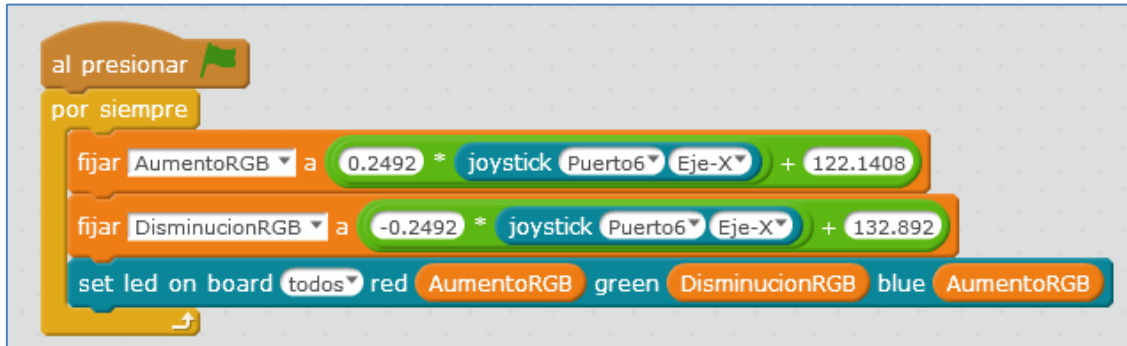
En este ejemplo vamos a construir dos funciones lineales de modo que, en una aumente el valor numérico del color, de 0 a 255, y en otra disminuya su valor de 255 a 0. Estas funciones las defino en el bloque de *Datos y Bloques* y las denoto por *AumentoRGB* y *DisminuciónRGB*.

Para calcular los parámetros pendiente y ordenada en el origen en la función lineal *AumentoRGB* utilizo los puntos (-490, 0) y (533, 255). Obviamente, en el cálculo de la función *DisminuciónRGB* usaría los puntos (-455, 255) y (533, 0).

Divirtiéndome con mBot Ranger

He decidido aplicarle el aumento al color rojo y azul y la disminución al verde. Es decir, cuando el color rojo y azul tomen el valor máximo (255) el color verde tomará el valor mínimo (0).

El script que cumple mis deseos es el siguiente:



5.19. 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.

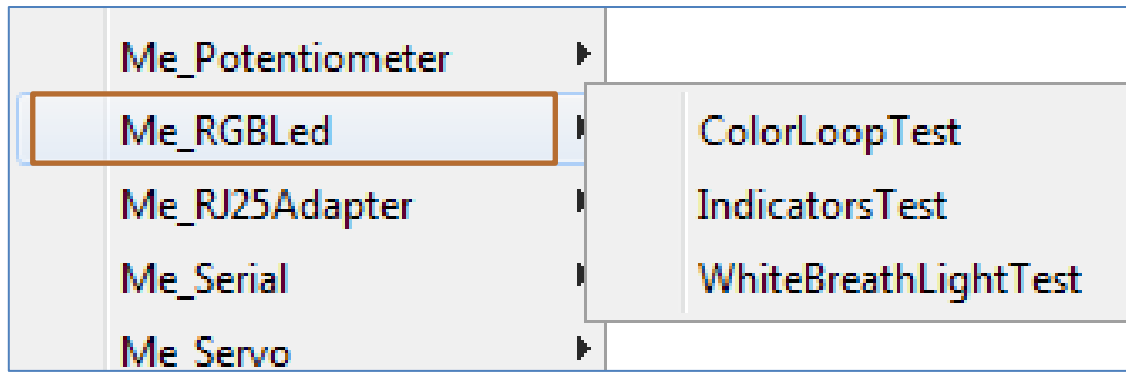


Módulo 4 LEDs RGB

Su conector es de color amarillo, así que, sólo lo podremos conectar a los puertos que dispongan de este color. En una placa Auriga nos vale cualquiera de los puertos numerados de 6 al 10.

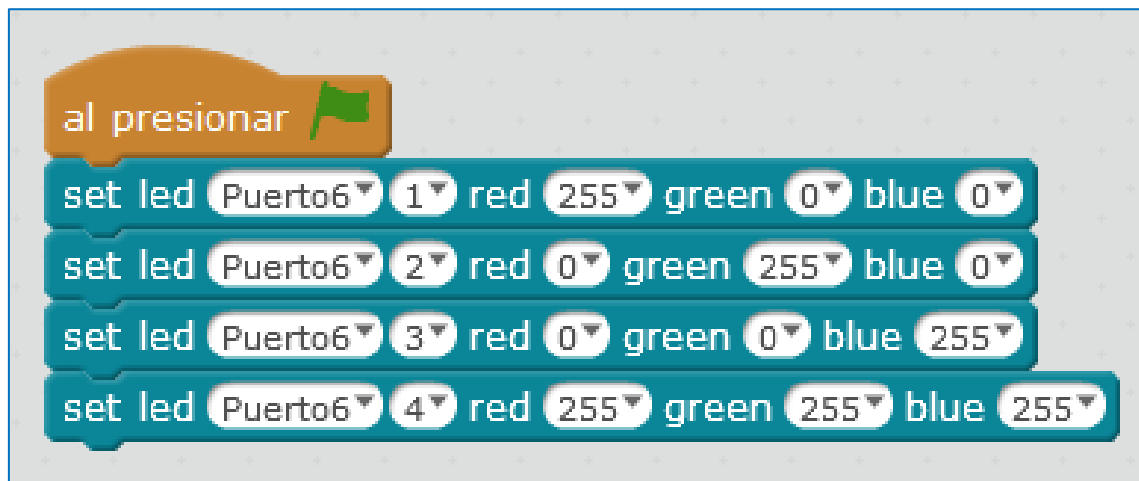
Al igual que en otros componentes, podemos usar este módulo directamente en una Arduino Uno, sin shield. En este caso, las tres conexiones se nos muestran en un lateral: Vcc, GND y SIG, siendo la señal un pin digital de la placa Arduino Uno.

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

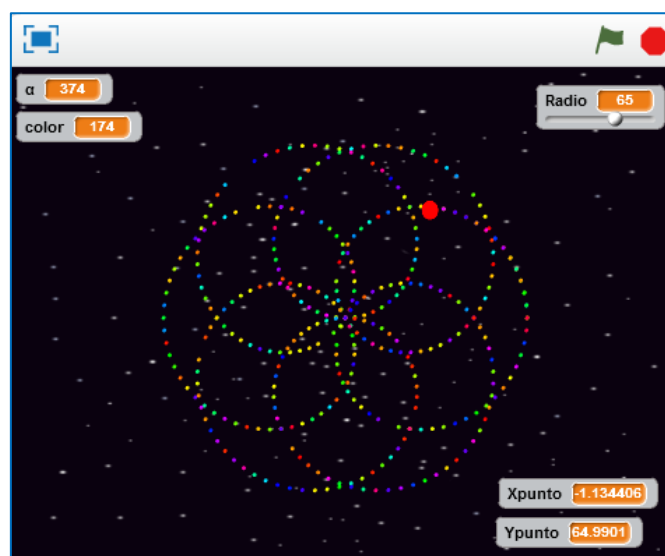


Ejemplos de las librerías de Makeblock

Desde mBlock podemos programar cada LED de forma independiente. Por ejemplo, si el módulo está conectado al puerto 6 (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:

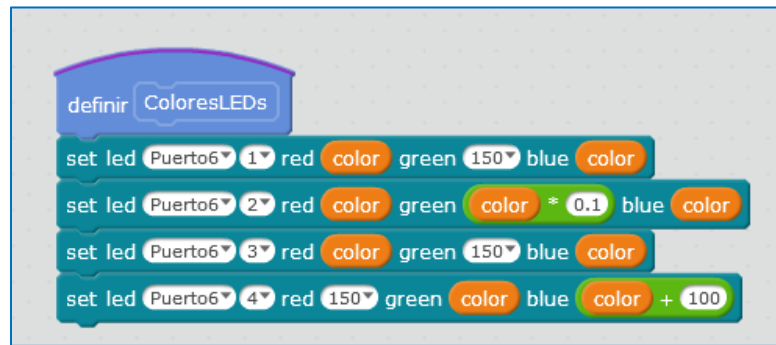
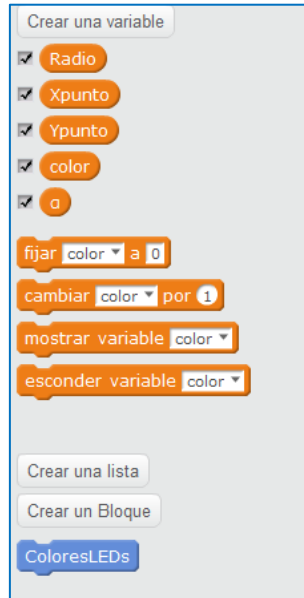


El programa anterior es muy simple y con mBlock podemos hacer mucho más. Por ejemplo, “*simular un arcoíris*” en el módulo 4 LEDs RGB al mismo tiempo que se *dibuja una espiral* en el escenario (ver siguiente imagen):



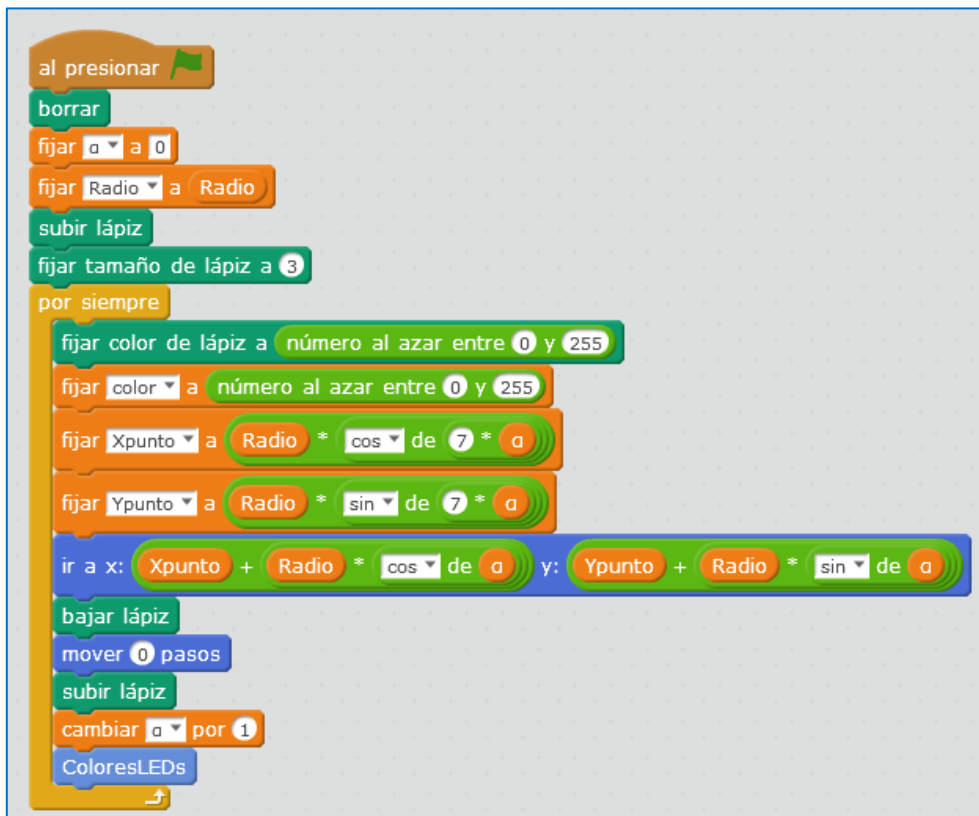
Divirtiéndome con mBot Ranger

En la figura anterior observamos 5 variables diferentes, de las cuales, el Radio, es del tipo deslizador. Antes de hacer clic en la bandera verde escogeré en el deslizador el valor numérico del radio de la espiral que quiera implementar. También se ha creado un bloque para describe los colores que se mostrarán en cada LED RGB del módulo. Colores que dependerán del valor aleatorio que tome la variable *color* entre 0 y 255:



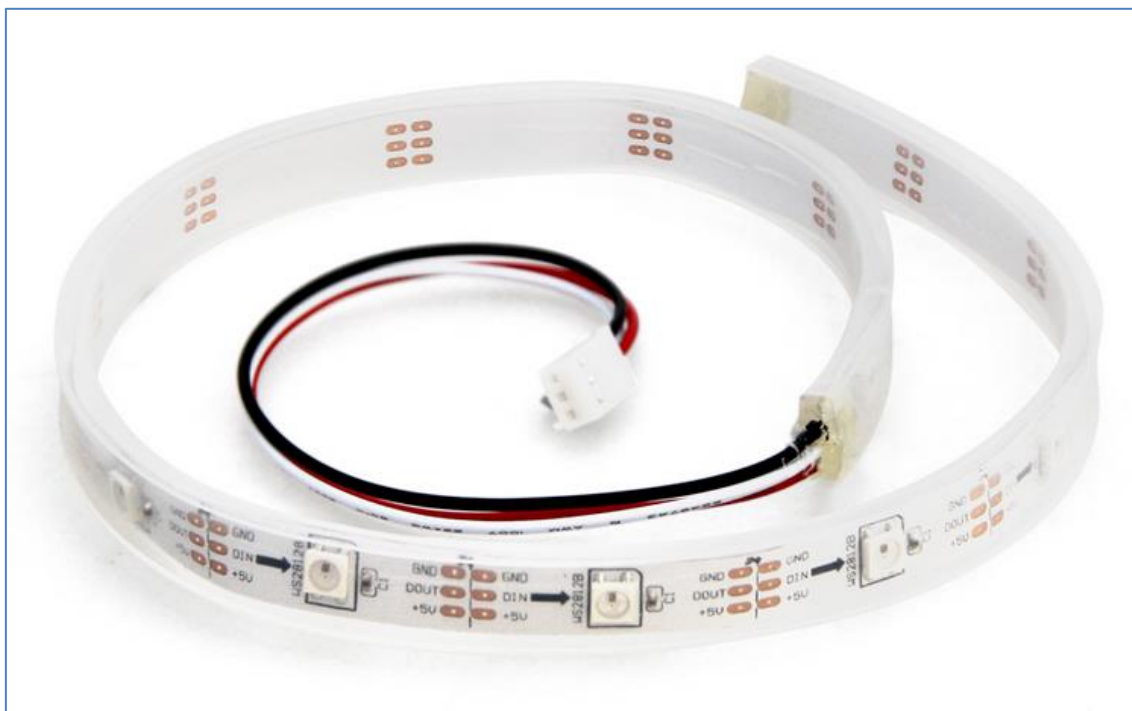
Variables y bloque ColoresLEDs

Finalmente, el script que dibuja la espiral en un gran abanico de colores y llama a cada uno de los LEDs del módulo a programar es el siguiente:



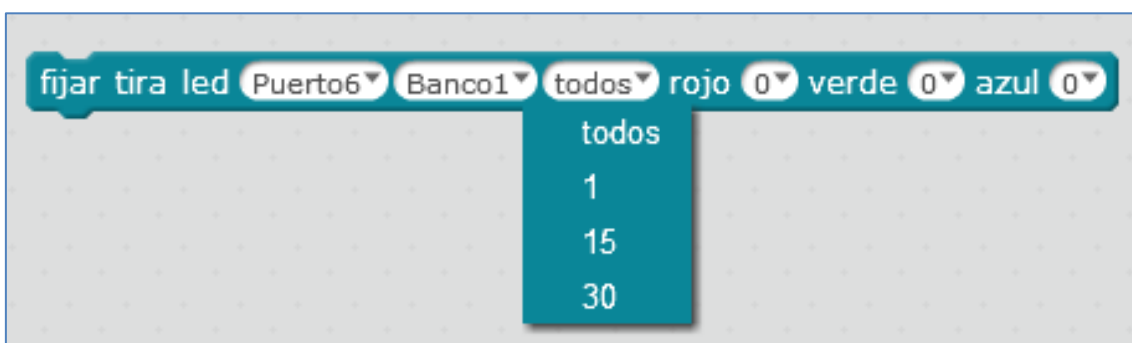
5.20. Tira de LEDs RGB

La casa Makeblock dispone de dos tiras de LEDs RGB de diferente longitud. Una de medio metro, que incluye 15 diodos, y otra de un metro que dispone de 30 LEDs.



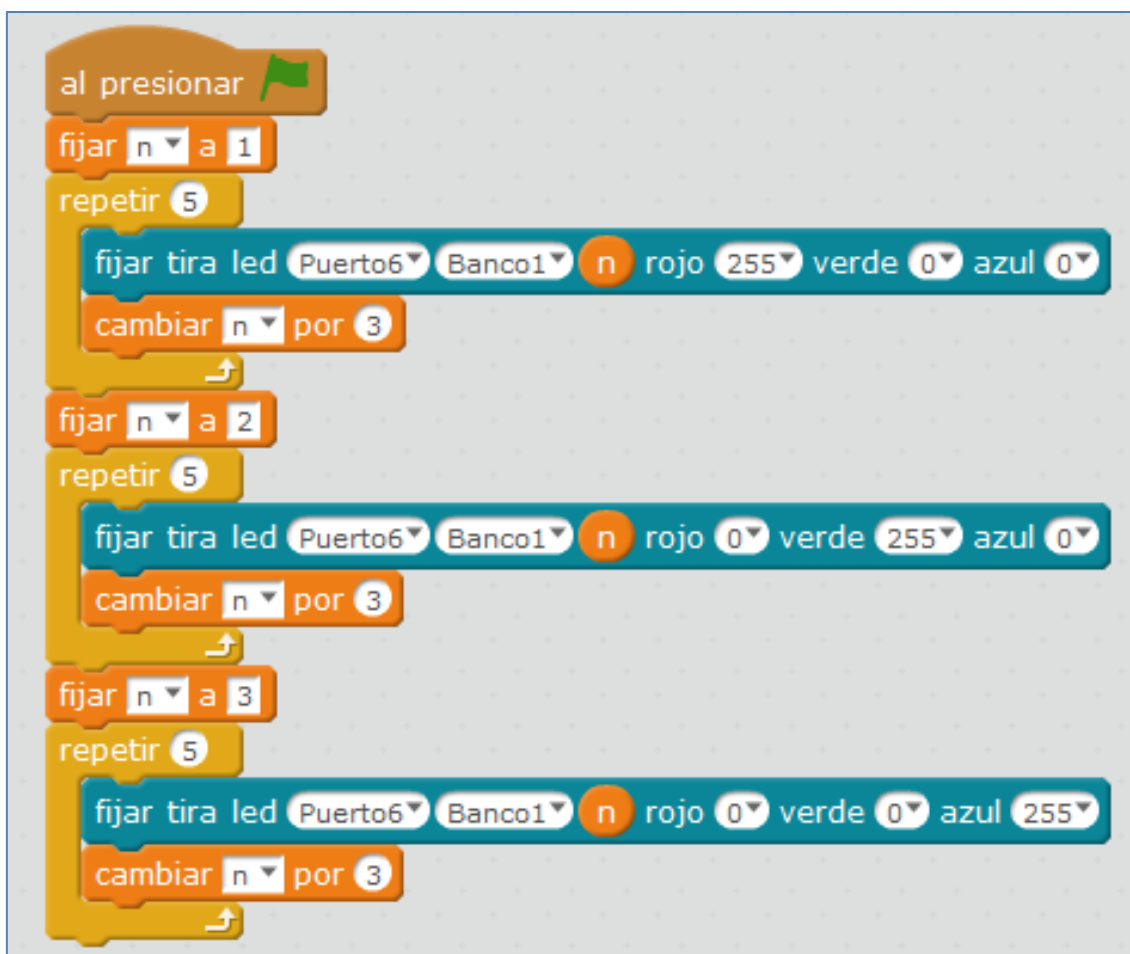
Tira de LEDs RGB de 50cm

Ambas se pueden programar desde el mBlock, con scratch, siendo posible controlar cada LED de forma independiente. Para su programación utilizamos el siguiente comando:



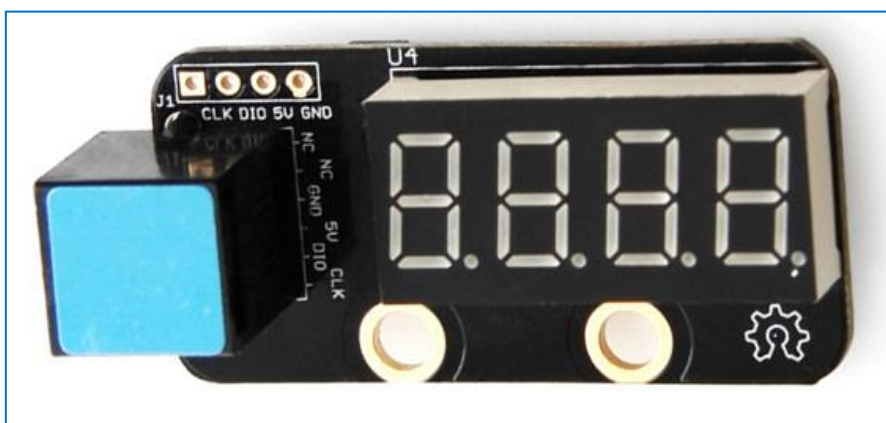
Este componente requiere de un adaptador RJ25 de modo que la tira de LEDs RGB puede conectarse al *Banco1* o al *Banco2* del adaptador.

En el siguiente ejemplo se ha utilizado la tira de LEDs RGB de 50cm, conectada al *Banco1* de una adaptador RJ25 que se une al puerto 6 de la placa Me Auriga. Se ha programado la tira para que los 15 LEDs sigan la secuencia Rojo-Verde-Azul.



5.21. Módulo Display 7 segmentos

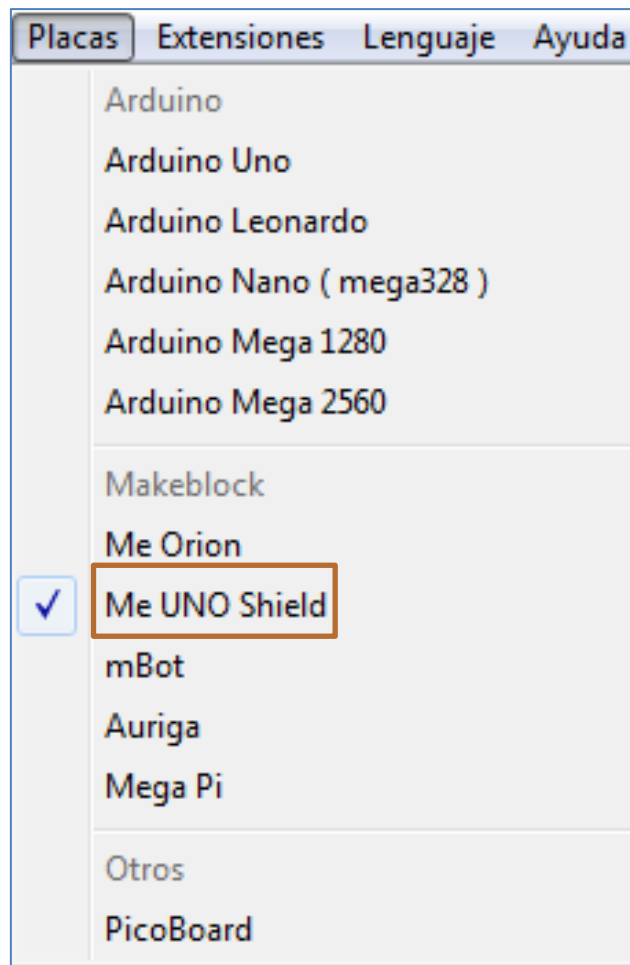
El módulo display 7 segmentos tiene su ID azul (puerto de señal digital doble). Color que nos indica que puede programarse con las diferentes placas de Makeblock que dispongan de un RJ25 con ese color. Otra opción es la placa Arduino Uno y en ella, si se desea, se le puede acoplar el shield para arduino 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.



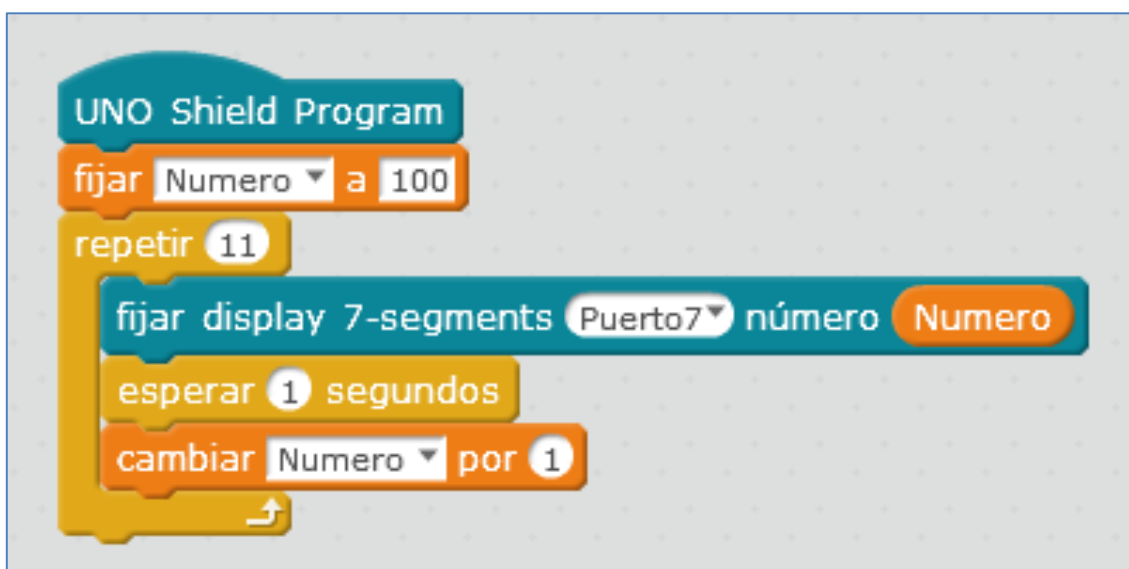
Módulo Display 7 segmentos

Divirtiéndome con mBot Ranger

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:

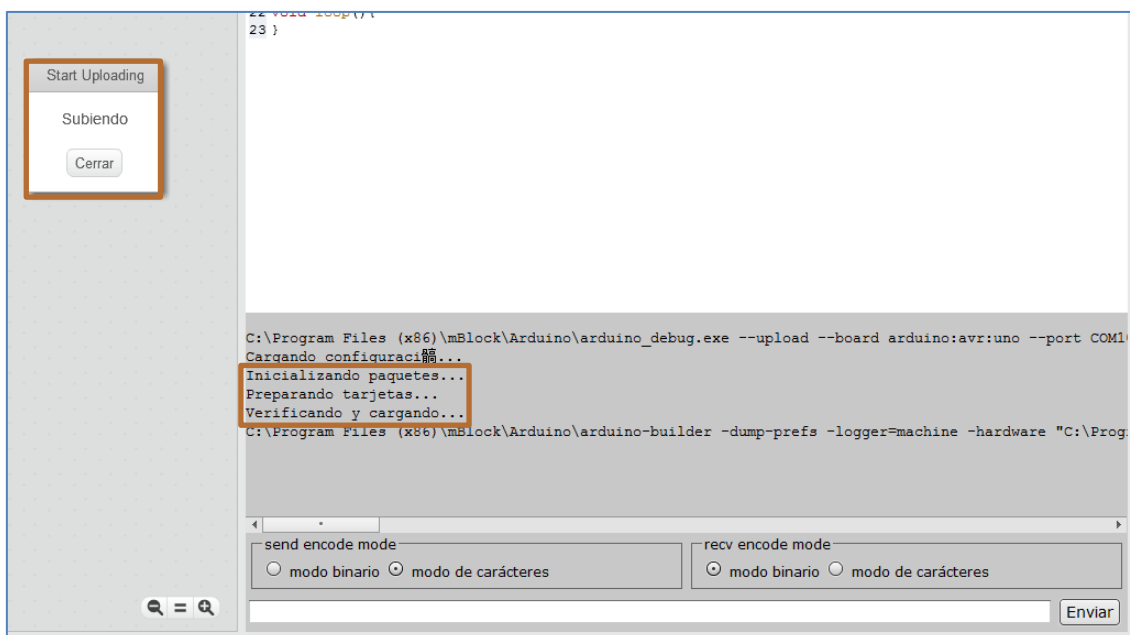


Supongamos que queremos visualizar un contador ascendente de 100 a 110, usando un display 7 segmentos conectado al puerto 7 del shield de arduino. El programa en mBlock sería el siguiente:

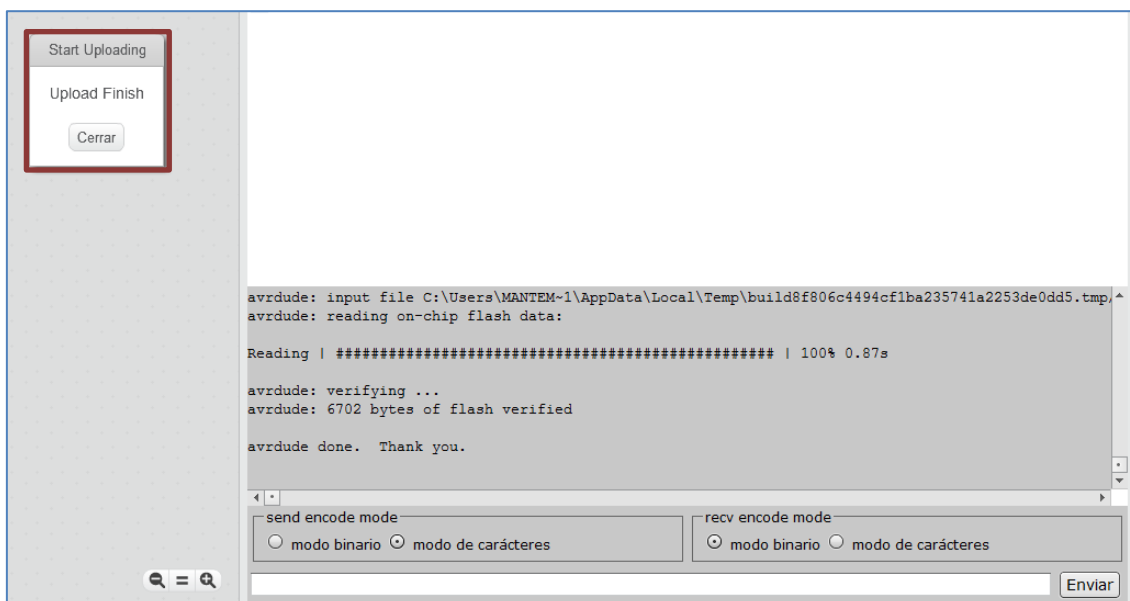


Divirtiéndome con mBot Ranger

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 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:

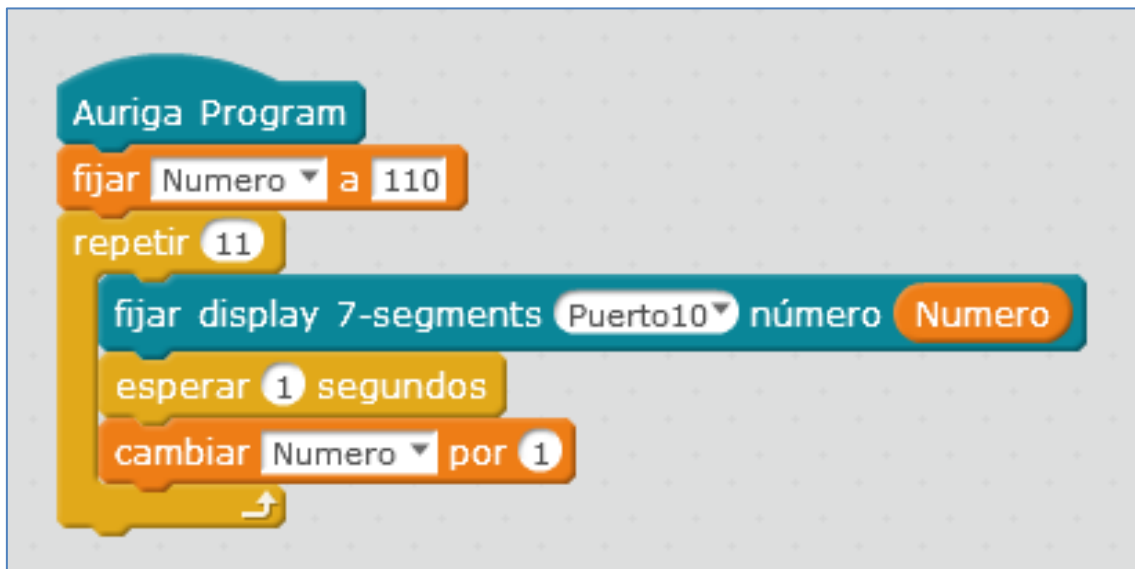


Si nos fijamos, el código arduino es el siguiente:


```
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
15 void setup(){
16   Numero = 100;
17   for(int i=0;i<11;i++){
18     {
19       seg7_7.display(Numero);
20       delay(1000*1);
21       Numero += 1;
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.

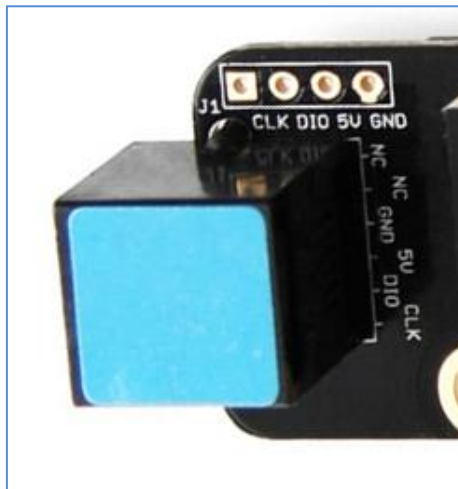
Si quisiéramos usar la placa Auriga, cargando el programa a la misma, el script que debemos ejecutar es el siguiente: (notar que el display está conectado al puerto 10 de la placa Auriga)



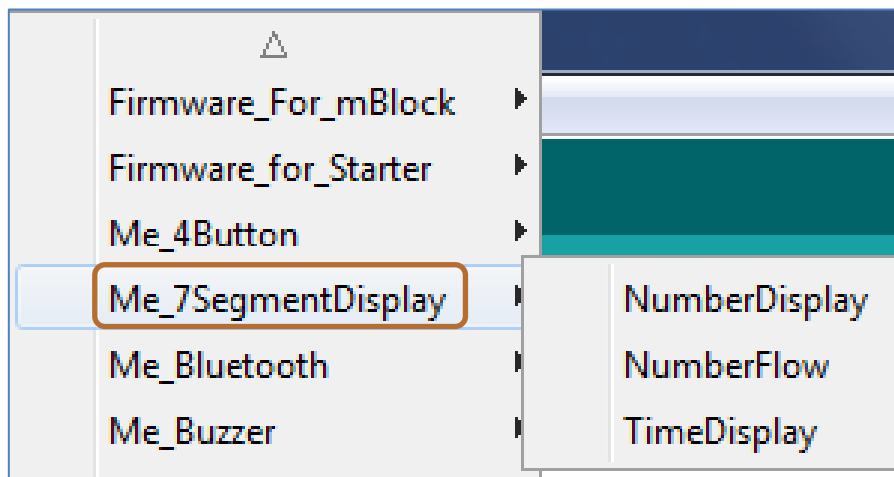
Como en otros módulos, también podemos conectar directamente el módulo display7 segmentos a la placa arduino. Para ello, debemos conectar las 4 conexiones de su lateral (ver siguiente imagen) a la placa Arduino Uno. Estas son:

Divirtiéndome con mBot Ranger

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:



En el ejemplo *NumberDisplay* podemos visualizar los números de 0 a 100 con un retraso de 100ms; modificar su código para un contador ascendente o descendente en concreto, es muy sencillo. Y si cargamos el ejemplo *TimeDisplay* a nuestra placa con el display, el módulo 7 segmentos se comporta como un cronómetro digital, contando segundos, minutos y horas.

5.22. 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. En realidad este sensor es un magnetómetro, ya que mide la fuerza del campo magnético terrestre, pero también se le suele llamar compás o brújula digital porque nos proporciona la orientación del chip respecto al polo magnético.

Divirtiéndome con mBot Ranger

Si lo usamos con nuestro robot, este, obviamente, tendrá motores y ellos afectarán a la medida que nos proporciona este sensor. La solución es, tras montarlo en el robot, calibrarlo, y de esa forma, conseguimos que el campo magnético de los imanes de los motores y otros de cuerpos metálicos, no afecten a la medida que nos proporcionará el sensor.

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 se 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.

Con mBlock, tras calibrarlo, conseguir programar una brújula digital es muy sencillo.

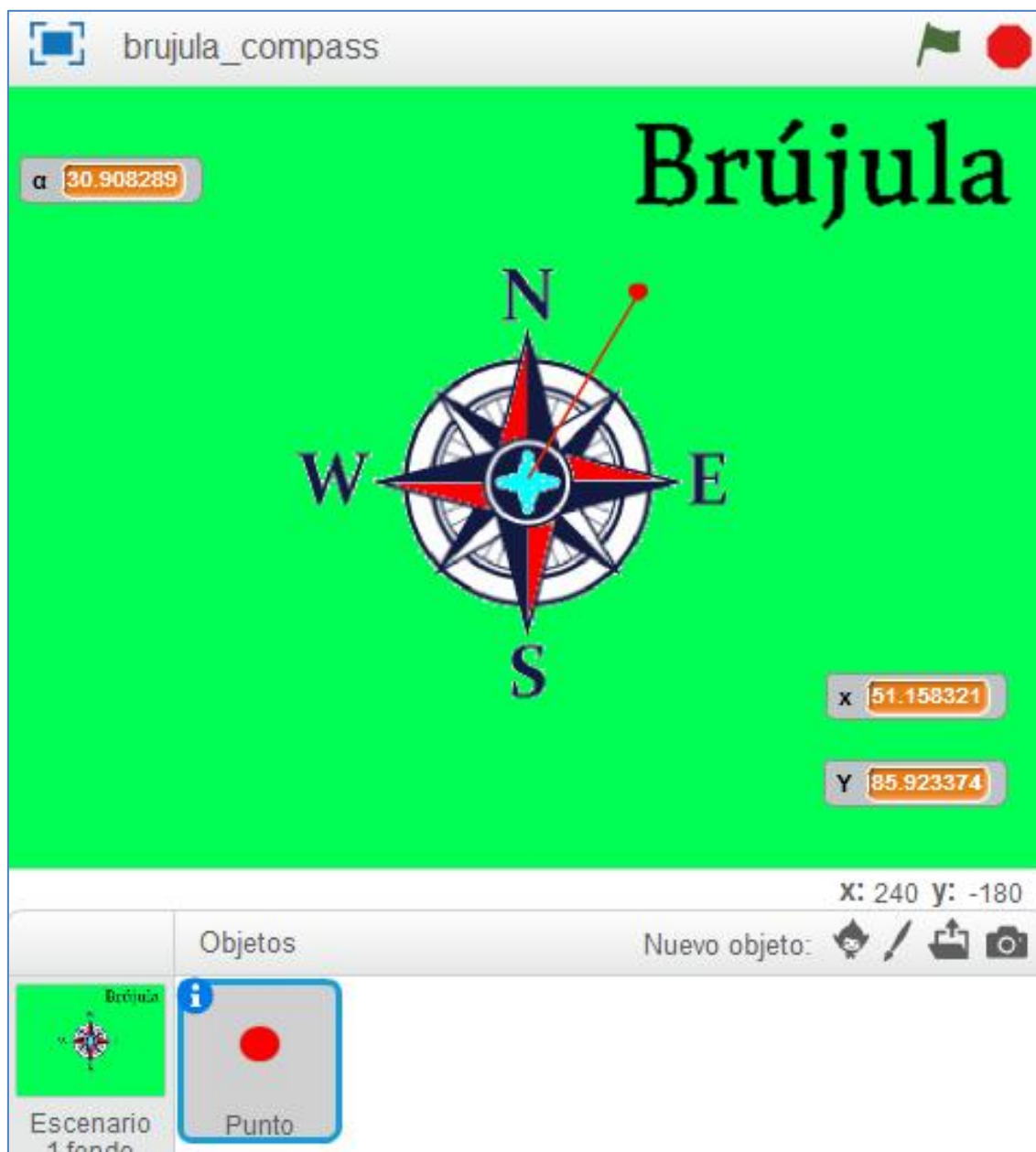


Icono brújula digital

Divirtiéndome con mBot Ranger

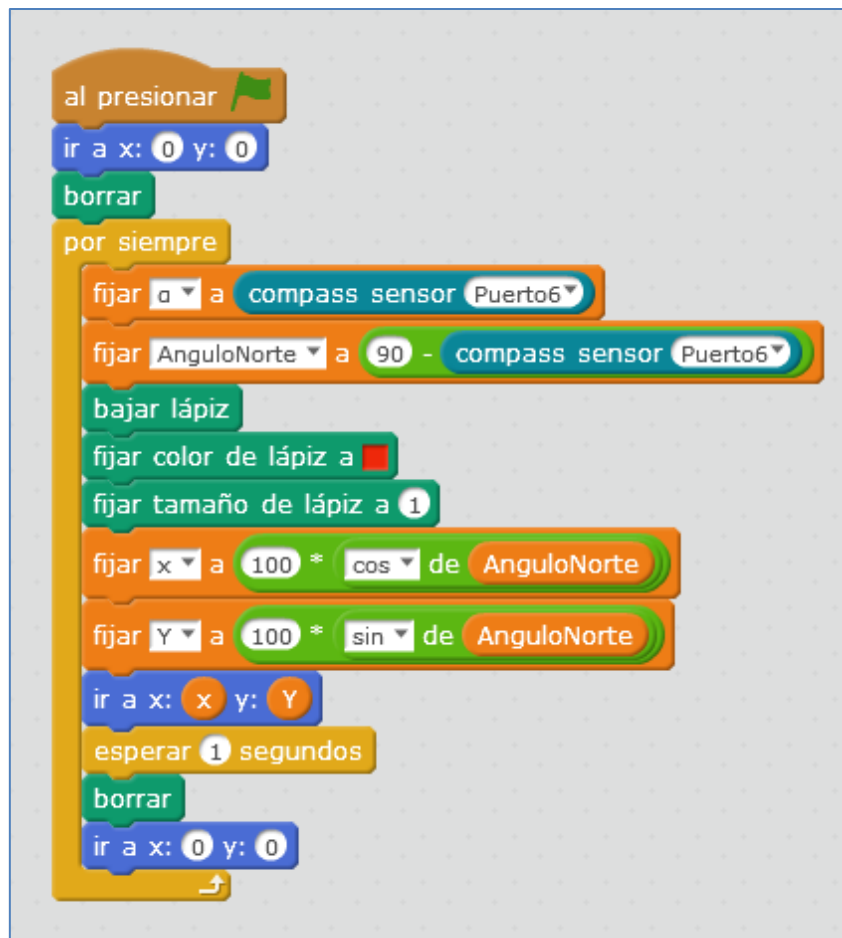
Diseñamos un escenario que simule una brújula como la de la imagen anterior. Nuestro programa `brújula_compass` requiere de único objeto, que he definido como “*Punto*” y ejecutará nuestra brújula.

El módulo Me Compass nos otorga el ángulo “ α ” respecto al polo magnético, ángulo que se orienta en el eje X y que convierto a mi eje Y (Norte) utilizando la variable “`AnguloNorte`”, que efectuará la operación $90-\alpha$. Después, simplemente, se programan las coordenadas X e Y de ese ángulo, que serán las que visionan en nuestra brújula, tal y como se observa en la siguiente imagen del escenario:



Objeto y escenario del programa Brújula

Por lo tanto, el script para el objeto *Punto* que debemos ejecutar es el siguiente:



5.23. 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

Divirtiéndome con mBot Ranger

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.



```
MeGasSensorTest Arduino 1.6.8
Archivo Editar Programa Herramientas Ayuda

MeGasSensorTest$

/**
 * \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 snesor 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"

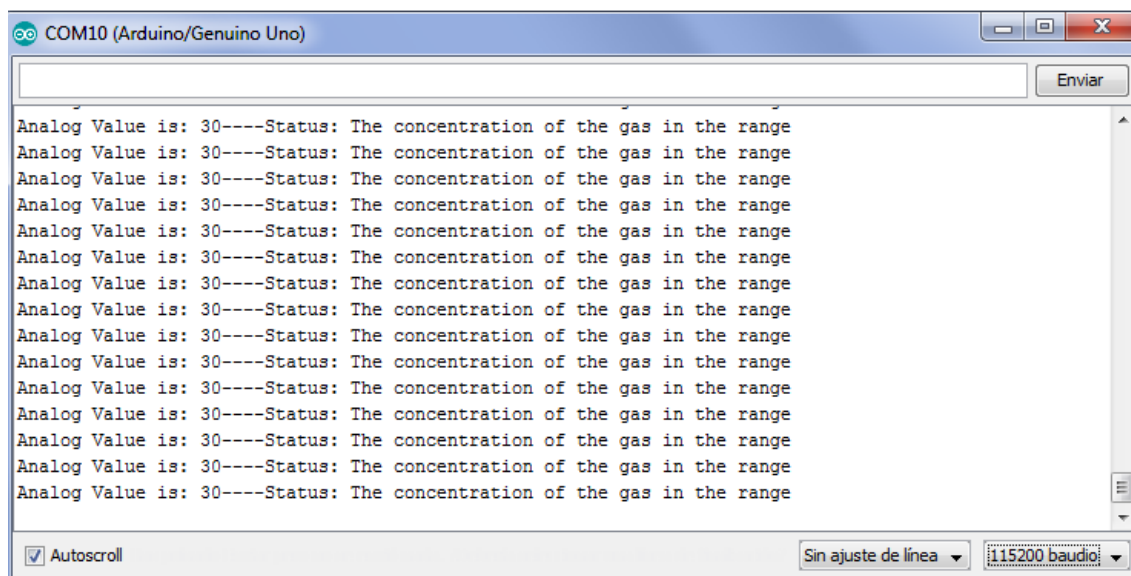
MeGasSensor GasSensor1(PORT_8)

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

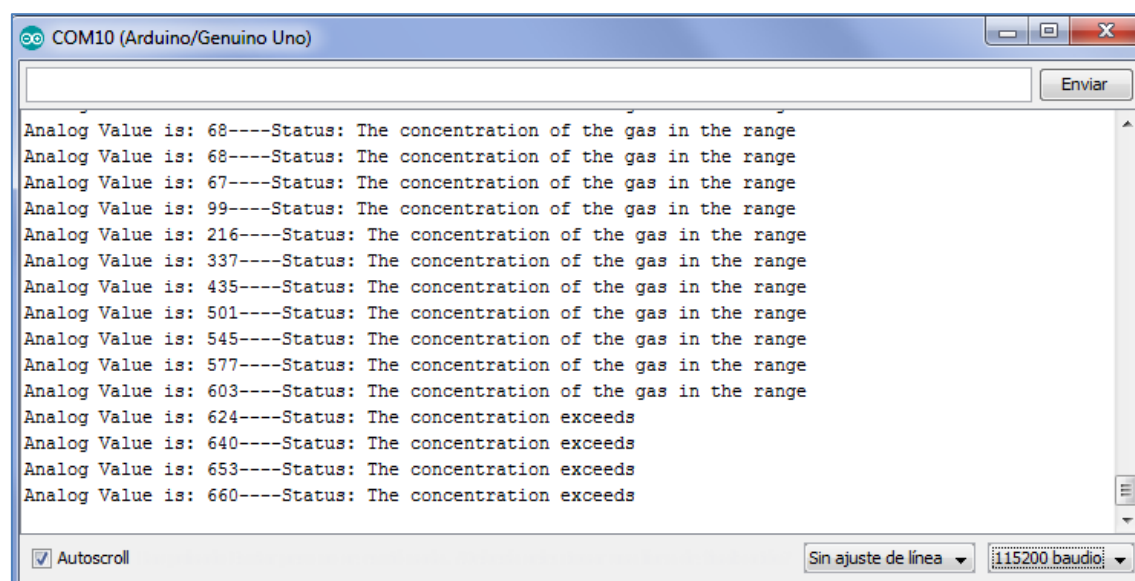
void loop()
{
  Serial.print("Analog Value is: ");
  Serial.print(GasSensor1.readAnalog());
  Serial.print("----Status: ");
  if(GasSensor1.readDigital() == Gas_Exceeded)
  {
    Serial.println("The concentration exceeds");
  }
  else if(GasSensor1.readDigital() == Gas_not_Exceeded)
  {
    Serial.println("The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


Divirtiéndome con mBot Ranger

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.

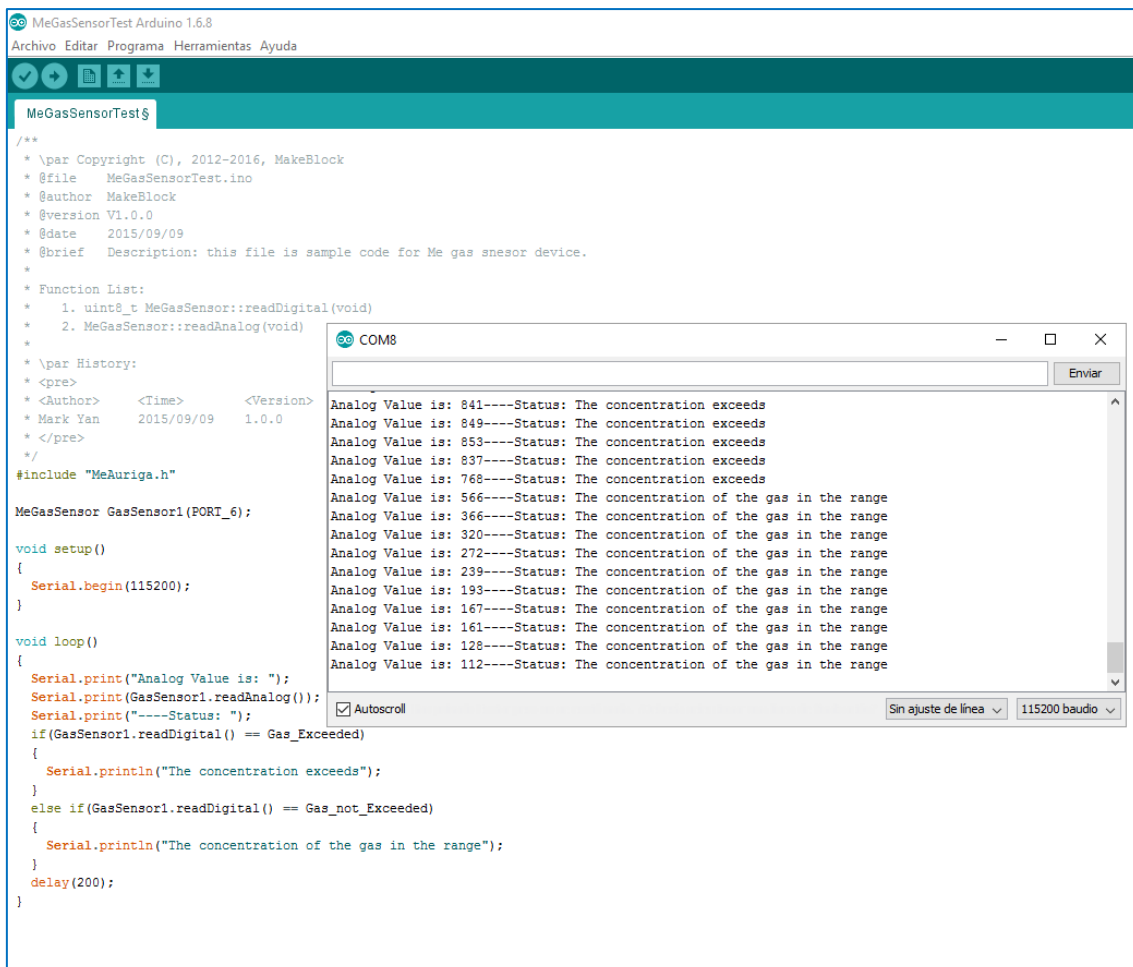
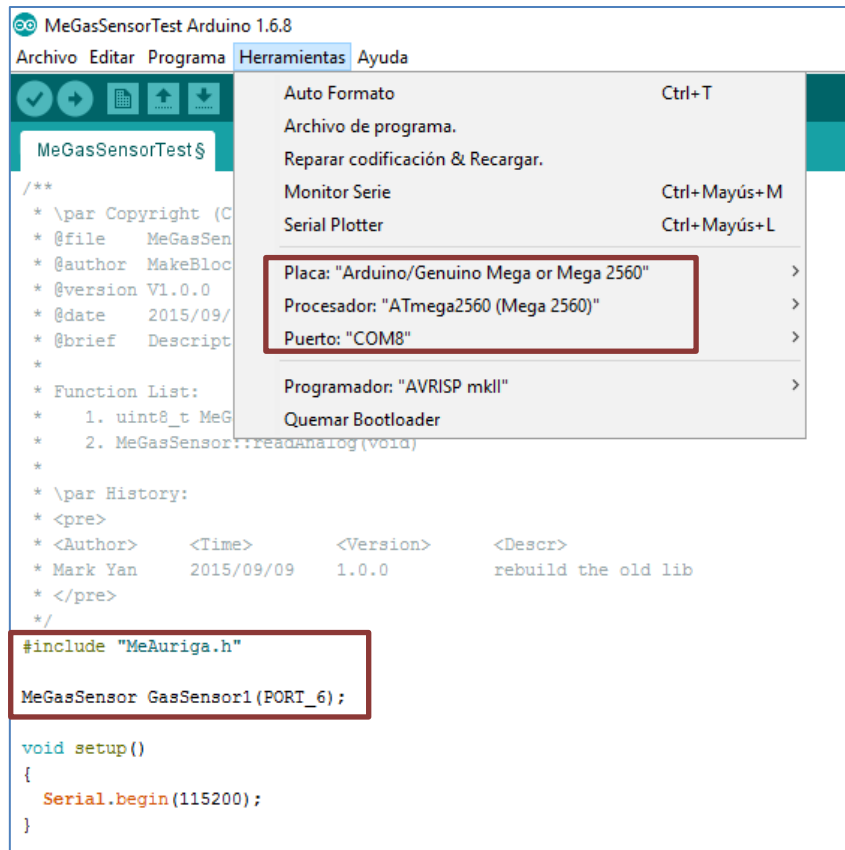


Pero, si dejamos escapar el gas del mechero, el valor analógico cambia, aumentando considerablemente. En el puerto podemos leer el mensaje de alerta informándonos que la concentración de gas es excesiva.



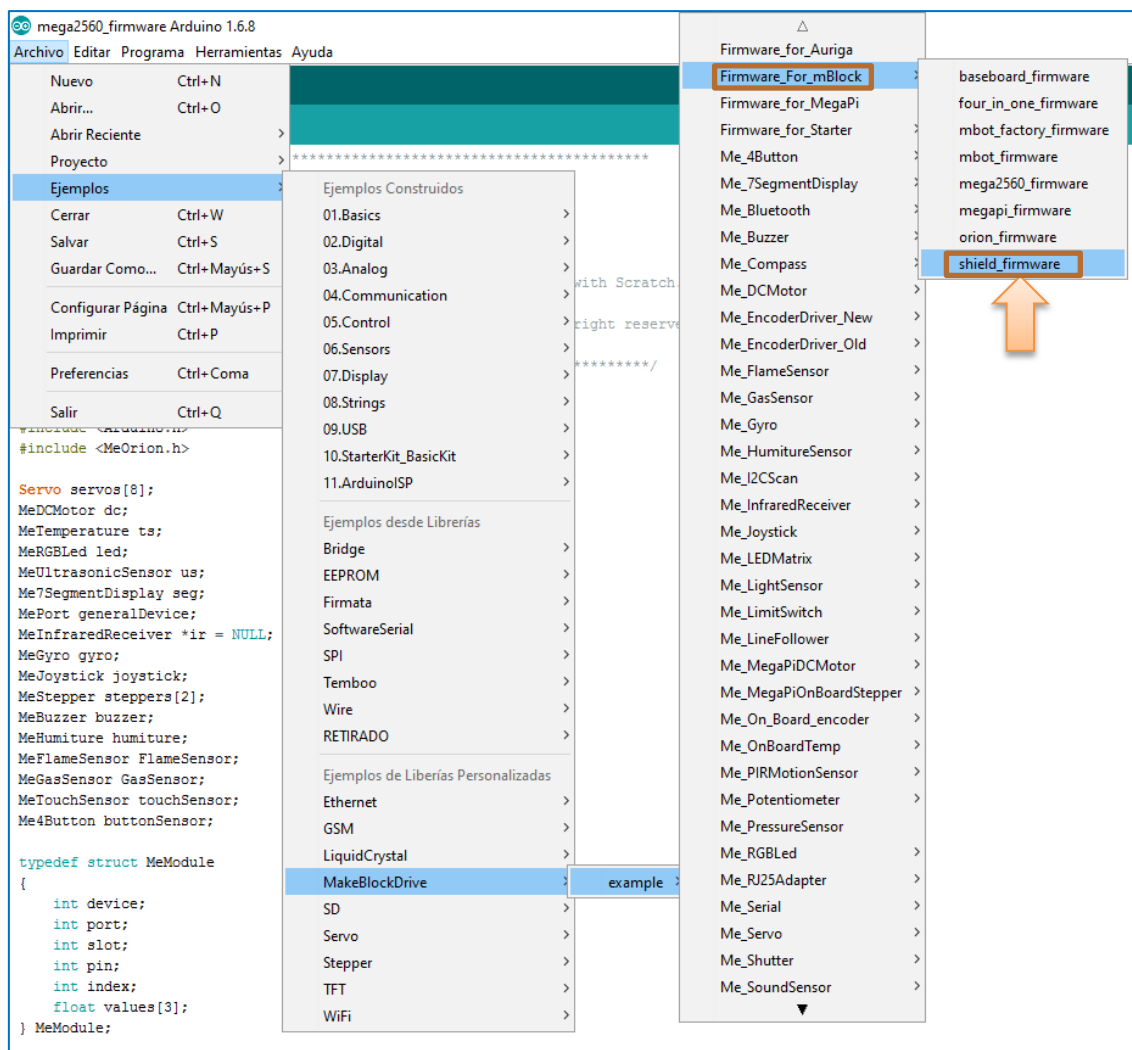
Si quisiéramos usar la placa Me Auriga en lugar del shield de Arduino conectando el sensor de gas al puerto 6 de la placa, los únicos cambios que debemos hacer se muestran en la siguiente imagen:

Divirtiéndome con mBot Ranger



Divirtiéndome con mBot Ranger

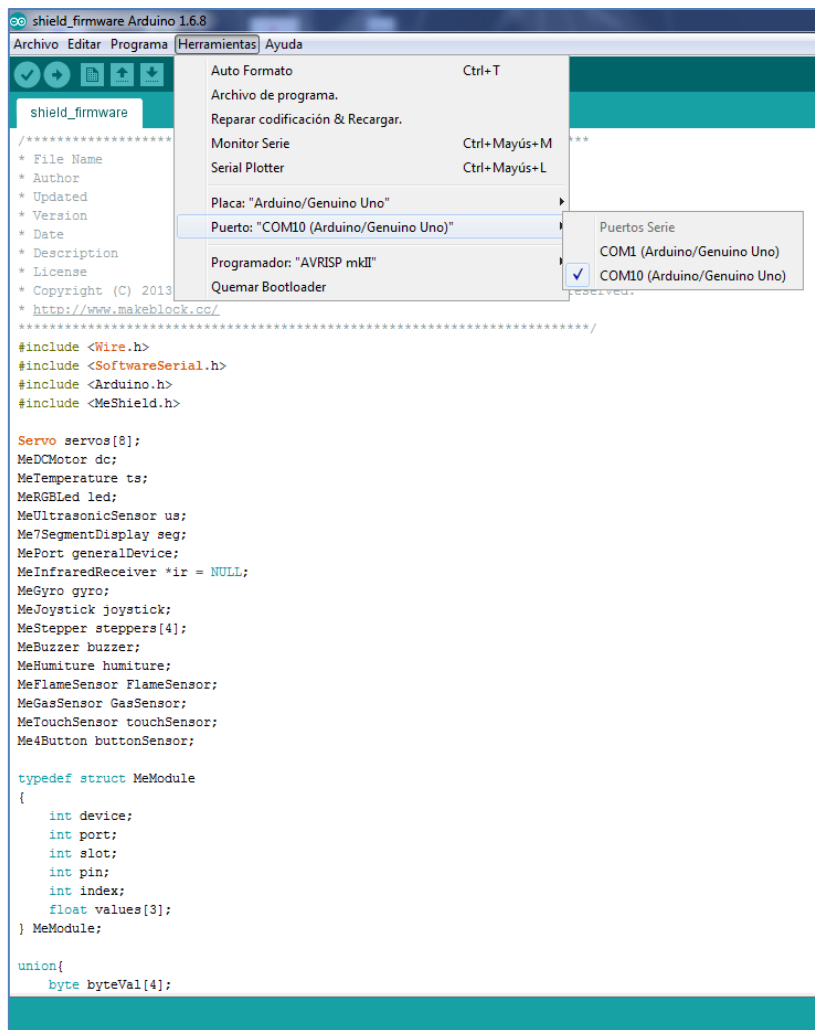
También podemos programar la placa Arduino Uno⁴ dentro de mBlock. Para ello debemos cargarle el firmware determinado, llamado “*shield_firmware*” a la placa Arduino Uno desde el IDE de Arduino:



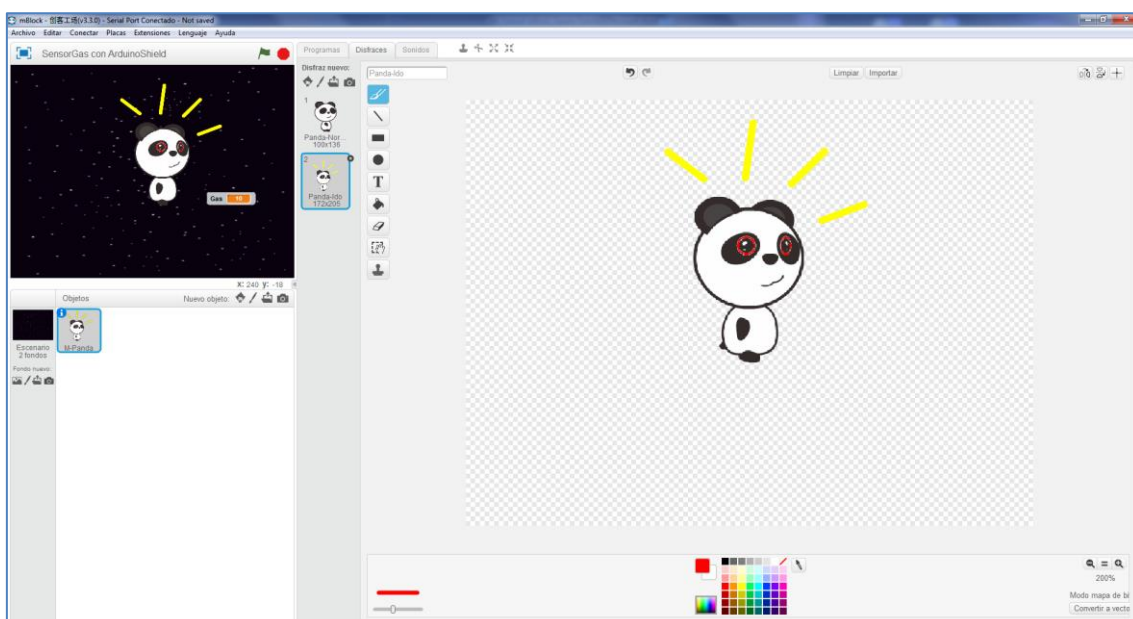
Tras abrirlo y seleccionar la placa Arduino Uno y su puerto serie de conexión, sólo debemos cargar el firmware en la misma. De esta forma, podemos programar la placa Arduino Uno con mBlock, con los comandos de scratch:

⁴ U otra placa Arduino como la Nano, Mega o Leonardo

Divirtiéndome con mBot Ranger



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:



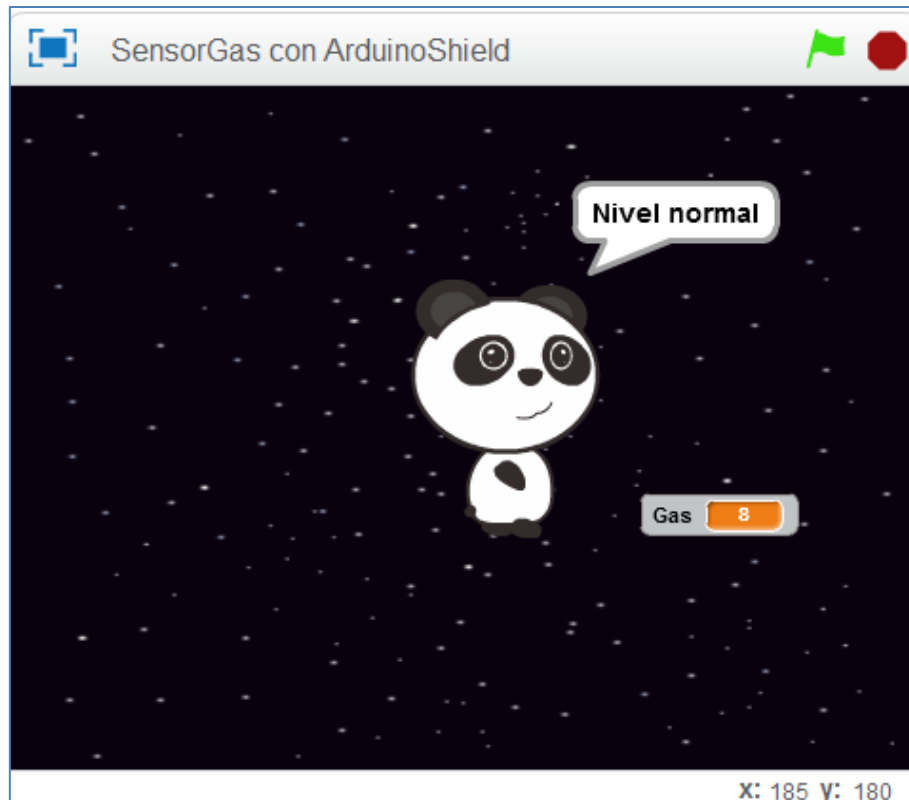
El pequeño script que vamos a ejecutar es el siguiente:



Script para la placa shield de Arduino

Al abrir el gas del mechero y acercárselo al sensor que está conectado al puerto 7 del shield de Arduino, si la concentración de gas que detecta es menor que 60, el objeto “M-Panda” dirá “Nivel normal” por 2 segundos. En caso contrario, si supera el valor numérico 60, cambia el disfraz y nos informa que está “gaseado”. Es decir, el nivel de gas es elevado. Las siguientes imágenes muestran las dos posibilidades que pueden darse:

Divirtiéndome con mBot Ranger



Concentración normal de gas (gas cerrado)

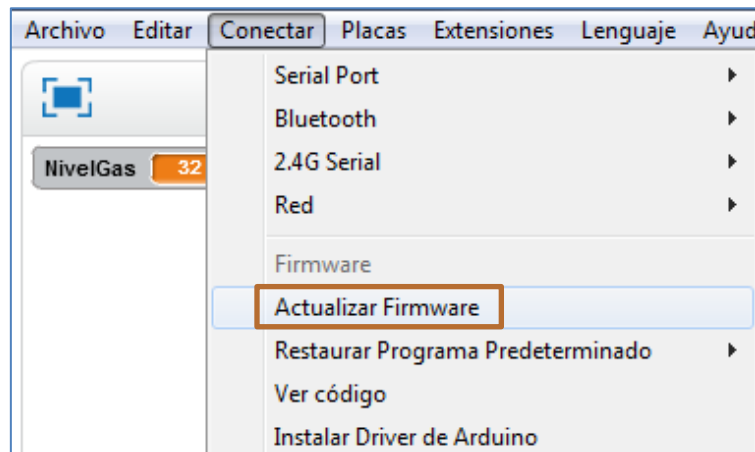


Concentración elevada de gas (gas abierto)

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

Divirtiéndome con mBot Ranger

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



Ahora, simplemente, nos queda programar. Si quisiéramos implementar el mismo ejemplo que hemos creado para la placa Arduino Uno con el shield de Arduino, como ahora usaremos la placa Me Auriga, el sensor de gas se puede conectar al puerto 7 y, por lo tanto, el script sería idéntico.

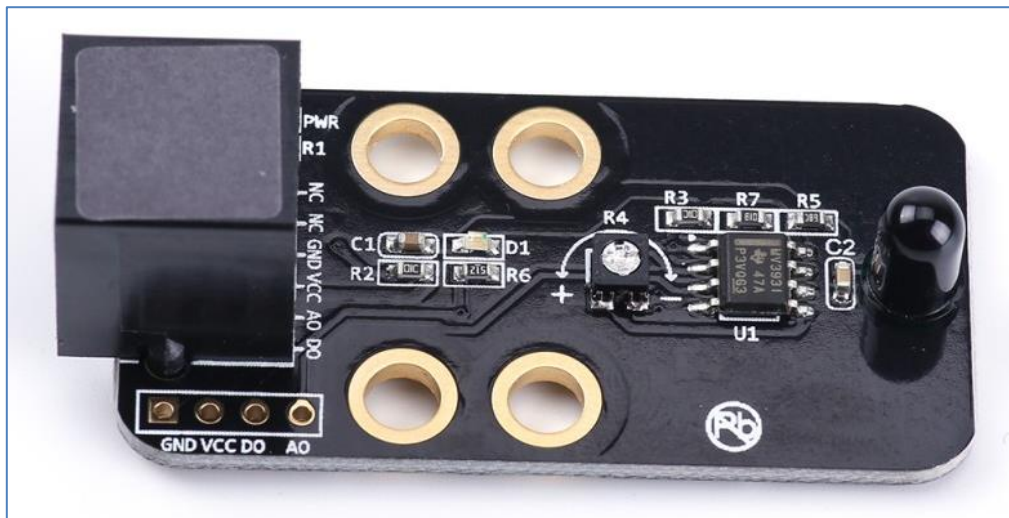
En los niveles normales de gas, tras una primera comprobación, el sensor, en su medida, nos ofrece el valor numérico bajo. Pero, si dejamos correr el gas del mechero, este valor numérico se dispara.

A veces nos interesa grabar el programa en la placa. Para ello debemos cambiar el comando de inicio de la bandera verde por el comando **Auriga Program** del bloque robots y cargarlo en la placa. En el siguiente ejemplo se usa la matriz de LEDs en el puerto 6 y el sensor de gas en el puerto 7. Se programa para que, si la concentración de gas es inferior a 60, en la matriz observemos una cara alegre, pero, en caso de que sea superior a 60, la expresión que se dibuja se muestra triste y nos avisa de la fuga de gas haciendo sonar la nota C4:



5.24. Sensor de llama

El sensor de llama o flama de la casa Makeblock es del tipo infrarrojo, con un comparador LM393 que permite obtener la lectura tanto como un valor analógico como de forma digital cuando se supera un umbral, que se regula a través de un potenciómetro que está incorporado en el módulo de llama. Este sensor es capaz de detectar una llama o una fuente de luz con una longitud de onda en el rango de 760nm a 1100nm. Este rango de longitud de onda que detecta no llega para considerarlo como un auténtico sistema de seguridad de llamas ya que puede ser afectado por la iluminación interior de la habitación, pero, si es suficiente para trabajar con él en pequeños proyectos didácticos, como por ejemplo, hacer sonar una alarma o activar un LED al detectar la llama de un mechero.



Sensor de llama o flama

Un sensor de llama permite detectar la existencia de combustión por la luz emitida por la llama. De hecho, la llama es un fenómeno de emisión de luz asociado a los procesos de combustión.

Este sensor se puede controlar por puerto analógico o digital, siendo su ángulo de detección máximo o cobertura de 60° y su tiempo de respuesta de 15µs. Su distancia de detección máxima es de 1m cuando la altura de la llama es de 5cm. Cuando detecta la llama o esa fuente de calor, se encenderá un indicador azul en el módulo del sensor de llama. Como su ID es negro se puede conectar a cualquier puerto de la placa Me Auriga numerado del 6 al 10.

Como ejemplo podemos programar el siguiente reto:

Realizar una simulación en el escenario del mBlock de una alarma de incendio de modo que, cuando encendamos la llama de un mechero, el programa nos avise que tenemos un fuego.

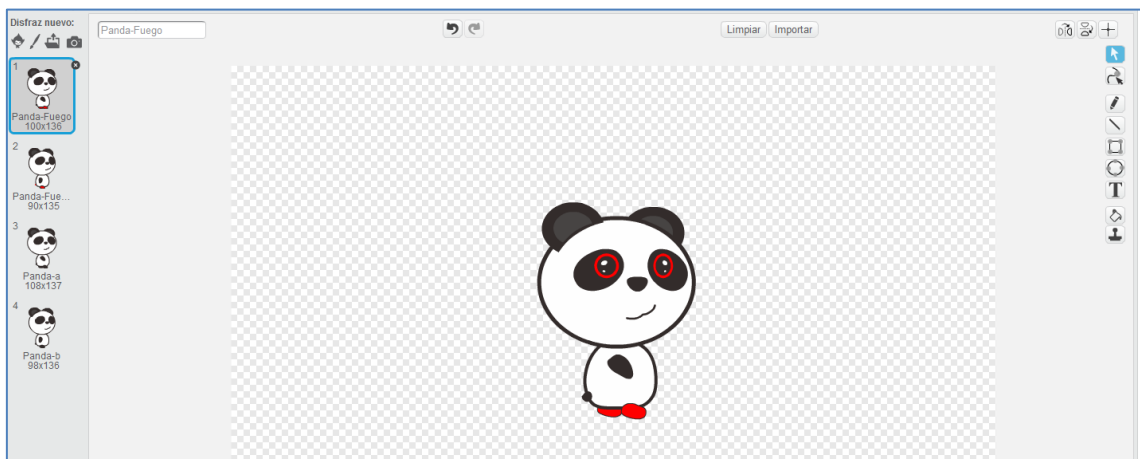
Para esta simulación diseño dos objetos. Uno será el propio del mBlock, el M-Panda, y el otro sprite será una animación de un fuego. En cuanto al escenario, utilizaré el fondo "forest".

Divirtiéndome con mBot Ranger



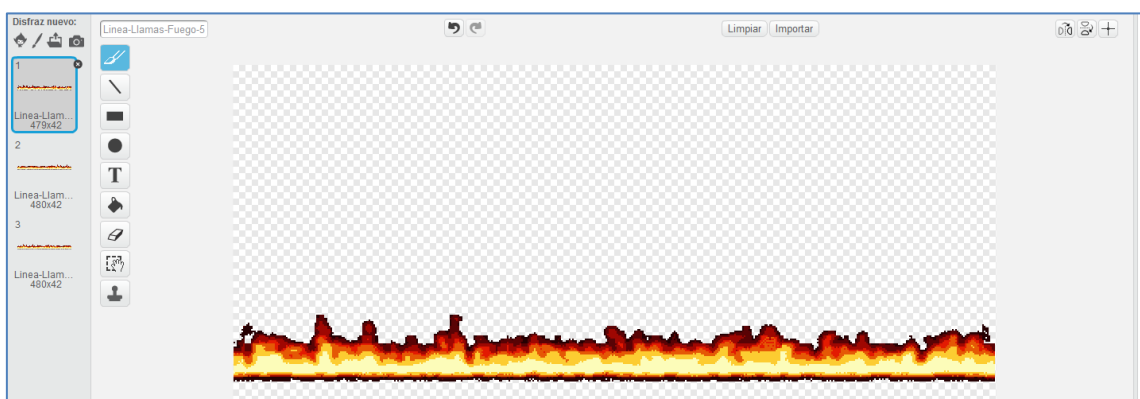
Objetos y escenario del programa

Para crear las animaciones, detección o ausencia del fuego, utilizo 4 disfraces en el objeto M-Panda:



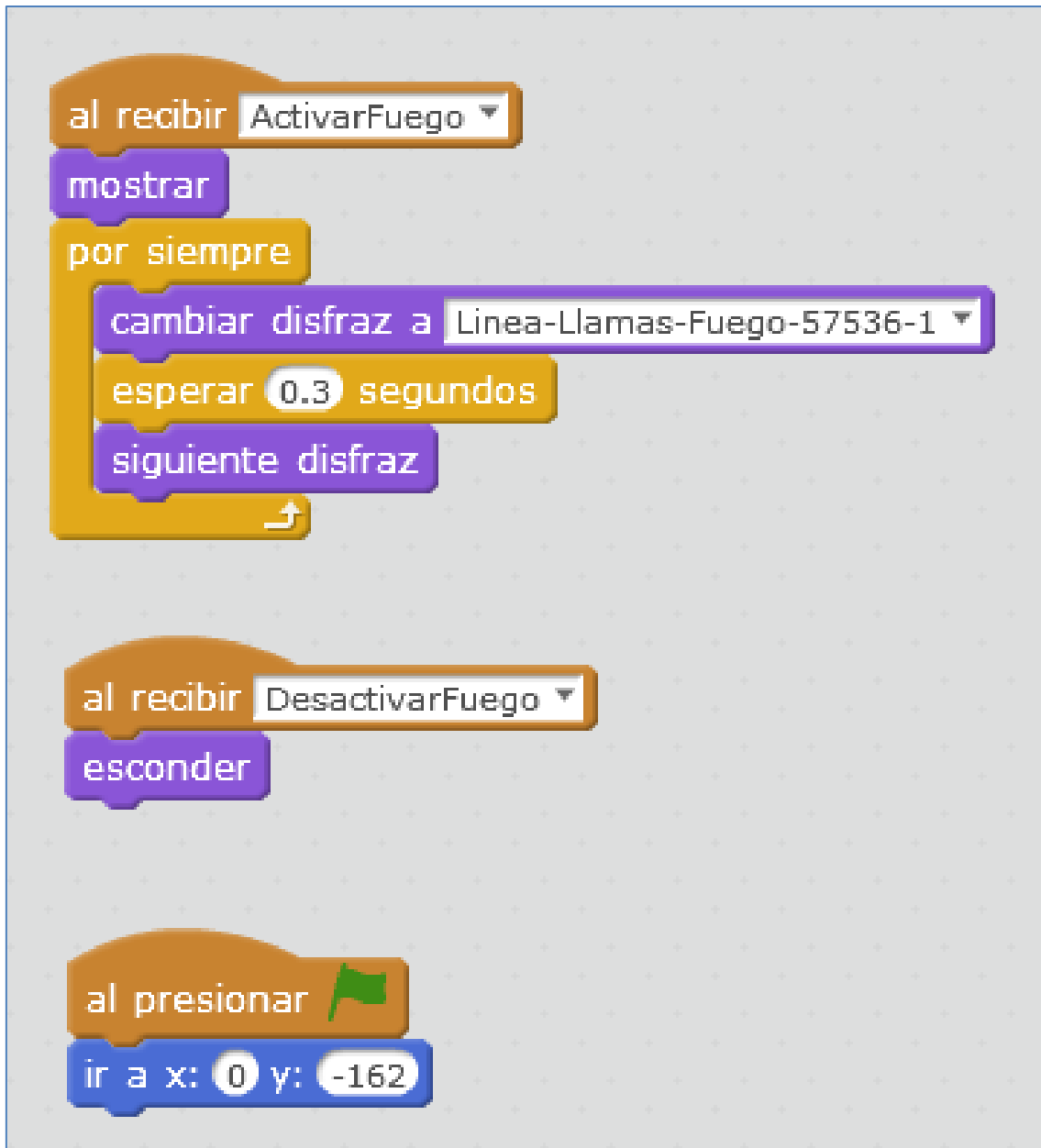
Disfraces del objeto M-Panda

La animación del objeto *Fuego* la creo a partir de un gif animado de tres disfraces:



Disfraces del objeto Fuego

Cuando se active el fuego, debe mostrarse la animación del mismo y cuando no tengamos llama, esta debe esconderse. Por lo tanto, el script del objeto *Fuego* que cumple estas condiciones es el siguiente:



Script del objeto "Fuego"

El script del objeto *M-Panda* llevará el peso del reto porque este sprite es el que detecta si hemos encendido o no la llama del mechero. El límite de detección lo he fijado a 400 (este límite dependerá de la iluminación de la habitación). El testeo del mismo lo realizará la variable "llama". En el caso de que detecte fuego ($llama < 400$), se visualizará la animación de ambos objetos (fuego y M-Panda) y el mensaje "Fuego!" en el escenario; así como, sonará la nota C4. En caso contrario, ausencia de llama, la medida del sensor de flama supera el valor numérico 400, escondiendo la animación del fuego y mostrando el mensaje "Todo tranquilo" con los disfraces característicos del objeto M-Panda.

Las siguientes imágenes muestran la simulación en el mBlock cuando detecta llamas y cuando no hay combustión:



Presencia de fuego



Ausencia de fuego

El script para el objeto M-Panda se divide en dos partes. En la primera programo el sensor de flamas y activo el disfraz correspondiente del objeto *Fuego*.



Primera parte (llamas)

La segunda parte del script (ver siguiente imagen) en el sprite M-Panda nos servirá para en el escenario (simulación) se muestren unos determinados disfraces en el objeto:

Dos disfraces para el caso que detecte llamas y otros dos diferentes para el caso contrario, no hay fuego.



Segunda parte (cambio de disfraces)

5.25. Sensor de luz

Desarrollado sobre la base del principio de efecto fotoeléctrico en semiconductores, este módulo 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

Divirtiéndome con mBot Ranger

El comando que se usa para programar el sensor de luz, esté o no incrustado en el anillo de LEDs es el mismo (ver imagen derecha). En el desplegable del comando podemos escoger si queremos programar los sensores de la placa (*on board 1* y *on board 2*) o si queremos programar el módulo sensor de la casa Makeblock conectado a un determinado puerto de la placa Me Auriga (del 6 al 10).



En el caso del módulo sensor de luz, este componente podemos conectarlo a cualquier puerto numerado del 6 al 10 en la placa Me Auriga, ya que su ID es negro. 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:



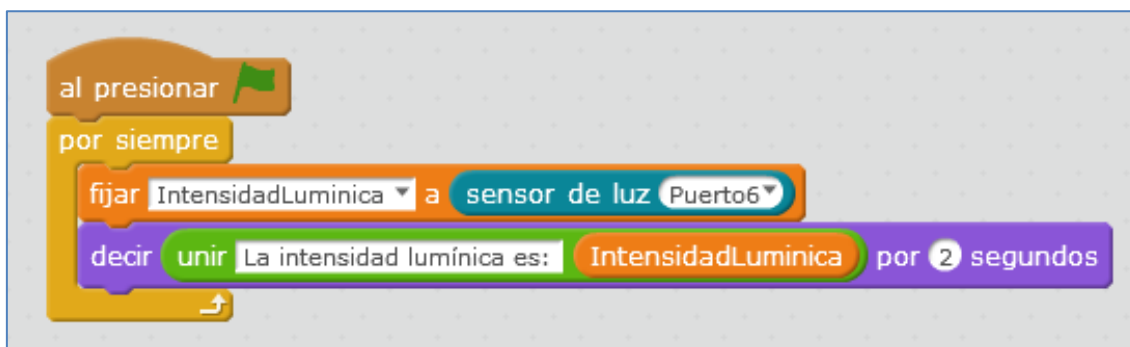
Luminosidad con la luz apagada



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

5.26. 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”:

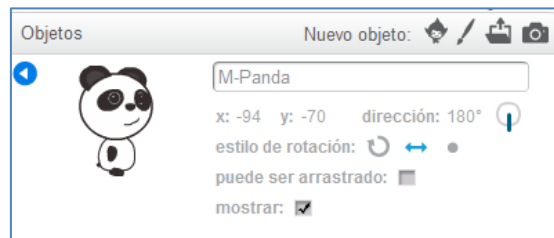


Módulo de 4 pulsadores

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 Arduino Uno. En el caso de la placa Auriga, este módulo puede ir conectado a cualquiera de los puertos numerados del 6 al 10.

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

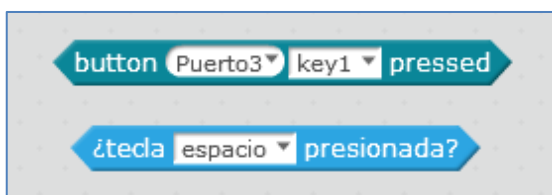
Si pulsamos el pulsador “key1”, el objeto “M-Panda”, que presenta estilo de rotación derecha-izquierda, se moverá 10 pasos hacia la derecha. Al pulsar “key3” se moverá 10 pasos hacia la izquierda. Y, al pulsar “key4” o “key2” se desplazará, respectivamente, 10 pasos verticalmente hacia arriba o hacia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



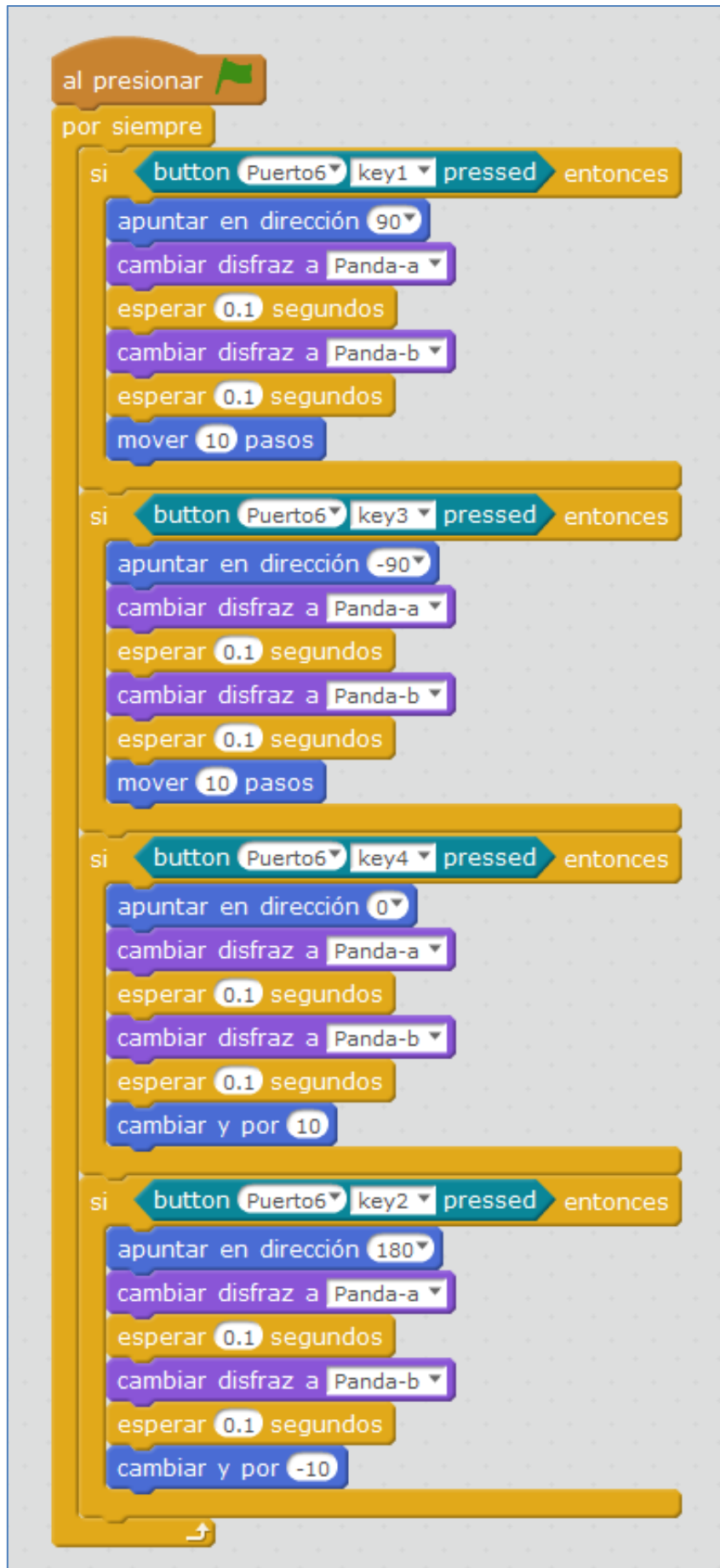
Estilo de rotación del objeto “M-Panda”

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 6 de la placa Auriga del robot, y que implementa el programa de movimiento requerido sobre el objeto “M-Panda”, sería el siguiente:



5.27. Pantalla TFT LCD 2.2

La pantalla TFT LCD tiene un tamaño de 2.2 pulgadas y una resolución de 220×176. 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. En cuanto a la placa Auriga, nos vale cualquier puerto numerado entre el 6 y el 10.



Parte trasera de la pantalla TFT LCD



Parte frontal de la pantalla TFT LCD

Divirtiéndome con mBot Ranger

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):



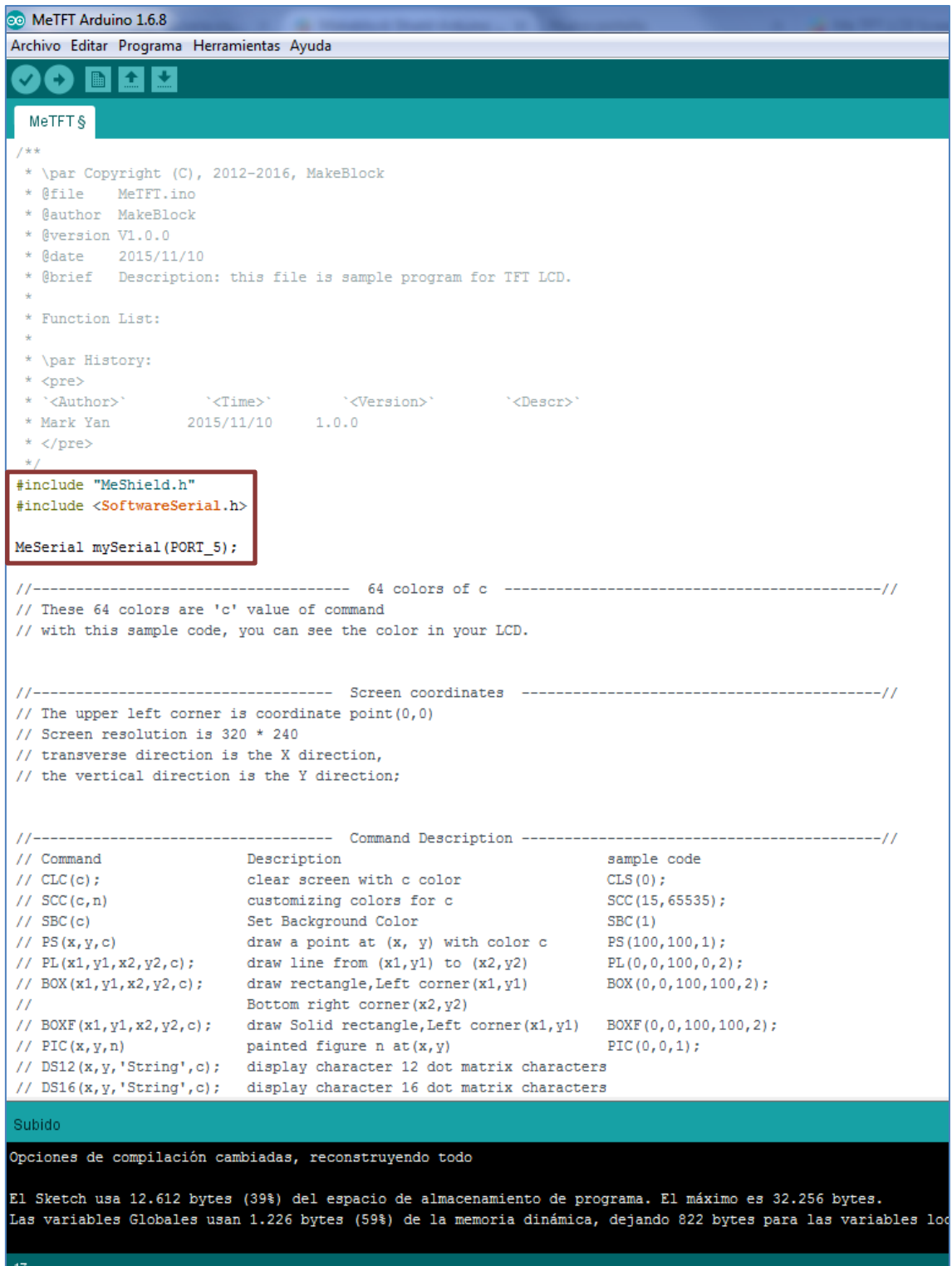
Pantalla TFT LCD	Arduino Uno
TX	RX (pin digital 0)
RX	TX (pin digital 1)
Vcc	5V
GND	GND

NOTA: La casa Makeblok dispone del cable Dupont que facilita la conexión de un RJ25 a pines hembra para las placas Arduino Uno (si no se dispone del Shield de Makeblock para esta placa) y otras.

Las siglas TFT LCD significan "*Thin Film Transistor Liquid Crystal Display*". Es decir, *pantalla* de cristal líquido y transistor de película fina que mejorará la calidad de la imagen. Su principal ventaja radica en que el coste de producción de estas pantallas no es elevado. Relativo a su principal desventaja, este tipo de pantallas necesitan de mucha energía y, en consecuencia, la vida de la batería que las alimenta queda reducida en pantallas de gran tamaño. Como la nuestra es de 2,2 pulgadas, su tamaño no será un hándicap hacia el consumo abusivo de baterías de nuestro robot.

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 Ranger



```
MeTFT Arduino 1.6.8
Archivo Editar Programa Herramientas Ayuda

MeTFT $

/**
 * \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);
//                               Bottom right corner(x2,y2)
// 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)

Divirtiéndome con mBot Ranger

- ❖ 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 espaciado 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

5.28. 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.

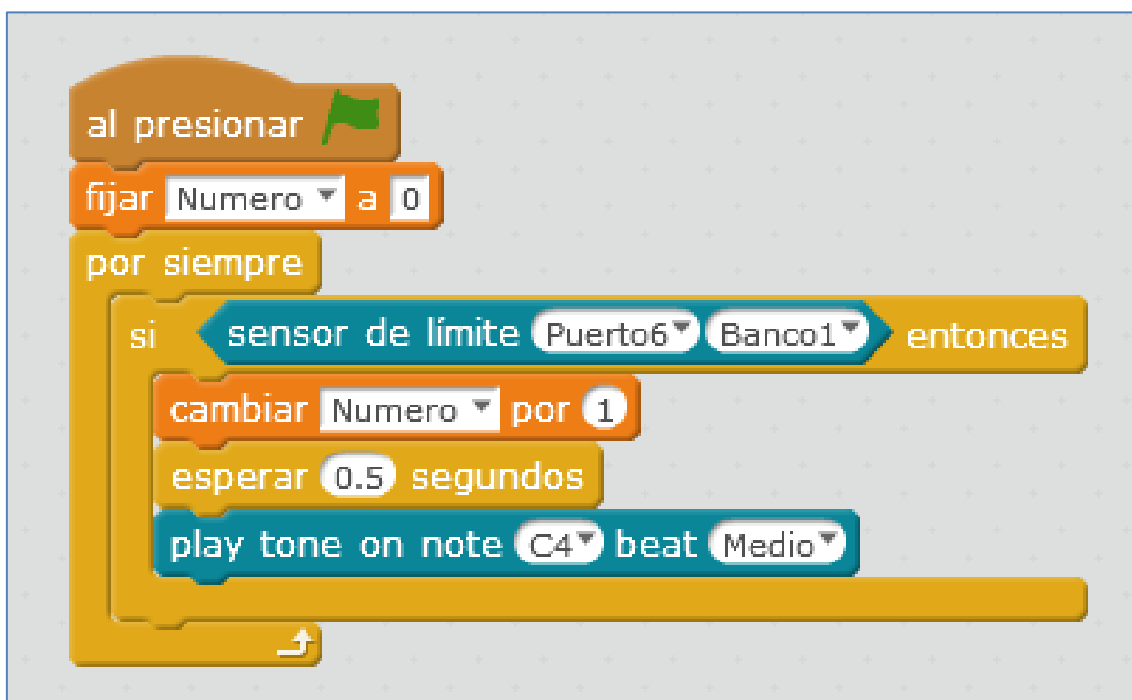


Micro Switch B

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 “Numero”. La placa a la que se ha conectado el Micro Switch (a través del adaptador) es la Auriga del mBot Ranger, usando el *Banco1* del adaptador y estando éste conectado al *Puerto6* de la placa.

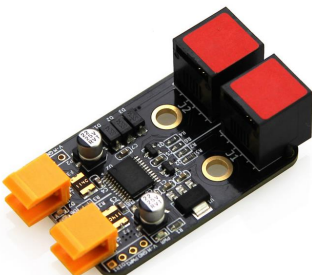
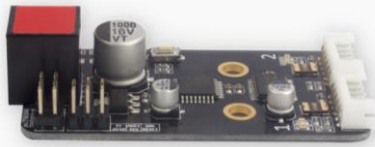
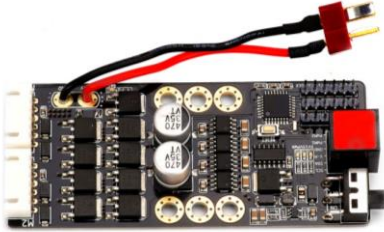
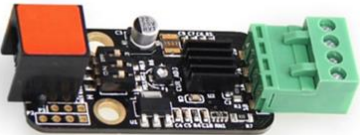
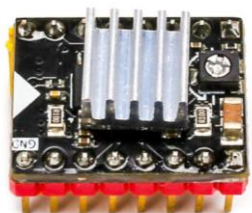
Al hacer contacto el final de carrera, este envía una señal que hará que el contador aumente en una unidad y suene la nota C4:



Divirtiéndome con mBot Ranger

5.29. Módulos controladores de motores

La casa Makeblock dispone de un amplio abanico de controladores de motores:

		
Controladora de motor paso a paso Microstep	1. Controladora de motores DC dual	2. Controladora de motores encoder
		
Controladora de motores encoder/DC para MegaPi	3. Controladora de motores encoder de alto voltaje	4. Controladora de motor paso a paso
	En este apartado explicaremos por encima las controladoras de motores <i>numeradas del 1 al 4</i>.	
Controladora de motores paso a paso para MegaPi		

Tipos de controladoras Makeblock

Las diferencias entre los controladores de motores numerados como 1 y 2 en la tabla anterior son mínimas. He hecho, se basan en el mismo driver TB6612 con puentes H que llevan transistores MOSFET. Con la controladora numerada por 1 podemos controlar hasta 2 motores DC y con la numerada por 2 podemos manejar hasta 2 motores encoder ópticos, que son los que vienen de serie en el robot Ranger.

La controladora de motores numerada por 3 es la versión más actual de su predecesora, la controladora de motores encoder. Su driver es diferente, utilizando MOS de alta potencia. Además, dispone de pines I²C, SPI y UART y un sistema de protección que evita que el circuito se queme.

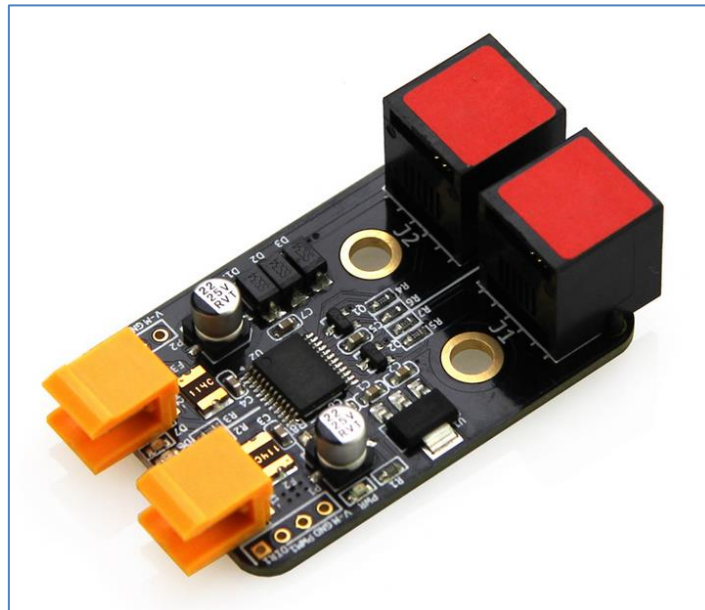
Divirtiéndome con mBot Ranger

En este documento, de los tres controladores anteriores, desarrollaremos la controladora de motores DC dual.

En el punto 5.13 de este documento ejemplificábamos la garra robótica. En su montaje con la placa Auriga usábamos la controladora de motores DC dual. Esta controladora de las serie 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 Dual 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. Es decir, el puerto J1 de la controladora gobierna al motor de continua M1 y el J2 con el motor M2. 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 a un puerto de color rojo. En la placa Auriga disponemos de cuatro puertos con ID rojo numerados del 1 al 4. No está de más recordar que la placa Auriga necesita estar encendida para poder alimentar a los motores. Por lo tanto, debemos asegurarnos que la luz "PWR" esté encendida para poder usar los motores.



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:



Divirtiéndome con mBot Ranger

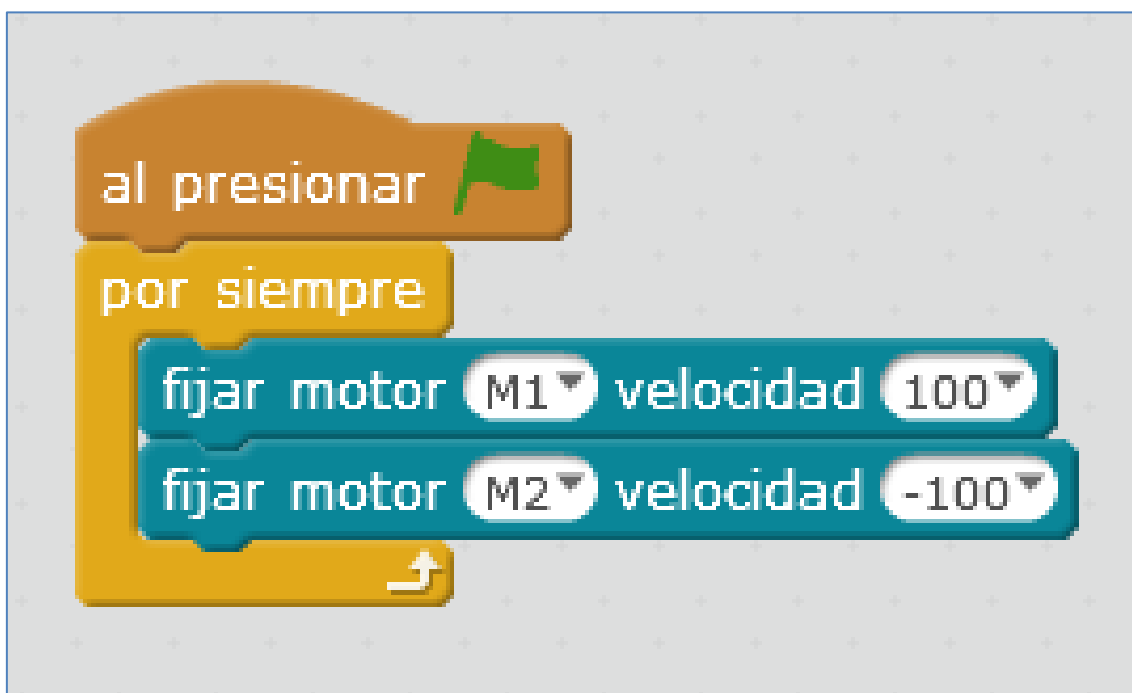
V-M	Potencia del motor (6V-12V)
GND	Tierra
PWM	Modulación por ancho de pulso (regulación de velocidad)
DIR	Control de dirección

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 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 en una placa mCore, con sentidos de giro contrarios (por ejemplo), deberíamos activar el siguiente script:



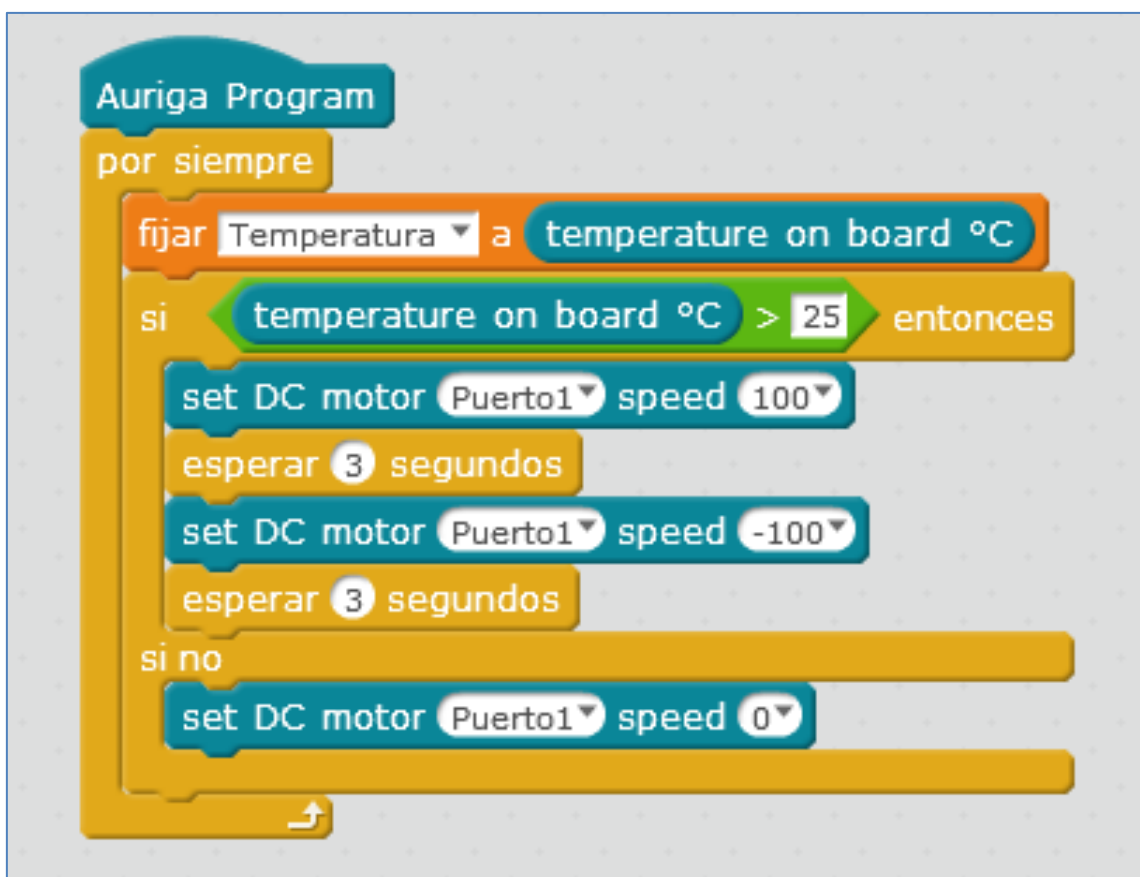
Como ejemplo en la placa Auriga podemos conectar el módulo 130DC con hélice a una controladora de motores DC dual e intentar diseñar y programar el siguiente reto:

Crear un ventilador que se accione sólo si la temperatura ambiente supera los 25°C.



130 motor DC con hélice

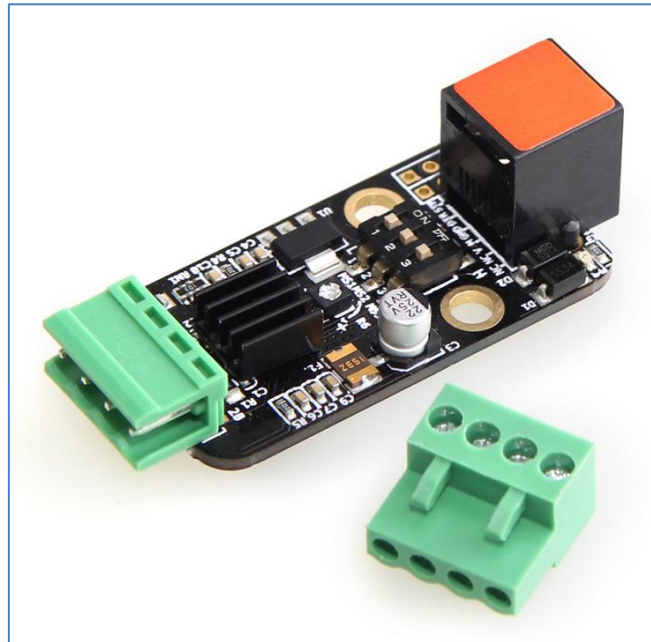
El script que controlará nuestro ventilador podría ser el siguiente:



En este caso, he decidido que la hélice del motor gire, alternado el sentido de giro, cada 3 segundos. Para testear la temperatura de la habitación uso el sensor de temperatura que viene incrustado en la placa Me Auriga y su medida se la asocio a la variable "Temperatura".

Divirtiéndome con mBot Ranger

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 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.

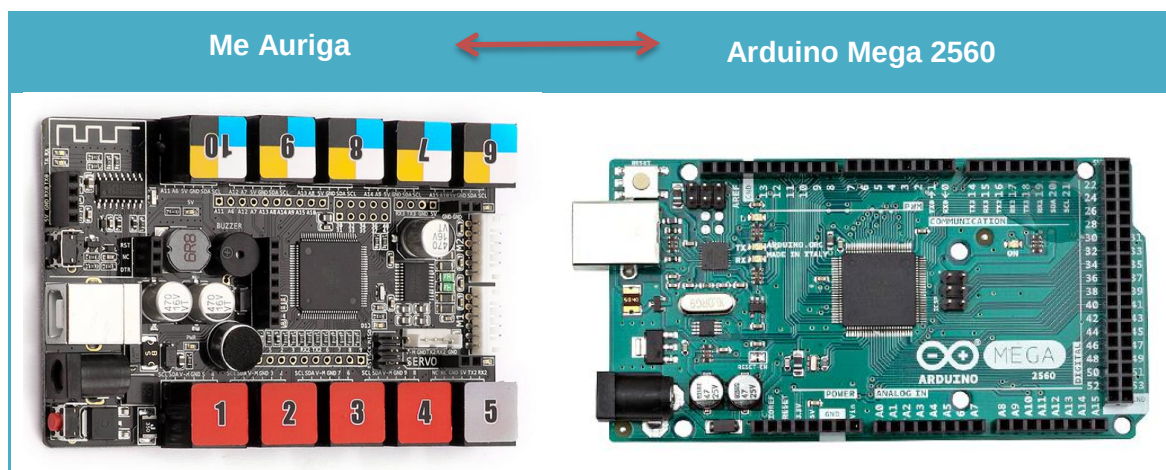


Divirtiéndome con mBot Ranger

6. Otras placas con mBlock

En los talleres de Tecnología a veces disponemos de placas Arduino. Por lo general son Arduino Uno pero podemos disfrutar de otras. Si comparamos estas placas con las de la casa Makeblock, vemos que la placa mCore del robot mBot no es más que una placa Arduino Uno. Y lo mismo ocurre con la placa Me Auriga, que no es más que una Arduino Mega 2560 (ver 3.3.). Esto quiere decir que tenemos un dos en uno y todo lo que se ha referido en este libro para la placa mCore se puede implementar con una Arduino Uno, y por ende, todo lo que se ha explicado para la placa Me Auriga, puede llevarse a cabo con una placa Arduino Mega 2560. En cuanto a su programación, para cualquiera de estas cuatro placas, podemos hacerlo tanto desde mBlock (scratch) como desde el IDE de Arduino.

A mayores, la casa Makeblock ha sacado al mercado el Shield para Arduino Uno. Un shield muy cómodo con 10 conectores RJ25 para encajar, de forma rápida y sencilla, la larga lista de componentes que nos ofrece la casa Makeblock para una Arduino Uno. Por ahora no disponemos del shield equivalente para una Arduino Mega, pero entiendo que a todo se llegará.



Me Auriga / Arduino Mega 2560

En conclusión, podemos afirmar que una placa Auriga no es más que placa Arduino Mega2560 con unos componentes electrónicos ya incrustados a ella y con la salvedad de que usa diferentes conectores. Me explico, la placa Auriga dispone de conectores RJ25 y una Arduino Mega 2560 carece de ellos.

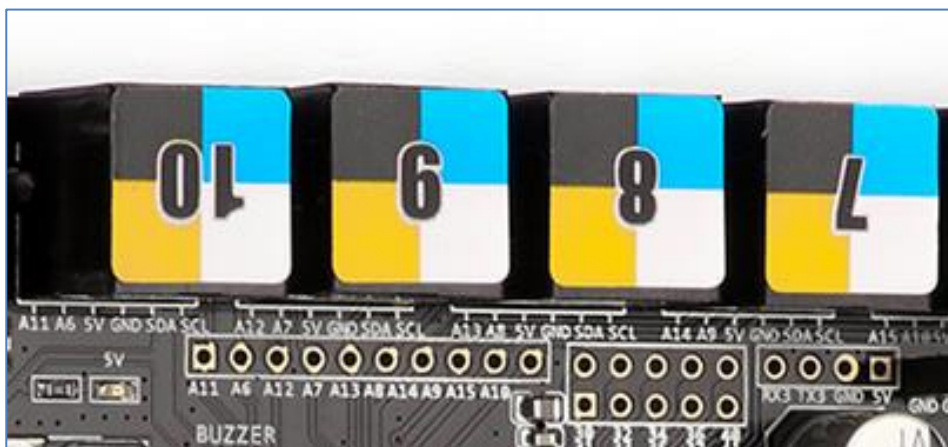
Para conectar, de forma sencilla y rápida, componentes electrónicos que dispongan de conexiones RJ25 a una placa Arduino Mega 2560, e incluso cualquier otro que deseemos utilizar en la misma, la casa Makeblock sacó al mercado el Adaptador RJ25 y el conector RJ25 a *Dupont*. Este último es un conector 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.

Divirtiéndome con mBot Ranger



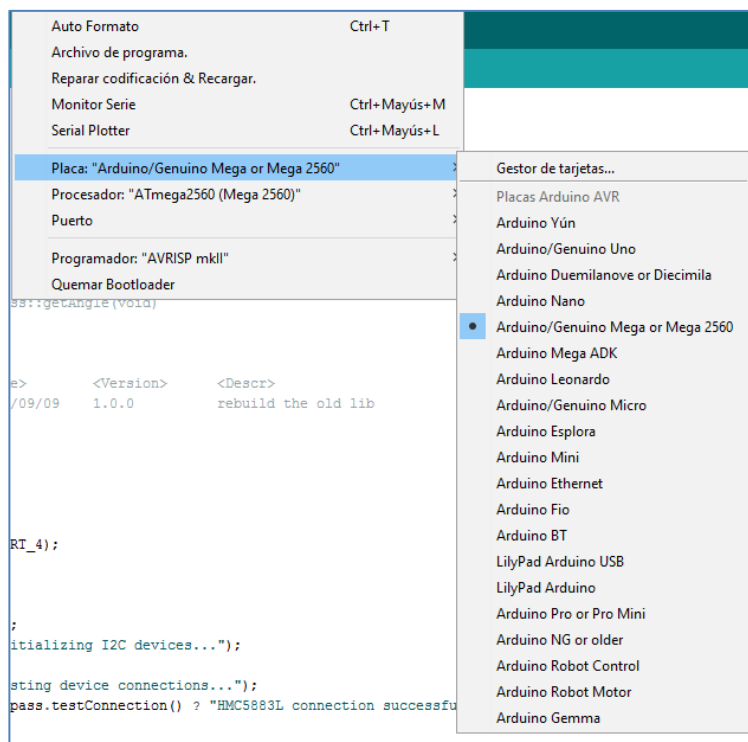
Conector RJ25 a Dupont

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:



Serigrafía en Me Auriga con Arduino Mega 2560

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



Como la placa me Auriga es una Arduino Mega 2560, podríamos cargar cualquier programa desde el IDE de Arduino a la placa Auriga, sin más que seleccionar la opción Arduino/Genuino Mega or Mega2560 con el puerto COM correspondiente (ver imagen izquierda).

Divirtiéndome con mBot Ranger

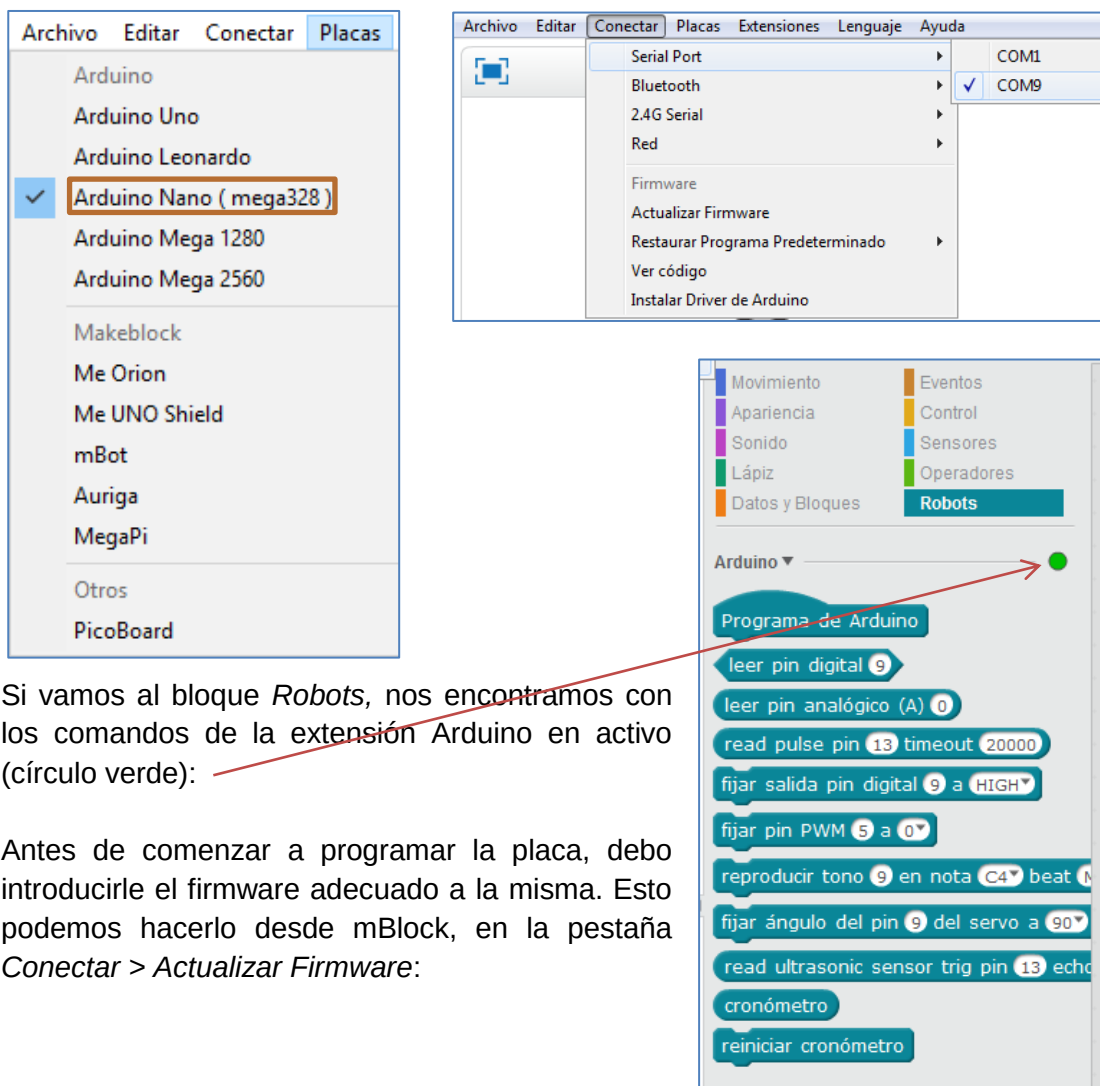
Pero aún hay más. Si nos fijamos en las opciones de la pestaña *Placas* del software mBlock, podemos programar una gran variedad de placas con scratch. Dentro del universo arduino tenemos: Arduino Uno (ya ejemplificada en muchas partes de este documento), Leonardo, Nano y Mega (también ejemplificada en este tutorial). Pero, a mayores, podemos programar la conocida tarjeta de sensores PicoBoard e incluso una Raspberry Pi con el shield MegaPi.

A continuación mostraremos sólo tres ejemplos: uno con la tarjeta Nano, otro con la PicoBoard y el último con la tarjeta Mega.

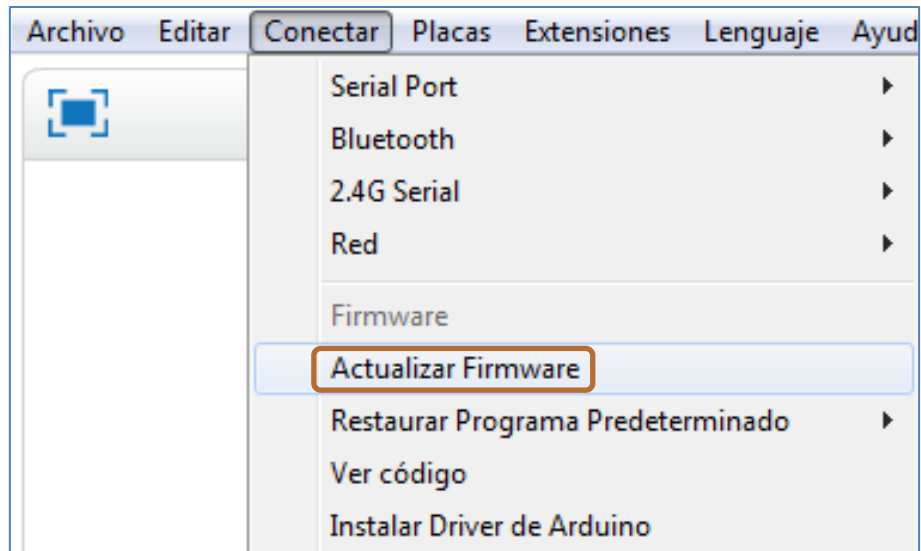
6.1. Ejemplo 1: Semáforo con Arduino Nano sobre mBlock

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 cualquiera de esas placas 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*):

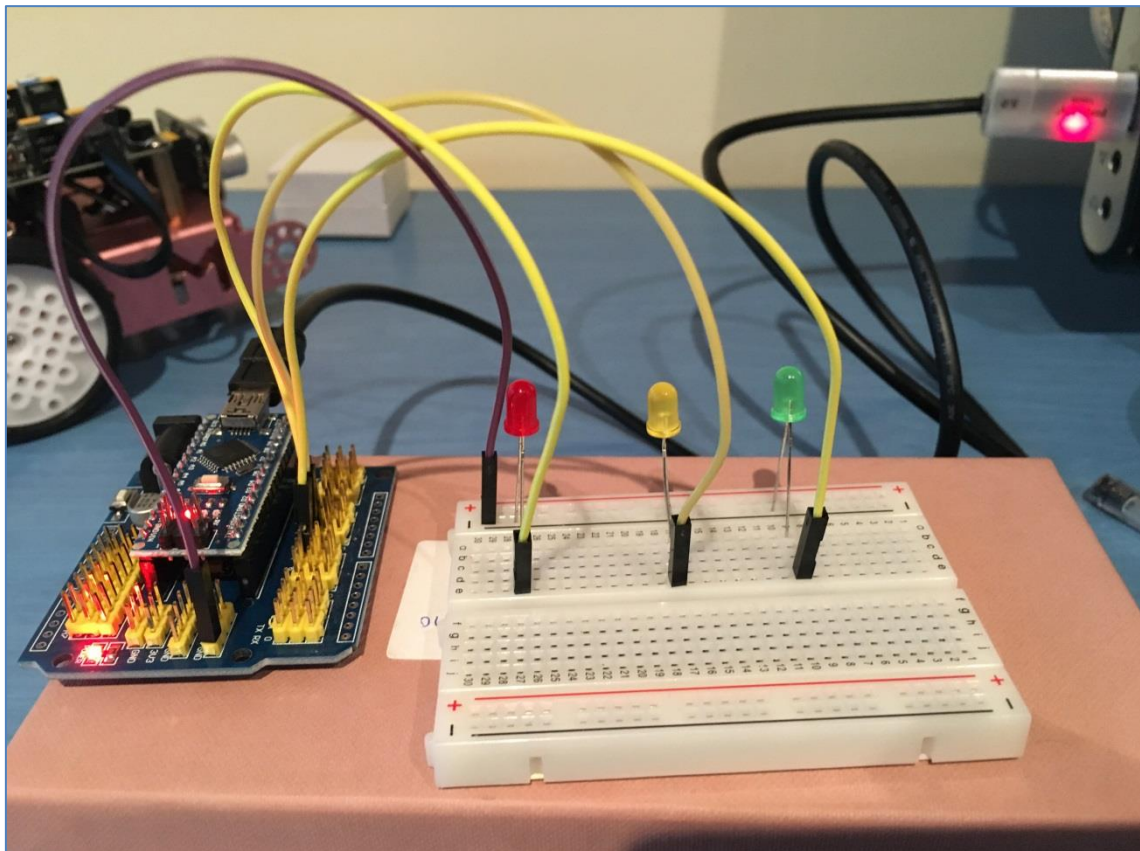


Divirtiéndome con mBot Ranger



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:



Divirtiéndome con mBot Ranger

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.
- Después 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.

Pin 7= Diodo Rojo
Pin 8= Diodo Ámbar
Pin 9= Diodo Verde

Primero, el diodo ROJO se enciende durante 4 segundos. Después el ÁMBAR se enciende y se apaga intermitentemente durante 3 segundos y, finalmente se enciende el VERDE durante 8 segundos. El programa finaliza apagándolos.

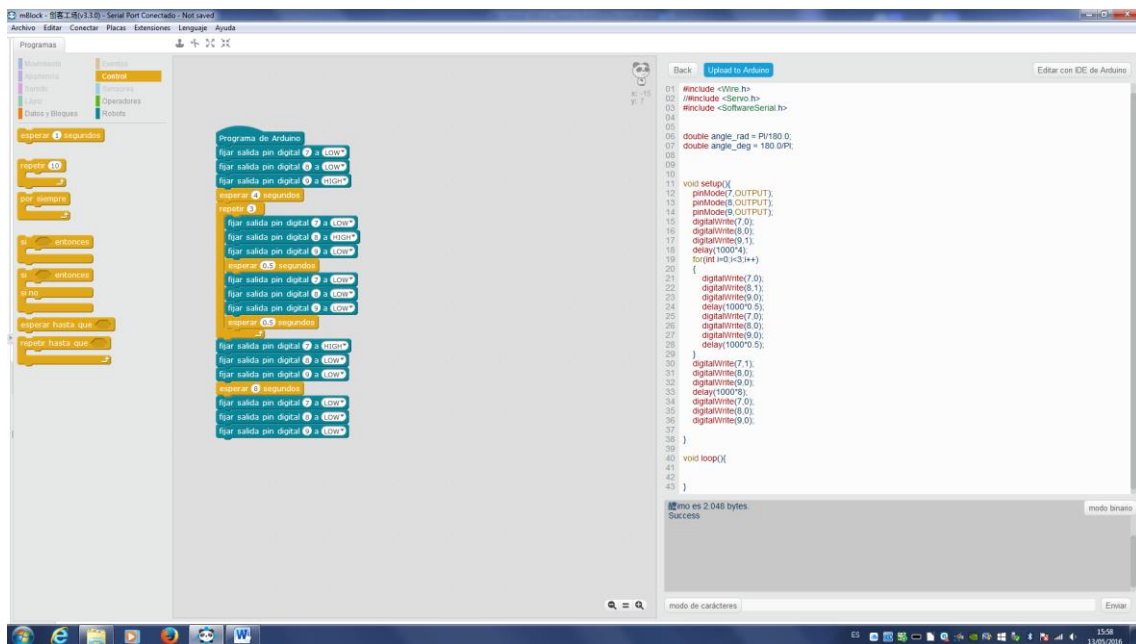
Script Semáforo

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 Ranger



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.

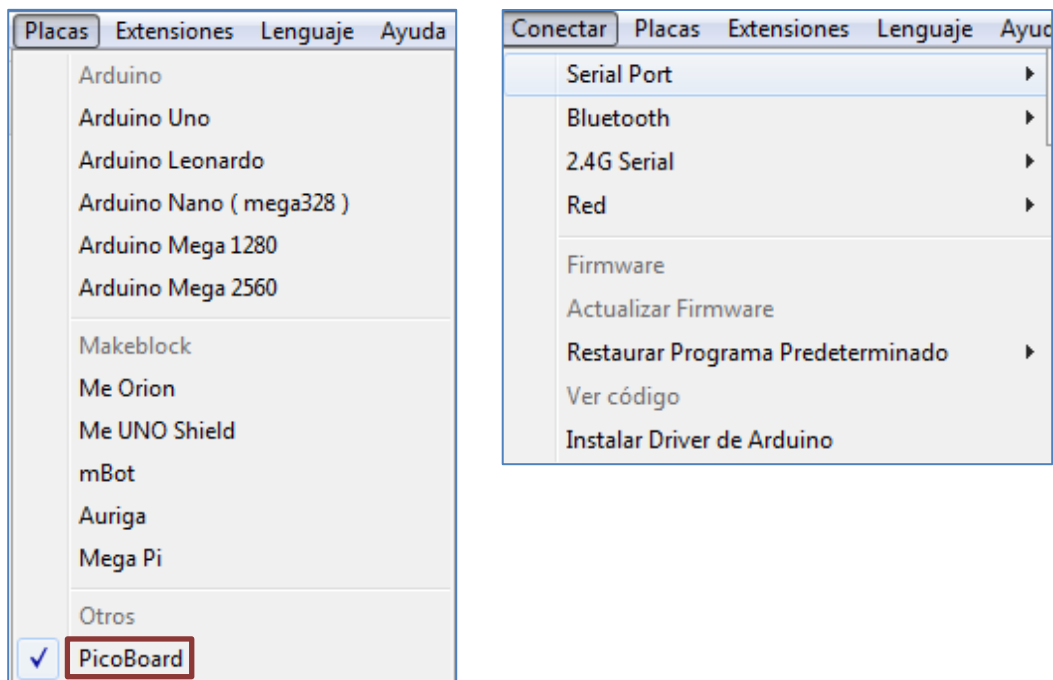
6.2. Ejemplo 2: Naves invasoras del espacio con PicoBoard bajo mBlock

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

Divirtiéndome con mBot Ranger

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*:



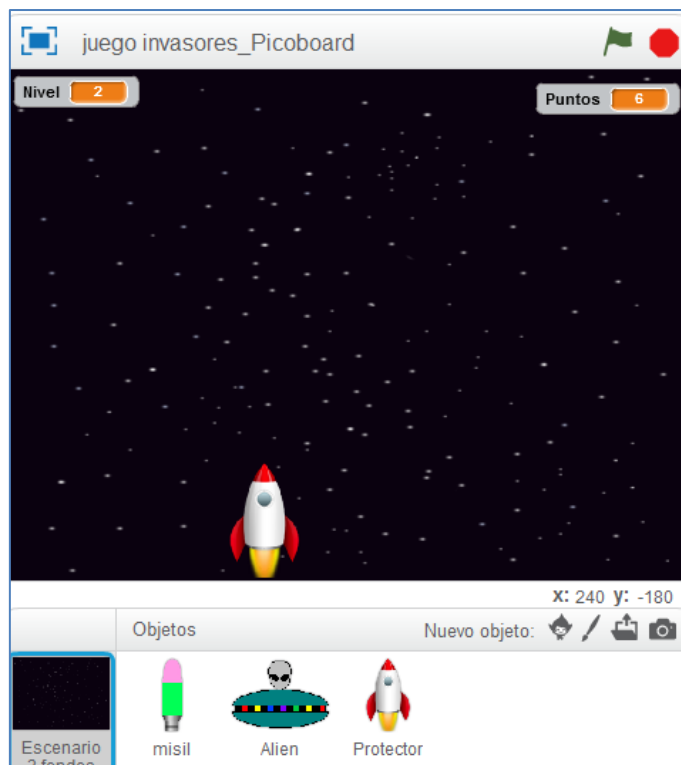
En el mercado actual existen diferentes modelos de picoboard. En la imagen inferior podemos ver el modelo más antiguo, modelo que soporta el software mBlock. También es compatible con el programa la siguiente versión de la picoboard. Sólo se necesita instalar el driver de cada respectiva tarjeta de sensores.



PicoBoard (Scratch Sensor Board)

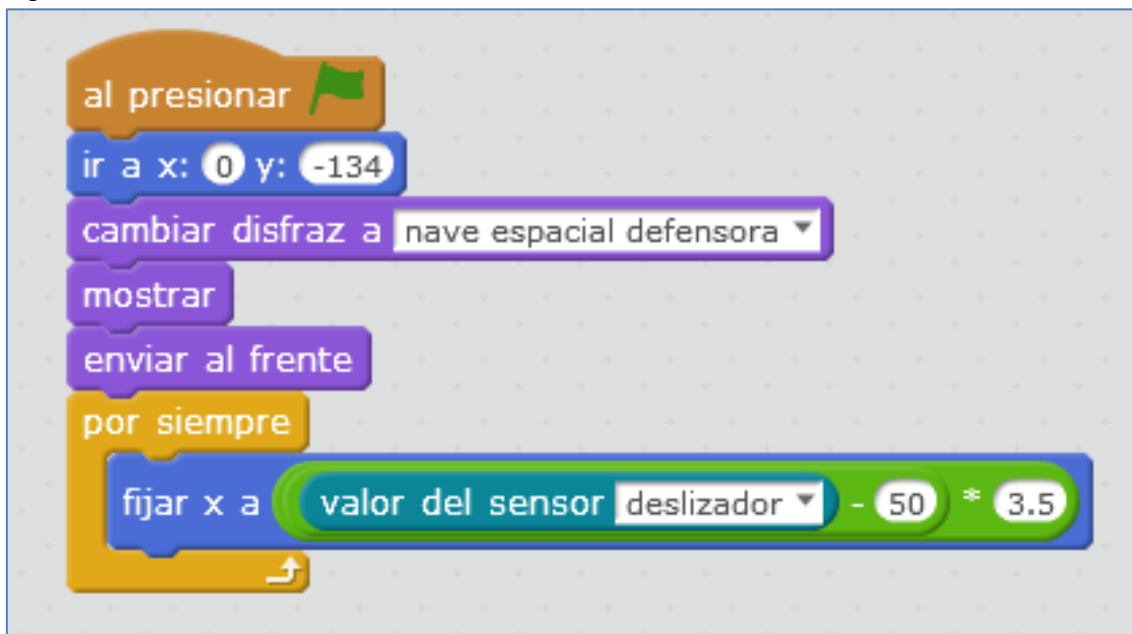
Divirtiéndome con mBot Ranger

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.



Objetos del juego

La nave protectora 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 Ranger

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

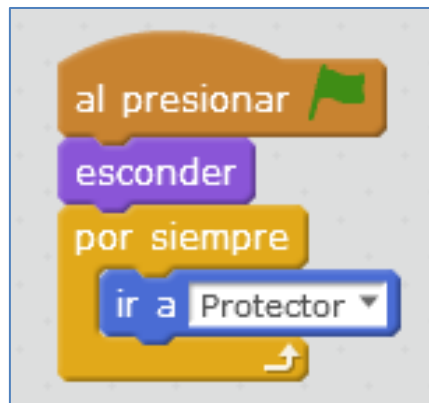
Si fijo la X al valor del deslizador, sólo se moverá 100 píxeles, desde la abscisa $x=0$ a $x=100$. Pero me interesa que recorra todo el escenario. Para que el objeto se vea completo y no cortado, decido que mi variable X del escenario, con el deslizador, no puede superar 175 píxeles, ni puede ser menor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

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

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

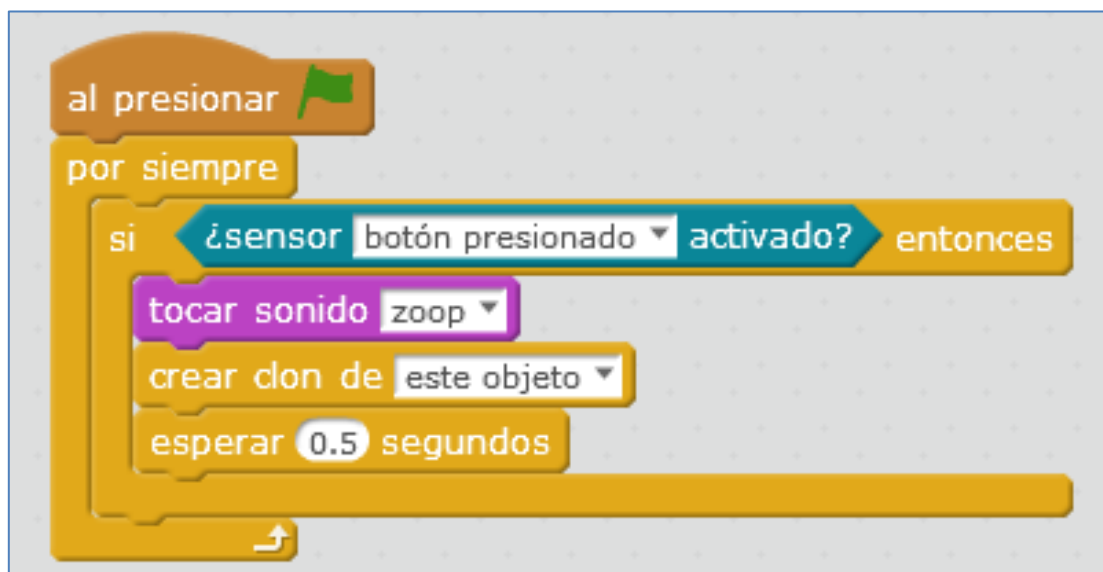
Con estas ecuaciones, el factor de corrección (X) es el mismo, y pasa a tomar el valor 3,5.

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”

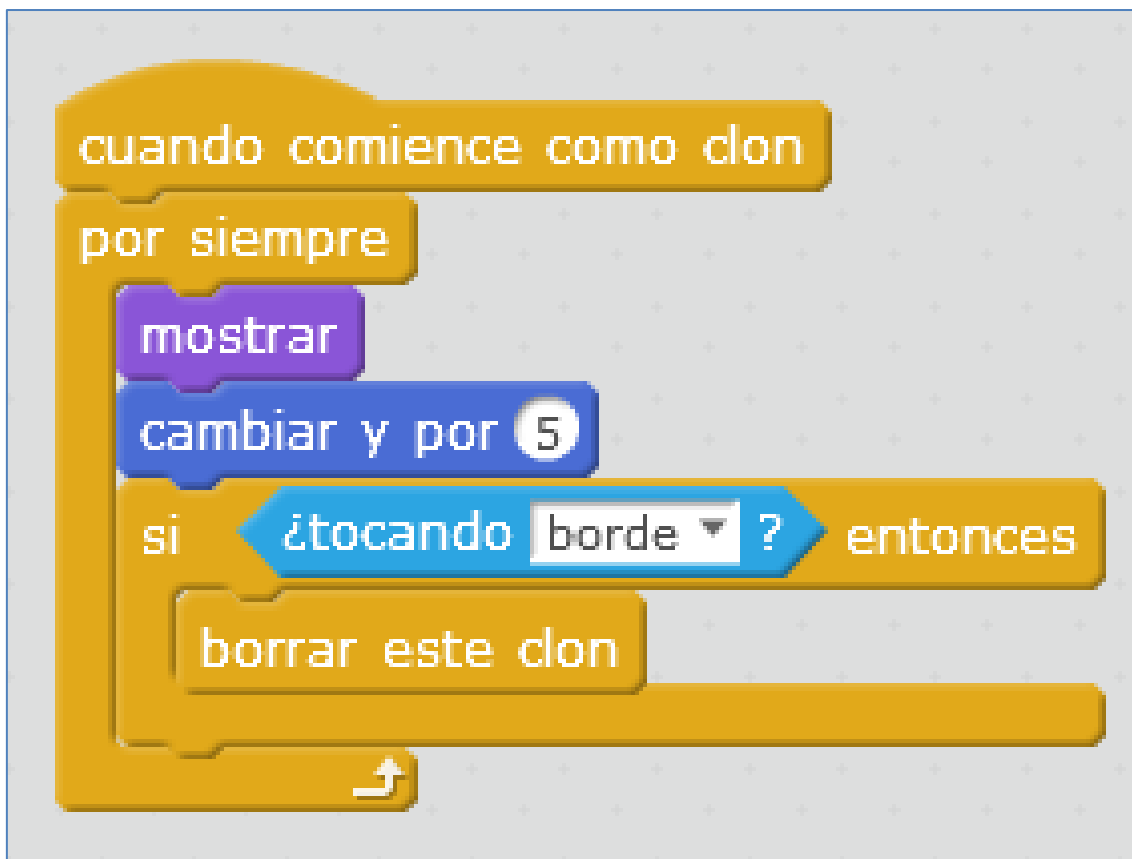
Cada vez que se presione el botón de la PicoBoard, se crea un clon del objeto misil cada 0,5 segundos:



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

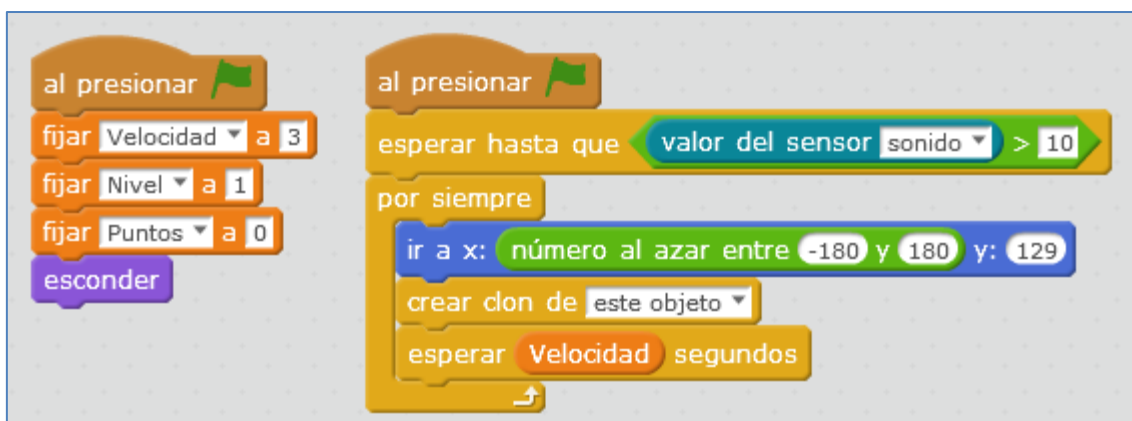
Divirtiéndome con mBot Ranger

Estos clones están escondidos porque el objeto principal, misil, también lo estaba. Por lo tanto, cuando comiencen como clon, deben mostrarse y realizar el movimiento que necesitamos. 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

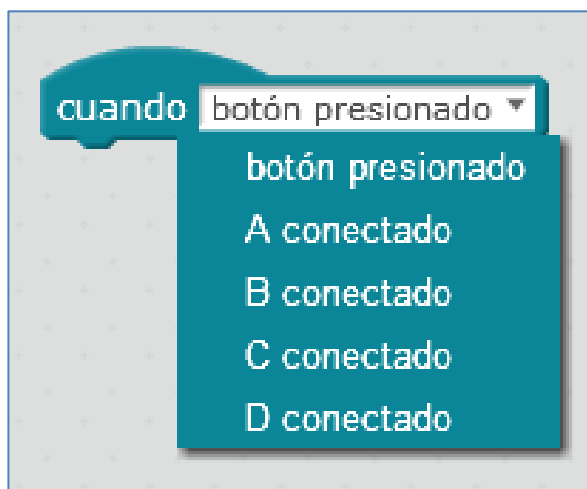
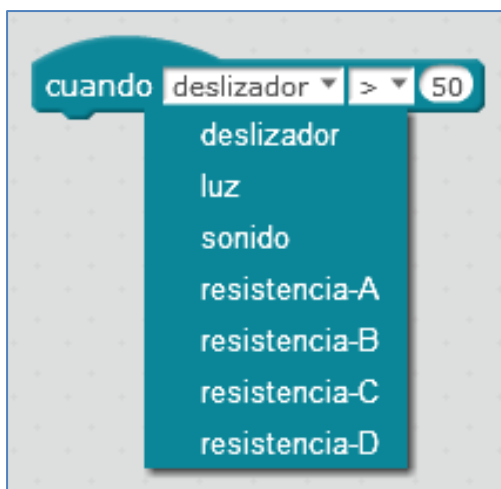
Divirtiéndome con mBot Ranger

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:

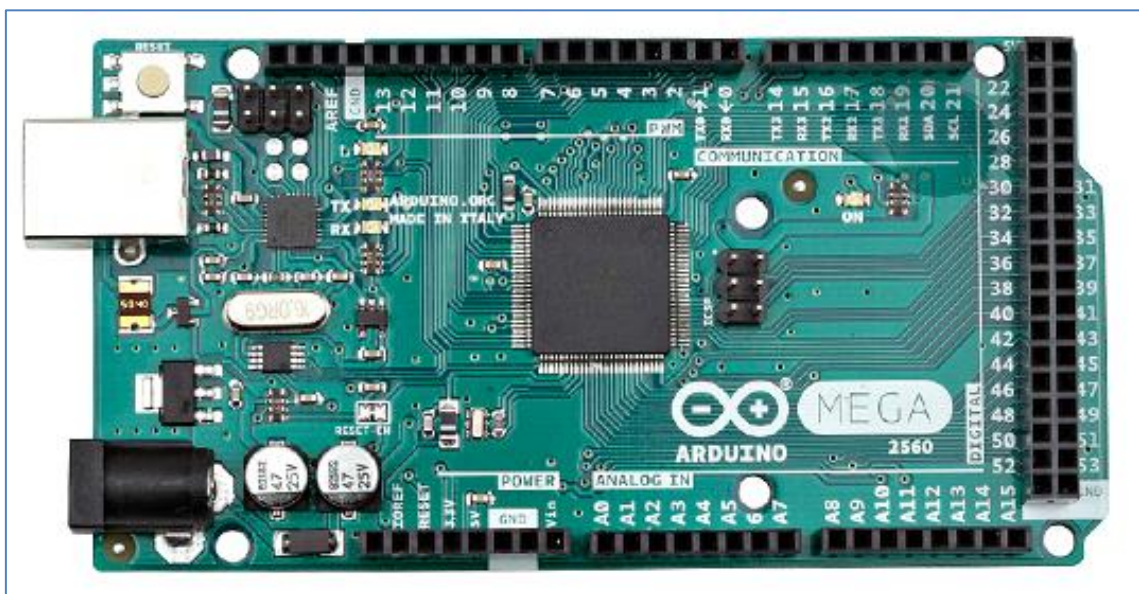
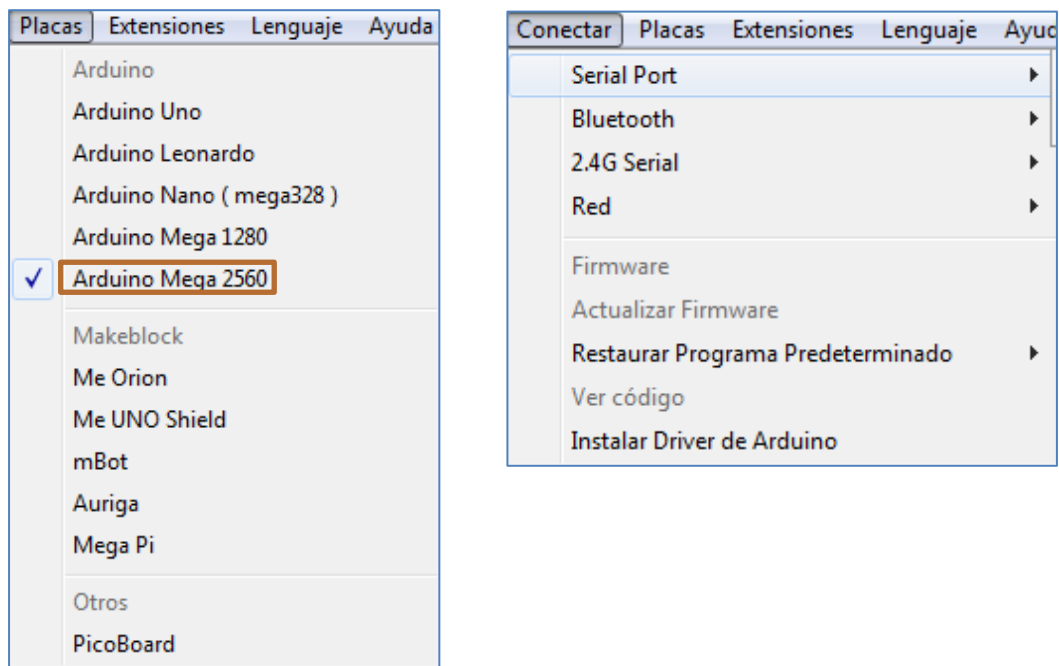


6.3. Ejemplo 3: Control de encendido/apagado de un LED por botón con Arduino Mega 2560 bajo mBlock

Se pretende controlar el encendido y apagado de un diodo LED a través de un botón pulsador, de modo que, si se presiona el botón el LED se enciende y si no se presiona, el LED se muestra apagado. A mayores, realizaremos una simulación en el escenario del mBlock.

Antes de programarla debemos conectar la placa Arduino Mega 2560 al software mBlock. Para ello seleccionamos la tarjeta Arduino Mega 2560 en la pestaña *Placas* del programa mBlock, así como, el puerto de conexión correspondiente en la pestaña *Conectar*:

Divirtiéndome con mBot Ranger

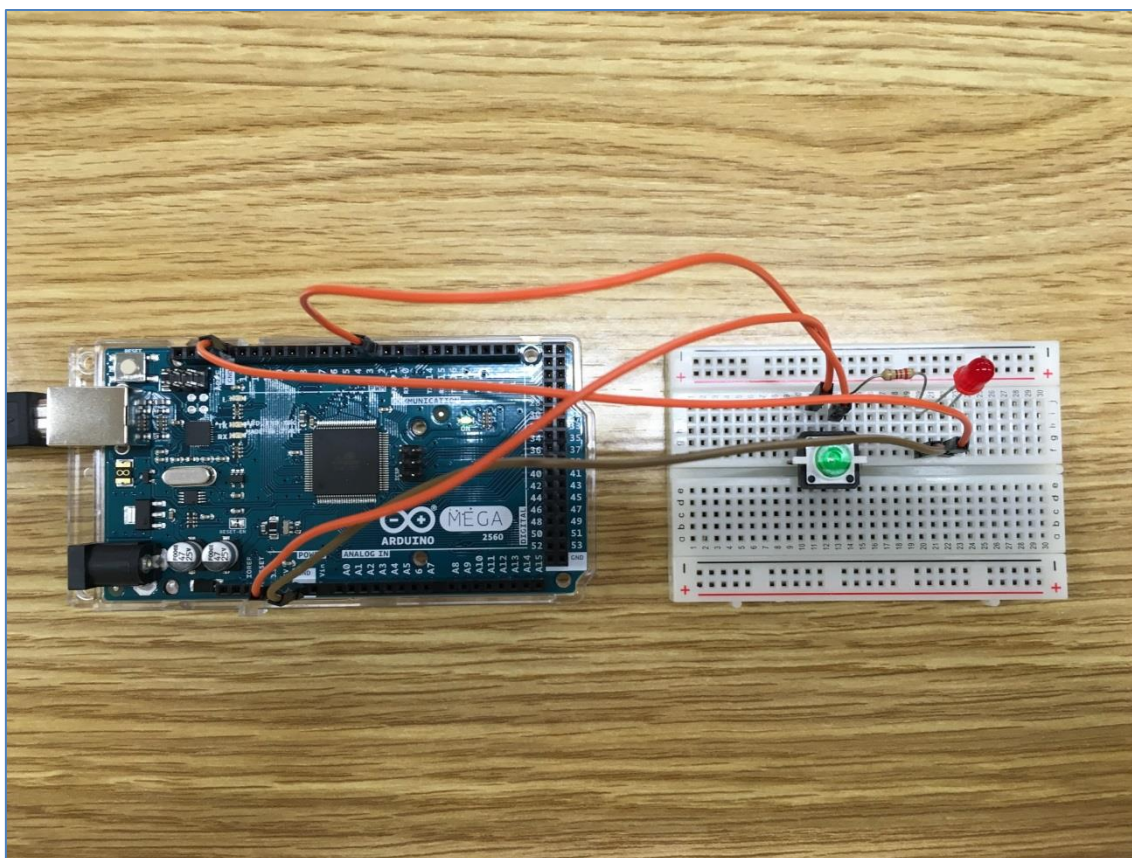
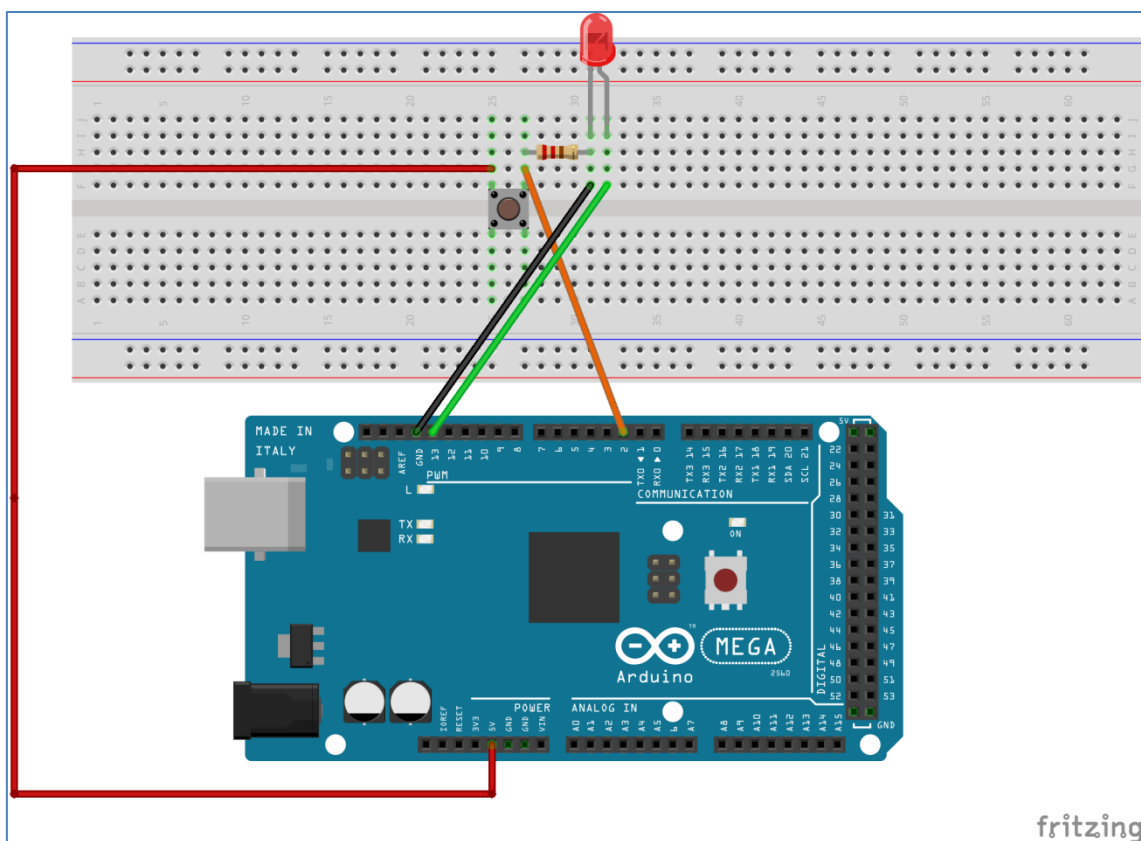


Placa Arduino Mega 2560

Para poder programar la placa necesitamos tener montado el circuito electrónico que queremos controlar:

La parte física del ejemplo requiere del montaje de la siguiente imagen: en ella vemos que estamos usando un diodo LED rojo (conectado al pin digital 13 de la placa Arduino Mega), una resistencia de 220Ω y un pulsador controlado por el pin digital 2 de la placa.

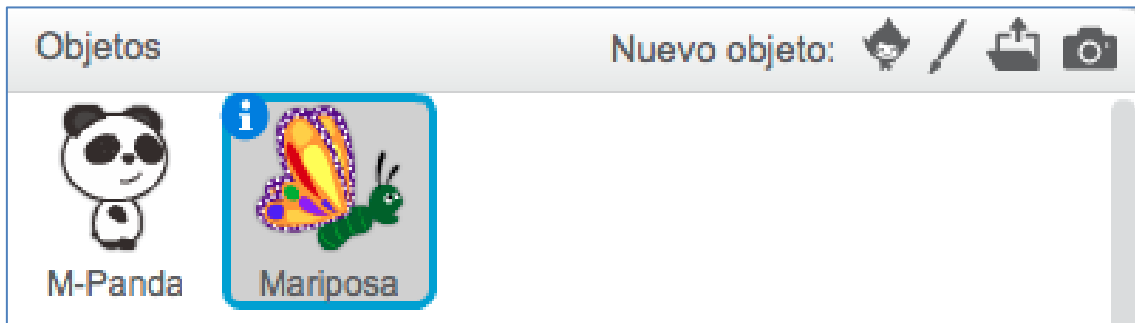
Divirtiéndome con mBot Ranger



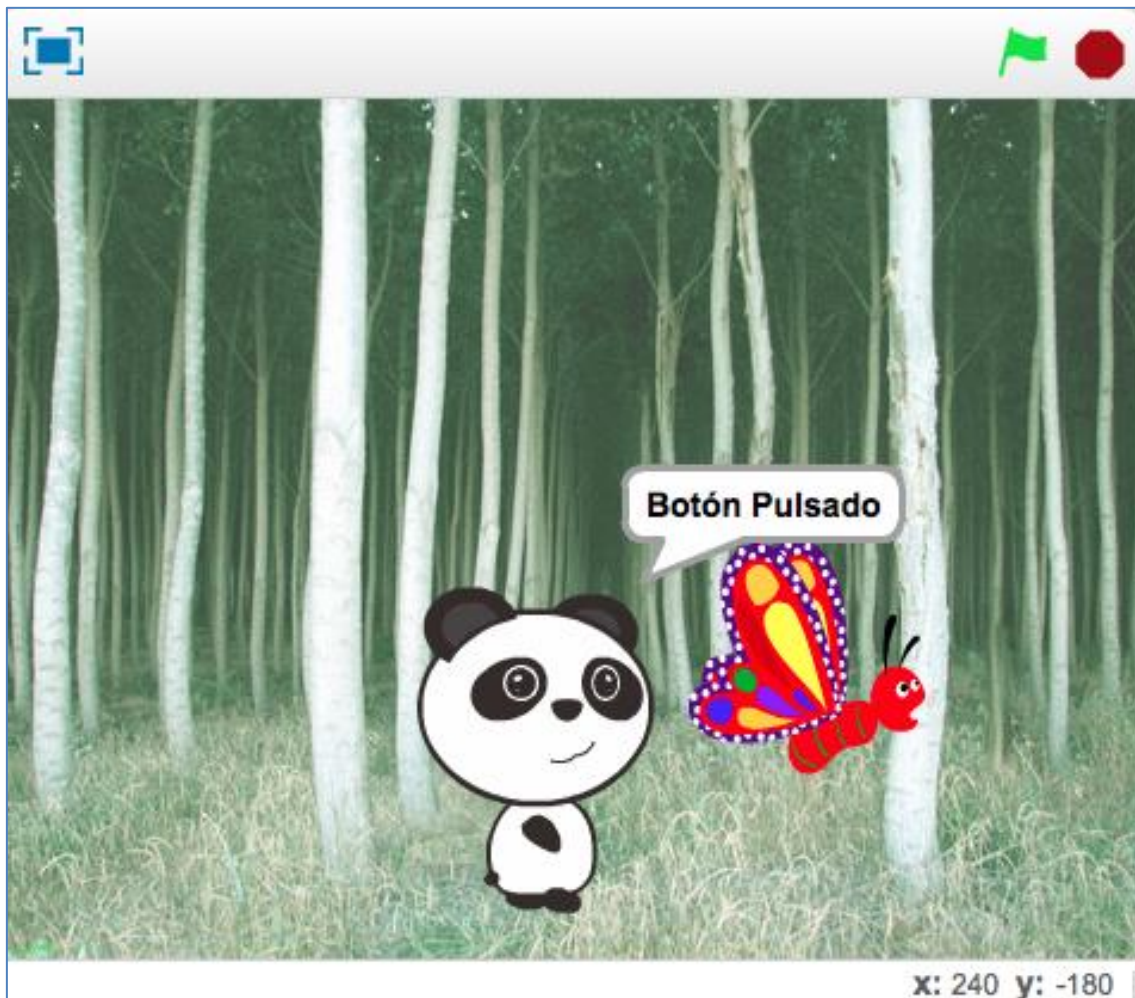
Circuito físico

Divirtiéndome con mBot Ranger

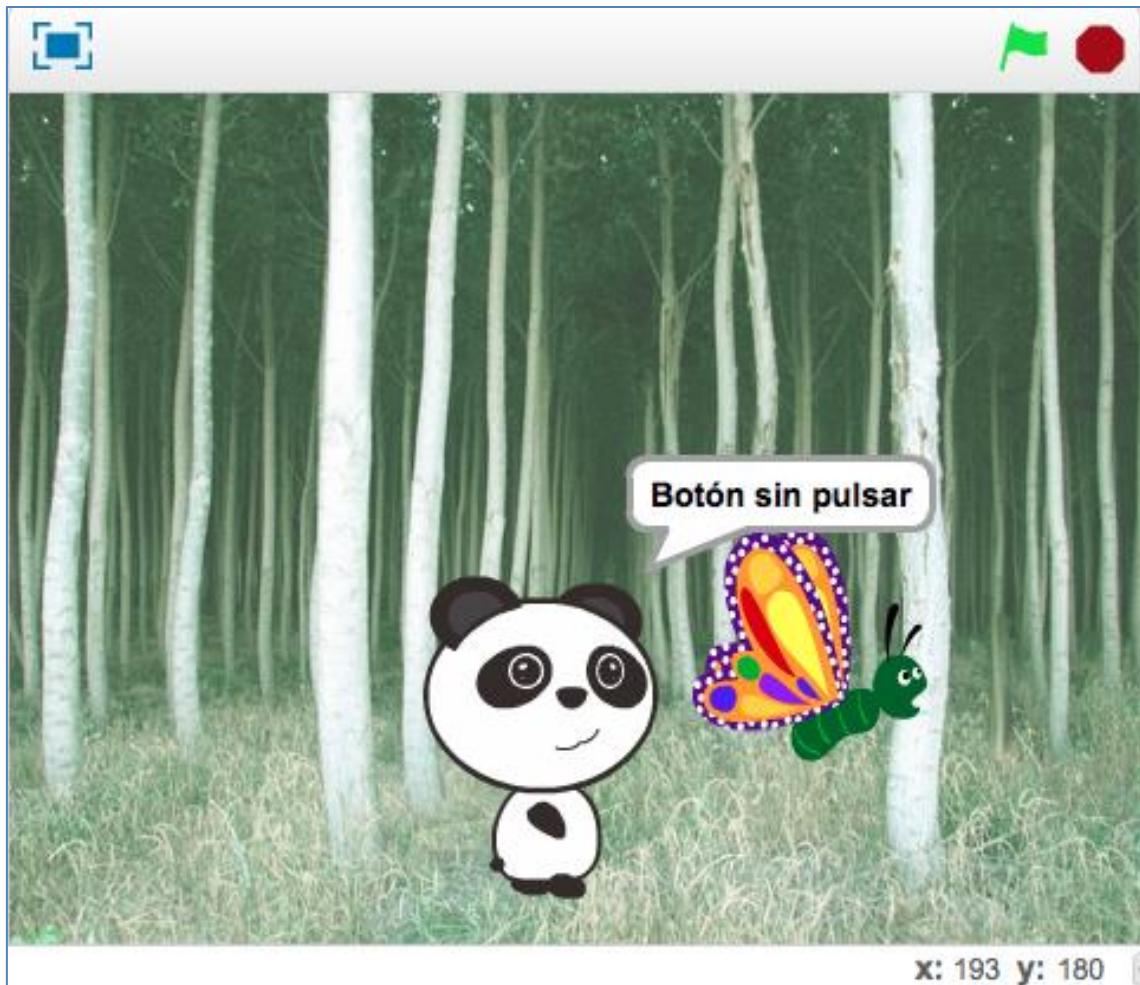
Sólo nos resta programar lo deseado. Para su simulación en el mBlock utilizo dos objetos: *M-Panda* y *Mariposa* (el objeto mariposa presenta dos disfraces)



En las siguientes imágenes podemos ver el escenario en las diferentes opciones programadas; pulsando el botón (mariposa roja) y sin pulsarlo (mariposa verde):

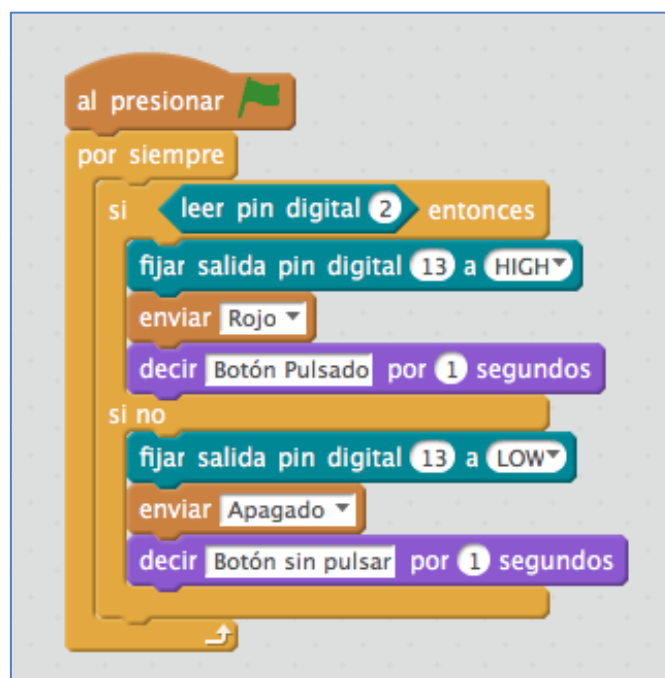


Escenario tras pulsar el botón



Escenario sin pulsar el botón

El objeto *M-Panda* es el que lleva el peso del programa. Su script es el siguiente:



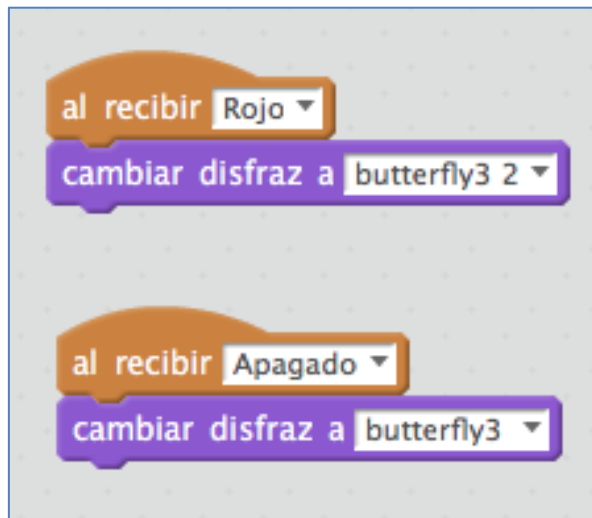
Script M-panda

Divirtiéndome con mBot Ranger

En él se observa un testeo continuo hacia el nivel lógico del pin digital 2. Si detecta que se ha pulsado el botón (pin digital 2 en alta) enciende el LED Rojo conectado al pin digital 13 a alta y envía el mensaje “Rojo”. Mensaje que es recogido por el objeto *Mariposa*, ordenándole cambiar el disfraz a “butterfly32” (mariposa en rojo).

En caso de no detectar que se ha pulsado el botón (pin digital 2 en baja) apagaría el diodo LED rojo y enviaría el mensaje “Apagado” al objeto *mariposa*. Cuando la *Mariposa* recibe el mensaje “Apagado” se le ordena que cambie su disfraz a “butterfly3” (mariposa verde).

El script del objeto mariposa es el siguiente:



Script Mariposa

7. Referencias

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

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

[Divirtiéndome con mBot: Guía de manejo y programación](#)

Divirtiéndome con Ranger

El robot Ranger es un robot pensado para que alumnos/as de 12 años en adelante exploren el mundo de la robótica.

Es un kit resistente y versátil, porque permite construir hasta 3 configuraciones diferentes; un robot tanque, otro de 3 ruedas y un robot balancín de 2 ruedas capaz de mantenerse estable gracias a su giróscopo.

Su placa controladora llamada "Me Auriga" está basada en el microprocesador open-source Arduino Mega 2560 y dispone de 10 puertos de expansión con conectores para añadir más funcionalidades.

Puedes programarlo desde tu iPad o tablet y controlar todas sus características y sensores mediante la aplicación móvil Makeblock HD.



Susana Oubiña Falcón