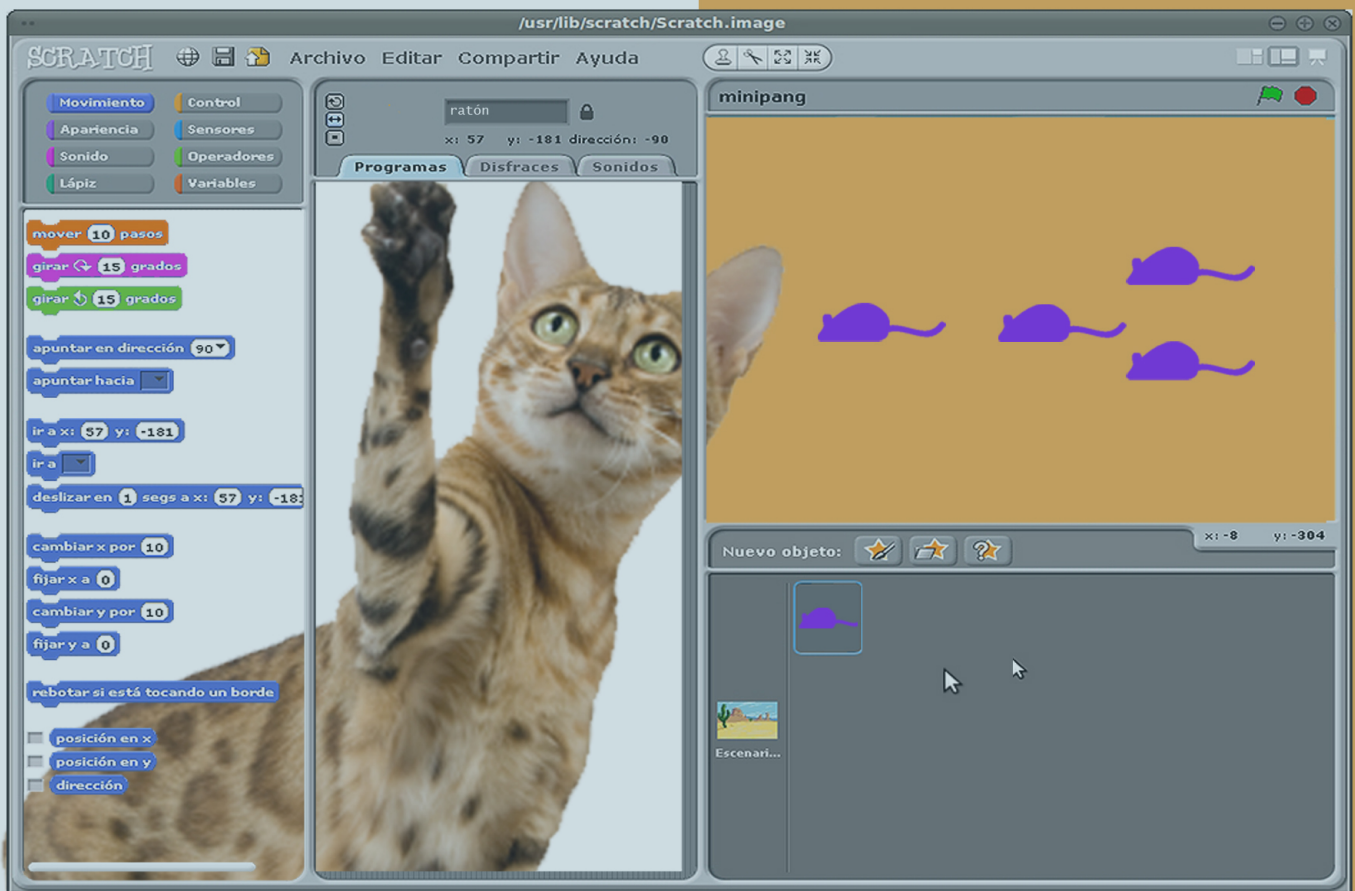


# Aprender haciendo con Scratch 2.0. Uso no ámbito científico, matemático, tecnolóxico.



***Susana Oubiña Falcón***

***Licencia: CC-BY***



"Aprender facendo con Scratch 2.0. Uso no ámbito científico matemático tecnolóxico", por *Susana Oubiña Falcón*, es publicado bajo la licencia [Creative Commons Reconocimiento 4.0 Internacional License](https://creativecommons.org/licenses/by/4.0/).

# Índice del curso

<b>1. Antes de comenzar.....</b>	<b>4</b>
1.1. Estructura del curso.....	4
1.2. Documentación del curso.....	5
<b>2. La herramienta Scratch 2.0 .....</b>	<b>8</b>
2.1. ¿Qué es Scratch? Modos de trabajo (Ventajas de trabajar en la nube).....	8
2.2. Manejar el programa scratch (sprites y fondos, diferentes bloques y herramientas) .....	13
2.3. Funcionalidad de la web de scratch: localizar programas compartidos, subir y compartir un programa.....	95
<b>3. Trabajar con Scratch bajo dos perspectivas.....</b>	<b>102</b>
3.1. Soy docente scratcher.....	102
3.1.1. Creación y modificación de programas funcionales en el aula para mi currículo (patrones). Ejemplos particularizados para las áreas de Tecnología, Matemáticas, Física y Química y Biología y Geología.....	103
3.1.2. Combinación de la herramienta Scratch con otros dispositivos y kits.....	112
3.1.2.1. Kit Makey Makey .....	112
3.1.2.2. Tarjeta de sensores Picoboard .....	114
3.1.2.3. Dispositivo Leap Motion.....	128
3.1.2.4. Robot WeDo de LEGO .....	145
3.1.2.5. Robot mOway.....	172
3.2. Soy alumno/a scratcher.....	207
3.2.1. Creación de videojuegos sencillos .....	207
3.2.2. Comprensión de contenidos .....	208
<b>4. Aumentar la motivación con la gamificación.....</b>	<b>219</b>
4.1. ¿Qué es la gamificación? (ventajas).....	219
4.2. ¿Cómo introducirla en el aula?.....	220
<b>5. Evaluar los trabajos de Scratch por rúbricas .....</b>	<b>229</b>
5.1. La rúbrica como forma objetiva de evaluación.....	229
5.2. ¿Cómo hacer una rúbrica? .....	232

# Módulo 1

---

## Antes de comenzar

---



## 1. Antes de comenzar.

Este curso se impartirá en el aula presentándose, por lo tanto, como un curso presencial. En él se pretende que el alumno/a logre desenvolver unas determinadas competencias. Para ayudar a conseguirlas, se ha decidido incluir este curso en un aula virtual, fomentando que el alumno/a pueda trabajar a su ritmo, según su disponibilidad de tiempo, presentar las dudas y compartir sus creaciones.

La utilización de esta aula virtual implica crear un módulo de inicio que denomino “*Antes de comenzar*” y que consiste en una simple presentación de los participantes y la documentación que se ha preparado para el curso.

Los **objetivos** de este módulo<sup>1</sup> son:

- ✓ Presentarnos para conocer a los participantes del curso.
- ✓ Conocer cómo se mueve el aula virtual
- ✓ Conocer la estructura del curso (contenidos, tareas, **proyecto final**).

### 1.1. Estructura del curso.

Este curso pretende desenvolver dos **competencias**:

- ✓ Tratamiento de la información y competencia digital.
- ✓ Aprender a aprender para la creación de programas educativos funcionales en el aula en áreas científicas tecnológicas (adquirir estrategias, destrezas e habilidades).

Para lograrlo, organizo el curso en los siguientes **módulos o temas**:

#### Módulo 1: “Antes de comenzar”

- ✓ Es un módulo informativo sobre el curso: documentación, forma de comunicarse, proyecto final.

#### Módulo 2: “La herramienta scratch”

- ✓ Se centra en el funcionamiento de los diferentes bloques del programa, diferentes formas de trabajo y como trabajar con la web de scratch.

#### Módulo 3: “Trabajar con scratch bajo dos perspectivas”

- ✓ Veremos cómo sacarle partido al scratch desde el rol de docente y

desde el alumno con ejemplos particularizados para contenidos de las áreas de Tecnología, Matemáticas, Física y Química y Biología y Geología. Para ello combinaremos el programa con diferentes kits y robots.

#### Módulo 4: “Aumentar la motivación con la gamificación”

- ✓ La gamificación es una disciplina que, bien aprovechada, nos ayuda a enganchar al alumno/a, a atraerlo. En este módulo veremos sus ventajas.

#### Módulo 5: “Evaluar los trabajos de scratch por rúbricas”

- ✓ Se centra en la creación de rúbricas como forma de evaluar los trabajos de nuestro alumnado que fueran creados con el scratch.

La temporalización del curso es la siguiente:

Este curso se diseñó para ser impartido en la modalidad presencial en 5 sesiones de 3h/s. Las horas totales del curso son 15h.

### 1.2. Documentación del curso.

La documentación del curso “Aprender haciendo con scratch 2.0. Uso no ámbito científico matemático tecnológico” se presenta de la siguiente forma:

a) Guía del curso. Documento que aporta:

Objetivos, competencias, contenidos, metodología, temporización de los módulos y tareas.

b) Manual del curso.

Reúne los contenidos del curso en un único documento en formato .pdf. La utilidad de este manual es que el alumno/a disponga de toda la información una vez finalizado el curso.

c) En el transcurso del curso se proporcionarán links de descarga para los siguientes programas y elementos de software.

- ✓ Scratch 2.0
- ✓ Scratch 1.4 (para utilizar la picoboard y WeDo)
- ✓ mOwayScratch (para utilizar mOway)

- ✓ Adobe AR (para trabajar con el Scratch2.0)
- ✓ Plugin de Scratch para LeapMotion
- ✓ Driver de la Picoboard y del robot mOwayScratch

# Módulo 2

---

## La herramienta Scratch 2.0

---

## 2. La herramienta Scratch 2.0

En este módulo se estudiará el manejo de los diferentes bloques que posee el programa scratch. Veremos que scratch no sólo es un lenguaje de programación intuitivo, sino que es una herramienta que ayuda a los docentes a trabajar con su alumnado utilizando conceptos como *el pensamiento crítico, la lógica, las habilidades de comunicación, etc.*

Considerando que queremos desenvolver las competencias “Tratamiento de la información y competencia digital” y la competencia “Aprender a aprender” con el fin de conseguir crear programas educativos funcionales en el aula según diferentes currículos dentro de las áreas científicas tecnológicas (adquirir estrategias, destrezas e habilidades), en este módulo conseguiremos ir abriendo este camino. Por lo tanto, los objetivos de aprendizaje de este módulo 2 se centrarán de la siguiente forma:

El **objetivo principal** de este módulo es comenzar a entender cómo funciona, manejarse con sus bloques y crear objetos y fondos desarrollando actividades sencillas que podrán ser parte de un proyecto o programa mayor cuando finalices todos los módulos. El **segundo objetivo** consiste en manejar la web de scratch ya que en ella podremos compartir nuestras creaciones, mejorar otras de otros scratcheadores y utilizar programas de interés para nosotros y que han sido creados por otros usuarios de la web.

### 2.1. ¿Qué es Scratch? Modos de trabajo (Ventajas de trabajar en la nube).

Scratch es un lenguaje de programación desarrollado por un equipo dirigido por Mitchell Resnick [1] 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).

[1] “Scratch: Programming for All | November 2009 | Communications of the ACM”. [En línea]. Disponible en: <http://cacm.acm.org/magazines/2009/11/48421-scratch-programming-for-all/fulltext>. [Accedido: 19 de julio de 2014].

#### A. ¿Por qué scratch en la educación?

Esta herramienta, llevada a las aulas y utilizada por nuestro alumnado, servirá para aumentar el aprendizaje de contenidos educativos y fomentará su motivación. Dentro de este entorno de programación libre, los usuarios podrán desarrollar juegos, expresar emociones y sentimientos, así como, reflejar

pensamientos e ideas de una forma interactiva y creativa. Por lo tanto, con él se trabaja el pensamiento o lógica computacional y habilidades como la responsabilidad social (al compartir proyectos) y el pensamiento crítico y creativo.

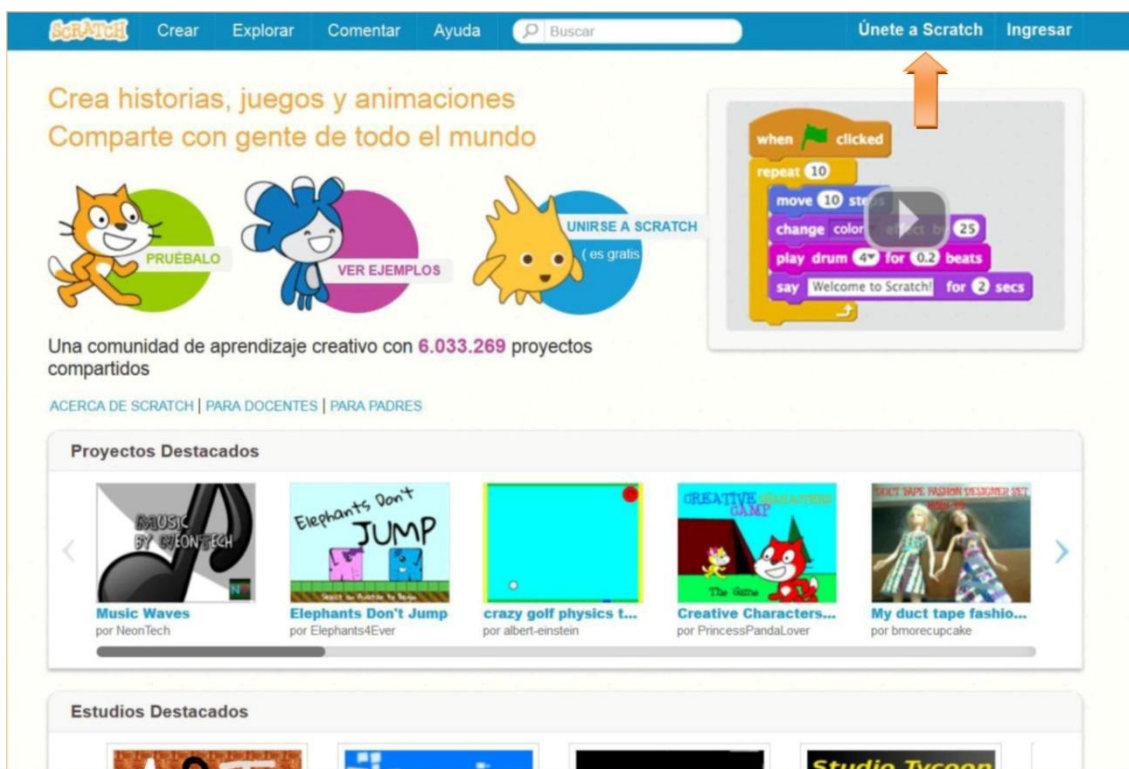
## B. Modos de trabajo

Con el programa Scratch 2.0 se puede trabajar de dos formas diferentes: de forma online, desde la web de scratch, o en el propio ordenador con la versión descargable del mismo.

### a) Online

Para trabajar de forma online los usuarios deben registrarse en la web de scratch. Para ello deben disponer de un correo electrónico (gmail u otra diferente). Para registrarse en la web debemos realizar los siguientes pasos:

1. Entrar en la web de scratch: <http://scratch.mit.edu/>
2. Pinchar en “Únete a Scratch”



Web de Scratch. Susana Oubiña Falcón (CC BY)

3. Cubrir los datos que se nos pide en los tres siguientes pasos del registro:

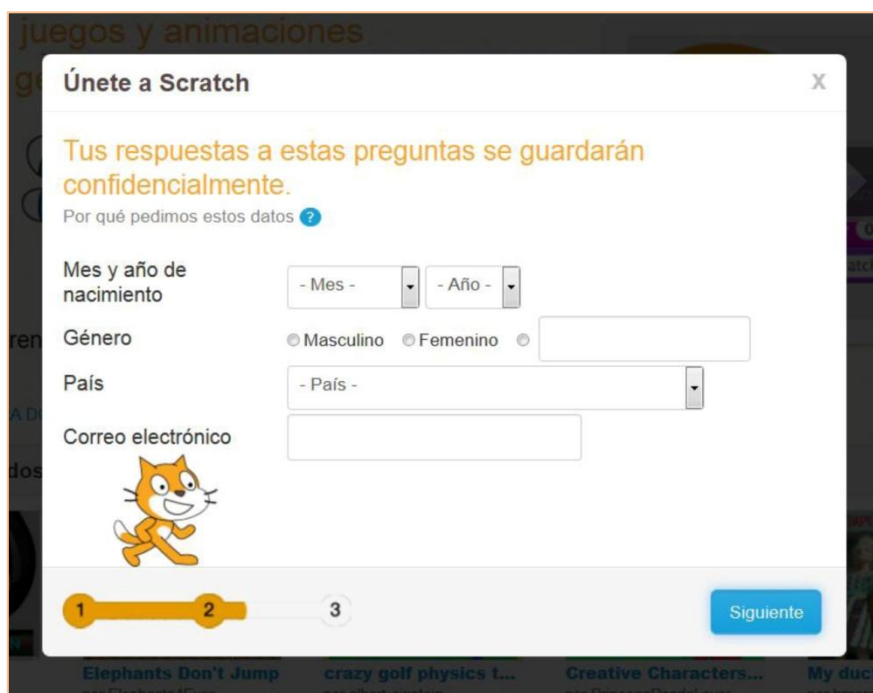
Paso1: Introducimos nuestro nombre de usuario (será el que se mostrará en la web de scratch) y elegimos una contraseña. Pinchamos en **Siguiente**.



The image shows the first step of the Scratch registration process. A modal window titled 'Únete a Scratch' is displayed over a background of game thumbnails. The window contains the text 'Es fácil (y gratis) registrar una cuenta Scratch.' followed by three input fields: 'Elige un nombre de usuario en Scratch', 'Elija una contraseña', and 'Confirmar contraseña'. A blue tooltip next to the username field says 'No uses tu nombre real'. At the bottom left is the Scratch cat logo, and at the bottom right is a 'Siguiente' button. A progress bar at the bottom shows step 1 as active, with steps 2 and 3 as inactive.

Registro Paso 1. Susana Oubiña Falcón ([CC BY](#))

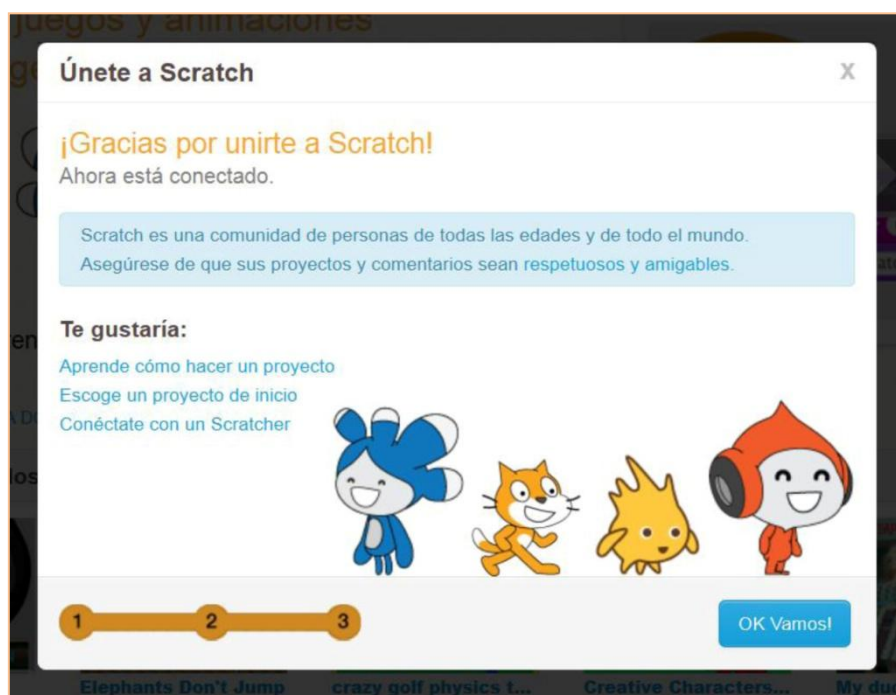
Paso 2: Cubrimos los datos relativos a edad, género, país y correo electrónico. Pinchamos en **Siguiente**.



The image shows the second step of the Scratch registration process. The modal window is titled 'Únete a Scratch' and contains the text 'Tus respuestas a estas preguntas se guardarán confidencialmente.' followed by 'Por qué pedimos estos datos ?'. There are four input fields: 'Mes y año de nacimiento' (with month and year dropdowns), 'Género' (with radio buttons for 'Masculino', 'Femenino', and an empty field), 'País' (with a dropdown menu), and 'Correo electrónico' (with a text input field). The Scratch cat logo is at the bottom left, and the 'Siguiente' button is at the bottom right. The progress bar at the bottom shows step 2 as active, with steps 1 and 3 as inactive.

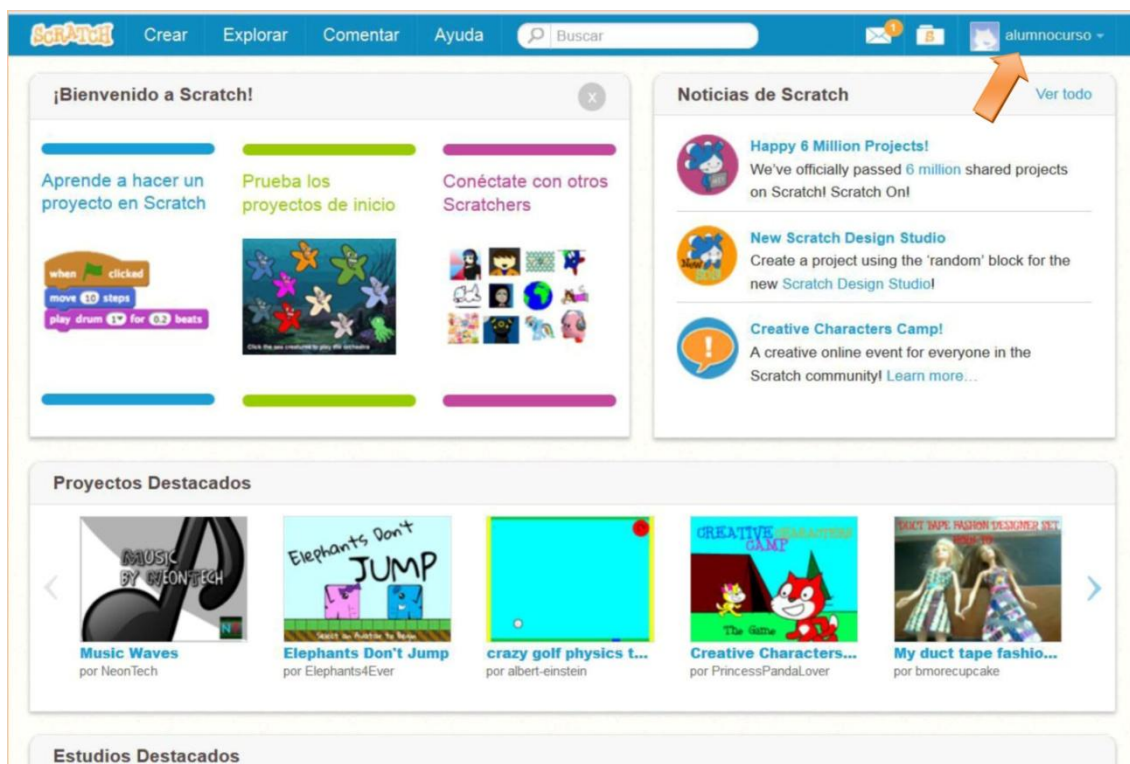
Registro Paso 2. Susana Oubiña Falcón ([CC BY](#))

Paso 3: Pinchamos en **OK Vamos**



Registro Paso 3. Susana Oubiña Falcón (CC BY)

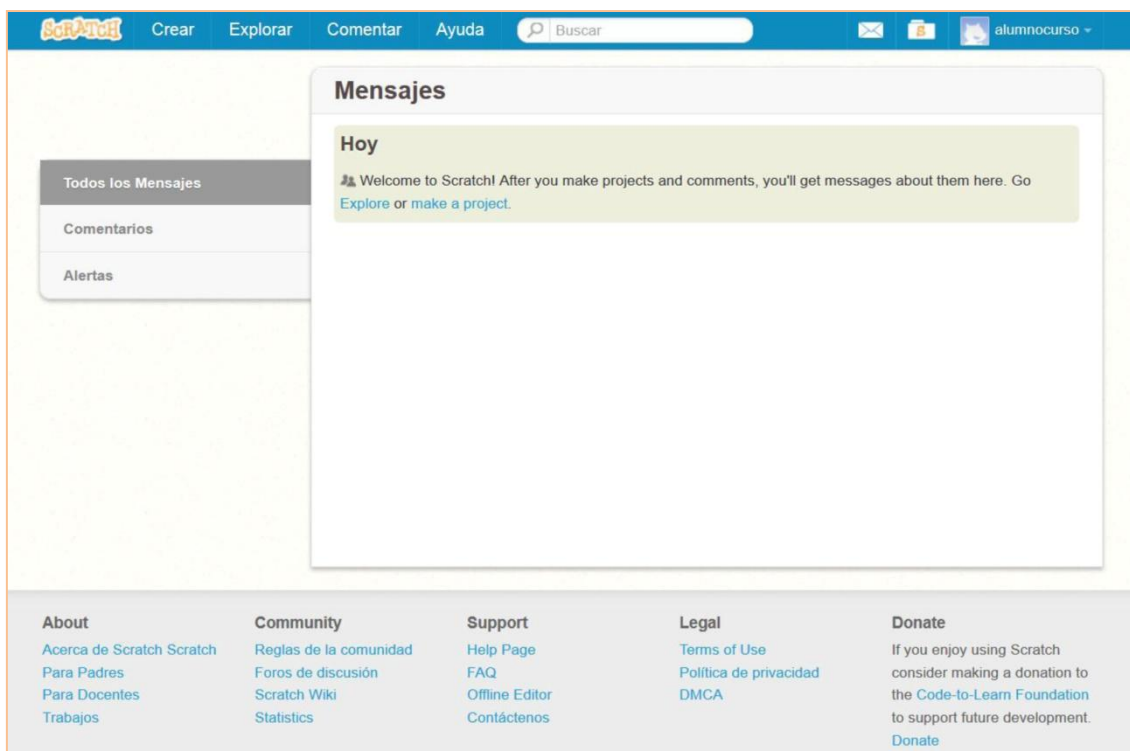
Tras estos tres pasos ya somos un usuario scratch. En la siguiente figura se muestra la interfaz del usuario scratch de prueba "alumnocurso":



Interfaz de un usuario scratch. Susana Oubiña Falcón (CC BY)



En la imagen “Interfaz de usuario scratch” podemos observar (icono mensaje) que hemos recibido un mensaje. Si pinchamos en él, lo abrimos, y veremos que es el mensaje de bienvenida que la web de scratch lanza a sus usuarios:



*Mensaje de bienvenida para un usuario scratch. Susana Oubiña Falcón ([CC BY](#))*

Las pestañas de la interfaz de usuario se explicarán en el punto 2.3 de este módulo.

## b) Versión descargable

Trabajar online implica que los usuarios dispongan en ese momento de conexión de internet y de un cierto ancho de banda que, a veces, resulta insuficiente en algunos centros educativos. Por eso es interesante y útil conocer y descargar la versión gratuita del programa **Scratch 2.0** que funciona en equipos con sistemas operativos iOS (MAC), Windows y en algunas versiones de Linux de 32bits.

El programa se puede descargar de la web de scratch. El link de su descarga es el siguiente: [enlace de descarga del programa scratch2.0](#)

Al instalar el programa debemos instalar el Adobe AIR (el enlace para este programa se encuentra en el link anterior y se realiza automáticamente en la instalación del scratch). También, en ese mismo link de descarga, al final de la página, hay un enlace para descargar la versión anterior **Scratch1.4**.

Pasos a seguir para instalar el programa Scratch 2.0:

1. Adobe AIR: Este paquete es necesario para que funcione el programa scratch. Según sea el sistema operativo de nuestro ordenador, el paquete de instalación es diferente. Para Windows e iOS (MAC), se debe instalar el paquete [Adobe AIR](#), pero para GNU/Linux debemos usar el paquete [AIR 2.6.0](#).
2. Scratch 2.0: Descargar e instalar el programa [Scratch 2.0 installer](#). Para ello hay que seguir los sencillos pasos que se indican.
3. Para instalar el programa Scratch 1.4 también hay que tener en cuenta los diferentes sistemas operativos (Windows, iOS (MAC) o Debian Ubuntu). En la siguiente página se accede a la [descarga del programa Scratch 1.4](#).

Se pueden tener ambos programas en el mismo PC, sin que entre ellos exista problema alguno de mal funcionamiento o bloqueo de funciones. Es decir, son programas diferentes y la nueva versión no borra la anterior, actualizándola.

## **2.2. Manejar el programa scratch (sprites y fondos, diferentes bloques y herramientas)**

En este curso trabajaremos con dos versiones del scratch. La nueva que es la versión Scratch2.0 y la anterior, que es la versión 1.4. La versión Scratch 2.0 presenta nuevos bloques que ofrecen mayores posibilidades, pero, la versión anterior también nos será útil cuando utilicemos otros dispositivos (Picoboard, WeDo y mOway) con el scratch. En un futuro cercano, no nos hará falta esta versión anterior, pero por ahora aún no es estable para ser utilizada con esos elementos con los que queremos combinarlo.

Pese a ser dos versiones distintas, con interfaces diferentes, al mismo tiempo son muy parecidas. Por ello, cuando sepamos manejarnos en la versión de scratch 2.0, que es la que estudiaremos, comprobaremos que la versión anterior la podemos manejar sin problema.

### **A. Interfaz de Scratch 2.0**

La interfaz del programa se muestra en la siguiente imagen:



Interfaz del programa Scratch 2.0. Susana Oubiña Falcón (CC BY)

En ella se observan unas zonas numeradas y que se explicarán de forma ordenada, un menú superior, un pequeño menú superior derecha de opciones reservadas para la web de scratch y una pestaña inferior llamada Mochila. Comenzaremos explicando la funcionalidad y manejo de las diferentes zonas de la interfaz:

Zona 1: Es la zona en la cual se prueba y se muestra cómo funciona el proyecto hemos diseñado. En el escenario es en donde se mueven e interactúan unos objetos con otros. Clicando en la bandera verde se ejecutan las sentencias de los diferentes bloques del programa y clicando en el círculo rojo paramos la ejecución del mismo. Al ser una zona de prueba, en ella se pueden probar partes de código del programa y no necesariamente el código completo.

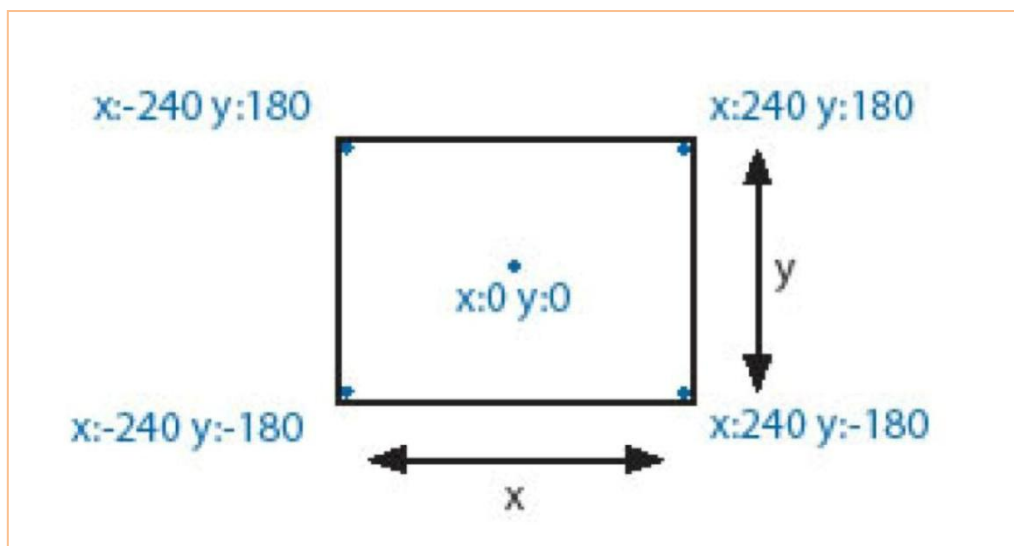
En la zona 1 (ver figura de abajo) hay un recuadro superior que por defecto, aparece con el título “Untitled-2”. En ese recuadro nosotros podemos introducir el título del proyecto que estamos creando. También podemos ampliar la visión de la pantalla haciendo clic en recuadro azul situado a la izquierda del título, que es recuadro que nos lleva a ver el proyecto en “Modo Presentación” o “Pantalla completa” (para retornar debemos pulsar la tecla escape).

En la figura podemos ver dos iconos: una bandera verde y un círculo rojo. La bandera verde activa el programa para que podamos ver su funcionamiento siendo, por lo tanto, el icono de inicio del proyecto. El círculo rojo es el icono de desactivación o de parada del proyecto que está corriendo en el scratch. Ambos son iconos representativos en un proyecto final, para dar el inicio o parar el programa que hemos diseñado.



*Zona de prueba.* Susana Oubiña Falcón ([CC BY](#))

Finalmente, en la pestaña inferior, se nos muestra la posición (x,y) del objeto seleccionado. En este caso, nos dice que el objeto o sprite gato está situado en la posición 240 del eje x horizontal y -117 del eje y vertical. Es importante saber que el escenario tiene 480 puntos (o píxeles) de ancho y 360 puntos de alto, dentro de un plano cartesiano XY que se ha representado en la siguiente imagen. Obviamente, el centro del escenario es el punto  $(x,y)=(0,0)$ .



*Representación cartesiana del Escenario.* Susana Oubiña Falcón ([CC BY](#))


Zona 2: Esta zona se utiliza para introducir los objetos (sprites) que programaremos en el proyecto. El objeto por defecto que introduce el scratch es el “gato”, pero podemos necesitar de otros objetos o incluso, no querer usar el objeto gato. En el proyecto podemos introducir objetos de cuatro diferentes formas: elegir un objeto de la biblioteca de scratch, dibujarlo nosotros, cargarlo desde un archivo o introducirlo capturando una imagen con la cámara de nuestro ordenador (ver figura siguiente). El scratch reconoce muchos formatos de imágenes: JPG, BMP, PNG, GIF (incluso GiFs animados).



*Zona de objetos u sprites.* Susana Oubiña Falcón ([CC BY](#))

Si queremos borrar el objeto, podemos seleccionarlo y después clicar en el botón derecho del ratón. De esta forma nos aparecerán las opciones: mostrar/esconder, exportar (lo guarda como archivo *.sprite* y puede ser




importado a otro proyecto), duplicar el objeto (crear otro igual), o borrarlo (eliminar el objeto).

También podemos configurar ciertas propiedades importantes del sprite como los son “el estilo de rotación”, “dirección” de movimiento, “mostrar” el objeto en el programa, “arrastrar” el objeto en el programa y su nombre. Para acceder a estos cambios debemos hacer clic en el icono azul , del sprite que puede verse en la figura anterior. De este modo, nos encontraremos en la siguiente ventana:



*Menú desplegable del sprite1. Susana Oubiña Falcón (CC BY)*

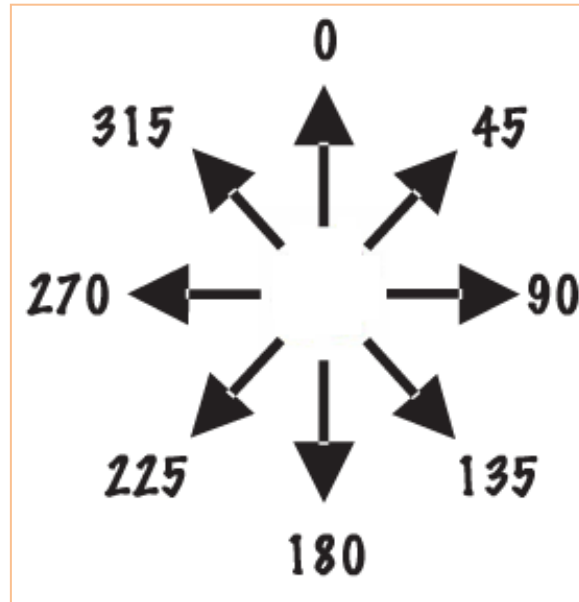
La orientación o *dirección* que le demos al objeto variará según el *estilo de rotación* elegido para él. Cuando el proyecto está finalizado, otro usuario puede interactuar con él. A veces, interesa que el usuario pueda arrastrar los objetos por el escenario. Scratch implementa 3 estilos de rotación: **rotar**, **pestaña izquierda-derecha** y **No rotar**:

Estilos de Rotación	Descripción
	El disfraz rota a medida que el objeto cambia de dirección
	El disfraz mira o a la izquierda o a la derecha
	El disfraz nunca rota, aun cuando el objeto cambie de dirección

*Estilos de rotación. Susana Oubiña Falcón (CC BY)*

La dirección del objeto indica en qué orientación (o dirección) se moverá cuando se ejecute un bloque de movimiento (0=arriba, 90=derecha, 180=abajo y -90=izquierda). En la figura del Menú desplegable del sprite1 está seleccionada la dirección derecha (90), pero ésta puede cambiarse arrastrando

esa línea. Por lo tanto, las direcciones, en scratch, son ángulos centrados en el eje Y ascendente, de modo que, si los medimos en sentido horario obtenemos valores positivos; en cambio, si los medimos en sentido antihorario, sus valores serán negativos. Por ejemplo, el ángulo  $-45^\circ$  se correspondería con el ángulo  $315^\circ$  de la siguiente figura:



Ángulos en scratch. Imagen recogida de [Código Octopus](#). (CC BY SA)

**Zona 3:** Es la zona reservada para nuestro escenario. En ella podemos introducir un escenario de la biblioteca de scratch, dibujarlo nosotros con el editor de scratch, cargarlo desde un archivo o introducirlo desde cámara.



Zona del escenario. Susana Oubiña Falcón (CC BY)

Scratch presenta un potente editor para crear y editar escenarios y objetos.

Zona 4: Es la zona de programación de objetos y escenarios, área de programas (ver figura siguiente). Scratch es un lenguaje de programación orientado a objetos. Para programar, no hace falta conocer código, siendo la programación mucho más sencilla: sólo hay que encajar diferentes órdenes (sentencias) de los bloques unos sobre otros formando lo que podríamos llamar pila. Esas órdenes se arrastran de su bloque correspondiente hacia la zona 4. Cada objeto debe tener sus respectivas pilas de órdenes, si queremos que el objeto haga algo. Esta zona también se utiliza para el “Escenario” siendo tratado en scratch como otro objeto (el escenario dispone de menos órdenes en los bloques que los sprites).

En la siguiente imagen se observan las sentencias u órdenes del bloque movimiento. Cualquiera de ellas se introduce en la zona 4 por medio del arrastre con el ratón (arrastrar y soltar).



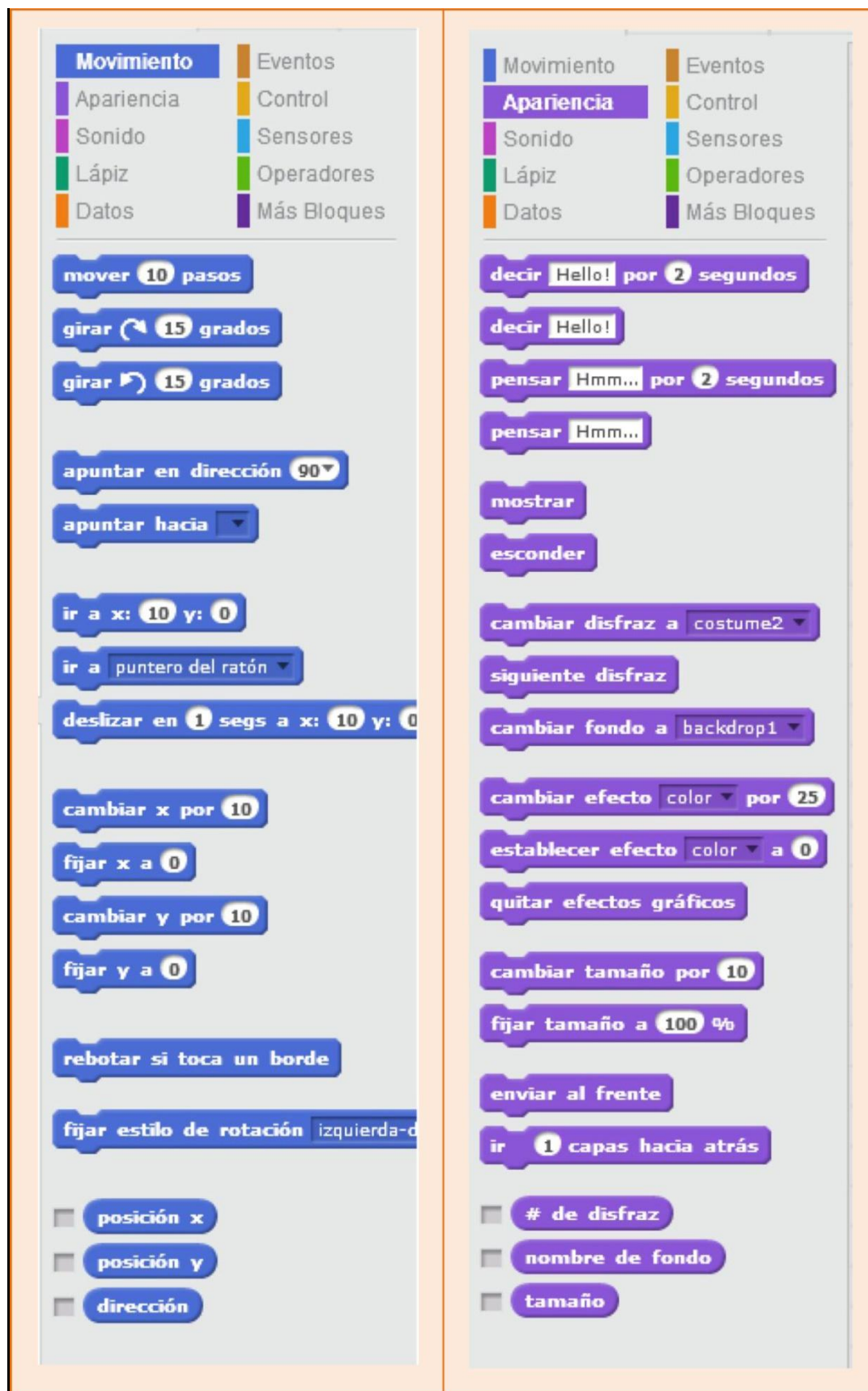


*Zona de programación.* Susana Oubiña Falcón ([CC BY](#))

Los objetos no sólo se pueden programar (crear scripts) utilizando las órdenes de los diferentes bloques de la pestaña **Programas**, sino que también pueden mostrarse con diferentes disfraces o incluir sonidos, clicando en las pestañas **Disfraces** y **Sonidos**. Para ello debemos hacer clic en su correspondiente pestaña de la figura que hemos denominado *Zona de programación*. Ahora nos centraremos en el control y manejo de estas pestañas.

### **Pestaña Programas**

En la pestaña Programas podemos acceder a los diferentes bloques de programación con sus sentencias. A continuación se describen



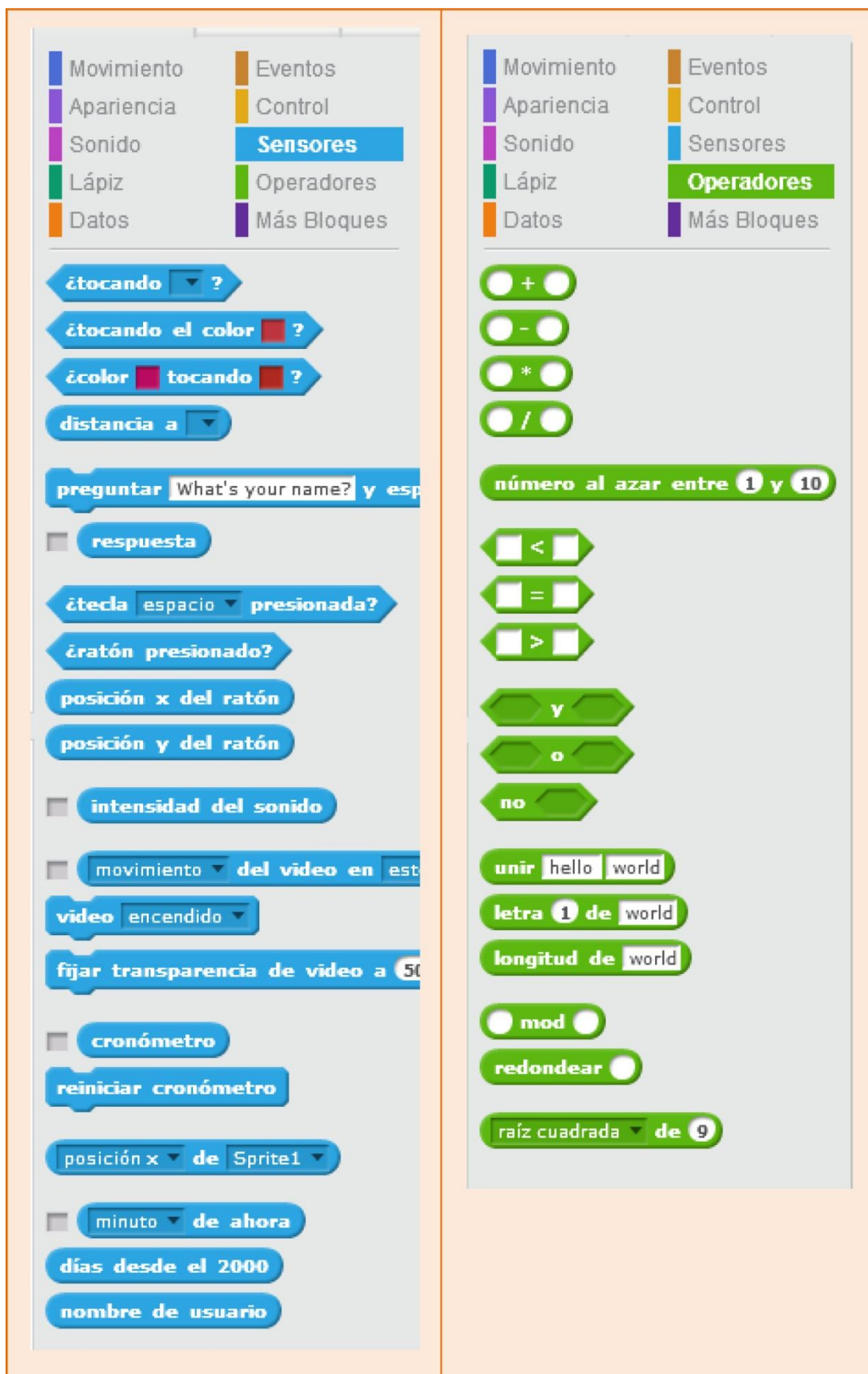
*Bloques de Movimiento y de Apariencia. Susana Oubiña Falcón (CC BY)*



Bloques de Sonido y Lápiz. Susana Oubiña Falcón ([CC BY](#))



*Bloques de Eventos y Control.* Susana Oubiña Falcón ([CC BY](#))



*Bloques de Sensores y Operadores. Susana Oubiña Falcón (CC BY)*

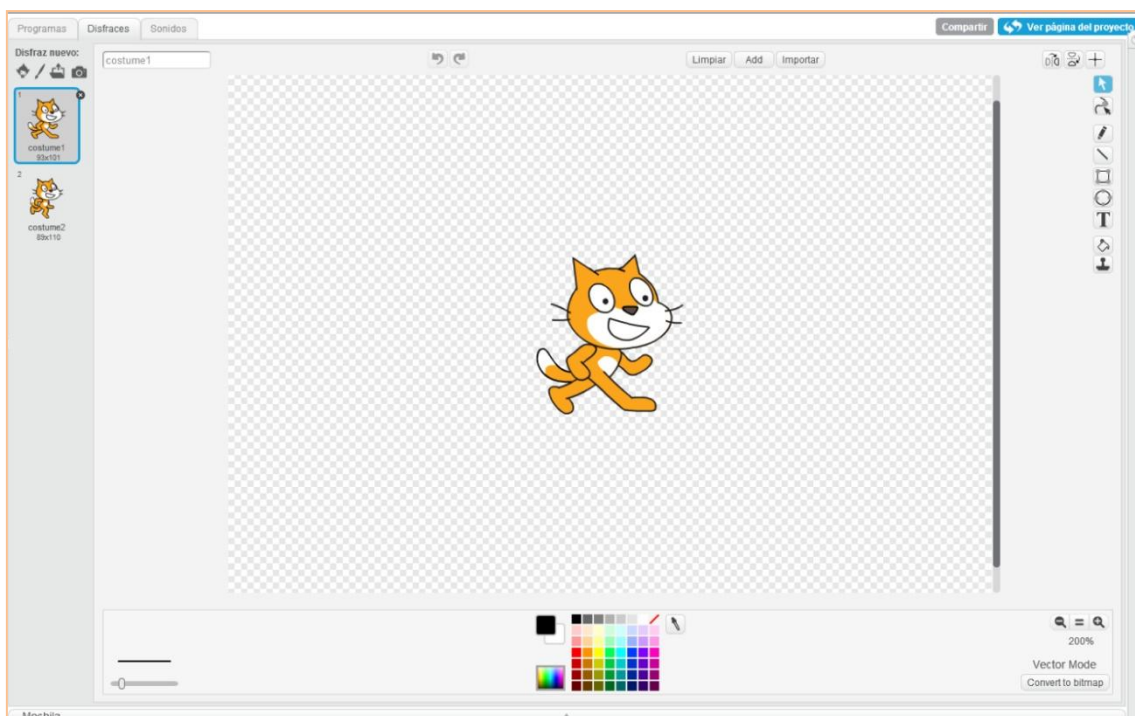


El bloque de **Datos** se utiliza para crear variables y listas de datos. (En la versión anterior Scratch 1.4, se denomina “Datos”).

El bloque de **Más Bloques** se utiliza para crear subprogramas (bloques de instrucciones) que se repetirán en varios objetos (programación recursiva) o para sumar extensiones como la de la Picoboard, Wedo o Leap Motion. Estos nuevos bloques tendrán sus propios procedimientos y funciones.

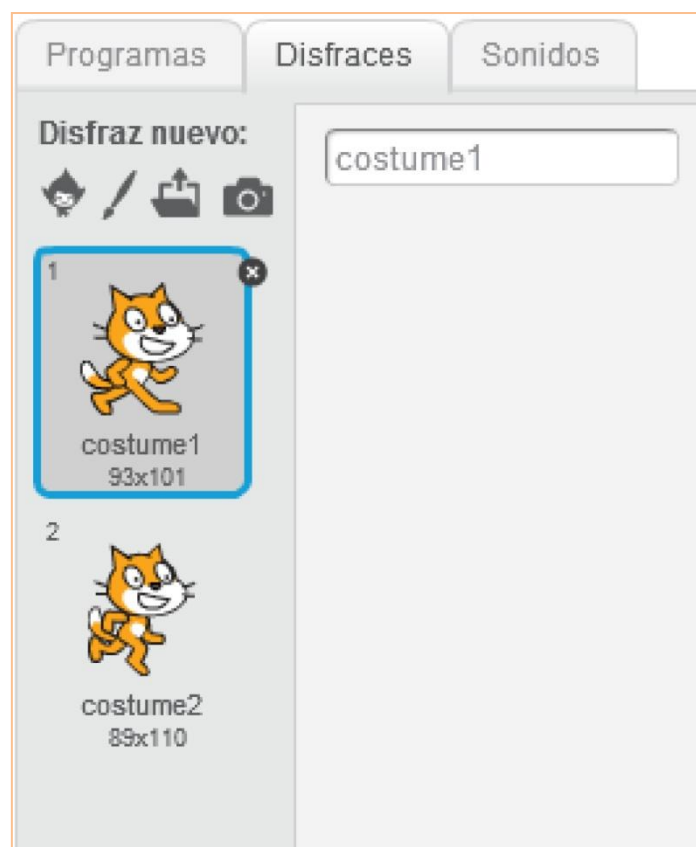
### Pestaña Disfraces

Los disfraces (costumers) se pueden ver y editan clicando la pestaña Disfraces. En el objeto por defecto del scratch, se observan dos disfraces o costumers







*Interfaz del Editor de Disfraces. Susana Oubiña Falcón ([CC BY](#))*

El objeto “gato” (sprite1, de la siguiente imagen) presenta dos disfraces: costume 1 y costume 2. En la imagen se observa el disfraz actual del objeto, costume1, que está resaltado en azul. Ese disfraz se puede editar y modificar con el editor de objetos. Además, el orden de los disfraces también se puede cambiar, sin más que arrastrarlo hacia otro. Al hacer clic en la **X** del disfraz resaltado, se abre un menú que nos da opción a eliminarlo o duplicarlo. Finalmente, podemos crear un disfraz nuevo. Esto puede realizarse de 4 formas, que se resumen en la tabla “Creación de disfraces”:



Diferentes disfraces del objeto “gato”. Susana Oubiña Falcón ([CC BY](#))

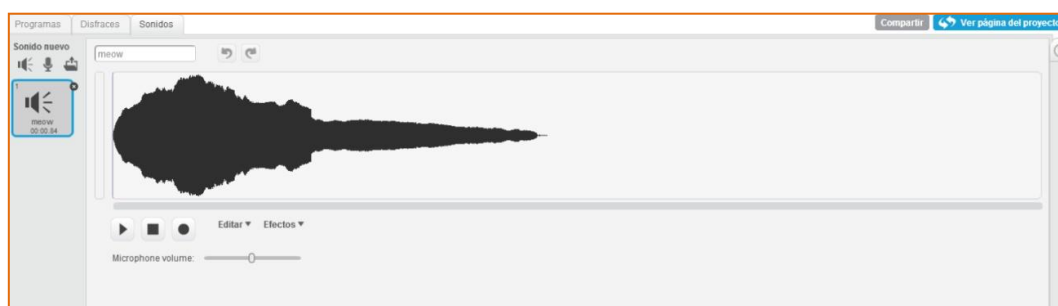
Creación de disfraces	Descripción
	Para dibujar un nuevo disfraz con el Editor de Objetos y disfraces
	Para importar un archivo de imágenes del disco duro
	Para tomar fotos con la cámara del ordenador
	Para importar un objeto disponible en el scratch 2.0

Creación de disfraces. Susana Oubiña Falcón ([CC BY](#))

El Editor de Pinturas, para crear y modificar disfraces y escenarios es muy potente y se describirá a lo largo de este documento.




## Pestaña Sonidos

Al hacer clic en la pestaña sonidos, entramos en la siguiente interfaz:



*Interfaz Sonidos.* Susana Oubiña Falcón ([CC BY](#))

El sonido por defecto es el del objeto “gato”; sonido “meow”, pero podemos incluir sonidos en los proyectos scratch (para objetos y escenario) de 3 formas diferentes, tal y como se expresa en la siguiente tabla:

Creación de sonios	Descripción
	Seleccionar un sonido de la librería de sonidos de scratch
	Grabar nuevos sonidos
	Importar sonidos de audio

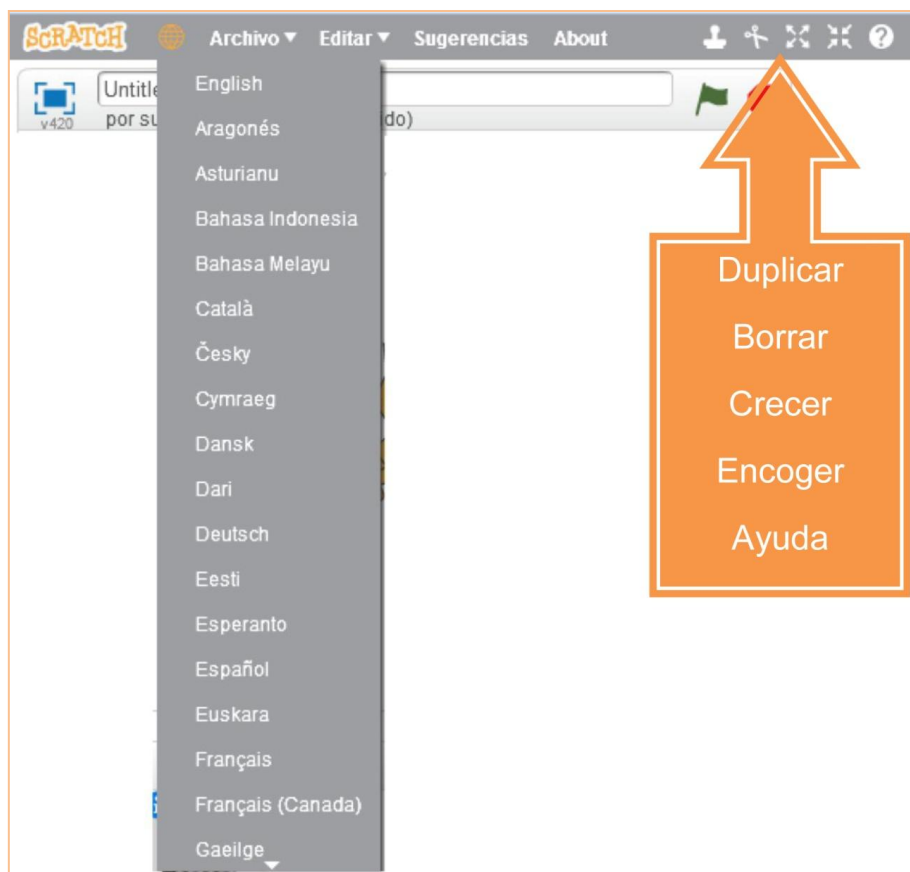
*Creación de sonidos.* Susana Oubiña Falcón ([CC BY](#))

Es importante conocer el formato de audio de este programa. Scratch puede leer archivos de audio en formato MP3 y archivos descomprimidos WAP, AIF y AU (sólo en 8 y 16 bits, pero nunca en 24 bits).

Los sonidos se pueden editar e incluso modificar con la inclusión de efectos.

El Menú superior: Este menú se muestra en la siguiente figura. La pestaña Scratch enlaza con la web de scratch, el icono bola del mundo nos sirve para cambiar de idioma (como puede verse en la imagen, existe una gran variedad de idiomas implementados en el scratch), la pestaña Sugerencias permite entrar en modo editor de un proyecto y la pestaña About nos lleva a una [página de scratch](#) que nos informa sobre el programa.





*Menú Superior Izquierdo.* Susana Oubiña Falcón ([CC BY](#))

Los iconos de la derecha de la imagen se utilizan directamente en los objetos y permiten, respectivamente, duplicar, borrar, hacer crecer o hacer encoger el objeto que estamos viendo en el escenario. Finalmente se encuentra el icono con la interrogación y que nos ofrece información o ayuda (en inglés) sobre la utilización de un comando u orden. Una descripción de los mismos se observa en la siguiente imagen:

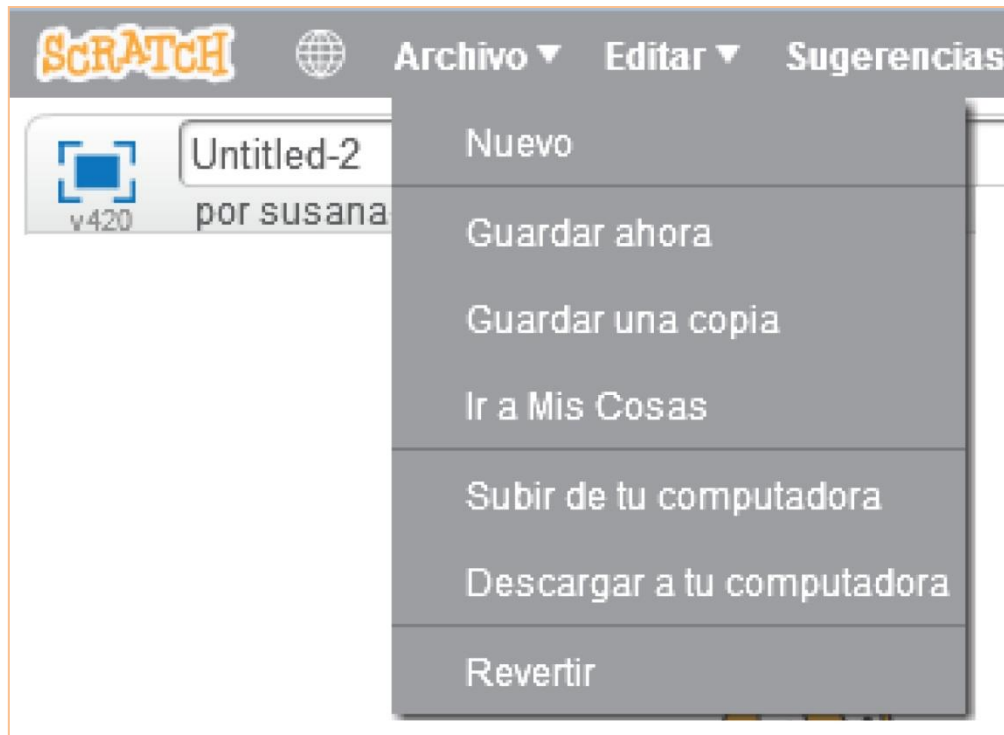
Haga clic en la **Barra de Herramientas** para seleccionar una herramienta y luego haga clic sobre otros Objetos para llevar a cabo una acción.

	<b>Duplicar:</b> Duplica Objetos, disfraces, sonidos, bloques y programas. (Shift+click para multiplicar la acción).
	<b>Borrar:</b> Borra Objetos, disfraces, sonidos, bloques y programas. (Shift+click para multiplicar la acción).
	<b>Agrandar Objeto:</b> Aumenta el tamaño de los Objetos. (Shift+click para acelerar esta acción).
	<b>Achicar Objeto:</b> Disminuye el tamaño de los Objetos (Shift+click para acelerar esta acción).
	<b>Ayuda:</b> Muestra una ventana de ayuda en la parte derecha del entorno.

Para regresar al cursor (flecha), haga clic en cualquier espacio en blanco de la pantalla.

*Barra de herramientas.* Susana Oubiña Falcón ([CC BY](#))

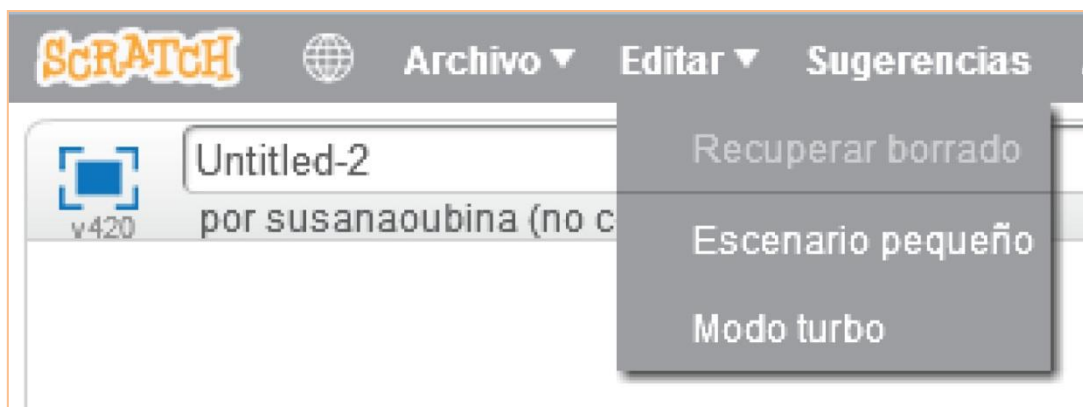
La pestaña **Archivo** presenta el desplegable en donde podemos: Crear un archivo Nuevo, guardar una copia o guardar ahora, ir al espacio Mis Cosas para recoger un Archivo, subir o descargar un archivo de nuestro ordenador y, finalmente, la opción revertir que no es más que una opción que te deshace los cambios que hicieras en tu programa devolviéndote un programa de la última versión guardada.



*Opciones de la pestaña Archivo. Susana Oubiña Falcón ([CC BY](#))*

La pestaña **Editar** presenta el desplegable en donde podemos: recuperar el último bloque, programa, objeto, disfraz o sonido que borráramos, trabajar con un escenario pequeño (permite agrandar o achicar el área del escenario) o arrancar el modo turbo. El modo turbo es interesante ya que acelera el programa. Por lo tanto, podemos utilizarlo cuando el programa se desarrolle de forma lenta y, para no esperar, nos interesa que las instrucciones que queremos que las instrucciones o pasos del programa se ejecuten de forma más rápida o para ver el resultado final de un proyecto.

El modo turbo se puede activar (y desactivar) pulsando simultáneamente la tecla Shift y la bandera verde.



*Opciones de la pestaña Editar. Susana Oubiña Falcón ([CC BY](#))*

En la versión online del Scratch 2.0, hay opciones que comunican directamente los proyectos creados con la web o los proyectos de la web con tu computador. Estas opciones se encuentran en el menú superior derecho (ver siguiente imagen). Como puede verse en la imagen, nos da opción a compartir de forma sencilla el proyecto creado o a ver la página de un proyecto determinado. También presenta otras opciones para el usuario scratch dentro de la web y que se verán en el punto 2.3 (Perfil, Mis Cosas, Configuración de la cuenta, Salir).

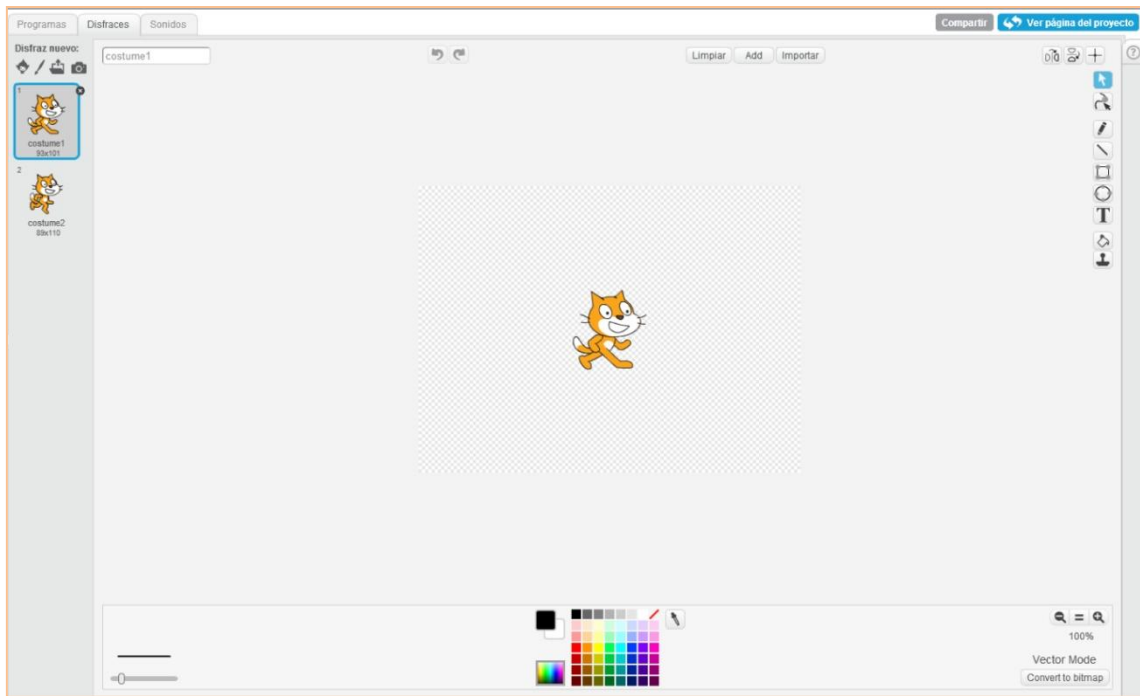


*Opciones de enlace con Scratch. Susana Oubiña Falcón ([CC BY](#))*

**Mochila:** La Mochila o Backpack es de gran ayuda para el usuario scratch. Permite copiar y mover objetos, disfraces, escenarios y scripts (programas) de un proyecto a otro. Trabajando de forma online (con usuario y contraseña), podemos abrir la mochila dentro de cualquier proyecto e introducir en ella (o sacar para otro proyecto), por el método de arrastre, cualquier elemento (objeto, disfraz, programas, escenarios). Su potencial es enorme, minimizándonos el tiempo de trabajo a la hora de crear un proyecto ya que podemos reutilizar, de forma sencilla, elementos de otros proyectos, mezclar varios proyectos para crear uno solo convirtiéndose en un proyecto “re-mix”.

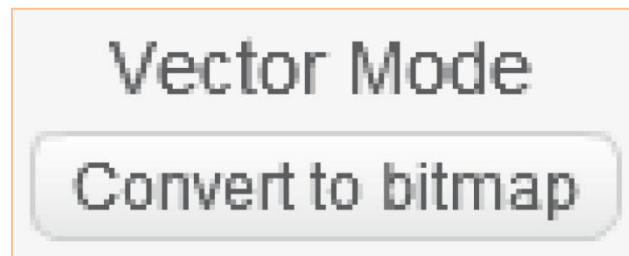
## **B. Editor de Pinturas**

Se utiliza para crear y modificar los disfraces de los objetos y los escenarios del proyecto. Es muy sencillo de utilizar y al mismo tiempo, es muy potente.



*Editor de Pinturas (imagen vectorial).* Susana Oubiña Falcón ([CC BY](#))

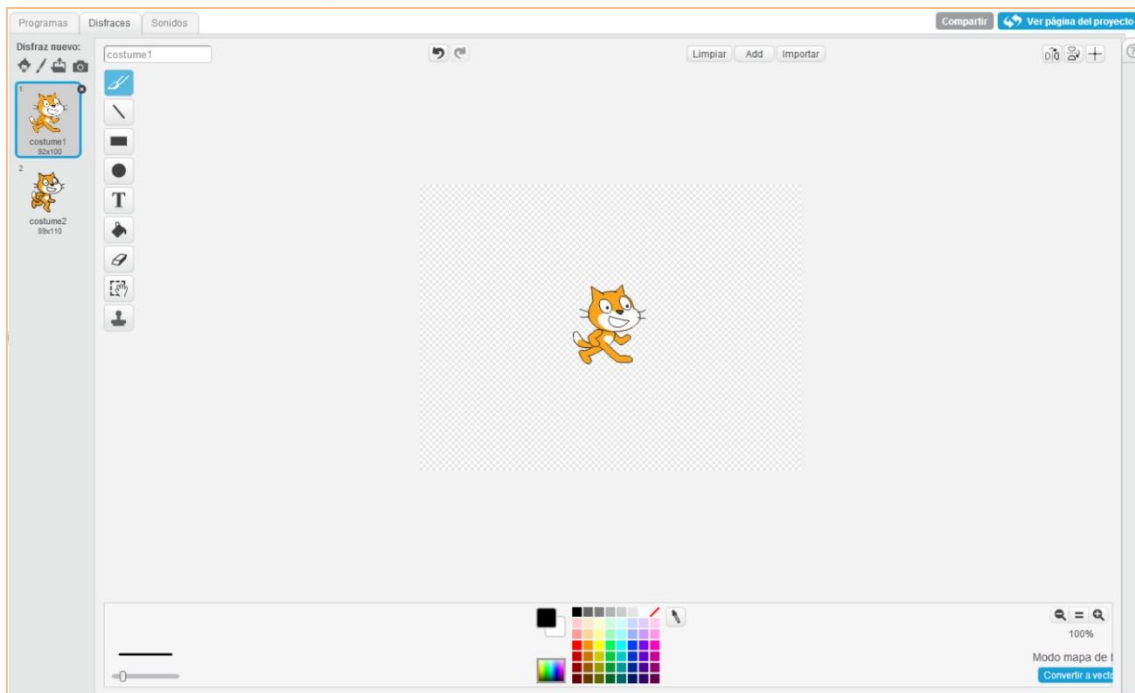
Presenta como novedad la opción de manejar y trabajar los gráficos en formato vectorial (de esta forma se puede aumentar el tamaño de los mismos sin que éstos pierdan resolución). Las imágenes, en scratch 2.0, se pueden trabajar en formato bitmap o en formato vectorial, según nos interese, sin más que hacer clic en un modo u otro (parte inferior derecha del Editor de Pinturas):



*Formatos de Imágenes.* Susana Oubiña Falcón ([CC BY](#))











Son formatos con características diferentes. La imagen bitmap (jpg, tif o bmp) se basa en un mapa de bits y, por lo tanto, contienen más información (ocupan más que las vectoriales). En ellas, cada bit (pixel) se puede modificar de forma independiente. El tamaño de la imagen es proporcional a la ampliación que se haga de la misma, de modo que, si queremos ampliar una imagen de tamaño pequeño, nos aparecerá el clásico pixelado y el diente de sierra en los bordes. Las imágenes vectoriales (crd, ink) se crean a partir de vectores y no se pueden dividir en píxeles y son idóneas para trabajar con ampliaciones, ya que mantienen la nitidez y definición de los bordes por mucho que se amplíen.

Al ser imágenes con características diferentes, la barra de herramientas de sus editores también difiere un poco. El Editor de Pinturas de una imagen en formato de bits es el siguiente:



*Editor de Pinturas (imagen mapa de bits). Susana Oubiña Falcón ([CC BY](#))*











La barra de herramientas del editor de pinturas, para una imagen en formato de mapa de bits, presenta las siguientes opciones:

Herramientas	Descripción
Mapa de Bits	
	Pincel: Permite pintar libremente usando el color del primer plano. Al hacer clic en esta herramienta, en la parte inferior izquierda (*) aparece el tamaño y grosor de la brocha (pincel).
	Línea: Permite pintar líneas rectas usando el color del primer plano. Al hacer clic en esta herramienta, en la parte inferior izquierda (*) aparece el tamaño y grosor de la brocha (pincel).
	Rectángulo: Dibuja un rectángulo sólido o su contorno usando el color actual del primer plano. Para hacer un cuadrado se debe presionar Shift+drag. Haciendo clic en esta herramienta, el Área de Opciones muestra el estilo de llenado, sólido o su contorno. El grosor de la línea de contorno la determina el tamaño de la brocha del lápiz.
	Elipse: Dibuja una elipse sólida o su contorno (presionar Shift+drag para hacer un círculo) usando el color actual del primer plano. Haciendo clic en esta herramienta, el Área de Opciones muestra el estilo de llenado sólido o su contorno. El grosor de la línea de contorno la determina el tamaño de la brocha del lápiz.
	Texto: Añade texto al dibujo.
	Rellenar con color: Con ella se rellena de color sólido o gamas (gradientes) de éste, áreas interconectadas. Al hacer clic sobre ella, el Área de Opciones muestra las posibilidades de llenado (color sólido, gradiente horizontal, gradiente vertical o gradiente radial). Los gradientes se mezclan partiendo del color del primer plano y van hacia el color seleccionado para el fondo.
	Borrar: Borra con movimientos libres de la mano. Las áreas que se borran se vuelven transparentes. Al hacer clic en esta herramienta, el Área de Opciones muestra los tamaños del borrador (*).
	Seleccionar: Selecciona una región rectangular y permite moverla a una nueva ubicación (presionar la tecla Suprimir/Delete para remover el área seleccionada; presionar Shift+delete o Shift+backspace para borrar todo lo demás y dejar solo el área seleccionada).
	Seleccionar y duplicar: Selecciona una región rectangular y permite copiarla en una nueva ubicación.
	(*) Deslice el control para escoger diferentes tamaños de brocha o del borrador.

Herramientas para una imagen mapa de bits. Susana Oubiña Falcón ([CC BY](#))

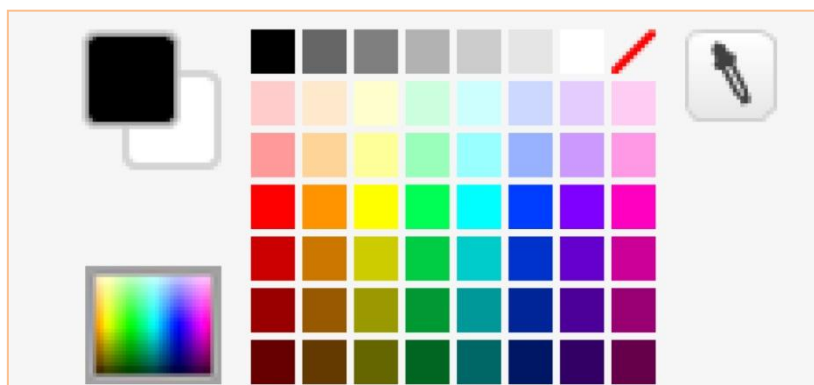
La barra de herramientas del editor de pinturas, para una imagen en formato vectorial, presenta las siguientes opciones:



Herramientas	Descripción
<b>Vectoriales</b>	
	Seleccionar: Selecciona una región rectangular y permite moverla a una nueva ubicación (presionar la tecla Suprimir/Delete para remover el área seleccionada; presionar Shift+delete o Shift+backspace para borrar todo lo demás y dejar solo el área seleccionada).
	Reformatar: permite modificar, desplazar, mover (o deformar) las líneas u contornos de una imagen o figura pintada.
	Lápiz: Permite pintar libremente usando el color del primer plano. Al hacer clic en esta herramienta, en la parte inferior izquierda (*) aparece el tamaño y grosor de la brocha (pincel).
	Línea: Dibuja una línea recta (presionar Shift+drag para trazar una línea horizontal o vertical) utilizando el color actual del primer plano. Al hacer clic en esta herramienta, el Área de Opciones muestra los diferentes tamaños de la brocha del lápiz (*).
	Rectángulo: Dibuja un rectángulo sólido o su contorno usando el color actual del primer plano. Para hacer un cuadrado se debe presionar Shift+drag. Haciendo clic en esta herramienta, el Área de Opciones muestra el estilo de llenado, sólido o su contorno. El grosor de la línea de contorno la determina el tamaño de la brocha del lápiz.
	Elipse: Dibuja una elipse sólida o su contorno (presionar Shift+drag para hacer un círculo) usando el color actual del primer plano. Haciendo clic en esta herramienta, el Área de Opciones muestra el estilo de llenado sólido o su contorno. El grosor de la línea de contorno la determina el tamaño de la brocha del lápiz.
	Texto: Añade texto al dibujo.
	Colorear una forma: Con ella se rellena de color sólido o gamas (gradientes) de éste, áreas interconectadas. Al hacer clic sobre ella, el Área de Opciones muestra las posibilidades de llenado (color sólido, gradiente horizontal, gradiente vertical o gradiente radial). Los gradientes se mezclan partiendo del color del primer plano y van hacia el color seleccionado para el fondo.
	Duplicar: Selecciona una región rectangular y permite copiarla en una nueva ubicación.
	(*)Deslice el control para escoger diferentes tamaños de lápiz.



Los colores, para ambos formatos de imagen, se muestran en la paleta de colores:





*Paleta de Colores.* Susana Oubiña Falcón ([CC BY](#))

Los **Colores Actuales** (para primer plano y fondo) se muestran debajo del Lienzo. Para cambiar los colores del primer plano y del fondo se selecciona el cuadrado del color deseado. Al hacer clic en uno de los **Colores de la Paleta** se elige un nuevo color para el primer plano.

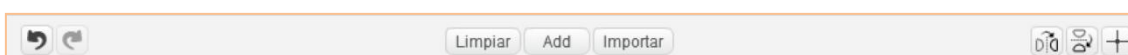
Para intercambiar la Paleta por defecto y la Paleta de color continuo, se hace clic en la **Paleta Alternar**.

Hay dos herramientas importantes en esta paleta de colores: la herramienta gotero y el color transparente:



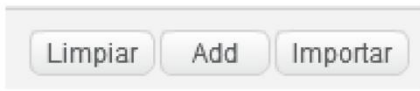
	<p><b>Gotero:</b> Para conocer el color determinado de una parte de una imagen podemos usar la herramienta gotero (Recoger un color).</p>
	<p>A veces, cuando importamos una imagen, podemos encontrarnos con que esa imagen presenta un fondo de color blanco, un fondo que no nos interesa conservar. Para recortar esa imagen podemos usar el color de relleno transparente para rellenar o cubrir parte de la imagen (en nuestro caso, el fondo que no nos interesa tener) con el color transparente. De esa forma, lo que veremos es un fondo transparente y nuestra imagen recortada.</p>

*Gotero y Color Transparente.* Susana Oubiña Falcón ([CC BY](#))

El editor dispone de unas herramientas en el menú superior:



Estas herramientas se describen en la siguiente tabla:

	<p>Gira horizontalmente la imagen.</p> <p>Gira verticalmente la imagen.</p> <p>Sitúa el origen que tendrá la imagen en el escenario en el lugar que deseemos (centrar la imagen)</p>
	<p>Volver hacia atrás (deshacer) un paso o hacia delante</p>
	<p>Borrar una imagen</p> <p>Añadir una imagen</p> <p>Importar una imagen</p>

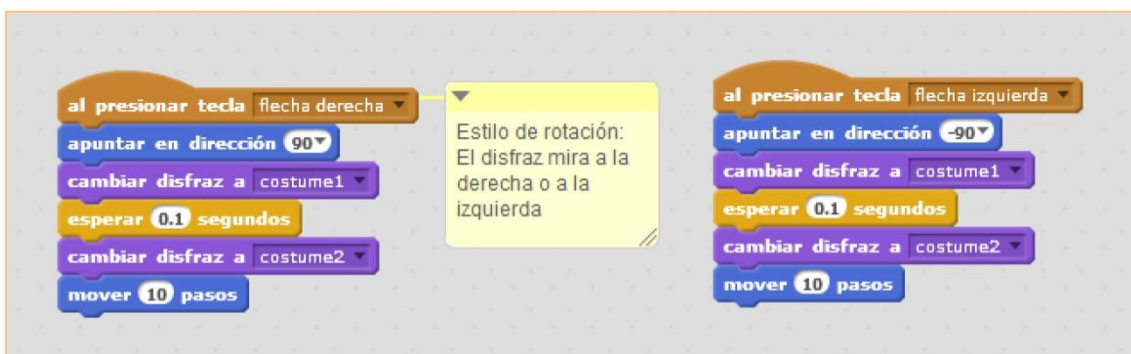
*Menú superior del Editor de Pinturas. Susana Oubiña Falcón ([CC BY](#))*

Finalmente, se puede hacer zoom en la imagen, ampliarla y hacerla más pequeña. Esto es de ayuda cuando se está retocando, para ver de forma más precisa posibles defectos.

### C. Ejemplos de los diferentes bloques (scripts)

#### ***Movimiento de un objeto por el escenario***

1. Ejemplo: En este programa, el objeto presenta el estilo de rotación derecha-izquierda. De esta forma, evitamos que se voltee cuando presionamos la tecla izquierda. Es decir, cuando gire, sólo lo hará de forma horizontal y no vertical.

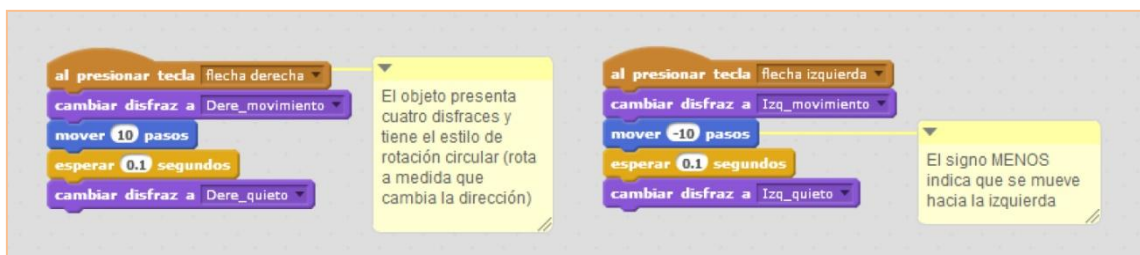


*Ejemplo1: Script para movimiento Horizontal. Susana Oubiña Falcón ([CC BY](#))*



Script para movimiento Horizontal y Vertical. Susana Oubiña Falcón ([CC BY](#))

2. Ejemplo: En este programa, el objeto presenta el estilo de rotación circular y utiliza 4 disfraces para el objeto.

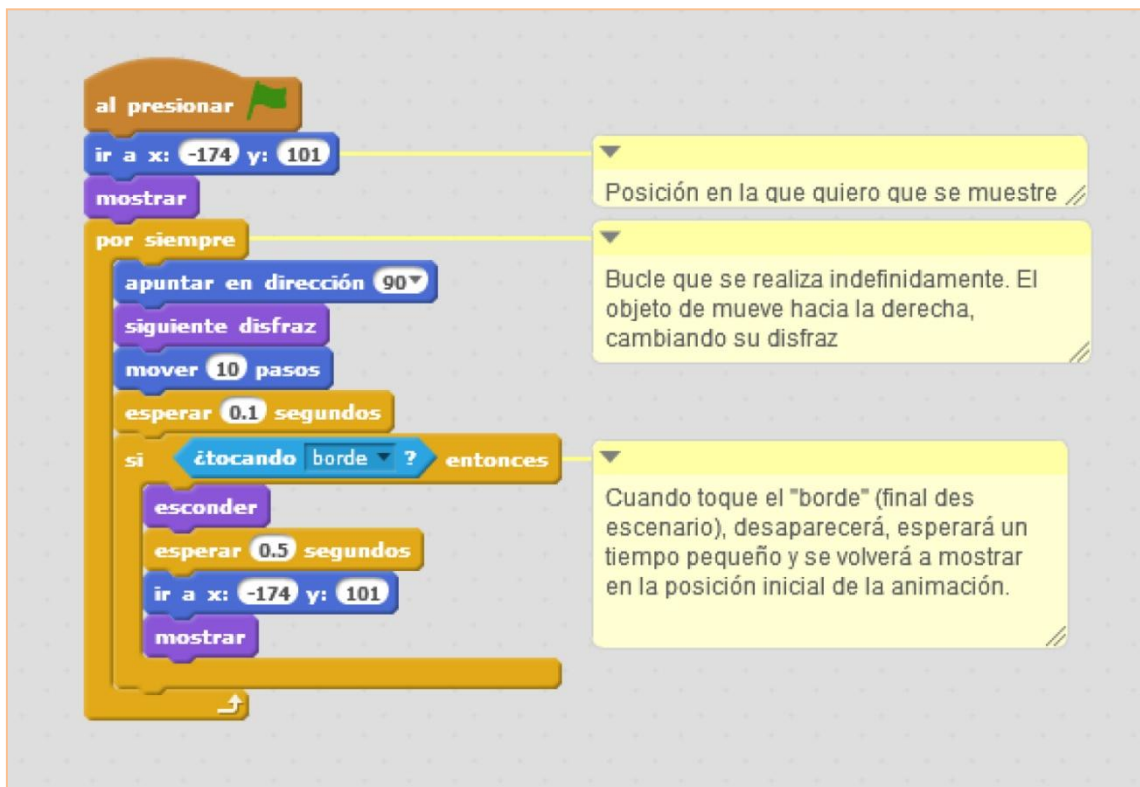


Script para movimiento Horizontal. Susana Oubiña Falcón ([CC BY](#))



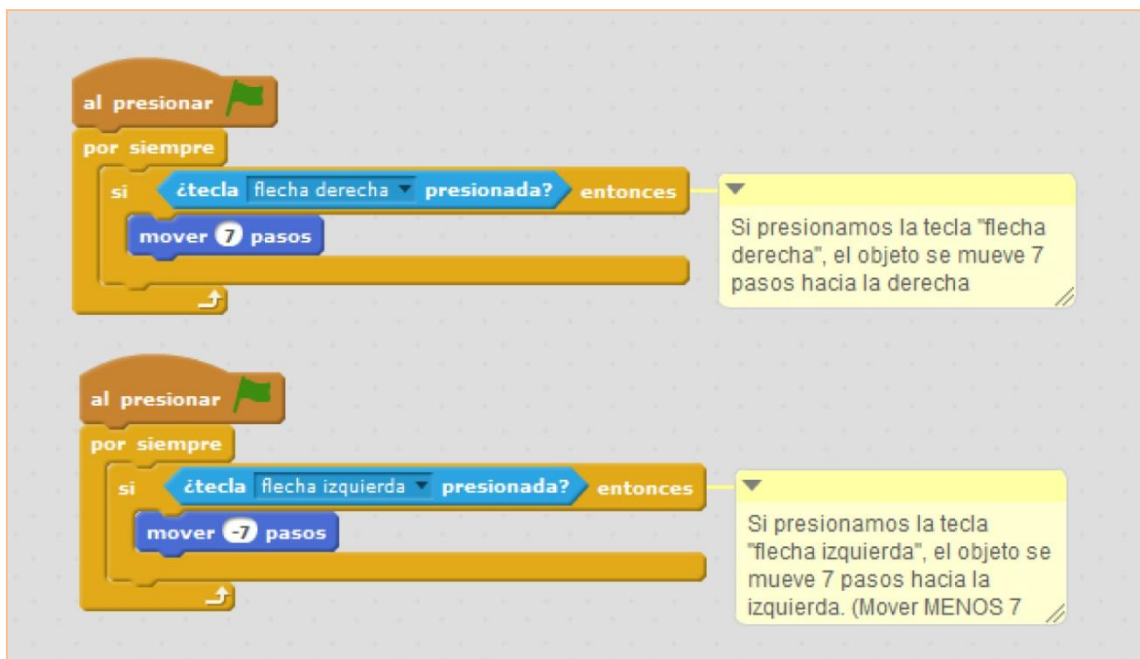
Disfraces de movimiento horizontal. Susana Oubiña Falcón ([CC BY](#))

3. Ejemplo: Movimiento indefinido utilizando un bucle para el objeto que llamé "pez grande".



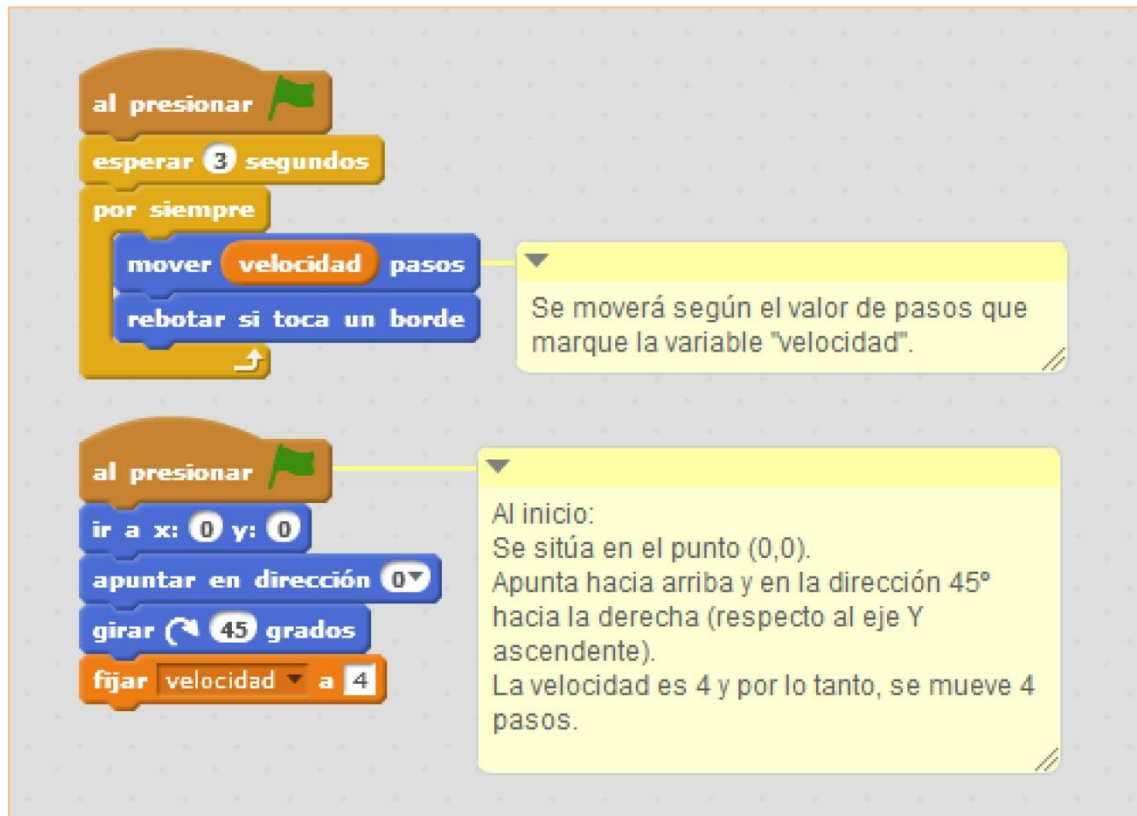
Script de una animación de movimiento horizontal continuo. Susana Oubiña Falcón ([CC BY](#))

4. Ejemplo: Movimiento con condicionales. En el ejemplo el objeto puede ser una barra, la cual queremos que se mueva de derecha a izquierda.



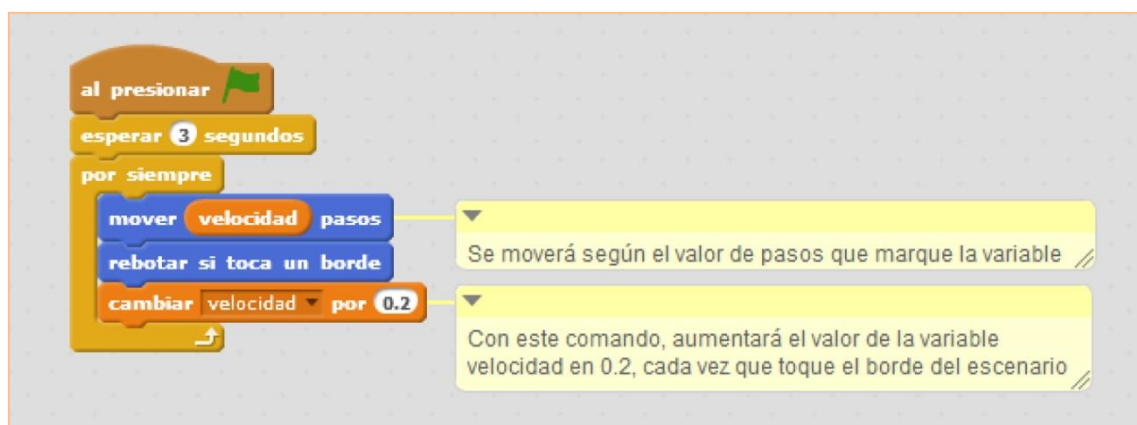
Script para un movimiento horizontal con condicionales. Susana Oubiña Falcón ([CC BY](#))

5. Ejemplo: Utilizar variables en el movimiento. En este ejemplo vamos a utilizar el objeto “Bola”. El objeto “bola” se moverá atendiendo a los pasos que le marque la variable “velocidad”, que se le da el valor “4”. Es decir, se moverá 4 pasos.



*Movimiento de un objeto utilizando una variable.* Susana Oubiña Falcón ([CC BY](#))

Si queremos que el valor de la variable “velocidad” vaya cambiando. Por ejemplo, aumentando en 0,2 pasos cada vez que rebote en el borde del escenario el objeto que se mueve, sólo debemos incluir una sentencia en el bucle por siempre, tal y como se muestra en la siguiente imagen.

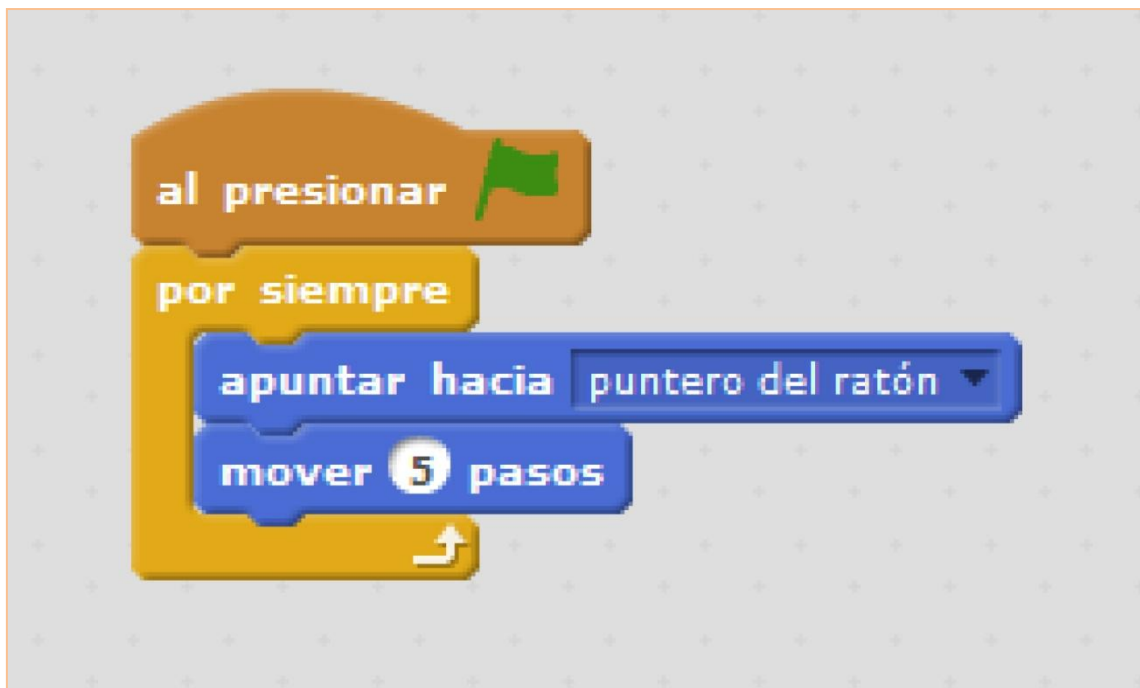


*Ejemplo del comando “Cambiar” del bloque Datos.* Susana Oubiña Falcón ([CC BY](#))



Es importante saber que, en Scratch, el comando **Cambiar...** en el bloque datos, significa sumar. Por ello, si queremos aumentar, debemos cambiar el valor de la variable por un número positivo y si queremos disminuir el valor de la variable, debemos cambiarlo por un número negativo.

6. Ejemplo: Movimiento siguiendo al puntero del ratón. A veces, bien sea para un juego o para otro tipo de proyecto, nos interesa que el objeto se mueva siguiendo al puntero del ratón del usuario que hace correr el programa. Para ello, bastaría con crear el siguiente script para el objeto que queremos que realice este tipo de movimiento:



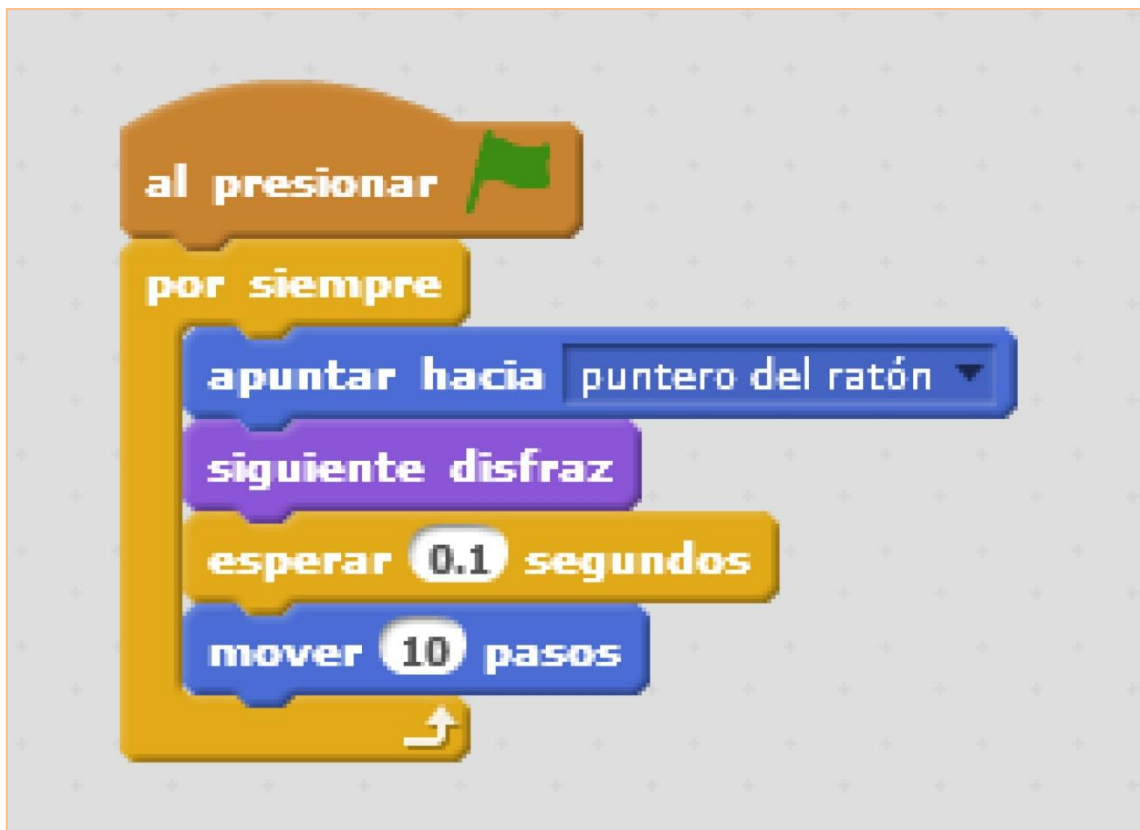
*El objeto se dirige al "puntero del ratón". Susana Oubiña Falcón ([CC BY](#))*

En nuestro ejemplo queremos además que en su movimiento vaya variando su disfraz, ofreciendo mayor movilidad y animación al proyecto. En ese caso, el script (para un objeto "pulpo" con 2 disfraces), sería el siguiente.



*El objeto se dirige al “puntero del ratón” con animación. Susana Oubiña Falcón ([CC BY](#))*

Si el objeto presenta más de 2 disfraces, el programa se ve demasiado largo. Particularmente, es más funcional utilizar el siguiente script, el cual es funcional independientemente de si el número de disfraces de un objeto es elevado o no:



*El objeto se dirige al “puntero del ratón” con animación. Susana Oubiña Falcón ([CC BY](#))*

**NOTA:** En estos ejemplos hemos ido introduciendo algunos comandos de la pestaña Control del scratch. Estos comandos de control son los bucles (o repeticiones) y los condicionales:

Los bucles nos permiten que una secuencia de comandos se repita: un número de veces, siempre, o hasta que ocurra un evento:




*Comandos de repetición. Susana Oubiña Falcón ([CC BY](#))*

Los condicionales son comandos “Si” y permiten programar una acción si ocurre una condición o; realizar una acción en si se cumple una condición pero en caso de que no se cumpla, programar otra acción:





Comandos condicionales. Susana Oubiña Falcón ([CC BY](#))

Estos comandos presentan un espacio hexagonal  y en él se introducen los elementos de los bloques de **Sensores** y **Operadores** disponibles en el scratch.

### ***Eventos internos y externos en el objeto***

Un objeto, en scratch, responde a eventos. Los eventos pueden ser internos o externos. Los internos son aquellos que comunican unos objetos con otros de forma interna, por ejemplo mediante mensajes; los externos requieren para esta comunicación de periféricos externos como pueden ser teclas del teclado o el ratón (cuando pulsamos una tecla, cuando hacemos clic en los objetos, etc).

En los ejemplos de movimiento ya hemos utilizado algún evento externo. Eran las teclas o flechas de movimiento (derecha, izquierda, arriba y abajo), de modo que, si pulsábamos una de esas teclas, apuntaba a esa dirección y se movía un determinado número de pasos.

En el siguiente ejemplo vamos a utilizar eventos internos a través de mensajes.

7. Ejemplo: Cambio de fondos, control del botón Play de un proyecto y control de una animación utilizando mensajes.

El programa tiene 2 objetos o sprites (pez grande y Play) y dos fondos cuyos nombres son "Comenzo" y "Fin".



Objetos del programa. Susana Oubiña Falcón ([CC BY](#))

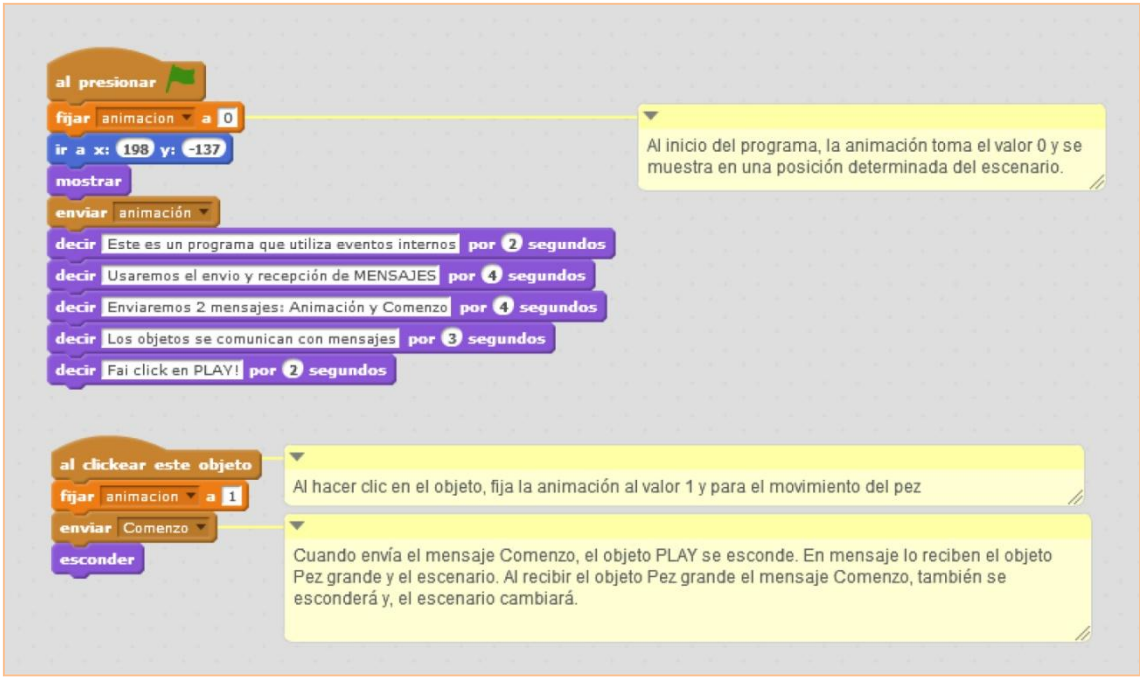
En el programa se utiliza la variable animación de modo que sólo puede tomar dos valores: 0 y 1. Al inicio toma el valor 0 y mientras tenga este valor, el objeto “pez grande” se moverá indefinidamente en dirección horizontal hacia la derecha por el escenario (rebotando y volviendo a hacer el movimiento). Pero, cuando tome el valor 1, el objeto pez dejará de hacer ese movimiento.

En el programa se han creado dos mensajes: Comenzo y Animación. El emisor y receptores de los mensajes así como la acción que se realiza se describen en la siguiente tabla:

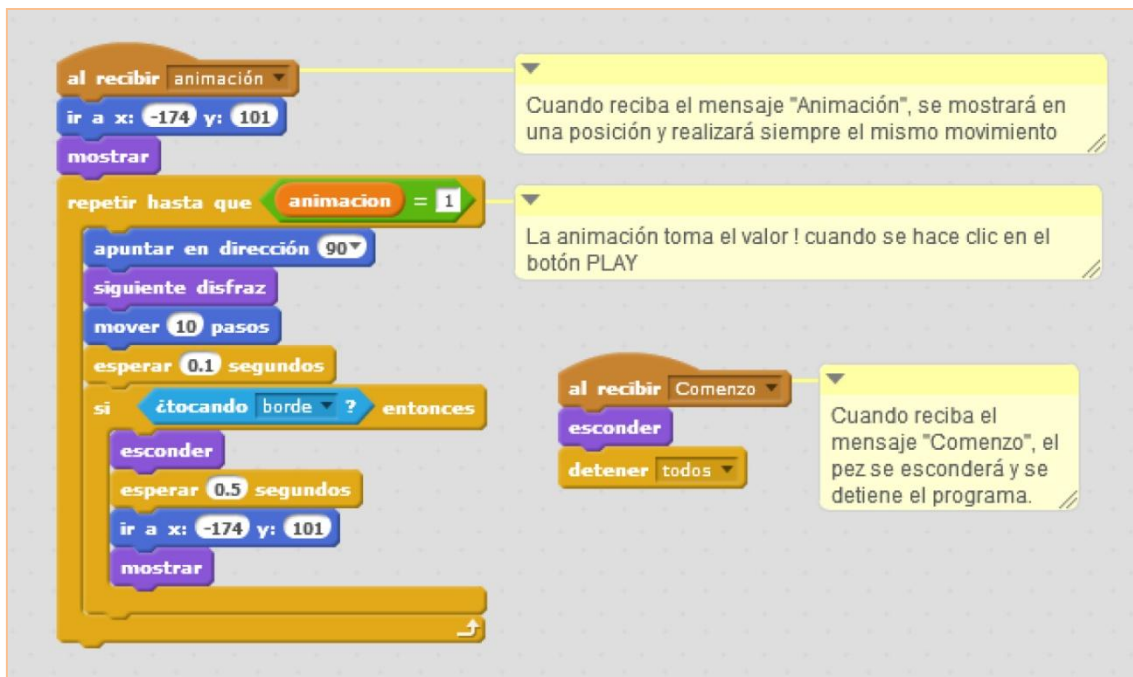
<b>Mensaje “Comenzo”</b>	Lo <i>envía</i> el objeto PLAY al hacer clic sobre el objeto.	Lo <i>recibe</i> el objeto “pez grande” y el Fondo y cada uno realiza una acción al recibirlo: <ul style="list-style-type: none"> <li>- Pez grande: se esconde y para el programa</li> <li>- Fondo: cambia el fondo a fondo “Fin”</li> </ul>
<b>Mensaje “Animación”</b>	Lo <i>envía</i> el botón PLAY al presionar nosotros la bandera verde (evento externo)	Lo <i>recibe</i> el objeto “pez grande” y, al hacerlo, realiza una acción. Su acción será que el pez se mueva por el escenario 8 en la forma programada) mientras la variable animación tome el valor 0.

*Descripción de los mensajes.* Susana Oubiña Falcón ([CC BY](#))

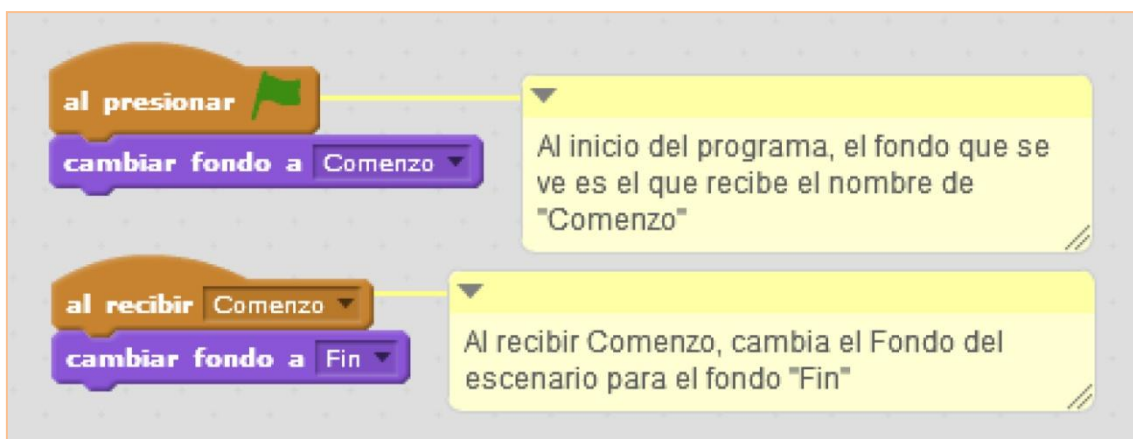
Los scripts de ambos objetos y de los fondos del escenario se muestran a continuación:



*Script del botón PLAY.* Susana Oubiña Falcón ([CC BY](#))



Script del Objeto "pez grande". Susana Oubiña Falcón ([CC BY](#))



Script del Fondo. Susana Oubiña Falcón ([CC BY](#))

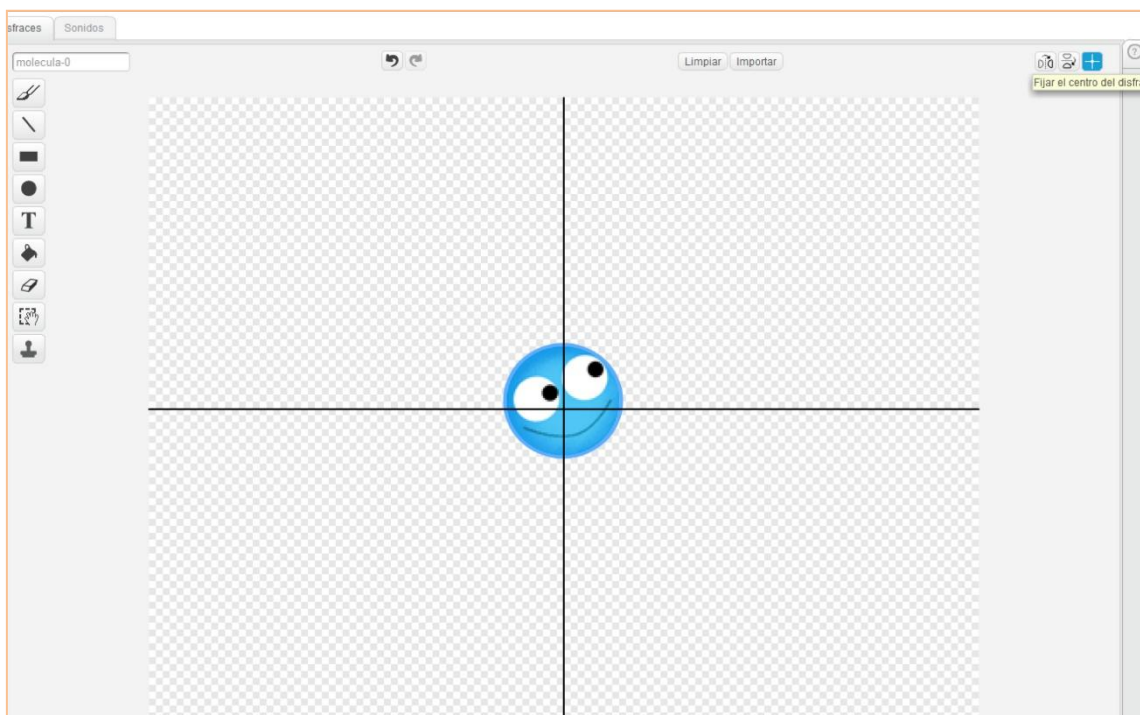
### **Nuevo elemento de control: "el clon"**

Scratch 2.0 permite realizar clonaciones de objetos. Esta función no estaba implementada en el scratch 1.4. Es muy interesante ya que a veces, en un programa, necesitamos muchos objetos iguales, que hagan la misma función. Por ejemplo, crear moléculas para un recipiente y que todas ellas se controlen de la misma forma. Por lo tanto, la clonación evita multiplicar objetos y su código correspondiente. Los comandos que se pueden utilizar en clones de un objeto son: Crear un clon, borrar un clon y programar qué acción debe hacer ese clon cuando comience como clon de un objeto.



Comandos para clones. Susana Oubiña Falcón ([CC BY](#))

8. Ejemplo: En el siguiente ejemplo tenemos el objeto molécula (centrada para asegurarnos de que se moverá como deseamos. Ver siguiente imagen).



Centrar un Disfraz. Susana Oubiña Falcón ([CC BY](#))

El movimiento del objeto molécula inicial es iterativo: se mueve 10 pasos de forma indefinida deslizándose hacia una posición y con una dirección, ambas aleatorias, por el escenario y dejando que rebote al tocar el borde del mismo:



Script del movimiento de la molécula. Susana Oubiña Falcón ([CC BY](#))

Los clones del objeto molécula se crearán cada medio segundo y esperarán 0.3 segundos antes de volver a moverse (el movimiento es el mismo que el del objeto molécula):

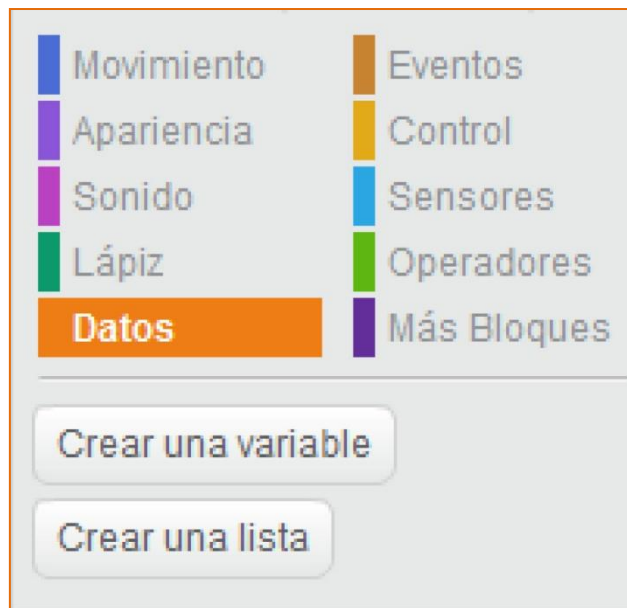


Clonar el objeto molécula. Susana Oubiña Falcón ([CC BY](#))

## Variables y Listas

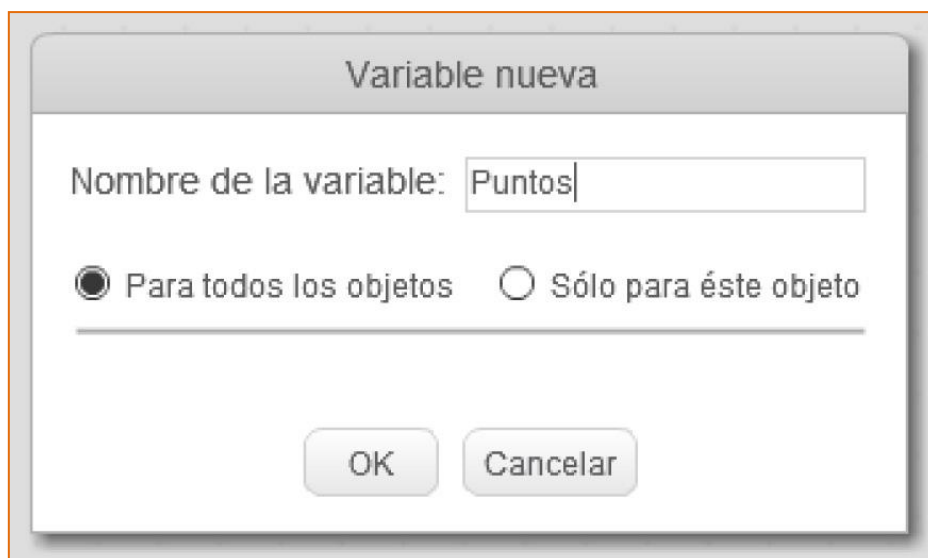
En el ejemplo 5 de movimiento ya habíamos introducido la variable “velocidad”. Scratch, en el bloque de **Datos**, permite crear variables y listas. Las variables son muy importantes porque nos permiten almacenar datos dentro de la memoria RAM del ordenador. En ella podemos almacenar un número, una palabra o un objeto. Una lista es un listado de número, palabras, etc, que podemos almacenar.

Si queremos crear una variable o una lista, hacemos clic en Crear una variable o Crear una lista en el bloque de Datos de la pestaña programas (ver siguiente imagen).



*Bloque Datos.* Susana Oubiña Falcón ([CC BY](#))

Al hacer clic en “Crear una variable” se nos pide que le demos un nombre a la misma. Por ejemplo, crearemos la variable puntos (escribimos Puntos y hacemos clic en OK). Ésta puede crearse sólo para ese objeto o para todos los objetos del programa (conocida por todos los objetos del programa y por lo tanto, ellos podrán acceder a su valor y modificarlo), que será lo que haremos. En la versión online hay una tercera opción que es una variable en la nube que es aquella a la que puede acceder cualquier persona que ejecute nuestro código (podría usarse para guardar las mejores puntuaciones en un juego o mensajes en el mismo).



*Ventana “Nombre de variable”.* Susana Oubiña Falcón ([CC BY](#))



Tras este paso, en el escenario de prueba del programa puede verse la variable Puntos (con el valor 0). Además, en el bloque Datos nos aparecen nuevos comandos o sentencias para esa variable.

Si no queremos que se muestre en el escenario, desmarcamos la variable Puntos:

Si queremos que la variable comience por otro valor diferente a 0, utilizamos el comando fijar...por..., y en el lugar de 0, introducimos el número deseado.

En comando cambiar...por... significa añadir o sumar. Así pues, si queremos que vaya sumando un punto al realizar una acción, introducimos el número 1, pero si queremos que reste un punto, introduciremos el número -1.

En ciertos momentos nos interesará mostrar o esconder la variable puntos y para ello utilizamos los siguientes comandos:

*Comandos de las variables.* Susana Oubiña Falcón ([CC BY](#))

Al crear una lista también se nos pide el nombre de la misma (con las mismas opciones para todos los objetos, sólo ese o en la nube).

*Ventana "Nombre de lista".* Susana Oubiña Falcón ([CC BY](#))

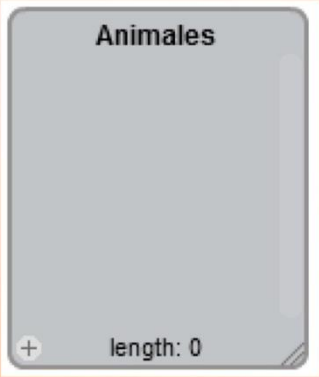





Los comandos de las Listas son similares a los de las variables, con un abanico más amplio de posibilidades: añadir “elementos” a la lista animales, borrar un ítem determinado o todos los elementos de la lista animales, insertar un elemento en un ítem determinado, reemplazar un elemento de la lista, calcular la longitud de la lista animales, decir si la lista contiene un determinado nombre y mostrar o esconder la lista en el escenario.

*Comandos de las Listas.* Susana Oubiña Falcón ([CC BY](#))

Al crear una lista nos aparece en el escenario un “monitor de lista”, que podemos desmarcar para no hacerlo visible. Ese monitor muestra todos los elementos de esa lista. Lista que podremos modificar e introducir elementos nuevos desde allí:



Inicialmente, la lista está vacía y en consecuencia, su longitud (length) marca el valor 0.

Al introducir datos en ella, la longitud variará. Para meter datos podemos hacerlo desde el propio escenario, haciendo clic en el icono inferior de suma: 

Si introducimos de esa forma dos animales a la lista animales, se mostrará la siguiente imagen:

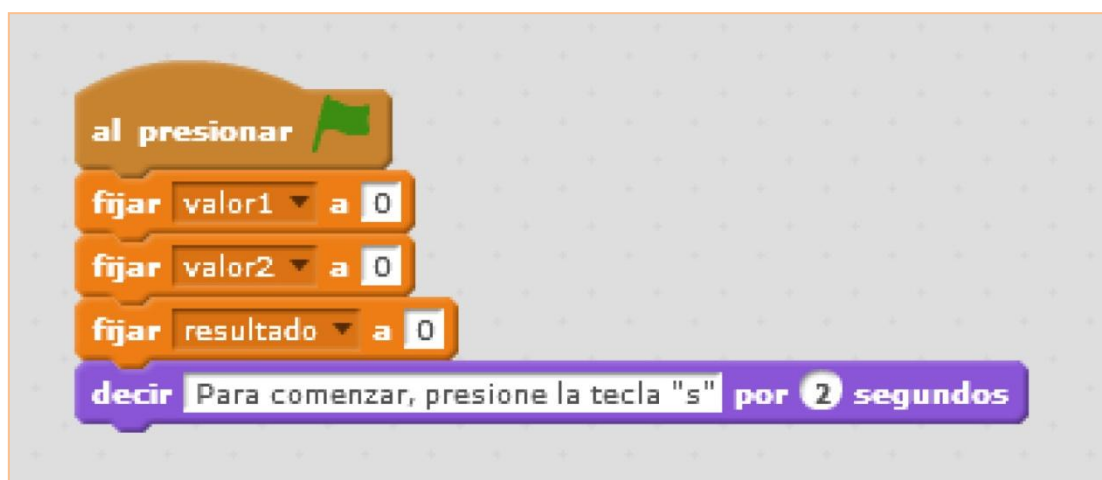
Al hacer clic sobre el monitor de lista del escenario, observamos que se nos abren 3 opciones: importar, exportar y esconder la lista. Para acceder a cualquiera de esas opciones, debemos hacer clic en ellas (para Mac usamos Ctrl+shift). La exportación de los elementos de la lista la realiza generando un archivo con extensión .txt, de modo que, para importar elementos con valores en líneas separadas, el archivo de importación también debe ser .txt.

*Monitor de Listas. Susana Oubiña Falcón* ([CC BY](#))

Veamos algunos ejemplos.

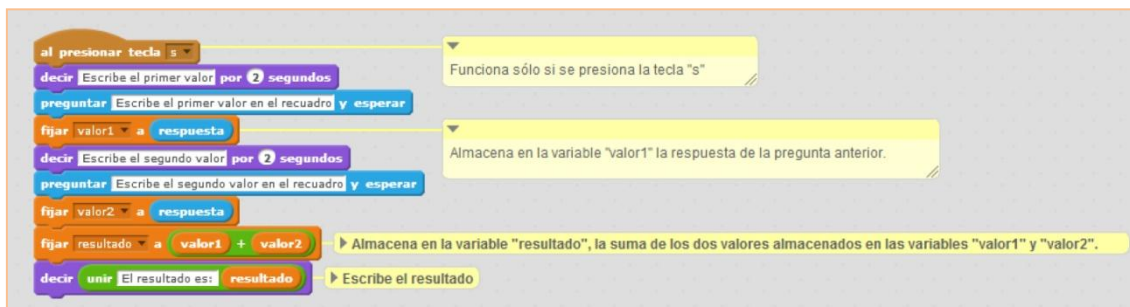
9. Ejemplo: Vamos a crear un pequeño programa que utilice variables. El programa almacenará el valor de dos números, los sumará y almacenará este valor para después mostrarlo. Para ello usamos un único objeto, el objeto “gato”, que será el que pida los datos y los vaya almacenando para finalmente entregar el resultado de su suma. Por lo tanto, necesitamos crear 3 variables que llamaremos “valor1”, “valor2” y “resultado”.

Para que el programa realice sumas sin tener en cuenta los datos introducidos con anterioridad (en otras ejecuciones del mismo), así como, su operación anterior, debemos inicializar las variables siempre al valor cero (ver siguiente imagen):



*Inicializando variables. Susana Oubiña Falcón* ([CC BY](#))

En el script anterior, podemos observar que nuestro programa comenzará cuando el usuario presione la tecla “s”. Por lo tanto, la comunicación con el objeto iniciará cuando se presione el comando “al presionar tecla...” del bloque Evento:



Script para sumar dos números. Susana Oubiña Falcón ([CC BY](#))

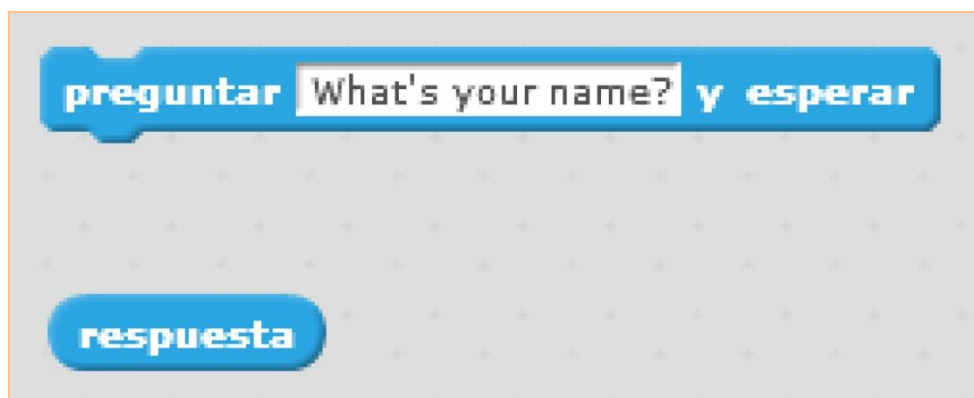
**NOTA:** Podríamos omitir el comando “decir Escribe el primer valor (o el segundo valor) por 2 segundos”. Es reiterativo ya que en la pregunta se debe dejar claro qué debe escribir el usuario del programa. Lo he introducido para que se vea la diferencia entre el comando “decir...por...segundos” y “preguntar....y esperar”.

En el bloque de sensores podemos ver que scratch presenta tres variables en color azul que se llaman, “cronómetro”, “día, minuto, hora, etc ...de ahora” y “respuesta”. También, en ese bloque, existe el comando “preguntar... y esperar”. Con ese comando podemos hacer la pregunta (petición de datos). Los datos introducidos se almacenan en la variable “respuesta” (uno a uno). Para la petición de datos, el comando abre un recuadro azul en el escenario, tal y como se observa en la siguiente imagen:



Funcionamiento del comando “preguntar...y esperar”. Susana Oubiña Falcón ([CC BY](#))

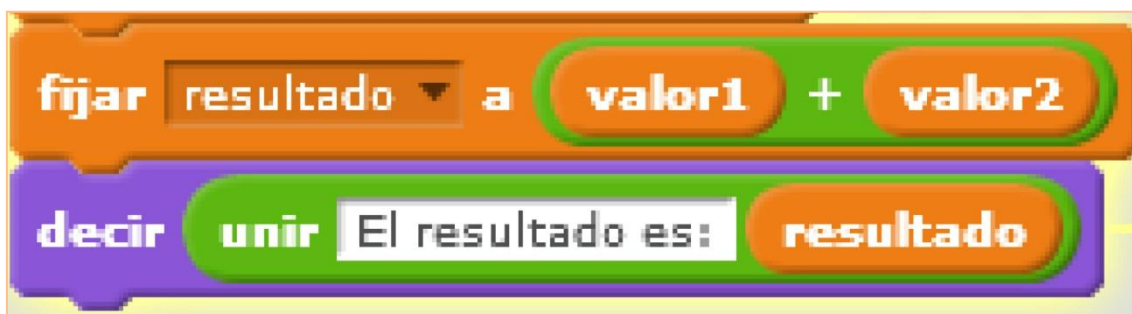
Por lo tanto, hay una conexión entre “preguntar...y esperar” y el valor o “respuesta” que nosotros introducimos. Forman el bloque Pregunta/Respuesta:



Comandos Pregunta/Respuesta. Susana Oubiña Falcón ([CC BY](#))

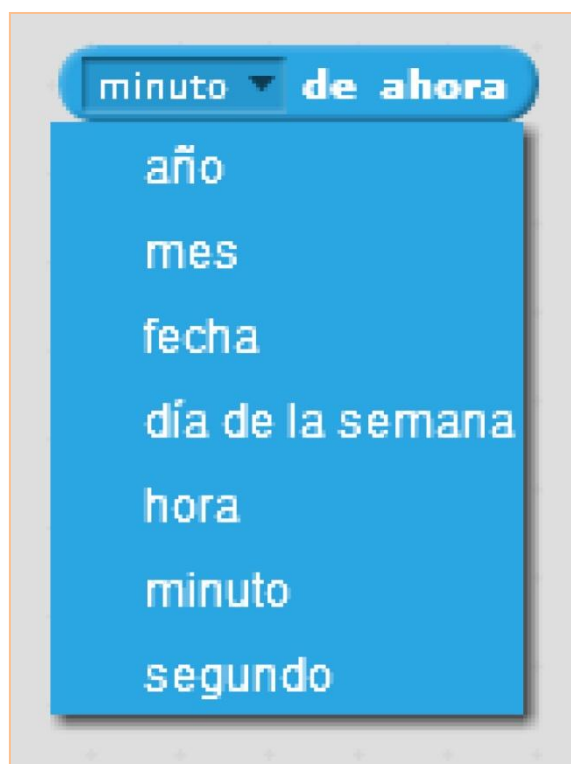
Como se muestra en el script de la figura siguiente, se define la variable resultado como la suma de las variables “valor1” y “valor2”. Para que muestre el valor de la suma el objeto “gato” debe “decir” el resultado. Una forma más elegante de escribirlo es con el comando “unir” del bloque Operadores. Con él

podemos unir un texto a la operación (suma de los dos números), tal y como se muestra en la siguiente imagen:



*Definir y mostrar "resultado". Susana Oubiña Falcón ([CC BY](#))*

10. Ejemplo: Vamos a crear un programa que nos diga el día de hoy. Scratch ya presenta un comando que nos hace esto: nos dice el año, mes, fecha, día de la semana, hora, minuto y segundo. Pero en la fecha no nos dice el nombre del mes ni el nombre del día de la semana. Nuestro programa lo hará.



*Opciones de la variable "...de ahora". Susana Oubiña Falcón ([CC BY](#))*

Para ello, necesitamos introducir dos variables, que llamaremos: "Nombre del día de la semana" y "Nombre del mes". Utilizamos un único objeto para mostrar la fecha. En él programamos los nombres de los meses y de los días de la semana.

En el script para los meses utilizamos un bucle por siempre en el que insertamos diferentes condicionales, de modo que si el mes es 1, fijará su nombre a Enero; si es 2, fijará su nombre a Febrero...y así sucesivamente hasta 12 con Diciembre. El script para los meses es el siguiente:





Script "Meses del año". Susana Oubiña Falcón ([CC BY](#))

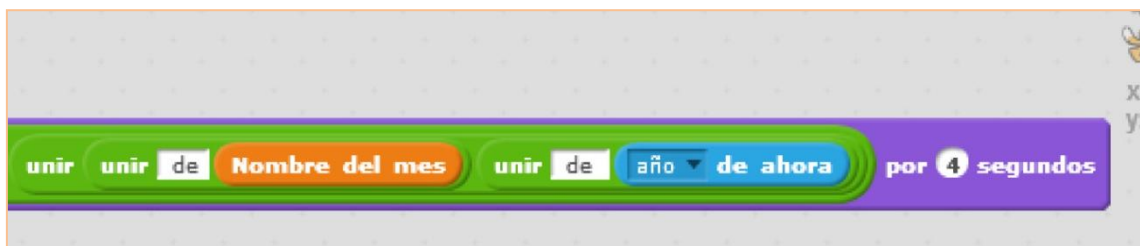
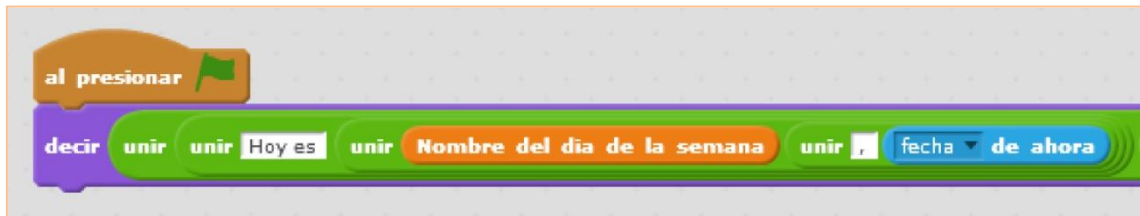


Para programar el nombre de los días de la semana, actuamos de la misma forma: usamos un bloque por siempre en el que incluimos condicionales de modo que a cada día de la semana (de la variable de ahora) le asociaremos un número, otorgándole a cada número el nombre de un día de la semana: 1 será Lunes, 2 será Martes y así sucesivamente hasta 7 Domingo. Es script para los días de la semana es el siguiente:



Script "Meses del año". Susana Oubiña Falcón ([CC BY](#))

Ya tenemos programado el nombre de los días de la semana y el nombre de los meses del año. Sólo nos falta conseguir que el objeto nos muestre esa fecha en el escenario. Para ello, introducimos el siguiente script:



Script “escribir la fecha”. Susana Oubiña Falcón ([CC BY](#))

La ejecución del programa nos muestra la fecha en el escenario:



Visualización en el escenario. Susana Oubiña Falcón ([CC BY](#))

11. Ejemplo: Vamos a crear, utilizando listas y variables, un programa que incluya 5 preguntas con sus respuestas. Las preguntas de nuestro ejemplo serán relativas al Sistema de Unidades Internacional. El objeto que interactuará con el usuario será el objeto “chico”.

Primero creo dos listas (ver Fig1), una para las preguntas y otra para las respuestas. A esas listas les doy el siguiente nombre: Preguntas creadas y Respuestas creadas.



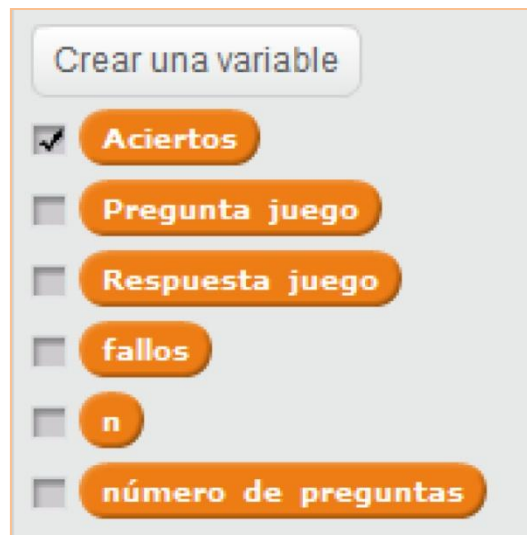
Fig1: Listas en el programa 11. Susana Oubiña Falcón ([CC BY](#))

Cada lista (Preguntas creadas o Respuestas creadas) tiene 5 ítems, es decir, he introducido 5 preguntas con sus correspondientes respuestas, tal y como se muestra en la siguiente imagen:



*Items de las listas del programa 11. Susana Oubiña Falcón ([CC BY](#))*

Además, necesitamos crear unas variables. En nuestro programa, las variables que he diseñado son las siguientes: Aciertos (que se mostrará en el escenario), Pregunta juego, Respuesta juego, fallos, número de preguntas, y "n" (que indicará el número de ítem (elemento) de las listas y, por lo tanto, comenzará necesariamente en el valor 1).



*Variables en el programa 11.* Susana Oubiña Falcón ([CC BY](#))

Estas variables deben siempre inicializarse al comienzo del programa con los valores deseados. En nuestro caso, los valores iniciales o de comienzo de ellas serán: 0 fallos, 0 aciertos, comienza por el ítem 1 de cada lista y comienza por el número de preguntas 1:



*Valores iniciales de las variables.* Susana Oubiña Falcón ([CC BY](#))

Como el programa está pensado para 5 preguntas, realizo un bucle que se repetirá sólo para esas 5 preguntas. Se repetirá hasta que la variable “número de preguntas” sea mayor que 5 (cuando el número de preguntas sea mayor que 5, el programa no ejecutará ese bucle). Cuando el programa comienza con cada pregunta hago coincidir las variables Pregunta juego y Respuesta juego

con el correspondiente ítem (variable n) de la lista de Preguntas y Respuestas creadas. Estos valores deben variar según el número de pregunta, por eso, las introduzco dentro del bucle, tal y como se muestra en la siguiente imagen:



*Asociación Preguntas/Respuestas de cada ítem.* Susana Oubiña Falcón ([CC BY](#))

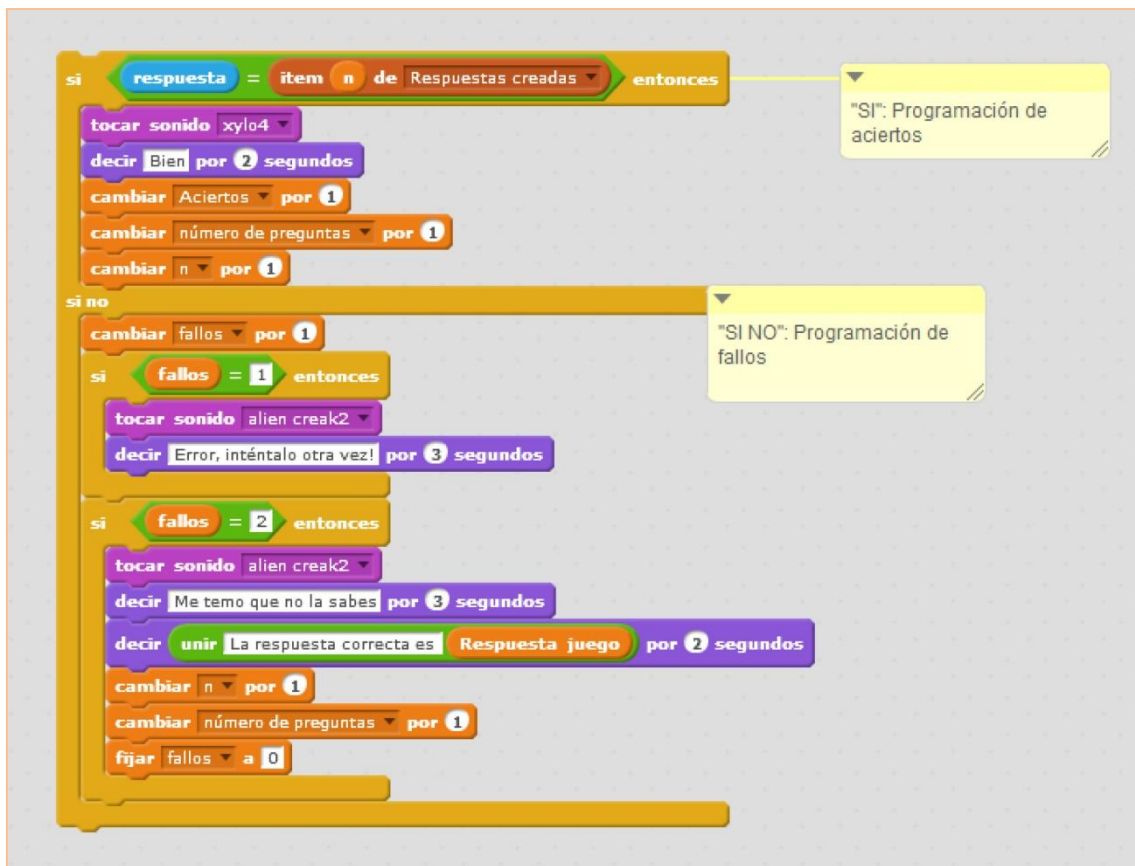
A continuación, debemos programar que el objeto “chico” realice la pregunta, haciendo variar en ella el correspondiente ítem de la listas de preguntas creadas:



*Pregunta asociada al ítem.* Susana Oubiña Falcón ([CC BY](#))

Tras realizar la pregunta, el programa obtiene una respuesta del usuario, que se almacena en la variable respuesta del scratch (del bloque sensores). La respuesta que da el jugador, puede ser acertada o no. Entonces, debemos tener en cuenta estas dos posibilidades y programar como sería acertar y como sería fallar. Para ello creamos un bloque “Si....si no...” de modo que, en la parte “si” se programará lo que queremos que haga cuando acierte y en la parte “si no”, lo que queremos que haga en caso contrario, es decir, cuando no acierte. El script de esta parte se muestra en la siguiente imagen:





*Aciertos y fallos. Susana Oubiña Falcón (CC BY)*

Como puede verse en la imagen anterior, si la respuesta cumple la condición de aciertos,

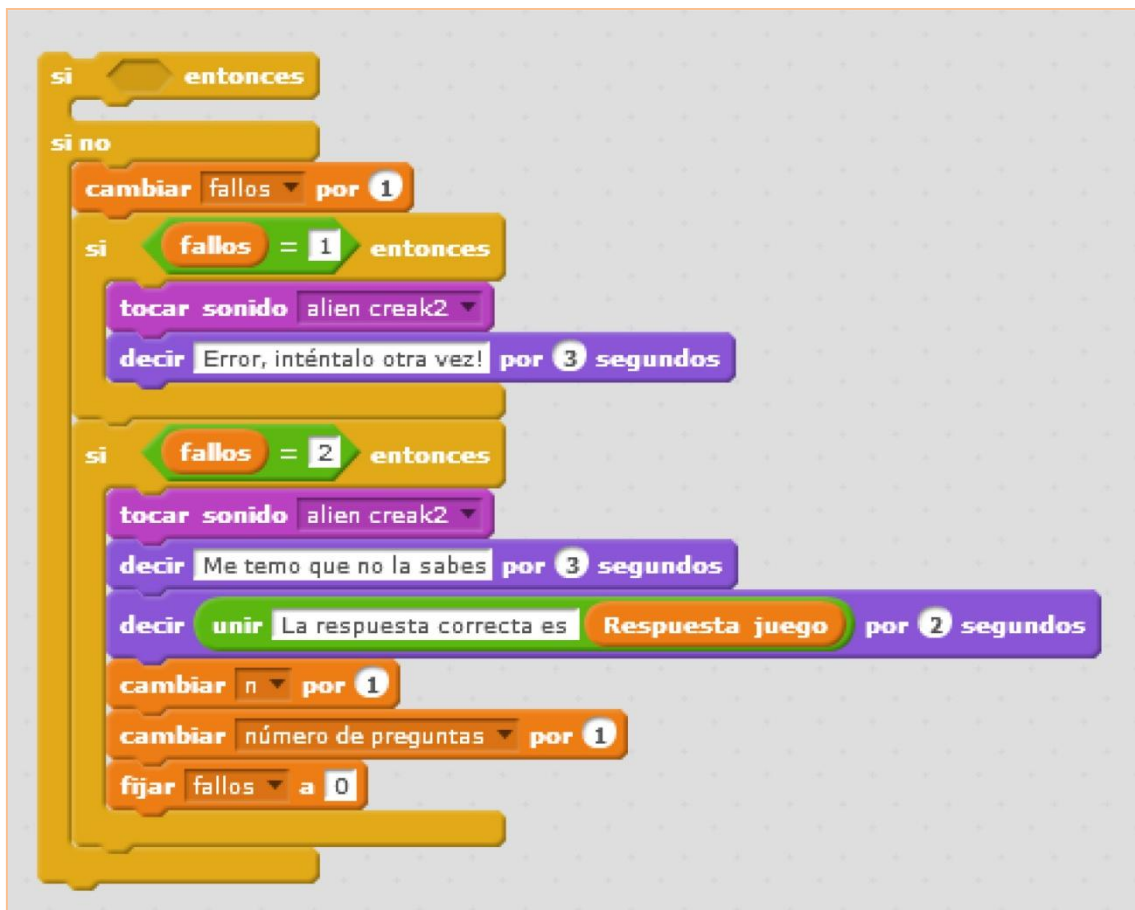


*Condición de aciertos. Susana Oubiña Falcón (CC BY)*

entonces, aumenta los aciertos en 1, al igual que el ítem (elemento) de las preguntas y el número de preguntas. Además, presenta un mensaje de acierto y hace sonar un sonido para ella.

En caso de que no se cumpla la condición de aciertos, programamos los fallos. Aquí, he querido dar otra posibilidad a la persona que ejecute el programa y que falle en el primer intento. Por lo tanto, debo programar lo que ocurrirá para 1 y 2 fallos.





Programación de fallos. Susana Oubiña Falcón ([CC BY](#))

En la parte “si no” del bloque se programan los fallos que sólo se realizará en el caso de que no se cumpla la condición de aciertos. Cuando falla, aumenta los fallos en 1, pero se pueden dar dos situaciones: que la falle en el primer intento o en el segundo intento

En el primer intento: simplemente mostramos el mensaje para que lo intente otra vez, con un sonido, pero queremos que mantenga la misma pregunta y su correspondiente respuesta de la lista, por eso, no cambiamos las variables “n” ni “número de preguntas”. Cuando la variable fallos=1, vuelve a hacer la misma pregunta que ha fallado.

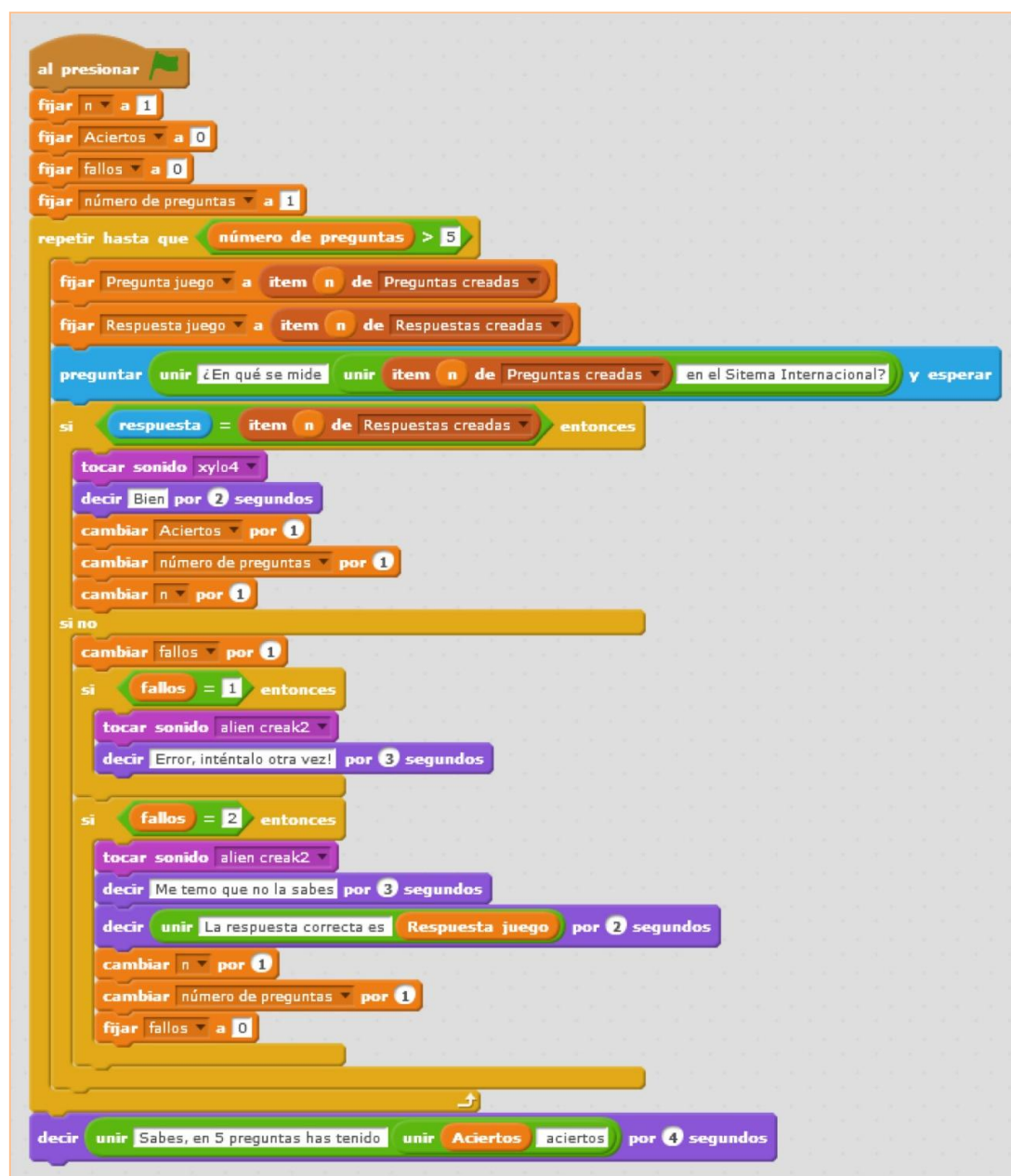
En el segundo intento: Si vuelve a fallar, se aumenta la variable fallos a 2 y se ejecuta el condicional de fallos=2 que, lo que hace, es decirle al usuario que ha vuelto a fallar y muestra el resultado correcto a la pregunta. Como queremos que ahora cambie de pregunta y del ítem de la lista de respuestas que hemos introducido, aumentamos la variable “n” y la variable “número de preguntas”. Para esa nueva pregunta, reinicio la variable “fallos” a cero ya que si no lo hago, partiría de 2 fallos.

Aunque en el escenario se muestra un recuento de aciertos (variable numérica Aciertos visible), quiero que el objeto “chico” muestre un mensaje con el recuento o cómputo final de aciertos. Para ello introduzco la siguiente línea de programa:



*Cómputo de aciertos.* Susana Oubiña Falcón ([CC BY](#))

El script completo del objeto “chico” es el siguiente:



*Script completo del programa 11.* Susana Oubiña Falcón ([CC BY](#))

## Ejemplos con sensores

Entre los ejemplos de variables y listas es interesante saber hacer un programa que reste vidas a un objeto, o que sume puntos. Ambas cosas las vamos a incluir en un programa mayor que utilice además comandos del bloque sensores.

12. Ejemplo: En este pequeño juego introducimos tres objetos: cangrejo, pulpo y pez. El cangrejo debe comerse a los peces y cada vez que toque al pez, éste desaparecerá y sumará un pez a su almuerzo. El juego finaliza si come 45 peces. Para darle más emoción al juego, introducimos un objeto “pulpo”. El pulpo es el enemigo del cangrejo y, cada vez que lo toque, le restará una de sus 5 vidas.

El programa requiere de 3 variables. Estas son: Peixes, Vidas do cangrexo y velocidad del pulpo. Esta última se crea para complicar el juego, de modo que, cuando el pulpo se encuentre cerca del cangrejo (a una distancia) se mueva hacia él. Inicialmente, se fijan las variables Peixes=0, Vidas do cangrexo=5 y velocidad del pulpo=5. Los objetos son:



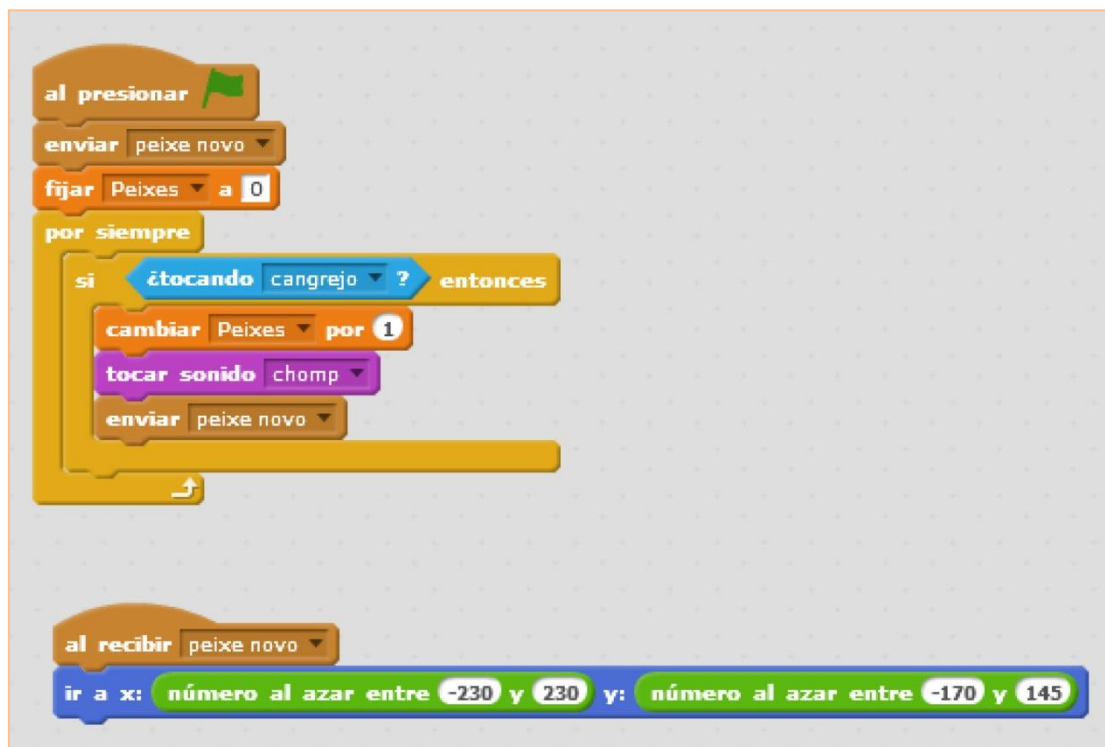
*Objetos del juego.* Susana Oubiña Falcón ([CC BY](#))

PEZ: El objeto “pez” es un gif que dispone de 12 disfraces. Para aprovechar sus disfraces hago que éstos cambien de forma continua (cada 0.1 segundos) utilizando un bucle “Por siempre”. También busco que el pez aparezca en un lugar aleatorio (X,Y) del escenario. Esto se consigue con el siguiente script:



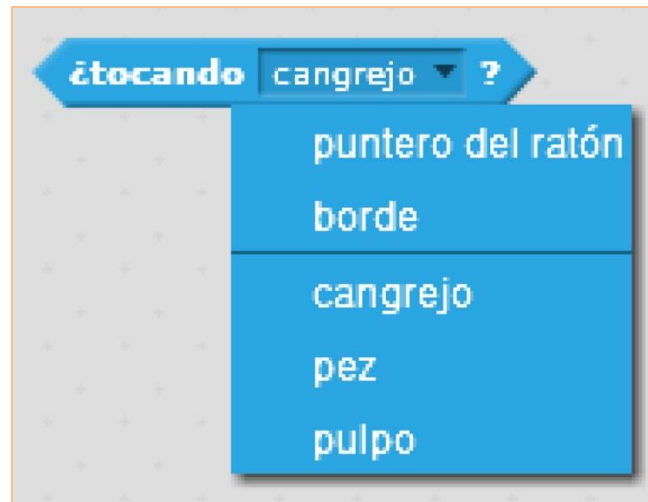
*Cambio continuo de disfraz.* Susana Oubiña Falcón ([CC BY](#))

Para el Pez, creamos un mensaje “Peixe novo” que lo que va hacer será mostrar el objeto “pez” en una parte aleatoria del escenario. Al iniciar el programa, ya se envía ese mensaje. También se diseña un bucle Por siempre de modo que, siempre que el pez toque al objeto cangrejo, SUME un pez (añadir un punto en el juego), toque un sonido y lo vuelva a mostrar en otro lugar aleatorio del escenario. (Simulará que desaparece y que aparece otro pez). El script para realizar lo explicado es el siguiente:



*Aumentar puntos.* Susana Oubiña Falcón ([CC BY](#))

En el script anterior vemos un nuevo comando en azul. La programación de Tocar al cangrejo se realiza por medio del siguiente comando del bloque sensores:



Comando “¿tocando...?”. Susana Oubiña Falcón ([CC BY](#))

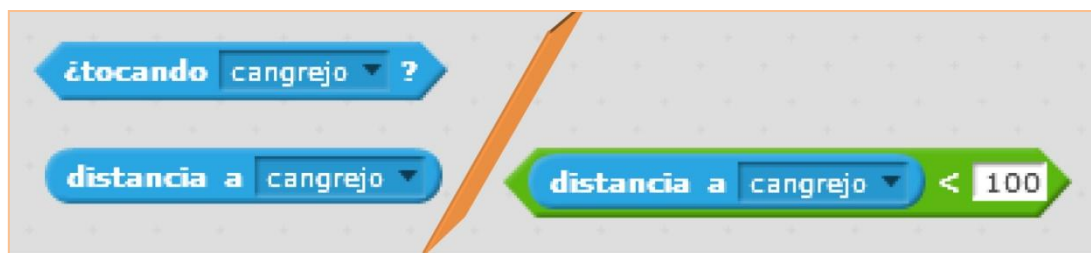
Scratch ya implementa la posibilidad de que el objeto toque el puntero del ratón, el borde del escenario u otro objeto del programa.

PULPO: Al comenzar el programa, se esconde y se crea un clon del objeto “pulpo”.



Clon del objeto pulpo. Susana Oubiña Falcón ([CC BY](#))

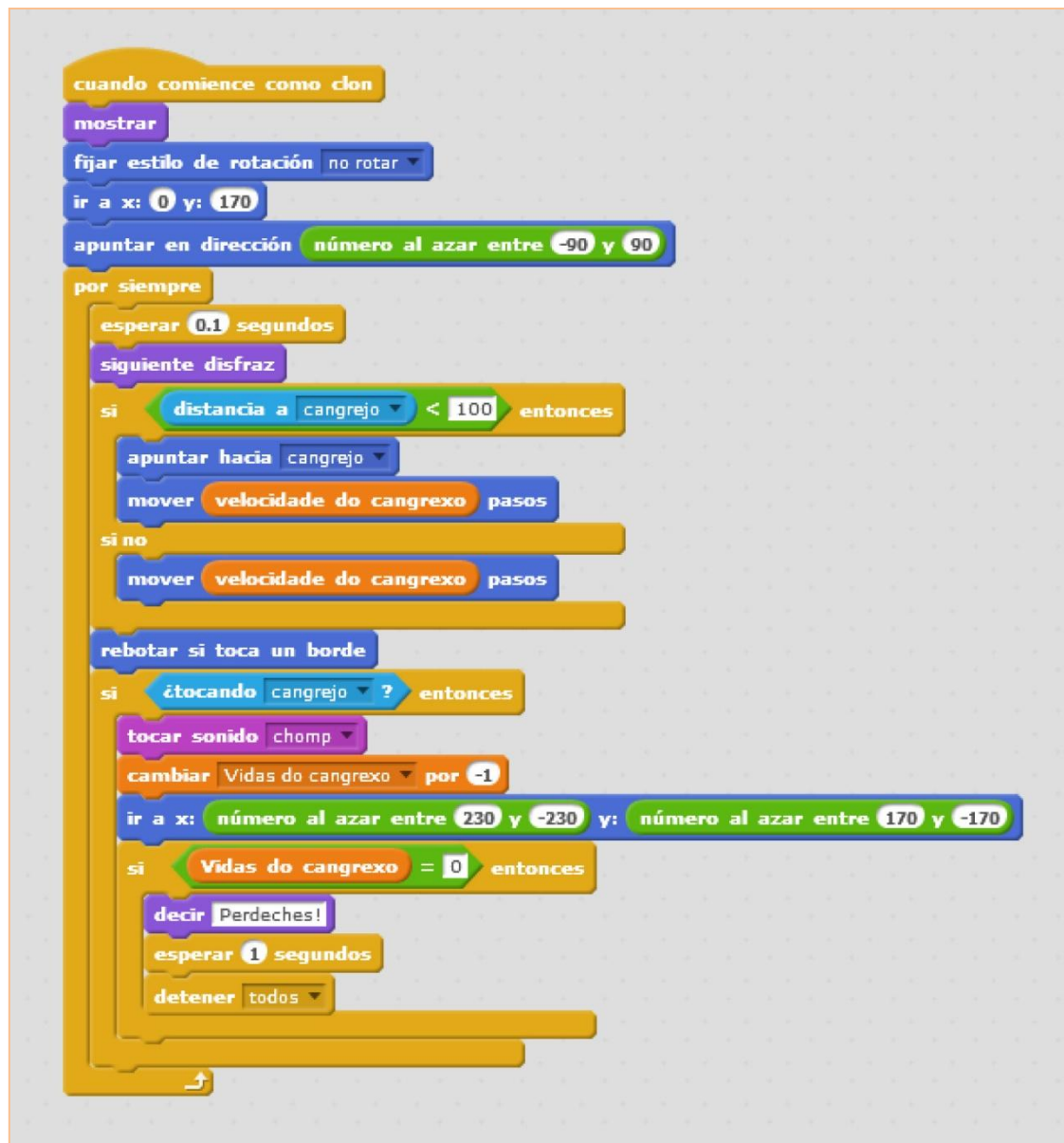
En el clon programaremos su movimiento y funcionalidad. El clon del pulpo se mostrará en el punto (0,170) apuntando en una dirección al azar entre -90 y 90. Fijamos su estilo de rotación en “no rotar”. Introducimos su movimiento dentro de un bucle por siempre programando que debe hacer si se encuentra a menos de 100 pixeles del cangrejo (o lo contrario) y que debe hacer si toca al cangrejo. Utilizamos los siguientes comandos:



*Comandos de Sensores.* Susana Oubiña Falcón ([CC BY](#))

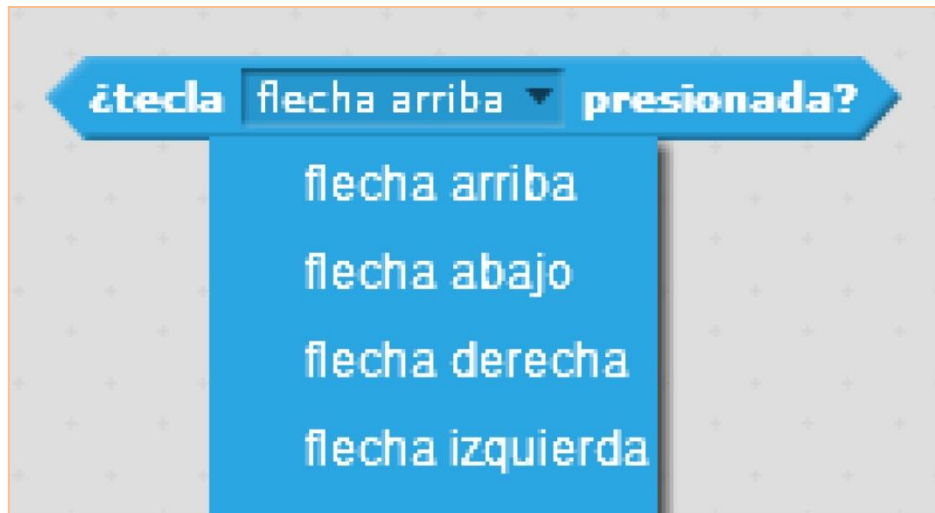
Cuando se encuentre una distancia menor que 100 pixeles del cangrejo, queremos que el pulpo apunte hacia él (modificando la dirección aleatoria que había cogido al inicio) con la velocidad dada al principio (5 pasos), pero, si se encuentra más alejado, simplemente se moverá 5 pasos en la dirección aleatoria que había cogido. También queremos que en ambas rebote si toca un borde. Cuando toque al cangrejo, el cangrejo pierde una vida (restamos -1 vida), suena un sonido y hacemos que el pulpo se vaya a una posición (x,y) aleatoria del escenario. Dentro de este condicional debemos programar la posibilidad de que se quede sin vidas. Por ello se programa que en el caso de que tenga 0 vidas, diga “Perdechess!!!” y detenga el programa.





*Cuando comience como clon.* Susana Oubiña Falcón ([CC BY](#))

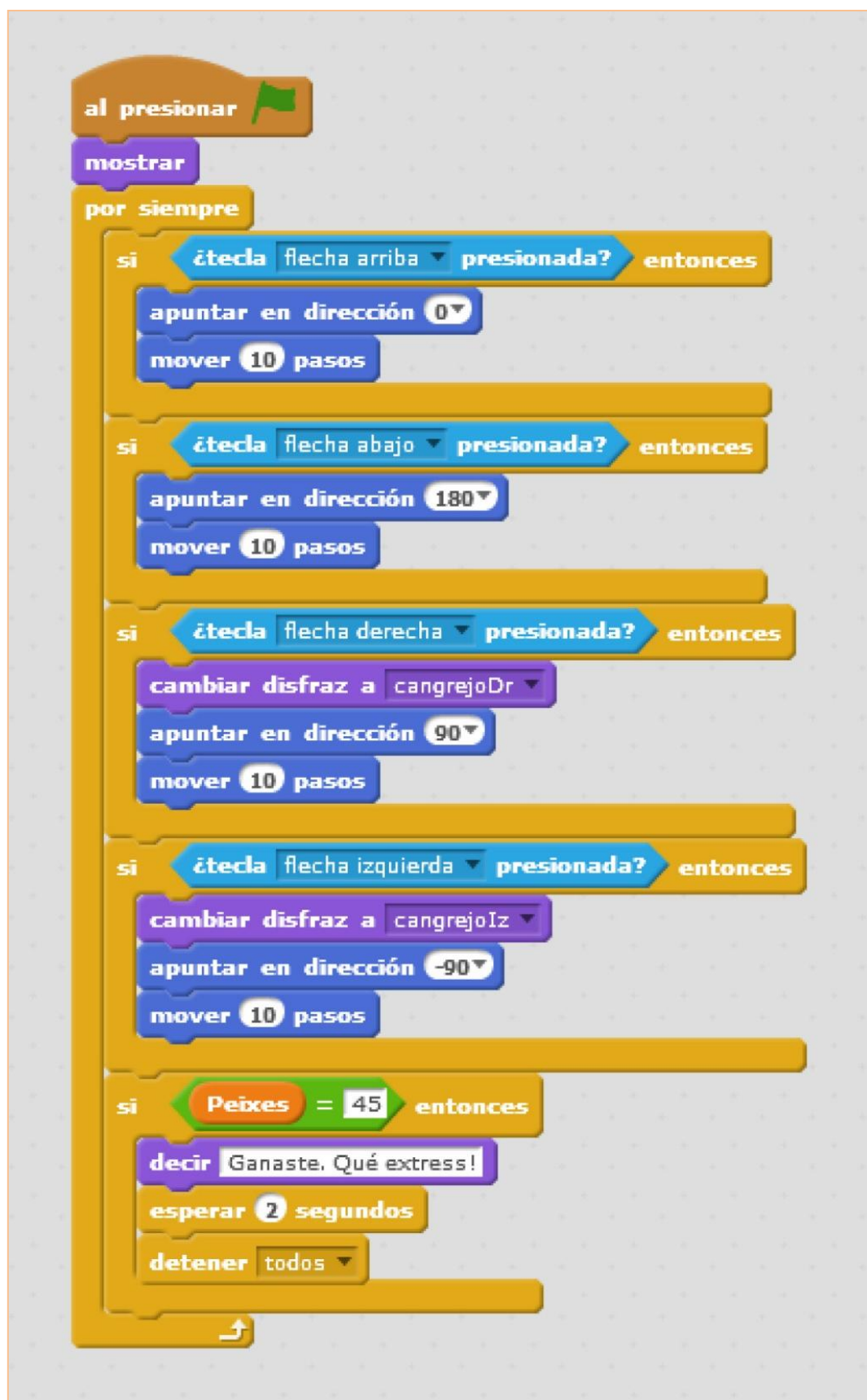
CANGREJO: Los comandos del bloque sensores que usamos para programar el movimiento del cangrejo será el comando repetido ¿tecla...presionada? El cangrejo se moverá con las teclas de movimiento (flechas) arriba, abajo, derecha e izquierda.



*¿Tecla...presionada?*. Susana Oubiña Falcón ([CC BY](#))

Cada diferente tecla presionada en el teclado se programa con su correspondiente condicional, apuntando a su dirección correspondiente y moviéndose 10 pasos.

El cangrejo debe ir comiendo pescados y aumentando puntos, pero hay un límite. Queremos que cuando consiga comer 45, el programa finalice. El script del cangrejo es el siguiente:



Script del objeto "cangrejo". Susana Oubiña Falcón ([CC BY](#))

13. Ejemplo: el siguiente programa ejemplifica un script sencillo en el que se usan 2 comandos del bloque sensores que detectan colores. Comandos que suelen ser muy útiles en el scratch. Para verlos utilizamos dos objetos, un círculo amarillo y el cangrejo del programa anterior.



*Objetos del programa.* Susana Oubiña Falcón ([CC BY](#))

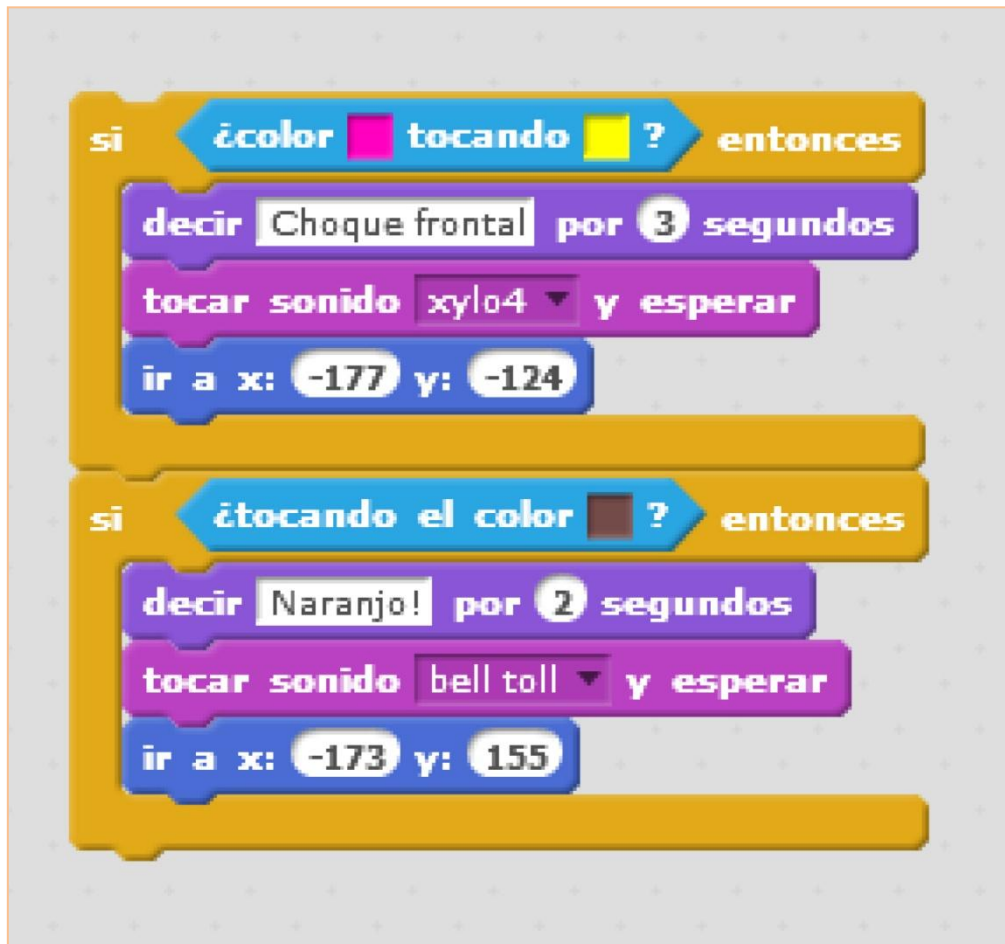
El objeto cangrejo lo hemos modificado. En él hemos pintado un rectángulo lila en una de sus patas.



*Objeto "cangrejo" modificado con color.* Susana Oubiña Falcón ([CC BY](#))

Utilizamos dos sensores diferentes de color: ¿tocando el color...? Y ¿color...tocando...?, de modo que, si el color lila de la pata del cangrejo toca la

bola amarilla, mostrará el texto “Choque frontal!”, tocará un sonido y se irá a una posición; pero, si el objeto cangrejo toca el color “naranja oscuro”, mostrará el texto “Naranja!”, tocará otro sonido e irá a otra posición diferente a la anterior. La programación de estos sensores de color es la siguiente:



*Sensores de color en el cangrejo.* Susana Oubiña Falcón ([CC BY](#))

El cangrejo se mueve al presionar las flechas de dirección del teclado. El script completo para este objeto es:







Script completo del programa 13. Susana Oubiña Falcón ([CC BY](#))



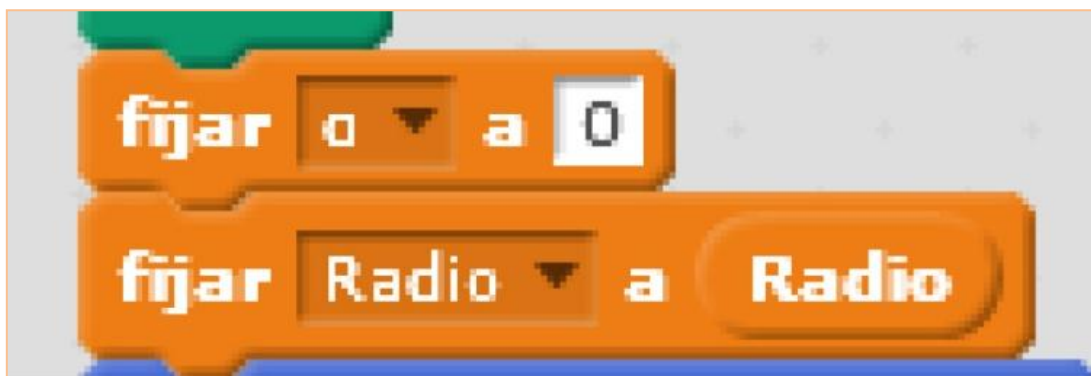
### ***Ejemplos con comandos del bloque Lápiz***

14. Ejemplo: En el siguiente programa vamos a ver como el scratch puede dibujar figuras y objetos. Para dibujar, debemos utilizar los comandos del bloque Lápiz. Algunos de ellos se muestran en la siguiente tabla:

	Borra todas las marcas de lápiz y de sellos (estampados) del Escenario.
	Baja el lápiz del Objeto, de manera que <b>este pinte</b> a medida que se mueve.
	Levanta el lápiz del Objeto, de manera que <b>no pinte</b> cuando se mueva.
	Estampa o copia la imagen del Objeto en el Escenario.

*Comandos del bloque Lápiz.* Susana Oubiña Falcón ([CC BY](#))

Vamos a crear un programa que dibuje una circunferencia a partir del movimiento de un punto por ella. Vamos a dar la opción de modificar su radio y así será más funcional. Por lo tanto, el ángulo se fijará al valor inicial 0, pero el Radio, se fijará al valor que deseemos dar en ese momento:



*Variables inicializadas.* Susana Oubiña Falcón ([CC BY](#))

Como sabemos, todo punto P de una circunferencia de radio R atiende a las siguientes coordenadas:

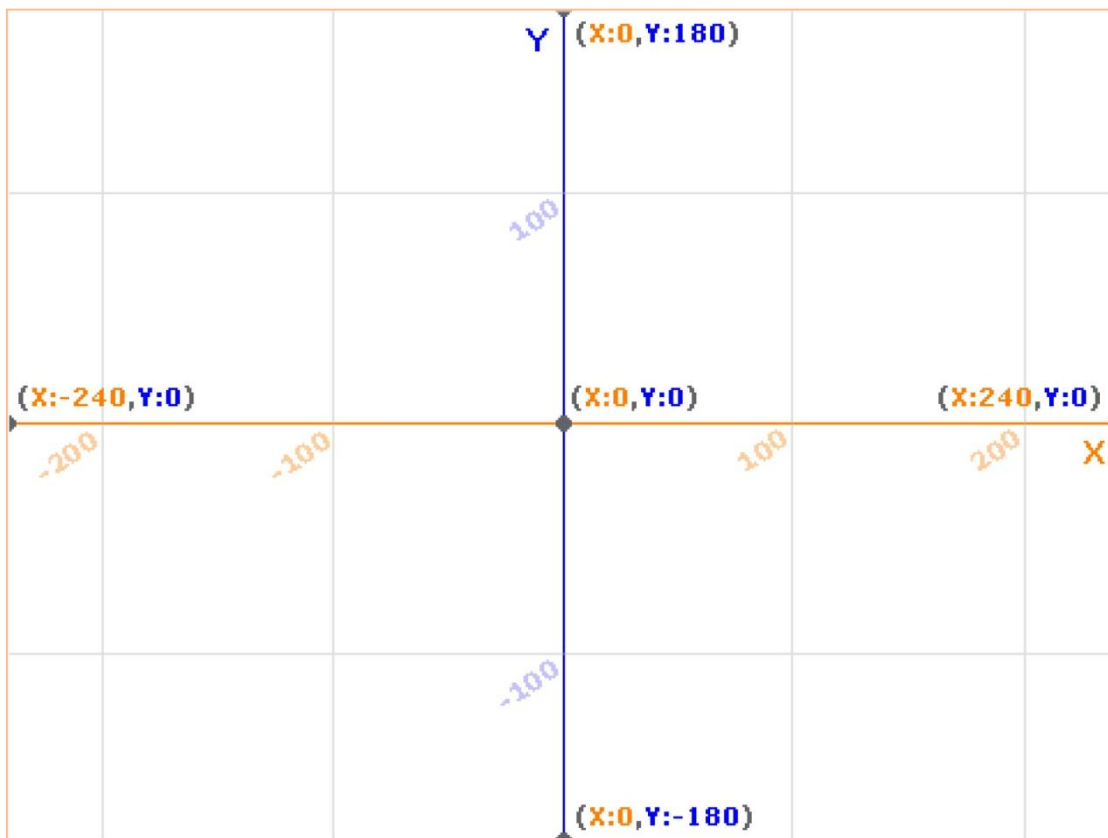
$$X_{punto} = R \cdot \cos\alpha$$

$$Y_{punto} = R \cdot \sen\alpha$$

*Coordenadas de un punto P de una circunferencia.* Susana Oubiña Falcón ([CC BY](#))

Comenzamos creando las variables. Éstas serán: Xpunto, Ypunto, Radio y  $\alpha$  (ángulo que forma el vector con el eje OX).

Como escenario, utilizamos uno apropiado para representaciones “xy-grid”, disponible en la biblioteca de scratch:



*Escenario “xy-grid”.* Susana Oubiña Falcón ([CC BY](#))

Creamos un objeto “punto” (lo dibujamos) y es el que debemos programar. Como quiero que la circunferencia esté centrada, su posición inicial será el (0,0), tal y como se muestra en la siguiente figura:



Origen del punto P. Susana Oubiña Falcón ([CC BY](#))

Al pulsar la bandera verde, siempre debe borrarse el escenario. Después de inicializar las variables, le damos al lápiz las características con las que queremos que pinte (tamaño, color e intensidad). También le decimos la dirección que debe tomar (queremos que apunte hacia arriba y por eso hemos puesto la dirección 0). Estas instrucciones se observan en el siguiente script:



Script “antes de pintar”. Susana Oubiña Falcón ([CC BY](#))

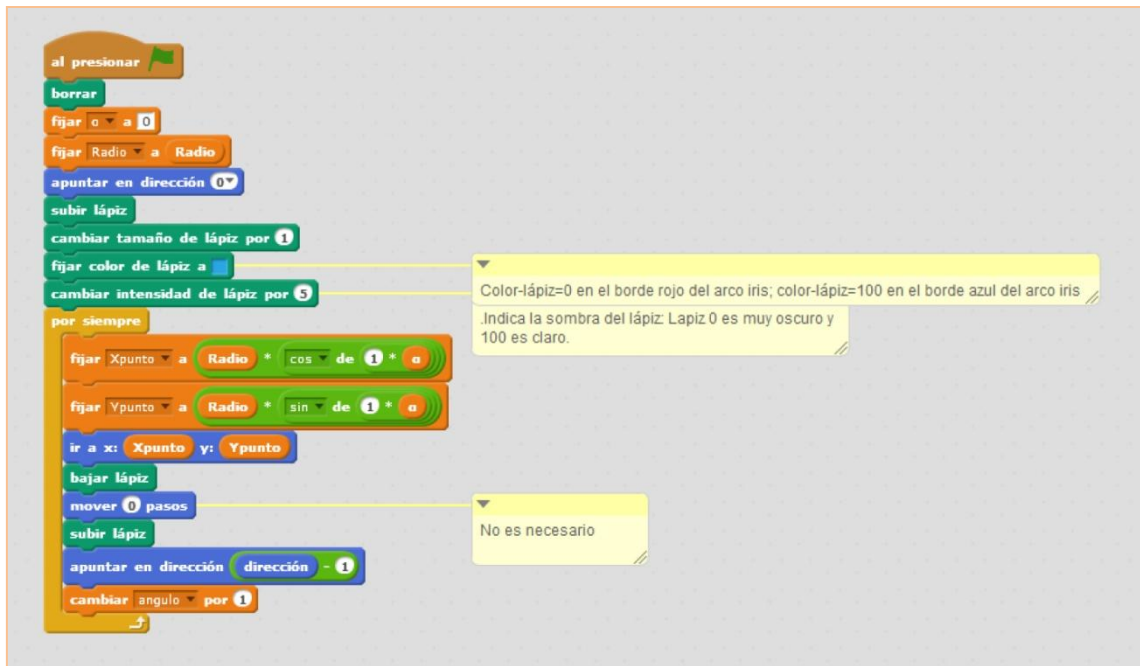
Ahora damos paso a la programación de la circunferencia. Para ello usamos un bucle Por siempre y, en él, fijamos las variables Xpunto e Ypunto que

habíamos definido al inicio de este ejemplo, le decimos que se desplace a esas variables y que allí baje el lápiz (que pinte), que lo suba (deje de pintar) y que disminuya en uno la dirección hacia la que apunta aumentando en 1 el ángulo con el eje x.



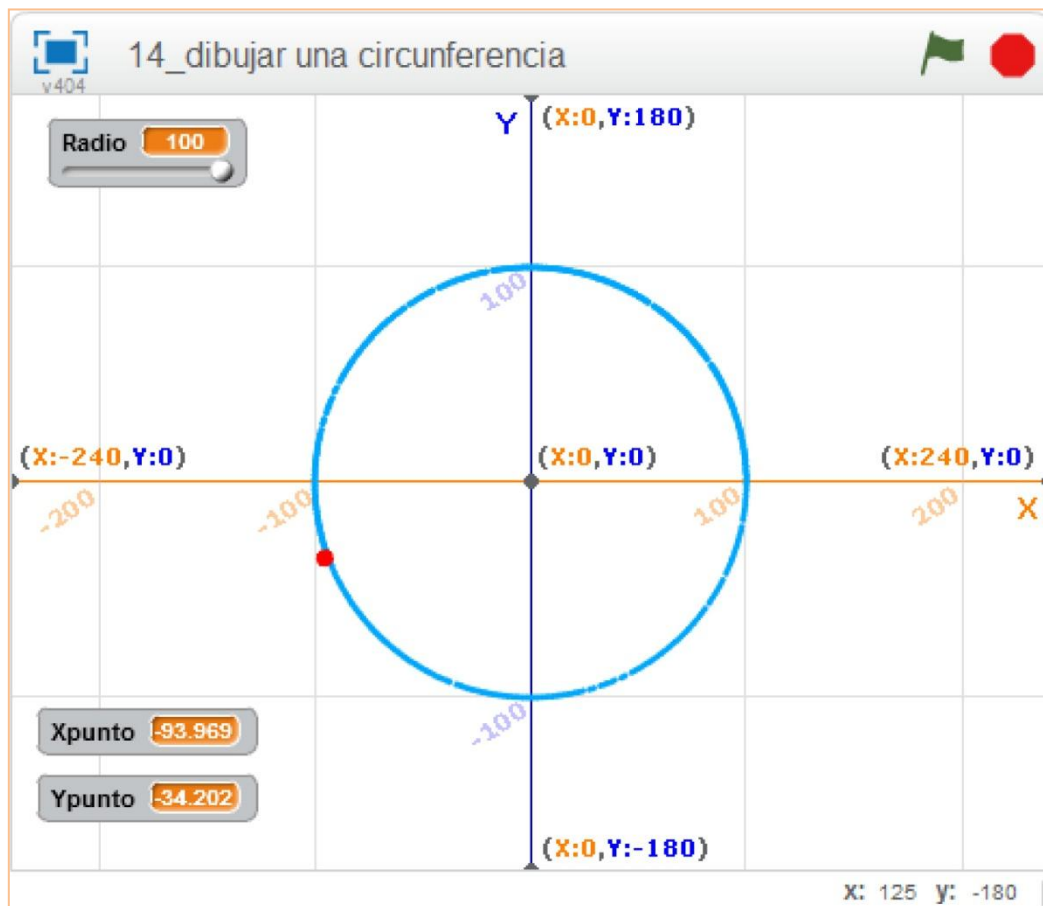
*Bucle para pintar la circunferencia. Susana Oubiña Falcón ([CC BY](#))*

Así pues, la unión de los dos scripts anteriores otorgan al objeto “punto” la funcionalidad de dibujar una circunferencia:



Representación de una circunferencia. Susana Oubiña Falcón ([CC BY](#))

Para un R=100 píxeles, vemos la siguiente imagen en el escenario:

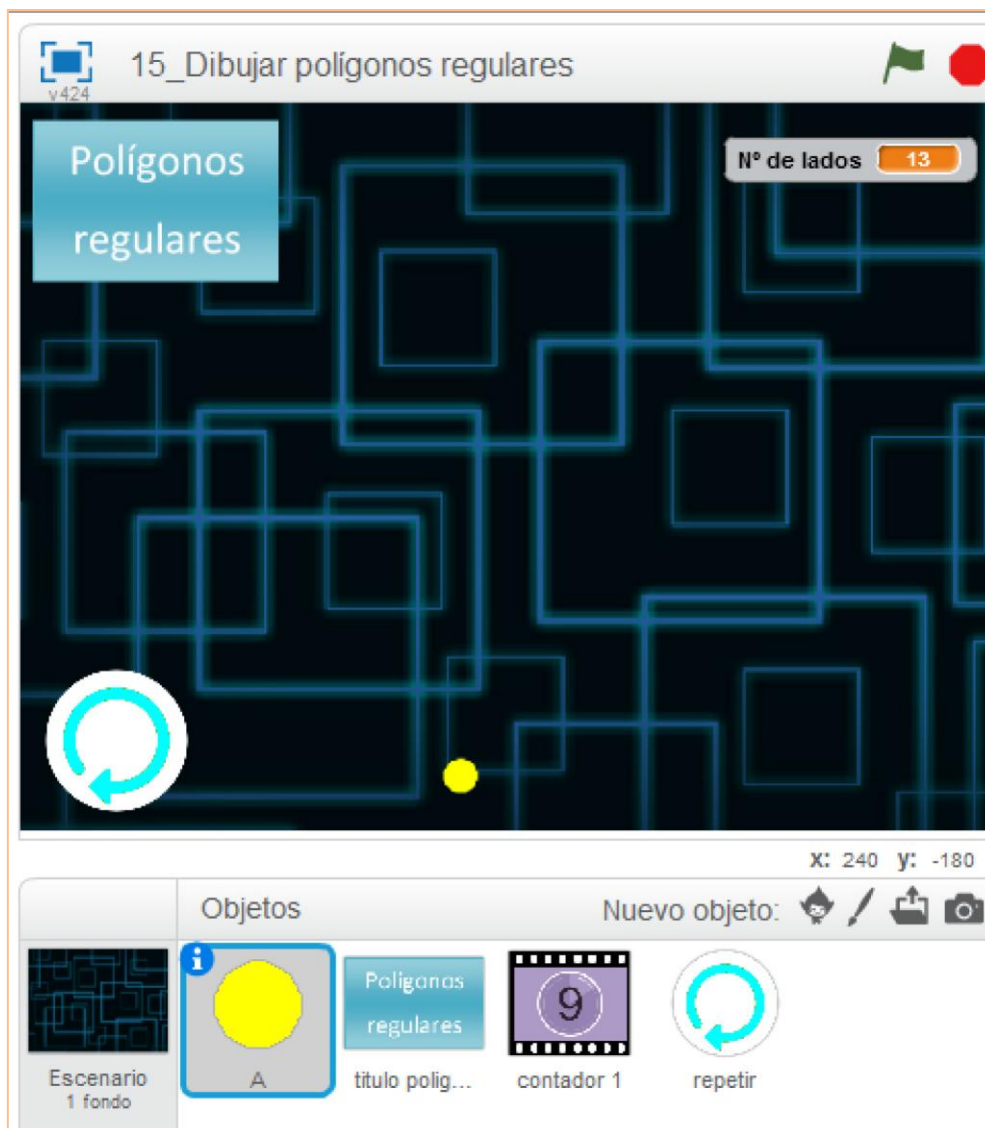


Programa corriendo en el escenario. Susana Oubiña Falcón ([CC BY](#))

En esta imagen se observa que la variable Radio se muestra en el formato “deslizador”. Moviendo el punto del deslizador, elegimos el radio de la nueva circunferencia.

15. Ejemplo: En el siguiente ejemplo quiero crear un programa que dibuje polígonos regulares, de forma continua. Para que me entren en el escenario del scratch y se vean de forma razonablemente bien, he decidido dar una longitud al lado del polígono regular de 75 píxeles y sólo se representará hasta un polígono de 13 lados (el de 14 lados, para esa medida del lado, saldría fuera del escenario y no se vería completo). También he querido crear un efecto de color para cada polígono representado así como, para los extremos de cada segmento.

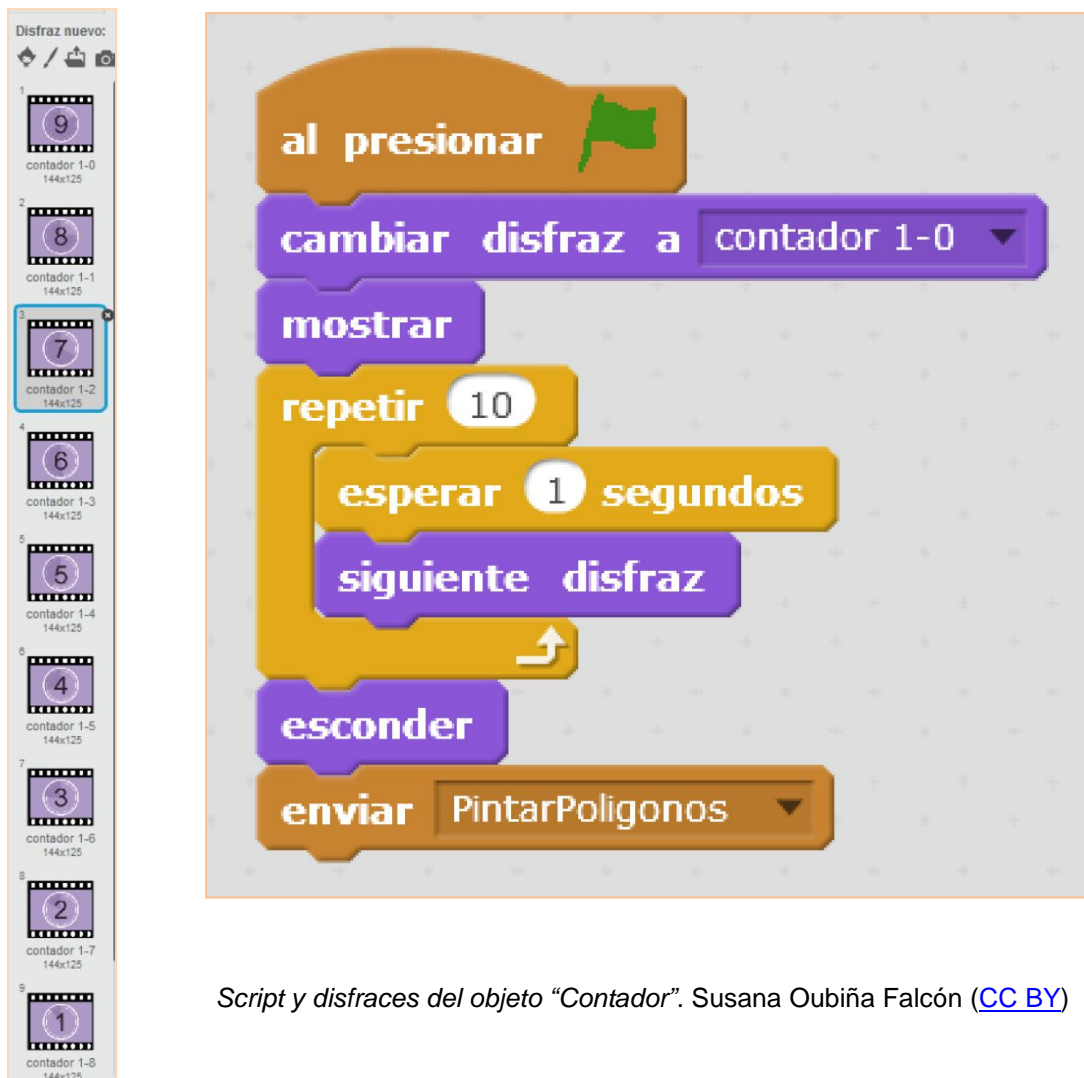
Vamos a utilizar 4 objetos: el punto “A”, “título polígono”, “contador” y “repetir”:



*Escenario y objetos del programa 15. Susana Oubiña Falcón ([CC BY](#))*



El objeto “Contador” es un gif animado 10 disfraces: mostrando los números del 9 al cero. Este se activa al inicio del programa, cambiando de disfraz cada segundo. Por lo tanto, es un reloj real. Su script y disfraces se muestran en la siguiente imagen:



*Script y disfraces del objeto “Contador”. Susana Oubiña Falcón ([CC BY](#))*

Puede observarse en el script que, después de que el contador pase por todos sus disfraces, envía el mensaje “PintarPoligonos”. Este mensaje lo recibe el objeto “A” que es el que realmente dibuja los diferentes polígonos en el escenario. No he querido crear un script para 3 lados, otro para 4, para 5, etc. Lo que busqué es crear un script genérico para todos los polígonos. Para ello, necesito crear 3 variables: Longitud (que la fijo a 75 píxeles), N° de lados (que la inicializo en 3 ya que comenzará representando un triángulo) y el ángulo que debe girar después de moverse 75 píxeles en la dirección del eje x positivo desde una posición inicial de (-25, -150). He situado ahí el punto inicial A para que me cojan todos los polígonos en el escenario.

Partiendo siempre de una dirección hacia el eje X positivo, se deduce que, matemáticamente, el ángulo que debe girar es siempre:  $360 / N^{\circ}$  de lados.

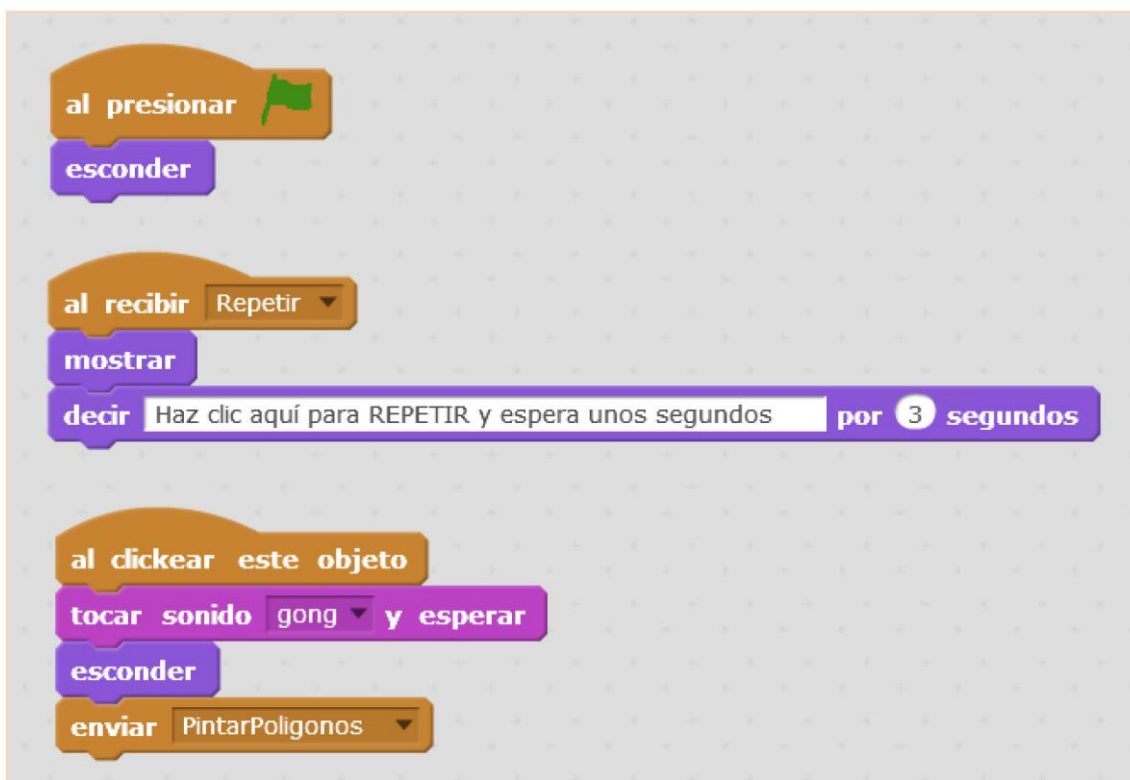
He utilizado un bucle de repetición que se ejecutará desde 3 hasta que el Nº de lados sea 13, cubriendo mis deseos. El script para el objeto “A” es el siguiente:



Script para el objeto “A”. Susana Oubiña Falcón (CC BY)

En la figura anterior, se observa el comando “sellar”. Lo he utilizado para que clone, es decir, represente, el punto inicial A en cada extremo que representa.

El script del objeto “A” finaliza enviando el mensaje “Repetir”. Este mensaje lo recibe el objeto “repetir” y lo que hace es dar opción a ver de nuevo el desarrollo de la representación gráfica de los diversos polígonos; informando de lo que se debe hacer para conseguirlo (hacer clic en el objeto). Obviamente, debe volver a enviar el mensaje “PintarPolígonos”. Su script es el siguiente:



*Script para el objeto “Repetir”. Susana Oubiña Falcón ([CC BY](#))*

### ***Ejemplos con crear “Más bloques”***

Scratch 2.0 permite crear bloque nuevos y con ellos, crear pequeños programas con diversas funcionalidades y que sean llamados por diferentes objetos del proyecto en cualquier momento, de forma recursiva.

16. Ejemplo: Imaginemos que queremos crear un programa de modo que, un objeto (o varios), por ejemplo el objeto mOway; realice siempre el mismo movimiento que consistirá en moverse por la 4 esquinas del escenario. Con lo que ya sabemos, crearíamos el siguiente script para cada objeto (o para ese objeto):



Script para “movimiento”. Susana Oubiña Falcón ([CC BY](#))

Esto puede hacerse creando un bloque que podemos llamar “movimiento”, dentro de la categoría “Más Bloques” del scratch.



*Opción “Más Bloques”.* Susana Oubiña Falcón ([CC BY](#))

Haciendo clic en “Crear un bloque”, se nos abre una ventana en la que escribimos el nombre de nuestro bloque. En nuestro caso, “Movimiento”:



*Bloque “Movimiento”.* Susana Oubiña Falcón ([CC BY](#))

Sólo falta escribir el pequeño script del bloque nuevo. Es decir, escribir que hace el bloque que hemos llamado “Movimiento”. Queríamos que el objeto se moviera por las 4 esquinas del escenario y que ese movimiento lo hiciera siempre, de forma continua. Por lo tanto, el script de “Movimiento” ha de ser el siguiente:



Script “Movimiento”. Susana Oubiña Falcón ([CC BY](#))

Para que cualquier objeto realice la operación que hemos definido como “Movimiento”, debemos ordenárselo. Una forma es la siguiente:

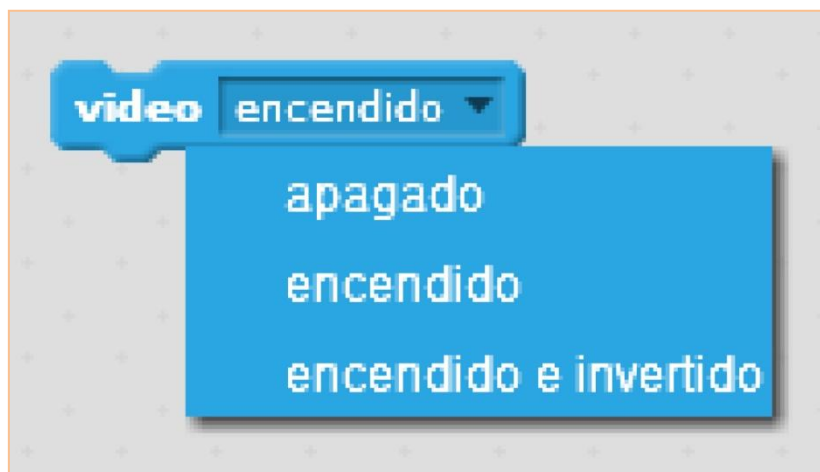




Ordenar "Movimiento". Susana Oubiña Falcón ([CC BY](#))

### ***Un nuevo elemento "El vídeo" (Realidad Aumentada)***

La grabación de vídeo entendido como forma de introducir imágenes a tiempo real dentro del scratch se ha implementado en la nueva versión del scratch (scratch 2.0) dentro del bloque "Sensores" con el comando "vídeo..." con sus opciones encendido, apagado y encendido e invertido.



Comando "vídeo...". Susana Oubiña Falcón ([CC BY](#))

**NOTA:** Con el scratch 1.4 esto no era posible, aunque si es cierto que existe una versión modificada del scratch (llamada Scratch Spot) que es capaz de recoger esas imágenes y lograr crear realidad aumentada con 3 tipos de marcadores. La Realidad Aumentada es una tecnología que añade una parte de software a un objeto físico. Para poder implementarse en el Spot, necesita de unos marcadores. Esos marcadores se asocian a objetos del scratch, de modo que, moviendo el marcador físico en un entorno externo (que se visiona

en el escenario gracias a la cámara web del ordenador) se mueve el objeto por el escenario.

Cuando se enciende el vídeo, la webcam de nuestro ordenador recoge la imagen y esta se ve en el escenario del scratch 2.0. Esta imagen puede mostrarse de forma más o menos transparente en el escenario del scratch. Para ello debemos usar el comando “fijar transparencia de vídeo a ...%”, siendo los valores extremos 0% (visionado nítido y claro de la imagen que rescata la webcam) y 100% (escenario de color blanco, no se visiona la imagen que rescata la webcam):



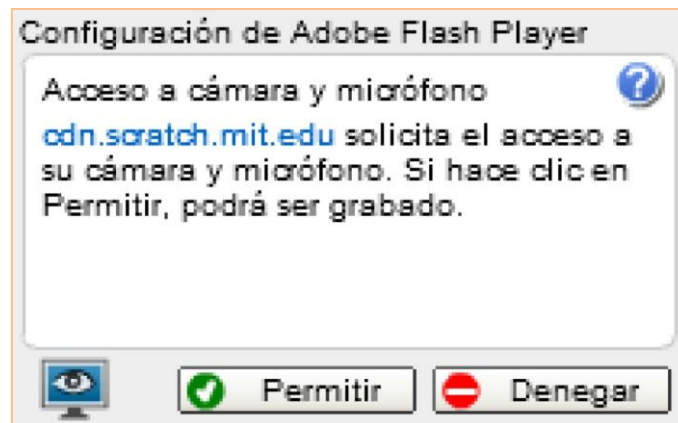
Comando “fijar transparencia de vídeo a ...%”. Susana Oubiña Falcón ([CC BY](#))

A mayores, scratch 2.0 da opción no sólo de visionar la imagen de la webcam sino también de interaccionar con ella. El comando que hace posible esta conexión y que implementa la Realidad Aumentada pertenece al bloque “Control” y se muestra en la siguiente imagen:



Comando “cuando el...sea >...” Susana Oubiña Falcón ([CC BY](#))

17. Ejemplo: “Estoupando Burbullas”. Este ejemplo podría ser un juego, y digo podría porque nunca se pierde y suena raro eso de jugar a algo donde nunca pierdes. Utiliza los comandos vídeo y fijar transparencia del bloque sensores y el comando de control del movimiento del vídeo del bloque control. Después de hacer clic en la bandera verde debemos dar permiso a que nuestra webcam recoja la imagen.



*Pantalla de “Permiso” a la webcam. Susana Oubiña Falcón ([CC BY](#))*

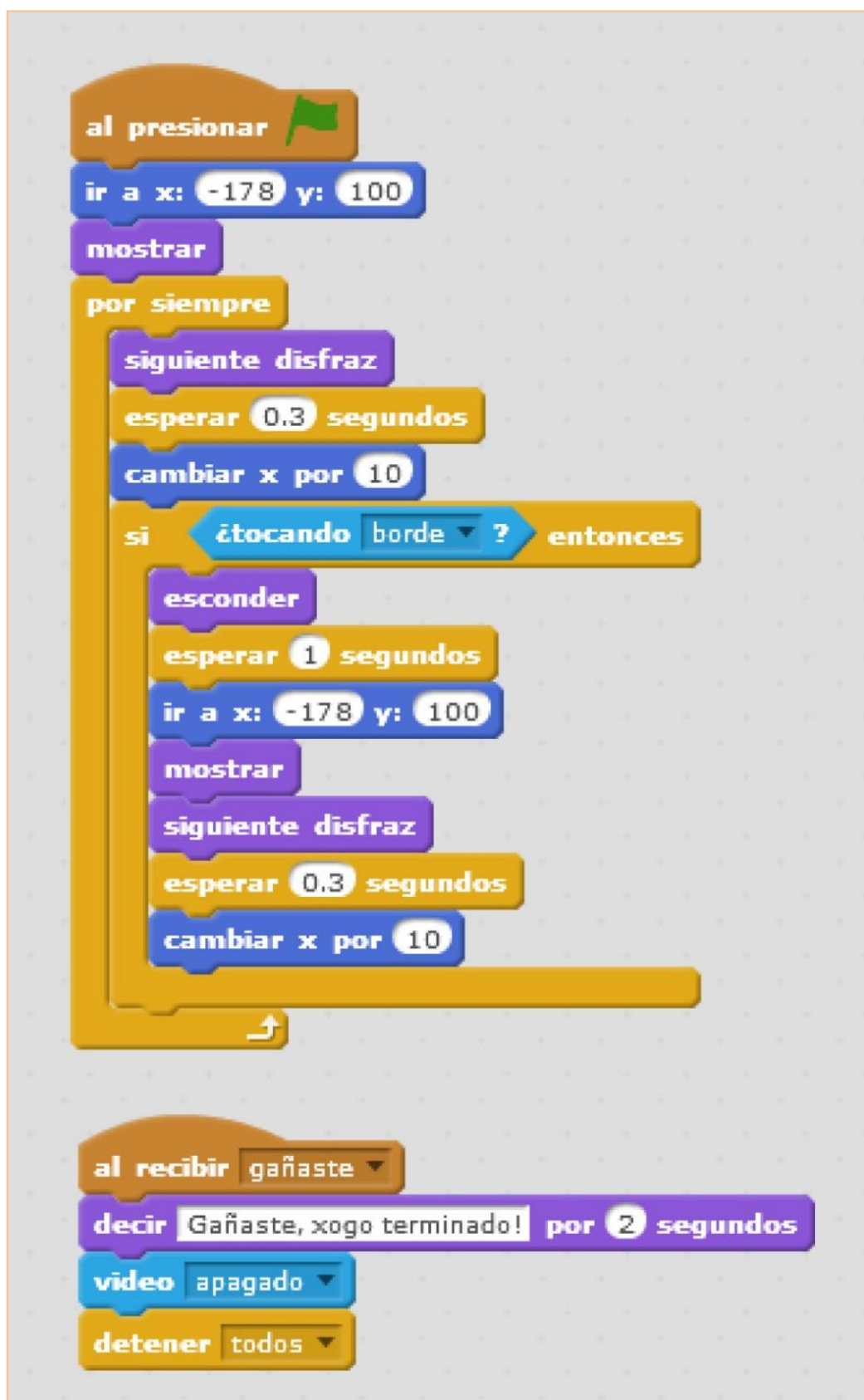
En el juego, el jugador (en este caso yo) interactúo con el objeto del scratch “burbuja de agua”, como puede verse en la siguiente imagen al 0% de transparencia:



*Imagen con “fijar transparencia de vídeo a 0%”. Susana Oubiña Falcón ([CC BY](#))*

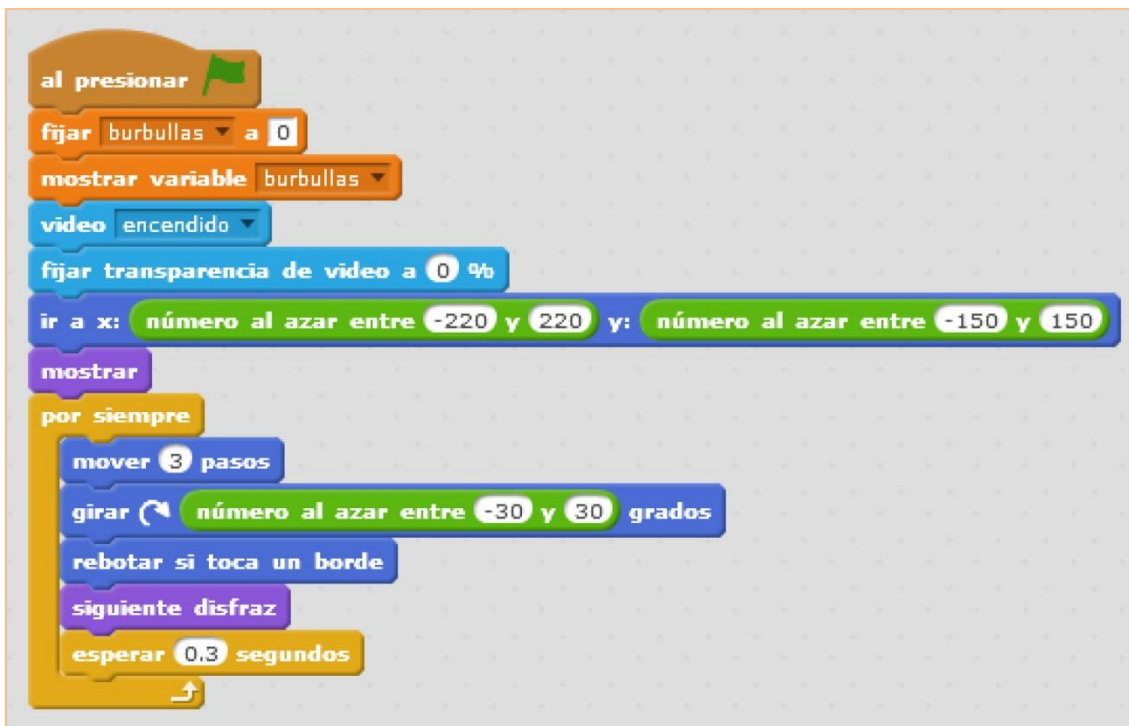
El programa presenta 2 objetos: Burbuja de agua y Pez Burbuja. El segundo se crea para dar sensación de juego en el proyecto. Su misión es muy sencilla: moverse por el escenario de forma horizontal hasta que toque el borde derecho y volver a aparecer en el lateral izquierdo efectuando el mismo movimiento. A mayores, lo aprovechamos para parar el juego y apagar el vídeo, cuando consigamos explotar 20 burbujas, ya que recibirá el mensaje “gañaste”.

El script para el objeto “Pez Burbuja” es el siguiente:



Script para el objeto “Pez Burbuja”. Susana Oubiña Falcón ([CC BY](#))

La “burbuja de agua” es el objeto interesante. Su script presenta dos partes. La primera parte comienza al hacer clic en la bandera verde: enciende el vídeo y fija la transparencia al 0% (se ve de forma nítida la imagen de la webcam), indica la puntuación de cero burbujas explotadas y genera el movimiento de la burbuja (rebotando si toca un borde y moviéndose lentamente girando al azar un grado entre -30 y 30):



*1ª parte del Script para el objeto “Burbuja de agua”.* Susana Oubiña Falcón ([CC BY](#))

La segunda parte del script para el elemento “Burbuja de agua” es la más interesante. Se entiende que, nosotros, con nuestra mano, intentaremos tocar el objeto “Burbuja de agua”. Cuando nuestra mano (parte real) se acerque al objeto (parte de software), el comando de control “cuando el movimiento del vídeo sea > número” detecta ese movimiento (acercamiento) y realiza una acción: tocará el sonido “pop” simulando que la burbuja explotó, suma 1 burbuja al contador de puntos y la esconderá. Como el juego finaliza cuando explotemos la cantidad de 20 burbujas, testamos si hemos o no superado 20 burbujas. Si no hemos superado 20 burbujas el juego continúa y debe aparecer otra vez (en un lugar aleatorio de escenario, para darle mayor movilidad); en caso contrario, hemos llegado a 20 y envía el mensaje “gañaste”, que apagará el vídeo y muestra el escenario que hemos implementado “Game Over” (juego finalizado). Este script puede verse en la siguiente imagen:



2ª parte del Script para el objeto “Burbuja de agua”. Susana Oubiña Falcón ([CC BY](#))



## 2.3. Funcionalidad de la web de scratch: localizar programas compartidos, subir y compartir un programa.

Cuando un usuario scratch entra en la web de scratch introduciendo su nombre de usuario y contraseña, la web adquiere la siguiente interfaz (fig1).

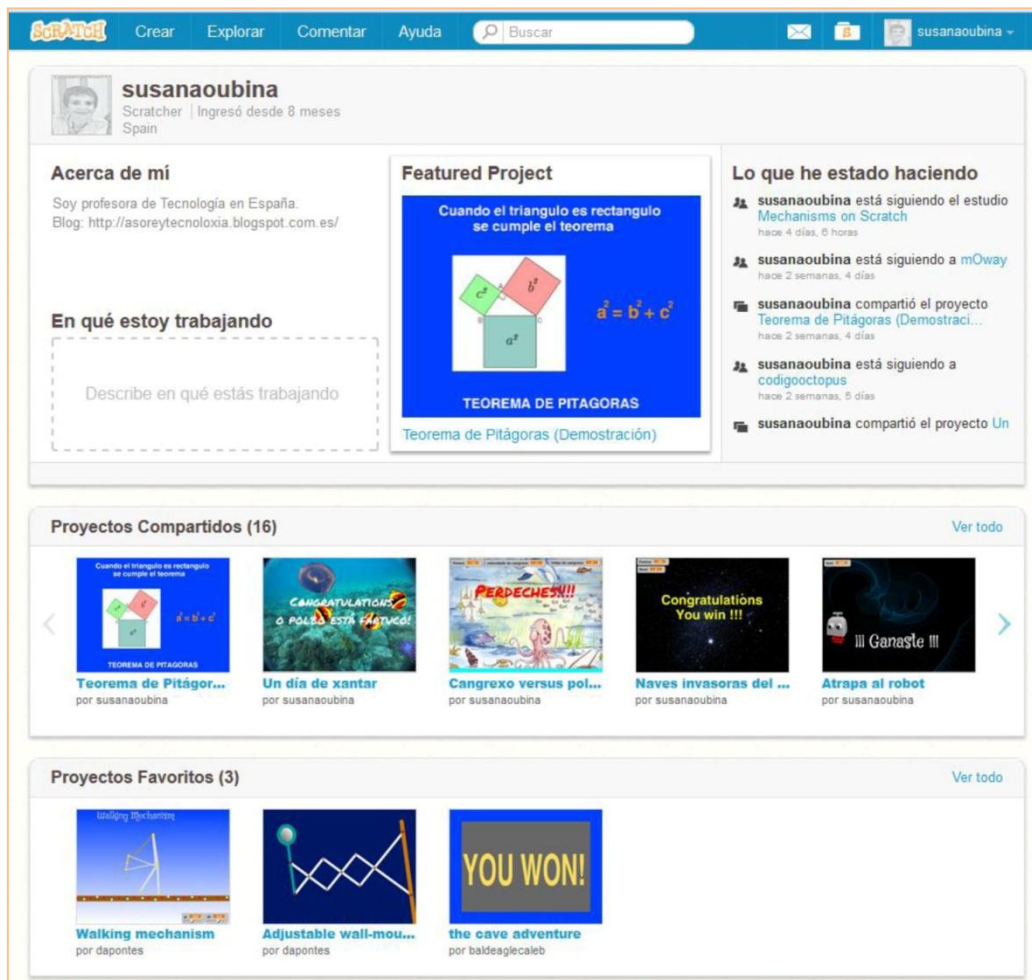


Fig1: Interfaz Perfil de la web de scratch. Susana Oubiña Falcón (CC BY)

En ella se observan las siguientes partes:

### 1. Menú superior principal



Menú superior. Susana Oubiña Falcón (CC BY)

### 2. Información más reciente del usuario


**susanaoubina**  
Scratcher | Ingresó desde 8 meses  
Spain

**Acerca de mí**  
Soy profesora de Tecnología en España.  
Blog: <http://asoreytecnologia.blogspot.com.es/>

**En qué estoy trabajando**  
Describe en qué estás trabajando

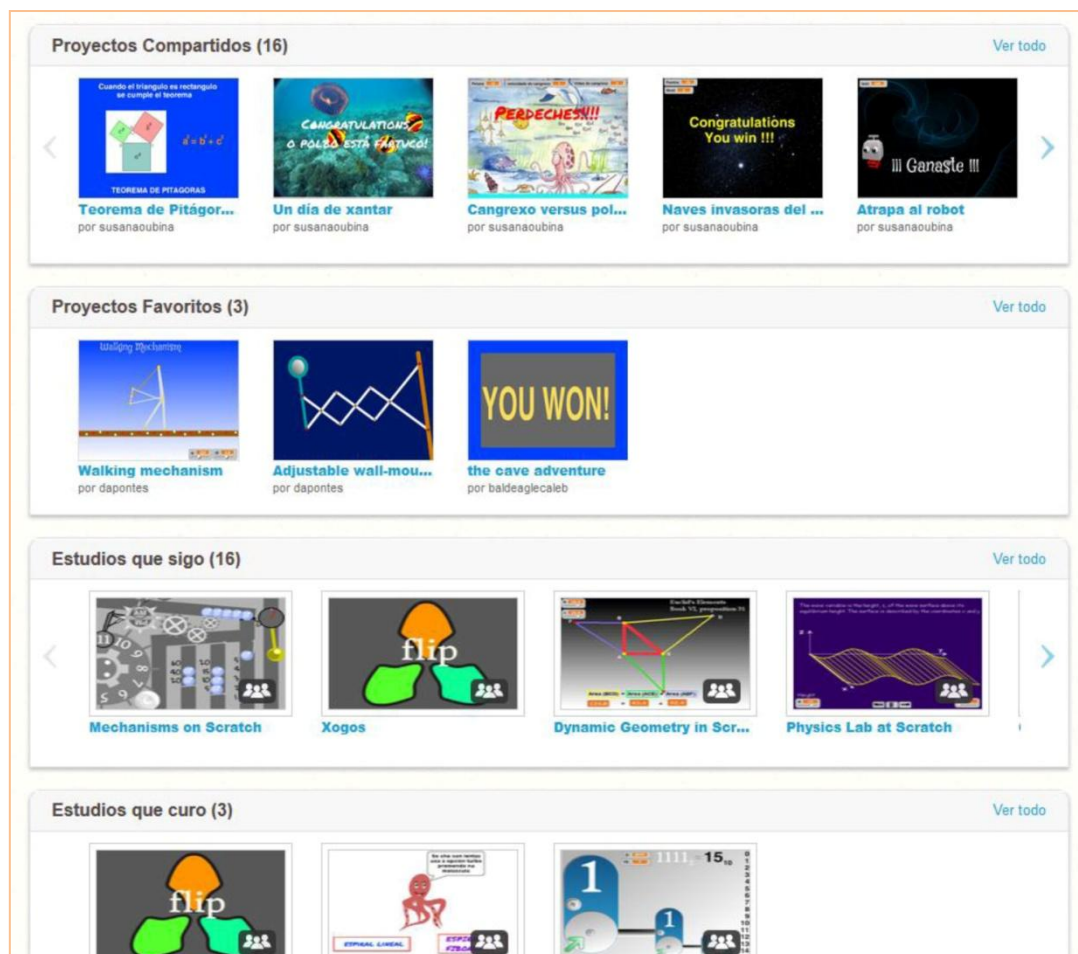
**Featured Project**  
  
**TEOREMA DE PITAGORAS**  
Teorema de Pitágoras (Demostración)

**Lo que he estado haciendo**

- susanaoubina está siguiendo el estudio [Mechanisms on Scratch](#)  
hace 4 días, 1 hora
- susanaoubina está siguiendo a [mOway](#)  
hace 2 semanas, 4 días
- susanaoubina compartió el proyecto [Teorema de Pitágoras \(Demostración\)](#)  
hace 2 semanas, 4 días
- susanaoubina está siguiendo a [codigooctopus](#)  
hace 2 semanas, 4 días
- susanaoubina compartió el proyecto [Un](#)

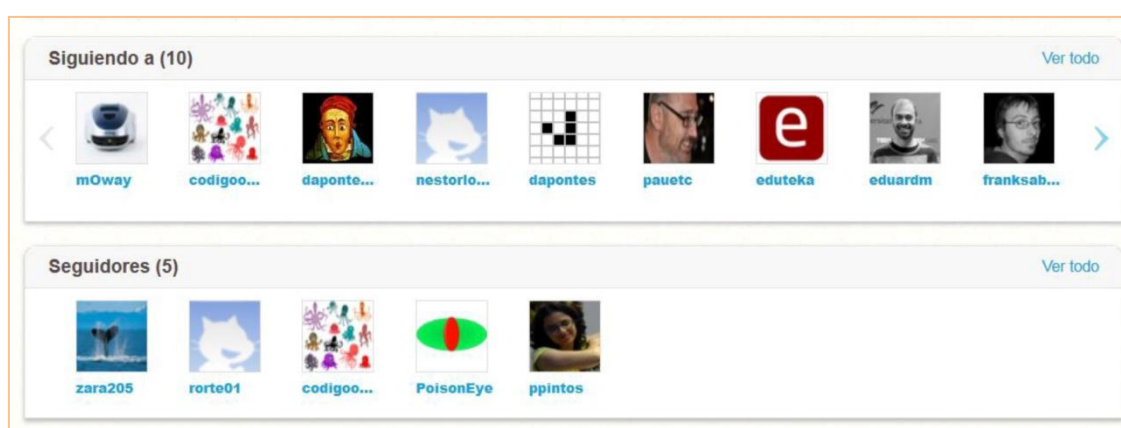
*Información Actual del usuario.* Susana Oubiña Falcón ([CC BY](#))

3. Información del usuario relacionada con: proyectos que ha compartido, proyectos que ha considerado favoritos, estudios que sigue (conjunto de programas para una temática en concreto), estudios que curo (un usuario puede tener el perfil “curador”. En ese caso se encargaría de comprobar que los proyectos de un determinado *Estudio* en el que es curador cumplen con los requisitos para pertenecer a ese estudio. La función de un usuario curador es revisarlos, incluir programas nuevos o incluso eliminar otros que no se adapten al Estudio en concreto).



Información sobre proyectos. Susana Oubiña Falcón ([CC BY](#))

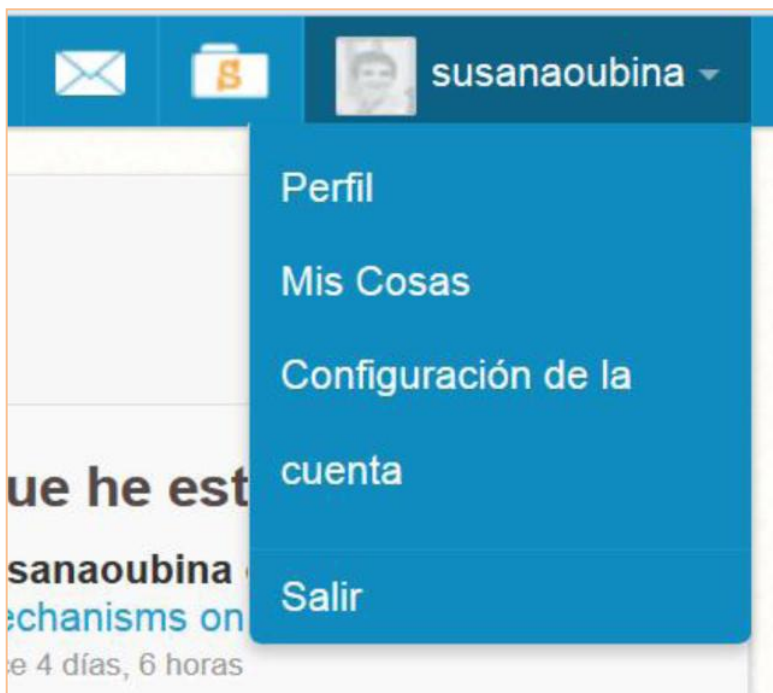
4. Información del usuario relacionada con usuarios scratch a los que sigue o usuarios que lo siguen a él.



Información sobre seguidores. Susana Oubiña Falcón ([CC BY](#))

Toda la información que se encuentra en los puntos 2 al 4 la va introduciendo la propia web de scratch cuando el usuario interactúa en la misma. Sin duda, la

parte más importante es el **menú superior principal**, y en él nos basaremos. De derecha a izquierda, el menú se compone de los siguientes iconos:



El icono Mensaje nos comunica los mensajes o alertas que tenemos sin leer de un usuario scratch que se comunica con nosotros.

El icono de carpeta con la S nos lleva a la pantalla de “**Mis Cosas**”. En ella veo mis proyectos, tanto los compartidos como aquellos en los que aún estoy trabajando (no compartidos), los

estudios que curo y proyectos que he borrado (basura).

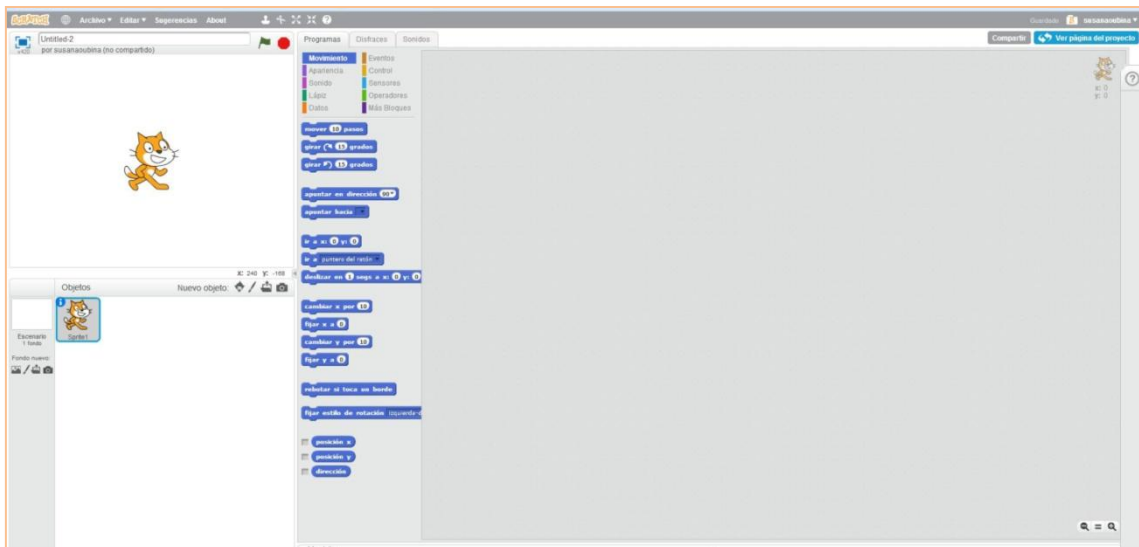
Por último, existe un menú desplegable que nos ofrece las siguientes opciones:

- ✓ Perfil: opción de configuración de nuestro perfil (fig1)
- ✓ Mis Cosas: Interfaz con todos nuestros proyectos separados en proyectos compartidos, no compartidos, estudios y basura
- ✓ Configuración de la cuenta: Para cambio de contraseña, correo electrónico o país.
- ✓ Salir: Para salir de la web de scratch

La parte superior izquierda del menú se muestra en la siguiente imagen. En ella se ofrecen las siguientes opciones:

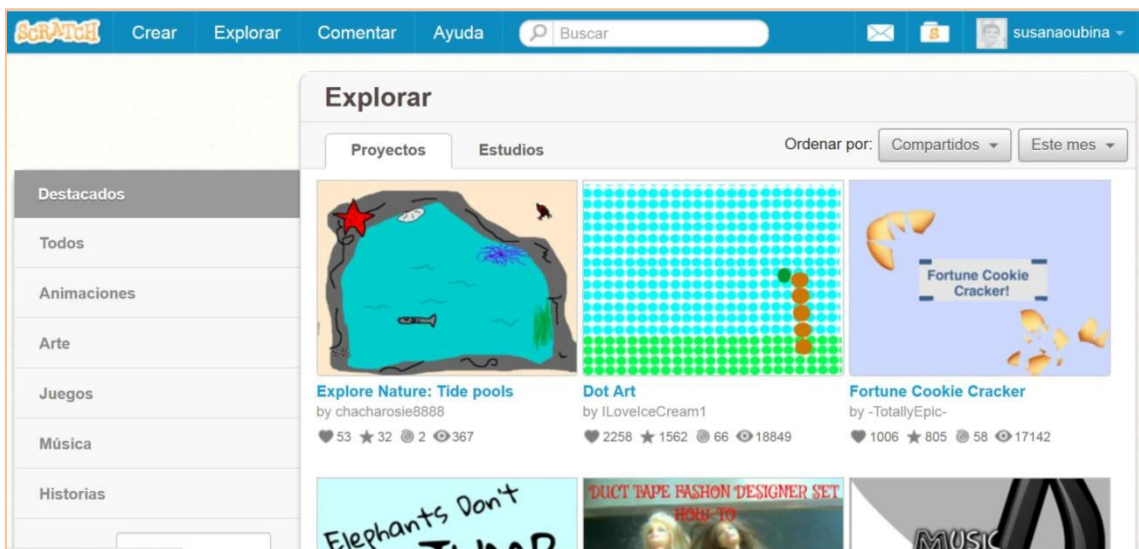


- ✓ Scratch: La pestaña **SCRATCH** nos lleva al enlace de inicio de la web: <http://scratch.mit.edu/>
- ✓ Crear: Esta opción es la más importante. Al pinchar en ella se nos abre la versión online del programa scratch 2.0. Esta opción es la que nos permite crear programas dentro de la web de scratch, trabajando online. El programa creado se puede compartir o no, según nuestros intereses. Su interfaz es la siguiente:



Interfaz de la pestaña Crear. Programa scratch2.0. Susana Oubiña Falcón (CC BY)

- ✓ Explorar: La pestaña Explorar nos ofrece una interfaz en donde localizar programas (proyectos) y estudios. Ambos tuvieron que ser compartidos. Su búsqueda podemos afinarla por mes y dentro de los grupos los “Más vistos”, “Más queridos” o “Más reinventados”.



Interfaz de la pestaña Explorar. Susana Oubiña Falcón (CC BY)

- ✓ Comentar: La pestaña Comentar nos abre una ventana que nos lleva a una página de discusiones sobre scratch. En ella se comentar dudas, intervenir en foros o hacer un nuevo foro con un temo en concreto.
- ✓ Ayuda: La pestaña Ayuda nos lleva a la interfaz de ayuda en donde podemos descargar tutoriales, guías y recursos de Scratch.
- ✓ Buscar: Es una opción para localizar proyectos de la web. Cuando se comparte un proyecto, a este se le atribuyen etiquetas. Etiquetas como, por ejemplo, “games”, “juegos”, “matemáticas”, “math”, etc. De esta

forma, es más sencillo localizar proyectos con una temática de interés para el usuario scratch. Cuando localizas un proyecto en la web de scratch, puedes ver su código, copiarlo y modificarlo y después compartirlo. De esta forma se fomenta la colaboración y conocimientos entre los usuarios de scratch. El nuevo proyecto, creado a partir de otro, debe presentar unos reconocimientos al proyecto o persona del que has recogido la idea, el código u otros elementos del proyecto origen. Por defecto, todos los proyectos compartidos en la web de scratch, como puede verse en el siguiente [enlace de licencia](#), están amparados por la licencia creative commons [CC-BY-SA](#). Esta licencia obliga a compartir los proyectos creados a partir de otros con la misma licencia, pero, hay algunos usuarios que utilizan para sus proyectos originales licencias menos restrictivas como la [CC-BY](#).



# Módulo 3

---

## Trabajar con Scratch bajo dos perspectivas

---

### 3. Trabajar con Scratch bajo dos perspectivas.

La herramienta Scratch 2.0 ayuda al docente y al alumno a interrelacionarse de una forma más colaborativa, motivadora, atractiva y experimental en el aula. El **docente y el alumno** trabajarán con conceptos tales como, creatividad, entretenimiento, lógica, experimentación, colaboración, interactuar con el entorno y aprendizaje significativo, entre otros muchos. Ambos necesitarán inicialmente de **unas guías** a partir de las cuales comenzar a crear, desarrollar y modificar según unos intereses, necesidades y funcionalidades concretas.

En este módulo 3 se pretende aportar esas necesidades de partida para que ambos, docente y alumno/a, se desenvuelvan con esta herramienta y utilicen la misma combinada con otros dispositivos que aumentarán el interés y el aprendizaje significativo del alumnado.

Persiguiendo obtener las competencias que se han descrito al inicio del curso, en este módulo 3, los **objetivos** de aprendizaje que nos llevarán a poder desenvolverlas son los siguientes:

- ✓ Conocer, experimentar, probar y modificar diferentes programas diseñados con la herramienta Scratch en las áreas de conocimiento científico tecnológico.
- ✓ Conocer y experimentar las diferentes funcionalidades que puede ofrecernos el uso de Scratch combinado con otras aplicaciones o kits como son la placa de sensores Picoboard, el kit Makey Makey, el dispositivo Leap Motion y los robots WeDo de LEGO y mOway.
- ✓ Conocer y experimentar con pequeños juegos que podrá hacer el alumnado.

#### 3.1. Soy docente scratcher

Como ya sabemos, *Scratch*, además de ser un programa de software libre, es un lenguaje de programación basado en LOGO. Este anglicismo puede traducirse literalmente por "arañar" o "rayar". En nuestro caso, hemos introducido este apartado con una palabra derivada del programa, "*scratcher*", pero nada tiene que ver con "ser un arañador" o "ser un rayador". El sentido de la misma en este documento es más simple y obvio: ser un *usuario activo y competente con la herramienta scratch*.

Pero, ¿cómo se consigue ser un "*scratcher*"? Hay dos formas o maneras de conseguirlo:

- ✓ Pasarse horas y horas creando, diseñando, experimentando y probando con el scratch
- ✓ Entender y modificar programas ya diseñados para un área de aprendizaje y materia concreta consiguiendo acercarlos a nuestras motivaciones e intereses.

Sin duda, el segundo método es el más rápido y es el que seguiremos. El curso se ha diseñado con una estructura lógica de contenidos para ayudar a alcanzar las competencias y objetivos descritos. Por lo tanto, es necesario aplicar lo que hemos aprendido en el módulo 2. Ese aprendizaje, unido a la experimentación con programas ya creados y que nos parecen atractivos, ayudará a conseguir que el docente adquiera las habilidades, estrategias y destrezas para finalmente llegar a hacer sus propias creaciones.

Se ha subdividido esta formación en dos partes: programas funcionales y ampliación a otros dispositivos y aparatos que se pueden combinar con el programa.

Comenzaremos por la comprensión y modificación de programas funcionales en el aula (válidos para diferentes materias) para continuar aprovechando todavía más esta herramienta logrando utilizarla en combinación con otros dispositivos y kits atractivos y motivadores, que son elementos que aumentan el "querer aprender" y creatividad de nuestro alumnado.

### **3.1.1. Creación y modificación de programas funcionales en el aula para mi currículo (patrones). Ejemplos particularizados para las áreas de Tecnología, Matemáticas, Física y Química y Biología y Geología.**

A continuación veremos cómo se han creado una serie de programas que considero que pueden ser útiles para el aula en diferentes materias. Por ejemplo, el programa "Pasapalabra de Tecnología" que puede utilizarse para descubrir los conocimientos iniciales o de partida que posee un alumno/a al inicio de un tema.

Al inicio de cada tema, muchos docentes realizamos lo que se denomina una "batería de preguntas" iniciales, con el fin de descubrir qué saben y qué no saben nuestros alumnos o qué ideas tienen sobre aspectos importantes del tema. Este programa, en concreto, con pequeñas modificaciones, nos servirá para ese fin: sólo hay que modificar las preguntas que se le quieren hacer.

Los ejemplos que se mostrarán, aunque se apliquen es un área en concreto, podrán ser modificados para ser utilizados en otras disciplinas. Para ejemplificar los diferentes currículos que vosotros enseñáis, los ejemplos se separarán en las siguientes secciones:

1. Tecnología
  2. Matemáticas
  3. Física y Química
  4. Biología y geología
- 

## 1. Tecnología

Como ejemplos de programas en esta materia, se incluyen los siguientes:

- a. Pasapalabra de Tecnología
- b. Juega con Bee Bot (tres niveles)
- c. Parquímetro
- d. Medida de la temperatura con una NTC

### a. Pasapalabra de Tecnología

Este programa ha sido creado para que ayude al docente a conocer los conocimientos iniciales que poseen sus alumnos en el comienzo de una nueva unidad didáctica. Los jugadores, sus alumnos, interactúan con el juego y responden a las cuestiones, pudiendo acertar o fallar.

- ✓ Entender el programa: La explicación de la programación del programa se muestra en el siguiente documento: Descripción y desarrollo del programa pasapalabra (**pendiente de creación**<sup>1</sup>)
- ✓ ¿Cómo Jugar? Al presionar la bandera verde, te preguntará si quieres o no jugar. Si escribes "s" o "si" comenzarás el pasapalabra de tecnología. También tiene un saco de preguntas y respuestas que se pueden añadir. Para ello debes pulsar la letra "n" e incluir la respuesta con la pregunta.
- ✓ Demostración visual de "Pasapalabra de Tecnología":

<http://scratch.mit.edu/projects/19342867/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: "Pasapalabra de Tecnología".

### b. Juega con Bee Bot (3 niveles)

Este programa ha sido creado para el alumnado de infantil (algunos ya han trabajado con el robot Bee Bot de Lego). Entender el programa dará paso a la

---

<sup>1</sup> Se explicará de forma presencial en el curso. Próximamente, se incluirá como documento de texto.

introducción de patrones que ahí se programan y nos pueden servir para otros programas diferentes.

- ✓ Entender el programa: La explicación de la programación del programa se muestra en el siguiente documento: Descripción y desarrollo del programa Juega con Bee Bot (**pendiente de creación**)
- ✓ ¿Cómo Jugar? Al presionar la bandera verde, comenzará el primer nivel. Dispones de una consola de mandados que hará que Bee Bot se mueva hacia delante, hacia atrás o gire en la misma cuadrícula hacia la derecha o izquierda. Bee Bot debe llegar a la flor para poder pasar de nivel. Y, ojo, hay que tener cuidado con salirse de las casillas y con los objetos que nos impiden continuar.
- ✓ Demostración visual de "Juega con Bee Bot":

<http://scratch.mit.edu/projects/20891022/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: "Juega con Bee Bot".

### c. Parquímetro

- ✓ Entender el programa: La programación de este proyecto fue creada por Marco Antonio Rodríguez Fernández en el curso "Robots y Videojuegos en las aulas: Scratch y Arduino para profesores". Utiliza tres variables que son: Cambio, Dinero y Tiempo de Parquing. El precio de 1h de parquing es de 1,2€ por lo que el  $Cambio = Dinero - (Tiempo\ de\ Parquing \times 0,02)$ .
- ✓ ¿Cómo Jugar? Al hacer clic en cada moneda, se incrementa el dinero (en euros). Al hacer clic en los botones más y menos del parquímetro, añadimos o restamos, respectivamente, 5 minutos en cada clic. Automáticamente, al introducir el dinero y el tiempo de parking, nos ofrecerá el cambio.
- ✓ Demostración visual de "Parquímetro":

<http://scratch.mit.edu/projects/19436377/>

### d. Medida de Temperatura con NTC.

- ✓ Entender el programa: La explicación de la programación del programa se muestra en el punto 3.1.2.2. de este documento.
- ✓ ¿Cómo Jugar? El programa necesita la placa de sensores Picoboard y una NTC. Tras presionar la bandera verde, se calibra el termistor. Ahora, ya podemos realizar la medición de la temperatura en el objeto deseado.
- ✓ Demostración visual de "Medida de Temperatura con NTC":

<http://scratch.mit.edu/projects/25453438/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: "Medida de Temperatura con NTC".

---

## 2. Matemáticas

Como ejemplos de programas en esta materia, se incluyen los siguientes:

- a. Demostración del Teorema de Pitágoras
- b. Dibujar una circunferencia
- c. Cálculo de la resultante de dos vectores.
- d. Curva de Koch

### a. Demostración del Teorema de Pitágoras

- ✓ Entender el programa: Este proyecto es el resultado de la modificación del proyecto original "Pythagorean\_theorem0" del usuario scratch "dapontes". Descripción y desarrollo el programa (pendiente de creación)
- ✓ ¿Cómo Jugar? Al presionar la bandera verde, debes hacer clic en Inicio. Comienza la animación que mueve el vértice A del triángulo rectángulo, mostrando, en cada punto, el área de la región cuadrada que forma cada lado del cuadrado. Si lo paramos, vemos que en ese punto, la suma de las áreas de los cuadrados de los catetos coincide con el valor del área del lado hipotenusa. En la pestaña Explicación se muestra esta suma de áreas.
- ✓ Demostración visual de "Teorema de Pitágoras (Demostración)":

<http://scratch.mit.edu/projects/24276842/>

### b. Dibujar una circunferencia

- ✓ Entender el programa: Este programa se ha explicado en el Ejemplo 14 del módulo 2.
- ✓ ¿Cómo Jugar? Al hacer clic en la bandera verde comienza a crearse una circunferencia de radio R. Para variar el Radio, actúa en el deslizador de la variable R. Al parar el programa (bandera roja) se muestra la posición (X,Y) de ese punto en la circunferencia.
- ✓ Demostración visual de "Dibujar una circunferencia":

<http://scratch.mit.edu/projects/25507776/>



Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: "[Dibujar una circunferencia](#)"

### c. Cálculo de la resultante de dos vectores

- ✓ Entender el programa: Este programa se ha explicado como Ejemplo 1 del módulo 3 en el apartado 3.2.2. de este documento.
- ✓ ¿Cómo Jugar? Al hacer clic en la bandera verde debes seguir las instrucciones: hacer clic en inicio e introducir el módulo de A y de B con su ángulo. Te ofrece el resultado gráfico la resultante de esos 2 vectores. Al hacer clic en "Saber Más" nos proporciona los valores numéricos del vector resultante: R, Rx, Ry y el ángulo ( $\gamma$ ) con el eje X.
- ✓ Demostración visual de "Cálculo de la resultante de dos vectores":

<http://scratch.mit.edu/projects/25508032/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: "[Cálculo de la resultante de dos vectores](#)".

### d. Curva de Koch (dibujo de un fractal)

- ✓ Entender el programa: La programación de este proyecto fue creada por Marco Antonio Rodríguez Fernández en el curso "Robots y Videojuegos en las aulas: Scratch y Arduino para profesores". Utiliza un bloque que llama Koch en el que define y programa dos variables: tamaño del eje X (480) y número de iteraciones o repeticiones (5). Cuando hace la llamada al bloque fracciona la curva de Koch según el número de iteraciones que tenga, ejecutándose de forma recursiva en iteraciones-1, e intercambiando sucesivamente el ángulo de giro entre  $60^{\circ}$  (izquierda) y  $120^{\circ}$  (derecha).
- ✓ ¿Cómo Jugar? Al hacer clic en la bandera verde y observar el fractal que realiza el lápiz.
- ✓ Demostración visual de "Curva de Koch":

<http://scratch.mit.edu/projects/19619702/>

### e. Función cuadrática

- ✓ Entender el programa: Este programa es muy sencillo presentando un único objeto que denomino "punto". El punto se dibuja desde (-240, 0) haciendo variar la x por 1 hasta 239 y la y según la fórmula de la función parabólica para los valores a, b y c que has introducido con el comando preguntar...y esperar del bloque "Sensores".

- ✓ ¿Cómo Jugar? Al hacer clic en la bandera verde debes seguir las instrucciones y escribir los valores de los parámetros “a”, “b” y “c” de la función parabólica. Te ofrece el resultado gráfico la parábola para esos valores.
- ✓ Demostración visual de "Función cuadrática":

<http://scratch.mit.edu/projects/19456682/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: “[Función cuadrática](#)”.

---

### 3. Física y química

Como ejemplos de programas en esta materia se introducen los siguientes:

- a. Cálculo de la resultante de dos vectores (Vector fuerza)
- b. Estados de la materia (Picoboard)
- c. Partículas en el átomo
- d. Tiro parabólico

#### a. Cálculo de la resultante de dos vectores

- ✓ Entender el programa: Este programa se ha explicado como Ejemplo 1 del módulo 3 en el apartado 3.2.2. de este documento. Se presenta en el área de matemáticas y de Física y Química porque los vectores pueden representar fuerzas, perteneciendo a la parte de estática del currículo.
- ✓ ¿Cómo Jugar? Al hacer clic en la bandera verde debes seguir las instrucciones: hacer clic en inicio e introducir el módulo de A y de B con su ángulo. Te ofrece el resultado gráfico la resultante de esos 2 vectores. Al hacer clic en “Saber Más” nos proporciona los valores numéricos del vector resultante: R, Rx, Ry y el ángulo ( $\gamma$ ) con el eje X.
- ✓ Demostración visual de "Cálculo de la resultante de dos vectores":

<http://scratch.mit.edu/projects/25508032/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: “[Cálculo de la resultante de dos vectores](#)”.

## b. Estados de la materia

- ✓ Entender el programa: Este programa se ha explicado como Ejemplo 2 del módulo 3 en el apartado 3.1.2.2. de este documento (Ver vídeo). Para su funcionamiento se debe disponer de la tarjeta de sensores Picoboard a la que conectaremos un sensor de luz en el puerto A.
- ✓ ¿Cómo Jugar? Al hacer clic en la bandera verde se muestra el escenario de inicio. Haciendo en el botón inicio nos muestra de forma gráfica la posición que tendrían las moléculas en los diferentes estados (sólido, líquido o gaseoso), indicándonos un valor numérico de temperatura y velocidad para ese estado.
- ✓ Demostración visual de "Estados de la materia".

<http://scratch.mit.edu/projects/25884076/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: "[Estados de la materia](#)".

## c. Partículas en el átomo

- ✓ Entender el programa: El programa dispone de 3 objetos protón y 3 neutrón que giran siempre y de forma continua  $10^0$  a la izquierda y 4 objetos órbitas (con su electrón) que giran siempre y de forma continua  $5^0$  hacia la derecha. a mayores dispone de 5 objetos (play, flecha, stop, flecha gif y átomo gif) cuya misión es usarse como pulsadores o como iniciadores del movimiento (activadores) de las partículas del átomo.
- ✓ ¿Cómo Jugar? Al hacer clic en la bandera verde deben seguirse las instrucciones. El programa muestra de forma no rigurosa una animación que simula el movimiento y situación de las partículas del átomo: protón, neutrón y electrón.
- ✓ Demostración visual de "Partículas en el átomo".

<http://scratch.mit.edu/projects/25063749/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: "[Partículas en el átomo](#)".

## d. Tiro parabólico

- ✓ Entender el programa: Este proyecto es el resultado de la modificación del proyecto original "[ParabolasV2](#)" del usuario scratch "cupati". Descripción y desarrollo el programa (pendiente de creación)

- ✓ ¿Cómo Jugar? Al presionar la bandera verde, comienza una animación. Si sigues las instrucciones el cañón dispara la bala. A partir de ahí existen varias formas de interaccionar con el programa.

Elegir los valores de  $\alpha$ ,  $V_0$  y  $H$

- Pulsar START para ver la trayectoria (repetir cuantas veces se quiera)
- Pulsar LIMPIAR para borrar la trayectoria del escenario
- Pulsar AUTO para la animación (tarda varios minutos en calcular  $H$  y  $D$  ángulo Máxima. Aquí se observa la influencia del ángulo de lanzamiento en la altura máxima y en el alcance del tiro). Para parar la animación hacer clic en la bandera roja

- ✓ Demostración visual de "Tiro parabólico":

<http://scratch.mit.edu/projects/25505649/>

---

#### 4. Biología y geología

Como ejemplos de programas en esta materia se introducen los siguientes:

- a. Grupos sanguíneos
- b. Partículas en el átomo
- c. Estructura dunha ameba

##### c. Grupos Sanguíneos

- ✓ Entender el programa: La explicación de la programación del programa se muestra en el siguiente documento: Descripción y desarrollo del programa Grupos sanguíneos (**pendiente de creación**).
- ✓ ¿Cómo Jugar? Al hacer clic en la bandera verde deben seguirse las instrucciones visuales con iconos o con texto. El programa presenta dos partes: la primera es de conocimiento y la segunda es un juego.

1º Te pedirá tu grupo sanguíneo y te dará información sobre a quién puedes donar y quién te puede donar (podrás repetir para otros grupos)

2º Juego: El vampiro tiene sed y quiere morder. Ganas si el vampiro muerde 25 veces y no pierde las 3 vidas (ojo con el zombie). El vampiro se moverá siguiendo tu cursor del ratón.

- ✓ Demostración visual de "Grupos Sanguíneos".

<http://scratch.mit.edu/projects/25323497/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: "[Grupos Sanguíneos](#)".

## **b. Partículas en el átomo**

Este programa se ha explicado en el ejemplo c del área de Física y Química. Esta temática también se encuentra incluida en la materia de Biología y Geología.

## **c. Estructura dunha ameoba**

- ✓ Entender el programa: El programa simula el típico juego (ejercicio) de situar las partes del objeto en su lugar correspondiente. Es muy útil ya que cambiando la imagen y el valor de los ítems de las listas (indicadores y respuestas) disponemos de otro proyecto nuevo con la misma funcionalidad. La explicación de la programación del programa se muestra en el siguiente documento: Descripción y desarrollo del programa Estructura dunha ameoba (**pendiente de creación**).
- ✓ ¿Cómo Jugar? Al hacer clic en la bandera verde deben seguirse las instrucciones. Seguir las instrucciones. Se aconseja hacer clic en los botones por este orden:

1º Automático: Realiza la actividad situando la respuesta en el indicador apropiado.

2º Xogar Agora (Arrastrar el NOMBRE de la parte de la estructura de modo que el círculo rojo se deje encima del NÚMERO deseado). Si es correcta, te dirá "OK" y después se pondrá en verde.

Para parar: Círculo rojo (al lado de la bandera verde)

- ✓ Demostración visual de "Estructura dunha ameoba".

<http://scratch.mit.edu/projects/25547790/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: "[Estructura dunha ameoba](#)".

### 3.1.2. Combinación de la herramienta Scratch con otros dispositivos y kits

Con el lenguaje de programación Scratch (por ahora en la versión 1.4, pero pronto en la 2.0) se puede sensorizar el entorno físico, es decir, conseguir leer datos de sensores de luz, sonido, temperatura o humedad, entre otros, y utilizarlos para que los objetos que hemos diseñado o creado en un programa de Scratch, respondan a diferentes estímulos del mundo físico real. En este apartado vamos a ver como se puede combinar el programa scratch con otros dispositivos con el fin de obtener una mayor funcionalidad del programa y con ello, conseguir sacarle más partido en el aula. 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 una forma más directa. El alumno/a, al sentirse atraído por la novedad y potencial de estos dispositivos, se sentirá motivado y deseará experimentar, crear y aprender.

Los dispositivos que vamos a utilizar son los siguientes:

1. Kit Makey Makey
2. Tarjeta de sensores Picoboard
3. Dispositivo Leap Motion
4. Robot WeDo de LEGO
5. Robot mOway

#### 3.1.2.1. Kit Makey Makey

El kit Makey Makey es un dispositivo que no requiere de ninguna instalación. Simplemente, utiliza un microcontrolador para comunicarse con el ordenador mediante USB HID. Esto hace que el ordenador lo detecte como si fuese un ratón o un teclado. Además, esta herramienta es apta para cualquier sistema operativo (Windows, iOS o Linux) abriendo una abanico de posibilidades de uso.

1. ¿Para qué sirve? Podríamos definirlo como una consola de mandos ya que puede convertir cualquier cosa en un teclado.
2. ¿Hasta qué límite? Su placa presenta seis entradas en la parte delantera, las cuales se pueden conectar fácilmente (mediante cables con terminales cocodrilo) a objetos (sean o no muy conductores, ya que estas entradas presentan una alta impedancia y logran detectar la corriente eléctrica). Por su parte trasera, dispone de 12 entradas



accesibles mediante cables con pines hembra, las cuales, seis son para elementos de teclado y otras seis para movimientos del ratón.

Para los más intrépidos, el kit permite una reprogramación en el entorno Arduino, pudiendo crear nuevos controles en el mismo y modificar el comportamiento del kit.

En el siguiente vídeo se observa la utilización del kit como consola de mandos. En él se describe, para un pequeño programa en el entorno Scratch 1.4., como puede conectarse para que el usuario interactúe con el programa.

<https://vimeo.com/94194005>

*Kit Makey Makey con el Scratch 1.4 . Susana Oubiña Falcón (CC BY)*

### **A. Plastilina conductora**

Sabemos que, para poder conectar la mayoría de elementos electrónicos en un circuito, hace falta colocar una resistencia cuya función es limitar la cantidad de corriente eléctrica que circula por el mismo, evitando así el deterioro de los componentes. Si esta resistencia, en lugar de conectarla externamente, estuviera contenida en el propio material conductor, nos ahorraríamos el tenerla que añadir y el montaje sería más directo. A mayores, si el material conductor fuera manejable, fácilmente moldeable, e incluso permitiera que los componentes electrónicos se pudieran unir sin necesidad de soldadura, sería fantástico. Por estas razones, se muestra en este curso como cocinar plastilina conductora.

Aunque el kit Makey Makey presenta una alta impedancia y casi cualquier objeto nos sirve para conectar al kit (nuestros dedos son conductores), para prácticas de electricidad y electrónica (o incluso para química para estudiar la composición de la plastilina conductora) o simplemente, como elemento de conexión y unión, es interesante utilizar plastilina conductora. Este material se puede construir fácilmente y hay varias recetas por internet. Particularmente, yo utilizo la siguiente receta.

Los ingredientes son:

- 100g de harina
- 100ml de agua
- Dos cucharadas de sal
- Una cucharada de aceite vegetal
- El zumo de dos limones
- Unas gotas de colorante alimentario (azafrán, etc). Este elemento le dará el color que deseamos.

Preparación:

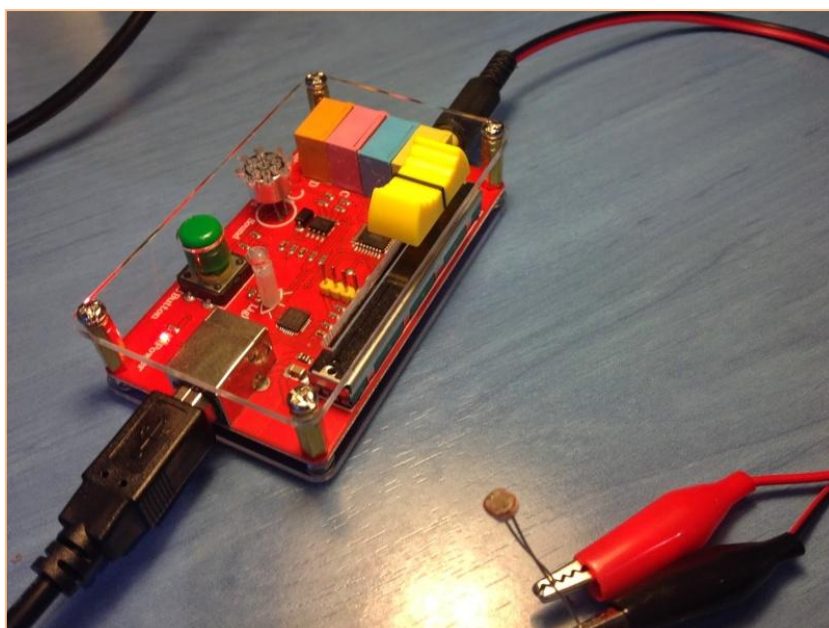
Se introducen todos los ingredientes juntos en un cazo u otro recipiente de cocción, y se remueve la mezcla hasta que quede uniforme y se observe que los componentes están todos disueltos. Se pone el recipiente a fuego medio sin dejar de remover hasta que se consigue una "bola" de plastilina en el centro del mismo. En este punto, se saca la bola sobre una tabla (plato, etc) previamente espolvoreada con harina y se deja enfriar. Una vez fría, se amasa hasta que queda una masa uniforme. NOTA: en el caso de que quede pegada en las manos se le puede añadir un poco de aceite.

## B. Aplicación

<http://www.codemads.com/2014/02/com-fer-un-piano-amb-plastilina-makeymakey-i-scratch-2-0-en-menys-de-5-minuts/>

### 3.1.2.2. Tarjeta de sensores Picoboard

Existen otros dispositivos con mayores funcionalidades que el kit Makey Makey ya que nos aportan mediciones sobre magnitudes de estudio en un objeto del entorno. Esos dispositivos son tarjetas de sensores y, como su nombre indica, incluyen un gran abanico de sensores (temperatura, luz, etc) ofreciendo con ello más posibilidades educativas. Entre ellas, se utiliza la PicoBoard, de la compañía "The Playful Invention", la tarjeta de Arduino o la tarjeta de sensores (TDS) de la empresa A&S Design. Similar a la PicoBoard es la placa PicoCat [1], usada tanto en Primaria como Secundaria.



*Picoboard con sensor de luz. Susana Oubiña Falcón. (CC BY)*

La PicoBoard [2] trae integrados 4 sensores: sensor de luz, sensor de sonido, sensor de toque (botón) y un sensor de deslizamiento (slider). Pero no son los únicos sensores que podemos activar. Esta placa dispone de 4 entradas que pueden servir como medio de acoplamiento de otros sensores (temperatura, humedad, etc) o para medir la resistencia eléctrica de un circuito. La placa se conecta al ordenador mediante un cable USB y, para que sea detectada, es necesario descargar un driver (en la caja del kit viene especificada la dirección web donde se puede descargar el driver).

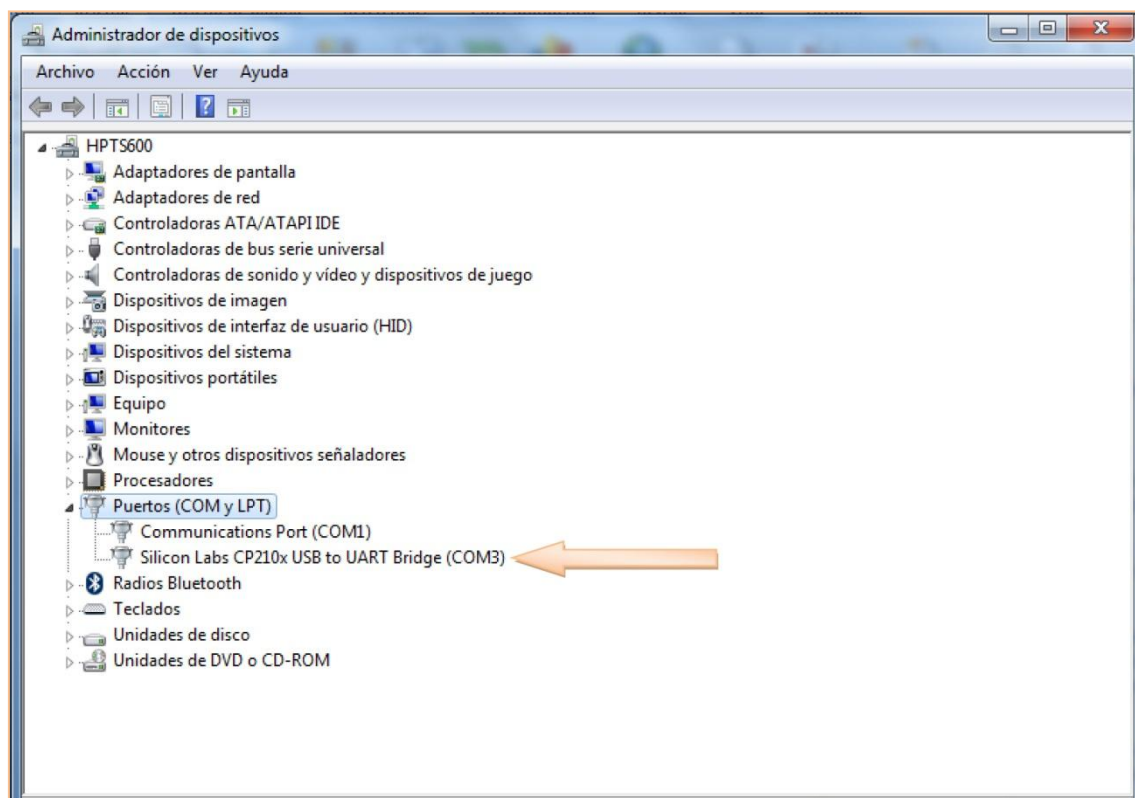
Las referencias que se han utilizado para explicar los contenidos de este subapartado son las siguientes:

[1] "PicoCat 3.0". E.R.I Didactic S.L. [En Línea]. Disponible en la dirección web: <http://www.erididactic.com/medias/documents/PicoCat3.0.pdf>. [Accedido el 25 de abril de 2014]

[2] "PicoBoard - Sensor Board that works with MIT's Scratch". PICO CRICKET. The Playful Invention Company. [En línea]. Disponible en: <http://www.picocricket.com/picoboard.html> . [Accedido el 25 de abril de 2014]

## A. Conectar la Picoboard

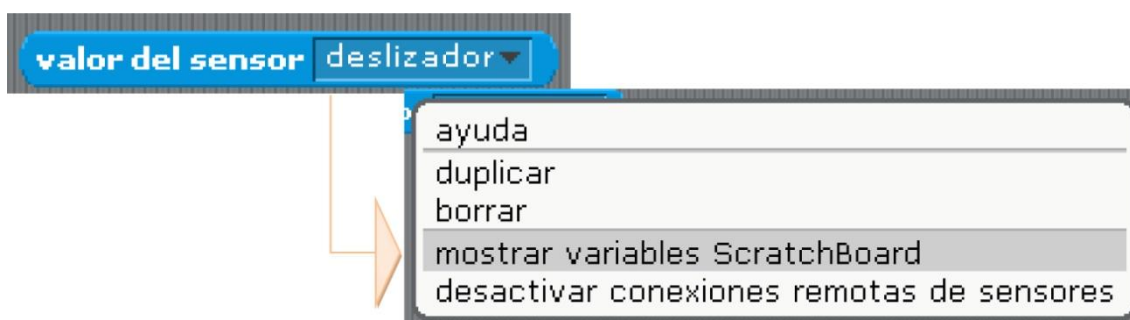
De estas posibles opciones de tarjetas de sensores, nosotros usaremos la picoboard. Antes de comenzar a probarla debemos instalar un driver. El driver lo podemos descargar del siguiente link: [Driver Picoboard](#). En nuestro caso, como puede comprobarse en la siguiente imagen, se ha elegido para él el puerto COM3.



*Pantalla que muestra la instalación del Driver en COM3. Susana Oubiña Falcón. (CC BY)*

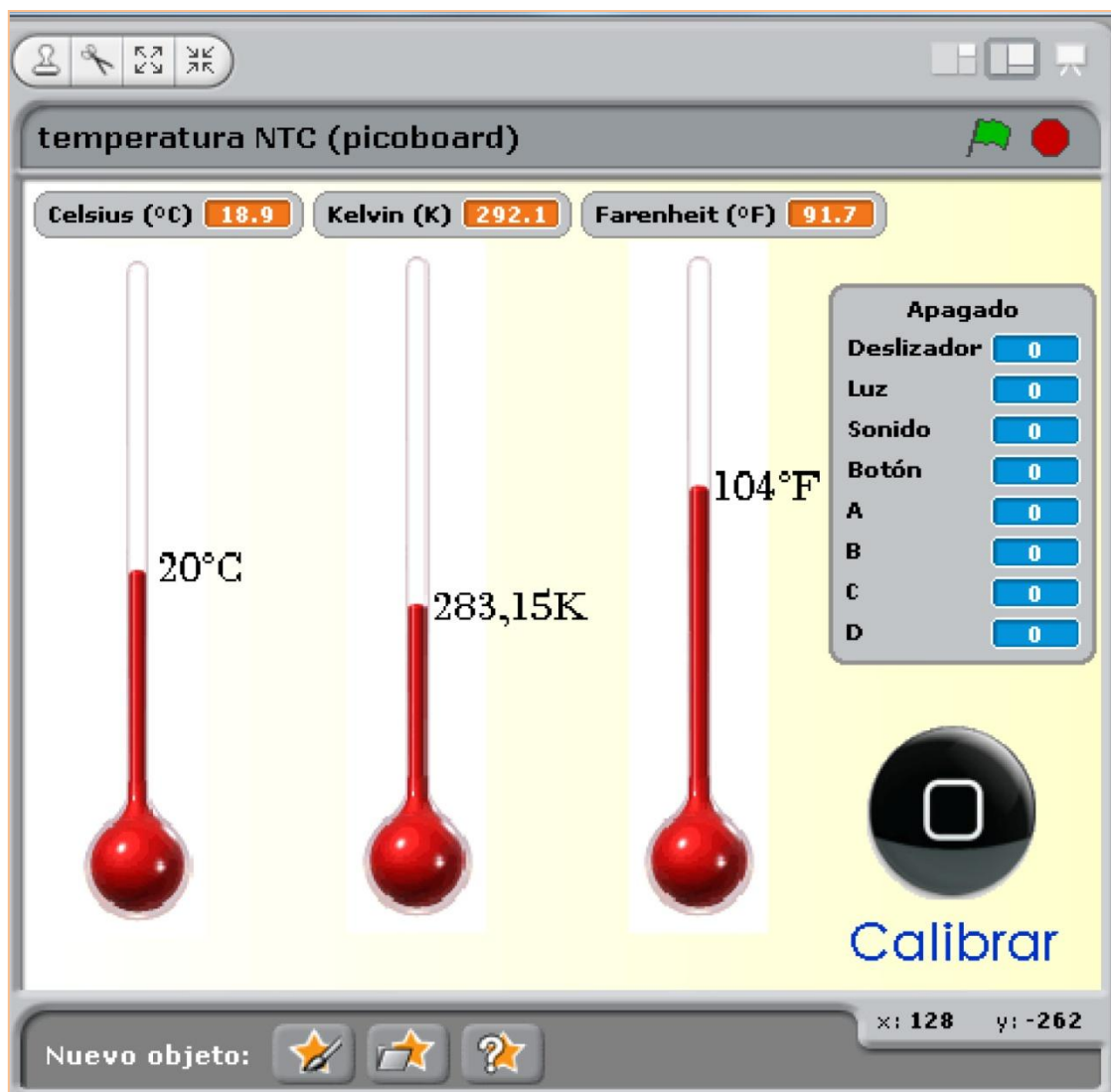
Los sensores de la picoboard reciben datos continuamente. Para que scratch los pueda leer e interactuar con ellos debemos realizar los siguientes pasos:

1. Instalar el driver (sólo se realiza una vez. En mi caso usaré el puerto COM3).
2. Abrir el scratch 1.4 y conectar la picoboard por su USB al ordenador.
3. Ir al bloque **Sensores** del scratch 1.4 y hacer clic con el botón derecho encima del comando “valor del sensor...”. Se nos abrirá un submenú y en el debemos escoger la opción “mostrar variables ScratchBoard”:



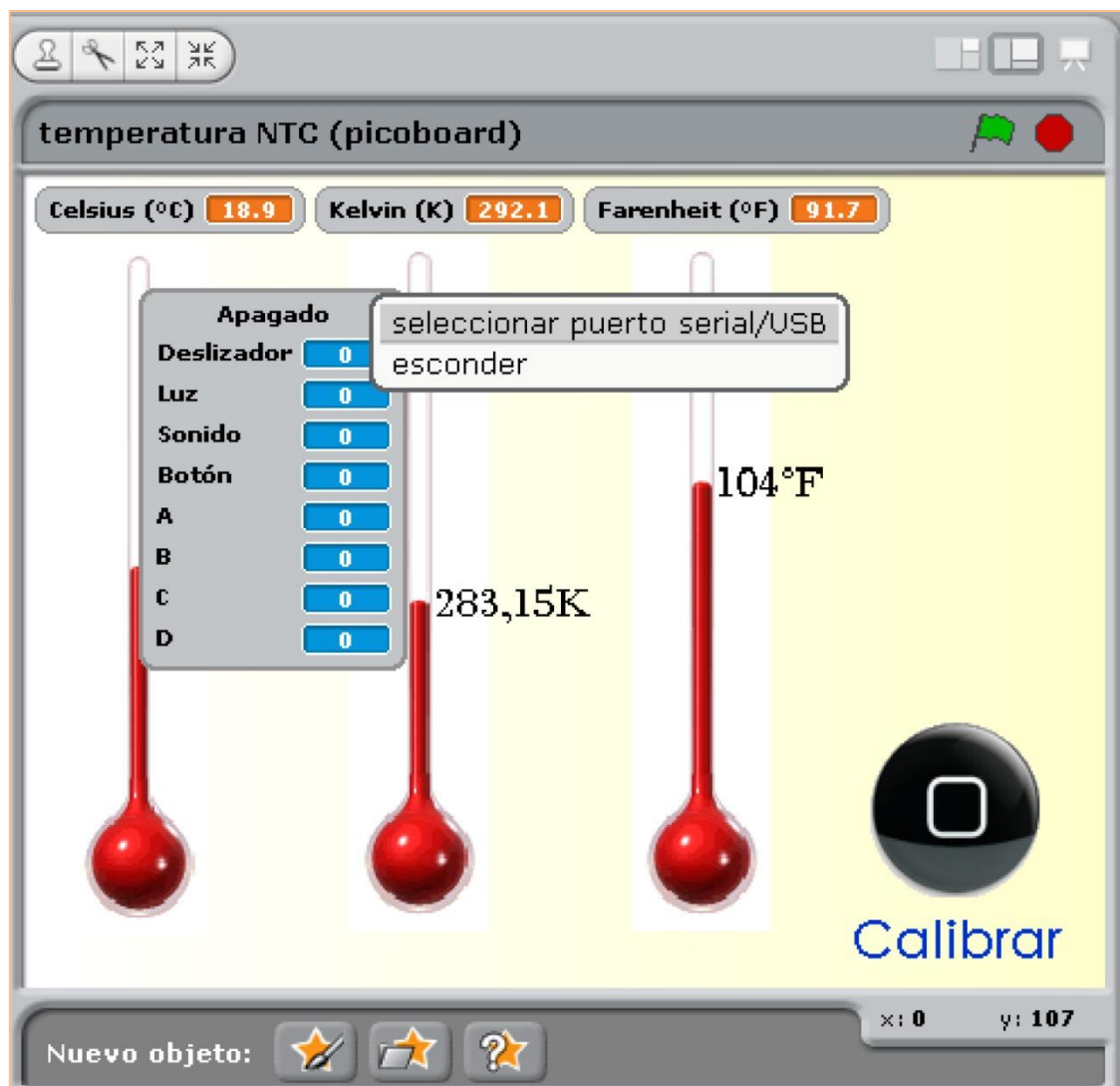
*Mostrar variables ScratchBoard. Susana Oubiña Falcón. (CC BY)*

4. Si ahora vemos el escenario del programa scratch, observamos que nos aparece una imagen gris/azul rectangular que simula la picoboard, con sus 4 sensores fijos y sus 4 puertos A, B, C y D. Por ahora la picoboard aun no está recibiendo datos, se encuentra en fase **Apagada**, tal y como puede verse en la siguiente imagen:



*Picoboard en fase "Apagada". Susana Oubiña Falcón. (CC BY)*

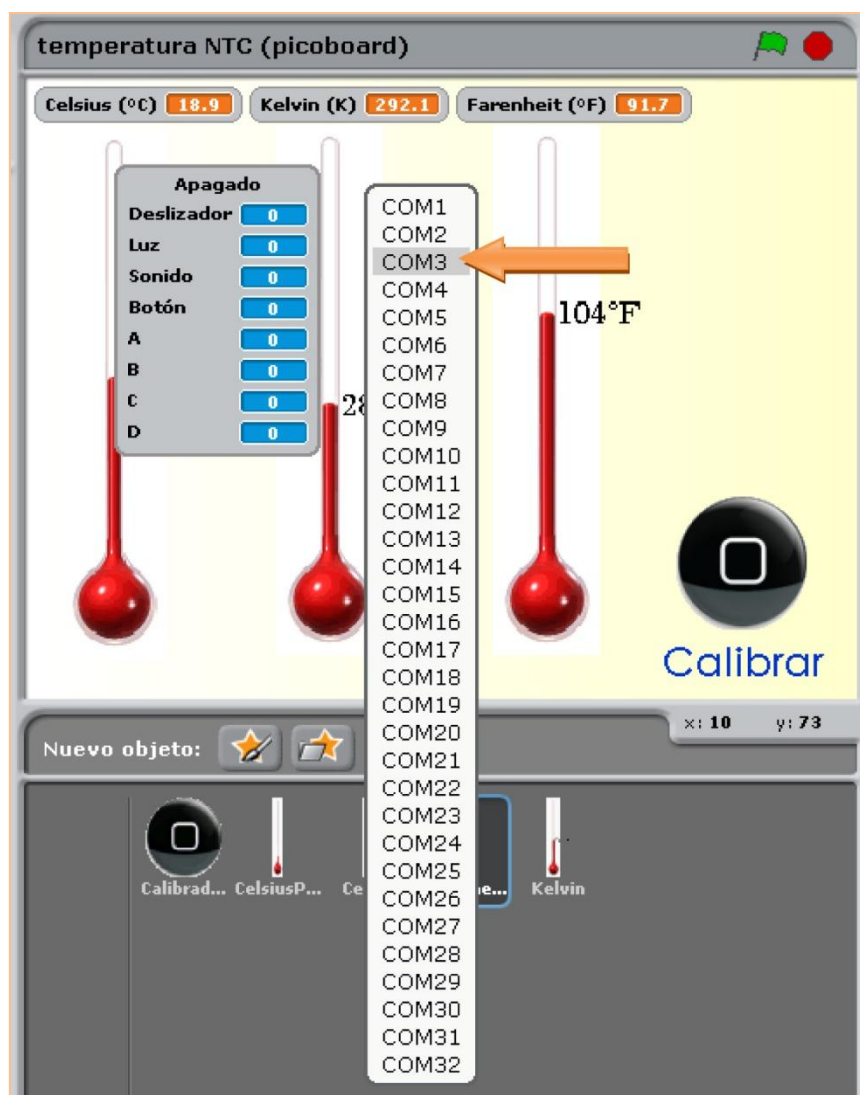
5. Para activarla debemos seleccionar el puerto que hemos elegido para la instalación del driver. Para ello, ponemos el ratón encima de la imagen de la picoboard y hacemos clic con el botón derecho marcando "seleccionar puerto serial/USB":



*Seleccionar puerto USB. Susana Oubiña Falcón. (CC BY)*

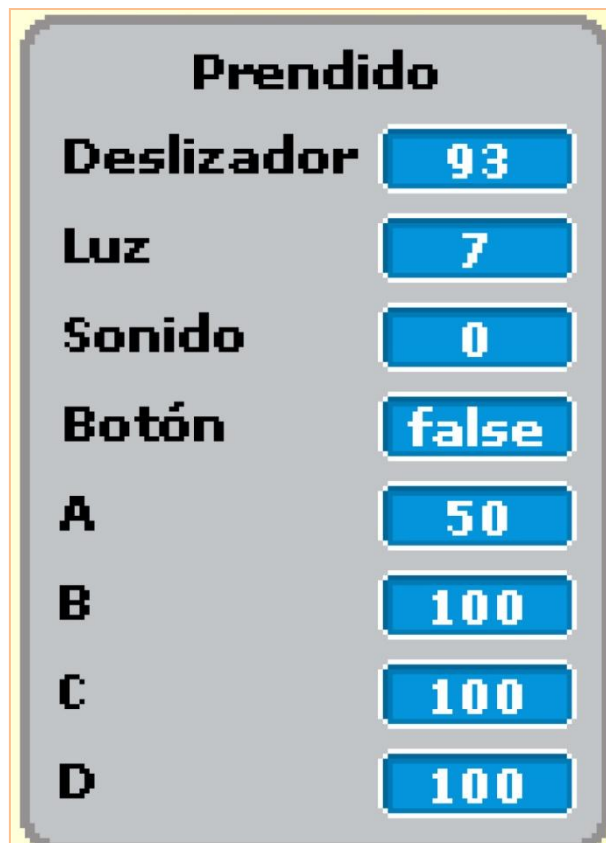
6. Finalmente, elegimos el puerto. En nuestro caso, COM3.





*Pantalla elección puerto COM3. Susana Oubiña Falcón. (CC BY)*

Desde este momento, los sensores de la picoboard envían datos al scratch y este los recibe, pudiendo procesarlos. En el escenario del scratch, puede observarse que la imagen de la picoboard se encuentra en fase “**Prendido**”:



*Picoboard en fase "Prendido". Susana Oubiña Falcón. (CC BY)*

## B. Ejemplos

A continuación vamos a mostrar algunos ejemplos en los cuales el scratch, por comunicarse con la picoboard, adquiere mayores utilidades.

- ✓ Ejemplo 1: En el siguiente vídeo se muestra cómo se pueden programar los 4 sensores fijos de la tarjeta Picoboard, así como, una termoresistencia conectada al puerto A y que controla la temperatura.

*"Prueba de sensores con Picoboard"* from Susana Oubiña Falcón on Vimeo.

Licencia CC BY

Se explica la programación y funcionamiento de la tarjeta de sensores Picoboard con el Scratch 1.4

- ✓ Ejemplo 2: Utilizado el programa scratch 1.4 y la tarjeta de sensores Picoboard a la que le acoplamos un sensor de luz a su puerto A, se diseña un programa que simula los tres estados de la materia: sólido, líquido y gaseoso. El siguiente video muestra el funcionamiento del programa.

*"Simulación de los estados de la materia con el Scratch 1.4 y Picoboard"* from Susana Oubiña Falcón on Vimeo. Licencia CC BY

Video tutorial que explica el programa de simulación de los estados de la materia utilizando un sensor de luz (5K-20K OHM 4.20mm) con la picoboard y el programa libre scratch 1.4

- ✓ Ejemplo 3: Medición de la temperatura en °C, K y °F. Para ello usamos un termistor NTC 10KΩ 5%.

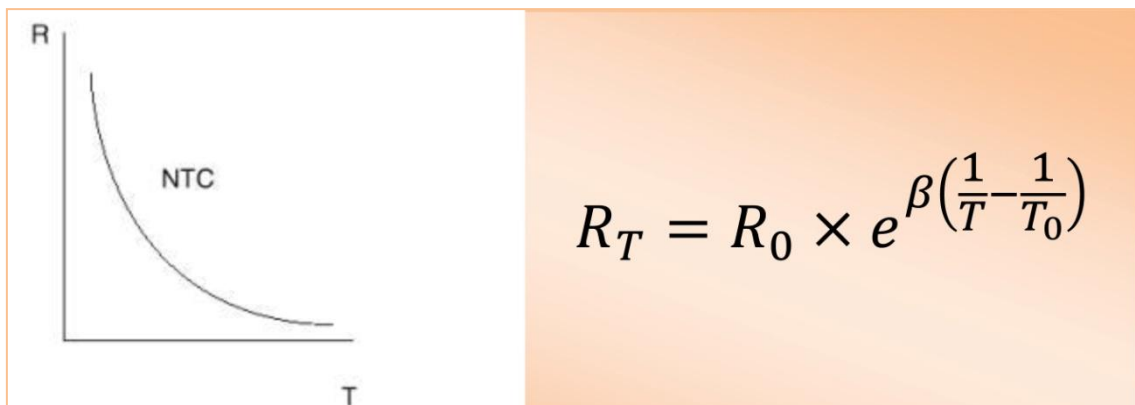


*Termistor NTC. Susana Oubiña Falcón. (CC BY)*

Un termistor o sensor de temperatura es una resistencia cuyo valor varía con la temperatura. En nuestro caso, disponemos de un NTC, es decir, el valor de la resistencia decrece según aumenta la temperatura. Un NTC está formado por materiales semiconductores, de modo que la relación de R con T no es lineal:

$R(T) = R_0 (1 - \alpha(T - T_0))$  Para materiales conductores (R lineales).

Un termistor cumple la ley de SteinHart-Hart, comportándose de forma exponencial:



*Comportamiento de una NTC. Susana Oubiña Falcón. (CC BY)*

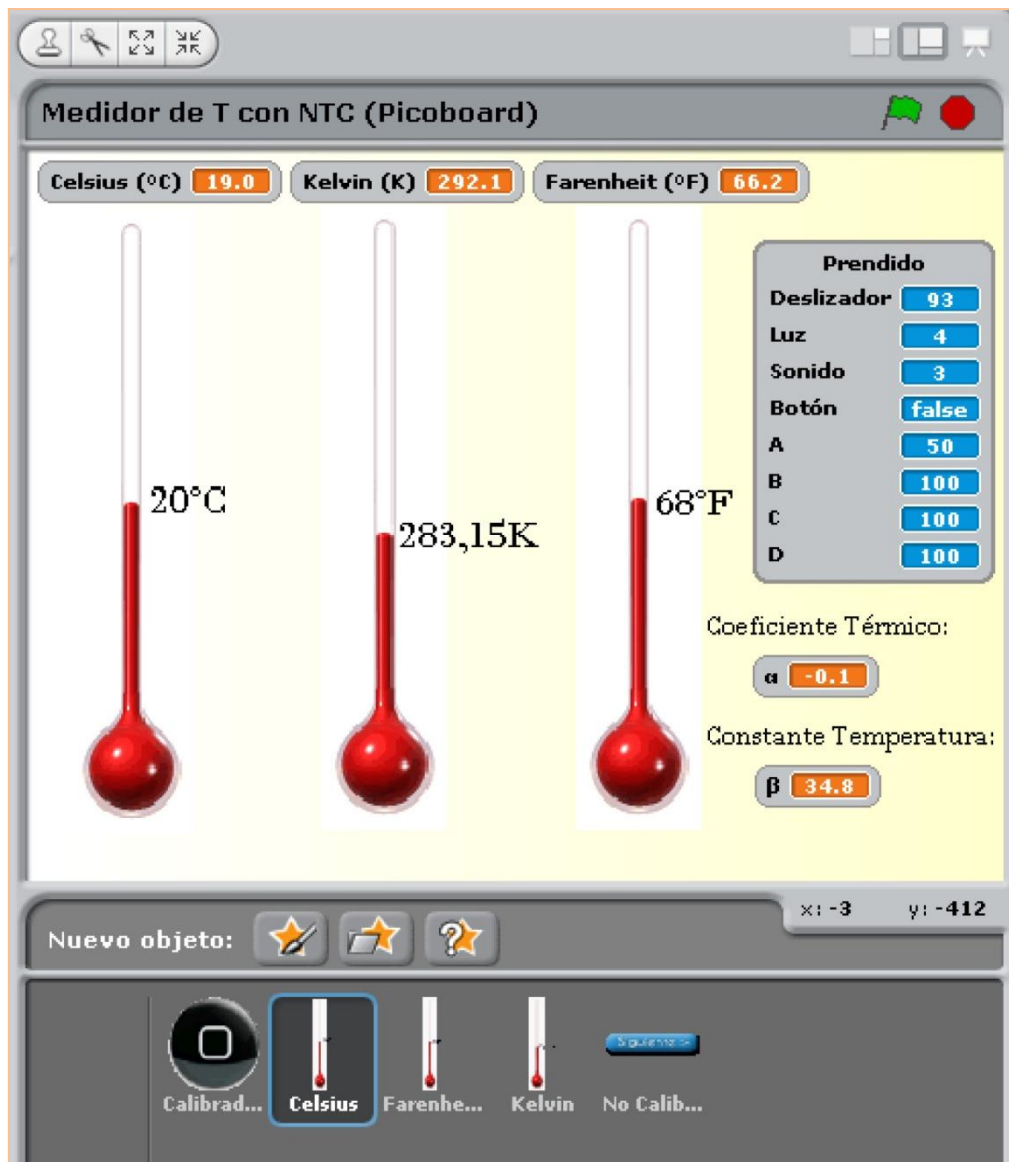
Necesitamos calcular las constantes  $R_0$  y  $\beta$ . Para ello introducimos en el programa dos puntos ( $T_1, R_1$ ) y ( $T_2, R_2$ ) que podemos conseguirlos de muchas formas (Por ejemplo, utilizando un termómetro y agua). Esta entrada de datos es lo que se conoce por “calibración” y gracias a ella obtenemos el valor de la constante de temperatura  $\beta$ . Conocido este factor, podemos calcular la  $T$  de un cuerpo que ponemos en contacto con una NTC. Las fórmulas que usaré y programaré son (despejadas de la anterior):

$$\beta = \frac{\ln \frac{R_1}{R_2}}{\frac{1}{T_1} - \frac{1}{T_2}} \quad T = \frac{\beta \cdot T_1}{\beta + T_1 \cdot \ln \frac{R_T}{R_1}}$$

Además, el coeficiente térmico es  $\alpha = \frac{-\beta}{T^2}$

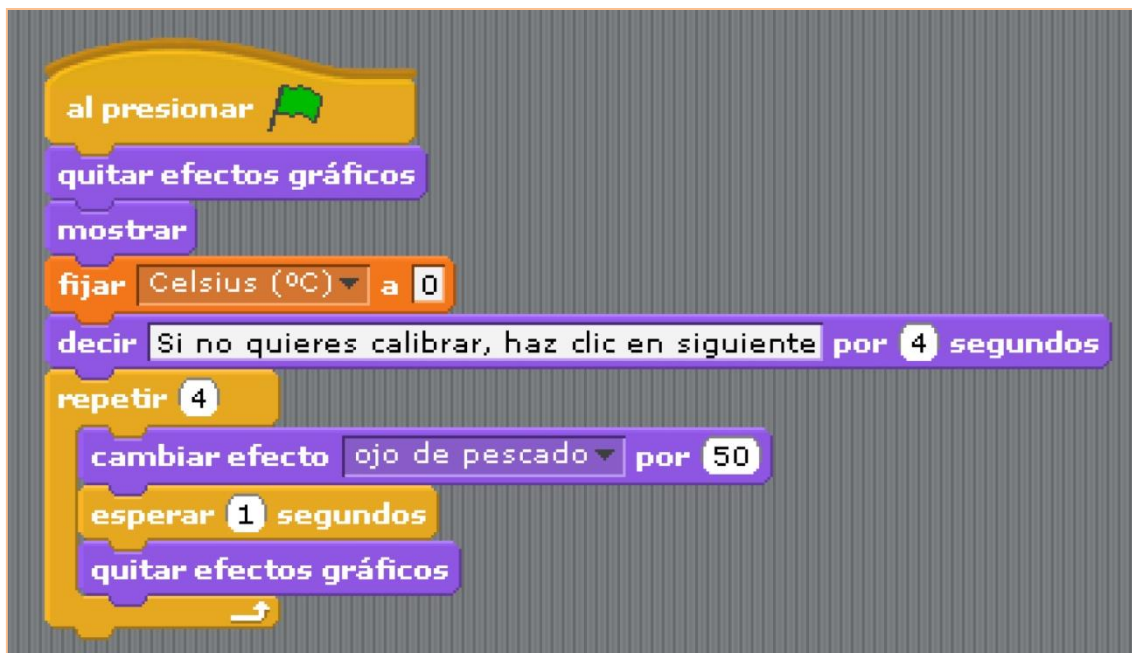
*Fórmulas para una NTC. Susana Oubiña Falcón. (CC BY)*

El programa presenta 5 objetos: el calibrador, el botón siguiente (por si no queremos calibrar), el termómetro en Celsius, el termómetro en Kelvin y el termómetro en Fahrenheit:



*Objetos en el programa. Susana Oubiña Falcón. (CC BY)*

**Calibrador:** El calibrador tiene forma de botón. Inicialmente se mostrará y realizará un bucle de 4 repeticiones aportando un efecto gráfico que haga que el usuario se de cuenta de que debe hacer clic en él:



*Script de efectos gráficos en el botón. Susana Oubiña Falcón. (CC BY)*

El script para realizar la calibración es el siguiente:

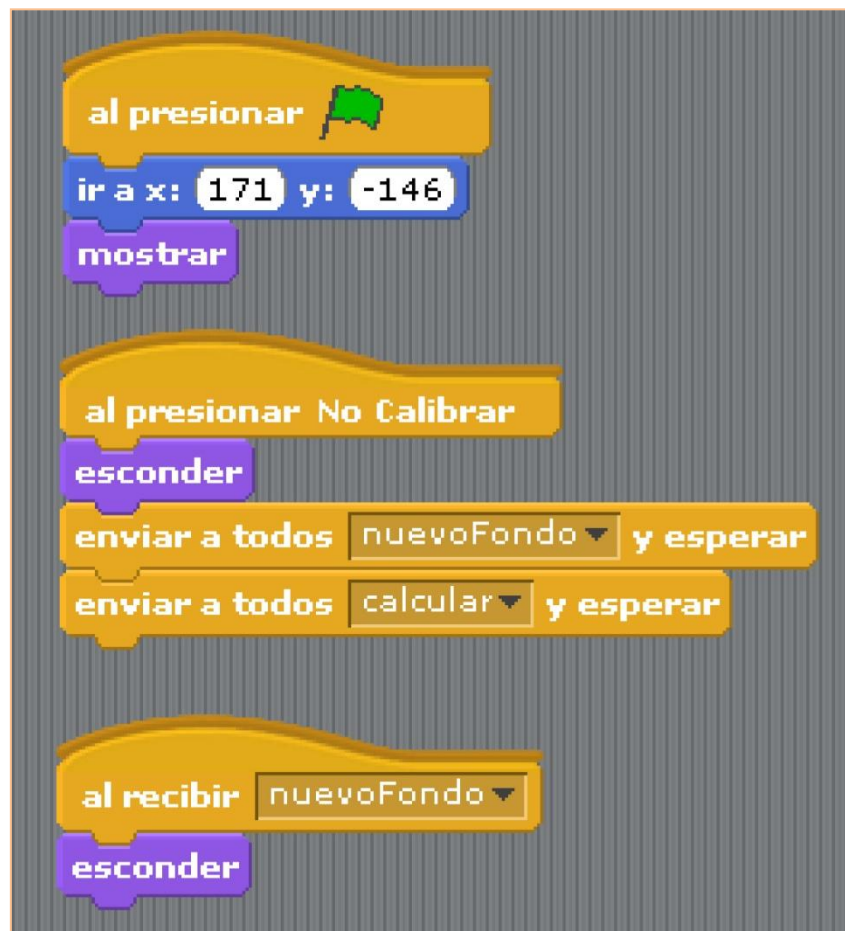


*Script para calibrar la NTC. Susana Oubiña Falcón. (CC BY)*

R1 es la resistencia que nos aporta el NTC a una temperatura fija T1 (idem para T2). Por lo tanto, necesitamos medir bien esas temperaturas y para ello necesitamos un termómetro.

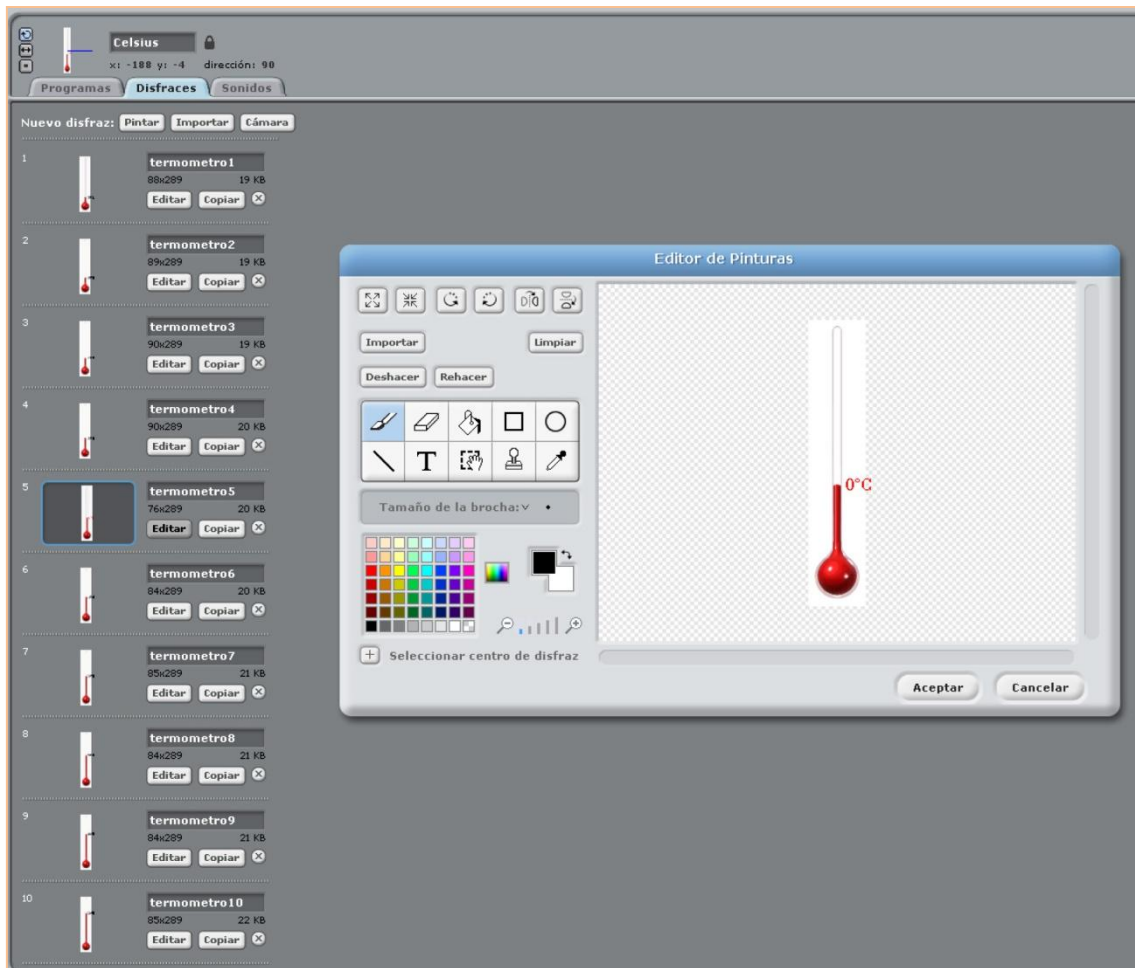


Botón No Calibrar: Es el botón “siguiente”. Inicialmente se muestra dando opción a no calibrar. Si se pulsa, se esconde, cambia el fondo y envía el mensaje “calcular” a los 3 objetos termómetros.



*Script para "Siguiendo". Susana Oubiña Falcón. (CC BY)*

Termómetro Celsius: Este objeto presenta 16 disfraces que se usarán para mostrar 16 temperaturas diferentes en grados Celsius, cubriendo temperaturas entre -40°C y 110°C. En la siguiente imagen se observa que el disfraz "termómetro5" muestra 0°C.



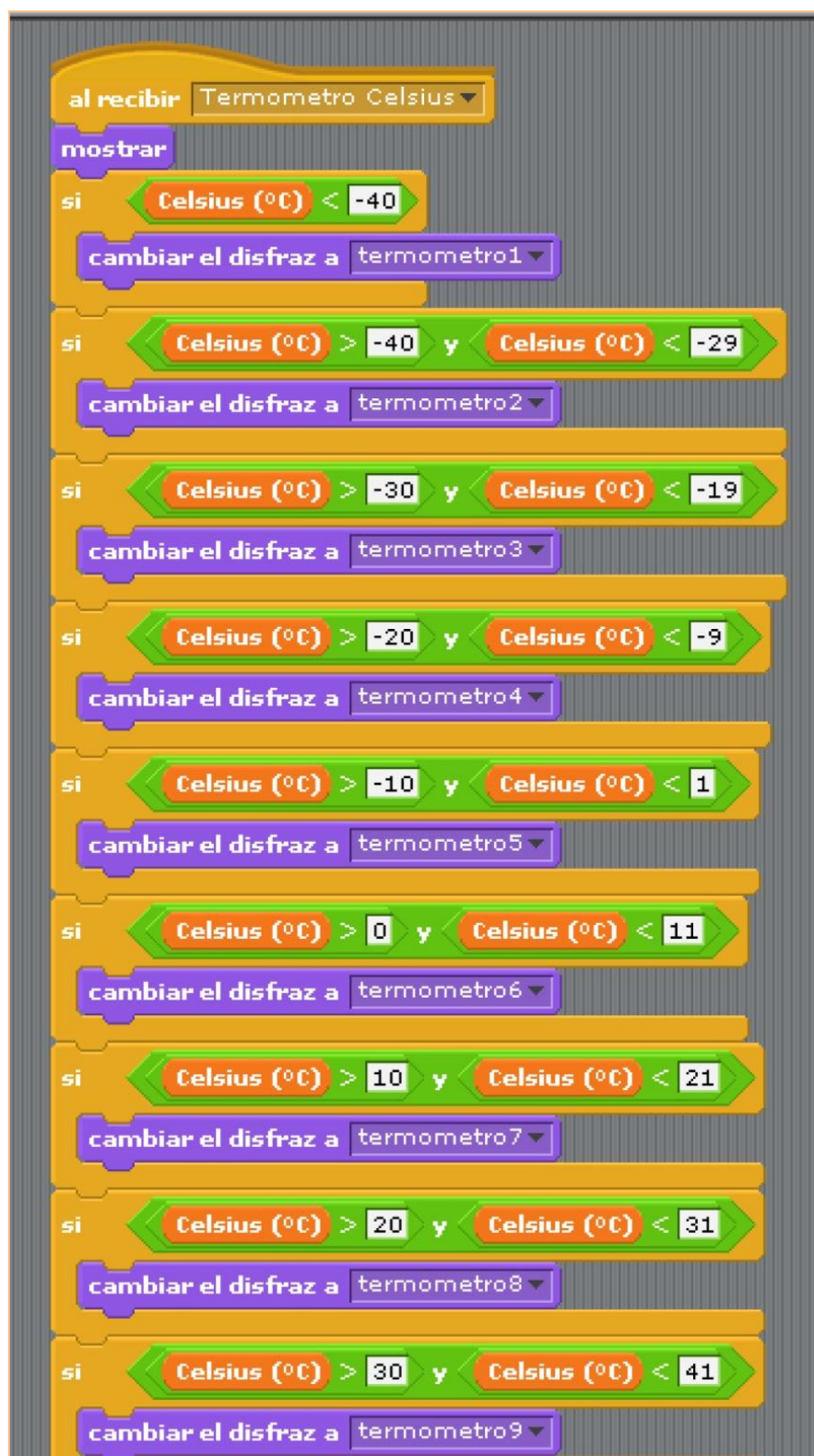
*Disfraces del objeto "Termómetro Celsius". Susana Oubiña Falcón. (CC BY)*

Inicialmente se esconde el objeto hasta que recibe el mensaje "calcular". Cuando le llega el mensaje de "calcular", realiza los cálculos de las variables  $\beta$ ,  $\alpha$  y la temperatura en grados centígrados. Sólo falta introducir el gráfico correspondiente a esa temperatura. Eso lo conseguimos enviando el mensaje "Termómetro Celsius":



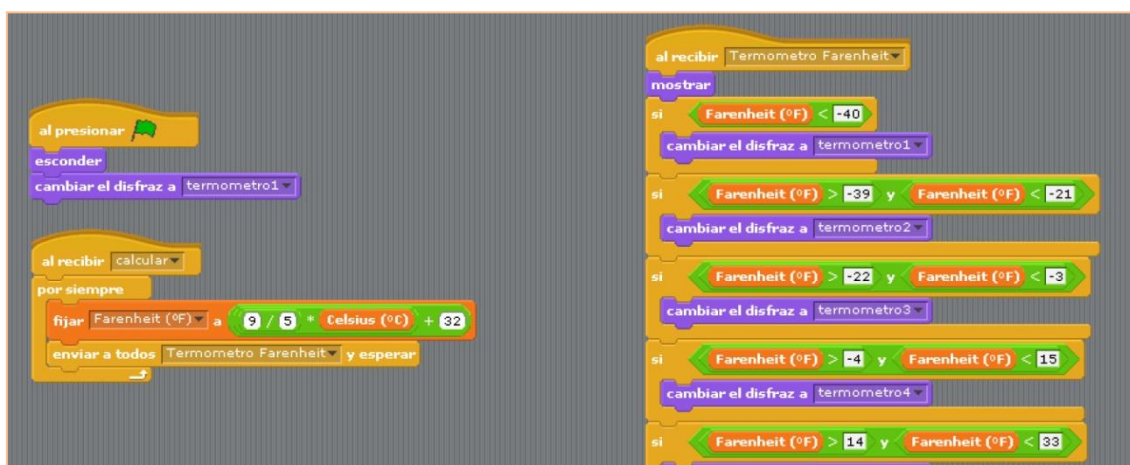
*Calculo de la Temperatura en °C". Susana Oubiña Falcón. (CC BY)*

Al recibir el mensaje “Termómetro Celsius”, se muestra el correspondiente disfraz de la temperatura calculada. Como hay 16 disfraces, este script se compone de 16 condicionales:

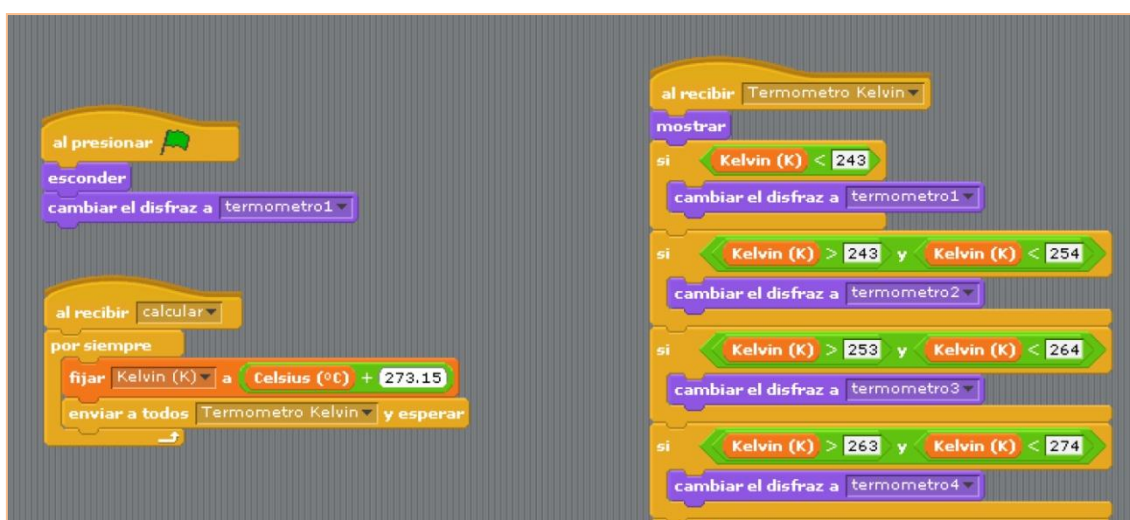


*Elección del disfraz con la T.* Susana Oubiña Falcón. (CC BY)

Los objetos Termómetro Fahrenheit y Termómetro Kelvin son similares al anterior: programamos su fórmula para calcular la temperatura y sus disfraces para esa T. Los scripts de ambos son los siguientes:



*Script para la temperatura en grados Fahrenheit. Susana Oubiña Falcón. (CC BY)*



*Script para la temperatura en grados Kelvin. Susana Oubiña Falcón. (CC BY)*

### 3.1.2.3. Dispositivo Leap Motion

*Leap Motion* es un dispositivo que nos aporta una tecnología que puede ser interesante no sólo por ella misma, sino también porque se puede combinar con otros softwares como el programa scratch 2.0, aportando motivación al alumnado. Este dispositivo sensor con tres infrarrojos (ver imagen) es capaz de detectar o captar el movimiento de nuestras manos y de cada uno de sus dedos en un espacio cúbico de 20cm de lado (20x20x20cm). Por lo tanto, nos permitirá controlar el ordenador (en nuestro caso, nuestro programa creado con scratch 2.0) por medio de gestos o movimientos que realizamos en el aire, ya

sea por medio de los dedos o de las manos desde la muñeca, añadiendo una nueva dimensión: la profundidad (variable z de un sistema XYX).

Considero que es un dispositivo que fomenta la motivación de nuestro alumnado, un alumnado muy acostumbrado a interactuar con juegos. Los usuarios tendrán la sensación de estar en esa típica escena de ciencia ficción donde los protagonistas se comunican con el ordenador haciendo movimientos en el aire.

La funcionalidad que adquiere con el scratch es la posibilidad de controlar los objetos que hemos creado para los programas, de tal forma que su movimiento lo realice el propio jugador interactuando con sus dedos y manos pero no desde un teclado tradicional o desde una consola externa creada, por ejemplo, con el kit Makey Makey. En un juego, esto aporta grandeza y enganche, consiguiendo que el jugador se sienta más partícipe del mismo.

En la imagen se observa este dispositivo rectangular y pequeño, que viene acompañado de dos cables de diferente longitud. Un extremo se conecta al Leap Motion y el otro extremo finaliza en un terminal USB que se unirá al ordenador:



*Leap Motion. Susana Oubiña Falcón. (CC BY)*

### **A. Conectarse a Leap Motion**

Leap Motion necesita, para funcionar, de una aplicación que se instala en el ordenador (Leap Motion Airspace), con los drivers y la plataforma «Airspace» (que nos permite ejecutar las aplicaciones que vayamos adquiriendo). Es muy sencillo de conectar ya que sólo hemos de enlazarlo al ordenador por medio de un cable USB e instalar su driver. Driver que se puede descargar del propio software de la plataforma, «Airspace Home».

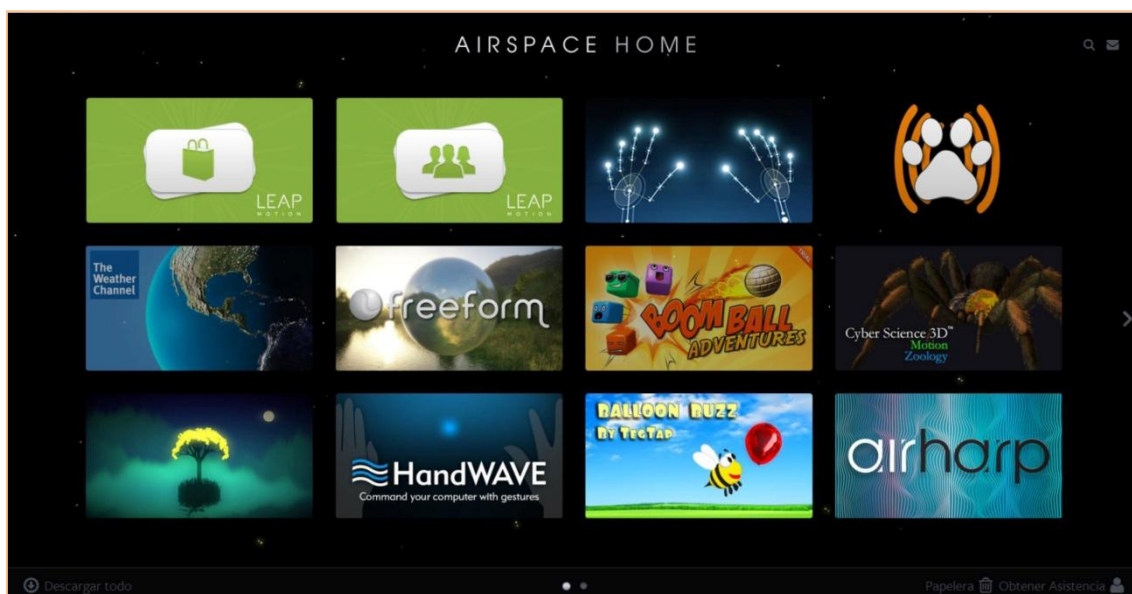




*Icono app Airspace para Windows. Susana Oubiña Falcón. (CC BY)*

Los pasos a seguir son los siguientes:

1. Descargamos, dentro del “Airspace Home,” el plugin para scratch.



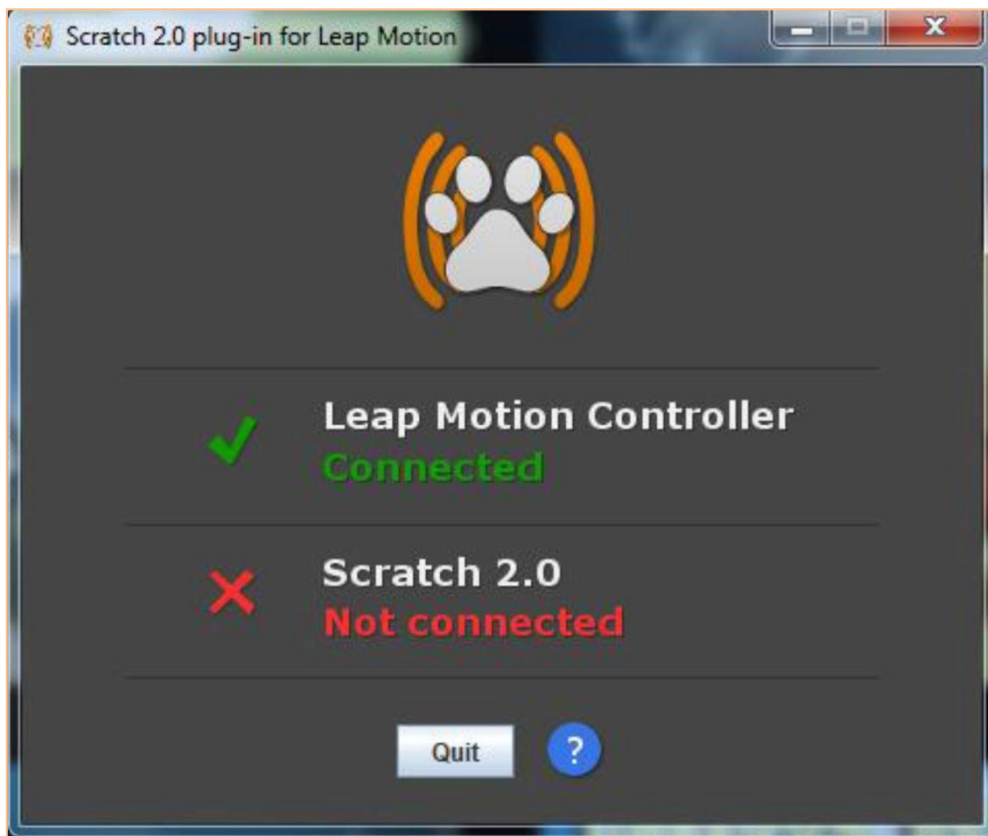
*Plataforma Airspace Home. Susana Oubiña Falcón. (CC BY)*



*Plugin para Scratch 2.0. Susana Oubiña Falcón. (CC BY)*

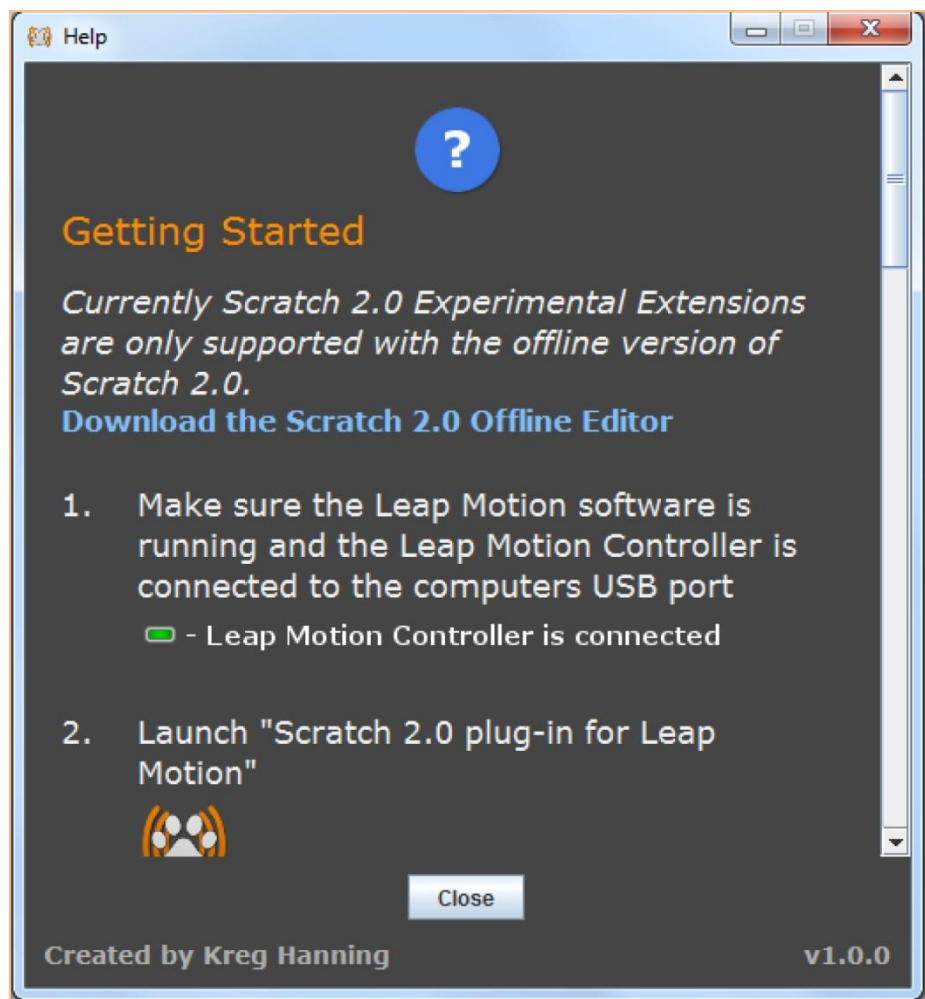


2. Al abrirlo, la aplicación detecta que el *leap motion* está conectado



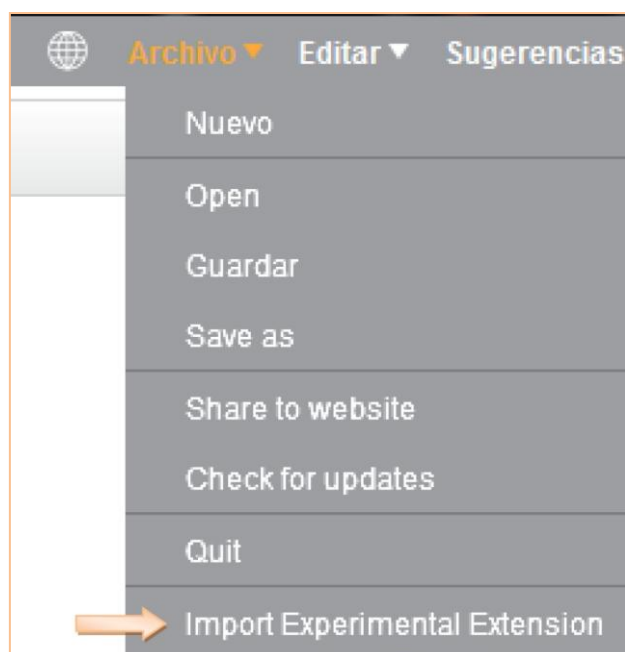
*Conexión del Controlador de Leap Motion.* Susana Oubiña Falcón. (CC BY)

3. En la figura anterior se observa que aun no está conectado con el scratch 2.0. Esta conexión requiere de la instalación de un fichero que se llama *LeapMotion.json*. La misma aplicación nos muestra de donde descargarlo al hacer clic en el símbolo de interrogación en azul (ver siguiente imagen):



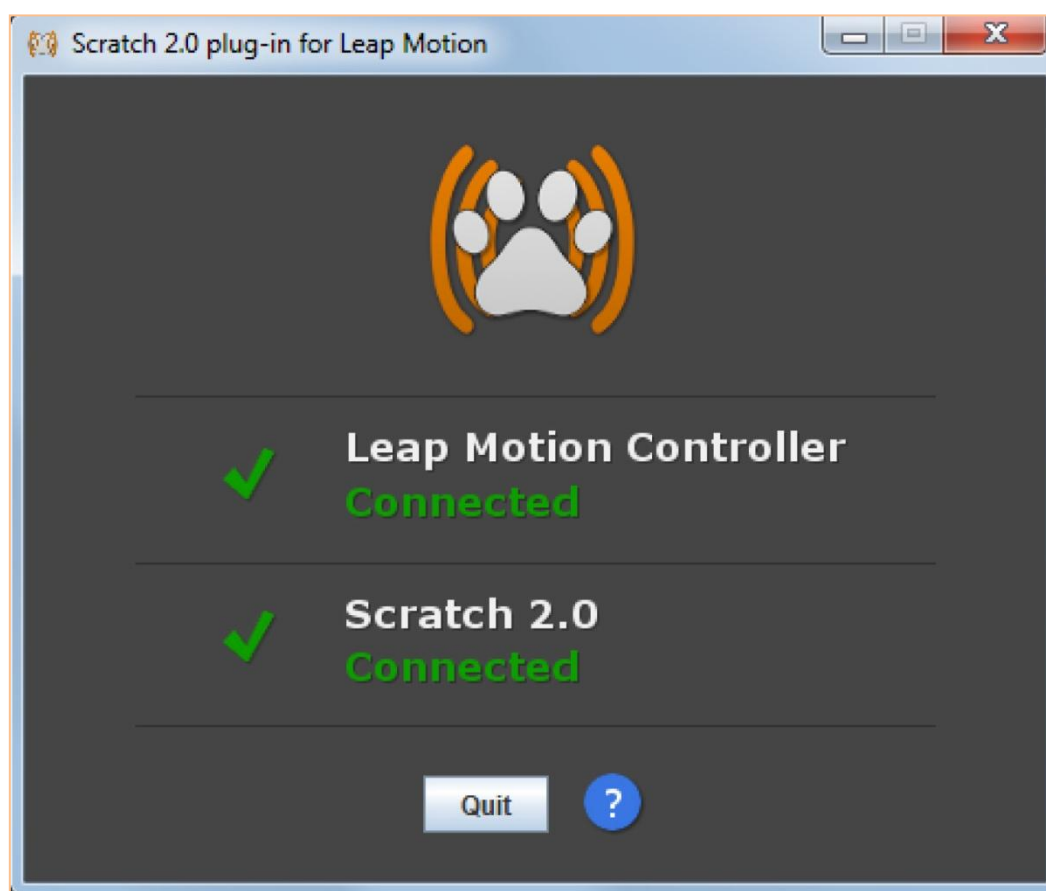
*Pasos a seguir para la instalación del Leap Motion. Susana Oubiña Falcón. (CC BY)*

4. A continuación, abrimos el programa scratch 2.0 (online o no) y hacemos, simultáneamente, Shift (mayúsculas)+clic en Archivo. Se nos despliega un menú con una opción que se llama: “importar extensiones experimentales” “Importar extensión HTTP experimental”, según sea o no la versión online. Al hacer clic sobre ella se nos abre una ventana que nos permite cargar el archivo LeapMotion.json del paso 3.



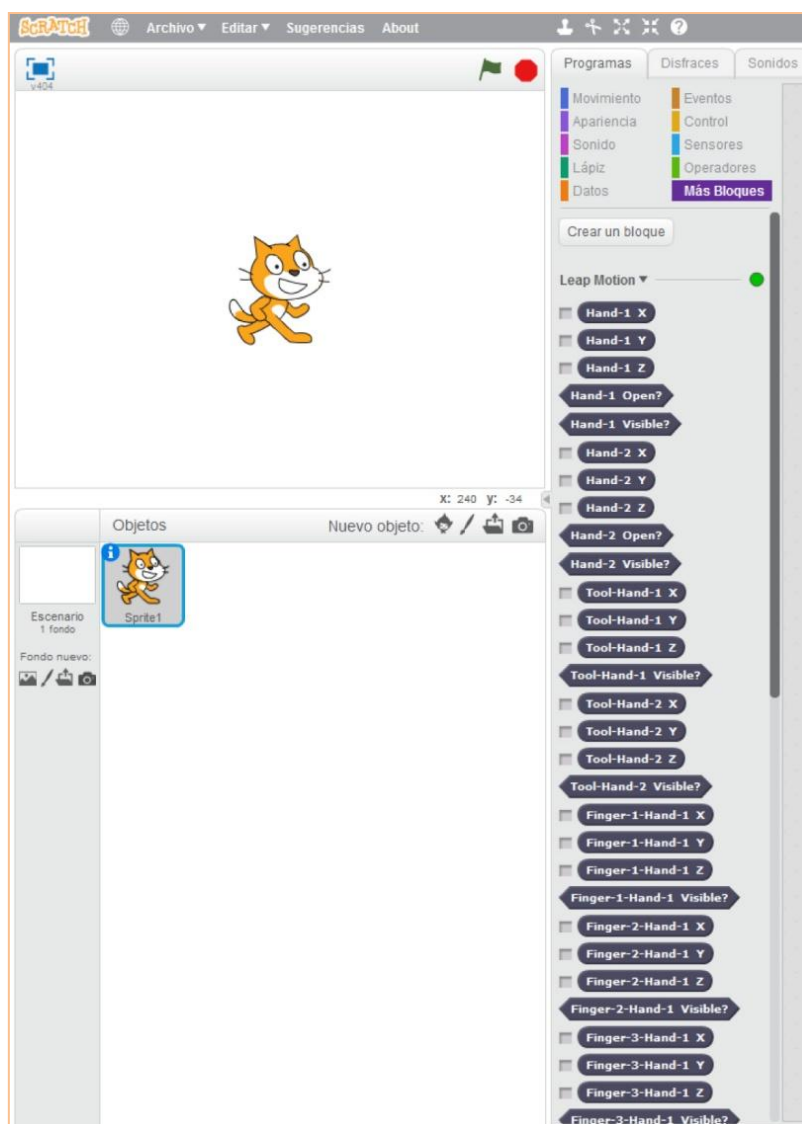
*“Importar conexiones experimentales”.* Susana Oubiña Falcón. (CC BY)

5. Si ahora volvemos al “Airspace” (plugin del scratch 2.0) nos detecta que el scratch está conectado.



*Scratch 2.0 conectado al plugin.* Susana Oubiña Falcón. (CC BY)

En la sección de **Más Bloques** del scratch 2.0 veremos que se han añadido más comandos (se verán en color negro). Son los comandos del leap motion. (El punto verde indica que el leap motion está conectado)



*Comandos del Leap Motion en Scratch 2.0. Susana Oubiña Falcón. (CC BY)*

## B. Ejemplos

Para ejemplificar el uso del Leap Motion en el scratch 2.0, usaremos 2 juegos:

- ✓ Ejemplo1: *Juego del comecocos*

En mi primera prueba he creado un programa que llamo "*Juego del comecocos con Leap Motion*" y que muestra un posible uso del dispositivo Leap Motion con el scratch 2.0. Es un programa sencillo donde el movimiento del comecocos es controlado por el jugador utilizando su **mano derecha** sobre el espacio cúbico que abre los infrarrojos del dispositivo Leap Motion. El jugador comienza el

juego con 0 puntos y 3 vidas y gana si consigue comer 50 bolitas, las cuales se van clonando cada 4 segundos y apareciendo en lugares aleatorios del escenario (dentro de unas dimensiones), y perderá, si no consigue hacer 50 puntos y es tocado 3 veces por una bola, llamémosla "mala", que se mueve constantemente por el escenario, perdiendo en cada toque una vida.

La explicación del programa, así como, una demostración de cómo se jugaría se describe en el siguiente vídeo:

*"Juego del comecocos con Leap Motion"* .Susana Oubiña Falcón. (Licencia Vimeo CC BY)

#### ✓ Ejemplo 2: *Balloom Buzz*

El juego "Balloom Buzz" es una aplicación (a modo de juego) que fue diseñada para usar con el Leap Motion. Por lo tanto, es un juego que fue programado en un lenguaje de programación determinado y adaptado a la estructura de control y detección de gestos que ofrece el SDK del dispositivo. En fin, demasiado complejo.

El protagonista del juego es una abeja, la cual se mueve siguiendo el dedo índice de la mano derecha del jugador. La abeja debe explotar globos y cada uno le sumará un punto. Sólo tiene 3 vidas, perdiendo una de ellas cada vez que deja llegar un globo a la parte superior de la pantalla (sin explotarlo). En este juego hay varios niveles de modo que por cada 10 globos explotados, estos suben a mayor velocidad (siguiente nivel).

Basándonos en el juego original, he creado mi versión del mismo para el scratch 2.0. Balloom Buzz es un juego para niños a partir de 3 años. Curiosamente, también es un juego que gusta a niños no tan pequeños, teniendo mucho éxito en alumnos/as de la ESO e incluso bachillerato en el curso pasado 2013/2014.

He creado un programa con Scratch 2.0 que simula ese juego, de modo que, para jugar, debemos disponer del dispositivo Leap Motion. Obviamente, la historia del juego es la misma y la abeja debe explotar los globos. En mi juego, sólo he creado 3 niveles pero es muy sencillo introducir más y más, tantos como se desee.

El juego presenta los siguientes elementos:

Objetos: Abeja (jugador), tres elementos para cada fallo (3 corazones), 4 tipos de globos de diferentes colores, tres niveles representados por las estrellas numeradas con los números 10, 20 y 30, la barrera de fallos azul, el elemento de comienzo del juego (start) y el elemento de felicitación cuando se superan los tres niveles.



*Objetos del juego Balloom Buzz. Susana Oubiña Falcón. (CC BY)*

Variables: Fallos, Puntos y Velocidad.

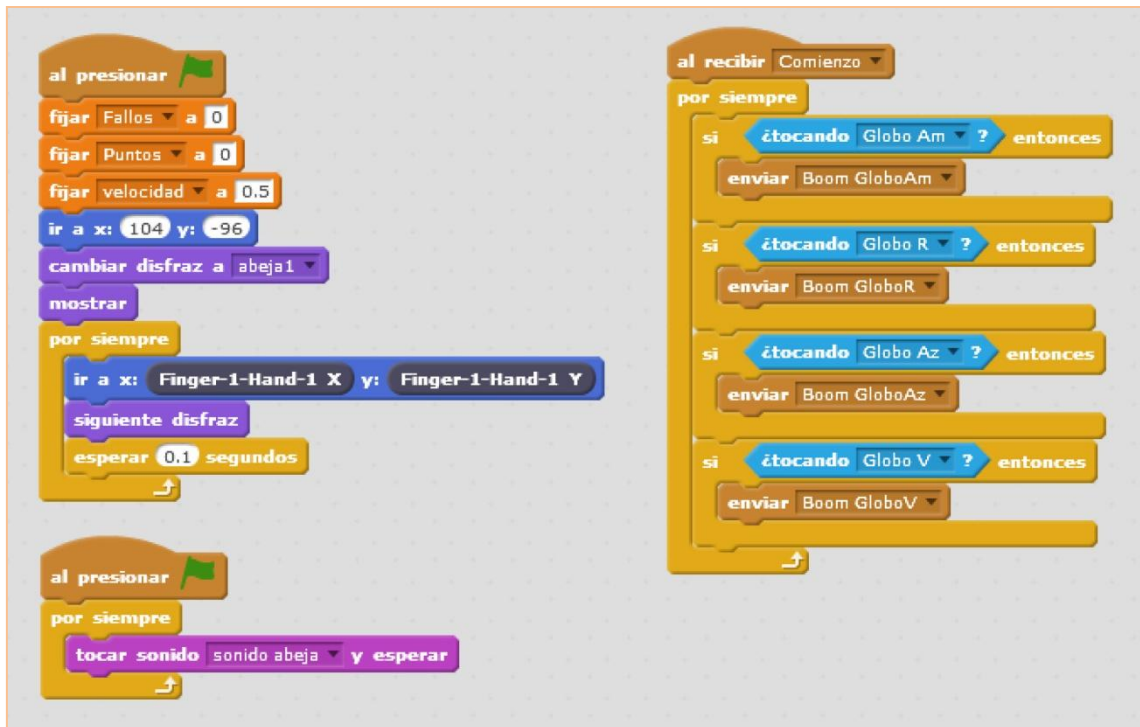
Escenarios: Inicio, juego y Game Over (este último sale si se pierde)

Los scripts son los siguientes:

Abeja: Se inicializan las variables puntos y fallos a cero y la velocidad a 0.5 (esta irá más tarde aumentando). Se sitúa en un lugar concreto del escenario y comienza con el primer disfraz. Un bucle continuo Por Siempre hace que siga al dedo índice (dispositivo Leap Motion) cambiando continuamente de disfraz y haciendo sonar el sonido abeja.

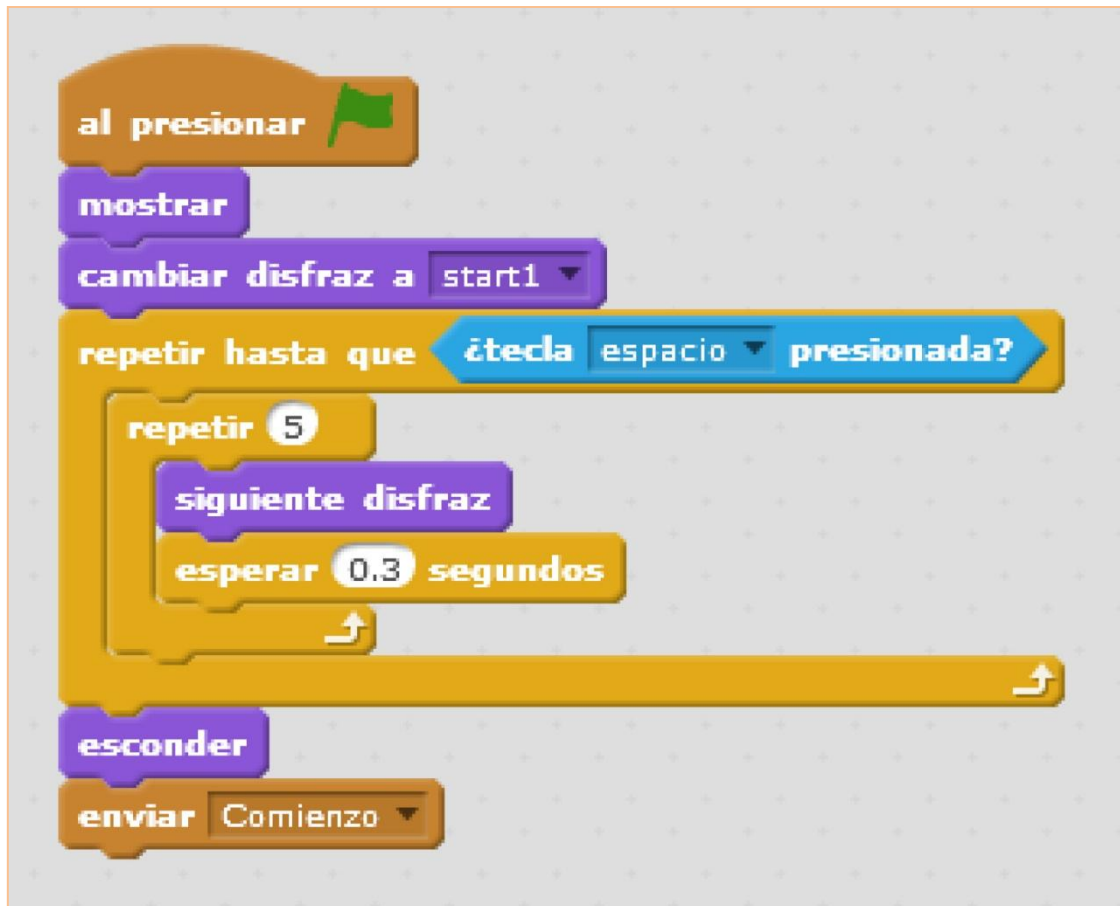
Al hacer clic en la tecla "Space" (espacio) se envía el mensaje "Comienzo". Este mensaje lo reciben los globos y también la abeja. La abeja se programa para que realice acciones si toca los diferentes colores de globos, enviando en cada caso un mensaje "Boom GloboCOLOR" que es recibido por cada globo.





Script del objeto abeja. Susana Oubiña Falcón. (CC BY)

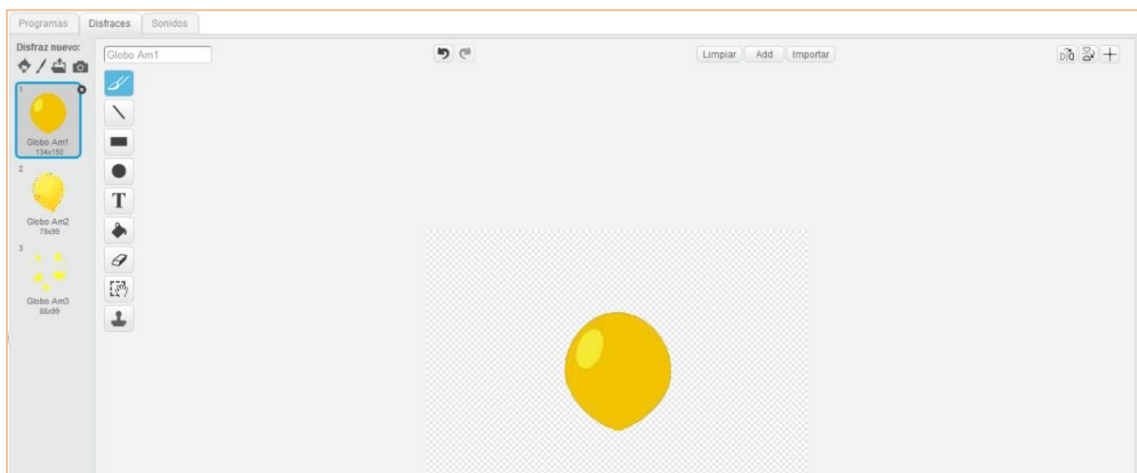
Start: El objeto start presenta 4 disfraces que van cambiando (repetición 5 veces) hasta que se presiona la tecla “espacio”. Cuando se presione esa tecla, se esconde y envía el mensaje “Comienzo” que es el que activa el juego, llegando a la abeja y a los globos.



*Script del objeto start. Susana Oubiña Falcón. (CC BY)*

### Globos: Ejemplifico el color amarillo

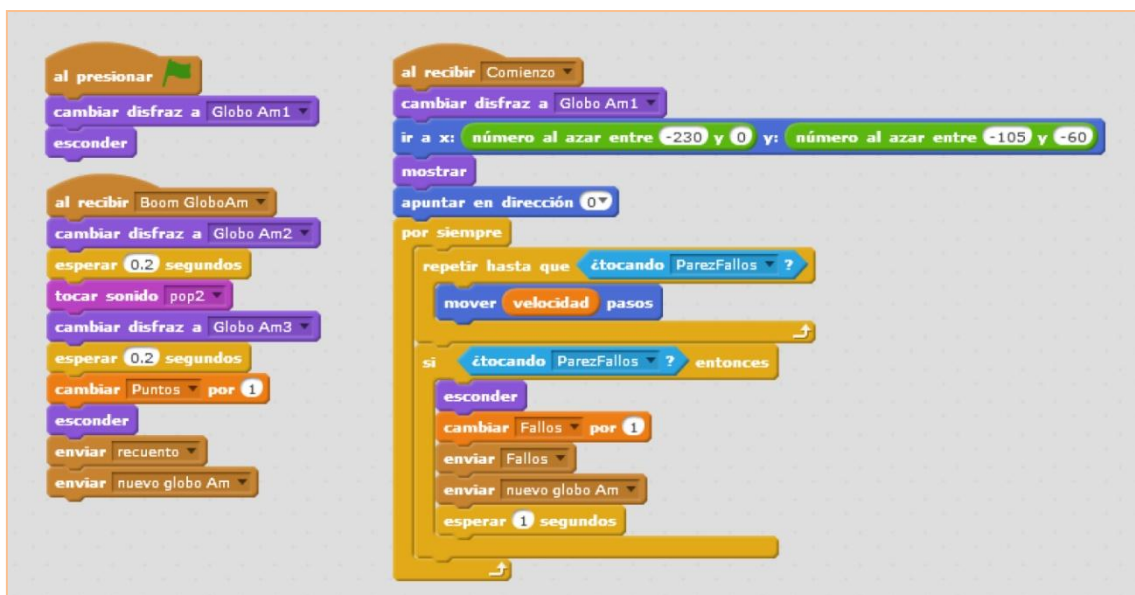
Cada objeto globo presenta tres disfraces (Globo Am1, Globo Am2 y Globo Am3) para representar el globo volando, el globo tocado y comenzando a desvanecerse y el globo en explosión.



*Disfraces del objeto "Globo". Susana Oubiña Falcón. (CC BY)*

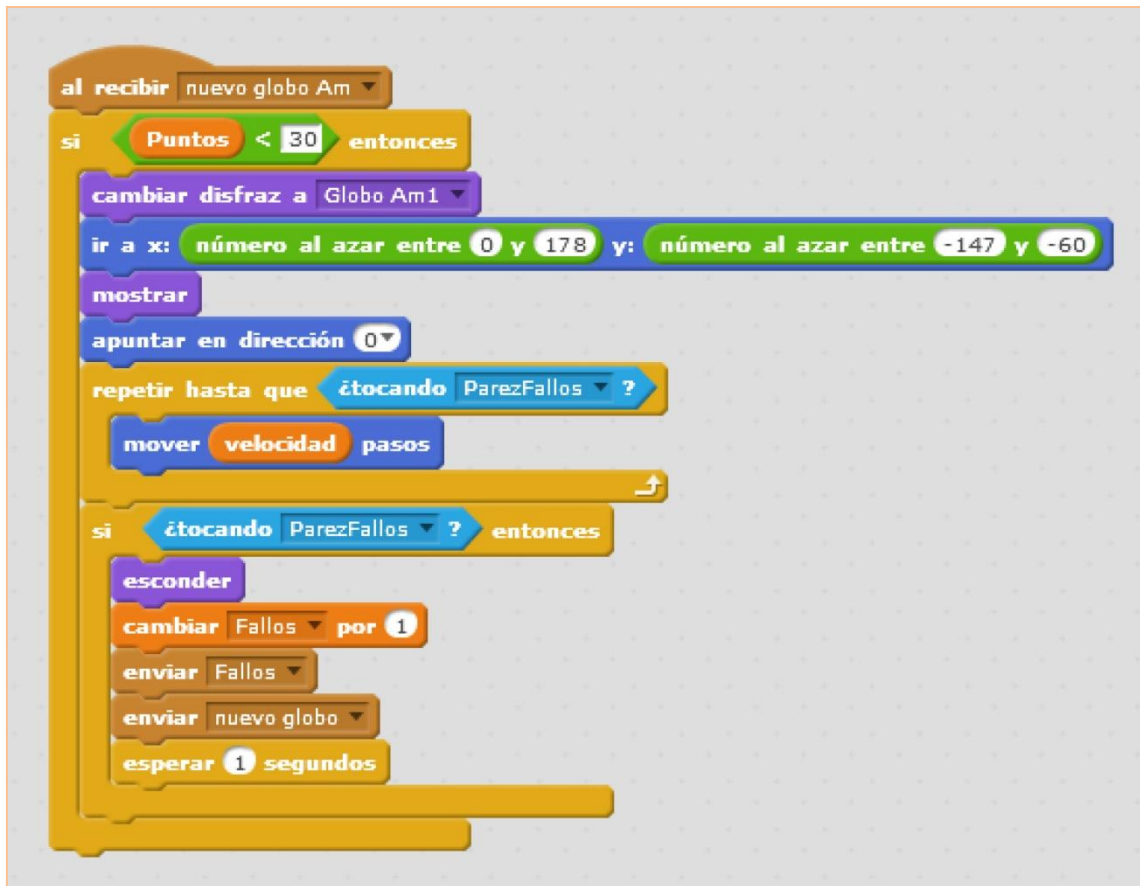
Inicialmente, el globo está oculto y se muestra con el primer disfraz cuando recibe el mensaje “Comienzo”. (Para que no aparezcan los 4 globos a la vez se ha hecho que los siguientes globos salgan con 4 segundos de diferencia de uno a otro. También, se ha querido que dos aparezcan en la parte del escenario correspondiente al eje X positivo y los otros dos colores a la parte del eje X negativo). En el caso del globo amarillo, aparecerá en el eje X negativo, tal y como se muestra en la siguiente imagen. El globo debe subir y por eso debe apuntar hacia arriba y se moverá según el valor de pasos de la variable “Velocidad” (que inicialmente vale 0.5) subiendo hasta que toque el objeto “ParedFallos”. Cuando toque el objeto “ParedFallos” el globo se esconderá, sumará 1 fallo y envía los mensajes “Fallos” (que son recogidos por los objetos Fallo (1, 2 y 3) y “nuevo globo Am” (que crea otro globo). El objeto “ParedFallos” no tiene script ya que sólo nos vale para detectar la posición de otros objetos en él.

El mensaje “Boom Globo Am” simula la explosión del globo al cambiar los disfraces, suma un punto al estar el globo explotado, simula un sonido de explosión y envía los mensajes “recuento” (se fija si se ha conseguido explotar 30 globos) y “nuevo globo Am”.



1ª Parte del Script del objeto globo Amarillo. Susana Oubiña Falcón. (CC BY)

El mensaje “nuevo globo Am” sólo debe ejecutarse si no se superan los 30 globos, ya que en ese caso, no tendría sentido que siguieran creándose globos. Para darle más suspense al juego, como el globo original aparecía en el eje X negativo, entonces, el nuevo globo aparecerá en el lado contrario. Como puede verse en la siguiente imagen, el script de movimiento es idéntico al del mensaje “Comienzo”.



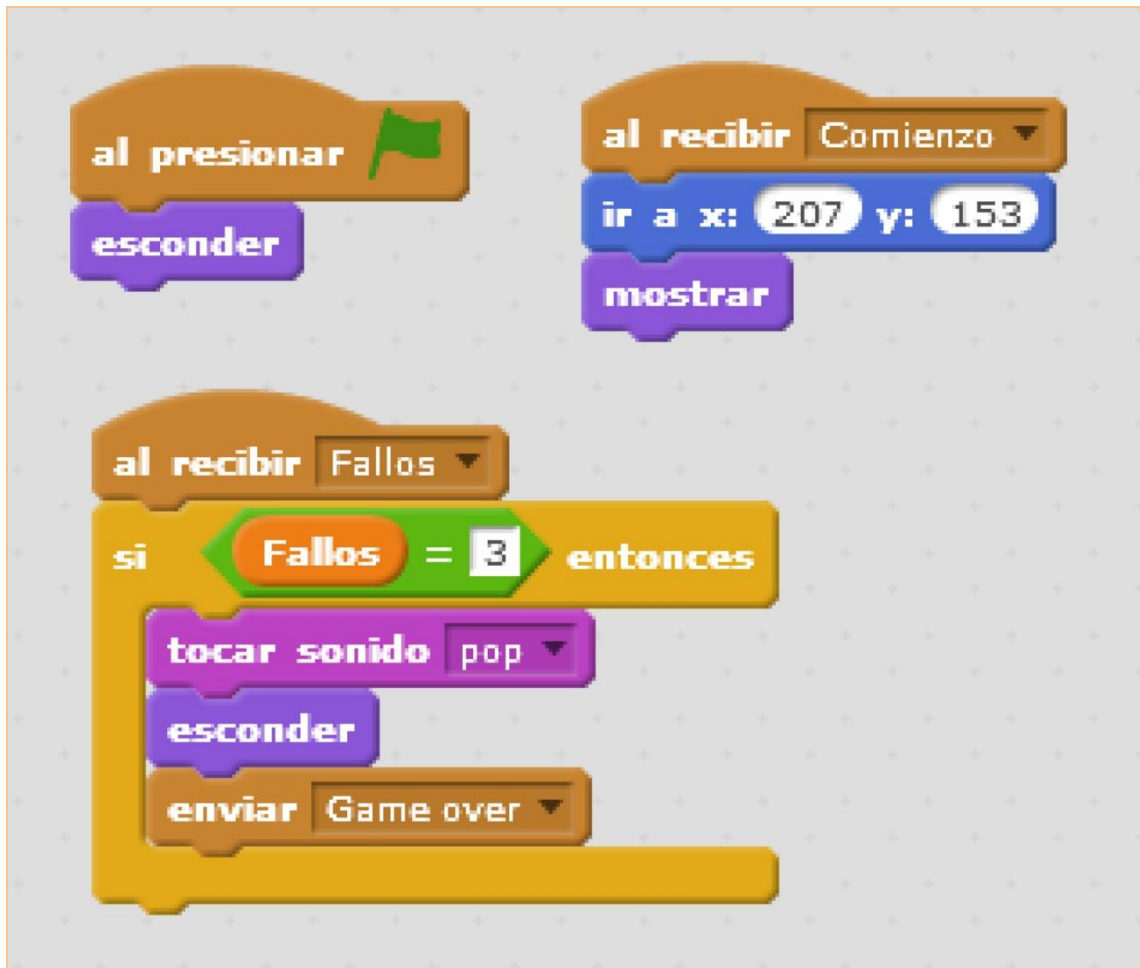
2ª Parte del Script del objeto globo Amarillo. Susana Oubiña Falcón. (CC BY)

Fallos 1, 2 y 3: Los scripts de Fallo1 y Fallo2 son similares (cambiar el número de fallos y mostrarse en el lugar correspondiente del escenario). Inicialmente se esconden y deben aparecer al comenzar el juego (mensaje “Comienzo”) para ir desapareciendo paulatinamente en el recuento de fallos (si hay 1 fallo, 2 fallos o 3 fallos).



*Script del objeto Fallo1. Susana Oubiña Falcón. (CC BY)*

A mayores, el objeto Fallo3 debe ser el detonante de que finalice el juego y por eso envía el mensaje “Game over”:



*Script del objeto Fallo3. Susana Oubiña Falcón. (CC BY)*

Premio 1, 2 y 3: Los objetos premio se refieren a los niveles. El juego está creado de modo que por cada 10 globos explotados se cambie el nivel. Superar un nivel implica que los siguientes globos se moverán con una velocidad aumentada en 0.5. Actúan cuando reciben el mensaje “Recuento”, mostrando un efecto ampliado de tamaño y desapareciendo, tal y como muestra la siguiente imagen del objeto “Premio1” (el objeto Premio2 cambia los puntos por 20):





Script del objeto Premio1. Susana Oubiña Falcón. (CC BY)

En el objeto Premio3, por ser el último nivel que supera, finaliza el juego y envía el mensaje “Felicidades” que hará que el programa se pare con el objeto Felicidades.



Script del objeto Premio3. Susana Oubiña Falcón. (CC BY)

Felicidades: Este objeto se activa cuando el jugador consigue 30 puntos. Se muestra en un lugar, toca un sonido y para el programa: has ganado!



Script del objeto Felicidades. Susana Oubiña Falcón. (CC BY)

### 3.1.2.4. Robot WeDo de LEGO

La robótica es una muy **buena herramienta** para explicar contenidos del nuevo Currículo de Ciencias Naturales de Primaria. Por ejemplo, los contenidos de "Máquinas y aparatos" o "Análisis de Estructuras", entre otros.

Hay un kit de LEGO llamado WeDo que está genial para ese fin. Este kit puede programarse a través de un software específico de LEGO o con el lenguaje de programación scratch.

- ✓ Software de LEGO: Utiliza bloques que se unen dando lugar al programa para el elemento diseñado. Es muy sencillo de usar y de fácil comprensión incluyendo proyectos completos diseñados desde cero (construcción paso a paso y programación), a modo de guía para nuestro alumnado.
- ✓ Scratch: hasta hace poco sólo funcionaba correctamente con la versión Scratch 1.4. La versión actual del Scratch, Scratch2.0, que está genial para crear videojuegos, y que en teoría ya presenta la extensión para la Picoboard y para WeDo, a veces falla y no se conecta con el WeDo. Sólo está operativa para Scratch2.0 en la versión online.

En la página de scratch se puede descargar un pugin que, tras instalarse, hace que la nueva versión 2.0 logre trabajar con el robot WeDo (y con la Picoboard). Por ahora, en agosto, sólo funciona para la versión online. Es decir, aún no ofrece el control sin bugs del motor del WeDo, ni de sus sensores de movimiento y de inclinación. Esperemos que pronto Scratch 2.0 SI SEA COMPATIBLE con WeDo en cualquier versión.

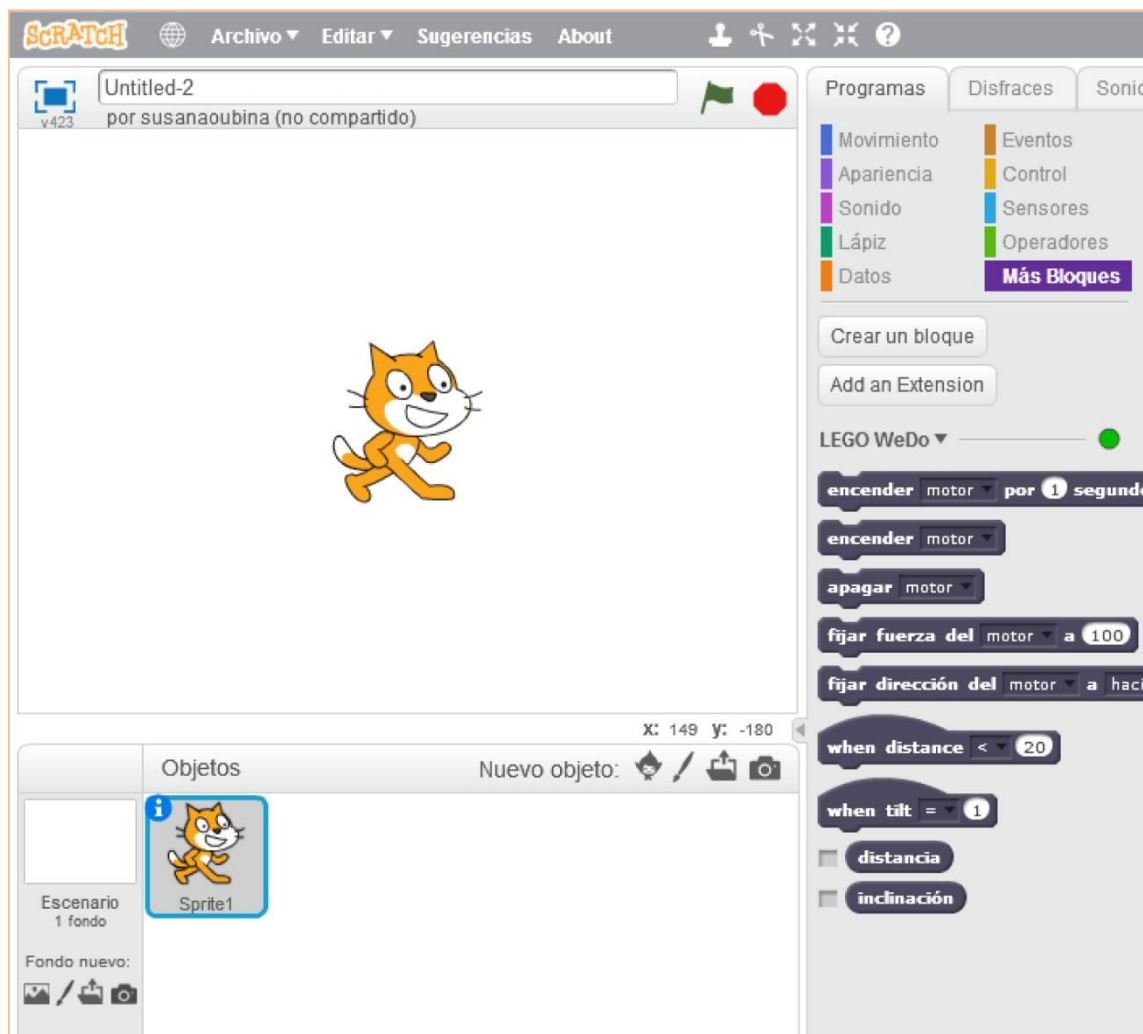
Seguramente esta actualización sin bugs estará disponible en breve y dará soporte a WeDo y a la ScratchBoard (Tarjeta de sensores). También mOway (otro robot genial que es más apto para alumnos/as de Secundaria) tiene pensado crear bloques de extensión para Scratch 2.0.

#### A. Conectarse con scratch 2.0

Pese a que la versión scratch 2.0 sólo trabaja con WeDo en versión online, será la que ahora explicaremos ya que se usará en un futuro. Por ello, es importante conocer como realizar esta conexión y como trabajar en ella. Los pasos a seguir para conectar el WeDo al scratch son los siguientes:

1. Conectar el WeDo (USB)
2. Entrar en la versión online del scratch.
3. Ir al bloque "Más Bloques" y hacer clic en "Add an Extensión" (Nos aparecerá una librería con 2 extensiones, WeDo y PicoBoard).

4. Hacer clic en WeDo y pulsar OK. Nos aparecerán los comandos del WeDo diciéndonos que el Lego WeDo no está conectado (color rojo). Para hacerlo, en el lado derecho nos muestra el enlace para descargar el plugin (ScratchDevicePlugin.msi) necesario para Windows o MAC.
5. Hacer clic en el plugin y ejecutarlo.
6. Recargar la página del editor del scratch
7. Repetir el paso 3
8. Hacer clic en WeDo y pulsar OK. Nos aparecerán los comandos del WeDo pero ahora, nos dirá con un círculo verde que el WeDo está conectado, tal y como se muestra en la siguiente imagen:



*WeDo conectado a la versión scratch2.0 online. Susana Oubiña Falcón. (CC BY)*

Los nuevos comandos que incorpora scratch para controlar el motor y los sensores de movimiento y de inclinación del WeDo así como, el sensor de luz (este último no pertenece al kit del WeDo), son:



*Comandos para usar con WeDo. Susana Oubiña Falcón. (CC BY)*

## **B. Usar WeDo con el software de LEGO**

Antes de explicar cómo podemos usar WeDo con Scratch muestro unos ejemplos (vídeos con calidad de móvil) de programación del WeDo con su propio software:

- ✓ *Pájaros Bailarines*: Se ha programado para que, al pulsar la tecla "i", los pájaros dancen girando hacia la izquierda (repito el bucle 3 veces) y si se pulsa la tecla "d", los pájaros danzan hacia la derecha.

["Pájaros bailarines con WeDo"](#) . Susana Oubiña Falcón. (CC BY)

- ✓ *Caimán Hambriento*: Se ha programado para que, mientras tenga la boca abierta, Si detecta un objeto, inmediatamente la cierra, hace un sonido simulando que mastica y retorna su posición abriendo la boca.

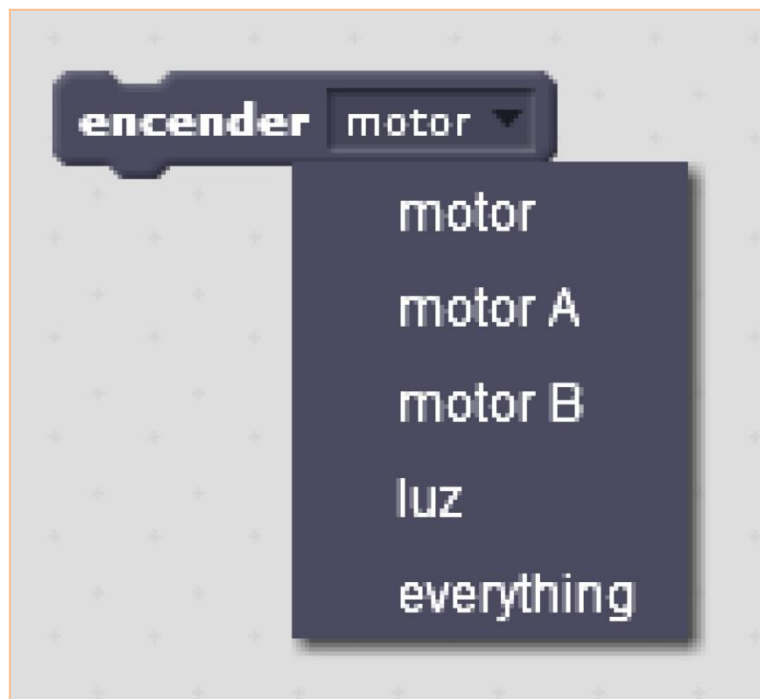
["Caimán hambriendo con WeDo \(Sensor de movimiento\)"](#) . Susana Oubiña Falcón. (CC BY)

A continuación, vamos a centrarnos en programar el WeDo con el scratch. Para ello utilizaremos la versión del scartch2.0 (online) y 1.4. Esta última es muy similar a la que ya conocéis y con la que ya hemos estado trabajando hasta el momento. La versión 1.4 permite conectar un motor pero la nueva versión 2.0 nos da opción a controlar 2 así como, a introducir un elemento de luz (diodos LED).

### C. Usar WeDO con Scratch 2.0 (explicación y pequeños ejemplos)

El kit WeDo incluye un motor y 2 sensores, uno de movimiento y otro de inclinación. A mayores, se puede conectar un sensor de luz que dispone de 2 diodos. Programar el WeDo implica controlar esos elementos y conocer qué pueden hacer. Veamos cada uno por separado:

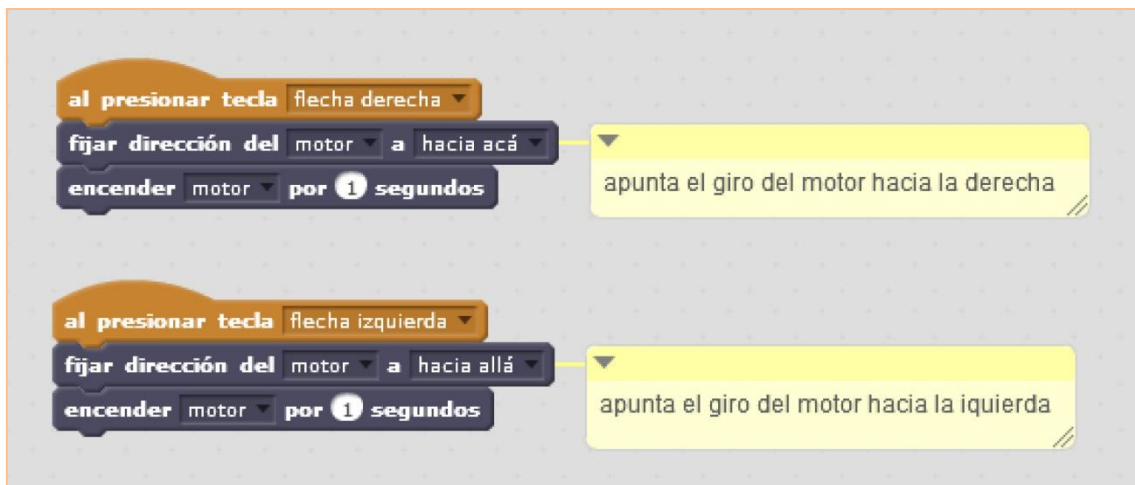
Control del motor: Ahora, el scratch 2.0 puede controlar hasta 2 motores (motor A y motor B) y luces por separado.



*Opciones de encendido con WeDo.* Susana Oubiña Falcón. (CC BY)

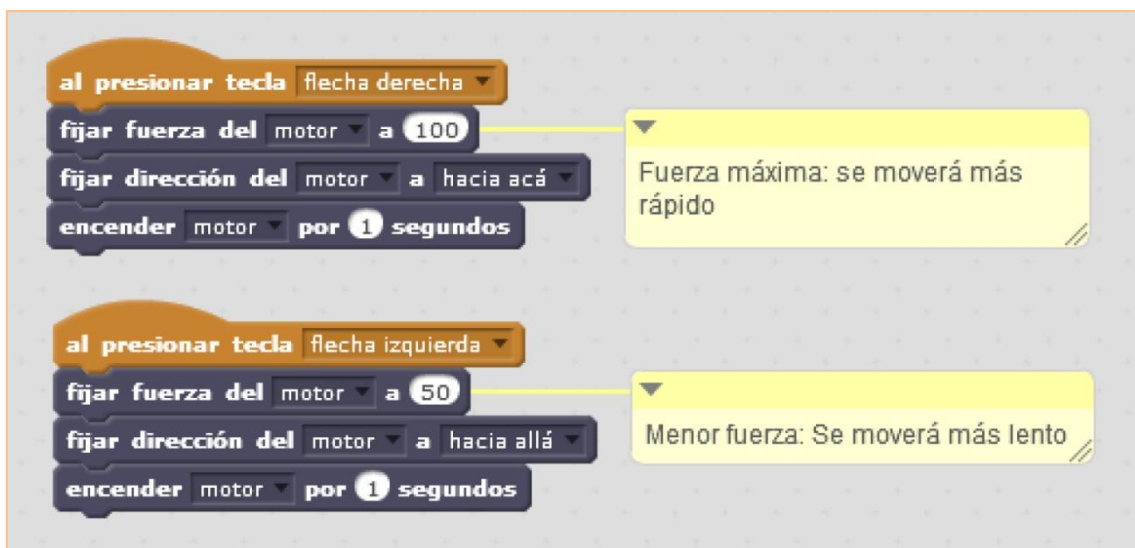


- ✓ Ejemplo 1: El siguiente script realiza el giro de un motor hacia la derecha o izquierda durante 1 segundo, presionando la flecha derecha o izquierda del teclado:



*Control de motor derecha/izquierda.* Susana Oubiña Falcón. (CC BY)

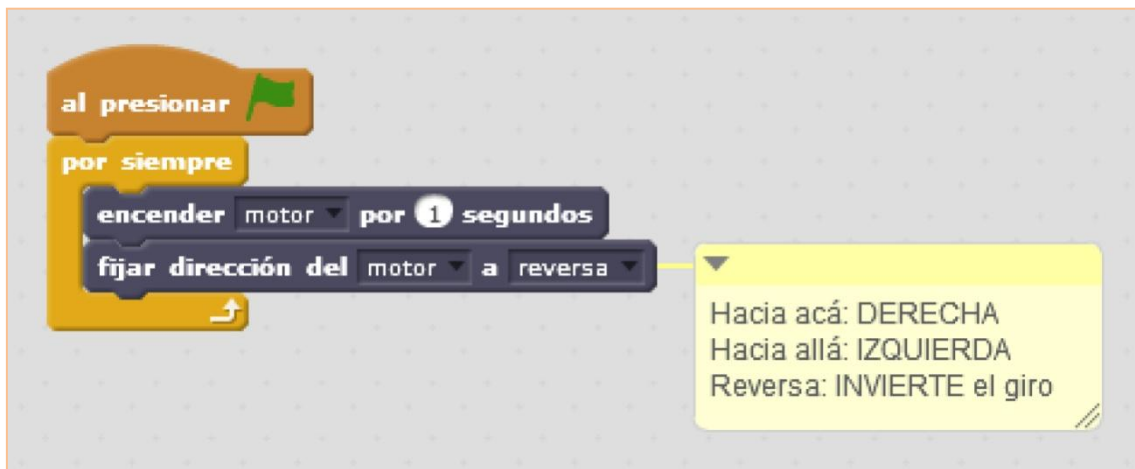
- ✓ Ejemplo 2: También podemos controlar la potencia con la que gira el motor. Supongamos que queremos que gire rápido hacia la derecha, pero, en el giro hacia la izquierda queremos lo haga más lento. El script sería el siguiente:



*Fuerza del motor.* Susana Oubiña Falcón. (CC BY)

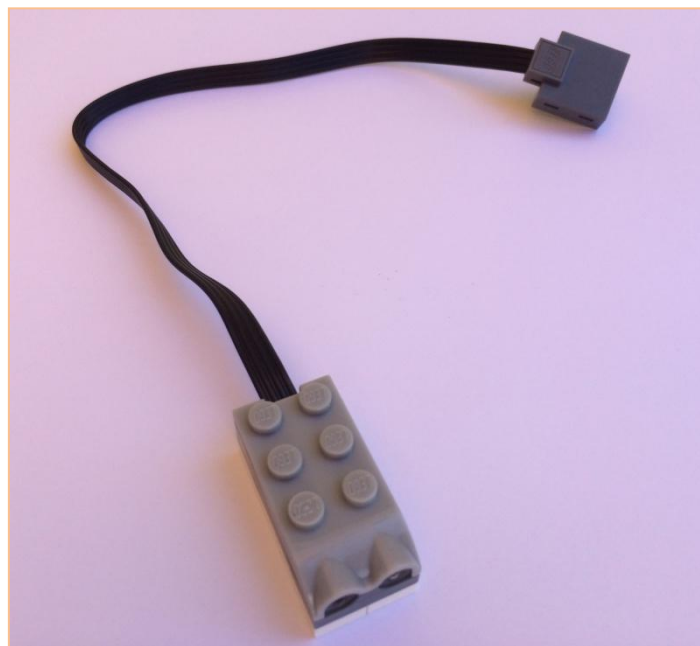
- ✓ Ejemplo 3: Las opciones de la dirección a la que girará el motor son 3: Hacia acá (derecha), hacia allá (izquierda) y reversa (invertir el giro del motor). En esta última, si el motor giraba hacia la derecha, después lo hará hacia la izquierda. En siguiente script muestra un bucle continuo,

por siempre, de modo que el motor alterne el giro del mismo cada segundo:



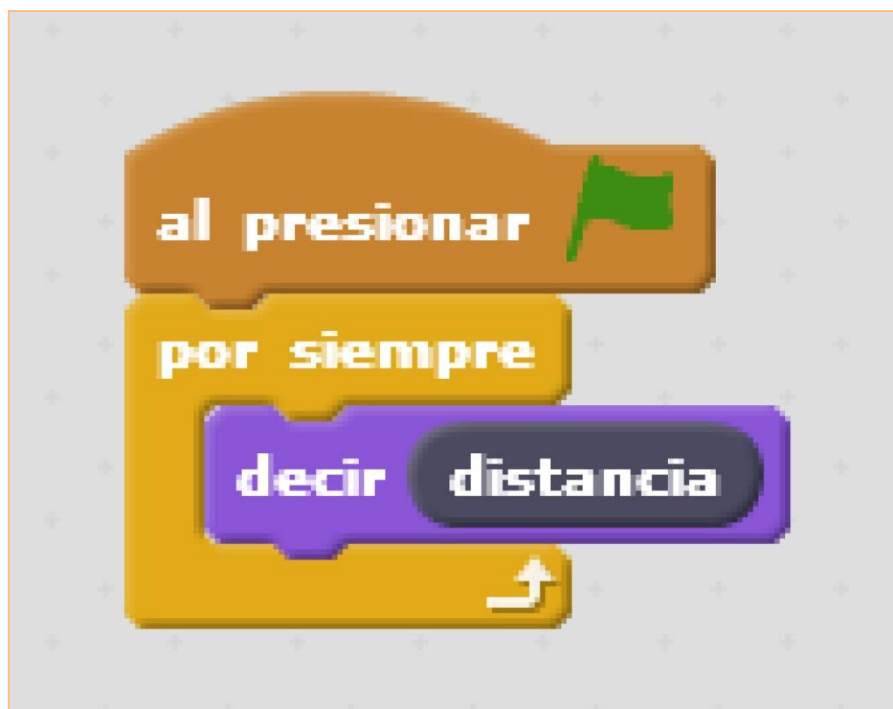
*Giro continuo del motor. Susana Oubiña Falcón. (CC BY)*

Control del sensor de movimiento (distancia): La distancia que aporta el sensor es un valor entre 0 y 100, de modo que: 0 indica que el objeto está a la distancia mínima del sensor (cerca) y 100 que está muy alejado.



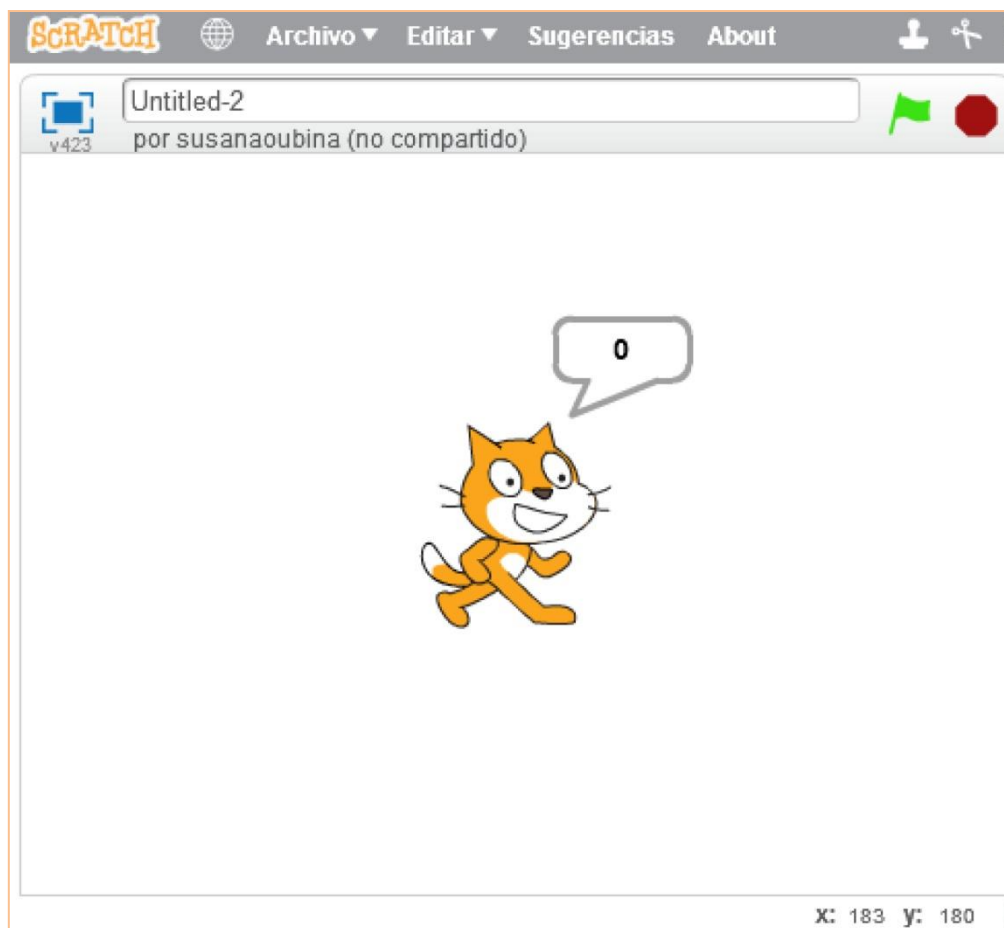
*Sensor distancia del kit de WeDo de Lego. Susana Oubiña Falcón. (CC BY)*

- ✓ Ejemplo 4: Un script sencillo que nos diría la distancia que mide el sensor sería el siguiente:

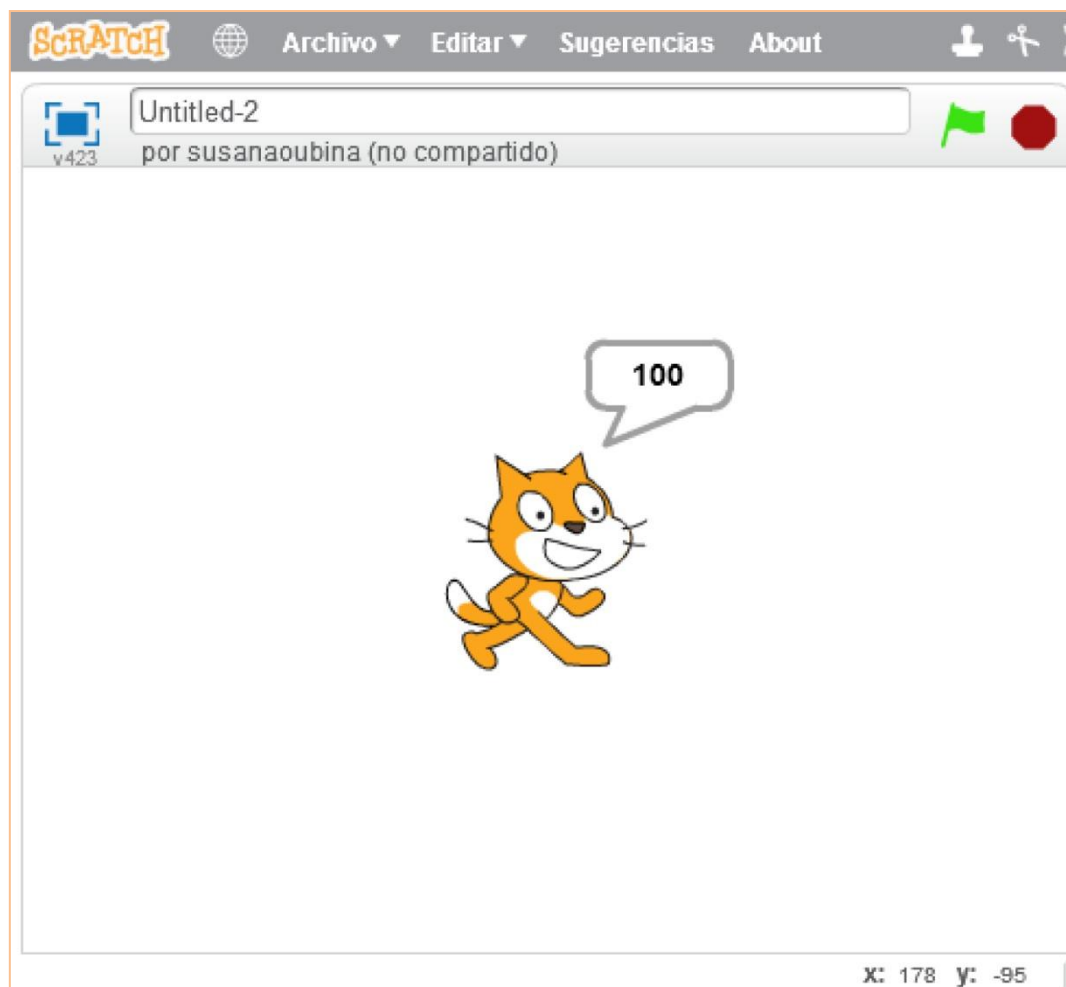


*Script para conocer la distancia que mide el sensor. Susana Oubiña Falcón. (CC BY)*

Si el script anterior se lo asociamos al objeto gato típico del scratch (es decir, el “gato” es el sensor) y utilizamos nuestra mano para hacer variar las medidas de distancia que detecta el gato hacia ella, alejándola o acercándola, observamos que a la mínima distancia nos marca 0 y a la máxima nos marca 100:

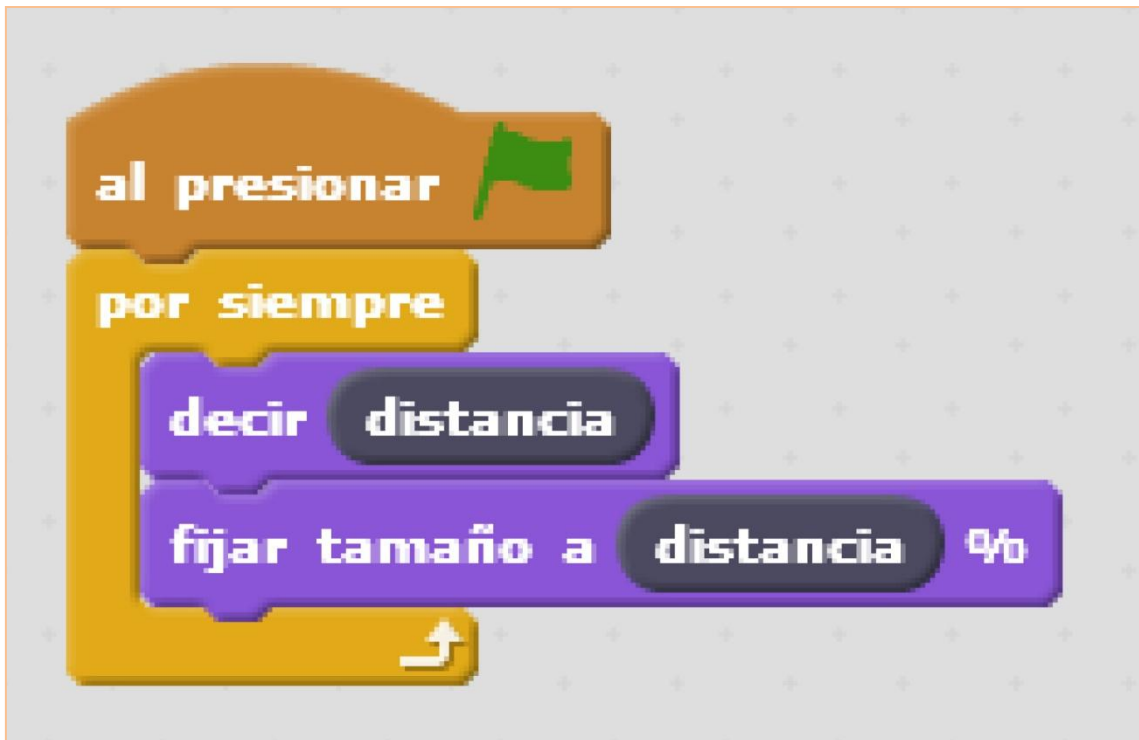


*Mínima distancia detectada.* Susana Oubiña Falcón. (CC BY)



*Máxima distancia detectada.* Susana Oubiña Falcón. (CC BY)

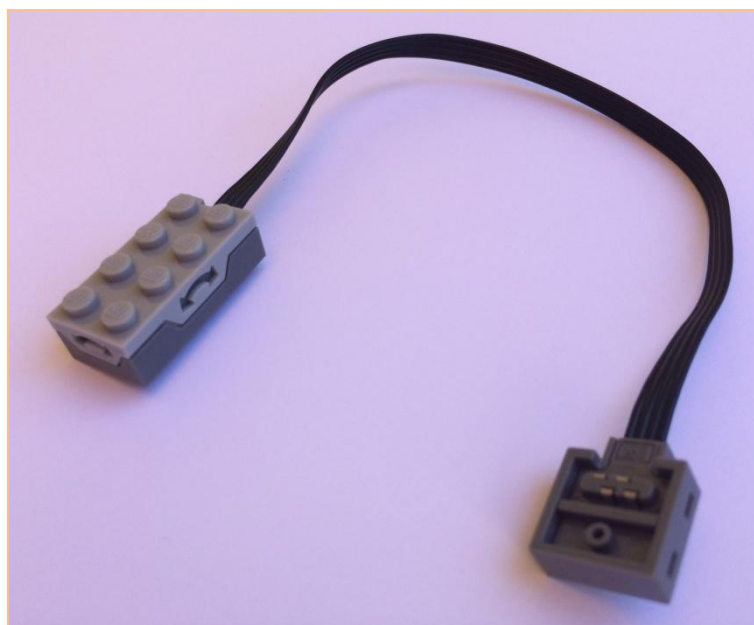
Con el sensor distancia podemos hacer que los objetos del scratch varíen su tamaño. El script para ello sería el siguiente:



*Variación del tamaño del objeto sensor con la distancia. Susana Oubiña Falcón. (CC BY)*

Obviamente, lo interesante es unir la “distancia” con el bloque de operadores para que realice diferentes opciones si esta distancia es mayor, menor o igual a un cierto valor.

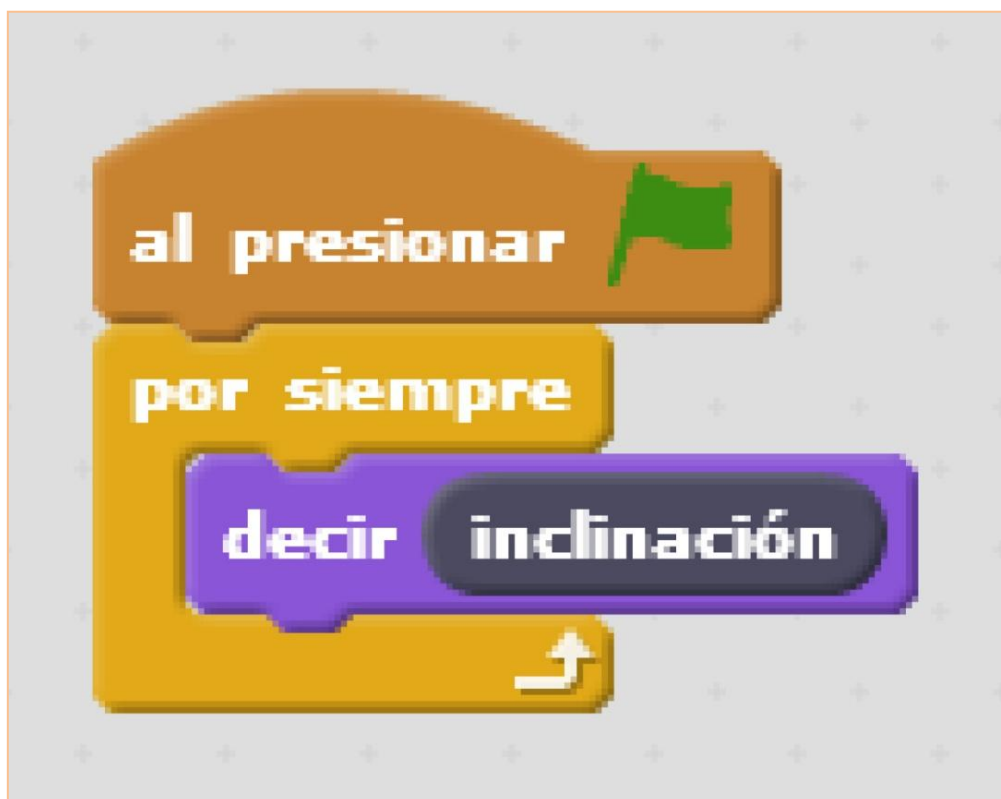
Control del sensor de inclinación: el sensor de inclinación detecta diferentes 5 posiciones: 0 (sin inclinación, recto), 1 (hacia abajo), 2 (hacia la derecha), 3 (hacia arriba) y 4 (hacia la izquierda).



*Sensor de inclinación del kit de WeDo de Lego. Susana Oubiña Falcón. (CC BY)*

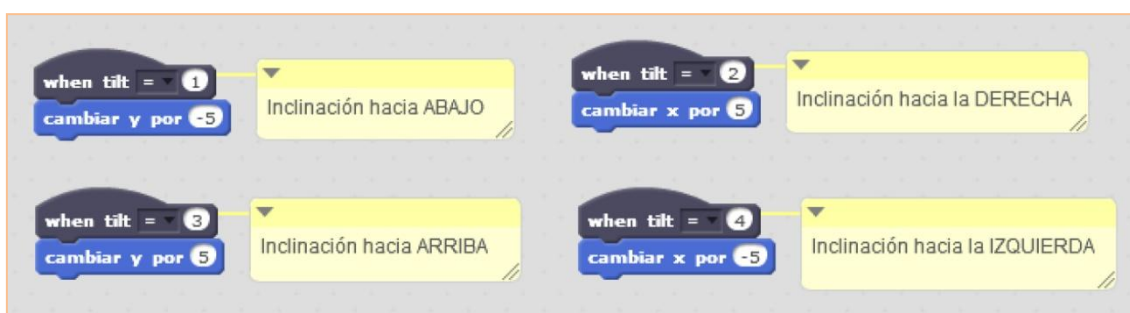


- ✓ Ejemplo 5: El script para conocer la inclinación que mide el sensor sería el siguiente:



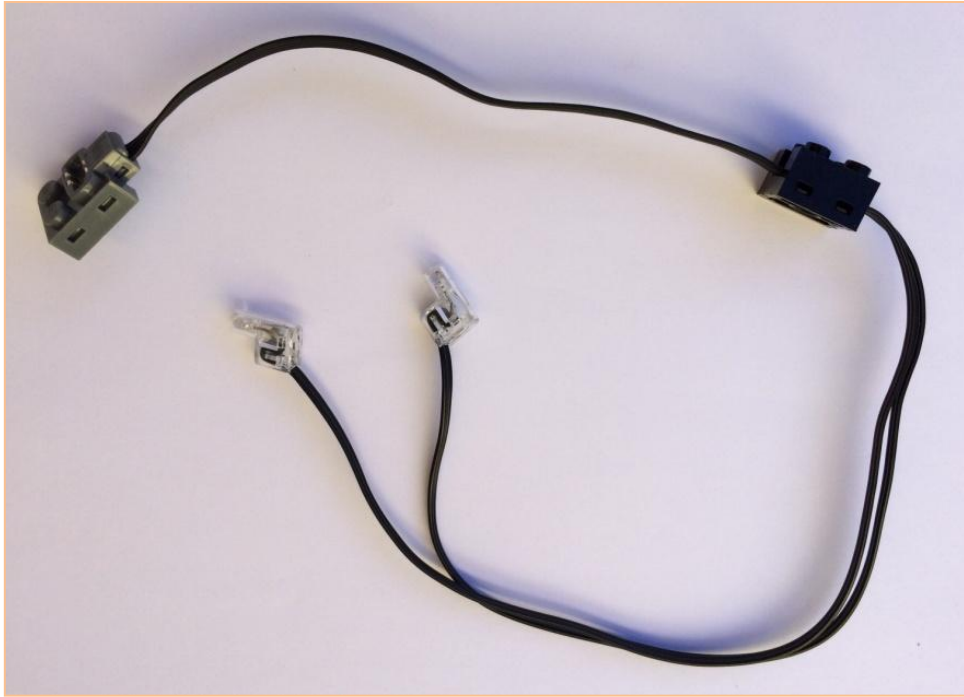
*Script para conocer la inclinación que mide el sensor. Susana Oubiña Falcón. (CC BY)*

- ✓ Ejemplo 6: El sensor de inclinación se puede usar para hacer que un objeto del scratch se mueva por el escenario dependiendo de la inclinación que tenga el sensor del kit de Lego, interactuando con él a tiempo real y como si fuera un elemento de un videojuego. El script para este ejemplo sería el siguiente:



*Script para consola de mandos. Susana Oubiña Falcón. (CC BY)*

Control del sensor de luz: el sensor de luz que se puede conectar al hub USB del WeDo se comercializa fuera del kit y utiliza diodos LED.



*Sensor de luz (Luces-LEGO Power Functions).* Susana Oubiña Falcón. (CC BY)

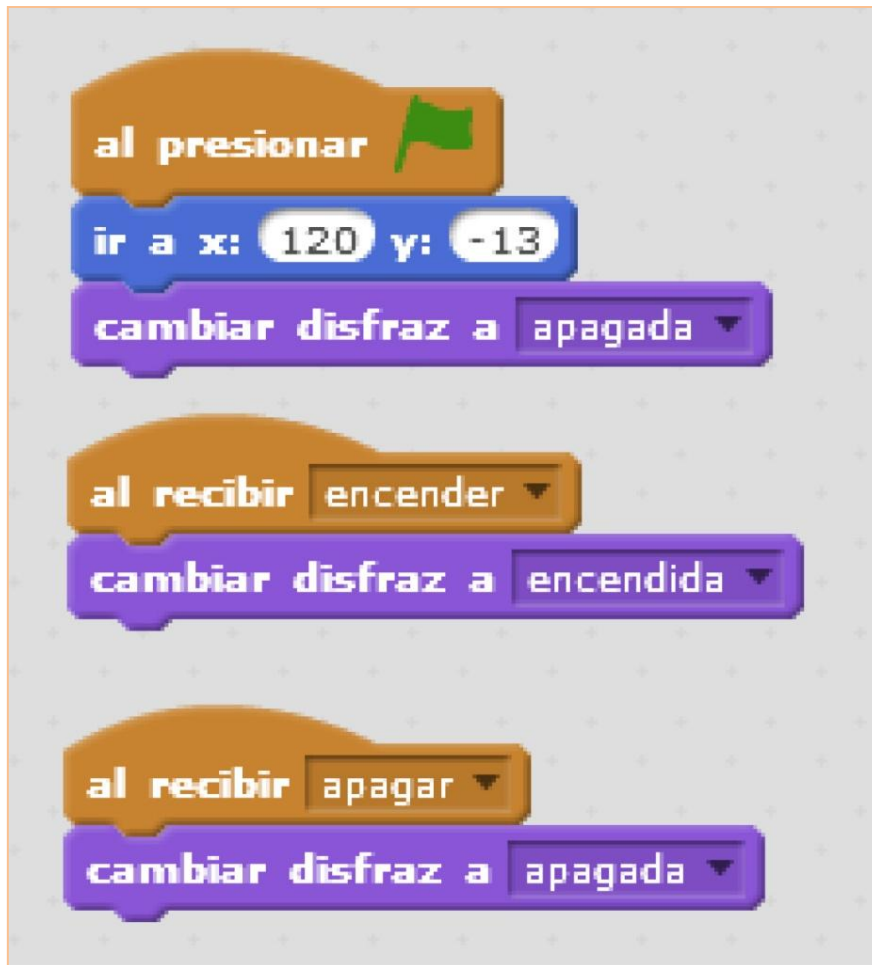
- ✓ Ejemplo 7: Encendido y apagado de una bombilla. Este programa simula el encendido y apagado de una bombilla. Conectado con el sensor de luz Power Functions no sólo lo simula sino que se observa el fenómeno.

El proyecto es muy sencillo y consta de dos objetos: el gato (que es el que encenderá y apagará la bombilla, dándole las órdenes al sensor de luz) y la bombilla (que se presenta para simular el proceso en el entorno scratch).

El script de ambos objetos se muestra en las siguientes imágenes:



*Script objeto "Gato" (Sensor de Luz) para scratch 2.0.* Susana Oubiña Falcón. (CC BY)



Script objeto “Bombilla” (Sensor de Luz). Susana Oubiña Falcón. (CC BY)

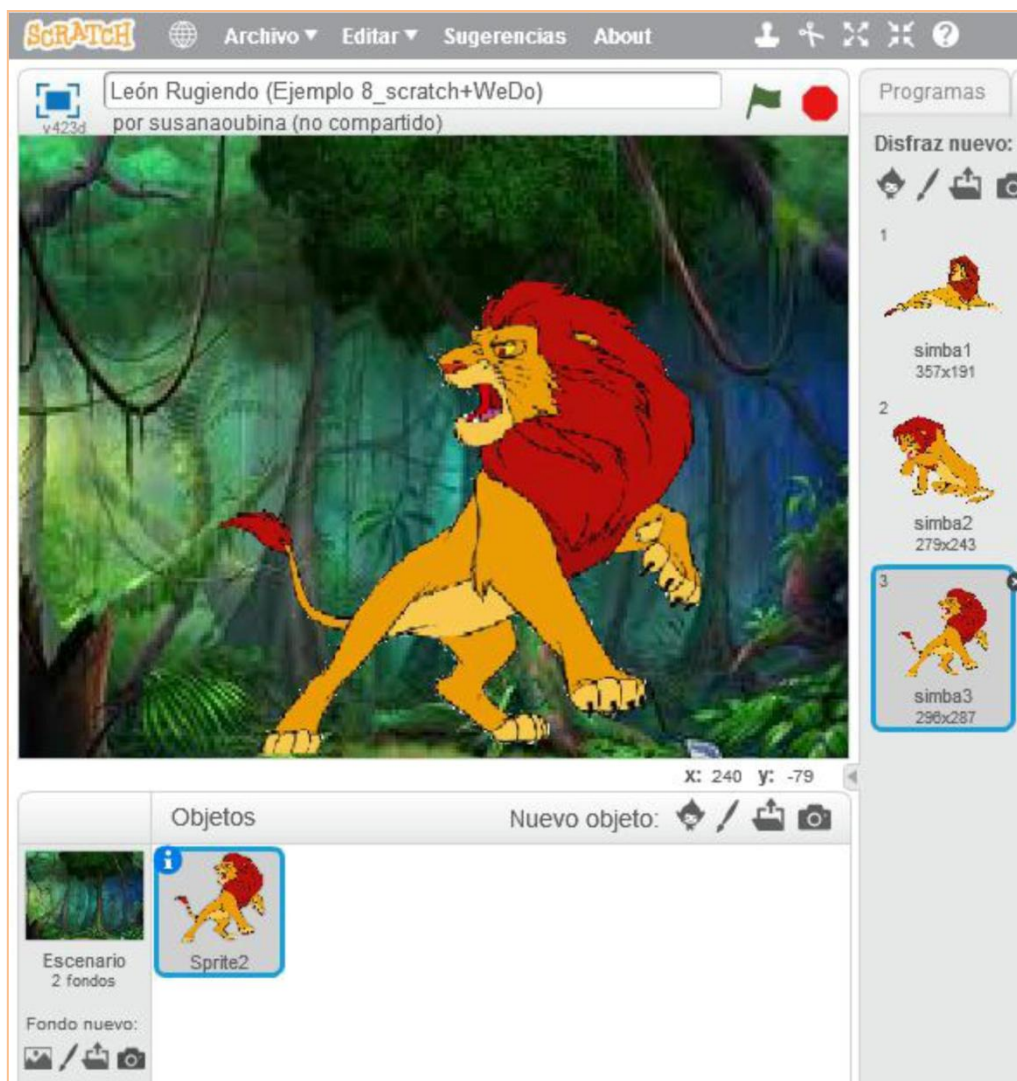
El siguiente vídeo explica el ejemplo 7. En él podemos observar cómo funciona el proyecto:

[“Sensor de luz con WeDO. Encendido y pagado de una bombilla”](#) Susana Oubiña Falcón. (CC BY)

#### D. Proyectos más complejos

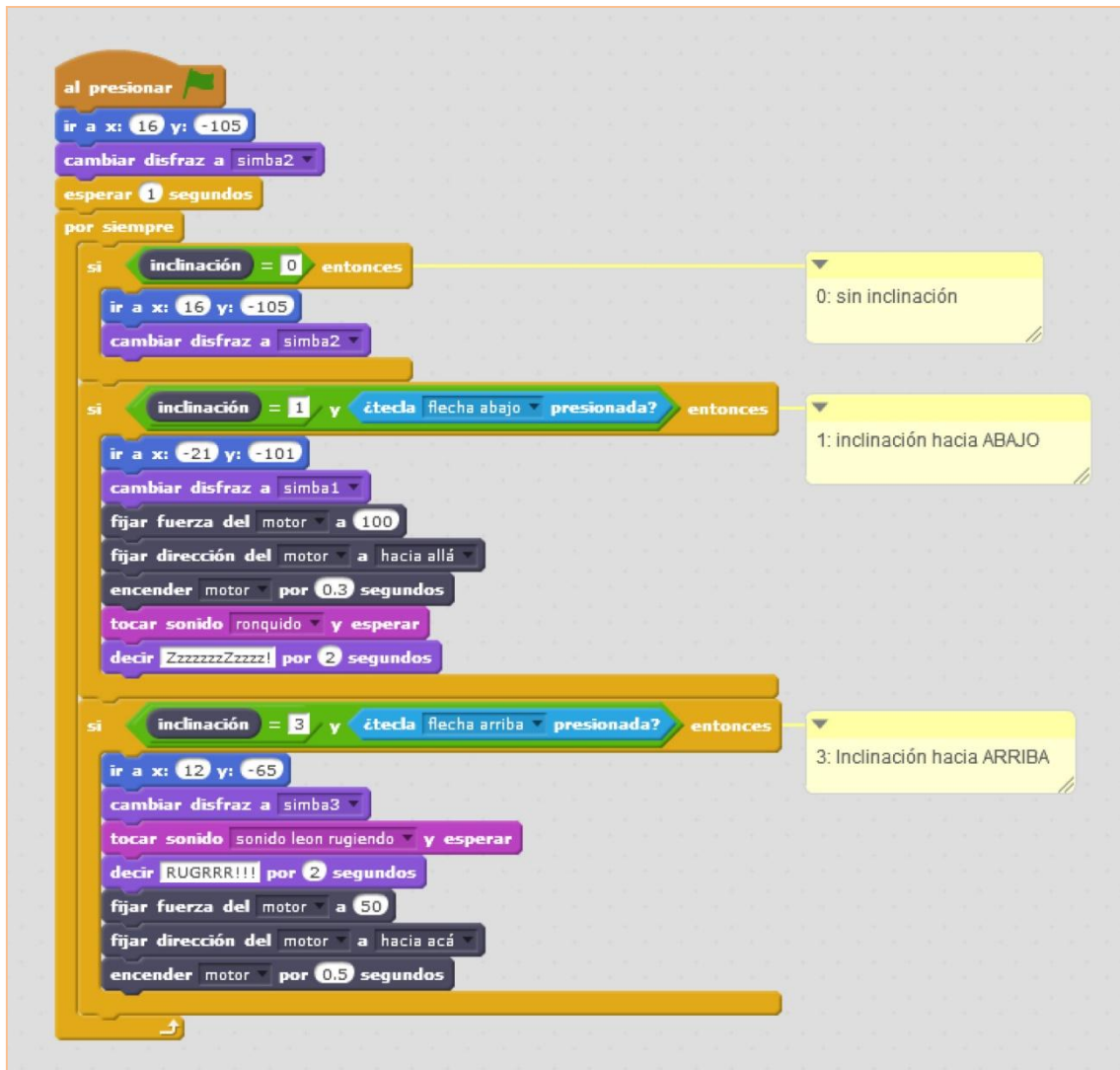
- ✓ Ejemplo 8: *León rugiendo*. Este proyecto es un diseño típico del WeDo. Lo he programado en el scratch 2.0 usando un motor y el sensor de inclinación.

Lo que busco es que se levante y ruja si presiono la tecla flecha derecha y la inclinación es hacia arriba y, se baje y emita un ronquido si presiono la tecla flecha abajo y el sensor de inclinación está posicionado hacia abajo. Necesitamos un único objeto el cual tendrá 3 disfraces, uno para la posición inicial y los otros dos para los efectos que queremos hacer, tal y como se muestra en la siguiente imagen:



*Disfraces del objeto "sprite2" (León). Susana Oubiña Falcón. (CC BY)*

Su script es el siguiente:



Script “León Rugiendo” para scratch 2.0. Susana Oubiña Falcón. (CC BY)

Al poner en funcionamiento el programa anterior, en el escenario del scratch se simula el proceso programado y, al mismo tiempo, este proceso ocurre con el robot real (león) montado con piezas del Lego del kit Wedo.

- ✓ Ejemplo 9: *Caimán Hambriento*. Este proyecto, también basado en los montajes básicos e WeDo para Lego utiliza un motor y un sensor de movimiento (distancia) que detecta el acercamiento de un objeto.

El script para este proyecto sería el siguiente:



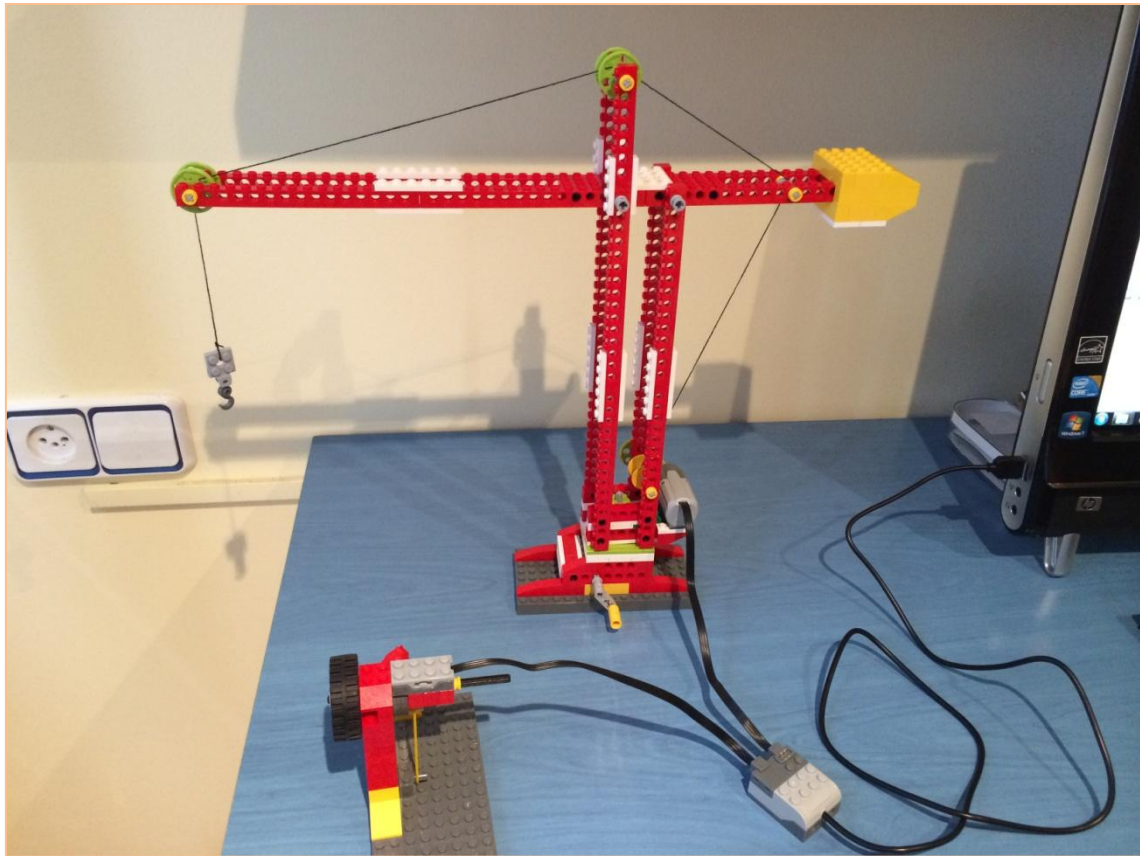


Script "Caimán Hambriento" para scratch 2.0. Susana Oubiña Falcón. (CC BY)

- ✓ Ejemplo 10: *Control de una grúa*. Este proyecto necesita, para su montaje, de dos kits de LEGO: el kit básico de WeDo y el set de recursos Lego WeDo.

El resultado de su montaje se observa en la siguiente imagen. La elevación o bajada del objeto que mueve la grúa puede realizarse de forma manual o programada:





*Grúa.* Susana Oubiña Falcón. (CC BY)

La programación es muy simple: inicialmente fijo la potencia del motor a 50, para que suba ni baje demasiado rápido. También quiero que me muestre en el escenario, la inclinación que va detectando, por ello, el programa comienza con el siguiente pequeño script:



*1ª Parte: Script inicial de la grúa.* Susana Oubiña Falcón. (CC BY)

A continuación, programo el sensor inclinación según las diferentes acciones para las posiciones que quiero incluir en el programa: posición 2 (derecha), 4 (izquierda) y 0 (sin inclinación). El script es el siguiente:



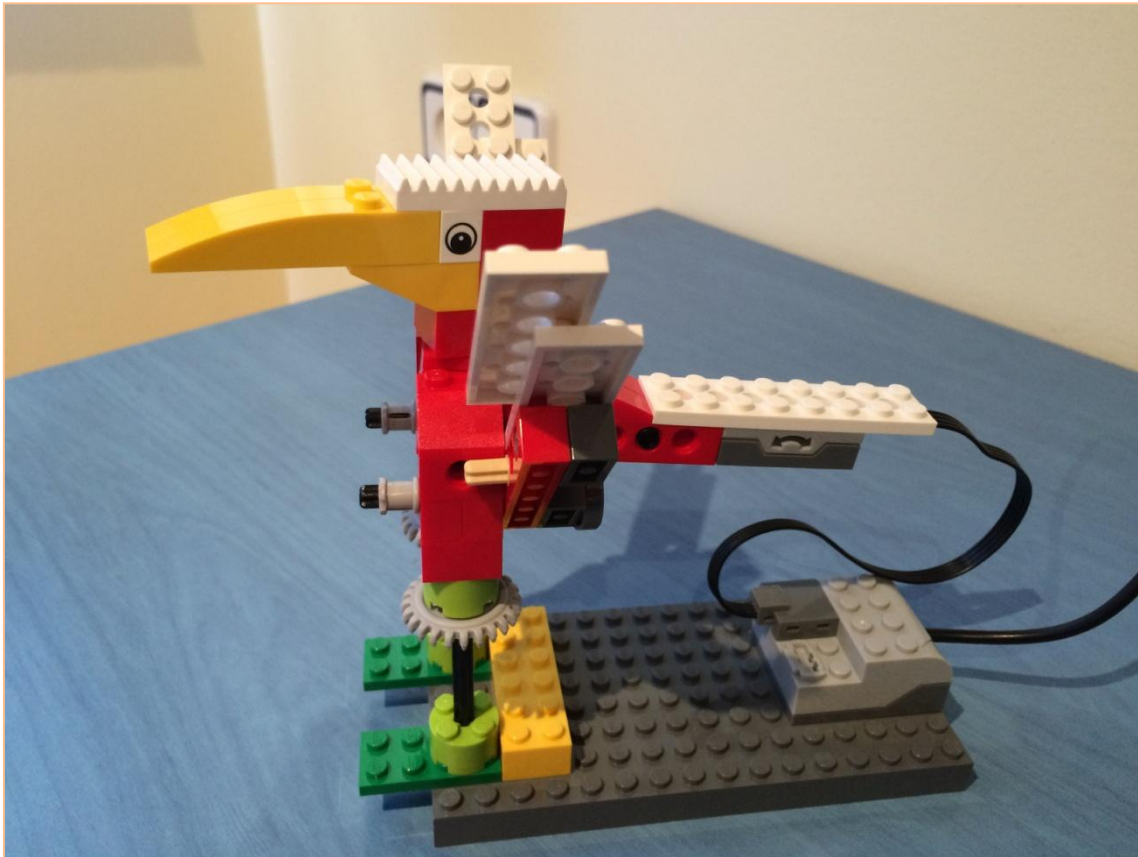
2ª Parte: Script del sensor de inclinación. Susana Oubiña Falcón. (CC BY)

El funcionamiento de la grúa puede verse en el siguiente vídeo:

["Funcionamiento de una Grúa \(WeDo\) de Lego programada con Scratch 2.0".](#)  
Susana Oubiña Falcón. (CC BY)

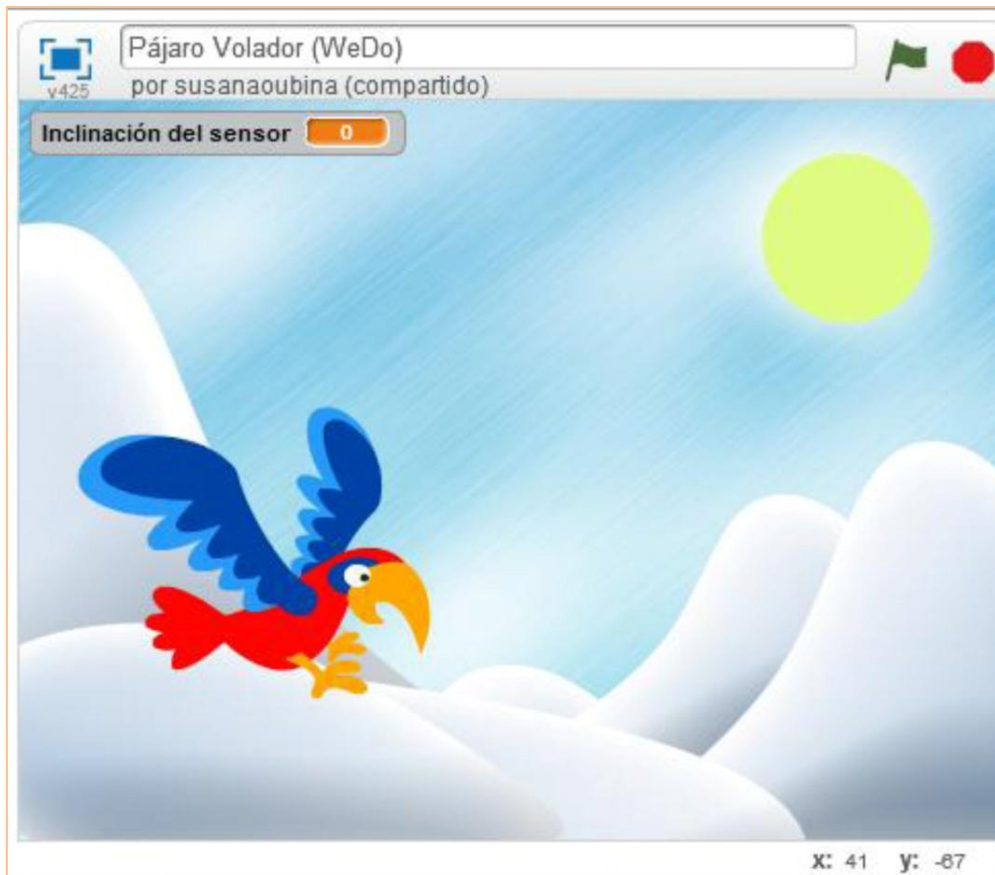
- ✓ Ejemplo 11: *Pájaro Volador*. Este proyecto, al igual que el del león, es un diseño típico del WeDo. Lo he programado en el scratch 2.0 usando el sensor de inclinación.

El robot físico, caracterizado como Pájaro Volador, se muestra en la siguiente imagen:



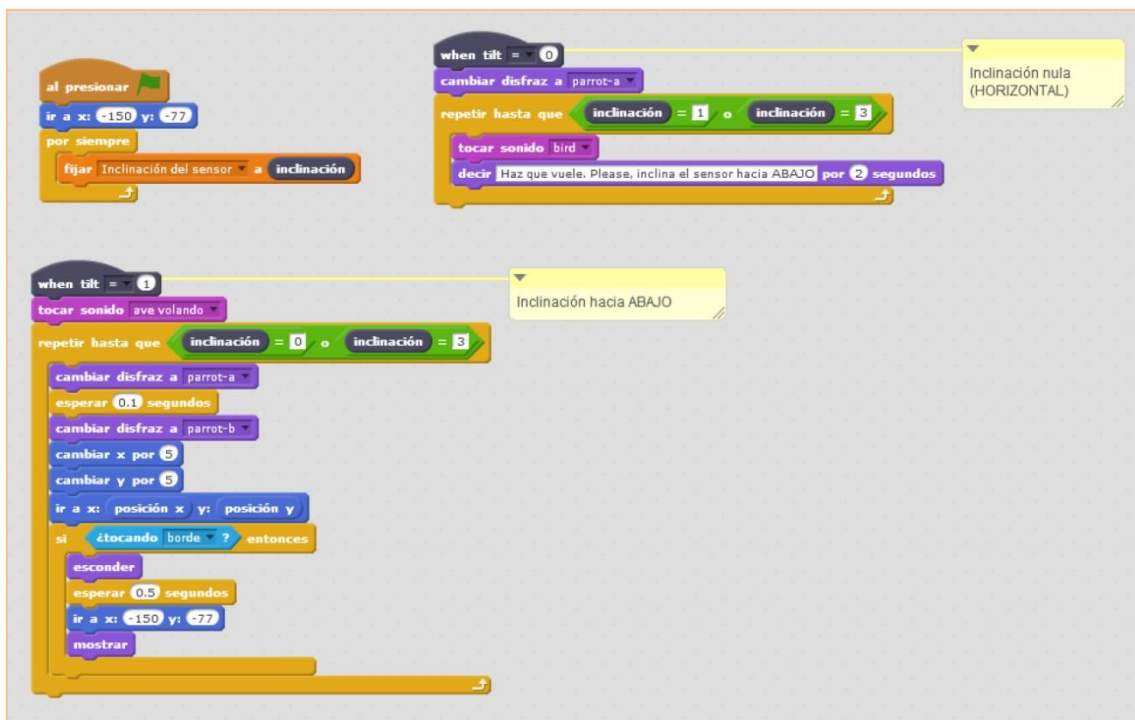
*Pájaro Volador*. Susana Oubiña Falcón. (CC BY)

Como puede verse en la siguiente imagen que muestra el escenario del programa, hemos creado una variable llamada “Inclinación del sensor”, para que nos muestre en cada momento el valor del sensor que manipulamos de forma física en nuestro robot. El único objeto del programa es un pájaro (Parrot) y las posiciones que programamos en el sensor de inclinación son 0 (inclinación horizontal) y 1 (inclinación hacia abajo):



Escenario del programa *Pájaro Volador*. Susana Oubiña Falcón. (CC BY)

El script del programa es el siguiente:



Script para el robot *Pájaro Volador*. Susana Oubiña Falcón. (CC BY)



La inclinación 1 repite un bucle mientras ésta no cambie a 0 o 3. Este bucle hace que se mueva el pájaro simulando su vuelo. La inclinación 0 repite un bucle mientras ésta no cambie a 1 o 3. Este bucle nos informa de lo que debemos hacer para conseguir que el pájaro vuele. También observamos que, al presionar la bandera verde, el objeto se sitúa en una posición y, en el escenario, siempre se mostrará la “inclinación” del sensor en la variable “Inclinación del sensor”.

El funcionamiento, puede verse en el siguiente vídeo:

[“Pájaro volador del WeDo con Scratch 2.0”](#). Susana Oubiña Falcón. (CC BY).

- ✓ Ejemplo 12: *Simulación del vuelo de un helicóptero*. Con el kit del WeDo se puede construir un avión. Ese avión, que en mi proyecto de scratch lo he simulado con el objeto helicóptero (ya incluido en los objetos del programa y en el que sólo he modificado sus disfraces), presenta un motor y un sensor de inclinación.

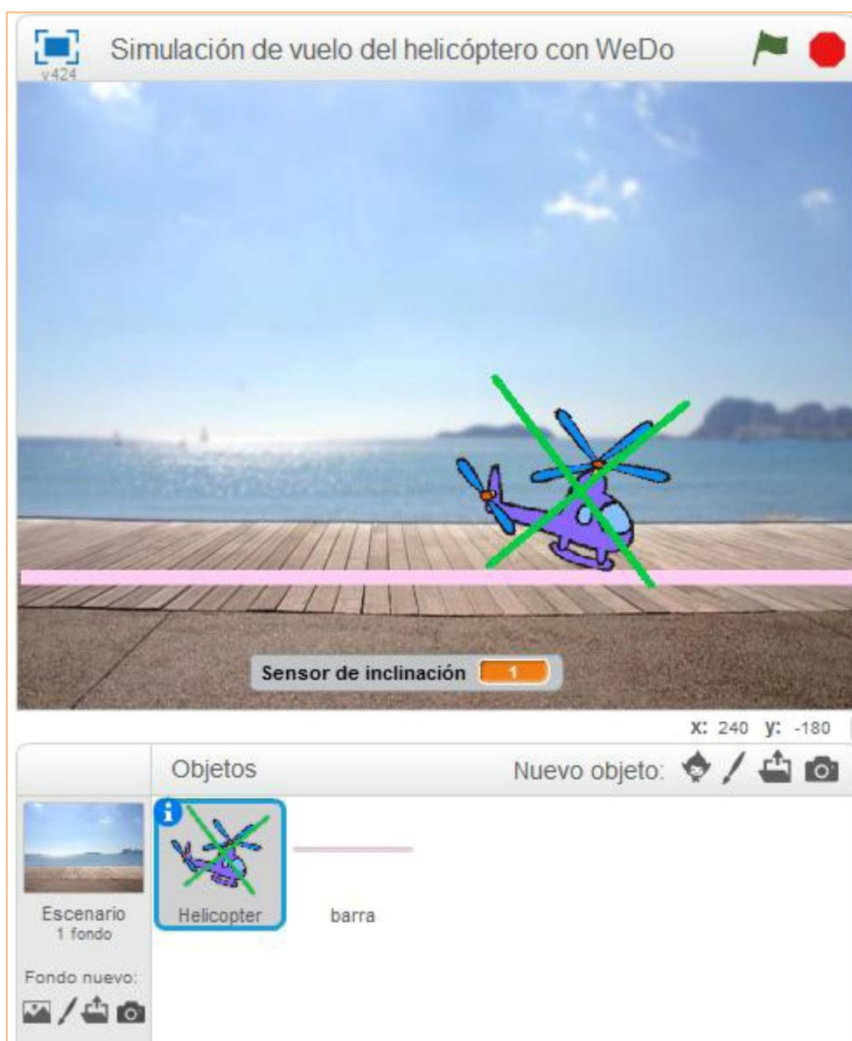
El robot físico, caracterizado como avión, se muestra en la siguiente imagen:



*Avión.* Susana Oubiña Falcón. (CC BY)

Nuestro programa controla el motor y su sensor de inclinación (en las posiciones hacia arriba y hacia abajo). Para ello, he utilizado dos objetos: el

helicóptero y el objeto barra. El escenario del proyecto en scratch, así como ambos objetos, puede verse en la siguiente imagen:



*Escenario de “Simulación del vuelo de un helicóptero”.* Susana Oubiña Falcón. (CC BY)

En la imagen anterior, se observa que se ha creado una variable “Sensor de inclinación”, la cual, mostrará a tiempo real como vamos variando la inclinación de nuestro robot WeDo físico. La inclinación “1” nos dice que lo estamos inclinando hacia abajo y la inclinación “3” que lo inclinamos hacia arriba.

El objeto “barra” simula el choque, de modo que, cuando el helicóptero toque la barra, éste se estrella. En ese momento, se cambia el disfraz (ver figura anterior), suena el sonido “helicóptero choque” y finaliza el programa.

Para simular un vuelo se han creado disfraces en el objeto “helicóptero”. Los 5 disfraces se usan de la siguiente forma: los dos primeros para el vuelo de bajada, los dos siguientes para el vuelo de subida y el último para el choque del helicóptero:



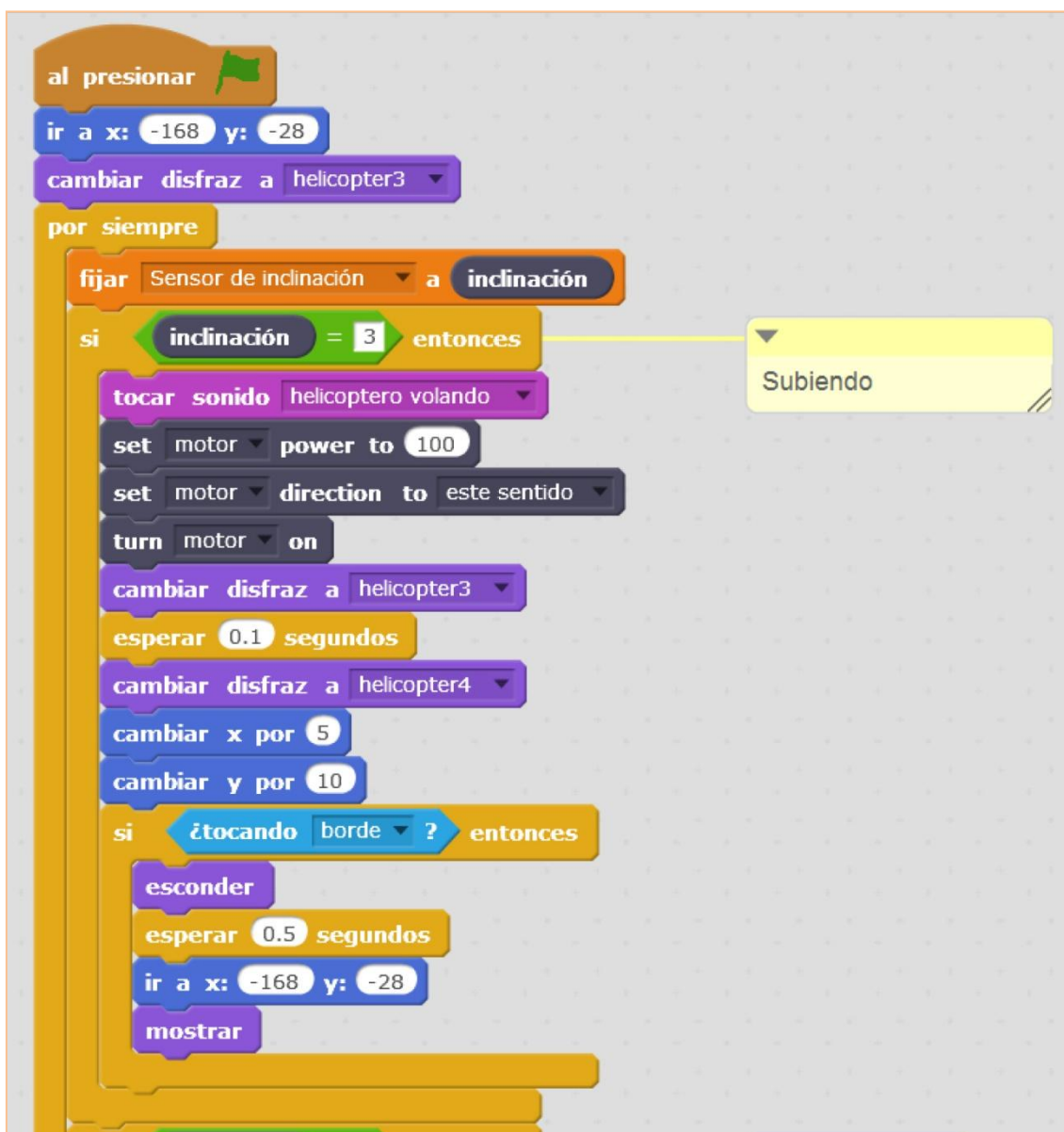


*Disfraces del objeto "Helicóptero". Susana Oubiña Falcón. (CC BY)*

El programa comienza situando el helicóptero en una posición determinada del escenario otorgándole el disfraz "helicopter3". A continuación se crea un bucle que se realizará "por siempre" en el que se incluyen dos condicionales. En el primer condicional se programa la "subida" (inclinación 3) y en el segundo la "bajada" (inclinación 1). En ambos, subida y bajada, el motor gira y también se programa de forma diferente. En la simulación de la subida, he decidido que el motor gire en "este sentido" (sentido de las agujas del reloj) y a la máxima potencia, pero, en la bajada, he preferido simular que las cosas no van bien en el helicóptero y, por ello, funcionará a una potencia inferior (50 en nuestro caso) y girará en el sentido contrario "ese sentido".

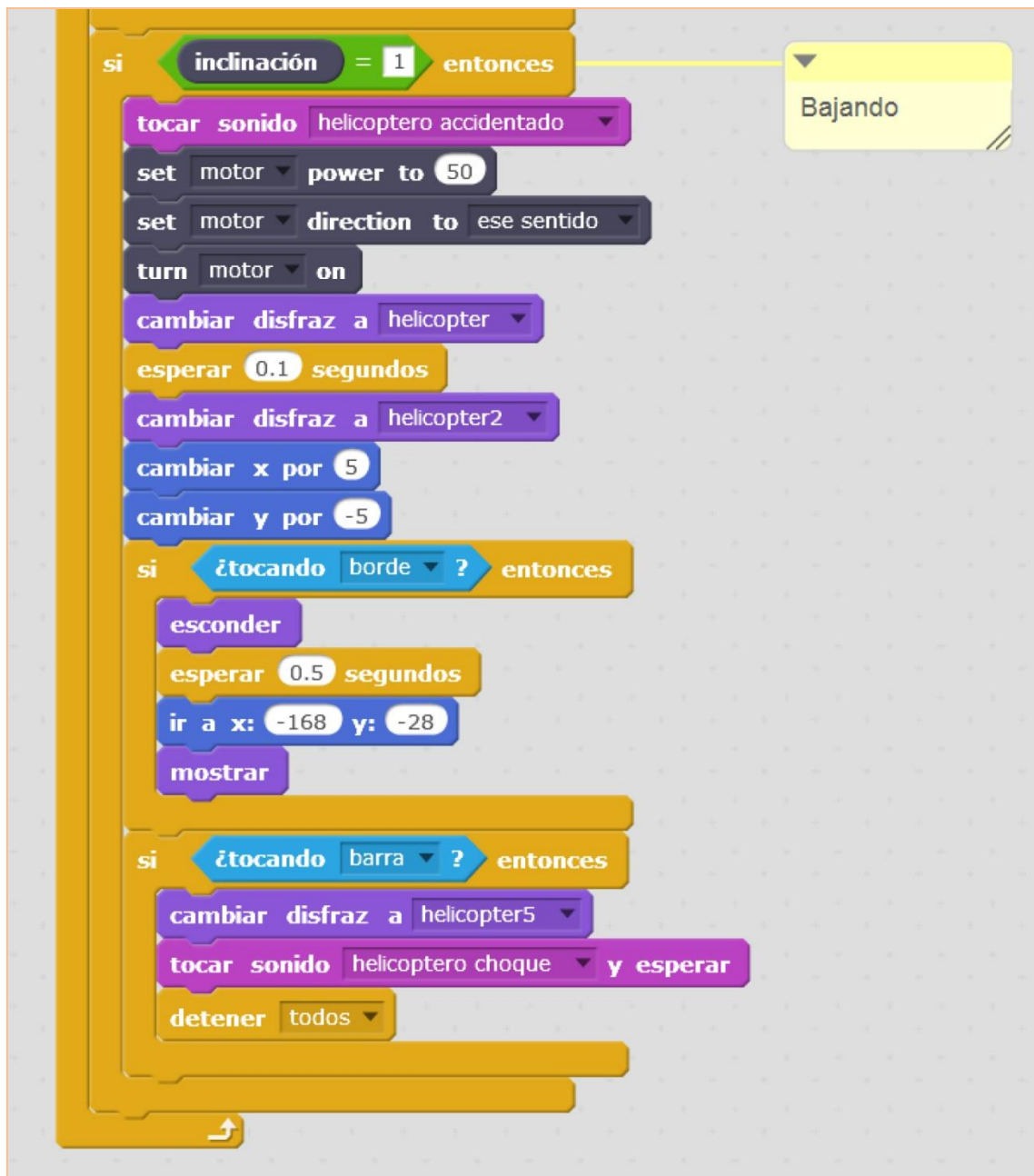
El script para la subida (dentro del bucle "Por siempre") se muestra en la siguiente imagen. Como puede verse, el objeto "Helicóptero" aumenta su

posición en el eje X por 5 y su posición en el eje Y por 10. Además, si toca el borde, vuelve a su posición de origen:



Script de subida del helicóptero. Susana Oubiña Falcón. (CC BY)

La bajada se programa con otro condicional que se incluye en el mismo bucle principal "Por siempre". Es similar al anterior con los cambios de sonido, disfraces y posición de la variable Y (que valdrá -5 porque se mueve hacia abajo). A mayores, en esta "inclinación 3", que simula que el objeto baje, se ha incluido la simulación de un choque con el objeto "barra". Cuando el helicóptero toque la barra, cambiará su disfraz a "helicóptero5" (ver la imagen de disfraces), sonará el sonido "helicóptero choque" y se detendrá el programa. El script para la bajada es el siguiente:



*Script de bajada del helicóptero. Susana Oubiña Falcón. (CC BY)*

## E. Diferencias con scratch 1.4

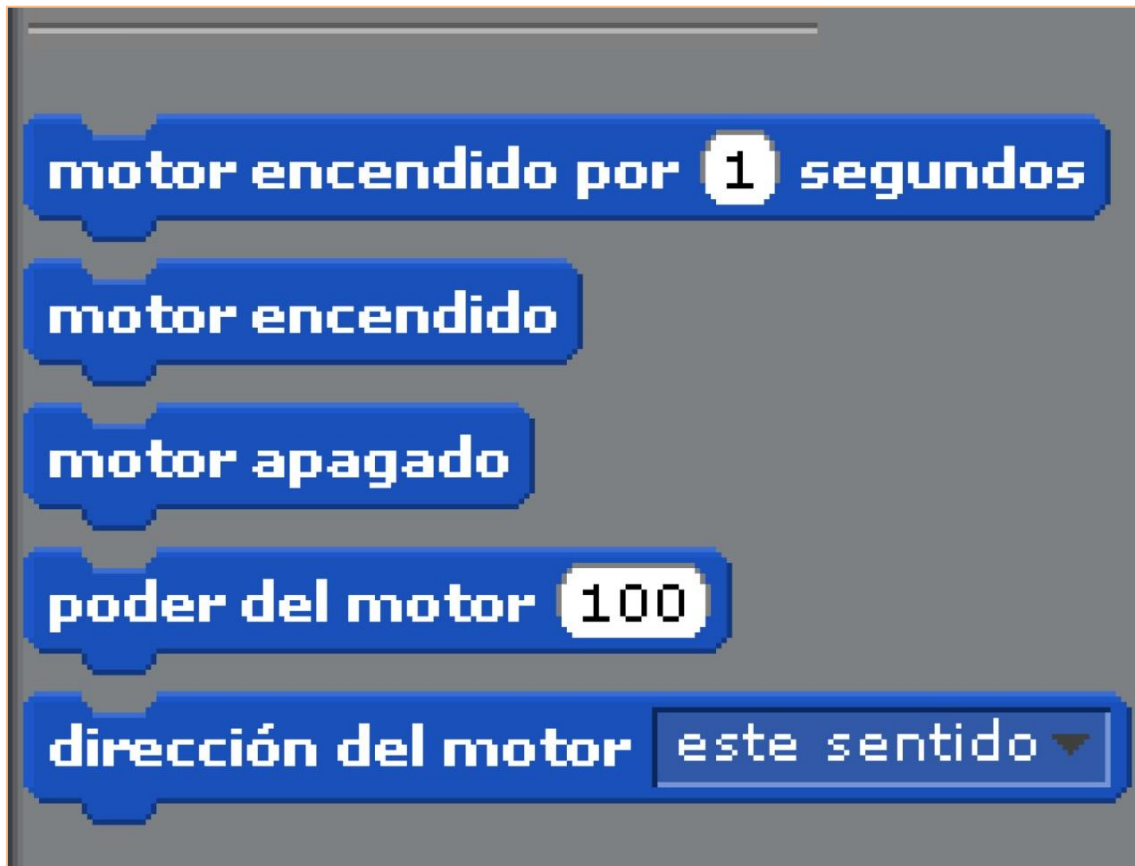
El kit del WeDo se puede conectar con el scratch 1.4. Este programa sólo permite, de forma sencilla, el control de un motor y de los sensores de inclinación y distancia o movimiento (kit básico del WeDo), pero no del sensor de luz.

Para ver los comandos de control del motor debemos hacer clic en “Editar” y seleccionar la opción “Mostrar Bloques de Motor”:



*Opciones “Editar” en scratch 1.4. Susana Oubiña Falcón. (CC BY)*

Si ahora entramos en el bloque azul de Movimiento, veremos que se han incluido unos nuevos comandos para controlar el motor:



*Comandos de control de motor para scratch 1.4. Susana Oubiña Falcón. (CC BY)*

Se observa que en el comando “dirección del motor...” utiliza otra nomenclatura para direccionar el motor en el giro a la derecha (en el sentido de las agujas del reloj), a la izquierda o revertir su giro:

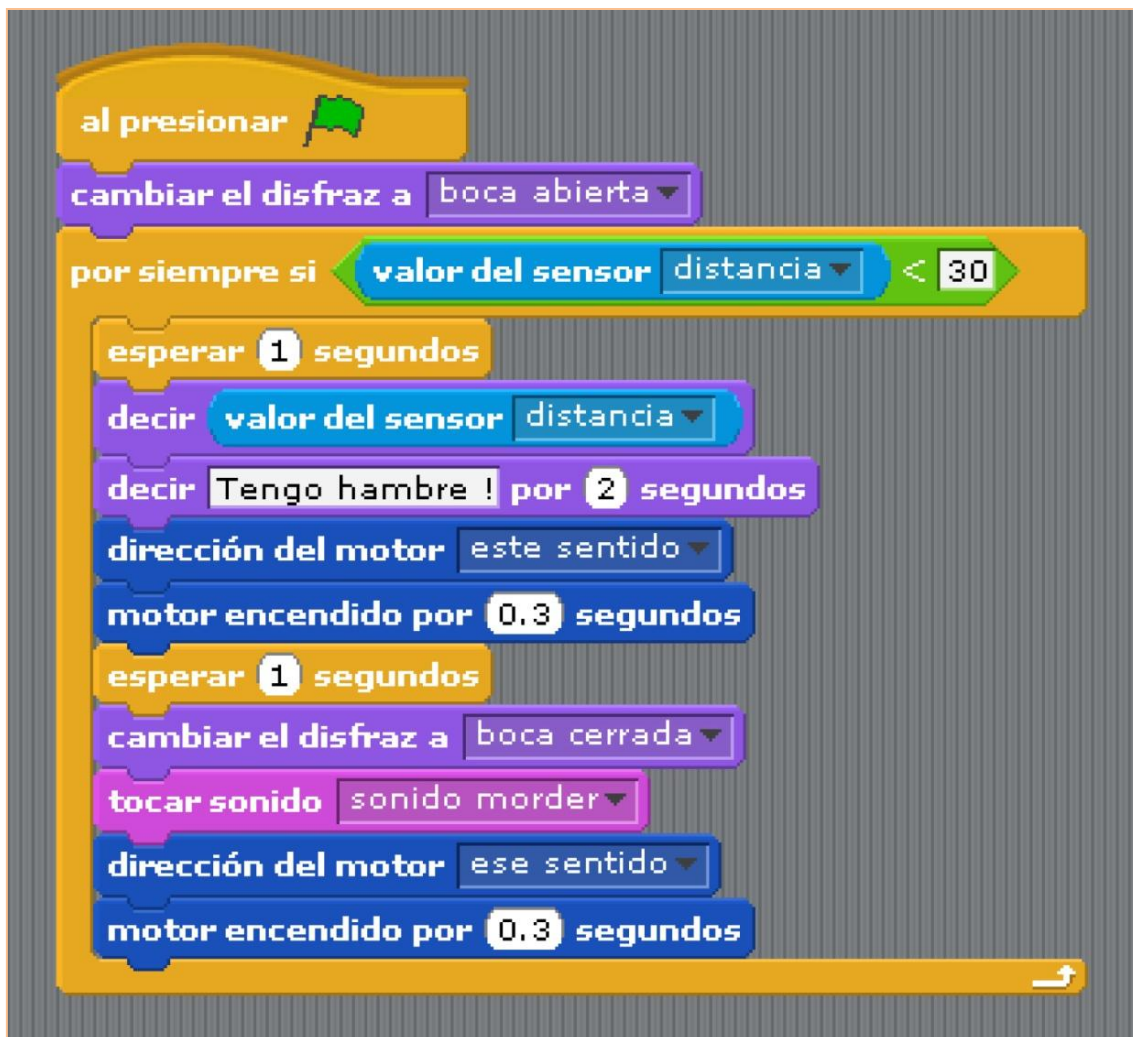


*Opciones de giro del motor para scratch 1.4. Susana Oubiña Falcón. (CC BY)*

Direcciones	Scratch1.4	Scratch2.0
<b>Derecha (sentido de las agujas del reloj)</b>	este sentido	hacia acá
<b>Izquierda (sentido contrario de las agujas del reloj)</b>	ese sentido	hacia allá
<b>Cambiar de dirección</b>	reversa	reversa

*Diferencias de nomenclatura en “direcciones”. Susana Oubiña Falcón. (CC BY)*

A modo de ejemplo, el script para el programa del Ejemplo 9 “*Caimán Hambriento*” para scratch 1.4 sería el siguiente:



Script "Caimán Hambriento" para scratch 1.4. Susana Oubiña Falcón. (CC BY)

### 3.1.2.5. Robot mOway

Con el Scratch 1.4 se puede programar el robot mOway. mOway es un pequeño robot programable diseñado para realizar prácticas de robótica móvil y que presenta una curva de aprendizaje muy rápida, potenciando la motivación de nuestro alumnado. Para ello, dispone de una serie de sensores que consiguen que interaccione con el mundo físico y de un conjunto motor que le permite desplazarse sobre una superficie. El robot se controla por un microprocesador que está conectado a todos los periféricos.

Las ventajas de mOway con el scratch se resumen como sigue:

- ✓ Se programa desde el scratch siendo un robot autónomo ya que no necesita estar conectado al ordenador.
- ✓ También se programan con el scratch sus sensores, motores y actuadores; los cuales pueden ser leídos y controlados en movimiento.

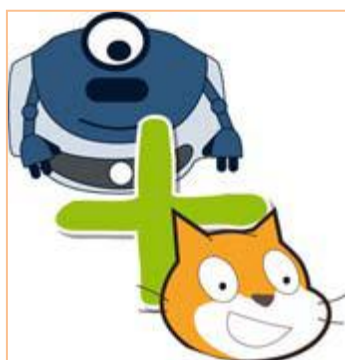


- ✓ mOway ayudará a nuestro alumnado a comprender y asimilar conceptos, otorgando al programa creado un objeto en movimiento en el mundo físico.

**NOTA:** Al igual que el robot WeDo, mOway dispone de su propio software (basado en el lenguaje C) y que, personalmente, lo considero muy útil para Tecnología en 4ºESO e incluso Bachillerato.

## A. Instalación

Para hacer correr al robot mOway con scratch debemos conectar ambos elementos. Para comenzar a trabajar con mOway primero hay que descargar el pack de instalación que contiene el software. El robot puede programarse directamente en Windows o Linux, con su propio software (MowayWorld, disponible en [www.moway-robot.com/descargas](http://www.moway-robot.com/descargas)) o con el Scratch 1.4 (para ello es necesario descargar la aplicación mOwayScratch disponible en el siguiente [link](#)).



Aplicación mOwayScratch. Susana Oubiña Falcón. (CC BY)

La propia web del robot presenta tutoriales muy sencillos y claros de su conexión y programación. Entre ellos y para el punto de explicación de los pasos a seguir para conectar mOway con el scratch, destaco el siguiente vídeo y archivo: Vídeo que describe los pasos a seguir para la [conexión mOway-Scratch](#), archivo que muestra paso a paso [cómo conectar mOway a scratch](#).

## B. ¿Cómo trabaja?

Es importante entender cómo se controla el robot mOway desde el entorno scratch. Como ya sabemos, scratch utiliza *comandos* (órdenes que escribimos en scratch) y que, obviamente, estos deben ser enviados al robot, de modo que, cuando reciba la orden, el robot realice la acción que se le ha pedido (activando el robot para esa acción). Por otra parte, el robot dispone de *sensores* que interaccionan en todo momento con el entorno en el que se encuentra el robot, de modo que, el robot está continuamente enviando datos de sus sensores al entorno scratch. Entre los sensores que dispone el mOway

se encuentra el sensor de obstáculos para detectar objetos delante del robot, el sensor de luz, el de temperatura, etc.

De forma resumida (tal y como se observa en la siguiente imagen), el scratch envía “comandos” u órdenes al mOway y éste se comunica con el scratch por medio de “sensores”. Ambas comunicaciones Scratch-mOway se realizan de forma inalámbrica (por radiofrecuencia), mediante los elementos RF-USB y el módulo RF, y para gestionar esta comunicación se requiere de la instalación de la aplicación “mOwayScratch”:

- ✓ El dispositivo RF-USB se conecta al ordenador, siendo su base de operaciones el Scratch 1.4. Por lo tanto, envía comandos desde el scratch y recibe el valor de los sensores.
- ✓ El módulo de RF (Radio Frecuencia) se conecta con el mOway, siendo su base de operaciones el propio robot. Por lo tanto, recibe los comandos del scratch y envía el valor de los sensores del robot.



*Comunicación Scratch-mOway. Susana Oubiña Falcón. (CC BY)*

### **Comandos (Scratch direccionado a mOway)**

Los comandos, en scratch 1.4, se envían con la orden “enviar a todos...” que se incluye en el bloque Control (ver siguiente imagen) de programación del scratch1.4.



*Bloques del Scratch 1.4. Susana Oubiña Falcón. (CC BY)*



*Orden “enviar a todos...” del bloque Control. Susana Oubiña Falcón. (CC BY)*

La orden “enviar a todos...” puede utilizarse con o sin variables, dando lugar a una serie de acciones que ejecutará el robot.

### **Sensores (mOway direccionado a Scratch)**

Cuando el robot está conectado, sus sensores están continuamente captando datos del entorno, del mundo real. Estos valores son direccionados al programa scratch a través de la aplicación “mOway Scratch”. Para visionarlos en el scratch hay que utilizar el comando “valor del sensor...” del bloque sensores del programa.

Se debe activar la conexión entre la aplicación y el programa scratch. Para ello es necesario “permitir las conexiones del sensor alejado”, proceso que se realiza de la siguiente forma:

1. Seleccionar el bloque “Sensores”.



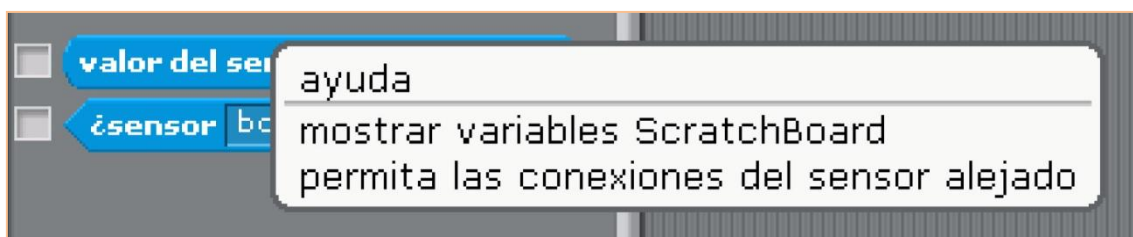
Selección del bloque “Sensores”. Susana Oubiña Falcón. (CC BY)

2. El bloque sensores presenta el comando “valor del sensor...”:



“valor del sensor...”. Susana Oubiña Falcón. (CC BY)

Nos situamos encima del comando “valor del sensor...” y hacemos clic con el botón derecho del ratón. Se nos abren las siguientes opciones:



Opciones del comando “valor del sensor...”. Susana Oubiña Falcón. (CC BY)

Finalmente, seleccionamos “permitir las conexiones del sensor alejado”.

### C. Elementos que se pueden programar

El robot mOway dispone de LEDs (5), altavoz, sensores (sonido, línea y obstáculos) y de dos ruedas para moverse. Todos estos dispositivos y elementos pueden ser programados para realizar diferentes acciones:

**1. Luces (leds):** mOway presenta 3 LEDs en su frontal y 2 LEDs de freno en su parte trasera. Todos se pueden controlar de forma independiente.



*LEDs delanteros del mOway. Imagen rescatada del Cuaderno de Prácticas (Scratch+mOway)*



*LEDs traseros del mOway. Imagen rescatada del Cuaderno de Prácticas (Scratch+mOway)*

Para hacer que estos LEDs se enciendan, se apaguen o parpadeen, se utilizan las siguientes acciones en el comando “enviar a todos...”:

<div> <div>enviar a todos</div> <div></div> </div>			
ledverde(on)	ledrojo(on)	ledfrontal(on)	ledfreno(on)
ledverde(off)	ledrojo(off)	ledfrontal(off)	ledfreno(off)
ledverde(parpadeo)	ledrojo(parpadeo)	ledfrontal(parpadeo)	ledfreno(parpadeo)
<div> <div>leds(on)</div> <div>leds(off)</div> <div>leds(parpadeo)</div> </div>			

*Órdenes para el control de luces de mOway. Susana Oubiña Falcón. (CC BY)*

**2. Altavoz:** El robot incluye un altavoz o zumbador para emitir sonidos. La orden requiere de una variable (variable “frecuencia”) que indique los hercios a los que vibrará la señal de sonido al transmitirse por el aire. Las órdenes que se pueden utilizar en el comando “enviar a todos...” son: *sonido(on)* y *sonido(off)*.



*Variable para controlar el sonido. Susana Oubiña Falcón. (CC BY)*

Presenta las órdenes “melodía(carga)” y “melodía(fallo)” que, respectivamente, hacen que suene la melodía “a la carga” y la melodía “fallo”.

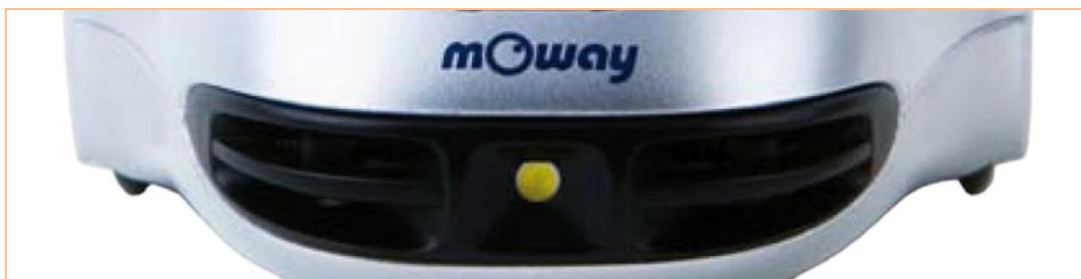
**3. Sensores:** Gracias a los sensores el robot mOway reacciona en su entorno físico con los elementos logrando detectar un obstáculo, un sonido o la luz. Los sensores que presenta mOway y que miden magnitudes son los siguientes:

- ✓ sensor sonido (micrófono, midiendo la cantidad de sonido recibido).
- ✓ sensor de línea (midiendo la cantidad de luz infrarroja para detectar el color, en escala de grises, del suelo en el punto en el que está el robot). Detecta contrastes fuertes (línea negra sobre fondo blanco), e incluso puede detectar tonos.



A mayores, mOway, internamente, lleva programado la función de seguir una línea negra sobre un fondo blanco. Para ello debemos usar las órdenes “seguirlinea(derecha)” o “seguirlinea(izquierda)”.

- ✓ sensor de obstáculos (Dispone de 2 emisores led infrarrojos y 4 receptores infrarrojos para detectar los objetos en la parte delantera del mOway. Mide la cantidad de luz infrarroja recibida, aportando valores numéricos entre 0 y 100). En el entorno scratch, mOway puede recoger los datos de luz infrarroja reflejada hacia sus 4 sensores de obstáculos receptores: lateral derecho e izquierdo y central derecho izquierdo.



*Detector de obstáculos del mOway. Imagen rescatada del Cuaderno de Prácticas (Scratch+mOway)*

A continuación se muestra un cuadro resumen que describe los comandos y órdenes en los diferentes sensores que dispone el robot mOway:

Sensores del mOway	Descripción
valor del sensor <input type="text" value="Microfono"/>	Nos ofrece un valor numérico, en % entre 0 y 100, del volumen del sonido detectado.
valor del sensor <input type="text" value="Linea derecho"/>	Nos ofrece un valor numérico, entre 0 y 100, que representa el color que detecta en el suelo, siendo 0 el color blanco y 100 el negro.
valor del sensor <input type="text" value="Linea izquierdo"/>	
enviar a todos <input type="text" value="seguirlinea(derecha)"/>	El robot mOway ajusta, automáticamente, su movimiento para seguir el contorno <b>derecho</b> (o <b>izquierdo</b> ) de la línea negra (sobre fondo blanco). Continuará en esta tarea hasta que reciba la siguiente orden de movimiento.
enviar a todos <input type="text" value="seguirlinea(izquierda)"/>	
valor del sensor <input type="text" value="Obstaculos central derecho"/>	Nos ofrece un valor numérico, entre 0 y 100, siendo 0 la usencia del obstáculo y 100 el umbral máximo de objeto detectado en el sensor <b>central derecho</b> (o <b>central izquierdo</b> o <b>lateral derecho</b> o <b>lateral izquierdo</b> ).
valor del sensor <input type="text" value="Obstaculos central izquierdo"/>	
valor del sensor <input type="text" value="Obstaculos lateral derecho"/>	
valor del sensor <input type="text" value="Obstaculos lateral izquierdo"/>	

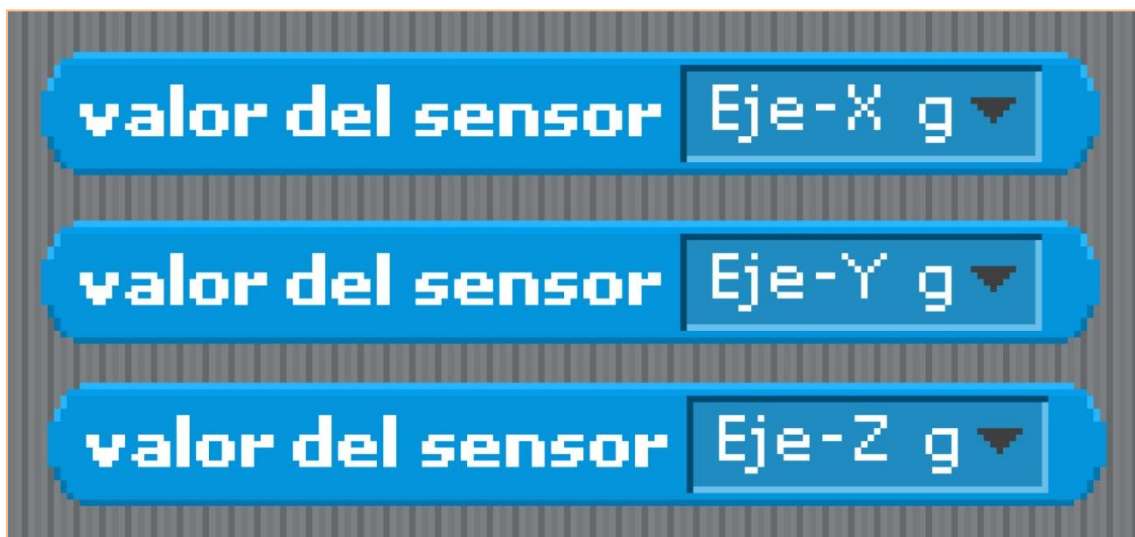
*Comandos para control de los sensores de mOway. Susana Oubiña Falcón. (CC BY)*

**4. Acelerómetro:** Este dispositivo que posee mOway nos permite detectar la inclinación que producimos en el robot, en los ejes X, Y y Z. Mide aceleraciones y las fuerzas inducidas por la gravedad: el movimiento y el giro.




*Posiciones del acelerómetro de mOway. Imagen rescatada del Cuaderno de Prácticas (Scratch+mOway)*

Los comandos que nos permiten conocer la lectura del acelerómetro en los diferentes ejes se muestran en la siguiente imagen y se incluyen dentro del bloque Sensores:



*Comandos de control del acelerómetro. Susana Oubiña Falcón. (CC BY)*

Es importante saber que, si inclinamos el robot mOway hacia la derecha (izquierda) el sensor de inclinación del eje X nos proporciona una medida negativa (positiva). Y, si lo inclinamos hacia delante (atrás), el sensor de inclinación del eje Y nos proporciona un valor positivo (negativo). Las diferentes posiciones e inclinaciones según su eje se observan en la siguiente imagen:

Inclinación	Delante y detrás	Derecha e izquierda	Arriba y abajo
Eje	Eje Y	Eje X	Eje Z
Posiciones			

*Posiciones, inclinaciones y ejes del acelerómetro. Imagen rescatada del Cuaderno de Prácticas (Scratch+mOway)*

**5. Movimiento:** El robot dispone de dos ruedas que le permiten avanzar, ir hacia atrás o girar. Este movimiento se puede controlar según las variables: velocidad, tiempo y distancia. Para realizar el movimiento, debe recibir desde el entorno scratch las órdenes a través del comando “enviar a todos (orden)” (dentro del bloque de control). Las órdenes en este comando se pueden dar con o sin variables, mostrándose en la siguiente tabla las diferentes opciones o posibilidades de actuación.

<div> <div>enviar a todos</div> <div></div> </div>			
Sin variables		Con variables	
Orden	Efecto	Orden	Efecto
<b>adelante</b>	Avanza recto de forma indefinida, con una velocidad del 50%	<b>adelante(recto)[1]</b> <b>adelante(izquierda)[2]</b> <b>adelante(derecha)</b>	Avanza hacia adelante según el valor numérico de una variable
<b>atras</b>	Retrocede recto de forma indefinida, con una velocidad del 50%	<b>atras(recto)</b> <b>atras(izquierda)</b> <b>atras(derecha)</b>	Avanza hacia atrás según el valor numérico de una variable
<b>izquierda</b>	Giro de 90° hacia la izquierda	<b>giro(izquierda) [3]</b>	Gira a la izquierda según el valor numérico de una variable
<b>derecha</b>	Giro de 90° hacia la derecha	<b>giro(derecha)</b>	Gira a la derecha según el valor numérico de una variable
<b>parar</b>	Para		
<b>mediavuelta</b>	Gira 180°		
<b>reset(distancia)</b>	Borra el contador de distancia total de los motores		

Órdenes de movimiento para mOway. Susana Oubiña Falcón. (CC BY)

[1] Con la orden *adelante(recto)*, mOway avanza recto a una velocidad determinada durante la distancia “distancia” o el tiempo “tiempo”. Si “tiempo” y “distancia” son iguales a cero, mOway avanzará indefinidamente. Con *atras(recto)* su comportamiento es similar a *adelante(recto)* pero en sentido contrario.



Variables que se usan en las órdenes *adelante(recto)* o *atras(recto)*. Susana Oubiña Falcón. (CC BY)

[2] Con la orden *adelante(izquierda)*, mOway avanza en una curva hacia la izquierda de un radio determinado a una velocidad determinada durante la distancia “distancia” o el tiempo “tiempo”. Si “tiempo” y “distancia” son iguales a

cero, mOway avanzará indefinidamente. Lo mismo ocurre con *adelante(derecha)* pero efectuando un giro hacia la derecha.



*Variables que se usan en las órdenes adelante(izquierda o derecha). Susana Oubiña Falcón. (CC BY)*

[3] Con la orden *giro(izquierda)*, mOway realiza una rotación a una velocidad determinada de un ángulo determinado hacia la izquierda, sobre su eje o sobre una rueda. Si el ángulo (“rotación”) es igual a cero, la rotación se realiza de manera indefinida. Lo mismo ocurre con *giro(derecha)* pero rotando hacia la derecha.



*Variables que se usan en las órdenes giro((izquierda o derecha). Susana Oubiña Falcón. (CC BY)*

Es importante conocer qué valores podemos introducir en las variables para que las órdenes se ejecuten correctamente. La siguiente tabla nos muestra ese rango de valores:

Variable	Descripción	Rango de Valores
distancia	Distancia a recorrer en mm	0 – 255 mm ( 0 – 25,5 cm )
frecuencia	Frecuencia de la señal del altavoz	0 – 16000 Hz
radio	Radio de curvatura	0- 100 (radio + velocidad < 100)
rotacion	Ángulo de rotación en grados	0 – 360 °
eje-rotacion	Eje de rotación	Wheel – Sobre rueda Cualquier valor – Sobre centro
velocidad	Velocidad de movimiento	30 -100 %
tiempo	Tiempo de movimiento en décimas de segundos	0 – 255 (0 - 25,5 segundos)

*Rango de valores de las variables. Imagen rescatada del “Manual Usuario mOway”.*

## 6. Otros subprogramas de mOway

mOway lleva incorporados unos pequeños programas. Dos de ellos ya los hemos mencionado, que son, “seguirlínea(izquierda)” y “seguirlinea(derecha).

Otros subprogramas interesantes son: “encerrado”, “empujar” y “defensor”.

- ✓ empujar: subprograma para detectar objetos, de modo que, el robot busca los objetos y los empuja.
- ✓ encerrado: subprograma para que el robot se quede encerrado dentro de un círculo negro. Avanza mientras esté en una superficie blanca pero, cuando llega a una línea negra, el robot retrocede.
- ✓ defensor: es una mezcla de los dos anteriores. Con este pequeño programa mOway expulsa objetos dentro del círculo negro.

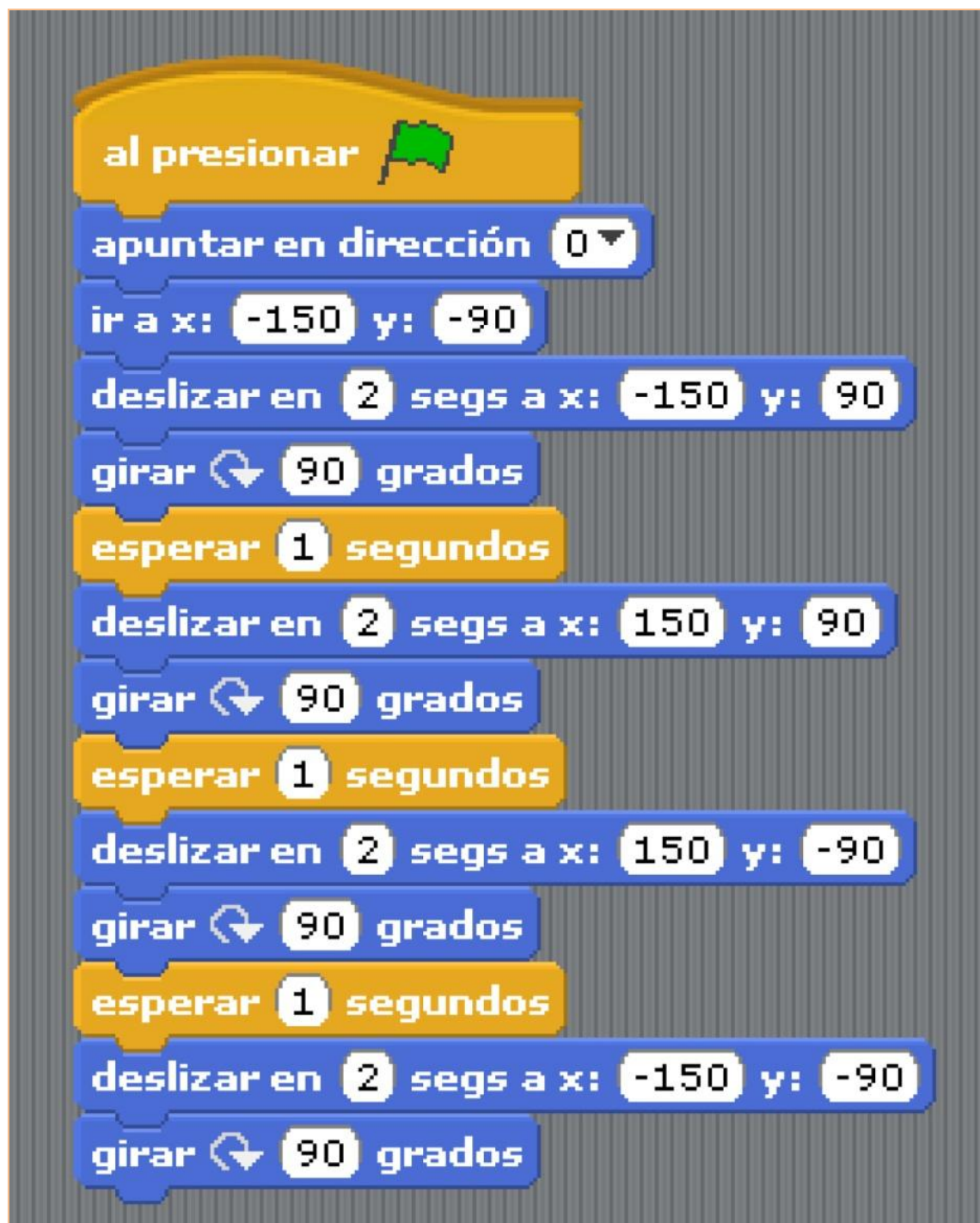
## D. Ejemplos

- ✓ Ejemplo 1: *“Movimiento del mOway formando un rectángulo o un triángulo equilátero”.*

Un ejemplo parecido a este se muestra en el libro “Cuaderno de prácticas. Guía para profesor” de scartch+mOway. Es muy representativo y por eso lo introduzco pero, a mayores le incluyo la creación de un triángulo equilátero. En este ejemplo, el objeto (mOway) y el robot mOway se mueven formando un rectángulo o un triángulo: el objeto lo hará por el escenario del scratch y el robot lo hará por una superficie física.

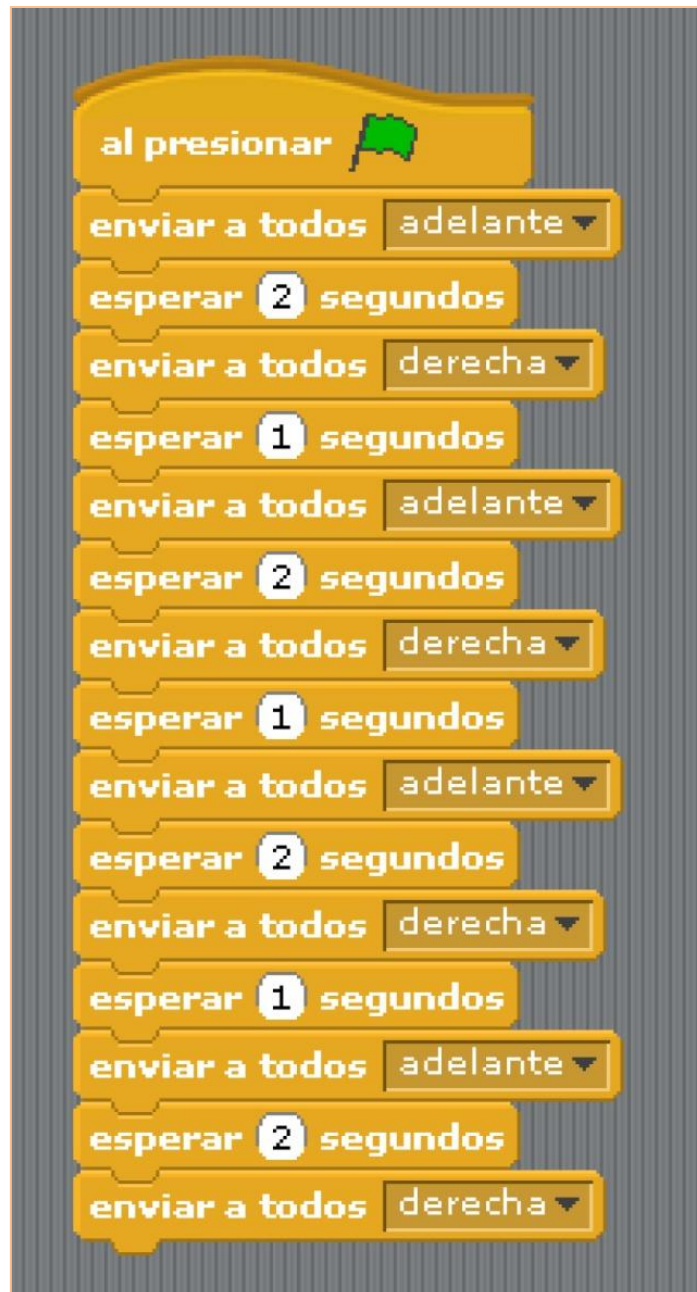
El script para el movimiento del objeto, para que realice un rectángulo sobre el escenario del scratch 1.4 sería:





*Objeto “mOway” formando un rectángulo (scratch1.4) Susana Oubiña Falcón. (CC BY)*

El script para el movimiento del robot mOway (sin interactuar con el objeto) para que se mueva formando un cuadrado sobre una superficie, sería el siguiente:



*Robot “mOway” moviéndose formando un rectángulo (scratch1.4) Susana Oubiña Falcón. (CC BY)*

Este script anterior utiliza el comando “enviar a todos...” para comunicarse por radio frecuencia con el robot. Cuando le envía “adelante”, anda; y cuando le envía “derecha”, gira 90° (necesitando para ello 1segundo).

El script que combina el objeto del programa con el robot físico es una mezcla de ambos programas. Se omite que el robot espere durante 2 segundos (moviéndose) ya que lo hará durante 2 segundos con la sentencia “deslizar en 2 segs a x:...y:...” una posición y que ejecuta el objeto en el entorno scratch:



Objeto y robot “mOway” moviéndose formando un rectángulo (scratch1.4) Susana Oubiña Falcón. (CC BY)

Finalmente, para que no sólo se mueva sino que lo dibuje en el escenario, utilizo los comandos del bloque “Lápiz”. El script definitivo, que se lleva a cabo cuando presiono la tecla “r” de rectángulo será:



Programa final rectángulo (scratch1.4) Susana Oubiña Falcón. (CC BY)

Para dibujar y moverse formando un triángulo equilátero, en nuestro caso cuando se presiona la tecla “t” de triángulo, creo una variable que llamo “rotación”. El objeto apunta al eje Y positivo, por lo que el primer ángulo de rotación (para conseguir que se mueva con ángulos internos de  $60^\circ$ ) debe ser de  $30^\circ$ , el segundo de  $120^\circ$  y el último, también de  $120^\circ$ . Para ello, en el comando “enviar a todos...” utilizo la orden “giro(derecha)”. El tiempo que le doy para que realice ese giro es de 1segundo. El programa final, para el triángulo, es el siguiente:

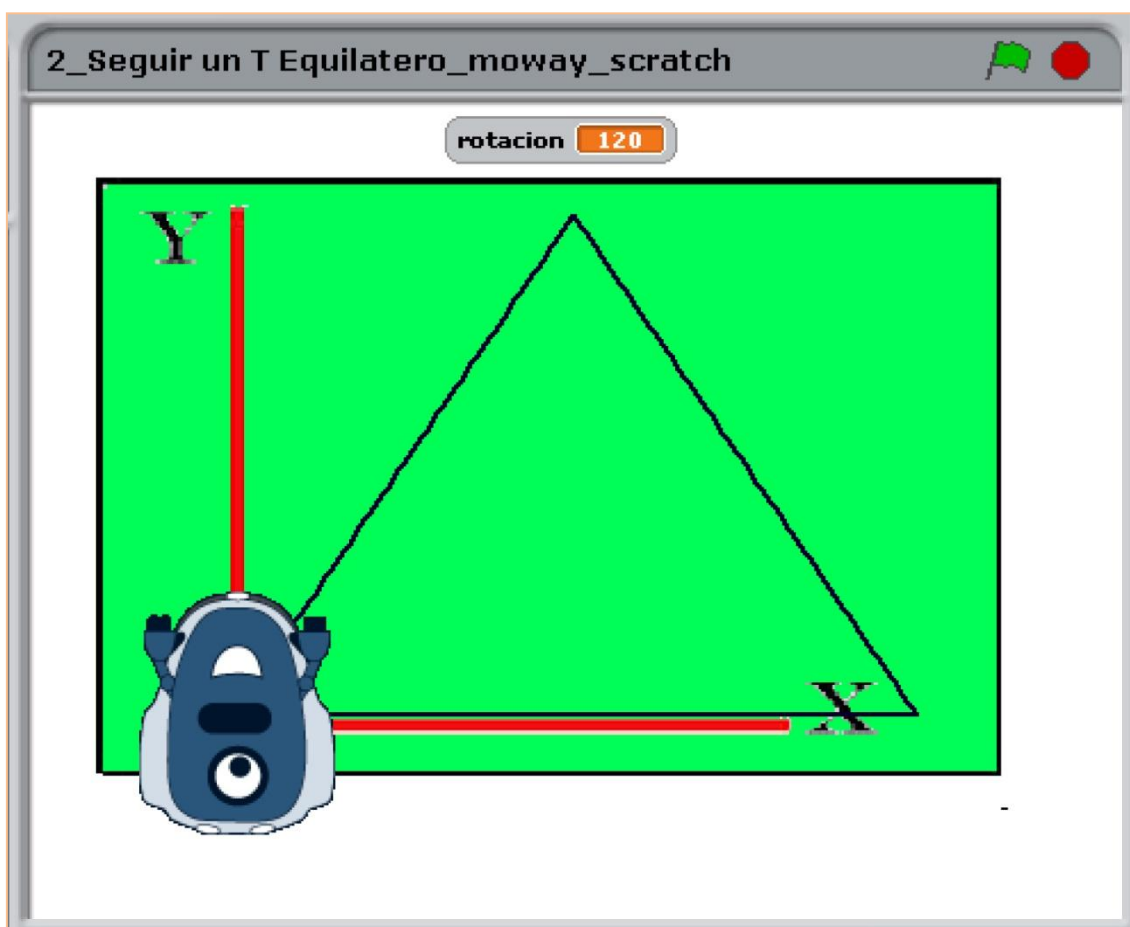




Programa final Triángulo Equilátero (scratch1.4) Susana Oubiña Falcón. (CC BY)



La imagen que veríamos dibujada en el escenario sería la siguiente:



*Dibujo de un Triángulo Equilátero (scratch1.4). Susana Oubiña Falcón. (CC BY)*

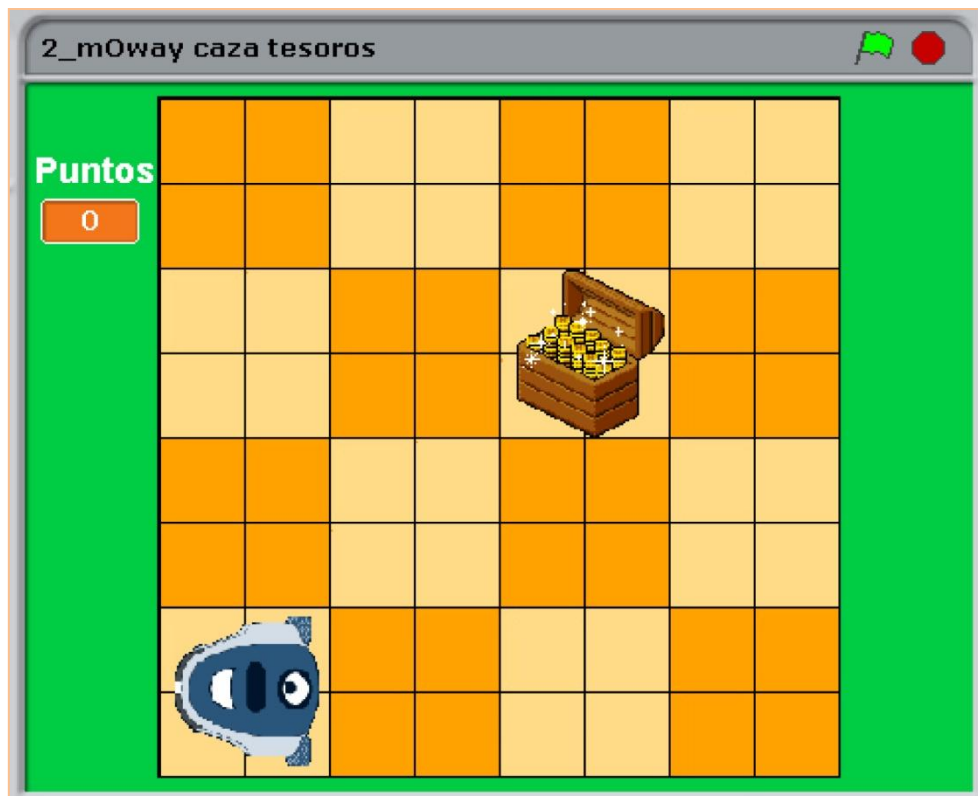
✓ Ejemplo 2: “mOway caza tesoros”

En este ejemplo el objeto “mOway” del juego se controla (maneja) con el robot físico mOway, a través de su acelerómetro programado en el eje X e Y, de modo que, si inclinamos el robot físico hacia la derecha (o izquierda) el objeto “mOway” del programa se moverá por el escenario hacia la derecha (o izquierda). De igual forma, si lo inclinamos hacia delante (o hacia atrás) el objeto “mOway” del programa subirá (o bajará) su posición Y por el escenario.

En el siguiente vídeo se muestra el manejo del robot con el juego:

[“mOway caza tesoros”](#). Susana Oubiña Falcón. (CC BY)

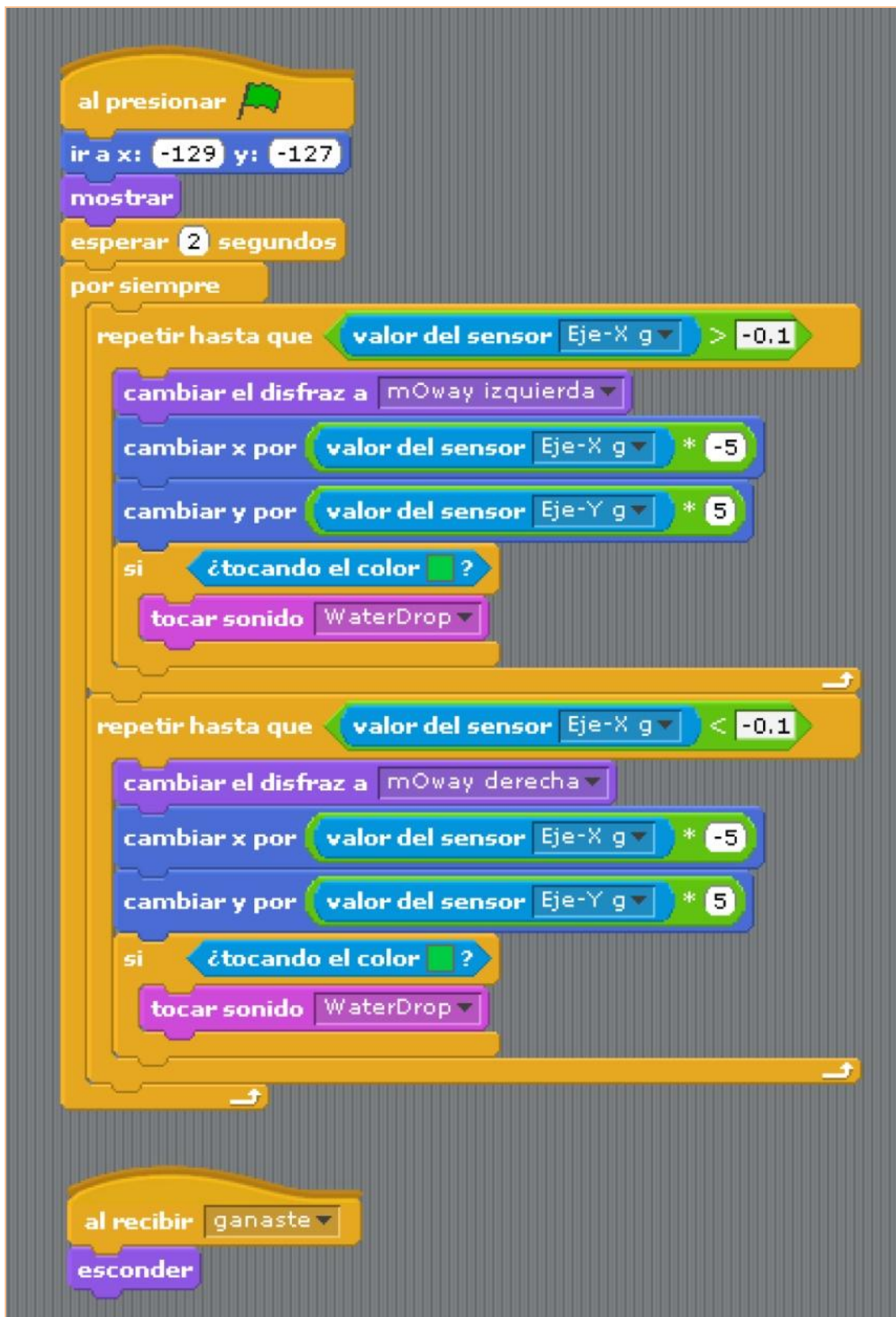
En la siguiente imagen puede verse el escenario del juego, un tablero en el que se mueven los objetos “mOway” y “tesoro”:



*Escenario de “mOway caza tesoros”. Susana Oubiña Falcón. (CC BY)*

El objeto “mOway” presenta el siguiente script:

Inicialmente se muestra en la posición que se indica en la figura anterior, primer cuadrado de la 4 fila (-129, -127) y se espera dos segundos para que nos de tiempo a coger el robot físico mOway. A partir de ahí realiza un bucle Por Siempre que dependerá del **valor del sensor del Eje-X g** y del **valor del sensor del Eje-Y g**.



Script del objeto “mOway” en el ejemplo 2. Susana Oubiña Falcón. (CC BY)

Si el robot no está inclinado, el sensor Eje-X g marca el valor -0.1. Al inclinar el robot hacia la derecha nos muestra un valor negativo (menor que -0.1) y hacia la izquierda, nos aporta un valor positivo (mayor que -0.1). Por lo tanto:

Si inclinamos el robot físico mOway hacia la derecha (izquierda), el valor del sensor Eje-X g será pequeño y negativo (positivo). Entonces, para que se desplace en el sentido del eje X positivo, deberé multiplicarlo por un valor negativo. Como quiero que se desplace de forma no muy rápida uso el valor numérico -5.

Además, si al robot físico mOway lo inclino hacia la adelante (atrás), el valor del sensor Eje-Y g toma un valor pequeño positivo (negativo). Entonces, para que se desplace en el sentido del eje Y positivo, deberé multiplicarlo por un valor positivo. Como quiero que se desplace de forma no muy rápida, también uso el valor numérico 5.

En cada caso, izquierda o derecha, cuando toque el tablero verde quiero que me avise y, para ello, sonará el sonido “WaterGrop”.

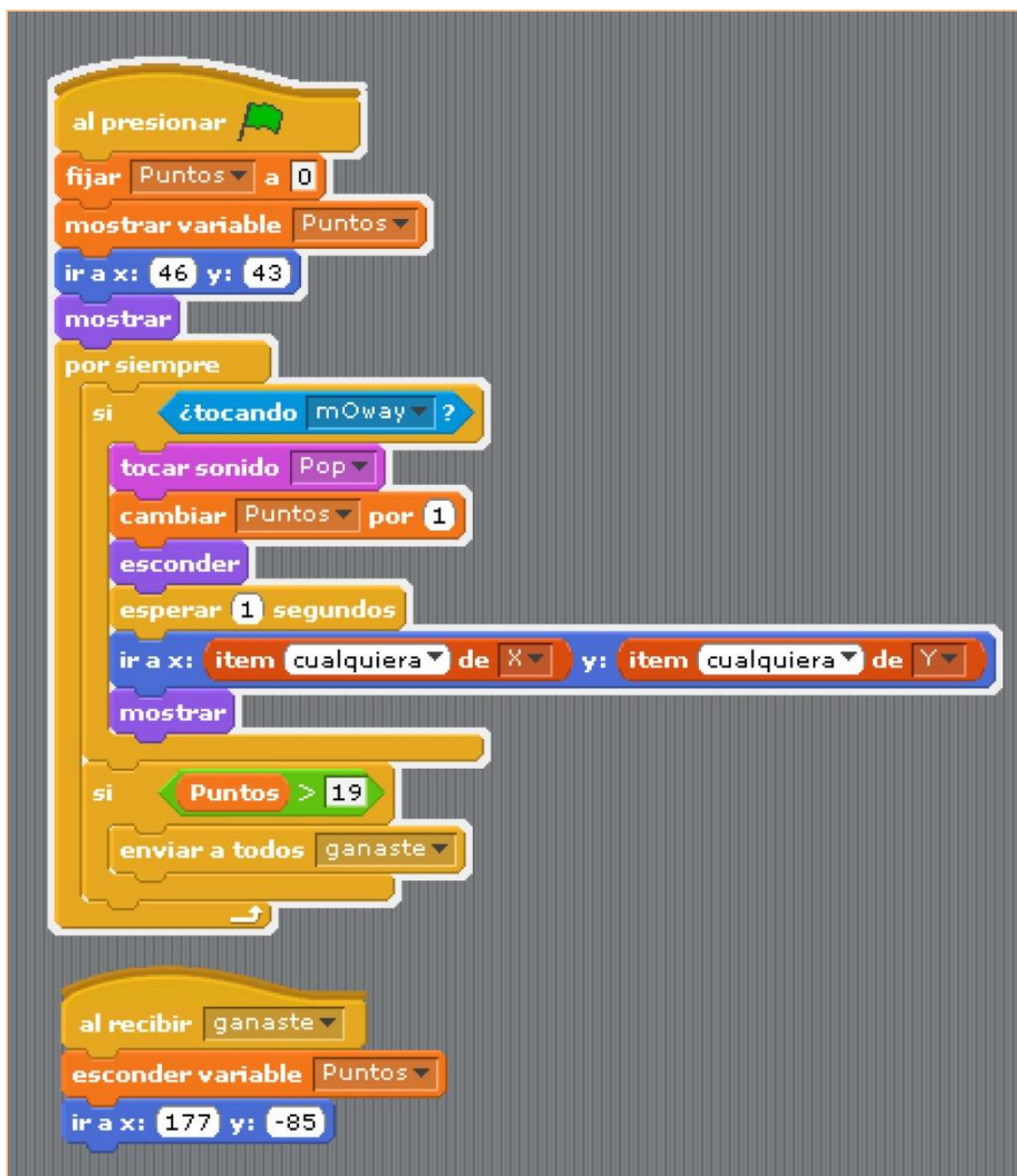
El objeto “tesoro” se muestra inicialmente en la posición (46, 43), que se corresponde con el cuadrado de la fila 2 y columna 3 del tablero).

El tablero está dividido en 16 cuadrados, de modo que, cuando el objeto “mOway” toque el objeto “tesoro”, éste debe esconderse (tocar el sonido pop y sumar 1 punto) y, tras un segundo, aparecer en una posición aleatoria relativa a cada uno de esos 16 cuadrados. Para ello creo 2 listas de datos: X (introduzco las posiciones del eje x para cada cuadrado) e Y (introduzco las posiciones del eje y para cada cuadrado), tal y como se muestra en la siguiente imagen:

X		Y	
1	-130	1	130
2	-40	2	42
3	45	3	-43
4	130	4	-129
+ longitud: 4		+ longitud: 4	

*Listas (posiciones X e Y). Susana Oubiña Falcón. (CC BY)*

Esta posición aleatoria la recoge de las listas X e Y, combinando cualquier posición de X con cualquiera de Y y mostrándose en ese punto, en ese cuadrado. Al superar los 19 puntos, el juego se acaba y envía el mensaje “ganaste” que cambiará el fondo, del tablero, a fondo “game over” y detendrá el programa. El script para el objeto “tesoro” se muestra en la siguiente figura:



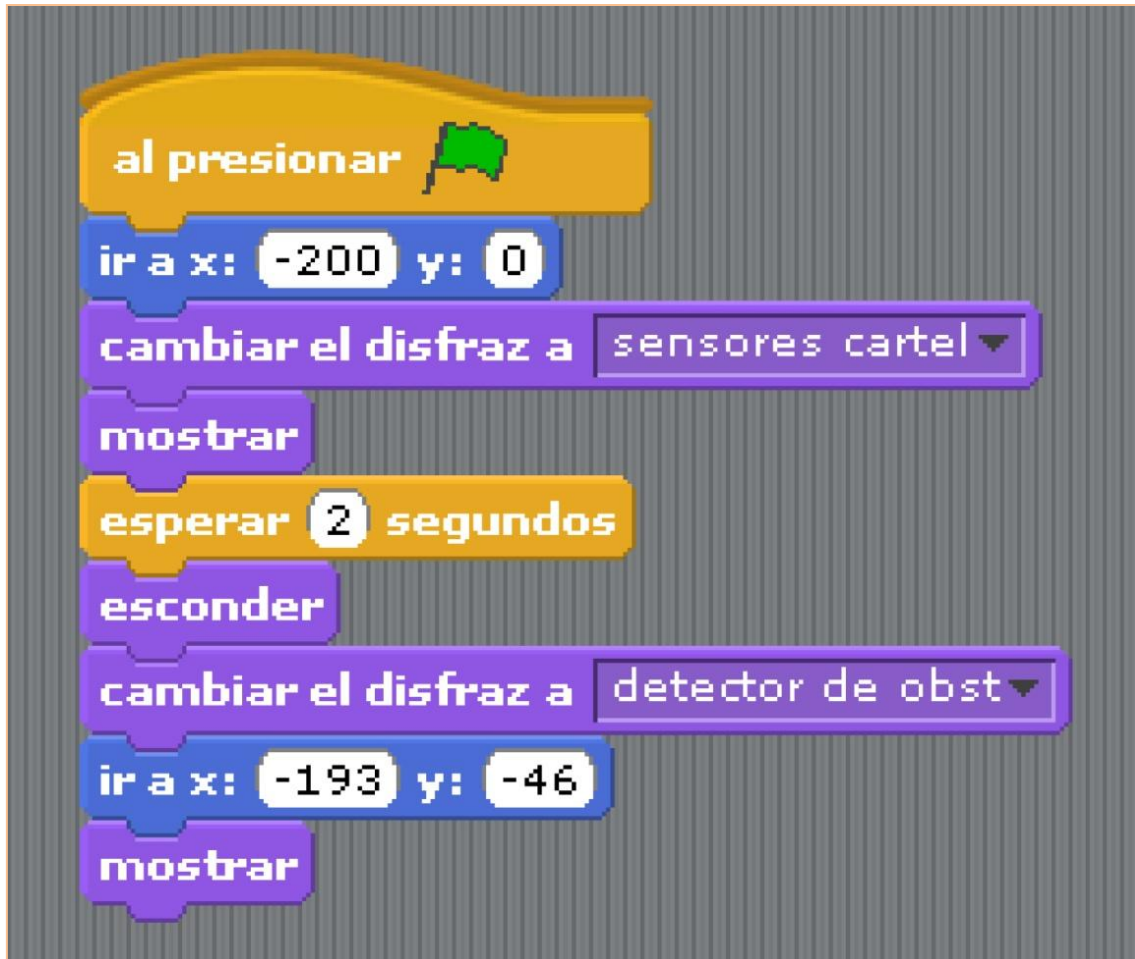
Script del objeto “tesoro” en el ejemplo 2. Susana Oubiña Falcón. (CC BY)

### ✓ Ejemplo 3: “Sensores de obstáculos”

El siguiente ejemplo se incluye en el *Cuaderno de Prácticas (Scratch+mOway)*. Lo introduzco en este curso porque lo considero un ejemplo idóneo para mostrar y observar la sensibilidad y efectividad de los 4 sensores infrarrojos

que posee el robot mOway. Voy a explicar la programación del proyecto “Sensor de obstáculos”.

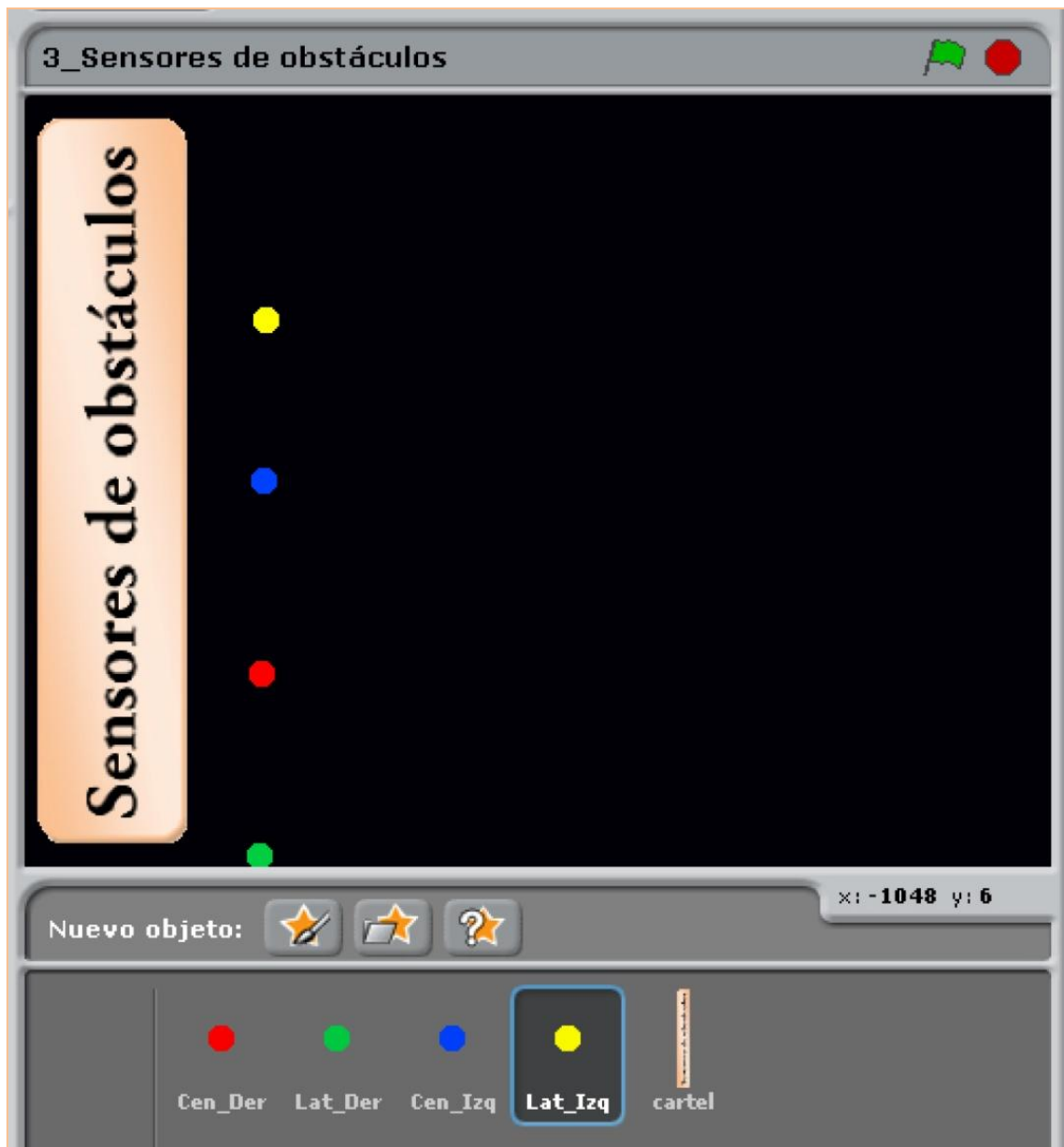
El proyecto original se compone de 4 objetos. En mi caso de 5, pero no he incluido nada original; sólo un objeto que llamo “cartel” y que sitúo en la posición fija (-200, 0) en su disfraz de cartel, disfraz que cambia a la imagen del detector tras pasar 2 segundos.



*Script para el objeto “cartel”. Susana Oubiña Falcón. (CC BY)*

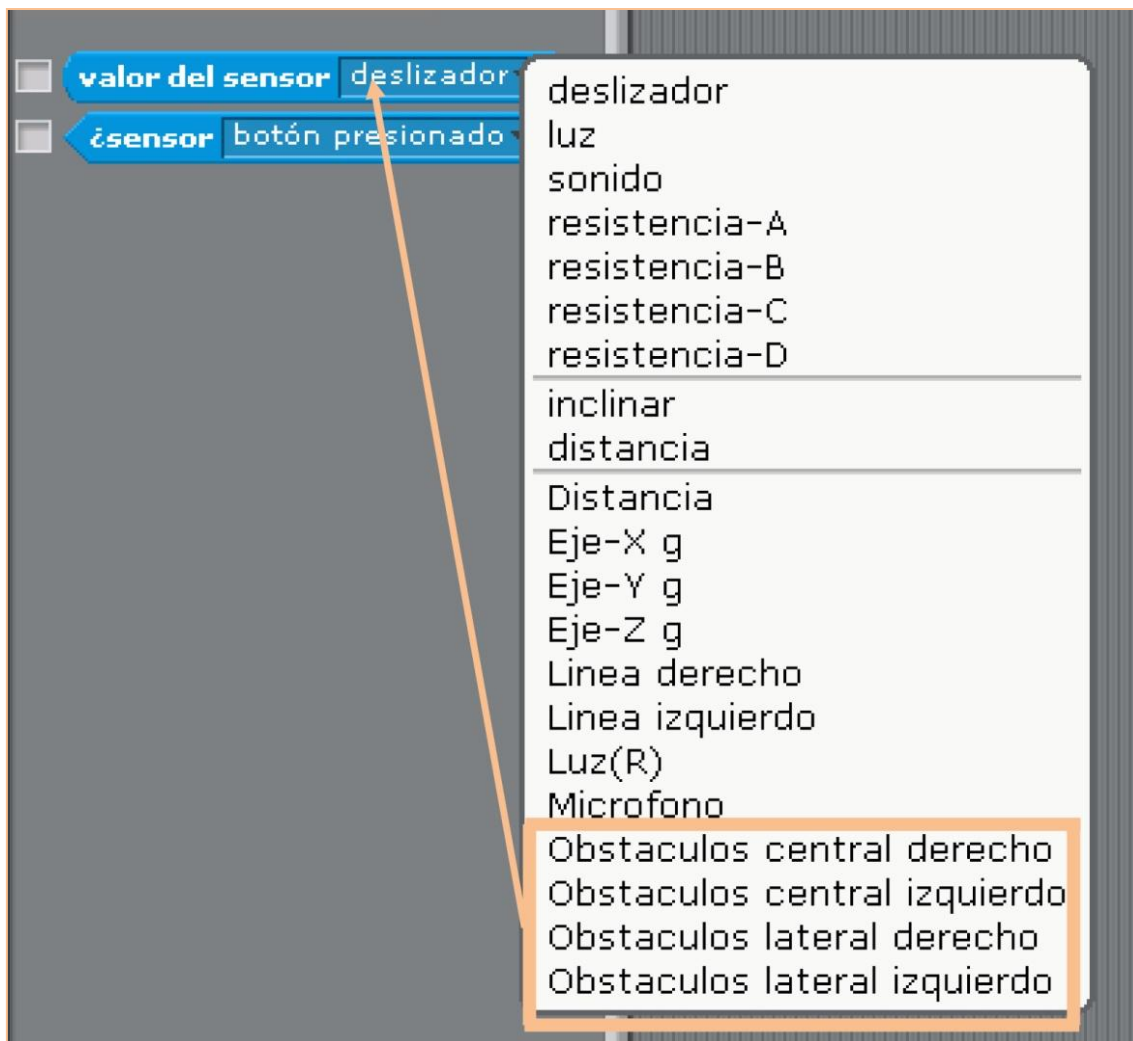
En la siguiente imagen, se observa el objeto cartel (con el disfraz inicial) con los 4 objetos a mayores que servirán para mostrar como varían las magnitudes de los sensores de obstáculos del mOway:





*Escenario del ejemplo 3. Susana Oubiña Falcón. (CC BY)*

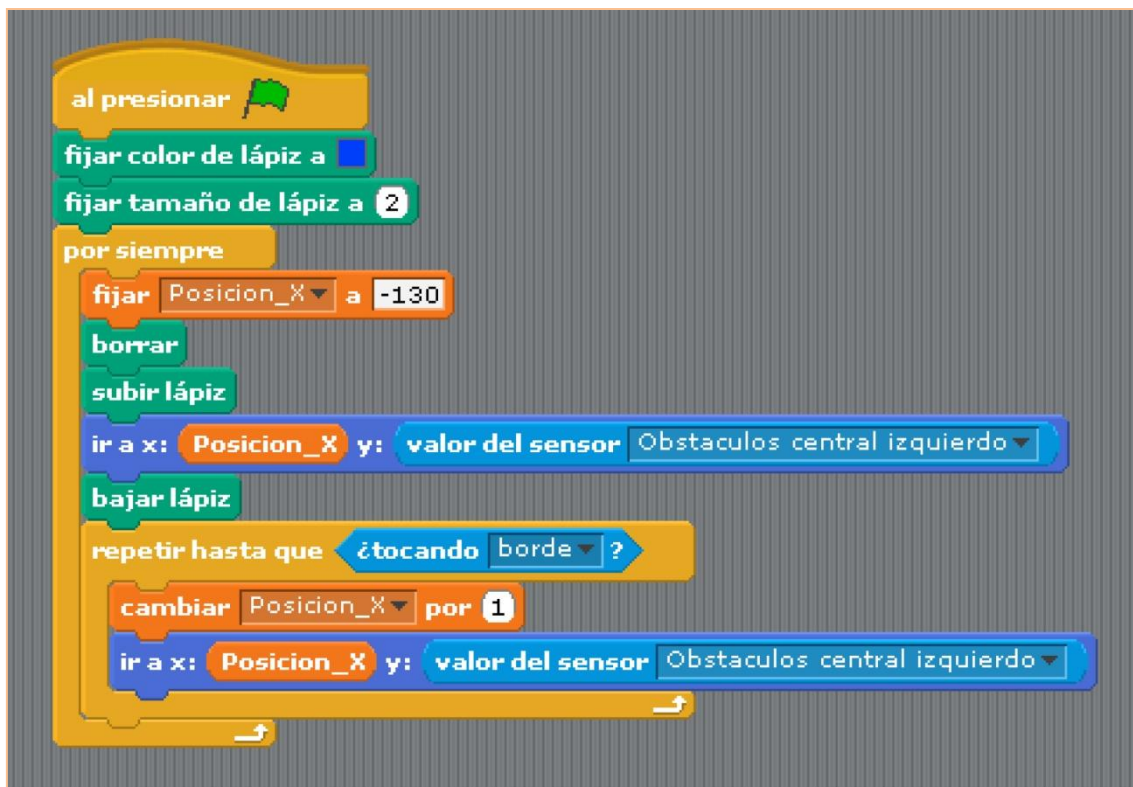
En el escenario de la figura anterior, vemos 4 objetos y, cada uno de ellos se encuentra simbolizado por un color determinado. El objetivo es que, al hacer correr el programa, representen y pinten en el escenario el valor numérico de su correspondiente sensor de obstáculos. El comando **valor del sensor...** utilizado en sus 4 opciones de detecciones de objetos, nos aporta un número entre 0 y 100, siendo 0 la ausencia del obstáculo y 100 el valor máximo (obstáculo detectado):



Opciones para el comando “valor del sensor...”. Susana Oubiña Falcón. (CC BY)

Con el objeto de color amarillo se representan las magnitudes (cantidad de luz infrarroja) detectadas por el sensor lateral izquierdo, el color azul recoge los valores aportados por el sensor central izquierdo, el color verde es para los valores del sensor lateral derecho y, finalmente, el color rojo muestra los valores del sensor central derecho.

El script para el sensor central izquierdo (azul), define un color azul de lápiz y tamaño del mismo, realizando un bucle Por Siempre (bucle continuo) partiendo de la posición del eje X fija de (-130), que es la que se observa en la imagen “Escenario del ejemplo 3”. A partir de esa posición del eje X, va representando la Y según el valor que le otorga el sensor de objetos central izquierdo. Dibuja esta posición hasta que toca el borde. En ese momento, aumenta la X y pintar el punto (x, y), siendo “y” el valor del sensor central izquierdo para cada aumento de x. De este modo, el script para el objeto que recoge los datos o mediciones de luz infrarroja reflejada en sensor central izquierdo es:

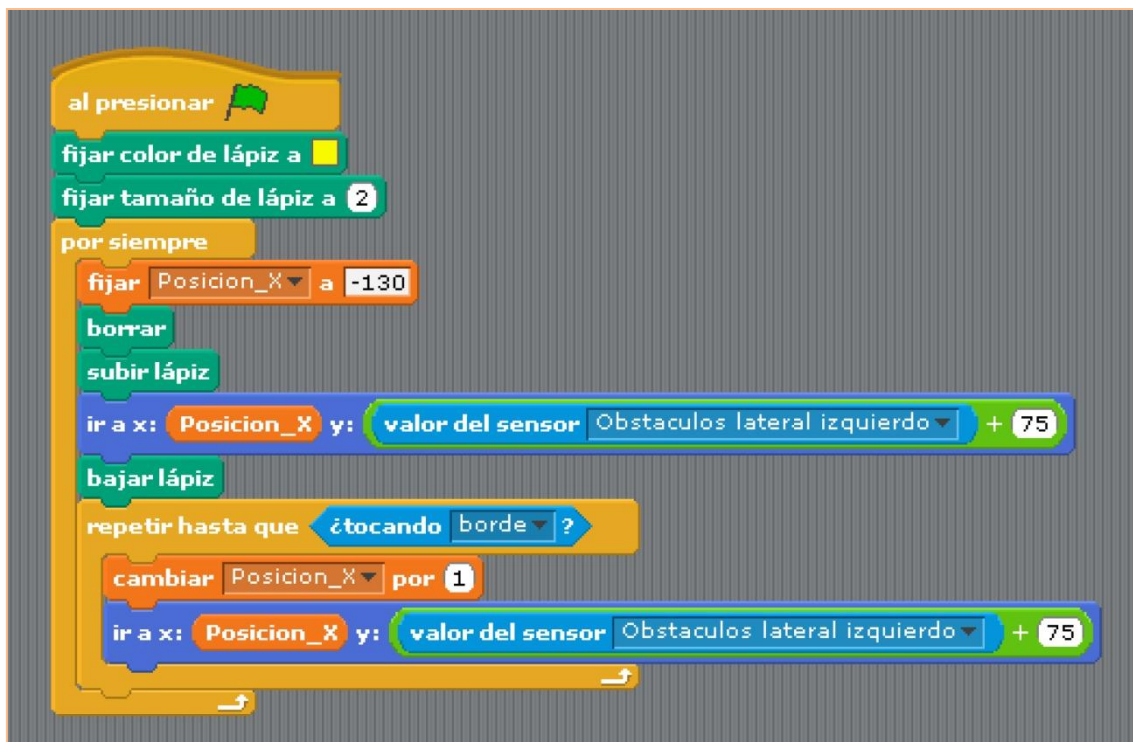


Script para el objeto Azul (Sensor Central Izquierdo). Susana Oubiña Falcón. (CC BY)

La gráfica aportada por el script anterior se representará en un determinado valor de X, pero la posición del eje Y depende del valor recogido por su sensor. De modo que, si no detecta un objeto, representará una línea recta continua hasta el borde lateral derecho del escenario.

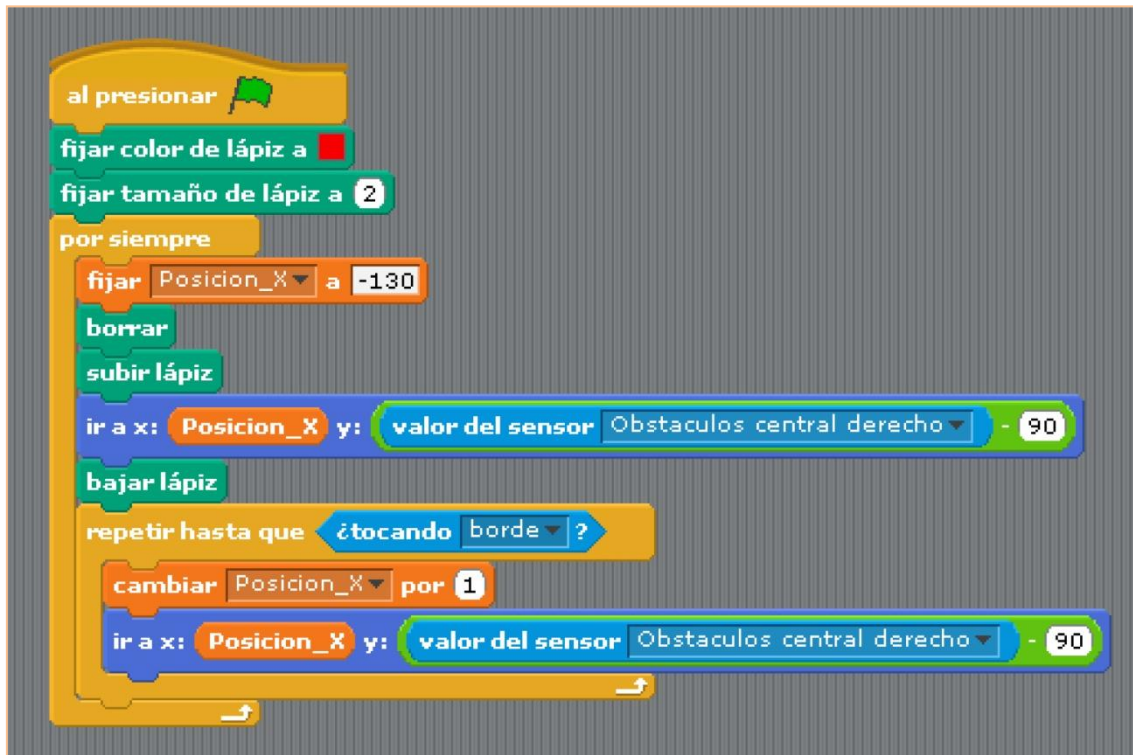
El programa no sólo detecta la cantidad de luz infrarroja reflejada en su sensor central izquierdo, sino que también recoge los valores numéricos de los otros 3 detectores. Por lo tanto, para evitar que las gráficas se solapen en demasía, simplemente, se traslada la posición del eje Y de los 3 objetos restantes de sensores de objetos, bien sea, aumentando en el eje Y o disminuyendo (+75, -90 y -175). Los scripts para el resto de objetos que pintan los valores de luz infrarroja reflejada detectada por sus sensores son:

El script para el sensor lateral izquierdo (amarillo, +75):



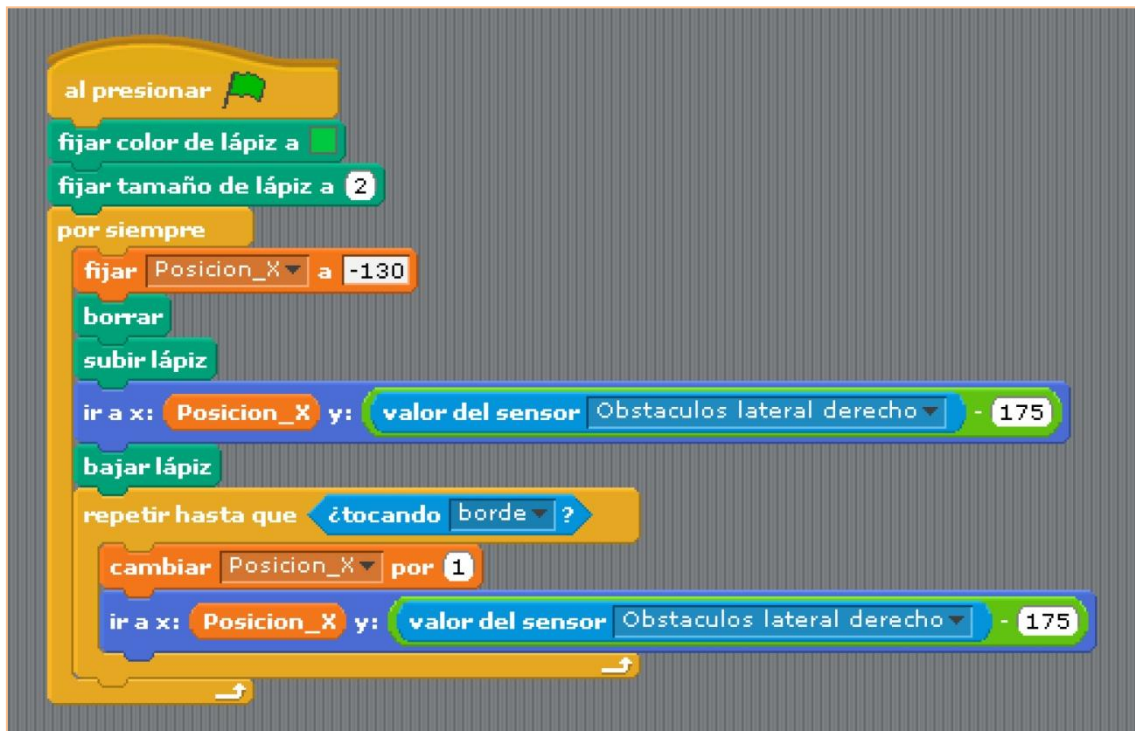
Script para el objeto Amarillo (Sensor Lateral Izquierdo). Susana Oubiña Falcón. (CC BY)

El script para el sensor central derecho (rojo, -90):



Script para el objeto Rojo (Sensor Central Derecho). Susana Oubiña Falcón. (CC BY)

Y finalmente, el script para el sensor lateral derecho (verde, -175):



Script para el objeto Verde (Sensor Lateral Derecho). Susana Oubiña Falcón. (CC BY)

✓ Ejemplo 4: “Seguidor de línea negra con control de luces”

Cuando se conecta la aplicación mOwayScratch al robot físico mOway mediante su cable USB, mOway se programa. Dos de sus programas (que se cargan en el robot) son: seguirlinea(derecha) y seguirlinea(izquierda). Por lo tanto, el ejemplo que vamos a implementar presenta una parte muy sencilla de realizar.

Por eso, a mayores vamos a incluir las luces, de modo que el robot encienda o apague sus 4 LEDs según nuestros deseos. En el programa, he querido que:

- ✓ Al presionar la “flecha derecha”, el robot encienda el LED verde
- ✓ Al presionar la “flecha izquierda”, el robot encienda el LED rojo
- ✓ Al presionar la tecla “espacio”, el robot encienda los LEDs de freno
- ✓ Y, al presionar la “flecha abajo”, el robot encienda el LED frontal

Para simularlo en el scratch he creado 5 disfraces para el objeto mOway, tal y como se muestra en la siguiente imagen:



*Disfraces para simular LEDs.* Susana Oubiña Falcón. (CC BY)

Además, quiero que el robot realice las siguientes acciones:

- ✓ Al presionar la “flecha derecha” quiero que siga el contorno derecho de la línea negra. Programa seguirlinea(derecha)
- ✓ Al presionar la “flecha izquierda” quiero que siga el contorno izquierdo de la línea negra. Programa seguirlinea(izquierda)
- ✓ Al presionar la “flecha abajo” quiero que gire 180 (el robot físico necesita de un tiempo para hacerlo, he puesto 2 segundos). Para que siga la línea debo presionar las teclas derecha o izquierda.
- ✓ Al presionar la tecla “espacio” quiero que el robot se pare.

Su funcionamiento se observa en el siguiente vídeo: [“Seguir línea negra con control de luces”](#) . Susana Oubiña Falcón. (CC BY)

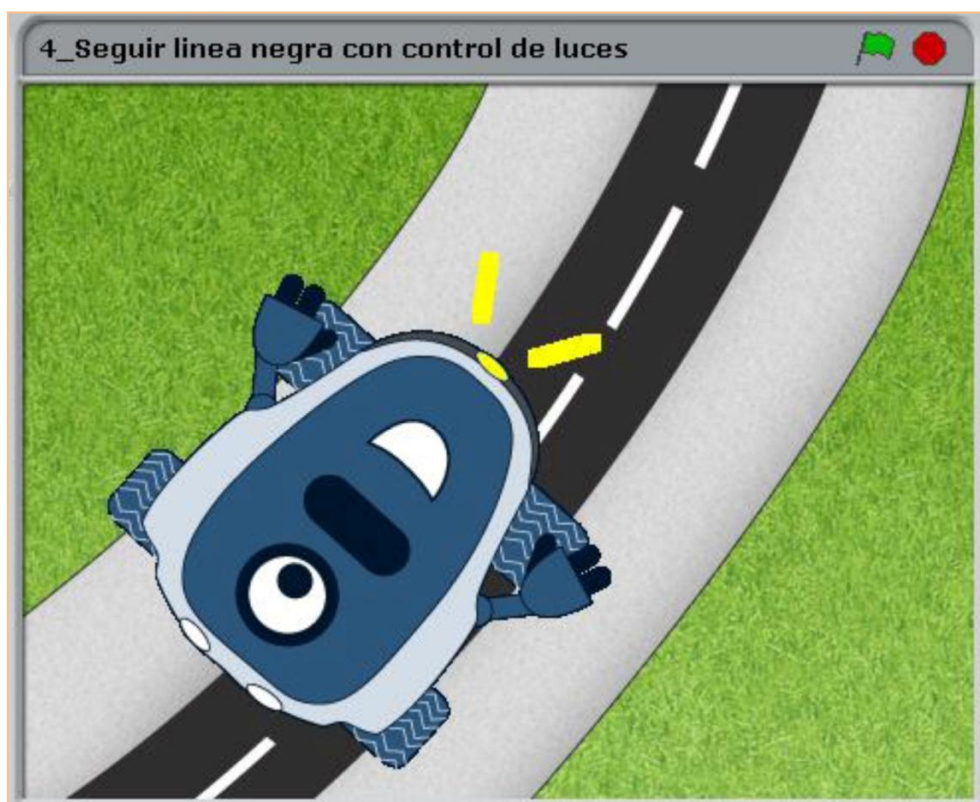
El script que cumple con mis especificaciones de encendido y apagado de los LEDs así como, de las acciones que me he propuesto es:





Script que implementa el Ejemplo 4. Susana Oubiña Falcón. (CC BY)

Uno de los escenarios que se dan en la ejecución del programa es el siguiente:



mOway con luz frontal encendida. Susana Oubiña Falcón. (CC BY)

✓ Ejemplo 5: “Variables en el movimiento”

En el ejemplo 1, con el triángulo equilátero, hemos utilizado la variable “rotación” para hacer que mOway gire unos determinados grados (y no necesariamente 90°). También podemos hacer que ande sólo una determinada distancia e incluso, que lo haga a una velocidad determinada.

*Variable distancia*

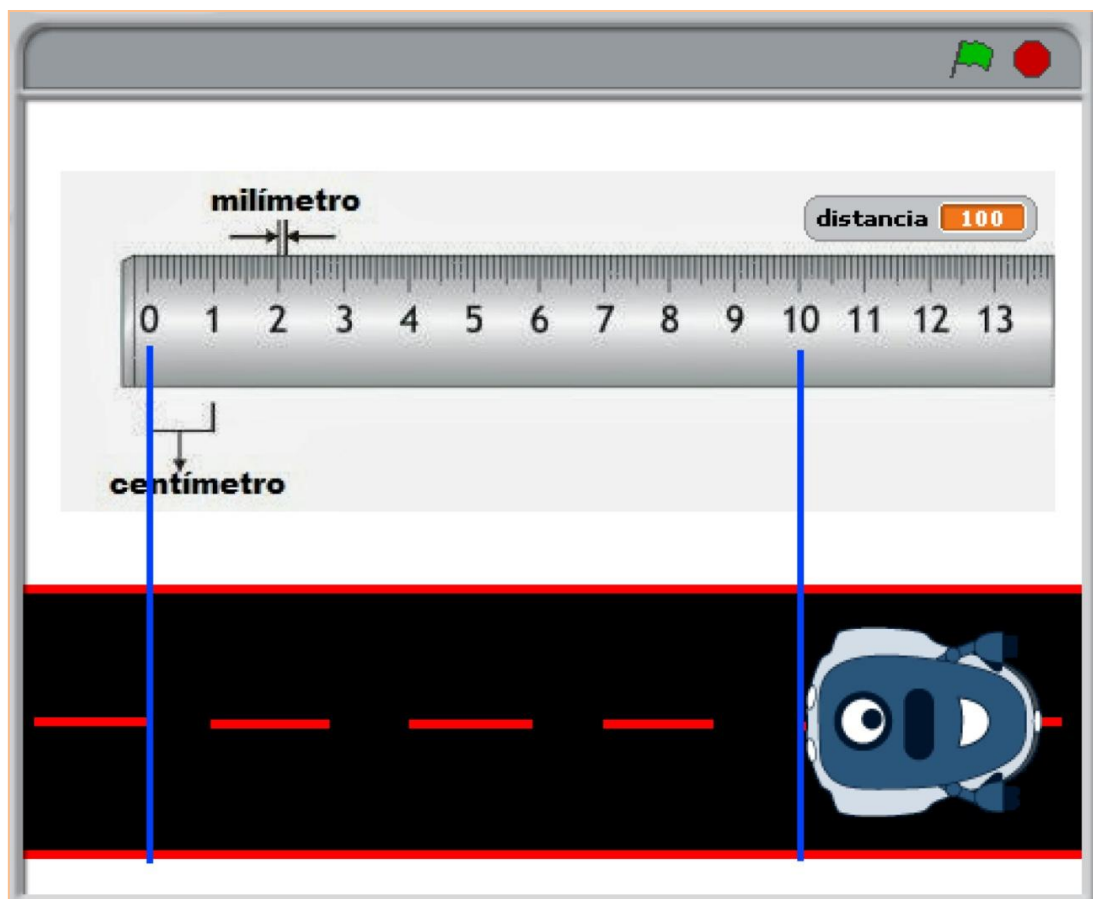
El rango de valores que podemos usar con la variable distancia es de cero a 25,5cm. En la siguiente imagen observamos un pequeño script. Programa que hace que el robot físico mOway se mueva de forma recta una distancia de 100mm, es decir, 10cm. Para ello hemos creado la variable “distancia” y, simplemente, la hemos fijado a la cantidad deseada:



*Variable “distancia” en el robot mOway. Susana Oubiña Falcón. (CC BY)*

En el siguiente vídeo puede verse como el robot mOway se desplaza esos 10cm: [“Control de distancia con mOway”](#). Susana Oubiña Falcón. (CC BY)

Si queremos que en el escenario de nuestro programa scratch un objeto mOway se desplace 100mm (10cm) y así tener una simulación en el ordenador, debemos introducir más comandos. El programa, así como, un posible escenario, podrían ser los siguientes:



*Escenario en el ejemplo 5 para mOway. Susana Oubiña Falcón. (CC BY)*



*Variable “distancia” en el objeto mOway. Susana Oubiña Falcón. (CC BY)*

Inicialmente, parte del punto (-127, -100) y que corresponde a la primera línea azul del escenario que marca 0cm en la regla. En esta posición inicial, la parte

trasera del objeto mOway se sitúa en la marca 0cm. Tras dejar pasar 1 segundo, la parte trasera del objeto mOway debe posicionarse en la segunda línea azul del escenario, es decir, en la posición (167, -100).

En el siguiente vídeo puede verse la simulación del desplazamiento del objeto mOway en el escenario del entorno scratch: [“Control de distancia con mOway en simulación con Scratch”](#). Susana Oubiña Falcón. (CC BY)

### *Variable velocidad*

El rango de valores que podemos utilizar con la variable “velocidad” es del 30% al 100%. Por defecto, mOway se mueve al 50%. Si queremos que se mueva de forma más lenta o que sea más veloz, debemos, respectivamente, disminuir o aumentar el valor numérico de la variable “velocidad”.

En el siguiente script se ejemplifica el uso de la variable “velocidad”. Simplemente, el robot se mueve durante 2 segundos, de forma recta a una velocidad del 30%, para después, hacerlo con una velocidad de 80% durante ese mismo tiempo y finalmente parar.



*Variable “velocidad” en el robot mOway.* Susana Oubiña Falcón. (CC BY)

En el siguiente vídeo puede verse como el robot ejecuta el script anterior: [“Ejemplo simple de control de velocidad con mOway+Scratch”](#). Susana Oubiña Falcón. (CC BY)

### 3.2. Soy alumno/a scratcher

El lenguaje de programación Scratch es bastante intuitivo favoreciendo el acercamiento de nuestros alumnos al mundo de la lógica computacional y programación. Presenta una interfaz atractiva y sencilla, muy fácil de manejar y con un gran abanico de posibilidades para ser utilizada en proyectos de diferentes asignaturas o materias. Podemos decir que **Scratch es una herramienta ideal para el alumnado de primaria y secundaria** permitiéndole desarrollar la creatividad, habilidades comunicativas, de escritura y lectura, trabajar en cooperación, resolución de problemas, experimentar y explorar en muy diversas temáticas. Se han creado algunos proyectos al respecto. Uno de ellos es el proyecto "Scratch Eguna":

*Scratch Eguna* [1] es un proyecto que persigue introducir scratch como herramienta educativa en las aulas de Primaria. En el proyecto se forma a docentes, independientemente de sus conocimientos de partida. El único requisito de los mismos es que deseen experimentar con esta herramienta en sus clases y con su alumnado.

La siguiente presentación hace referencia al proyecto Scratch Eguna:

Pablo Garaizar. *Scratch Eguna: From Scratch Day to Scratch every day* (CC BY-SA)

[1] L.M I. Albarrán. "Scratch Eguna": *Acercando Scratch a las aulas, desde la Educación Primaria (II)*". Red de Buenas Prácticas 2.0. [En línea]. Disponible en: <http://recursostic.educacion.es/heda/version/v2/es/primaria/995-scratch-eguna-acercando-scratch-a-las-aulas-desde-la-educacion-primaria-ii>. [Accedido el 4 de mayo de 2014].

#### 3.2.1. Creación de videojuegos sencillos

Con un juego, al jugar, el alumno/a aprende sin darse cuenta; de una forma menos tradicional, menos formal. Engloba conceptos como participación social y colectiva, competitividad y repetición de una acción para obtener un logro, mérito y recompensa.

Los juegos son una herramienta muy útil en la educación para conseguir que los alumnos se sientan atraídos en positivo, modificando actitudes y logrando que la escuela, algo que por lo general les parece aburrido, les sea divertido.

A continuación presentamos algunos juegos:

- ✓ Ejemplo 1: "Un día de xantar"

<http://scratch.mit.edu/projects/24160595/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: "[Un día de xantar](http://scratch.mit.edu/projects/24160595/)".



- ✓ Ejemplo 2: “Cangrexo versus polbo”

<http://scratch.mit.edu/projects/23991606/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: “[Cangrexo versus polbo](#)”.

- ✓ Ejemplo 3: “Naves invasoras del espacio”

<http://scratch.mit.edu/projects/23939330/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: “[Naves invasoras del espacio](#)”.

- ✓ Ejemplo 4: “Atrapa al robot”

<http://scratch.mit.edu/projects/23377435/>

Siéntete libre de utilizarlo para experimentar, modificar y crear un programa derivado. El enlace para proceder a ello es el siguiente: “[Atrapa al robot](#)”.

### 3.2.2. Comprensión de contenidos

Nuestro alumnado puede utilizar el programa scratch para asimilar y profundizar en diferentes contenidos de nuestra área. Todos, como docentes, hemos vivido la experiencia de presentir que un alumno/a en concreto entiende un concepto o un desarrollo que hemos explicado, pero tras ahondar un poco, nos hemos dado cuenta que no era cierto y lo que el alumno/a si sabe hacer es una mecánica que le lleva a presentarnos un resultado acertado. Esa forma automática de hacer las cosas, sin una comprensión real ni una reflexión al respecto no se traduce es un aprendizaje significativo. Es más, al poco tiempo se olvida y se pierde.

En este sentido, programar en scratch ayuda a afianzar el aprendizaje activo porque el alumno/a ha de tener muy claro qué quiere hacer, qué lógica debe seguir para conseguirlo y cómo lo hará, investigando para superar los problemas que le vayan surgiendo. De esta forma, el aprendizaje no se olvida fácilmente.

En este apartado, como ejemplo, me he decidido por un programa que podría encajar en las materias de matemáticas, física y química e incluso tecnología. Un programa similar a mi propuesta, mejor o peor, podría ser desarrollado por nuestro alumnado.



- ✓ Ejemplo: "*Cálculo de la resultante de dos vectores*". El programa calcula, de forma gráfica y analítica, el vector Resultante de 2 vectores. Esos dos vectores los introduce el jugador (módulo y ángulo con el eje X). Con esos datos, el programa nos aporta el vector resultante y sus parámetros.

El link para acceder al proyecto es el siguiente:

<http://scratch.mit.edu/projects/25508032/>

[\*Cálculo de la resultante de dos vectores\*](#) .Susana Oubiña Falcón. (CC BY)

- ✓ Explicación del programa:

Un alumno puede programar la resultante de la suma de dos vectores por el método analítico o por el método gráfico. Al hacerlo, no sólo asimila el concepto resultante de dos vectores, sino que también comprende otros contenidos como módulo, componentes de un vector y sus operaciones. El programa refuerza el pensamiento lógico.

El programa podría ser el siguiente:

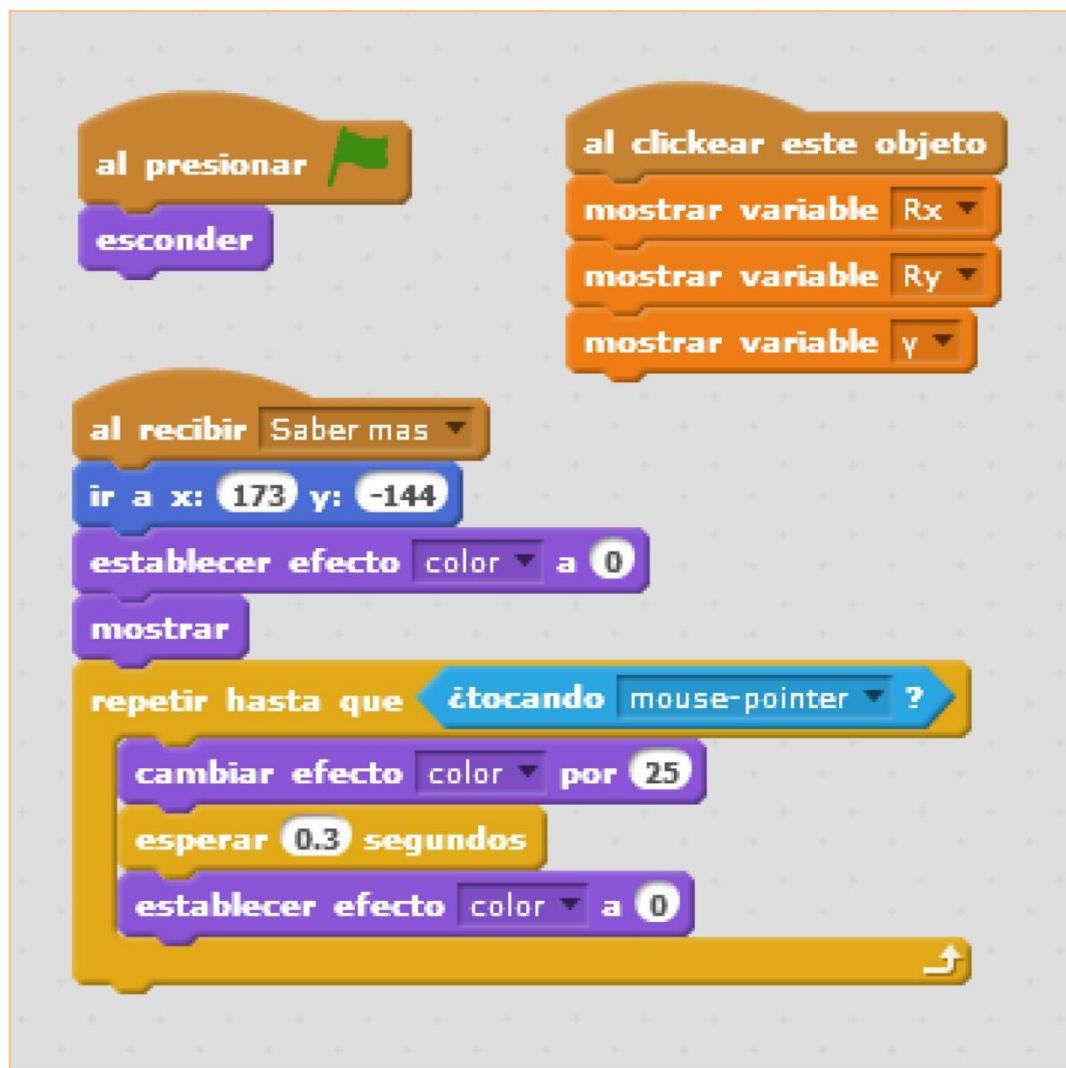
Como objetos utilizamos 3: botón inicio, punto y botón saber más. El escenario será el que tiene por defecto scratch para representaciones gráficas "xy-grid". Los programas para el botón inicio y el botón saber más son muy sencillos y son los siguientes:



*Script para el botón "Inicio". Susana Oubiña Falcón. (CC BY)*

Al presionar la bandera verde, borra el escenario y muestra el botón en la posición (185,-128). Después, dice el texto "Para comenzar, haz clic aquí".

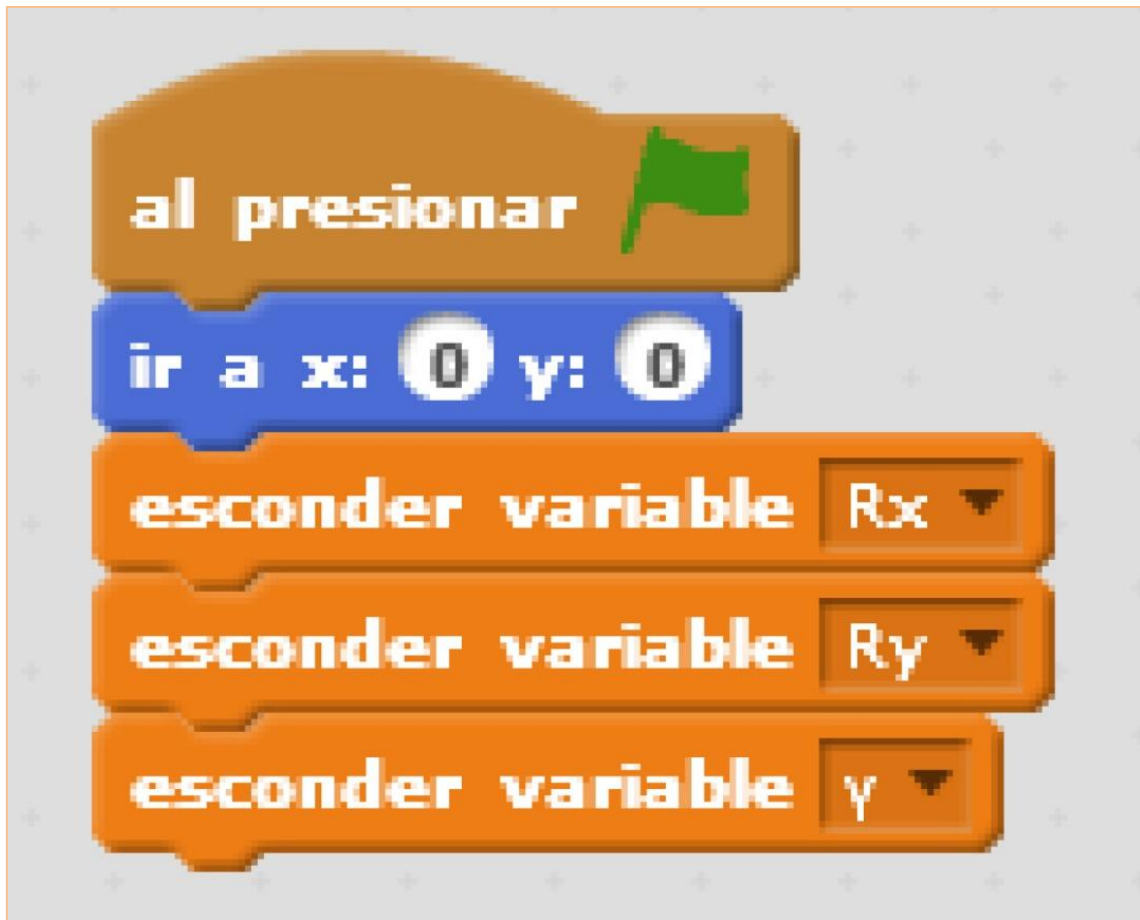
Al hacer clic en el objeto botón Inicio, se esconde (se deja de ver) y envía el mensaje "Inicio". Este mensaje lo recoge el objeto "punto".



Script para el botón “Saber más”. Susana Oubiña Falcón. (CC BY)

El botón “Saber más” sólo debe visionarse al final. Por lo tanto, al iniciar el programa, debe estar escondido u oculto. Cuando el objeto “Punto” representa la resultante de dos vectores, enviará el mensaje “Saber más”. Ese mensaje es recibido por el botón y lo que hace es hacerlo visible en la posición (173, -144). Para animar al usuario a hacer clic en ese objeto con el puntero del ratón, creamos un bucle que se repita mientras el puntero del ratón no pinche en el botón. Ese bucle es un efecto de color, variación de color. Finalmente, cuando hace clic en el botón, se mostrará el valor de las componentes Rx, Ry de la resultante y el ángulo que forma el vector con el eje x.

La programación del objeto “Punto” es el meollo del programa. Más laboriosa pero no compleja. Se crea por pasos mediante envíos y recepciones de mensajes.



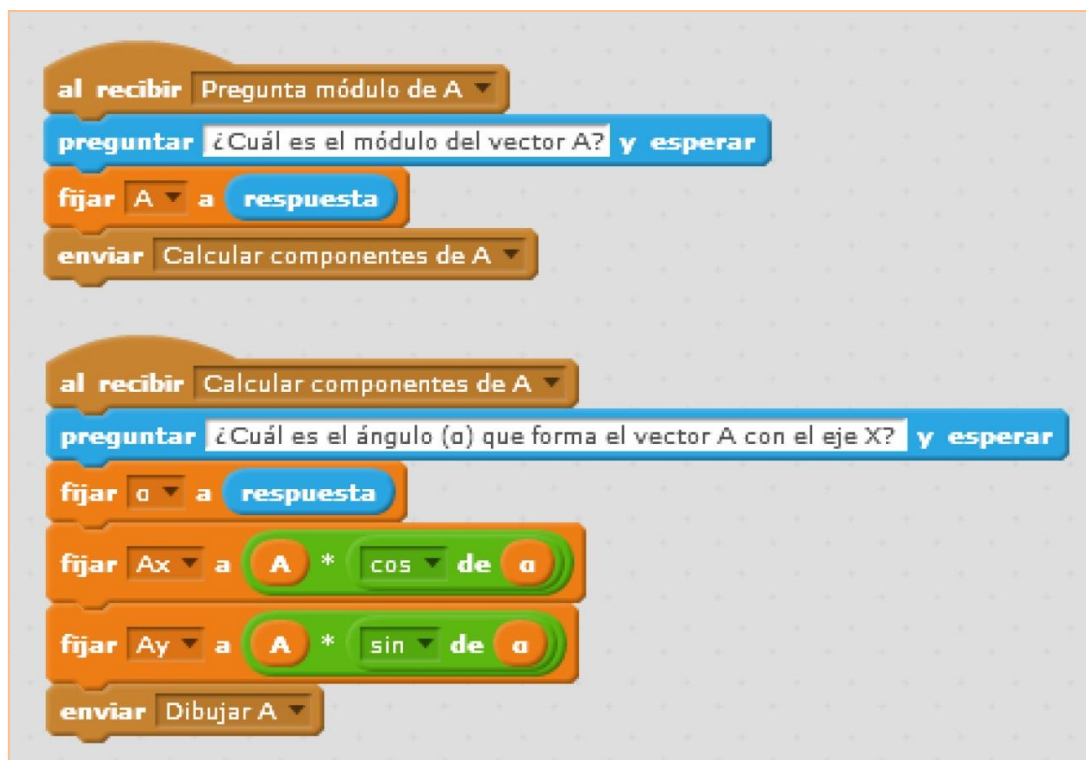
*Al presionar la bandera verde.* Susana Oubiña Falcón. (CC BY)

Sitúa el punto en el origen de coordenadas y esconde las variables Rx, Ry y el ángulo (que se han mostrado con el botón “Saber más”).



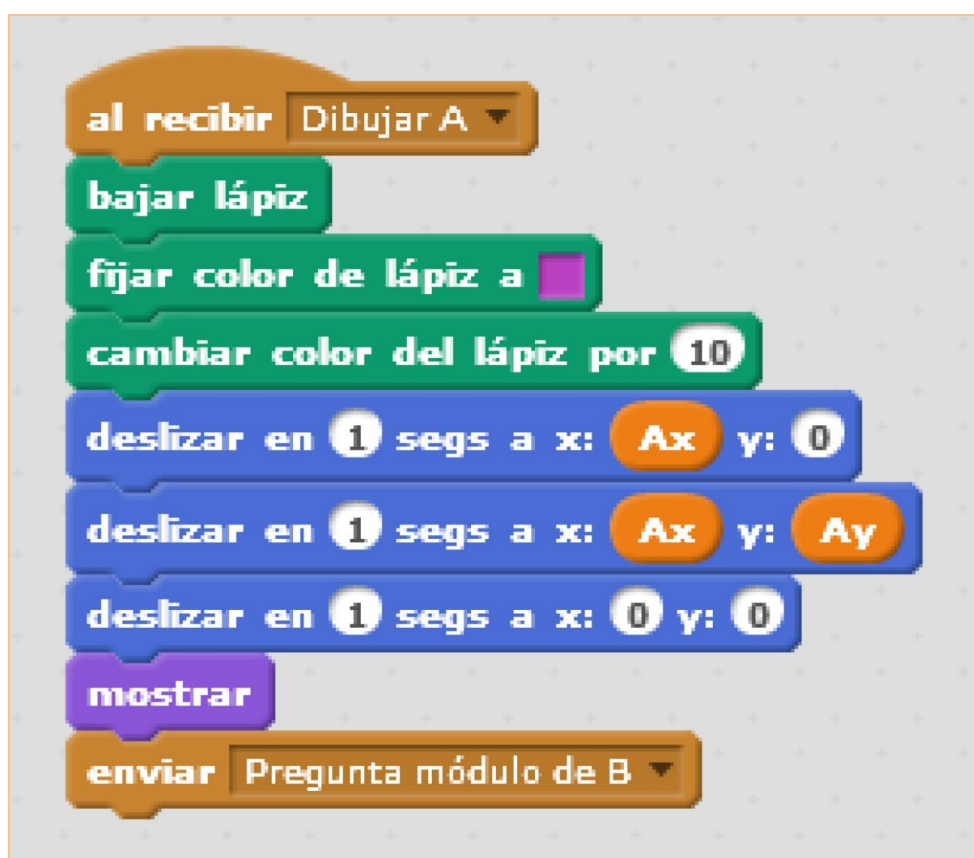
*Script del mensaje "Inicio!". Susana Oubiña Falcón. (CC BY)*

Borra el escenario y fija un tamaño para el lápiz con el que dibujará. Inicializamos todas las variables a cero y comienza a pedir datos del vector A mediante mensajes: primero su módulo y después el ángulo del vector A con el eje OX. Con las respuestas (módulo y ángulo), calcula las componentes Ax y Ay, pero aun no las ha dibujado:



*Scripts de módulo y componentes para A.* Susana Oubiña Falcón. (CC BY)

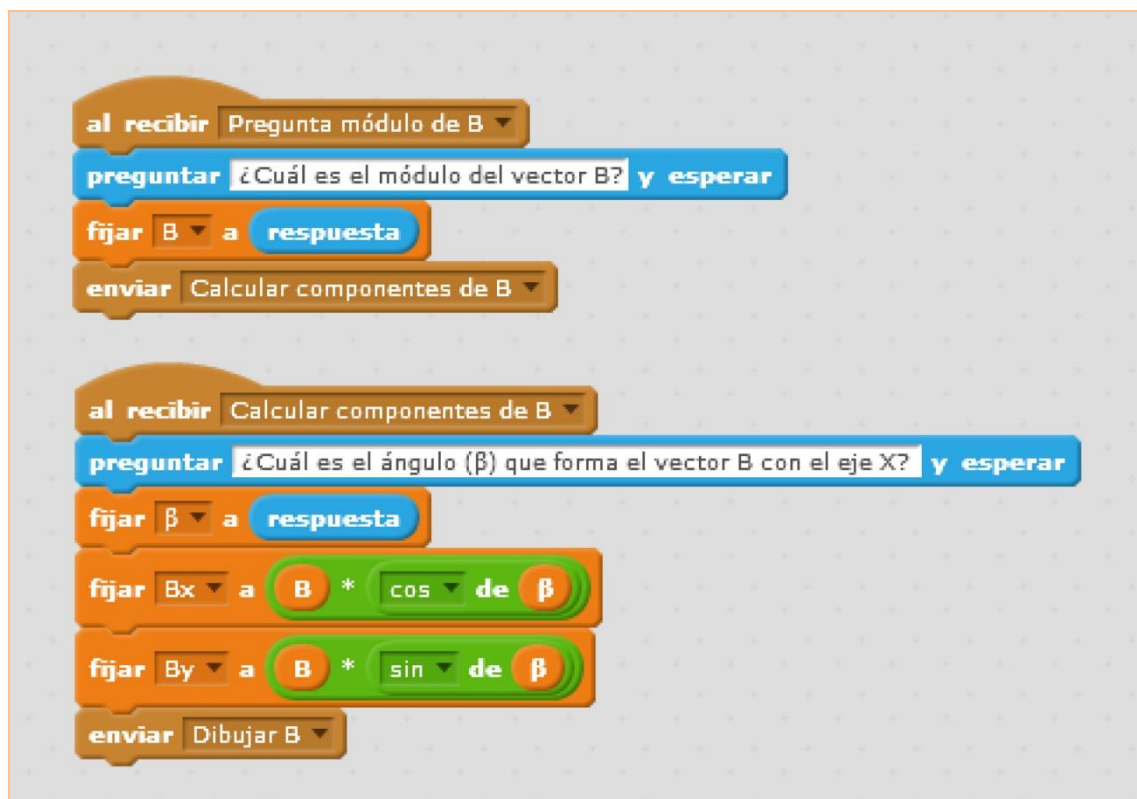
Con las coordenadas calculadas, ya podemos mandar dibujar el vector A:



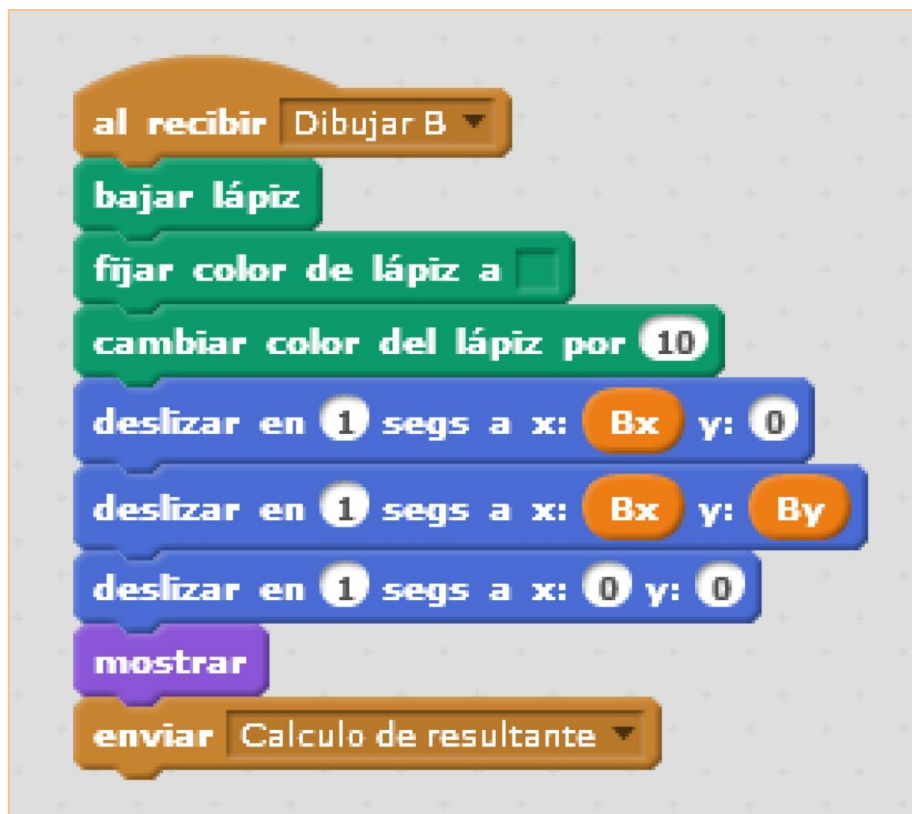
*Script para dibujar A.* Susana Oubiña Falcón. (CC BY)



Actuamos de la misma forma para el vector B:

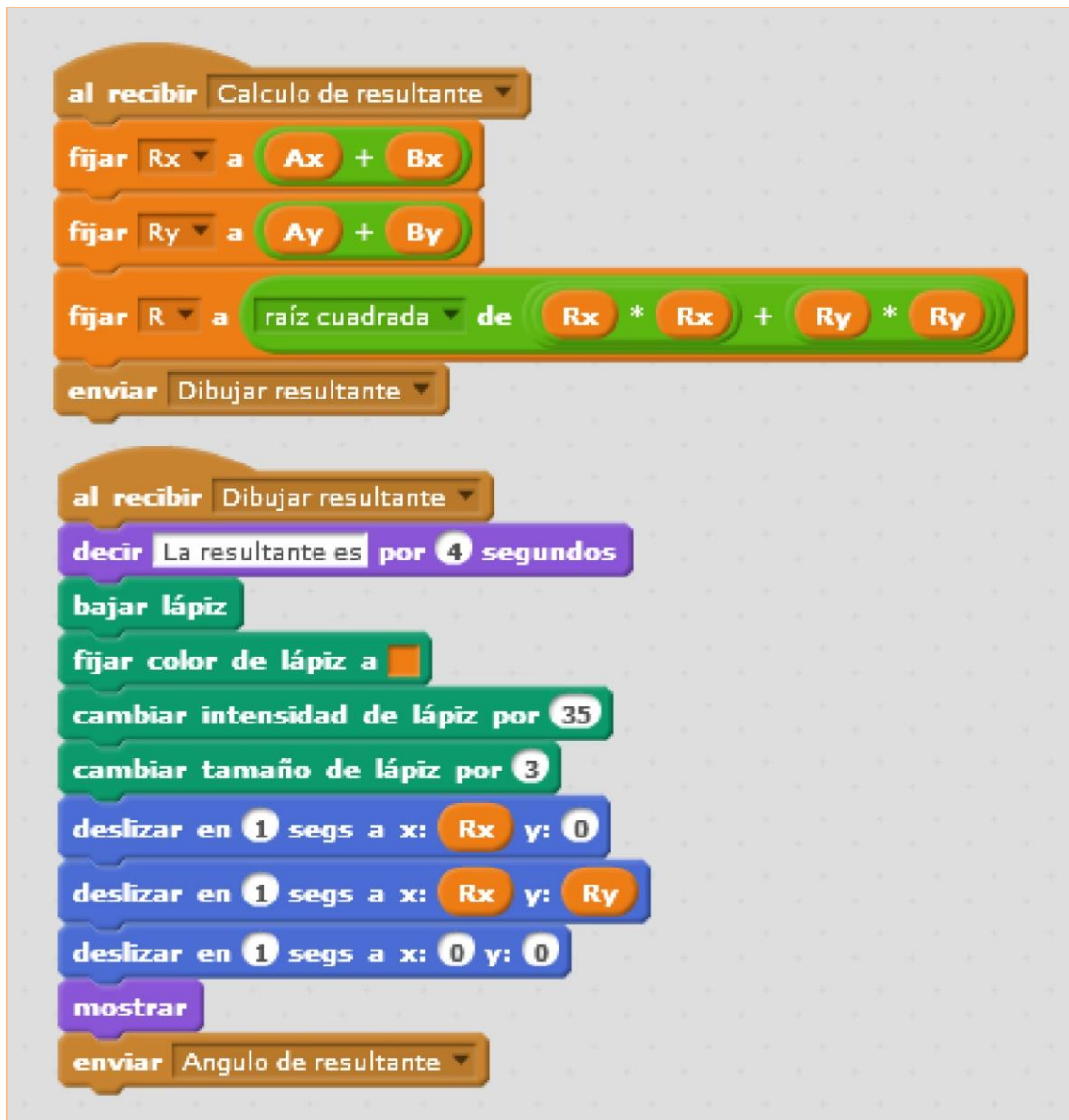


*Scripts de módulo y componentes para B. Susana Oubiña Falcón. (CC BY)*



*Script para dibujar B. Susana Oubiña Falcón. (CC BY)*

A continuación, abordamos el cálculo y dibujo del vector Resultante:



*Scripts para calcular y dibujar R.* Susana Oubiña Falcón. (CC BY)

Finalizamos calculando en ángulo del vector Resultante:



Script para calcular en ángulo en R. Susana Oubiña Falcón. (CC BY)

# Módulo 4

---

## Aumentar la motivación con la gamificación

---

#### 4. Aumentar la motivación con la gamificación.

Hoy en día, la gamificación es una corriente que ayuda a modificar el comportamiento de las personas con el fin de alcanzar unos objetivos de negocio. Para ello utiliza y aplica técnicas y dinámicas de diseño de juegos en contextos no específicos de juego.

Esta herramienta, utilizada de forma coherente, potencia la motivación, aumenta un deseo de compromiso, fomenta la colaboración entre las partes y, en consecuencia, ayuda a conseguir de forma exitosa unos “objetivos de negocio”. La gamificación ha sido señalada como una doctrina que lleva a la motivación y al compromiso en una serie de ámbitos funcionales. En este sentido, implementada en un entorno educativo, favorece el aprendizaje y con ello a la educación, de modo que, un proyecto bien gamificado logrará que el alumno/a quiera volver y probar otra vez, aprendiendo, hasta que sea el ganador.

A la hora de crear un programa que sea verdaderamente funcional en el aula, dirigido a nuestro alumnado, nuestro proyecto debe presentarse de forma atractiva y motivadora. Con esta finalidad se introduce la gamificación en este curso. Los **objetivos** que se persiguen en este módulo 4 son:

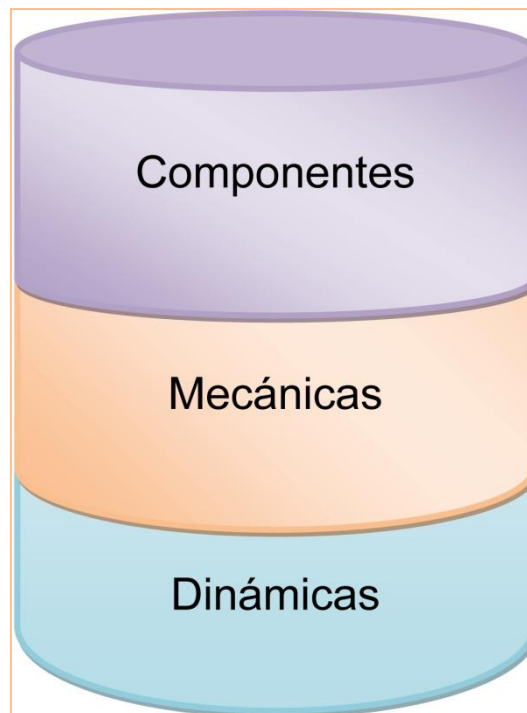
- ✓ Conseguir crear programas atractivos al alumnado.
- ✓ Conocer y experimentar patrones que puedan enganchar a nuestros alumnos por presentar características de juegos.
- ✓ Saber introducir los retos y la diversión en el aprendizaje en la escuela.

##### 4.1. ¿Qué es la gamificación? (ventajas)

La gamificación es una apasionante disciplina, estrechamente unida a la psicología del pensamiento<sup>2</sup>, y que utiliza dinámicas, mecánicas y componentes de juego en contextos ajenos al mismo, como puede ser la escuela. Su objetivo es enganchar a los usuarios, motivar a la acción, atraerlos, logrando con ello modificar actitudes y comportamientos. En el proceso, se necesita un espacio para interactuar, y ese espacio puede ser, desde una red social educativa hasta un entorno educativo (como un aula), siempre diseñados con unos objetivos prefijados.

---

<sup>2</sup> El Modelo de Fogg, la Teoría de la Autodeterminación y la Teoría de Flow.



*Elementos de un proyecto Gamificado. Susana Oubiña Falcón. (CC BY)*

Es importante aceptar que no siempre podremos gamificar. Lo correcto es realizar un estudio previo y con él conocer si es posible y factible desarrollar una gamificación en un proyecto para nuestro alumnado.

En cuanto a las ventajas, la gamificación permite, de forma directa, potenciar:

- ✓ Un incremento del *engagement* (compromiso, involucración e implicación) que ayudará a obtener una lealtad y satisfacción del jugador/a. A su vez, potenciará su rendimiento ya que el usuario sentirá el deseo de hacerlo cada vez mejor.
- ✓ La competitividad de modo que el jugador vea que su trabajo es reconocido en un entorno social que presenta importancia para él.
- ✓ La colaboración y la participación, amplificando la sensación de progreso en el jugador/a.

#### 4.2. ¿Cómo introducirla en el aula?

Todo proyecto gamificado, de forma coherente, debe buscar producir en el usuario (jugador) la sensación de experimentar una “experiencia emocionante”, en el sentido de conseguir potenciar unas emociones como pueden ser el refuerzo de la curiosidad, la competitividad e incluso la felicidad. Aunque un proyecto gamificado parte de la concepción de que se aplicará en un entorno no lúdico, esta premisa no significa que los usuarios no logren emociones ni se diviertan. Para conseguirlo, debemos conocer a los jugadores: las motivaciones



y comportamientos de los usuarios que van a experimentar el proyecto gamificado.

Para un docente, el juego está siempre vinculado a acciones que, bajo una capa de “ocio” o “diversión” tienen un trasfondo distinto. La gamificación requiere de una adaptación entre las mecánicas y las dinámicas del juego al contexto específico de los usuarios. En este sentido, debemos aceptar que los usuarios y sus motivaciones son diferentes y diversas y que las mecánicas que son óptimas para unos, pueden no serlo para otros.

Obviamente, el factor estético del proceso influye en la experiencia. Un proyecto implementado usando elementos que le proporcionen una estética visual y auditiva acorde con los deseos del usuario potenciará el enganche y la lealtad de los mismos al proyecto. Por ello, es importante intentar conseguir en el entorno scratch y sus combinaciones unos escenarios y objetos atractivos e integrar sonidos y canciones.

Por otro lado, la narración o historia del proyecto gamificado, también es otro factor clave en la gamificación. Los usuarios deben conocer el conjunto de reglas del juego y que éstas, sean percibidas por ellos como justas. Pero no sólo eso, también deben tener en mente y de forma clara lo que tienen que hacer para cumplir con las tareas que se le piden.

El elemento principal para el éxito de una enseñanza basada en juegos es el docente. Él es el que tiene que identificar y diseñar el contexto en el cual el juego puede ser educativamente relevante. La gamificación, esta apasionante disciplina, le ayudará a elaborar proyectos propios que aumenten la motivación, la satisfacción, la productividad y el compromiso de su alumnado. Es esencial sustentar la propuesta en unos objetivos muy claros y obviamente en los jugadores, que son los que vivirán la experiencia y los auténticos protagonistas.

## **Elementos de la gamificación: Dinámicas, mecánicas y componentes**

### **A. Dinámicas**

Según Kevin Werbach y Dan Hunter [1], las dinámicas de juego son aspectos más globales que las mecánicas de juego. Las mecánicas de juego están unidas a los deseos, objetivos y motivaciones que se quiere reconducir y potenciar en el usuario, de modo que, para alcanzarlos se utilizan las mecánicas. En cambio, las dinámicas se encuentran ligadas a los deseos, necesidades e inquietudes humanas que construyen la motivación de los usuarios.

Algunas de las dinámicas más relevantes son:

- ✓ *Restricciones*: Un juego resulta motivador dentro de una serie de restricciones. Por ejemplo, un entorno de libertad limitada, estableciendo soluciones de compromiso, etc.
- ✓ *Emociones*: Teniendo en cuenta que la gamificación se realiza en un entorno de no juego, debemos intentar conseguir unas emociones que, aunque limitadas, sean suficientes para conseguir mantener la atención y motivación del usuario. Éstas pueden ser: un refuerzo de la curiosidad, la competitividad, etc.
- ✓ *Narrativa*: Los usuarios deben tener claro en qué consiste el juego, la actividad, la tarea.
- ✓ *Progresión*: Se pretende que el usuario tenga una sensación de progreso y mejora de sus habilidades.
- ✓ *Relaciones*: Se quiere fomentar la necesidad del usuario de interactuar con otros, bien sea logrando un estatus, adquiriendo un reconocimiento social, etc.

Para llevar a cabo las dinámicas de juego, se utilizan diferentes mecánicas de juego.

[1] Werbach, K., & Hunter, D. (2012). *For the Win: How Game Thinking Can Revolutionize Your Business*. Wharton Digital Press.

## **B. Mecánicas**

Las mecánicas de juego son elementos claves que se utilizan para implementar dinámicas de juego. El objetivo es alcanzar la motivación del usuario. Las mecánicas incluyen los principios, reglas y mecanismos que dirigen el comportamiento a través de un conjunto de incentivos, *feedback* y recompensas. A veces, las mecánicas se entremezclan con los componentes del juego. Además, cada mecánica de juego es una forma de conseguir o lograr una o más dinámicas de juego.

Se pueden clasificar englobándolas en dos grandes grupos: mecánicas enfocadas al juego o mecánicas enfocadas al deseo de reconocimiento social.

Los tipos más relevantes de mecánicas de juego son los siguientes:

- ✓ *Retos*: Cualquier tarea o actividad propuesta que el usuario debe alcanzar, debe requerir un esfuerzo por su parte para ser cumplida.
- ✓ *Oportunidades*: En esta mecánica se introduce un componente de suerte o un elemento de azar. El usuario no interviene.

- ✓ *Competición*: En toda competición existen ganadores y perdedores, pero también debe existir un premio que motive al usuario y que lo presente ante el grupo como alguien de valor reconocido.
- ✓ *Cooperación*: Se entiende como la contraria a la competición. En esta mecánica se potencia la colaboración grupal o individual (pero con un objetivo común).
- ✓ *Feedback*: Es información positiva que le llega al usuario a tiempo real y que le indica cómo está haciendo la tarea. Aumenta la sensación de progreso del usuario.
- ✓ *Recopilar recursos*: Los usuarios sienten la necesidad de conseguir recursos.
- ✓ *Recompensa e incentivos*: Es la mecánica que incentiva una buena acción o logro del usuario. Las recompensas, por lo tanto, pueden conseguirse no sólo con la consecución de la tarea perfecta, sino también por el esfuerzo que esta le ha supuesto al usuario. De este modo, se puede crear una escala de recompensas que tenga en cuenta el esfuerzo de los usuarios.
- ✓ *Transacciones*: Consiste en intercambiar un recurso por otro, bien sea entre los propios usuarios o entre el usuario y un agente virtual de gestión de los recursos del proyecto gamificado.
- ✓ *Turnos*: Una participación de usuarios alternada, por turnos.
- ✓ *Estados ganadores*: Identifica el estado que debe lograr el usuario para ganar.

### **C. Componentes**

Los componentes del juego se asocian con las formas o maneras concretas de conseguir los objetivos de las dinámicas y mecánicas de juego. En este sentido, no es un conjunto cerrado de elementos e inclusive pueden combinarse creando otros nuevos.

Algunos componentes de juego más relevantes son: Logros, avatares, insignias, luchas con el jefe, colecciones, combate, desbloqueo de contenido, regalos, tablas de clasificación, niveles de progresión, puntos, conquistas, equipos, etc.

### **D. Tipos de jugadores**

Cuando se gamifica un proyecto debemos aceptar que los usuarios y sus motivaciones son diferentes y diversas y, en consecuencia, las mecánicas que

son óptimas para unos pueden no serlo para otros. Es decir, cada usuario es un mundo. Aún así, los jugadores se suelen clasificar en 4 grandes grupos o perfiles, cada uno con su propia personalidad de jugador.

Es imprescindible conocer qué clase de perfil predomina en nuestro proyecto gamificado para satisfacer sus necesidades y conseguir unos mejores resultados.

La descripción de los 4 tipos de jugadores es la siguiente:

- ✓ El *perfil ambicioso* persigue quedarse el primero, por encima de los demás y su única motivación es escalar posiciones y ganar. ¿Cómo podemos motivarlos? Mostrando cómo escala posiciones y se acerca a la tan ansiada primera posición.
- ✓ El *triunfador* aspira a cumplir los objetivos marcados por el juego. Para retenerlo hay que lanzar y comunicar nuevos retos que le permitan acumular logros.
- ✓ El *rol sociable* aspira a conseguir una red de contactos y de amigos. A este usuario se le motiva facilitándole agregar a nuevos amigos con funcionalidades como chats, etcétera.
- ✓ El *explorador* quiere descubrir lo desconocido. Prefieren juegos con pocas restricciones y que le permitan un amplio abanico de movimientos. ¿Cómo lo retenemos? Ampliando el conjunto de retos, haciendo que cada vez sea más complicado alcanzarlos.

## E. Métricas

La gamificación debe estar diseñada teniendo como centro a la persona, al ser humano. Debe ser un diseño centrado en el usuario, en el jugador, en sus motivaciones, para así lograr estimular el compromiso del mismo. Es una tarea complicada ya que no se trata de crear un juego educativo y premiar con puntos y con insignias las actividades; el éxito de un proyecto gamificado radica en que consiga explotar las motivaciones intrínsecas de los jugadores. ¿Se ha conseguido?

Para saberlo necesitamos elementos que nos sirvan de métricas y nos aporten datos para valorarlo. Es más, para que sea realmente efectivo, el proyecto debe calibrarse, mejorarse, ajustarse y en consecuencia, pasar por una fase de evaluación constante que nos dirá cuáles son sus errores y sus éxitos. Es un proceso repetitivo de prueba y error que nos lleve a averiguar si estamos aplicando las dinámicas y mecánicas acertadas para esa tipología de jugadores y si se están cumpliendo los objetivos marcados.

Se utilizan métricas para cada una de las mecánicas y componentes del juego o proyecto gamificado y éstas deben definirse con el objetivo de involucrar a personas en el proyecto. Las métricas más relevantes que se pueden trabajar son las siguientes:

- ✓ *Reciente*: Consiste en medir lo novedoso, fresco y reciente que es la idea.
- ✓ *Frecuencia*: Se trata de medir cada cuanto tiempo hay que participar en el juego. El hecho de que el usuario pueda jugar permanentemente no alarga mucho la vida del juego. En cambio, si se establece la existencia de un tiempo máximo por día o por intentos, el juego durará más en el tiempo.
- ✓ *Duración*: Mide la duración del juego considerándolo como el tiempo que tardaríamos sin parar para llegar al final. Un buen *engagement*, nunca permitirá que llegues al final del juego.
- ✓ *Viralidad*: Mide la socialización, cuanto comparto en el juego. Compartir logros, necesidades, pedir ayudas; de forma que se interactúe con otros posibles usuarios.
- ✓ *Popularidad*: Esta métrica trabaja la clasificación, los puntos o *badges* (medallas o insignias) de forma que siempre aspiremos a tener un mayor reconocimiento.

Uno de los puntos clave que aporta la implementación de una estrategia de gamificación es el aumento de la viralidad de nuestro contenido, incrementando su alcance. Es una métrica que usada de forma creativa, se aplica en un amplio abanico de proyectos. Por ejemplo, el servicio de almacenamiento de archivos en la Nube ofrecido por Dropbox que permite ampliar tu capacidad de almacenamiento invitando amigos a la aplicación. O Foursquare<sup>3</sup>, Nike<sup>4</sup> o Starbucks<sup>5</sup>, que son algunas de las compañías que ya activamente aplican las técnicas de juego con sus usuarios.

Esta viralidad también tiene su aplicación en la web de scratch. En ella hay dos iconos “me encanta este proyecto” y “marcar como favorito” que nos pueden dar una medida de la aceptación de nuestros trabajos.

---

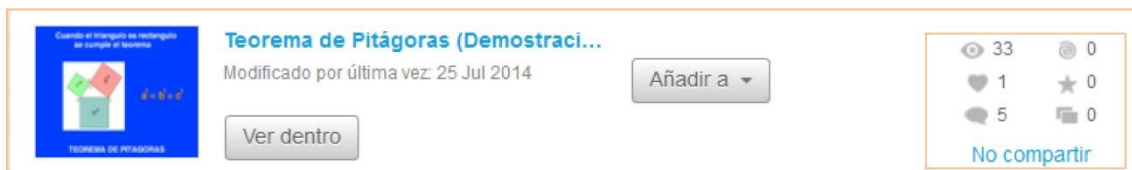
<sup>3</sup> <https://es.foursquare.com/>

<sup>4</sup> [http://www.nike.com/es/es\\_es/?cp=EUNS\\_KW\\_ES\\_1\\_Brand\\_Core\\_Nike](http://www.nike.com/es/es_es/?cp=EUNS_KW_ES_1_Brand_Core_Nike)

<sup>5</sup> <http://www.starbucks.com/>



*Métricas en cada proyecto de la web de scratch. Susana Oubiña Falcón. (CC BY)*



*Métricas en la pestaña "Mis cosas". Susana Oubiña Falcón. (CC BY)*

## F. Un elemento necesario: la diversión

La diversión es un aspecto fundamental en cualquier aprendizaje y en la gamificación ya que crea un buen ambiente en el aula; los alumnos se relajan, se ríen y a la vez aprenden y mejoran sus habilidades. Personalmente opino que, en el campo educativo, el juego y el aprendizaje se encuentran unidos: un tiempo para jugar es un tiempo para aprender.

Las motivaciones de los usuarios o jugadores se clasifican en motivaciones intrínsecas y motivaciones extrínsecas. Las primeras se relacionan con la conducta que lleva a cabo una persona, de forma asidua, sin ningún tipo de influencia externa. Por ejemplo, sería el caso de un jugador que presenta como incentivo el llevar a cabo el reto de la mejor forma posible. Este grupo de jugadores ya sienten un grado de felicidad y diversión por el mero hecho de estar realizando el reto e intentar presentarlo mejor que nadie. En cambio, las motivaciones extrínsecas se asocian con aquellas actividades en las cuales los motivos que le impulsan al jugador a la acción son externos a la actividad. Es



decir, el jugador busca incentivos y refuerzos por sus acciones y por sus hechos. El componente de felicidad que se acerca a la diversión, lo consigue recopilando retos, objetos y logros.

A la hora de realizar actividades, se admite que la motivación extrínseca es necesaria para empezar rutinas en usuarios jóvenes, pero una vez iniciadas las actividades lo correcto sería que se mantuvieran por motivaciones intrínsecas. En la vida práctica es difícil discernir con claridad si sólo nos encontramos ante una motivación intrínseca o extrínseca, ya que habitualmente se ven entremezcladas; podemos sentir un claro interés personal intrínseco y a la vez nuestra conducta se puede ver reforzada por valores extrínsecos.

Considero importante intentar introducir la diversión apoyándonos en un refuerzo con motivaciones extrínsecas pero intentando que éstas no sean el motivo principal que vea el jugador a la hora de cumplir el reto. Para ello, la actividades o retos deben hacer sentir al jugador que “él es competente” para llevarlas a cabo, y lo hará. Lo hará por curiosidad, porque quiere descubrir, por experimentar; lo hará porque se siente capaz de lograrlo y lo hará porque quiere seguir creciendo en el juego. La situación óptima sería lograr que los jugadores se diviertan y que decidan participar incluso cuando no existieran motivaciones extrínsecas que les proporcionaran: Querer participar simplemente porque se lo pasan bien, por placer.

El entretenimiento es el conjunto de actividades que realizan las personas y que le producen diversión. Por lo tanto, el entretenimiento ayuda a ocupar el tiempo de las personas y no debe verse, en ningún momento, como un tiempo inútil o perdido. El divertimento y el aprendizaje (reforzado con actividades atractivas) están estrechamente unidos y no deben entenderse como conceptos reñidos.

Una estrategia de gamificación bien desarrollada aprovechará los principios de recompensas e incentivos para fomentar la acción del jugador, la acción del alumno. Usando elementos de fidelización, de reconocimiento y de competición, los jugadores volverán al juego y lo harán porque se están divirtiendo.

El scratch se puede presentar como un tipo de actividad complementaria en la que los alumnos/as aprendan divirtiéndose. Con las actividades de Scratch comienzan a pensar de forma estructurada: el alumno/a se da cuenta que para que el personaje haga lo que él/ella desea, debe presentarle las órdenes de forma clara y ordenada. Con estas actividades se potencia la persistencia, la centralización de ideas, el pensamiento y cierta libertad buscando la creatividad de cada jugador. Estas características les motivarán a cumplir el reto y a hacerlo bien.

# Módulo 5

---

## Evaluar los trabajos de Scratch por rúbricas

---

## 5. Evaluar los trabajos de Scratch por rúbricas

Las rúbricas se presentan en este curso como una estrategia de evaluación que permiten conocer, tanto al alumno como al educador, el progreso y el desempeño o cumplimiento de la tarea que entrega el alumno. Es decir, es un instrumento que ayuda a discernir el grado de competencias que ha adquirido nuestro alumnado. Para el docente, es una herramienta de evaluación, pero para el alumno, además de ello es una herramienta que le indica su nivel de progreso; como una guía gracias a la cual toman conciencia de los criterios que serán usados para evaluar su aprendizaje en esa tarea que se le ha mandado realizar (autoevaluación) e incluso recibir la evaluación de la misma por su compañeros de grupo (coevaluación). Hay que tener en cuenta que se trata de una herramienta que no se limita al ámbito de la evaluación calificativa, sino que también es enormemente formativa al establecer con detalle cómo deben hacerse las cosas correctamente.

En el campo de la programación scratch, nuestro alumnado crea un programa o proyecto. Ejercicio o tarea que es propuesta por el docente con un propósito determinado. A veces ocurre que el alumno no conoce realmente qué se espera de su trabajo y en qué grado de rendimiento, asumiendo que, para él existe una cierta subjetividad a la hora de ser evaluado. Gracias a la rúbrica, el alumno/a toma conciencia de la forma en la que se evaluará su trabajo ya que conoce de antemano los criterios de calificación de la tarea, participando en los mismos y permitiéndole describir, de manera cualitativa (y cuantitativa), los diferentes niveles de logro alcanzados. Por lo tanto, la rúbrica le ayudará a conocer las debilidades y fortalezas de su trabajo fomentando la retroalimentación en su aprendizaje.

Persiguiendo obtener las competencias que se han descrito al inicio del curso, en este módulo 5, los **objetivos** de aprendizaje que nos llevarán a poder desenvolverlas son los siguientes:

- ✓ Conocer y experimentar con rúbricas.
- ✓ Conseguir crear rúbricas para las tareas scratch.

### 5.1. La rúbrica como forma objetiva de evaluación

Partiendo del hecho de que evaluar es valorar, introduzco el concepto de rúbrica. Una rúbrica, también llamada matriz de valoración, es una herramienta creada inicialmente por el docente para una tarea concreta y consensuada con su alumnado, que permite conocer el “desempeño” (o competencias) que ha adquirido el alumno/a en la tarea realizada. En ellas se diseñan y especifican un conjunto de criterios (aspectos a tener en cuenta) y su grado de consecución (cuantificadores y cualificadores), para conocer el mayor o menor

dominio que el alumno tiene de los conocimientos que se pretendía que aprendiera, de su aprendizaje y de sus logros en competencias.

## **A. Elementos de una rúbrica**

Por lo general, una rúbrica se compone o está formada por tres elementos principales: aspectos, criterios y escala.

- ✓ Aspectos o indicadores: Conceptos (*rubros*) que el docente quiere valorar en la tarea en concreto.
- ✓ Criterios (descriptores): Cada uno de los aspectos a valorar en una rúbrica se definen y especifican en diferentes criterios y estos se asocian para cada nivel de ejecución.
- ✓ Escala (grado) de calificaciones y/o nivel de ejecución: cualitativo, cuantitativo o mixto.

Para ejemplificar y aclarar la función de estos elementos, apporto una rúbrica que he creado para valorar el efecto que la red social Edmodo ha supuesto en la actividad de gamificación “Tecnopoly” para el alumnado de un curso de 2º ESO (ver Fig.1). En esa rúbrica:

Los aspectos que pretendo valorar son: la sencillez, el aprendizaje, los servicios y la sociabilidad. Estos aspectos se sitúan verticalmente, en la primera columna y se le indica un peso o ponderación. El porcentaje que se le ha dado a cada aspecto dependerá del valor o relevancia que para mí, como docente, presenta cada aspecto en el proyecto. En este ejemplo, el porcentaje o peso para cada uno de los 4 aspectos es un 25%, sumando un total del 100%. Obviamente, un aspecto puede presentar más importancia que otro y tener más peso en la rúbrica.

Mi escala de calificaciones y/o nivel de ejecución presenta un enfoque mixto, es decir, medidas cualitativas y cuantitativas. He decidido introducir una escala par (de 4 opciones) y no impar (de 5 opciones) buscando que el alumno/a sea más objetivo a la hora de marcar el nivel que corresponde a cada criterio del aspecto a valorar. Es decir, que se piense más la opción a marcar. En este sentido, es importante hacer notar que hay una tendencia por parte de los alumnos/as a marcar el nivel intermedio (nivel 3 en una escala de 1 a 5). La escala se sitúa de forma horizontal (ver Fig.1), a continuación de los “aspectos”, presentando tantas columnas como grados pretendamos aportar como opciones.

Los criterios se asocian con los niveles de ejecución y se especifican para cada aspecto que se pretende valorar en la rúbrica. Particularmente, los relaciono con mi definición de cada grado de calidad, comenzando a definirlos por el mejor (excelente o 4), pasando al peor (necesita mejoras o 1) para finalmente describir los grados intermedios (bien o 3 y regular o 2).

Escala de calificaciones (mixta)

ASPECTOS	%	Excelente	Bien	Regular	Necesita mejoras
		4	3	2	1
Sencillez	25%	La interfaz es intuitiva, de sencillo manejo y con todos los menús en castellano	La interfaz es intuitiva, de sencillo manejo y con algún menú en inglés	La interfaz no siempre es intuitiva ni de fácil manejo	La interfaz es difícil de entender y su uso es complicado
		"Criterios" para el aspecto "Sencillez"			
Aprendizaje	25%	Los jugadores piensan que les ha ayudado en el aprendizaje	Los jugadores piensan que les ha ayudado en el aprendizaje	A veces ha ayudado en el aprendizaje	Nunca ha ayudado en el aprendizaje
		"Criterios" para el aspecto "Aprendizaje"			
Servicios	25%	El uso de alguna de sus app ha facilitado a los jugadores el tratamiento de contenido multimedia. Su manejo es sencillo	El uso de alguna de sus app ha facilitado a los jugadores el tratamiento de contenido multimedia. Su manejo es complicado	El uso de alguna de sus app ha facilitado a los jugadores el tratamiento de contenido multimedia. Su manejo es complicado	El uso de sus aplicaciones es complicado y no las usan
		"Criterios" para el aspecto "Servicios"			
Sociabilidad	25%	El grupo de jugadores ha mantenido comunicación por mensajes con la profesora y con otros compañeros	Han mantenido algunos mensajes con compañeros	Han mantenido algunos mensajes con compañeros	No han usado la plataforma para enviar mensajes
		"Criterios" para el aspecto "Sociabilidad"			
		Valoración:			3,5

Fig 1. Elementos de una rúbrica analítica. Susana Oubiña Falcón. (CC BY)

En la Fig 1 se observa una nota final. Esta nota debe ser entendida como una valoración hacia ese proyecto. Ha sido calculada a partir de la elección, por parte del alumno/a (profesor o compañero), de un criterio específico para cada aspecto a valorar. En la figura, la opción marcada sobre cada aspecto la represento con el recuadro de color lila. A partir de aquí y teniendo en cuenta los pesos que he querido darle a cada aspecto, el valor numérico de la valoración la obtengo del siguiente modo:  $(25\% \text{ de } 3) + (25\% \text{ de } 3) + (25\% \text{ de } 4) + (25\% \text{ de } 4) = 3,5$  (sobre 4).

## 5.2. ¿Cómo hacer una rúbrica?

Puede parecernos que crear una rúbrica es un proceso laborioso e incluso engorroso, aportándonos una mayor carga de trabajo no lineal con sus beneficios en la búsqueda de un sistema de aprendizaje significativo y colaborativo. Pero también podríamos pensar que la rúbrica es una herramienta muy útil hacia el mismo objetivo, participando, de manera conjunta, el docente y sus alumnos/as. Obviamente, requiere de un gran trabajo por parte del profesor, decidir qué valorar y cómo puntuarlo y todo esto expresarlo en una rúbrica que, en general, está limitada a la actividad para la que fue diseñada.

En la actualidad existe un recurso gratuito en internet, llamado [RubiStar](#), que nos ayuda a implementar nuestras rúbricas. En él, un usuario (tras registrarse) crea, edita, guarda y accede a sus rúbricas. Este espacio de internet también permite que el usuario pueda realizar búsquedas de rúbricas, bien sea por título, por el nombre la persona que la creo o incluso a través de su correo electrónico.

Powered by 4Teachers.org 4Teacher Tools

**RUBISTAR**

Inicio | Buscar rúbrica | Crear rúbrica | Ingresar | Registrarse | Manual

RubiStar English

Crea esquemas para tu proyecto de actividades de aprendizaje

RubiStar es una herramienta gratuita que ayuda a los educadores a crear rúbricas de calidad.  
Más | ¿Qué es una rúbrica (matriz de evaluación)? | Manual

**Bienvenido** | **Proyectos destacados**

¿Desea crear rúbricas excelentes en poco tiempo? Utilice RubiStar! Los usuarios registrados pueden guardar y editar sus rúbricas en línea y acceder a ellas desde sus casas, la institución educativa o durante viajes. Tanto registrarse como usar esta herramienta es gratuito así que haga clic en el enlace para Registrarse a su derecha para empezar a usar RubiStar.

[Registrarse](#)  
[Tour Rápido](#)

**Crear una rúbrica**

Escoja un tema de los de abajo para crear una nueva rúbrica basada en una de las plantillas:

Proyectos Multimedia Matemáticas Escritura Productos  
Lectura Arte Destrezas Ciencias Música

**Ingresar** [Registrarse](#)

Primera inicial: Apellido: Modificador:  
Código postal: Su contraseña: Ingresar

**Búsqueda de una Rúbrica**  
VER o EDITAR una Rúbrica ya guardada  
Escriba el número de ID de su rúbrica:  
Ver Editar Analizar

**Búsqueda de una Rúbrica**  
Debajo, escriba por favor el nombre de su rúbrica:  
☒ Buscar rúbrica **Títulos**  
☐ Buscar autor **Nombre**  
☐ Buscar autor **Correo electrónico**

Palabras claves:  
Tipo de búsqueda: Cualquier palabra  
Buscar

4teachers.org QuizStar | TrackStar | NoteStar | Profiler Pro | Más herramienta  
Derechos de autor. © 2000-2008, ALTEC at the University of Kansas

RubiStar English | Contáctenos | Privacidad



Una rúbrica puede no estar bien construida y en ese caso, no será beneficiosa para el aprendizaje: Pueden fallar tanto los criterios seleccionados como los distintos niveles de la escala. Además, a veces los criterios de evaluación se ciñen a las tareas (no van buscando la competencia general, sino la especificidad en un aspecto reducido de la misma) y otras veces los criterios son excesivamente generales y, por lo tanto, difícilmente evaluables en las tareas desarrolladas.

### **A. Pasos a seguir para crear rúbricas**

Particularmente, utilizo los siguientes pasos a la hora de crear mi rúbrica:

1. Definir de forma clara la tarea que le propondré a mis alumnos/as, revisando objetivos y contenidos.
2. Identificar los criterios de evaluación para ese producto, que no son más que los aspectos o indicadores que valoraré.
3. Decidir el peso (%) de los aspectos en la tarea, según su importancia buscando que el alumno logre alcanzar las competencias marcadas.
4. Elaborar el nivel de escala a utilizar (aconsejo niveles pares).
5. Definir los criterios para cada aspecto y en cada nivel de escala. Podemos comenzar definiendo el óptimo, después el pésimo y finalizar completando los intermedios.
6. Crear un borrador de la rúbrica, con aspectos, escales, criterios, ponderación y la forma de valoración final
7. Presentar la tarea y su rúbrica a la clase y consensuarla con nuestro alumnado.

### **B. Ejemplos de rúbricas para scratch**

A modo de guía, es interesante disponer de alguna rúbrica creada para proyectos implementados con scratch.

Introduzco como ejemplos las rúbricas desarrolladas por [Código Octopus](#) (*Programación multidisciplinaria con Scratch na Educación Primaria e Secundaria*): Rúbrica general para valorar tareas y rúbrica para valorar historias y narraciones. Ambas se muestran en su apartado de [rúbricas](#):

- ✓ Ejemplo 1: Rúbrica general para valorar tareas. Accesible en el siguiente [link](#). ([CC-BY-SA](#)).
- ✓ Ejemplo 2: Rúbrica para valorar historias y narraciones. Accesible en el siguiente [link](#). ([CC-BY-SA](#)).



***Susana Oubiña Falcón***

***Licencia: CC-BY***