*Manual de usuario*

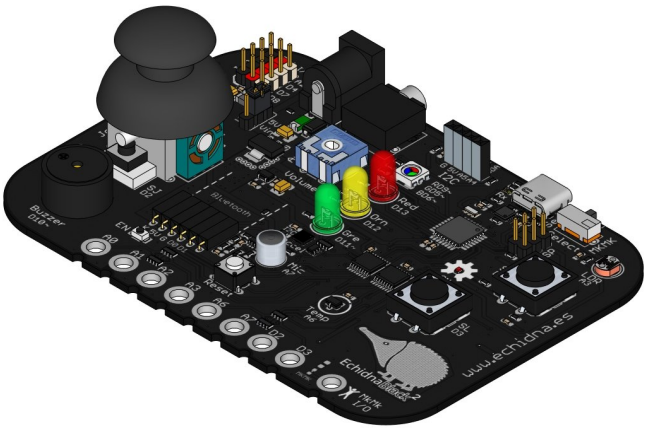
# EchidnaBlack y EchidnaML

# Índice

# 1. INTRODUCCIÓN

**EchidnaBlack** (hardware) y **EchidnaML** (software) constituyen un **sistema integrado**: están diseñados para aprender los fundamentos de la **programación**, la **robótica** y la **inteligencia artificial** (IA). Su **objetivo** principal es fomentar el desarrollo del **pensamiento computacional** en estudiantes de niveles básicos y avanzados (Primaria, Secundaria, Bachillerato y F.P.). Además, busca estimular la creatividad a través de la realización de proyectos y actividades prácticas que relacionan el mundo digital y el analógico.

El **sistema** se compone de una **placa** con diversos componentes integrados y un **entorno de programación** diseñado a medida para aprovechar todas sus posibilidades. Además, este conjunto se complementa con **materiales educativos** creados por docentes para facilitar el uso de la placa robótica y el software en el aula.




Esta **guía** ofrece una **introducción** al trabajo con la placa **EchidnaBlack2** y su entorno de programación **EchidnaML**. No se requieren conocimientos de programación previos; sin embargo, se recomienda tener experiencia con el entorno visual de Scratch, ya que facilita la comprensión de la programación por bloques y el conocimiento del entorno.

Para **ampliar** y complementar la **información** ofrecida en esta guía, puede consultar recursos adicionales y material didáctico en la **web** oficial del **proyecto**: www.echidna.es.

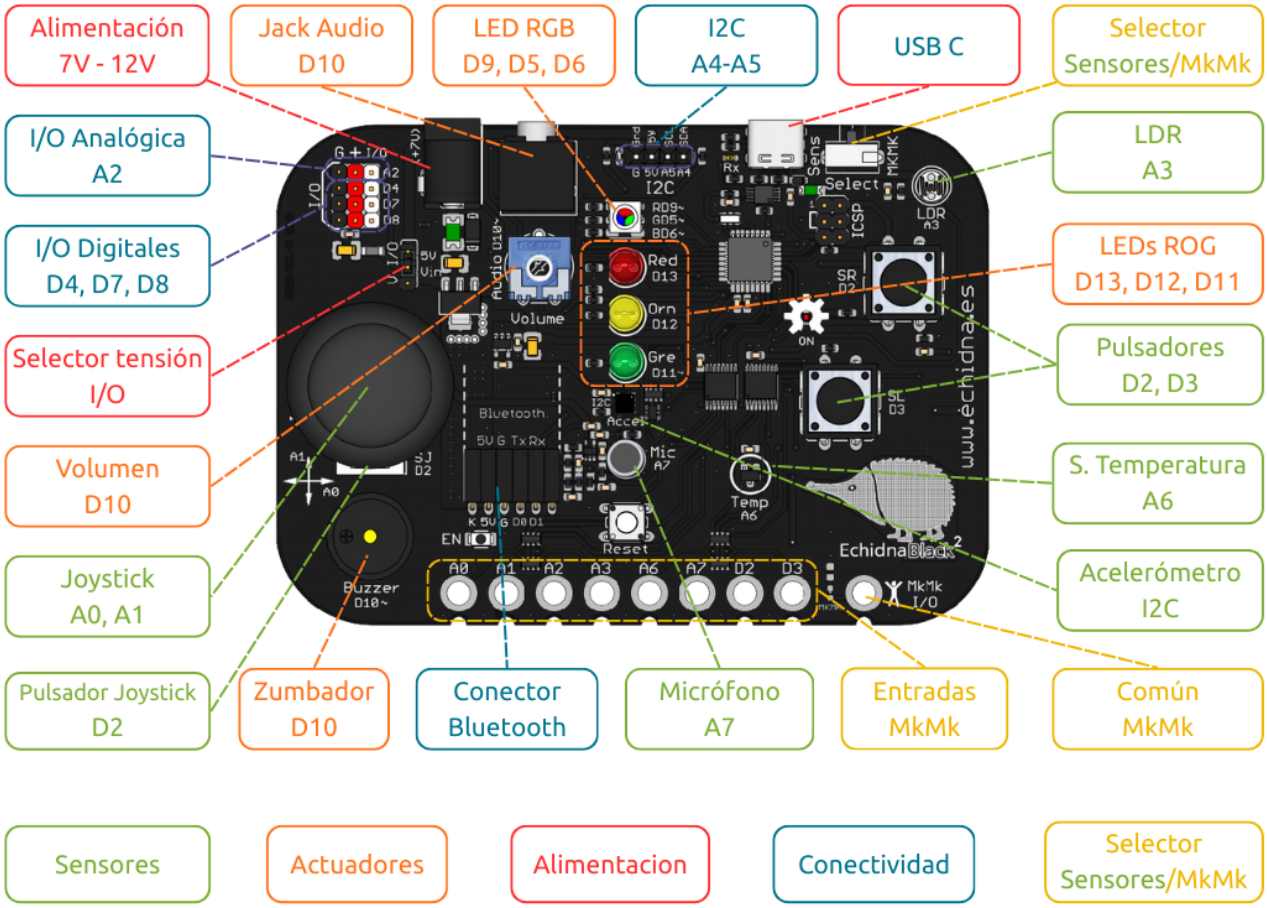
# 2. LA PLACA ECHIDNABLACK

La placa **EchidnaBlack2** es una **placa autónoma basada** en la **arquitectura Arduino** (es compatible con Arduino Nano y UNO).

La placa dispone de su propio entorno de programación por bloques, EchidnaML. Además, su arquitectura permite la integración con otros entornos de programación comunes en el ecosistema Arduino, tales como Arduino IDE, Snap4Arduino o mBlock. Esta versatilidad facilita el desarrollo de proyectos con distintos lenguajes y niveles de complejidad.

## 2.1 Componentes

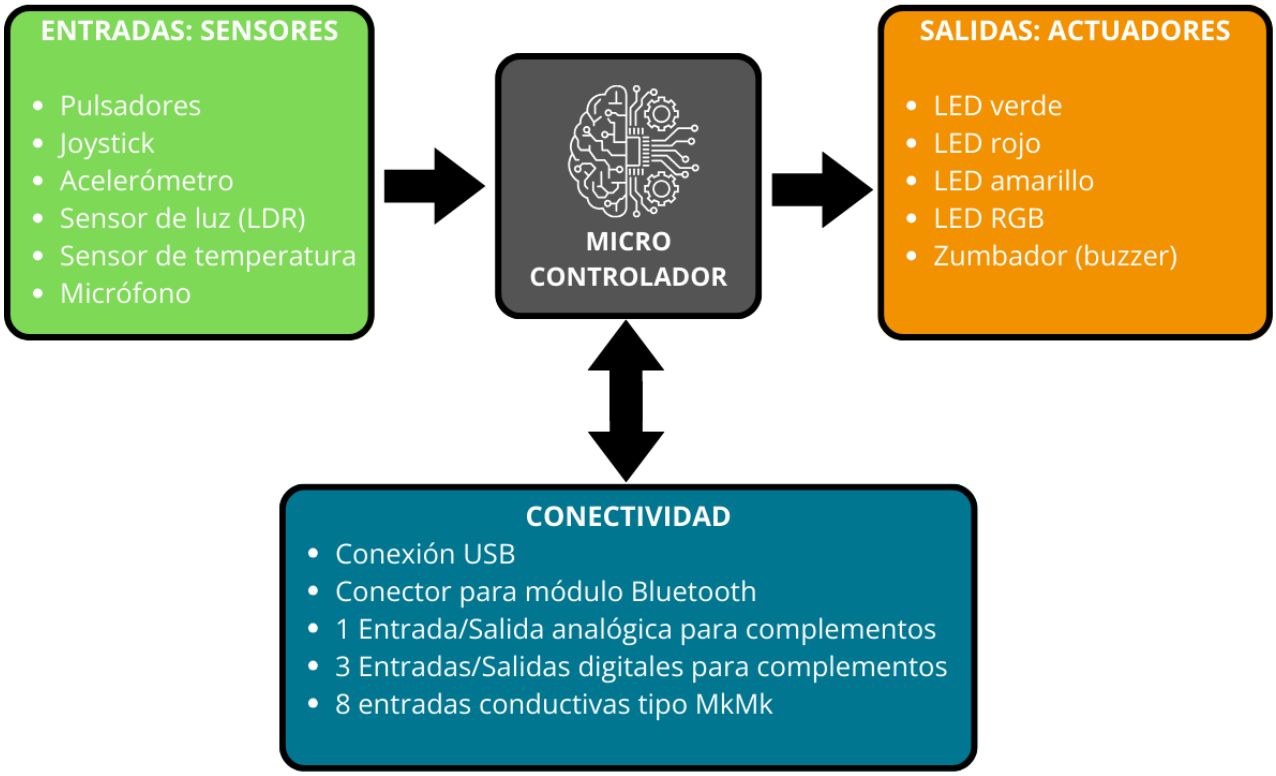
La **placa** cuenta con los siguientes **componentes**:




Para conectar la placa EchidnaBlack2 al ordenador es necesario un cable USB-C. En el apartado "3.3 Puesta en marcha" lo veremos con detalle.

#### Clasificación de los componentes de la placa según su función

Los componentes de la placa pueden clasificarse en cuatro categorías principales según la función que desempeñan dentro del sistema:




**Entradas** (Sensores): son los elementos que capturan información del entorno (variables como temperatura, luz, sonido, gravedad, etc.) y la traducen en datos que el sistema puede procesar.

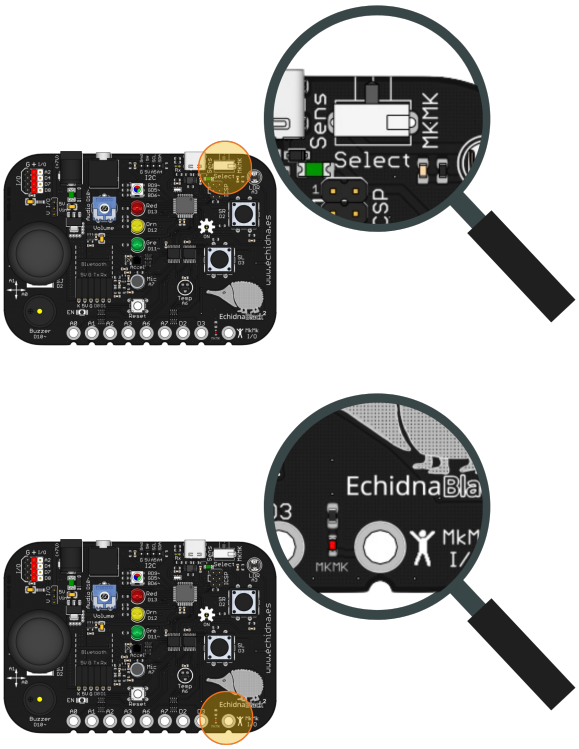
**Microcontrolador** (Cerebro del Sistema): es el procesador central de la placa. Su función es recibir los datos de las entradas, ejecutar las instrucciones del programa y enviar las órdenes a las salidas.

**Salidas** (Actuadores): son los elementos encargados de producir una acción o respuesta física basada en las órdenes del microcontrolador. En esta placa, esto se manifiesta principalmente en forma de luz (LEDs) y sonido (Buzzer).

**Conectividad**: son los componentes que facilitan la comunicación de la placa, permitiendo la interacción con otros dispositivos (como el ordenador) o la conexión de componentes externos (sensores y actuadores adicionales).

## 2.2 Modos de funcionamiento

La placa EchidnaBlack2 dispone de dos modos de funcionamiento, el **Modo Sensores** y el **Modo MkMk**. Mediante un conmutador podemos seleccionar el modo en el que queremos trabajar.




**¡ATENCIÓN!** Un testigo luminoso rojo nos indica cuando estamos en modo MkMk.

#### Modo Sensores

Recoge la lectura de todos los sensores integrados (pulsadores SR y SL, joystick, sensor de temperatura, micrófono, acelerómetro XYZ) y de todas las conexiones I/O, incluidas las I2C.

#### Modo MkMk

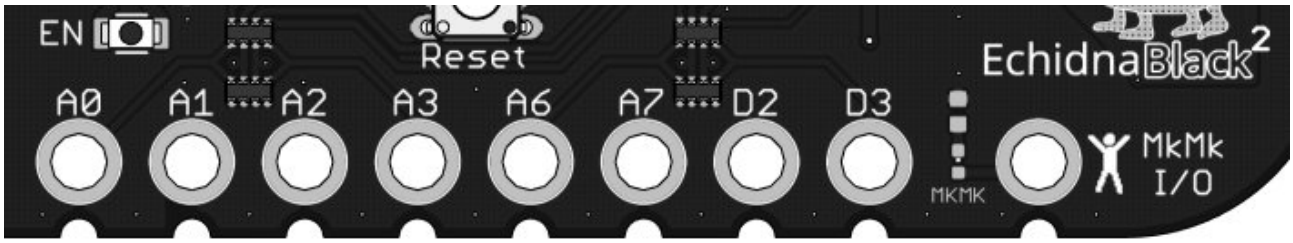
Nos da acceso a las 8 entradas conductivas tipo Makey Makey (MkMk) y al resto de las entradas no usadas en este modo.

En ambos modos tenemos disponibles todos los actuadores.

## 2.3 Entradas MkMk

Las **entradas** tipo **Makey Makey** (MkMk) se **habilitan** al desplazar el **conmutador** de modos de funcionamiento hacia la derecha. Un **LED rojo** se ilumina para avisar al usuario cuando el modo MkMk está **activo**.

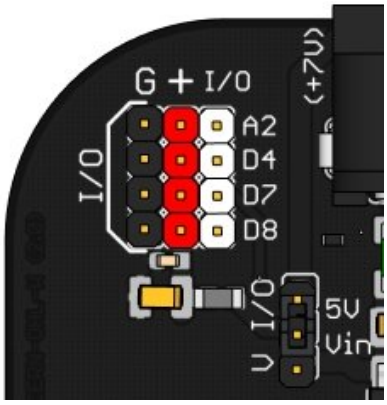
Las entradas **MkMk** son capaces de **detectar conductividad** eléctrica a través de una amplia variedad de materiales. La detección de la señal se produce cuando el circuito se cierra al tocar, de forma simultánea, el elemento conductor conectado a la entrada y el conector MkMk (o la superficie del símbolo Echidna) de la placa.




La placa dispone de **8 entradas MkMk**: A0, A1, A2, A3, A6, A7, D2, D3.

Este modo **permite** crear **proyectos** que unen el **mundo físico** más cercano con el virtual **digital** que se desarrolla en el ordenador de forma muy sencilla y manipulativa. Como por ejemplo mediante la creación de pulsadores hechos con plastilina conductiva, frutas, papel de aluminio o agua, añadiendo un componente lúdico, creativo y de exploración a los proyectos.

## 2.4 Pines de entrada/salida




Los **pines** de **entrada/salida** (I/O) son **conexiones** que nos permiten conectar dispositivos adicionales a la placa, actuando como entradas (sensores) o salidas (actuadores).

- Cuando funcionan como **entrada** reciben información de sensores externos.
- Cuando funcionan como **salida**, envían señales desde la placa para controlar actuadores externos.

La placa cuenta con **tres pines digitales** de entrada/salida (D4, D7, D8) y **uno analógico** (A2), a los que podemos conectar una gran variedad de componentes, como servomotores, sensores de distancia infrarrojos, sensores de humedad de suelo, etc.

Cada pin de entrada/salida cuenta con una conexión a 0 V (G), 5 V (+), y el pin de señal (I/O).

#### Selector de alimentación

Junto a la zona de pines de entrada/salida se encuentra un selector que permite elegir el origen de la energía mediante la posición de un jumper:

**Posición 5V**

La corriente proviene del regulador de tensión interno de la placa (o del puerto USB). Es la opción ideal para alimentar sensores y componentes estándar de bajo consumo.

**Limitación de Corriente:** El consumo total de los componentes externos conectados a la línea de 5V con un límite absoluto 500 mA, se recomienda no superar los 300 mA Exceder este límite puede provocar el apagado por protección o sobrecargar el regulador interno.

**Posición Vin**

La alimentación se toma directamente de la fuente externa conectada a la toma de corriente (jack). Esta opción es la recomendable al conectar actuadores de mayor consumo, como servomotores o motores DC, para no saturar la línea de 5V de la placa.

## 2.5 Alimentación


**Alimentación por USB-C**

La **alimentación** a través de **USB-C** es la forma más **recomendable** para la **mayoría** de los **usos**.

Esta es la forma más sencilla y común de usar la placa, mediante un cable USB-C conectado a un ordenador. La placa recibe una tensión de 5 V y suficiente energía para las funciones básicas.

**Alimentación por Jack**

La usamos cuando queremos conectar elementos con mucha **potencia**, como varios servomotores, o para proyectos autónomos. Si queremos que funcione EchidnaML debemos conectar también por cable USB para que se comuniquen el microcontrolador con el ordenador.

# 3. ECHIDNAML: ECHIDNABLOCKS

**EchidnaML** es el ecosistema de software oficial **diseñado** para **programar** las **placas Echidna**. Se trata de una aplicación de escritorio que integra dos potentes herramientas que trabajan de forma conjunta: **EchidnaBlocks** y **LearningML**.

En este apartado nos centraremos en EchidnaBlocks; más adelante veremos cómo trabajar con LearningML.

## 3.1 EchidnaML

**EchidnaML** es el ecosistema de software oficial **diseñado** para **programar** las **placas Echidna**. Se trata de una aplicación de escritorio que unifica tres áreas clave en una sola interfaz: programación por bloques, robótica e Inteligencia Artificial.

Este entorno integra dos potentes herramientas que trabajan de forma conjunta:

- **EchidnaBlocks**: Una versión personalizada de Scratch que añade bloques específicos para el control del hardware (sensores y actuadores) y la interacción con modelos de IA.
- **LearningML**: Un módulo especializado en aprendizaje automático (machine learning) que permite crear, entrenar y desplegar modelos de IA sin salir de la aplicación.


**Ventajas del Sistema Integrado:**

- **Plug and play**: reconoce la placa y permite empezar a trabajar con ella directamente.
- **Todo en uno:** No es necesario cambiar de programa para pasar de la programación robótica a la creación de modelos de IA.
- **Continuidad**: Permite usar los modelos generados en LearningML directamente dentro de los proyectos de la versión de Scratch en EchidnaBlocks.
- **Seguridad**: Al ser una aplicación de escritorio, facilita el trabajo sin conexión y garantiza un entorno controlado para el aula.

#### Descargar el programa

Para empezar a trabajar con EchidnaML, es necesario descargar el programa desde la página web de Echidna Educación.

#### Detección de la placa

El **programa detecta automáticamente** la **placa** y la versión de la misma que estamos utilizando: EchidnaBlack o EchidnaBlack2. Para ello debemos conectar la placa antes de abrir EchidnaML.

**Trabajo sin la placa conectada:**

EchidnaML permite que trabajemos sin conectar la placa. Sin la placa conectada podemos:

1. Usar LearningML para construir modelos.

2. Usar el clon de Scratch.
3. Programar un programa de robótica y guardarlo en el equipo para cuando conectemos la placa.

**Reconectar la placa con el programa abierto:**

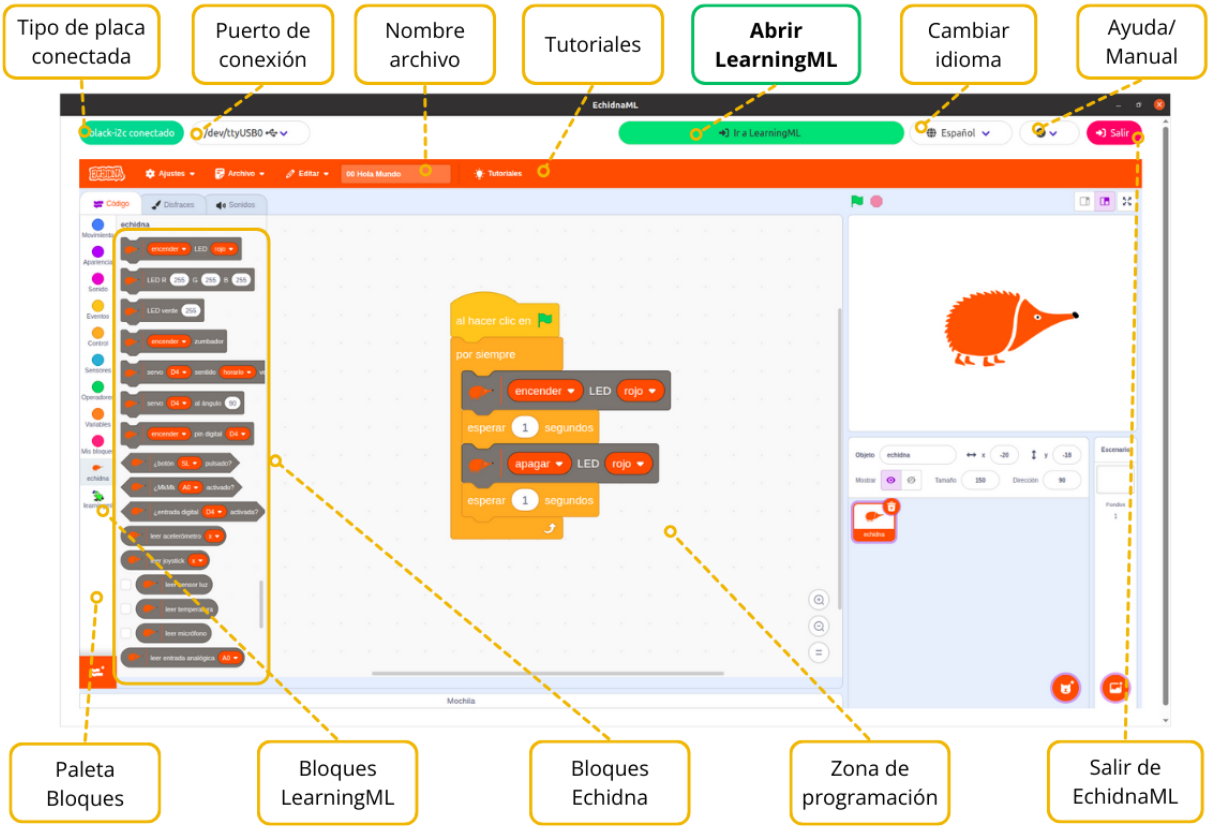
En el caso de que abramos el programa sin la placa conectada, podemos conectar la placa posteriormente. Para que el programa EchidnaML detecte la placa tenemos que pulsar en el icono de USB.

**¡ATENCIÓN!** Si reconectas la placa una vez abierto EchidnaML perderás todo el trabajo, por lo que debes guardarlo antes para poder recuperarlo luego.

## 3.2 EchidnaBlocks

**EchidnaBlocks** es una versión de **Scratch** que incorpora **bloques** específicos para controlar la placa **EchidnaBlack2** y para integrar modelos de **machine learning** mediante LearningML.

**Entorno de programación**

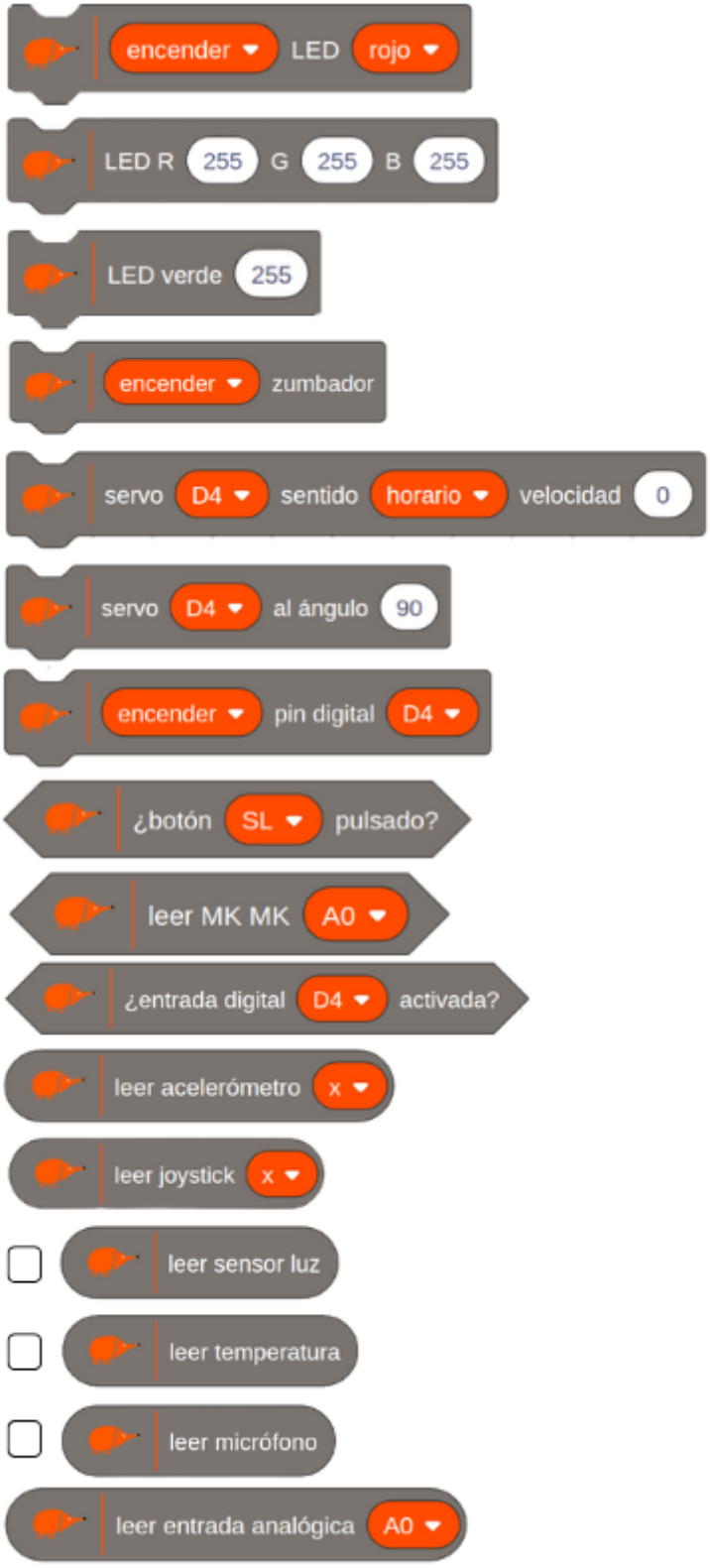



**Comunicación placa-programa**

**EchidnaBlocks** se **comunica** con el **microcontrolador** a través del **puerto serie**, mediante el cable USB. La placa envía información sobre el estado de los sensores y el programa que realizamos en EchidnaBlocks lo procesa y devuelve información sobre el estado de los actuadores de la placa.


al hacer clic en
por siempre
encender LED rojo
esperar 1 segundos
apagar LED rojo
esperar 1 segundos
Recibe Información / Actuadores
Envía Información / Sensores
Standard
Firmata

**Bloques de programación de la placa**




**EchidnaBlocks** ha añadido los siguientes **bloques** de **programación** a Scratch que permiten **controlar** el funcionamiento de la **placa**. Estos bloques permiten tanto leer la información de los sensores como enviar señales para controlar los actuadores de forma sencilla.

Si observamos los **bloques** por su **forma**, podemos distinguir 3 tipos de bloques: rectangulares, trapezoidales y redondeados.

**Rectangulares**:

Son bloques dedicados a programar los actuadores (LED, Zumbador, servo, etc.).

**Trapezoidales**:

Son bloques destinados a leer sensores y entradas digitales, cuya lectura devuelve un true (1) o false (0).

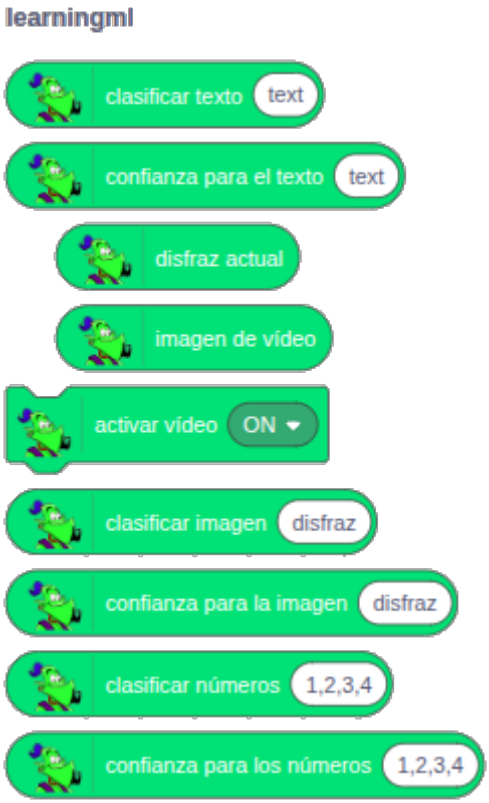
**Redondeados**:

Son bloques destinados a leer sensores, entradas analógicas, estos bloques devuelven el valor del sensor.

Hay tres de ellos, luz, temperatura y micrófono, que permiten activar la **casilla** de **verificación** permitiendo visualizar el valor registrado.

No te preocupes por lo que hace cada uno de los bloques ahora; lo veremos en el siguiente apartado a través de ejemplos.

**Bloques de programación de LearningML**




**EchidnaML** integra **LearningML**, una **plataforma educativa** diseñada para la enseñanza de los **fundamentos** del **machine learning** (aprendizaje automático). Esto nos permite incorporar Inteligencia Artificial a nuestros proyectos de robótica.

EchidnaBlocks incorpora bloques específicos de machine learning (ML) que permiten construir aplicaciones de inteligencia artificial (IA) capaces de reconocer imágenes, números y textos escritos en lenguaje natural.

Estos bloques, que se encuentran en la sección LearningML, pueden combinarse con los bloques de control de las placas Echidna y con las funcionalidades estándar de Scratch.

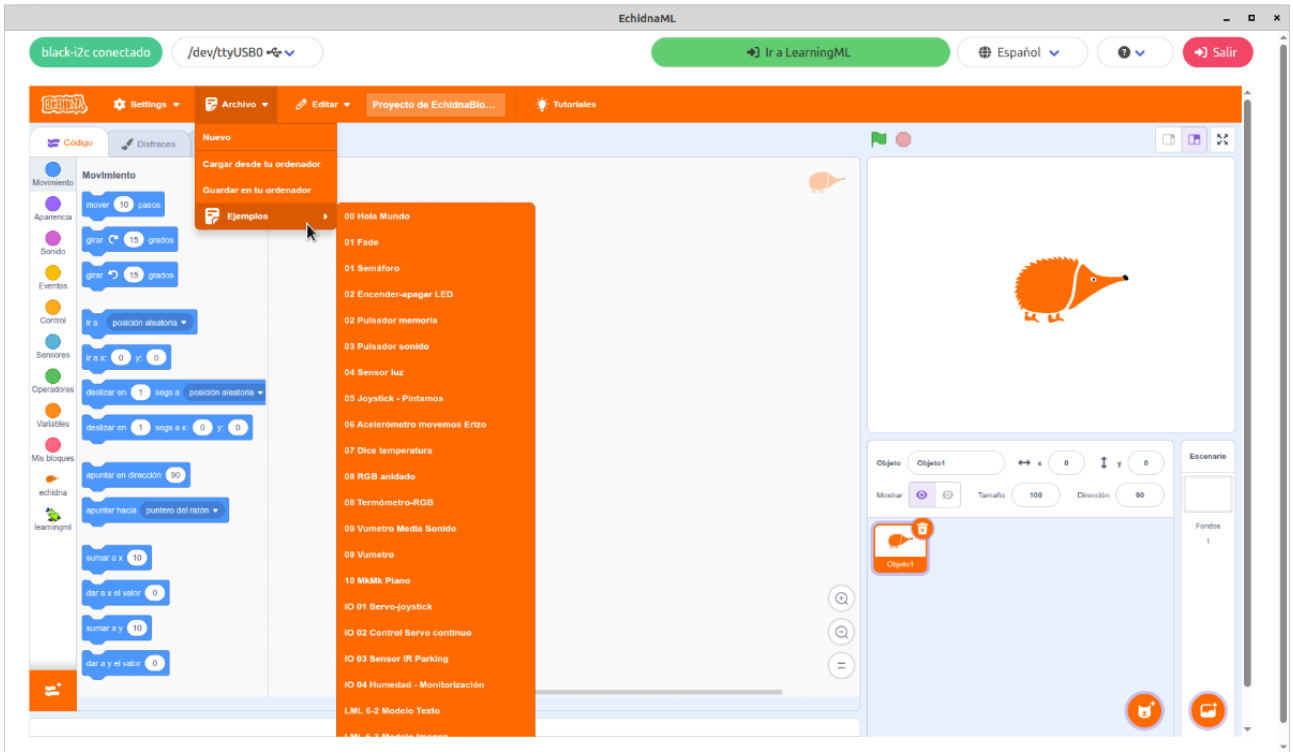
De esta forma, los estudiantes pueden unir el mundo de la IA con la robótica educativa para crear proyectos avanzados.

Para usar los bloques de machine learning, primero debes crear un modelo de reconocimiento de imágenes o textos. Pero eso lo contaremos en el apartado 5 de este manual que está dedicado a LearningML.

## 3.2.1 Menú Archivo

En el **menú Archivo** encontramos:

- Nuevo
- Cargar desde tu ordenador
- Guardar en tu ordenador
- Ejemplos




**Nuevo**

Si quieres empezar un nuevo proyecto.

**Guardar en tu ordenador**

EchidnaBlocks es un programa de Escritorio, por lo que si quieres guardar tus proyectos debes descargarlos y almacenarlos en tu ordenador. Para ello debes pulsar en Archivo → Guardar en tu ordenador y almacenarlo en la carpeta que elijas.

**Cargar desde tu ordenador**

Para **recuperar** un proyecto, pulsa en Archivo → Cargar desde tu ordenador y buscar el archivo en la carpeta donde lo guardaste.

Los proyectos se almacenan en **formato sb3**, que es el mismo tipo de ficheros que **Scratch**.

EchidnaBlocks es **compatible** con los **proyectos** realizados en **Scratch**, por lo que puedes descargar cualquier proyecto realizado en Scratch y cargarlo en EchidnaBlocks dotándolo de interactividad a través de los sensores de la placa.

#### Ejemplos

En este submenú puedes encontrar todos los **ejemplos** que desarrollamos a continuación en el apartado 5 del **manual**.

En Echidna creemos que una de las mejores formas de **aprender** es a partir de **ejemplos** por lo que os facilitamos estos programas para que podáis probarlos, estudiarlos y modificarlos, una de las esencias del software libre.

## 3.3 Puesta en marcha

Comenzar a trabajar con el entorno Echidna es muy sencillo, ya que la placa cuenta con sensores y actuadores integrados y el software la detecta automáticamente permitiendo que podamos empezar a programar directamente.

En este gráfico podemos ver los **pasos** para **comenzar** a usar **EchidnaML** y **EchidnaBlack**:


**1- Conecta la placa Echidna al ordenador mediante el cable USB-C**

Antes de abrir el programa EchidnaML conecta la placa al ordenador mediante el cable USB-C.

Las placas Echidna tienen cargado de fábrica el programa StandardFirmata, necesario para la comunicación con el ordenador a través del puerto serie (USB).

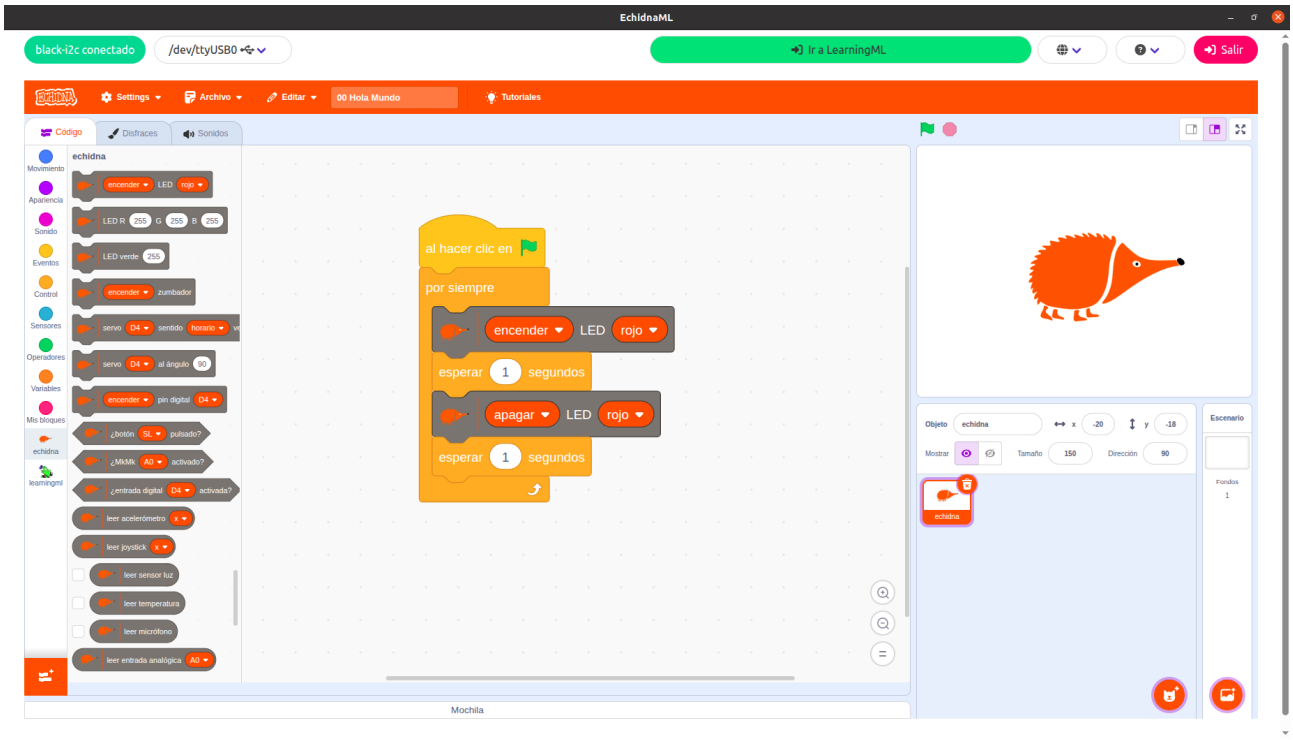
**2- Abre el programa EchidnaML**

En tu ordenador clica en el icono de EchidnaML para abrir el programa. Previamente debes haber instalado el programa.

**3- ¡Empieza a programar!: Hola, Mundo con EchidnaBlocks!**

Al iniciar EchidnaML, este se conecta automáticamente con la placa. Si el programa no detecta la placa, visita el apartado 6.2 de este manual donde explicamos cómo cargar el programa StandardFirmata.

Al combinar los nuevos bloques específicos de la placa Echidna con los bloques clásicos de Scratch, podemos programar dispositivos físicos de manera sencilla.




Como primer ejercicio de iniciación, vamos a crear el "**Hola, Mundo!**" de la robótica: un LED que parpadea de forma intermitente.

**Lógica del Programa:**

Este programa utiliza un bucle continuo para ejecutar la siguiente secuencia lógica, creando un parpadeo constante en el LED Rojo:

1. Activación: Se enciende el LED Rojo.
2. Espera: Se detiene la ejecución del programa durante un segundo.
3. Desactivación: Se apaga el LED Rojo.
4. Espera: Se detiene la ejecución del programa durante un segundo antes de volver a empezar el ciclo.


al hacer clic en
por siempre
encender
LED
rojo
esperar
1
segundos
apagar
LED
rojo
esperar
1
segundos

# 4. COMPONENTES Y BLOQUES DE PROGRAMACIÓN: DESCRIPCIÓN Y EJEMPLOS

En este apartado vamos a analizar tanto los **componentes** que integra la placa como otros que podemos conectar.

Revisaremos las principales **características** que tienen, los **bloques de programación** que podemos usar para controlarlos, y **ejemplos** de su aplicación en el entorno de programación.

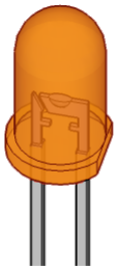
# 4.1 Componentes de la placa

A continuación se **describen** los **componentes** que lleva integrados la **placa**. En cada uno de ellos se explica uno o varios ejemplos de uso, recuerda que los tienes accesibles en EchidnaBlocks.



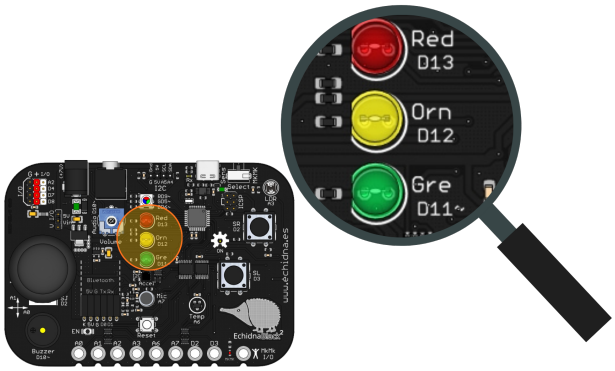
## 4.1.1 LEDs: Semáforo e intensidad luminosa

### COMPONENTE:



**Diodos LED**: es el acrónimo de Light Emitting Diode (Diodo emisor de luz), y está basado en el fenómeno de electroluminiscencia. Se usan como testigos (indicadores) y como fuente de iluminación.

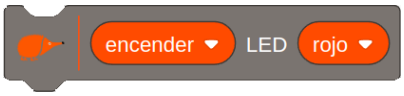
En la **placa** tenemos 3 diodos LED: Verde, Naranja y Rojo.




Los **LEDs Naranja y Rojo** pueden ser controlados únicamente de forma **digital** (estados de encendido/apagado). Por otro lado, el **LED Verde** puede ser controlado **digitalmente** y, además, permite el control de su intensidad luminosa (modulación analógica o **PWM**).

### BLOQUE DE PROGRAMACIÓN:

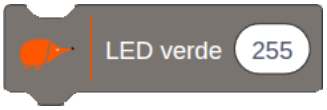
Para **controlar digitalmente** los LEDs podemos usar el siguiente bloque:




En el que podemos:

- Controlar el estado: encender o apagar
- Controlar qué LED queremos encender/apagar

Para **controlar la intensidad luminosa** del **LED Verde** podemos usar el siguiente bloque:




En el que podemos regular la intensidad luminosa entre 0 (apagado) y 255 (máxima intensidad luminosa).

### EJEMPLO: Semáforo

Este ejemplo muestra cómo programar un semáforo utilizando los LEDs de la placa. La programación se basa en una secuencia cíclica donde cada LED permanece encendido durante un tiempo específico y luego pasa al siguiente estado de forma automática.


**Estados**:

1. El LED verde se enciende durante 5 segundos. Al finalizar este tiempo, se apaga.
2. El LED naranja se enciende durante 2 segundos, y luego se apaga.
3. El LED rojo se enciende durante 5 segundos. Transcurrido este tiempo, se apaga.

Luego, el ciclo vuelve a comenzar con la luz verde y se repite de forma indefinida.

### EJEMPLO: Control intensidad luminosa LED verde

Este ejemplo muestra cómo **controlar la intensidad luminosa del LED verde** para crear un **efecto** de "**fundido**" (fade) gradual, simulando un ciclo suave de encendido y apagado.

Para lograrlo, el programa utiliza dos bucles secuenciales que modifican progresivamente el valor de intensidad del LED (de 0 a 255):


**Bucle de encendido progresivo:**

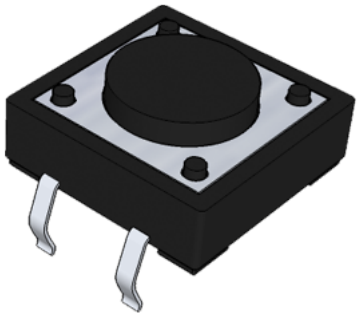
El bucle controla la intensidad del LED aumentando su valor de forma gradual desde 0 (apagado) hasta 255 (máxima luminosidad). Este proceso se realiza en incrementos de una unidad, lo cual simula un efecto de fundido (fade) muy suave.

**Bucle de apagado progresivo:**

La intensidad del LED disminuye de 255 (máxima luminosidad) a 0, creando un efecto de fundido hasta apagarse por completo.
Ambos bucles se ejecutan consecutivamente para generar el efecto de fundido suave y cíclico. El ciclo completo vuelve a repetirse al final.

## 4.1.2 Pulsadores: Encender/Apagar Led

### COMPONENTE:



El **pulsador** es un **componente electromecánico** que permite **abrir** o **cerrar** un **circuito** con un solo estado estable. La clave de su funcionamiento es que solo posee un estado estable o de reposo:

**Estado Estable Abierto** (Normalmente Abierto / NO): por defecto, el circuito está abierto (interrumpido) y solo se cierra mientras se mantiene presionado. Al soltarse, regresa inmediatamente a su estado abierto.

En la placa tenemos **2 pulsadores**, **SL** (Switch Left) y **SR** (Switch Right), situados en la parte derecha.

### BLOQUE DE PROGRAMACIÓN:

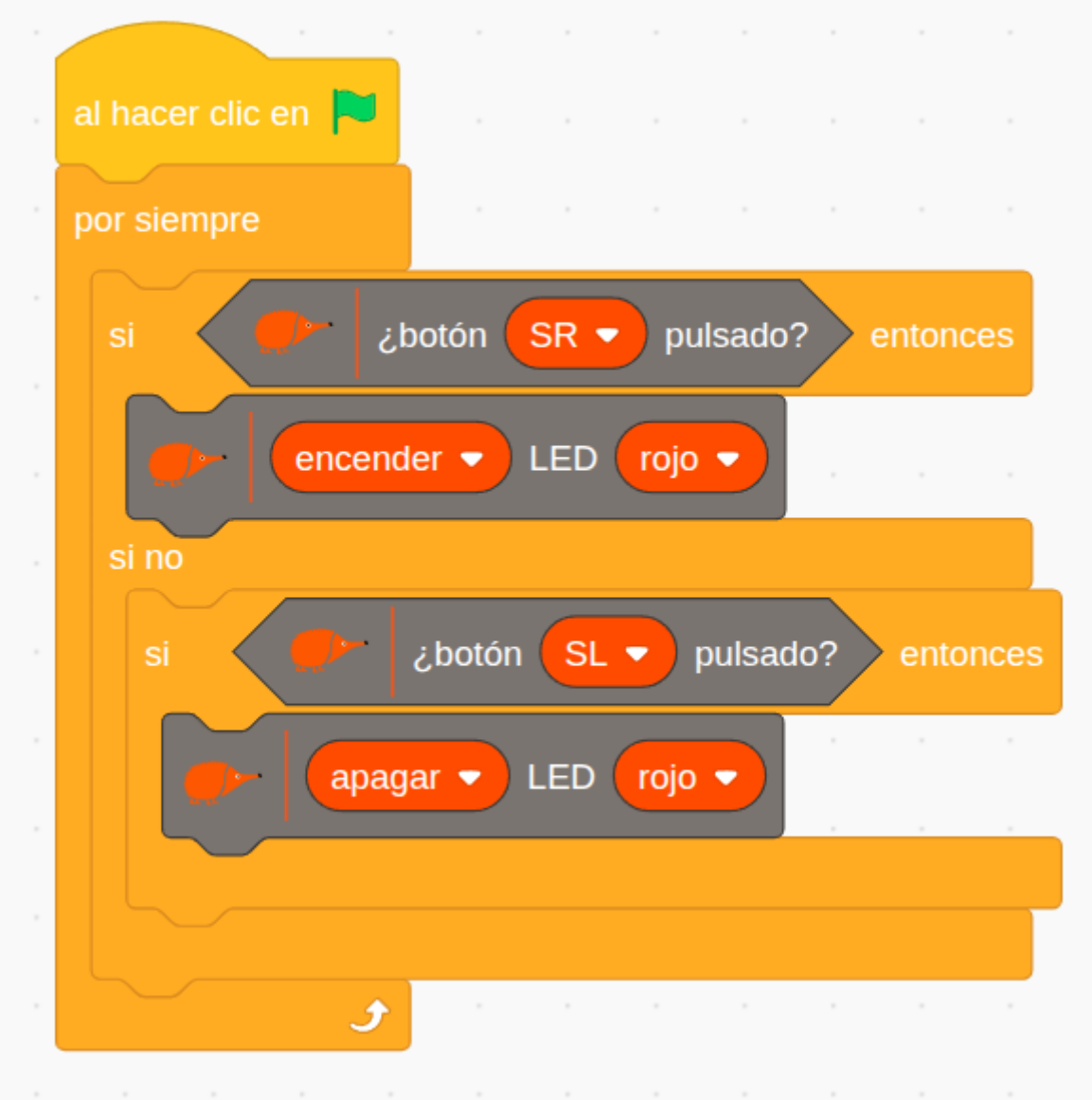
Para **leer** el **estado** del **pulsador** podemos usar el siguiente **bloque de programación**:


**Valor**: cuando leemos el estado del pulsador nos **devuelve true** (1) si está pulsado y **false** (0) si no está pulsado.

El bloque nos permite seleccionar el pulsador derecho SR o el izquierdo SL.

### EJEMPLO: Control encendido LED con dos pulsadores

Este ejemplo utiliza **dos pulsadores** para **controlar** el **encendido** y **apagado** del **LED rojo**. El pulsador **derecho** (SR) controla el **encendido** del LED rojo y el pulsador **izquierdo** (SL) el **apagado**.




**Lógica de programación:**

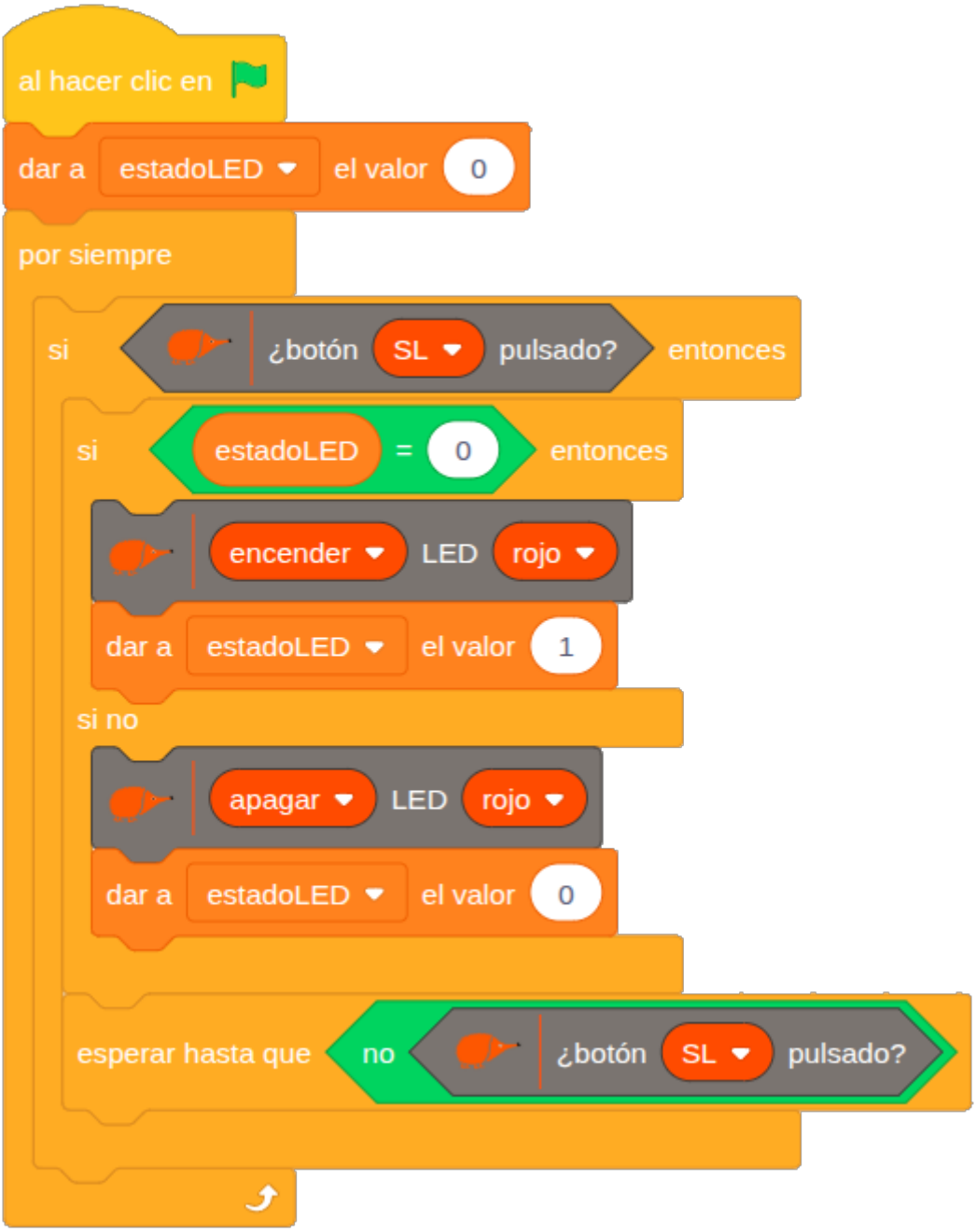
El programa comprueba continuamente:

```
SI el pulsador derecho (SR) está presionado:
    --> Enciende el LED Rojo inmediatamente.

SI NO (es decir, si SR no está presionado el programa comprueba la segunda
condición):
    SI el pulsador izquierdo (SL) está presionado:
        --> Apaga el LED Rojo.
```

### EJEMPLO: Pulsador con memoria

Este ejemplo ilustra cómo **programar** un **pulsador** para que actúe como un **interruptor**. Es decir, la primera pulsación enciende un LED y la siguiente pulsación lo apaga.

Para que el sistema pueda "**recordar**" si el LED está actualmente encendido o apagado, utilizamos una **variable** de estado denominada **estadoLED**. De esta forma, la variable estadoLED actúa como la memoria que permite al pulsador alternar entre los dos estados con cada activación.




**Lógica de programación:**

```
SI se pulsa el botón, el programa consulta el valor actual de la variable estadoLED:
    SI estadoLED indica APAGADO (valor 0), el programa:
        --> Enciende el LED.
        --> Cambia el valor de estadoLED a 1 (encendido).

    SI NO (si estadoLED indica ENCENDIDO valor 1):
        --> Apaga el LED.
        --> Cambia el valor de estadoLED a valor 0 (apagado).
```

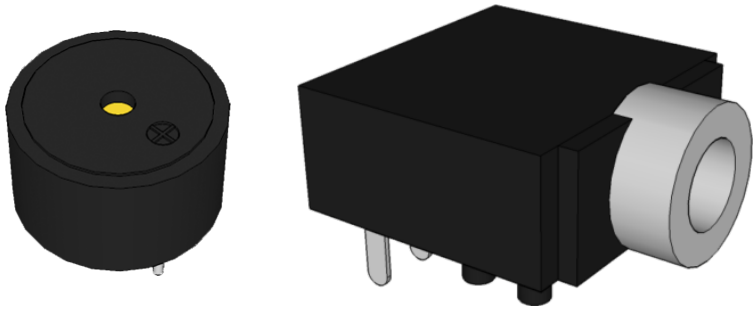
Para asegurar que el cambio de estado (encendido → apagado, o viceversa) ocurra una única vez por cada pulsación, es crucial evitar que el programa registre múltiples activaciones mientras el botón se mantiene presionado.

Para ello, se añade un bloque "esperar hasta que" que mantiene la ejecución del programa en pausa hasta que el pulsador sea liberado. Esta técnica previene eficazmente el efecto de rebote y garantiza que la variable de estado se altere una sola vez por cada interacción física, logrando un control de memoria estable y predecible.

### 4.1.3 Zumbador: Pulsador-sonido

#### COMPONENTE:

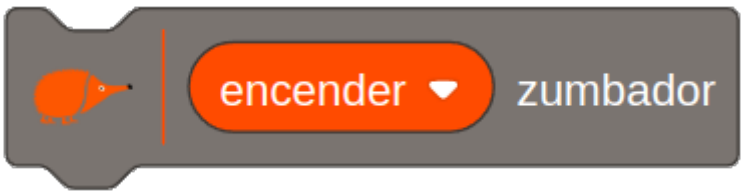
Disponemos de dos salidas para reproducir audio, el **zumbador** y el **jack** al que podemos conectar auriculares o altavoces autoamplificados. Al conectar una clavija de audio en el jack se desconecta el zumbador.



Además, contamos con un potenciómetro que permite ajustar el volumen del sonido.

#### BLOQUE DE PROGRAMACIÓN:

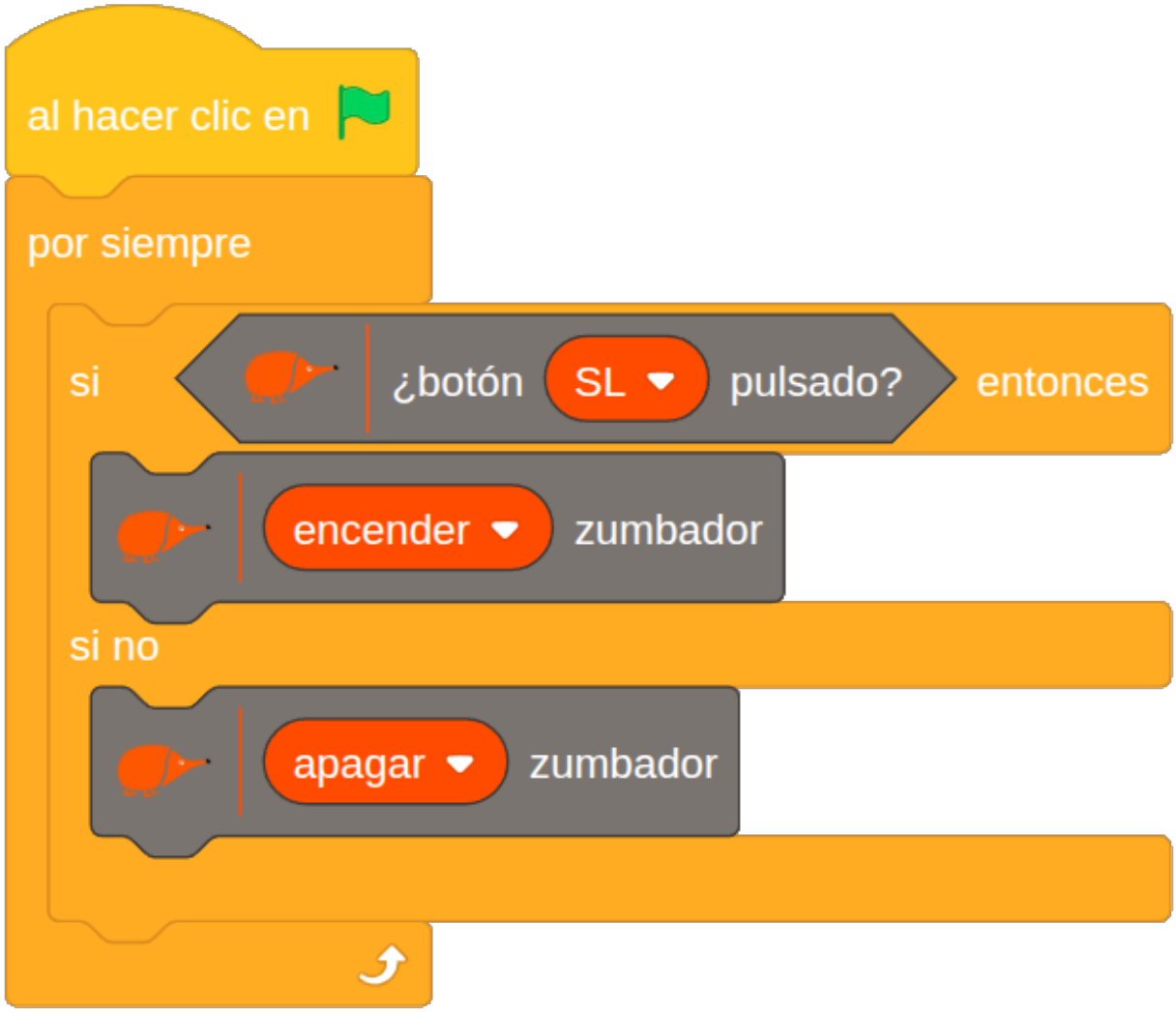
Para controlar el **zumbador** podemos usar el siguiente **bloque**:




En el cual podemos cambiar su estado a encender o apagar.

### EJEMPLO: Timbre

Este ejemplo muestra cómo **controlar** el **zumbador** (actuador de sonido) utilizando un **pulsador** como **entrada**.




**Lógica de programación:**

```
SI el pulsador es presionado (o activado):
    --> El zumbador suena.

SI el pulsador es liberado (o soltado):
    --> El zumbador deja de sonar.
```

De esta forma, el zumbador solo se activa mientras el pulsador se mantiene presionado.

### 4.1.4 Sensor de Luz (LDR): Interruptor crepuscular

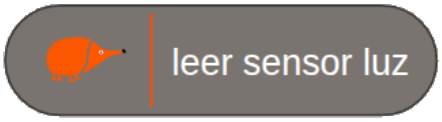
**COMPONENTE:**

**LDR** es el acrónimo de “Light Dependent Resistor” (resistencia dependiente de la luz), y es una **resistencia** cuyo **valor** depende de la **cantidad** de **luz** que incide sobre ella.

En la placa EchidnaBlack2 podemos encontrar la LDR en la esquina superior derecha.

**BLOQUE DE PROGRAMACIÓN:**

Para **leer** el valor del **sensor** podemos usar el siguiente **bloque**:



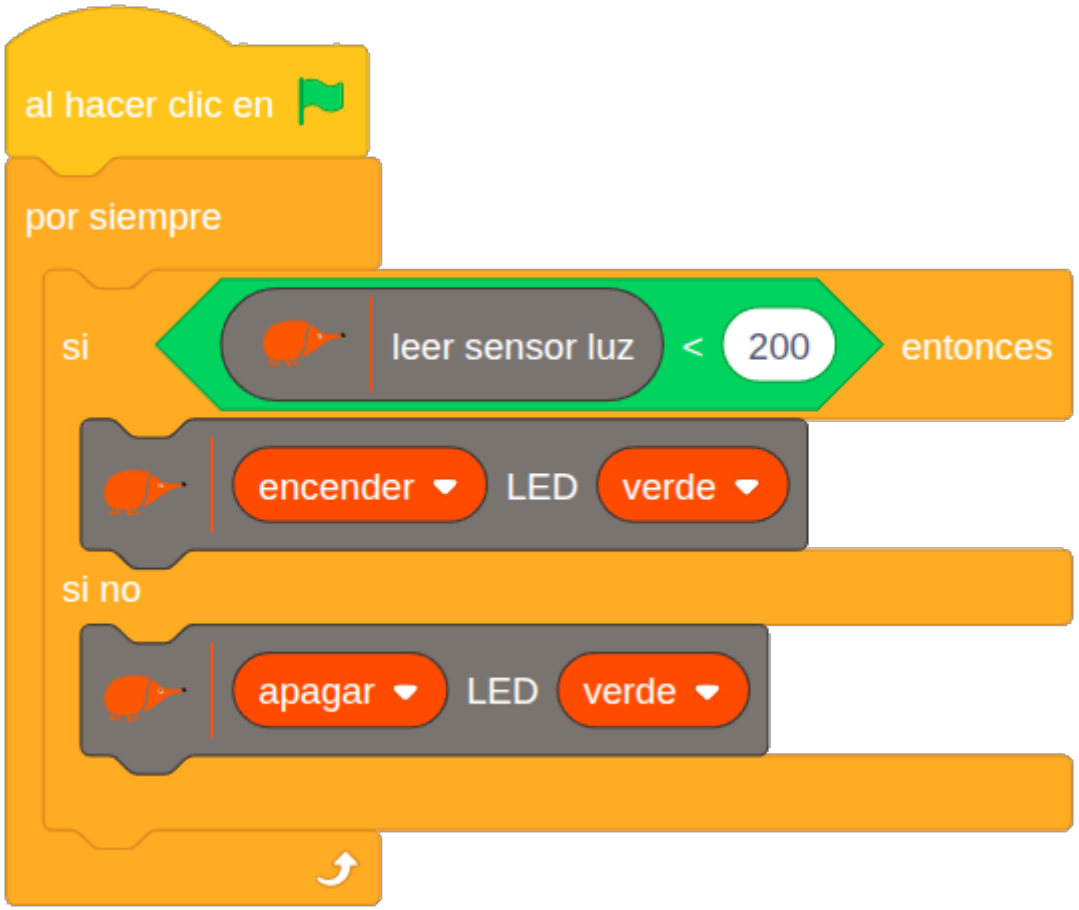

Puedes activar la casilla de verificación para ver el valor registrado.

**Valores**: el sensor de luz proporciona valores bajos con poca luz y valores altos con mucha luz. Con una variabilidad entre 0 (no hay luz) y 1023 (mucha luz).

- 0 ausencia de luz
- 1023 mucha luz

**EJEMPLO: Interruptor crepuscular**

Este ejemplo muestra cómo **controlar** automáticamente el **encendido** de un **LED** en función de la **intensidad** de la **luz** ambiental detectada por la LDR.




**Lógica de programación:**

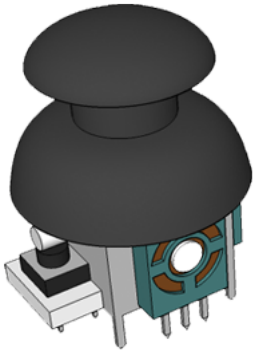
El programa revisa continuamente:

```
SI el sensor de luz registra valores menores de 200:
    --> Se enciende el LED verde.

SI NO (si registra valores mayores):
    --> Se apaga el LED.
```

El valor 200 actúa como el umbral que define cuándo debe encenderse o apagarse la luz.

## 4.1.5 Joystick: Pintamos

### COMPONENTE:



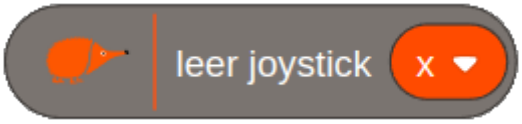
Es un **dispositivo** de **entrada** con una palanca que se mueve en varias direcciones. Al inclinar la palanca, el joystick envía señales que indican hacia dónde y cuánto se ha desplazado respecto a dos ejes: horizontal (X) y vertical (Y).

Consta internamente de dos potenciómetros, uno para el eje X (movimiento horizontal) y otro para el eje Y (movimiento vertical), que convierten la posición del stick en valores de resistencia.

**Pulsador adicional:** este modelo incorpora además un pulsador (botón) que se activa al presionar el stick hacia abajo. En EchidnaBlack2, el pulsador del Joystick está conectado directamente con el pulsador "SR".

### BLOQUE DE PROGRAMACIÓN:

Para leer el valor del joystick podemos usar el siguiente bloque:

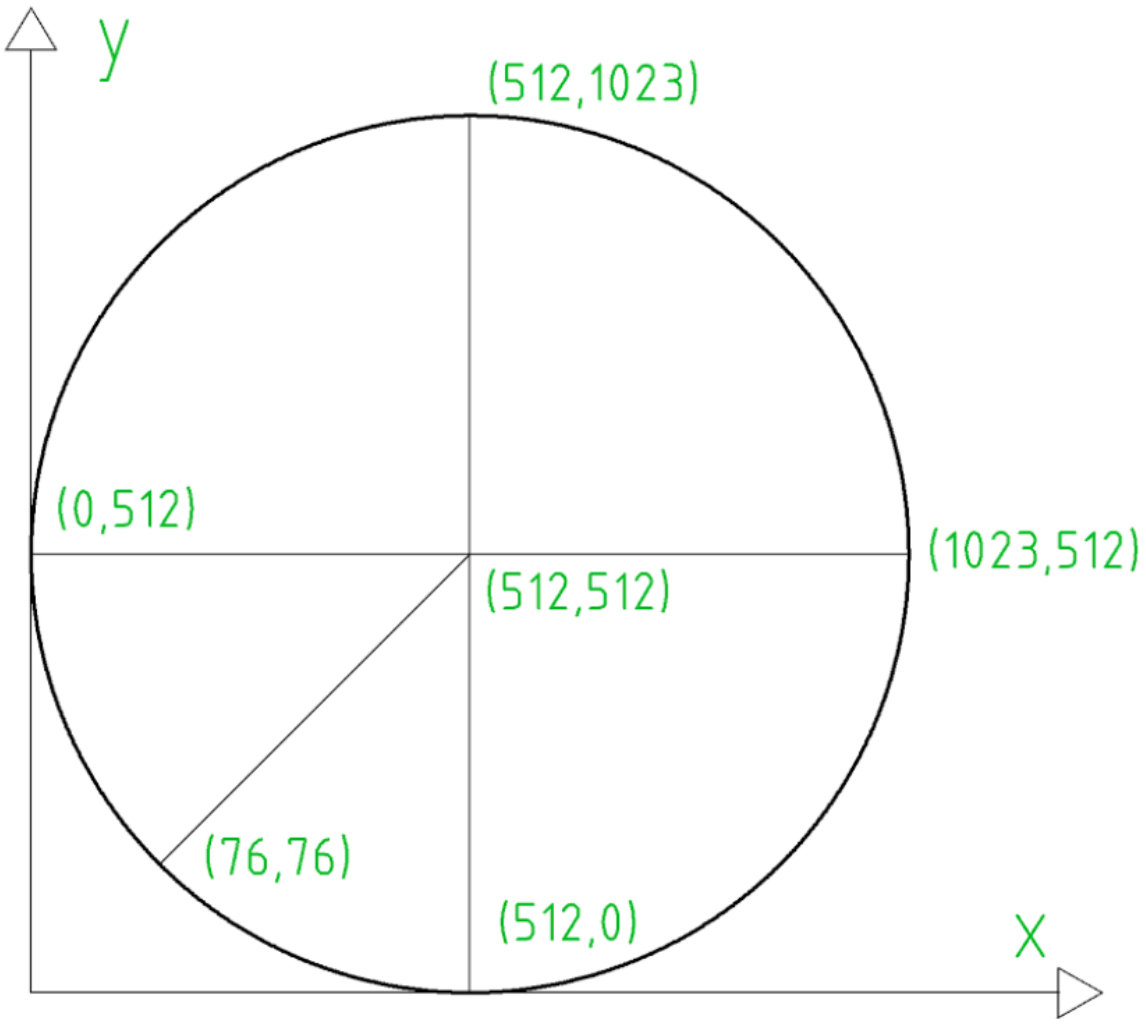



En el bloque podemos seleccionar eje x, o eje y

**Valores proporcionados por el joystick:**

- El joystick en reposo proporciona valores en torno a 512 en el eje x e y.
- En el eje x proporciona valor de 0 a la izquierda y valor 1023 a la derecha.
- En el eje y valor 0 abajo y 1023 arriba.

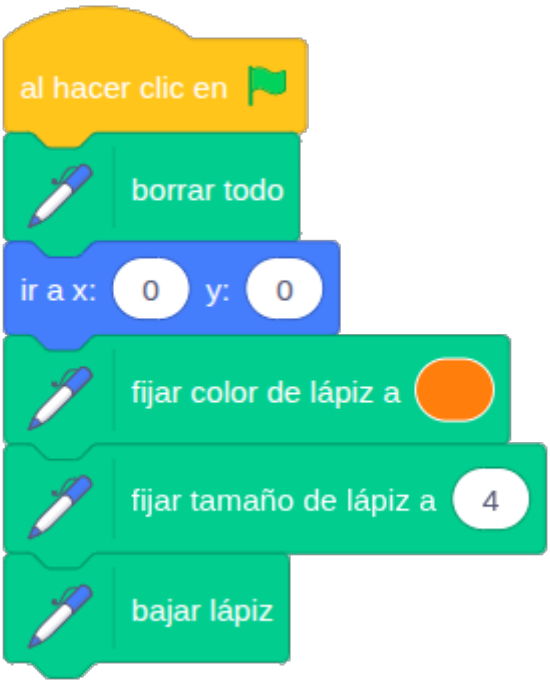
**Gráfica de valores del joystick:**




**EJEMPLO: Pintamos**

Este ejemplo utiliza el Joystick (control de dos ejes) para controlar el movimiento de un puntero o lápiz virtual en la pantalla del entorno de programación.

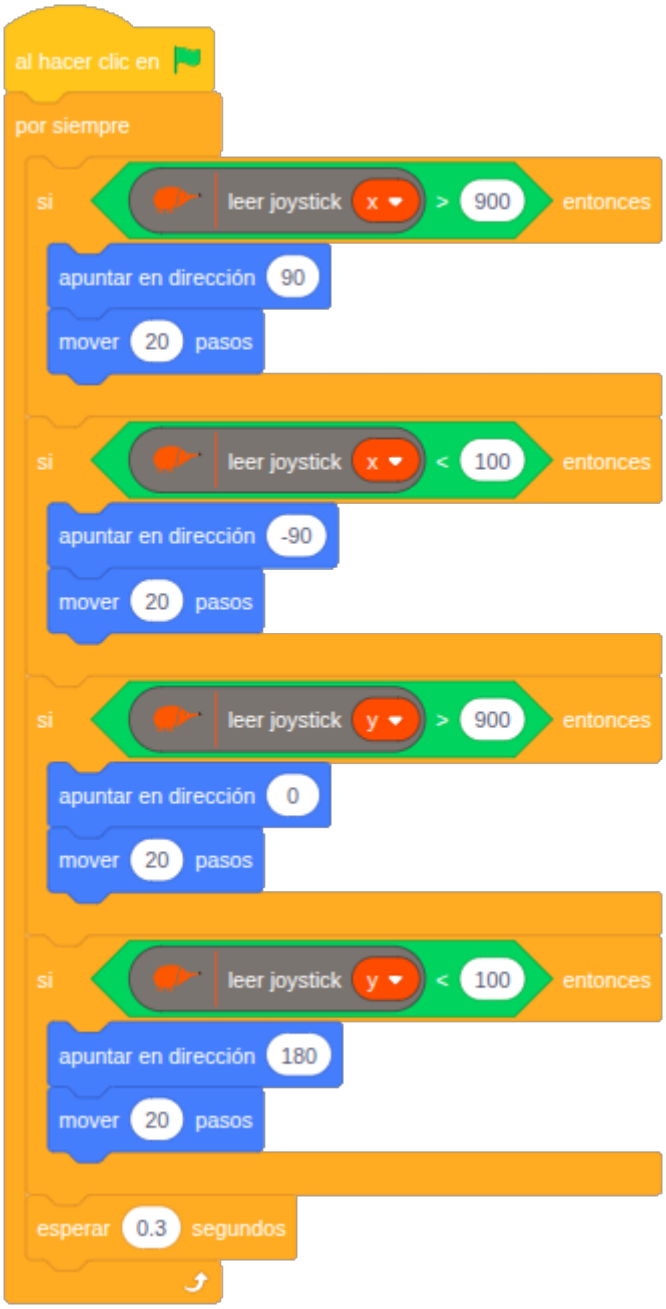
**Cargar la herramienta lápiz:** Lo primero que tenemos que hacer es cargar la herramienta lápiz.




**Configuración inicial:** Configuramos la herramienta lápiz para que al inicio

- Borre el fondo.
- Se vaya al punto (0,0).
- Fije el color y el grosor con el que se va a pintar.
- Baje el lápiz.

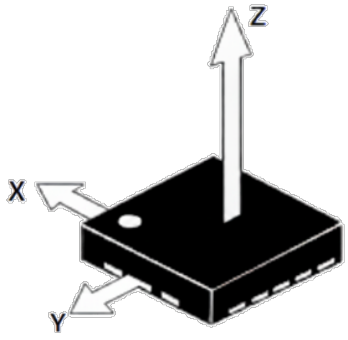
**Lógica de programación de control del joystick:**


al hacer clic en
por siempre
si
leer joystick x > 900 entonces
apuntar en dirección 90
mover 20 pasos
si
leer joystick x < 100 entonces
apuntar en dirección -90
mover 20 pasos
si
leer joystick y > 900 entonces
apuntar en dirección 0
mover 20 pasos
si
leer joystick y < 100 entonces
apuntar en dirección 180
mover 20 pasos
esperar 0.3 segundos

```
SI el joystick se desplaza a la derecha y registra valores mayores de 900 en el eje
x:
    --> El puntero se desplaza hacia la derecha.

SI el joystick se desplaza a la izquierda y registra valores menores de 100 en el eje
x:
    --> El puntero se desplaza hacia la izquierda.

SI el joystick se desplaza hacia arriba y registra valores mayores de 900 en el eje
y:
    --> El puntero se desplaza hacia arriba.

SI el joystick se desplaza hacia abajo y registra valores menores de 100 en el eje y:
    --> El puntero se desplaza hacia abajo.
```

### 4.1.6 Acelerómetro: Movemos el echidna

**COMPONENTE:**



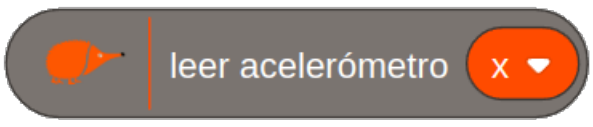

Es un **sensor microelectromecánico** (MEMS) de aceleración que **mide** los **movimientos** en los tres ejes: X, Y y Z.

**Mide la inclinación en los ejes X e Y: e**sto es posible porque la aceleración de la gravedad (1g) genera un cambio de capacitancia proporcional al ángulo de inclinación del sensor

**Permite detectar cambios bruscos o dinámicos de movimiento en el eje Z.: c**ualquier movimiento vertical repentino provoca una variación rápida en la aceleración medida a lo largo de este eje.

**BLOQUE DE PROGRAMACIÓN:**

Para leer el valor del acelerómetro podemos usar el siguiente **bloque**:




En el bloque **seleccionamos** eje x, eje y o eje z.

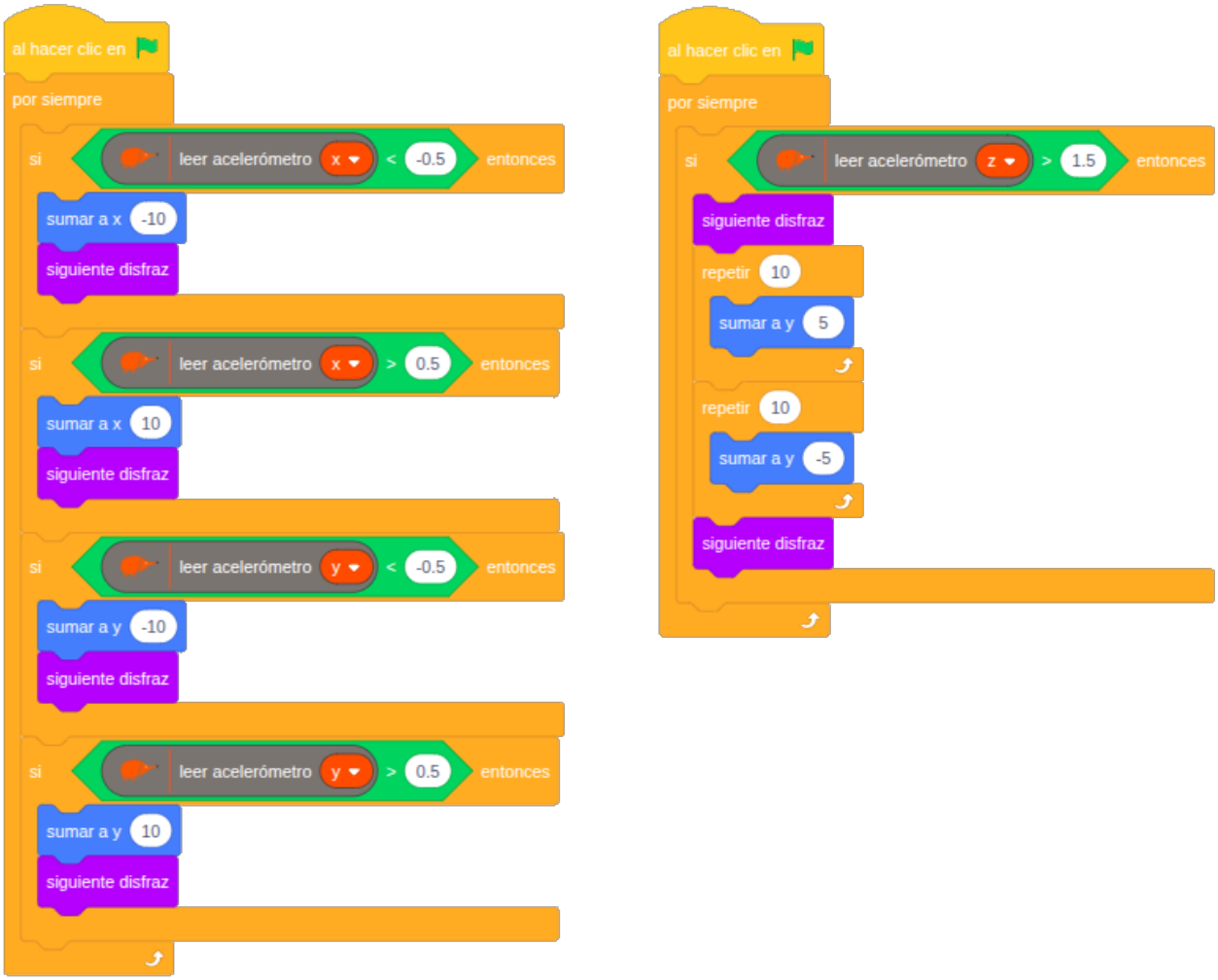
**Valores:**

- El acelerómetro en reposo proporciona valores en torno a 0 en los ejes x e y, y 1 en el eje z.
- En el eje x proporciona valor de 0 a -1 al elevar la parte derecha y de 0 a 1 al elevar la parte izquierda.
- En el eje y proporciona valor de 0 a -1 al elevar la parte trasera y de 0 a 1 al elevar la parte delantera.
- En el eje z se pueden registrar valores de menores de -2 al subir la placa rápidamente y de más de 2 al bajarla rápidamente.

### EJEMPLO: Movemos el echidna

Este programa permite que **movamos** el **personaje** en la **pantalla** mediante la **inclinación** de la **placa**.

Programamos la placa en dos hilos de ejecución:




**Hilo de control de movimiento:**

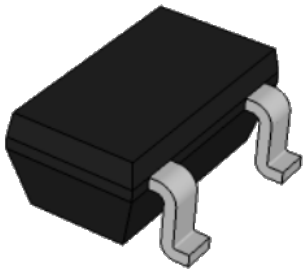
```
SI la placa se inclina a la izquierda y registra valores menores de -0,5 en el eje x:
    --> El personaje se desplaza hacia la izquierda.

SI la placa se inclina a la derecha y registra valores mayores de 0,5 en el eje x:
    --> El personaje se desplaza hacia la derecha.

SI la placa se inclina hacia atrás y registra valores menores de -0,5 en el eje y:
    --> El personaje se desplaza hacia abajo.

SI se inclina hacia delante y registra valores mayores de 0,5 en el eje y:
    --> El personaje se desplaza hacia arriba.
```

**Hilo de control de salto:**

```
SI se eleva bruscamente la placa y el eje z registra valores mayores de 1,5:
    --> El personaje efectúa un salto.
```

### 4.1.7 Sensor de temperatura: El echidna dice la temperatura

**COMPONENTE:**



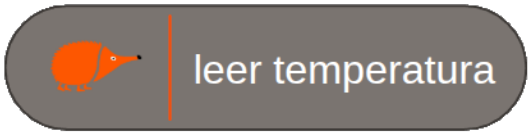
El MCP9700T es un sensor que entrega un voltaje analógico proporcional a la temperatura en grados Celsius (°C).

- **Sensibilidad**: cada 10 mV equivalen a 1°C.
- **Offset**: posee un voltaje de 500 mV (0.5V) a 0°C.

**Fórmula de Conversión:** la temperatura en grados Celsius se calcula como: $T(°C)=(V-0.5)\times100$.

**BLOQUE DE PROGRAMACIÓN:**

Para leer la temperatura podemos usar el siguiente **bloque**.

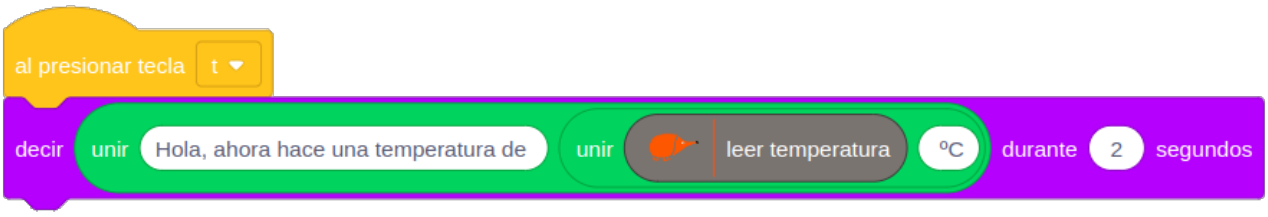



El bloque utiliza la fórmula de conversión anterior para convertir la tensión de lectura en voltios a °C.

Puedes activar la casilla de verificación para ver el valor registrado.

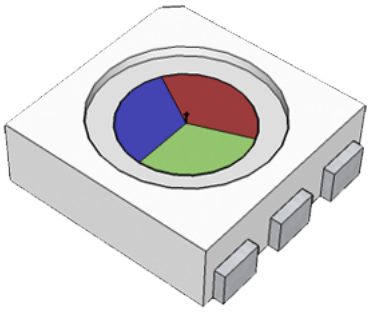
**EJEMPLO: El echidna dice la temperatura**

En el ejemplo el echidna nos dice qué temperatura hace cuando pulsamos la letra "t" del teclado del ordenador.


## 4.1.8 LED RGB: Mezclamos colores

**COMPONENTE:**



El **LED RGB** es un único componente que **integra tres diodos** emisores de luz (LEDes) independientes **Rojo, Verde y Azul** dentro de la misma cápsula.

El acrónimo significa Light Emitting Diode (Diodo Emisor de Luz) y Red Green Blue (Rojo, Verde, Azul).

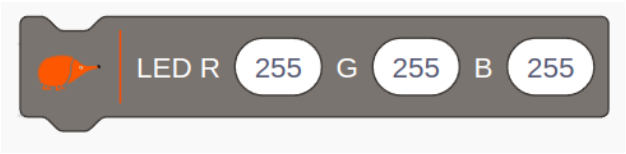
**Control de Luminosidad LED (PWM):** la luminosidad de cada uno de los tres LEDs se puede ajustar individualmente utilizando el control PWM (Modulación por Ancho de Pulso), a menudo denominado "analógico" en este contexto.

**Generación de Colores:** podemos controlar el brillo de cada color (R, G y B) variando su intensidad desde 0 (apagado) hasta 255 (máxima intensidad).

**Capacidad Cromática:** dado que cada canal ofrece 256 niveles de intensidad, la combinación de los tres colores (Rojo, Verde y Azul) permite generar un total de 256×256×256 = 16.777.216, más de 16 millones de colores diferentes.

### BLOQUE DE PROGRAMACIÓN:

Para controlar la luminosidad y el color del LED RGB tenemos el siguiente **bloque**:



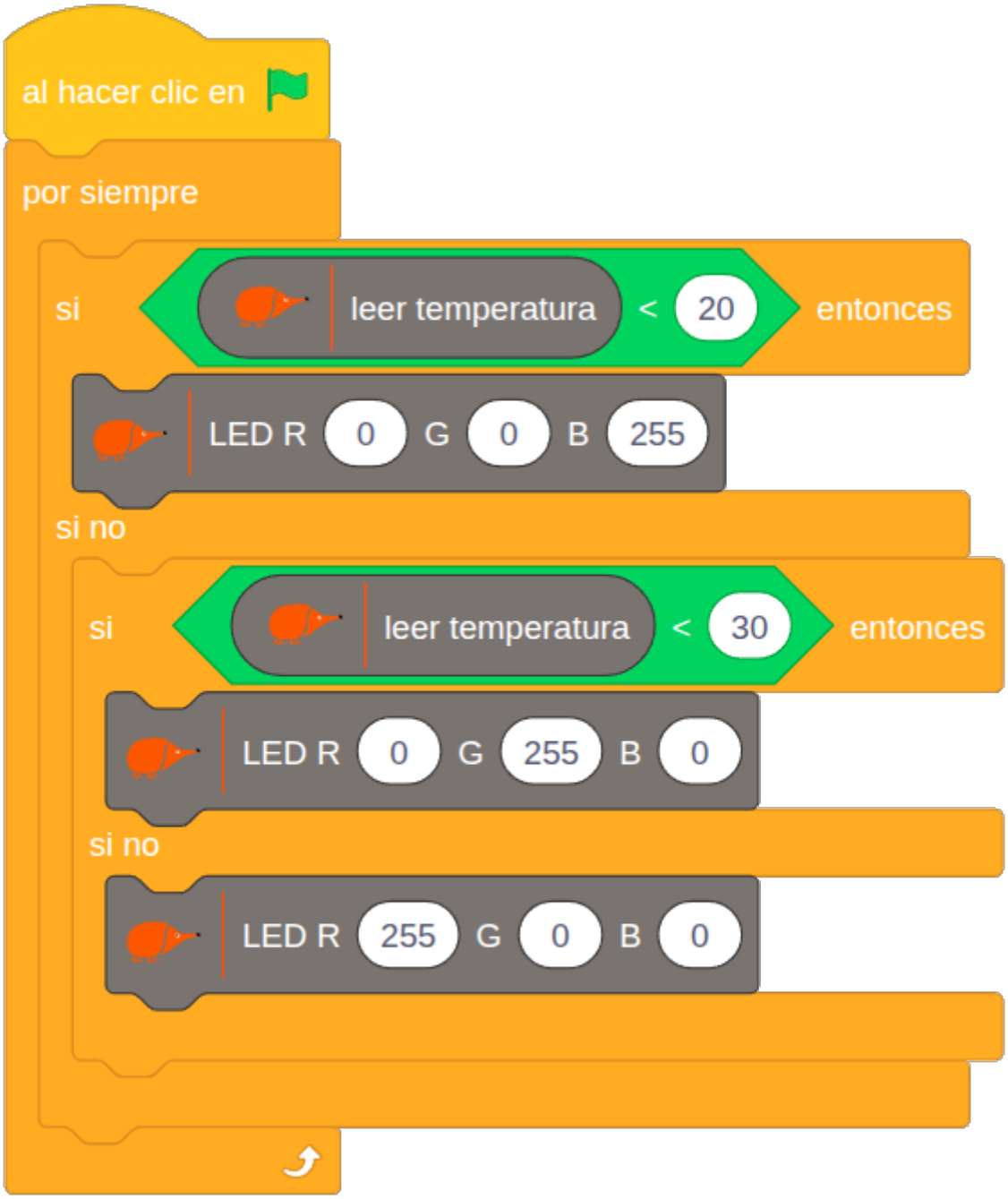

En el bloque podemos ajustar el valor de cada LED entre 0 y 255.

**Por ejemplo:** si queremos reproducir el naranja Echidna debemos establecer los siguientes valores:


### EJEMPLO: Indicador de Temperatura RGB

Este **ejemplo** utiliza el **sensor de temperatura** como dato de entrada para **controlar** el **color** del **LED RGB** (actuador). El **objetivo** es que el LED cambie de color automáticamente para indicar visualmente si la temperatura ambiente es baja (Azul), media (Verde) o alta (Rojo), funcionando como un termómetro visual.

El programa evalúa continuamente la temperatura ambiente y asigna un color específico según el umbral que se cumpla.




**Lógica de programación:**

**Zona Fría** (Alerta Azul):

```
SI la temperatura es inferior a 20°C:
    --> El LED RGB se ilumina en color azul.
```

**Zona Media** (Temperatura Óptima/Verde):

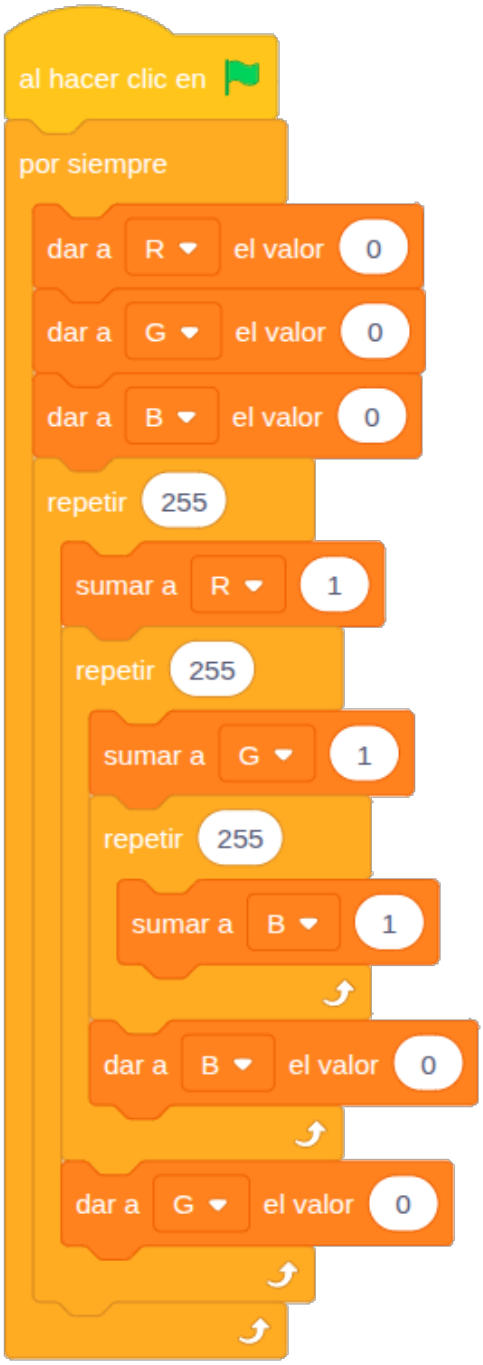
```
SI NO:
    SI la temperatura es menor de 30°C (está entre 20°C y 30°C):
        --> El LED RGB se ilumina en color verde.
```

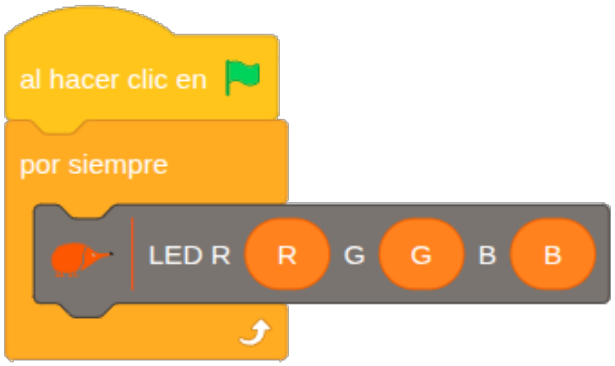
**Zona Caliente** (Alerta Roja):

```
SI NO (es decir, si la temperatura supera los 30°C):
    --> El LED RGB se ilumina en color rojo.
```

### EJEMPLO: Recorremos los 16.7 millones de colores

Este programa utiliza tres bucles anidados de repetición para recorrer sistemáticamente la totalidad de las combinaciones de color. Al variar la intensidad de cada canal (Rojo, Verde y Azul) en sus 256 niveles posibles, el sistema es capaz de generar los más de 16.7 millones de colores únicos que componen el espectro RGB.

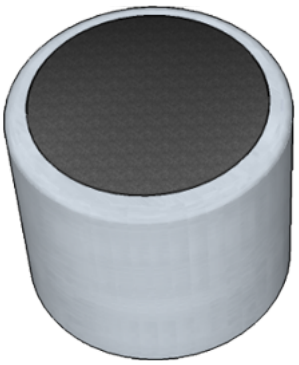



256×256×256 = 16.777.216 colores

### 4.1.9 Micrófono: Vúmetro

**COMPONENTE:**



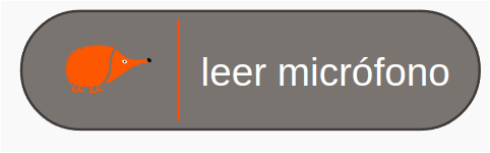
Es un transductor acústico-eléctrico. Utiliza el efecto piezoeléctrico para convertir las vibraciones de sonido en una señal eléctrica.

**Principio de funcionamiento**: al recibir una onda sonora, el material piezoeléctrico genera una señal eléctrica que reproduce las mismas características (frecuencia y amplitud) del sonido captado.

**Variabilidad de la señal:** la señal eléctrica refleja directamente el sonido recibido, por lo que presenta una gran variabilidad. Se trata de una señal analógica compleja y que cambia constantemente, por lo que para poder trabajar adecuadamente con ella, se requiere un procesamiento posterior para su análisis (por ejemplo hallando la media aritmética), o, como alternativa, puede limitarse a detectar únicamente la intensidad del sonido.

**BLOQUE DE PROGRAMACIÓN:**

Para leer el valor del sensor podemos usar el siguiente **bloque**:




Puedes activar la casilla de verificación para ver el valor registrado.

**Valores**: el micrófono proporciona valores bajos en presencia de poco sonido o silencio, y valores más altos cuando capta sonidos intensos. Con una variabilidad entre 0 (no hay sonido) y 1023 (sonido de alta intensidad).

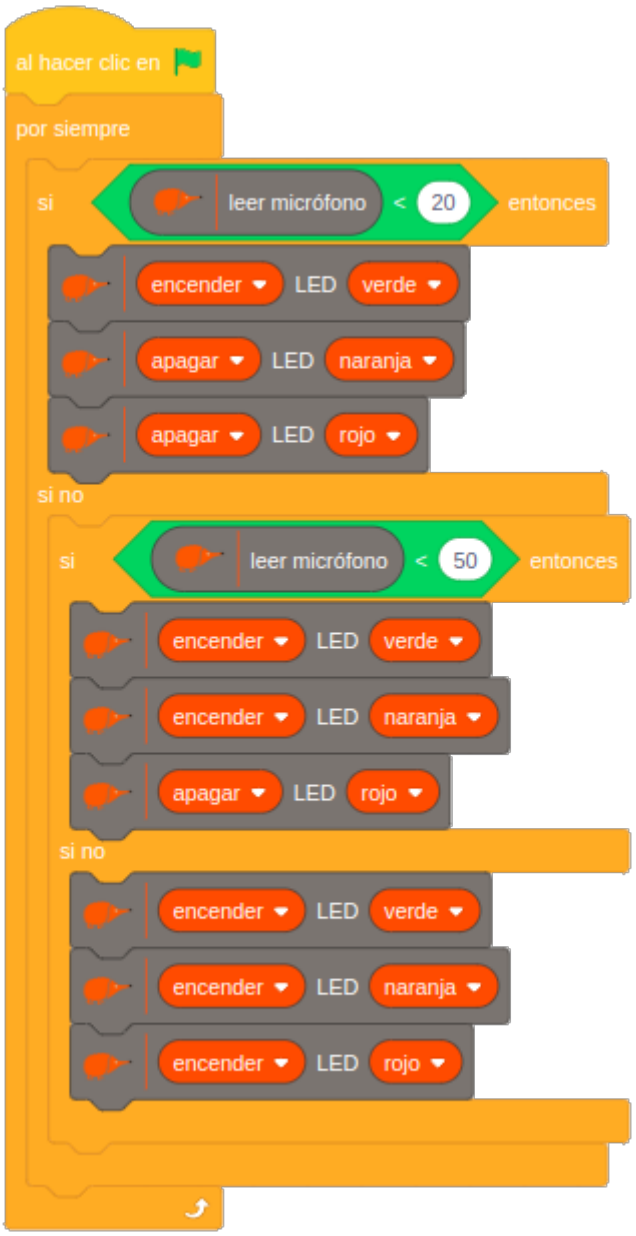
**EJEMPLO: Vúmetro**

Este ejemplo programa la placa para actuar como un semáforo de ruido o vúmetro, que indica visualmente la intensidad del sonido ambiente. El sistema opera según tres umbrales de sonido:

**Nivel Bajo:** si el sensor de sonido registra valores menores a 20 (ambiente silencioso), solo se enciende el LED verde.

**Nivel Medio:** si los valores están entre 20 y 50, se encienden los LEDs verde y naranja (precaución).

**Nivel Alto:** si el valor es superior a 50, se encienden los tres LEDs (alerta).



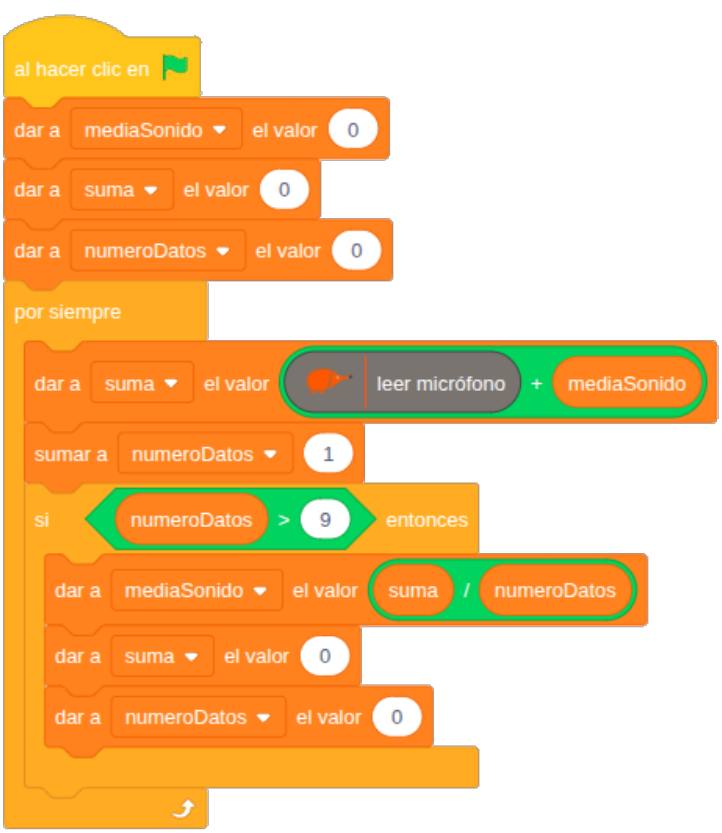

Es probable que, al ejecutar este código, se observe que el funcionamiento es inestable y los LEDs parpadean constantemente. Esto ocurre debido a la variabilidad de la señal de sonido.

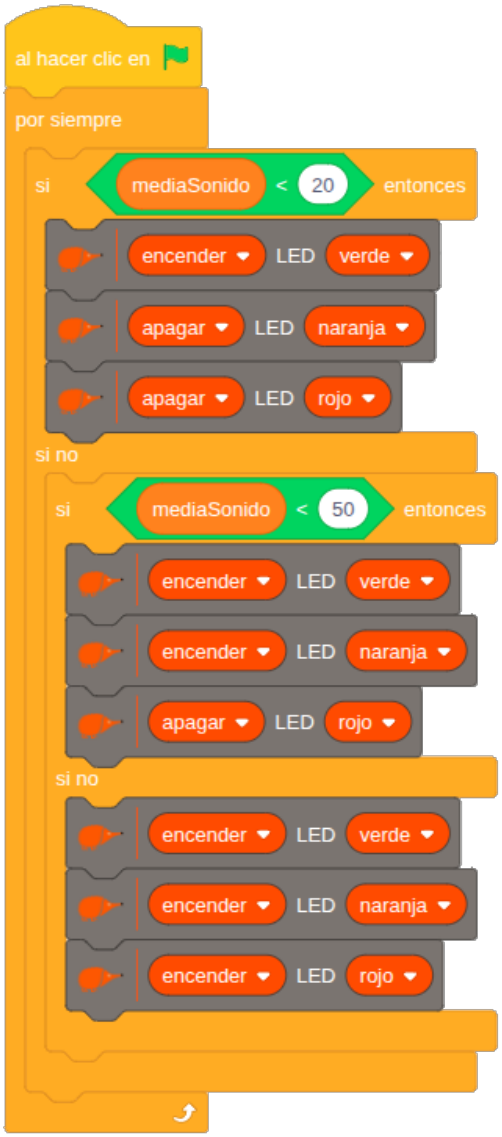
Para solucionar esto mostramos el siguiente ejemplo.

### EJEMPLO: Vúmetro con media

Como hemos visto, la señal de sonido, se caracteriza por tener mucha variabilidad. Por ello, resulta conveniente aplicar un **tratamiento** o **filtrado** a la **señal**.

Una técnica efectiva para reducir esta variabilidad es el cálculo de la **media móvil** (o promedio). Esto implica tomar un número fijo de mediciones sucesivas y promediarlas.




**Cálculo de la media:** para implementar el cálculo de la media

1. Creamos las siguientes variables:
    1. Variable suma para acumular los valores medidos en un ciclo.
    2. Variable numeroDatos para llevar la contabilidad del número de medidas.
    3. Variable mediaSonido para almacenar el valor de la media.
2. Cada diez medidas (o el número de muestras predefinido):
    1. Calculamos el valor promedio (mediaSonido) dividiendo la suma entre el número de muestras.
    2. Inicializamos las variables suma y numeroDatos.

De esta forma, se suaviza la lectura y se obtienen valores más estables y representativos de la intensidad del sonido ambiente.

## 4.1.10 Entrada MkMk: Piano

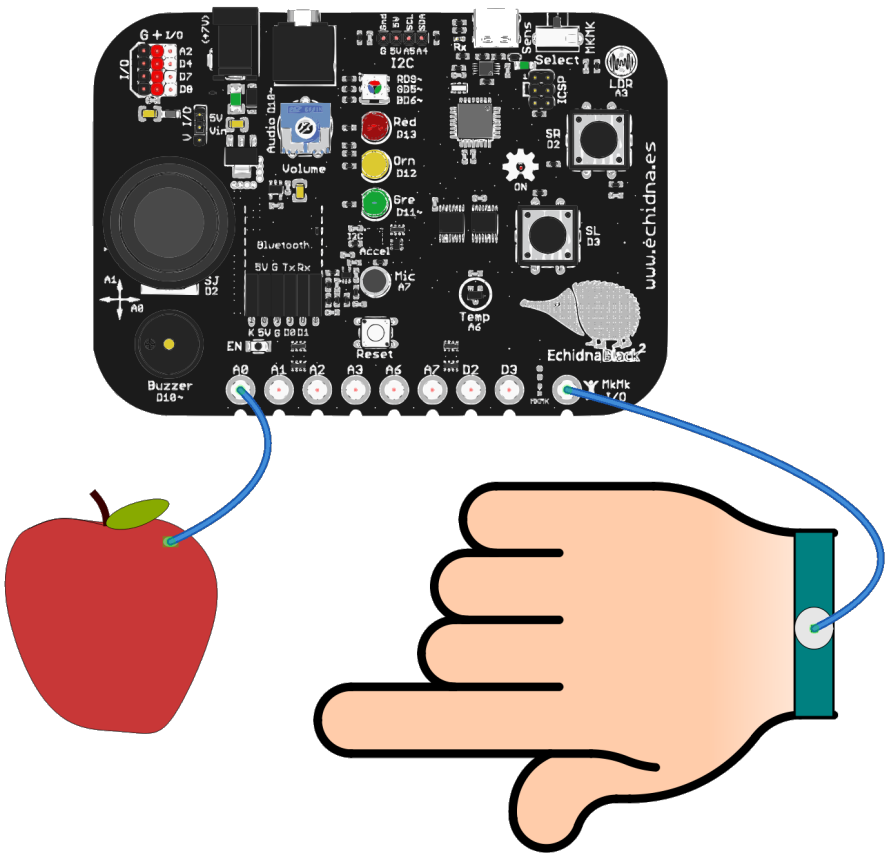
### COMPONENTE:

**¡ATENCIÓN!** Para que funcione el modo MkMk debemos poner el selector del modo de funcionamiento hacia la derecha, y se nos encenderá el LED testigo en la parte inferior.

Echidna dispone de 8 conexiones MkMk.

Una conexión MkMk es un conector que permite detectar gran variedad de objetos al conectarlos entre una entrada y el común (El logo Echidna también se comporta como común).

Para detectar elementos debemos conectar un cable al común y otro a una de las entradas.




Entradas MkMk: A0, A1, A2, A3, A6, A7, D2, D3.

### BLOQUE DE PROGRAMACIÓN:

Echidna dispone de un bloque de programación específico para las entradas MkMk que nos devuelve un **true** o un **false** en función de si detecta que el **circuito** se ha **cerrado** o es un circuito **abierto**.

En el bloque podemos seleccionar la entrada MkMk que queramos utilizar.

**Valores**:

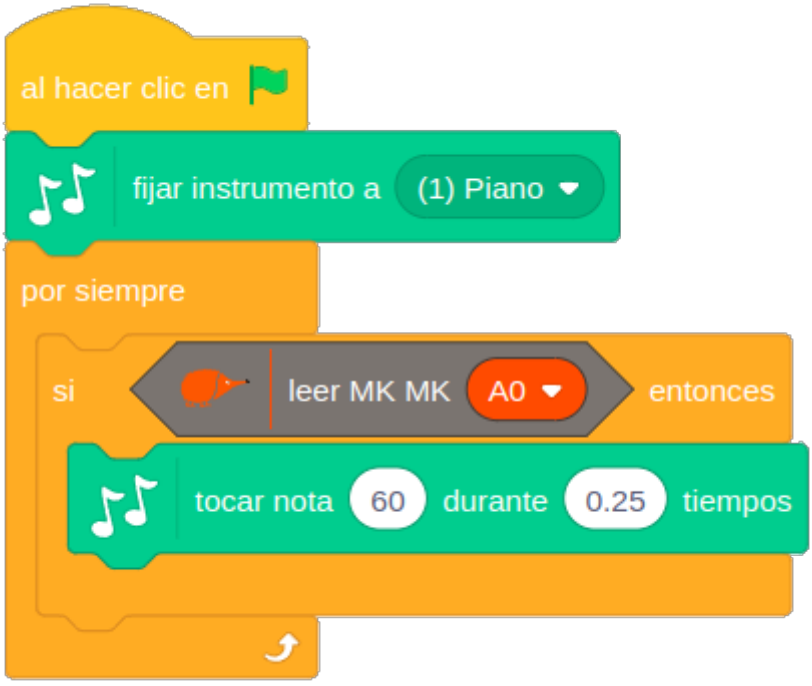
- Con el circuito abierto el sensor da valor **false** (0).
- Cuando cerramos el circuito, si el valor leído en la entrada analógica es mayor de 350, el valor reportado por el bloque es **true** (1).
- El **umbral** para cambiar de **false** a **true** está establecido en 350 dentro del rango de lecturas analógicas que puede registrar la entrada MkMk (0–1023).

Si queremos cambiar el valor del umbral podemos usar el bloque leer entrada analógica tal como se muestra en el segundo ejemplo: Ajustar sensibilidad.

### EJEMPLO: Piano

En este ejemplo vamos a ver cómo programar una nota de piano, que suena cuando tocamos la entrada MkMk A0.

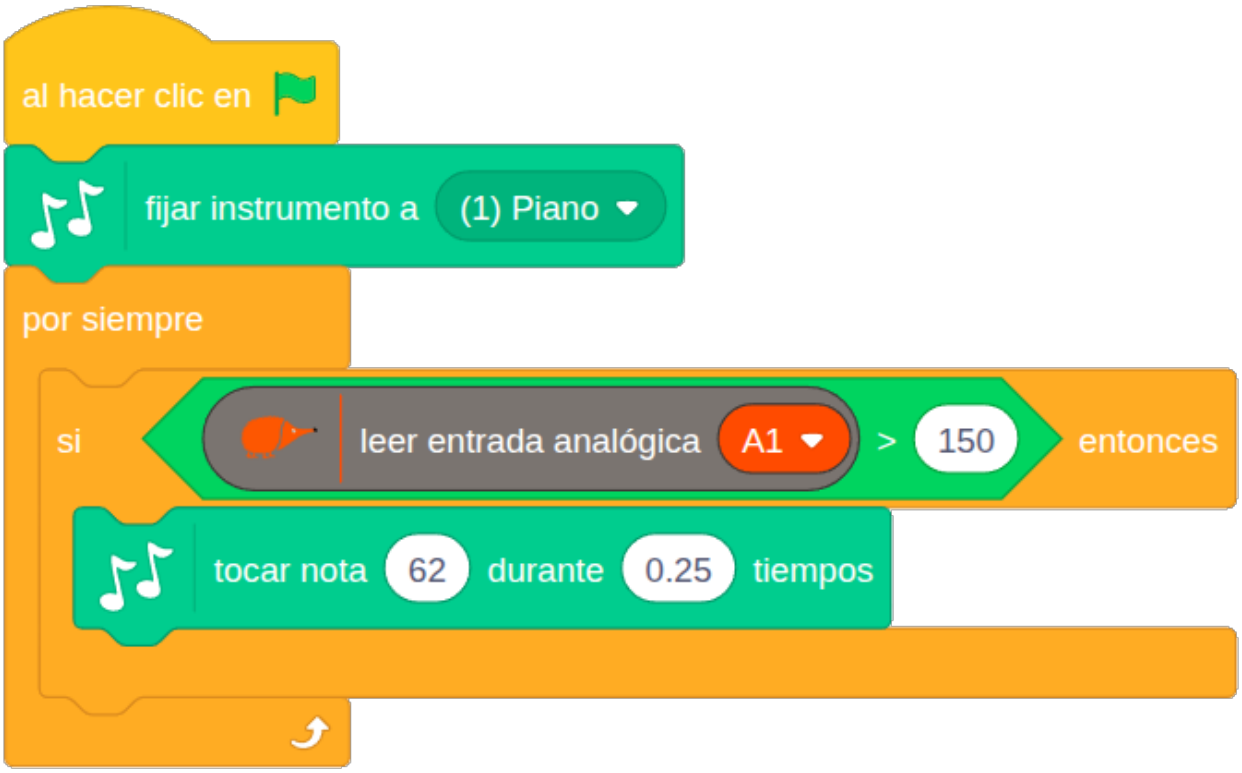
En este caso sonará la nota 60 del piano durante 0,25 s cada vez que se activa la entrada MkMk A0.




### EJEMPLO: Ajustar la sensibilidad

Para ajustar el umbral de activación de las entradas analógicas del módulo MkMk, podemos utilizar el bloque "leer entrada analógica Ax".

En este caso el piano suena cuando el valor de la lectura es mayor de 150.




Las entradas analógicas (A0, A1, A2, A3, A6, y A7) permiten definir un límite conductivo específico. Sin embargo, las entradas D2 y D3 son digitales. Por ello, solo detectan dos estados y no es posible realizar un ajuste del límite conductivo o umbral de activación.

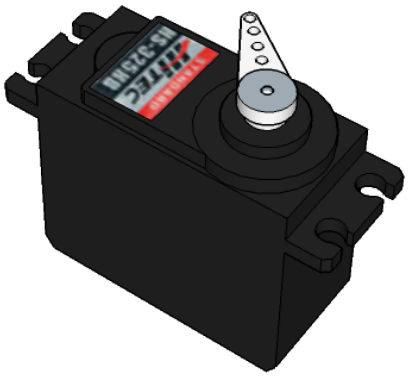
## 4.2 Componentes complementarios

En este apartado vamos a ver algunos de los componentes que podemos conectar a las entradas/salidas (I/O) de EchidnaBlack. Estos componentes no vienen incluidos con la placa.



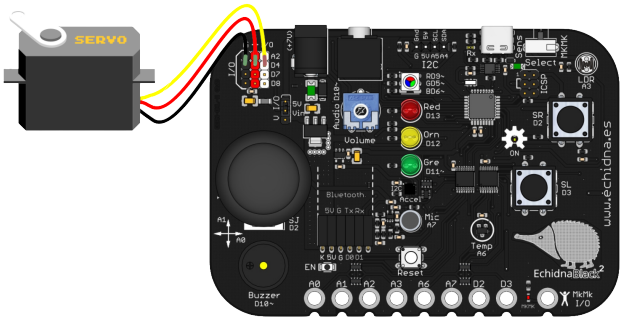
### 4.2.1 Servomotor de posición: Control posición servo con joystick

**COMPONENTE:**



Son **motores** de corriente continua con una reductora y electrónica de control que permiten **posicionarlo** en un **ángulo** entre **0** y **180º**.

Para conectarlo usamos los **pines I/O** de entrada-salida**: D4, D7, D8, A2**.

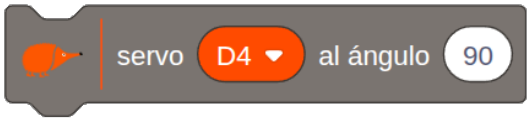



Presta atención al conectar los cables: Vcc, GND y señal, que están indicados por los colores rojo, negro y amarillo.

En caso de que vayas a conectar varios servomotores usa alimentación externa y coloca el selector de alimentación en la posición Vin. Ver apartado 2.4. Pines de entrada/salida.

**BLOQUE DE PROGRAMACIÓN:**

Para **controlar** el **servomotor** de **posición** podemos usar el siguiente **bloque**:



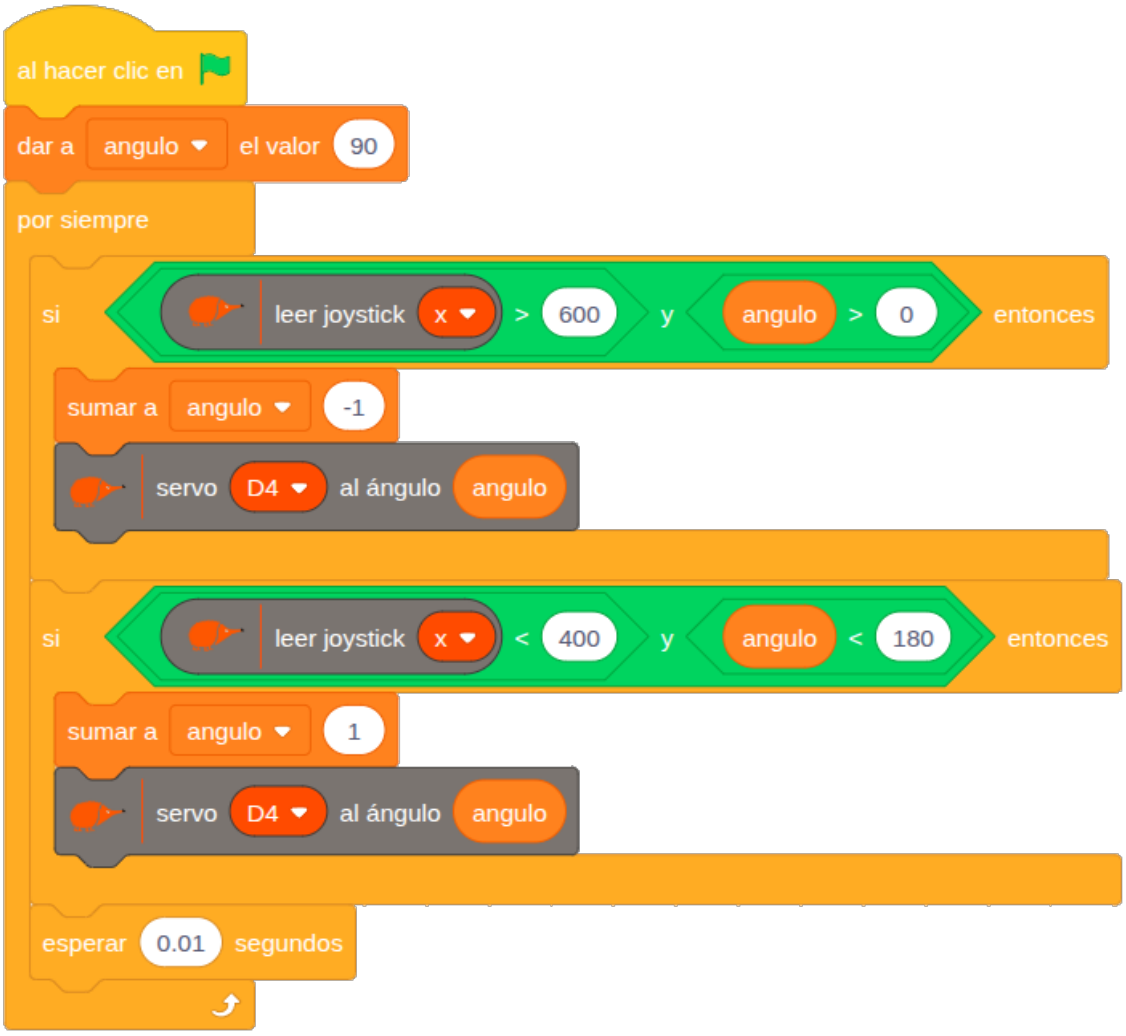
En el **bloque** podemos seleccionar el **pin** al que conectamos nuestro servomotor y el **ángulo** de **giro**.

**Pines**: podemos seleccionar los pines; **D4, D7, D8 y A2**.

**Ángulo** de giro: **0-180**º.

### EJEMPLO: Control de posición de servo con joystick

Este ejemplo muestra cómo **controlar** la **posición angular** de un **servomotor** de posición utilizando el **eje x** del **joystick** como entrada. La variable ángulo almacena la posición deseada y es la que determina la posición del motor.




**Lógica de programación:**

El programa ajusta el valor de la variable ángulo mediante dos condiciones:

```
SI el valor del joystick x es mayor de 600 y el ángulo menor de 180º:
    --> Sumamos uno al valor de la variable ángulo.
    --> Ajustamos la posición del servo.

SI el valor del joystick x es menor de 400 y el ángulo mayor de 0º:
    --> Restamos uno al valor de la variable ángulo.
    --> Ajustamos la posición del servo.
```

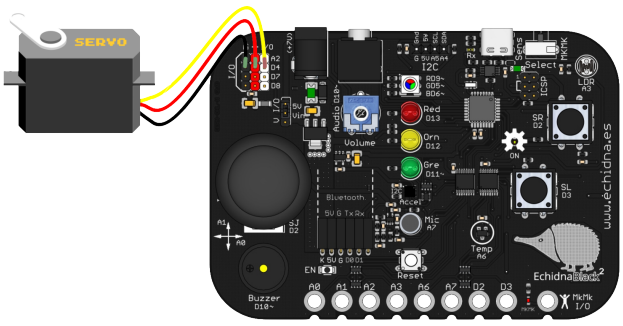
Introducimos un tiempo de espera de 0,1 s que evita que el servo esté continuamente reposicionándose, lo cual podría llevar a un funcionamiento inestable.

### 4.2.2 Servomotor de rotación continua: Control de sentido de giro con pulsadores

**COMPONENTE:**


Son **motores** de corriente continua con una reductora y electrónica de control que permiten **controlar** el **sentido** de **giro** del motor. Estos motores además permiten **ajustar** una pequeña variación en su **velocidad**.

Para conectarlo usamos los **pines I/O** de **entrada-salida** (D4, D7, D8, A2).




Presta atención al conectar los cables: Vcc, GND y señal, que están indicados por los colores rojo, negro y amarillo.

En caso de que vayas a conectar varios servomotores usa alimentación externa y coloca el selector de alimentación en la posición Vin. Ver apartado 2.4.

**BLOQUE DE PROGRAMACIÓN:**

Para **controlar** el **servomotor** de **rotación continua** podemos usar el siguiente **bloque**:


En el **bloque** podemos **seleccionar** el **pin** al que conectamos nuestro servo, el **sentido de giro** y la **velocidad**.

**Pines**: podemos seleccionar los pines; D4, D7, D8 y A2.

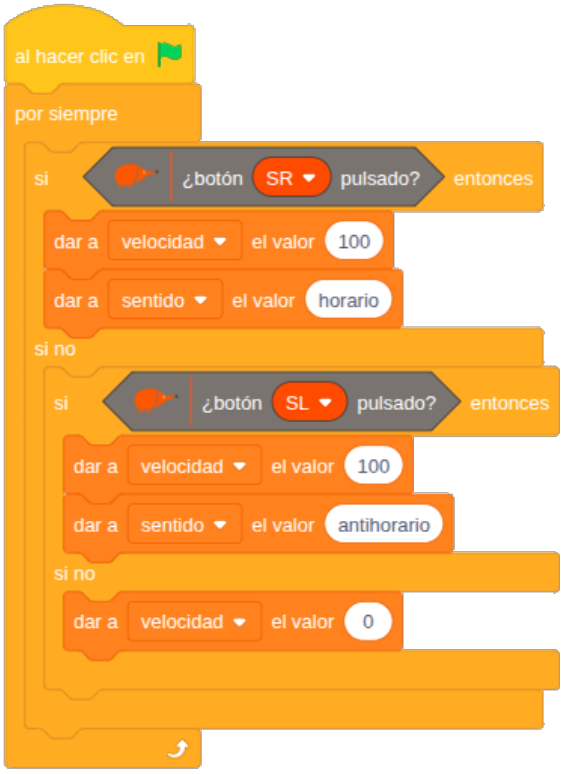
**Sentido de giro**: horario/ antihorario

**Velocidad**: 0-100%

### EJEMPLO: Control de sentido de giro de servomotor continuo con pulsadores

Este **ejemplo** ilustra cómo **controlar** el **sentido de giro** de un servomotor continuo (360 grados) utilizando **dos pulsadores** como entradas.

Si pulsamos el **botón SR** el motor gira en sentido **horario**. Si pulsamos el botón **SL** el motor gira en sentido **antihorario**. Si **no presionamos** ninguno de los dos pulsadores, el motor se **para**.





**Hilo de control:**

El programa evalúa el estado de los pulsadores para determinar la dirección del giro o la detención del servomotor.

Creamos las **variables**:

- **velocidad**: para controlar encendido (100)/ apagado (0) del motor.

- **sentido**: para controlar el sentido de giro, horario/antihorario, del motor.

**Giro Horario:**

```
SI el pulsador SR (Derecho) está presionado:
    --> La variable sentido = horario (indica que el giro será en sentido horario).
    --> La variable velocidad = 100.
```

**Giro Antihorario:**

```
SI NO:
    SI el pulsador SL (Izquierdo) está presionado:
        --> La variable sentido = antihorario (indica que el giro será en sentido
anti-horario).
        --> La variable velocidad = 100.
```

**Detención:**

```
SI NO (es decir, si ninguno de los dos pulsadores está presionado):
    --> La variable velocidad = 0.
```

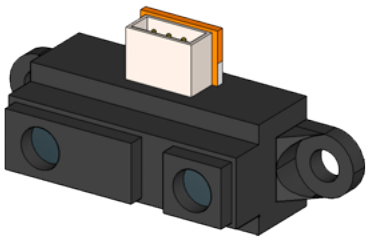
**Hilo de actuación:**

EL servomotor continuo gira en el sentido que le indica la variable sentido y a la velocidad indicada por la variable velocidad.

Introducimos un tiempo de espera de 0,1 s que evita que el servomotor esté continuamente reposicionándose, lo cual podría llevar a un funcionamiento inestable.

### 4.2.3 Sensor de distancia de infrarrojos: Sistema de asistencia al estacionamiento

**COMPONENTE:**

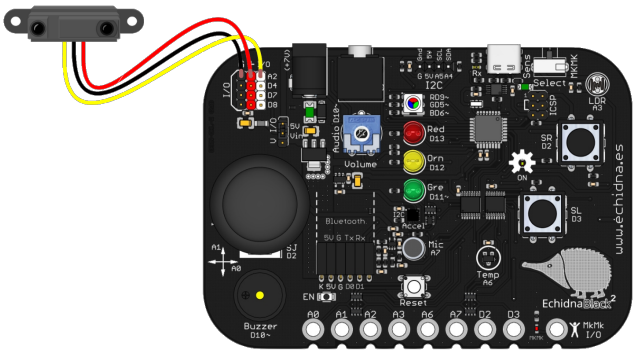


Es un **sensor** de **distancia** que proporciona una tensión según la cantidad de infrarrojo que rebota en una superficie.

**Salida Inversamente Proporcional y No Lineal:** el **valor** de salida es **inversamente proporcional** a la **distancia** al objeto. Es decir, el valor aumenta cuanto más se acerca el objeto. Además, la relación entre el valor de salida y la distancia no es lineal, por lo que requiere calibración o el uso de tablas para obtener una distancia precisa.

**Rango de medida:** El rango efectivo de medición varía significativamente según el modelo específico del sensor utilizado.

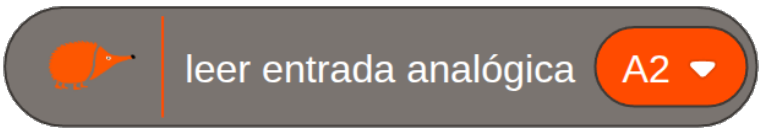
**Susceptibilidad a la luz:** La precisión del sensor puede verse afectada por la presencia de luz ambiente intensa o fuentes de infrarrojos externas.




**Conexión:** Se conecta directamente a **5V, GND y A2.**

## BLOQUE DE PROGRAMACIÓN:

Utilizaremos el bloque genérico, leer entrada analógica, seleccionando la entrada A2:




**Valores**: el rango de valores de la entrada analógica es 0-1023.

## EJEMPLO: Sistema de asistencia al estacionamiento

Este **ejemplo** simula un **sistema** de **ayuda** al **estacionamiento**.

El **sensor de distancia IR** conectado en el pin de entrada/salida (I/O A2) **mide** continuamente la **distancia** a un objeto. La funcionalidad del programa es modular la **frecuencia** del **zumbador** en función de esta distancia.

### 1º Medir valores de distancia:

El primer paso es medir los valores de distancia y almacenarlos en la variable lecturaIR, que utilizaremos para monitorizar y mostrar el valor del sensor.


Para ello podemos usar este programa que lee continuamente el valor del sensor conectado al pin analógico A2 y se guarda en la variable lecturaIR. Cada medida se realiza con un intervalo de 0,2 segundos para estabilizar la lectura.

Podemos activar la casilla de verificación de la variable para ver el valor registrado.

**2º Selección de umbrales:**

El siguiente paso será seleccionar tres umbrales diferentes poniendo un obstáculo frente al sensor a las tres distancias que queramos definir. En este caso usamos estos tres valores:

- **Distancia Pequeña**: 200
- **Distancia Muy Pequeña**: 400
- **Distancia Mínima**: 600

**3º Programa**

Tenemos dos hilos de ejecución:


**Hilo 1 de adquisición de datos:**

Medimos el valor del sensor y lo almacenamos en la variable lecturaIR. Tal como hemos visto en el paso 1º Medir valores de distancia.

**Hilo 2 Lógica de programación:**

**Zona Segura:**

```
SI la distancia es menor de 200:
    --> El zumbador permanece apagado.
```

**Zona de Advertencia:**

```
SI NO:
    SI la variable lecturaIR es menor de 400 (200 ≤ distancia < 400):
        --> Los pitidos se encienden y apagan con pausas de 0,2 segundos.
```

**Zona de Peligro Alto:**

```
SI NO:
    SI la variable lecturaIR es menor de 600 (400 ≤ distancia < 600):
        --> El zumbador se enciende y apaga cada 0,1 segundos, produciendo un ritmo
más rápido.
```

**Zona Crítica** (Colisión Inminente):

```
SI NO (si ninguna de las condiciones anteriores se cumple, es decir, la variable
lecturaIR es mayor o igual a 600):
    --> El zumbador emite pitidos muy rápidos, encendiéndose y apagándose cada 0,05
segundos.
```

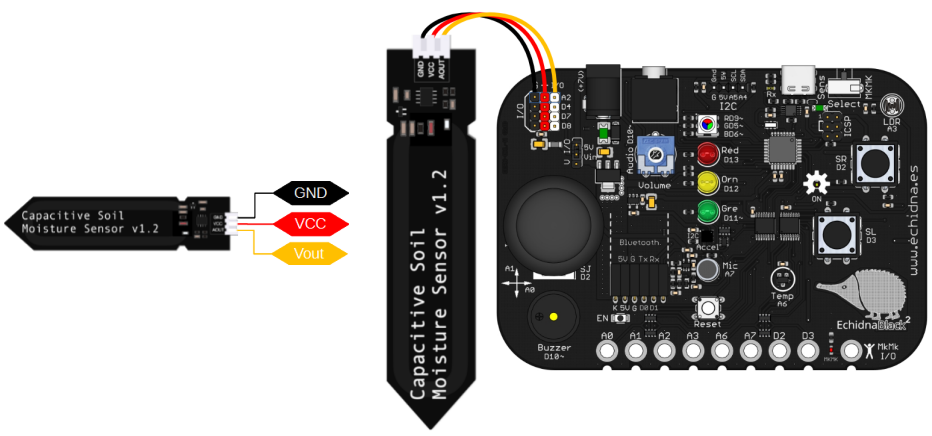
Todo este proceso se repite mediante un bucle por siempre, ajustando el sonido según lo cerca o lejos que esté el objeto.

## 4.2.4 Sensor de humedad del suelo: Monitorización de riego

### COMPONENTE:

El **sensor** de **humedad** del suelo permite **medir** la **cantidad** de **agua** en la **tierra**.

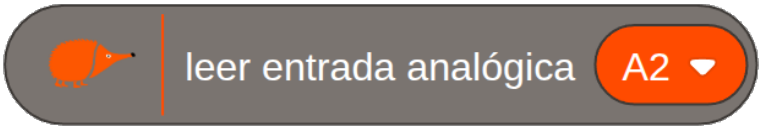
Tiene dos componentes principales: los electrodos, que se introducen en la tierra, y un circuito electrónico que convierte la humedad detectada en una señal eléctrica. Cuando el suelo contiene más agua, la conductividad entre los electrodos aumenta y, por tanto, el sensor devuelve un voltaje mayor; cuando el suelo está seco, la conductividad disminuye y el voltaje es menor.




**Conexión:** Se conecta directamente a 5V, GND y A2.

### BLOQUE DE PROGRAMACIÓN:

Utilizaremos el **bloque genérico**, **leer entrada analógica**, seleccionando la entrada **A2**.

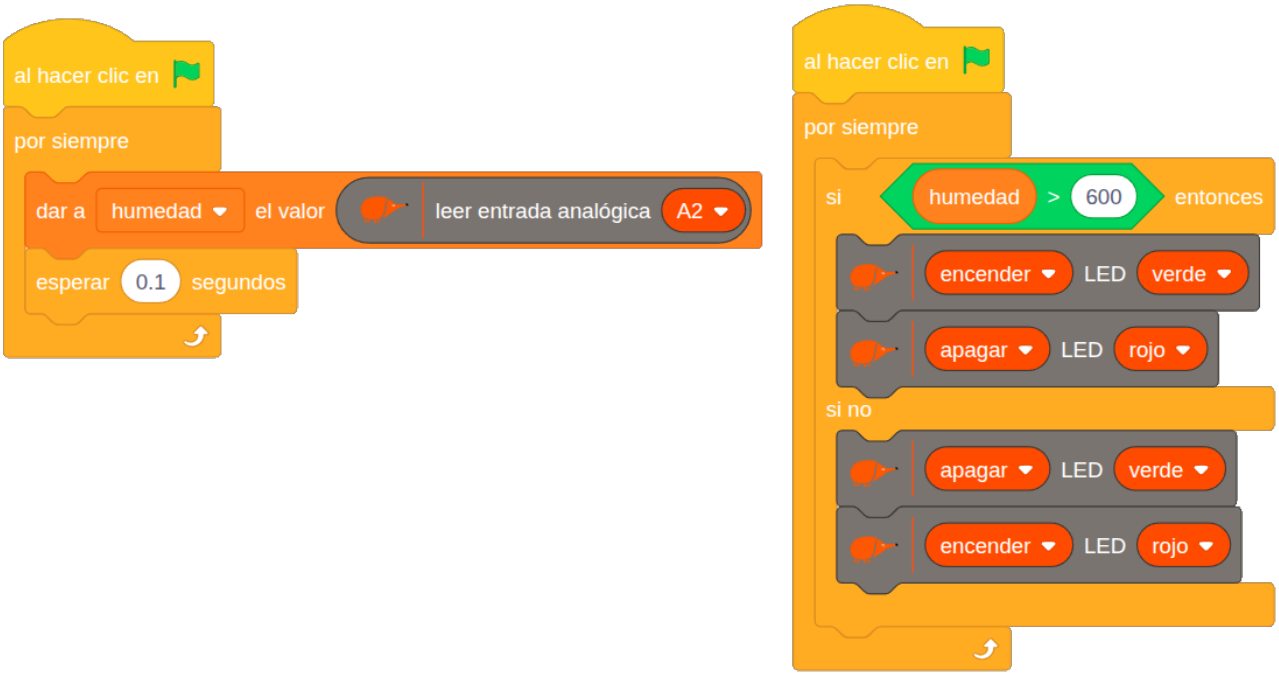



**Valores**: el rango de valores de la entrada analógica es 0-1023.

### EJEMPLO: Monitorización de riego

En el **ejemplo** vemos cómo hacer un sistema que nos **indique** cuando hay que **regar** una **planta**.

- Si la **humedad** es **adecuada** lo indicamos con el **LED verde**.
- Si la **humedad** es **baja** y la planta necesita ser regada lo indicamos con el **LED rojo**.




Lo primero es leer y almacenar los valores que proporciona el sensor de humedad en función del grado de humedad de la tierra.

En función de estos valores fijamos el umbral (en este caso 600) que determina que si la humedad es inferior se enciende el LED rojo y si es mayor se encienda el LED verde.

# 5. LEARNINGML

**EchidnaML** integra **LearningML**, una **plataforma educativa** diseñada para la **enseñanza** de los fundamentos del **machine learning** (aprendizaje automático). Esto nos permite incorporar inteligencia artificial a nuestros proyectos de robótica.

## 5.1 Introducción a LearningML

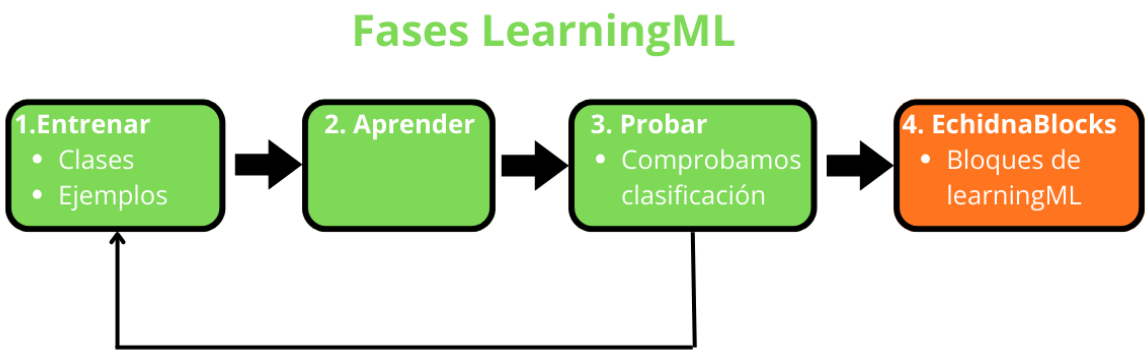
**LearningML** es una herramienta educativa diseñada para que el alumnado aprenda los conceptos básicos del **machine learning** de forma sencilla, visual y manipulativa. El alumnado puede crear modelos que clasifican **imágenes**, **textos** o **números** y después utilizarlos dentro de proyectos con bloques de Scratch.

Su principal **objetivo** es que el alumnado comprenda cómo se **entrenan** los **modelos**, qué **datos** necesitan, cómo influyen los **sesgos** y cómo se usan después para tomar **decisiones**. Todo ello sin programación compleja, favoreciendo el pensamiento computacional y el análisis crítico de la inteligencia artificial.

EchidnaML incluye esta herramienta, que puede usarse por sí misma o en combinación con los bloques de robótica con EchidnaBlocks.

#### Fases para crear un modelo LearningML

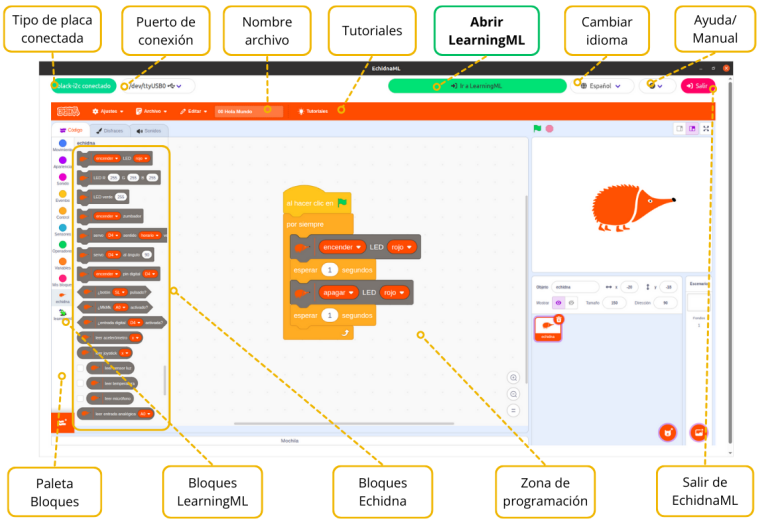
Para crear un modelo de machine learning tenemos que seguir los siguientes pasos.

1. 


   **Entrenar**: creamos las clases e introducimos los ejemplos.
2. **Aprender**: creamos el modelo.
3. **Probar**: comprobamos que el modelo clasifica correctamente.
    1. Si no clasifica correctamente volvemos a la fase **Entrenar**.
4. Abrimos **EchidnaBlocks** y usamos los bloques de LearningML para construir nuestra aplicación.

#### Acceder a LearningML

Para acceder a LearningML lo podemos hacer a través del icono situado en la pantalla de EchidnaBlocks.




**Entorno de LearningML**


Al acceder a la pantalla de LearningML podemos:

1. Ver si la placa está conectada y su modelo.
2. Ver puerto de conexión y reconectar si es necesario.
3. Abrir EchidnaBlocks.
4. Cambiar el idioma.
5. Acceder a la ayuda y el Manual.
6. Elegir el tipo de modelo que queremos crear: texto, imagen o números.
7. Elegir Nombre del archivo.
8. Salir del programa.

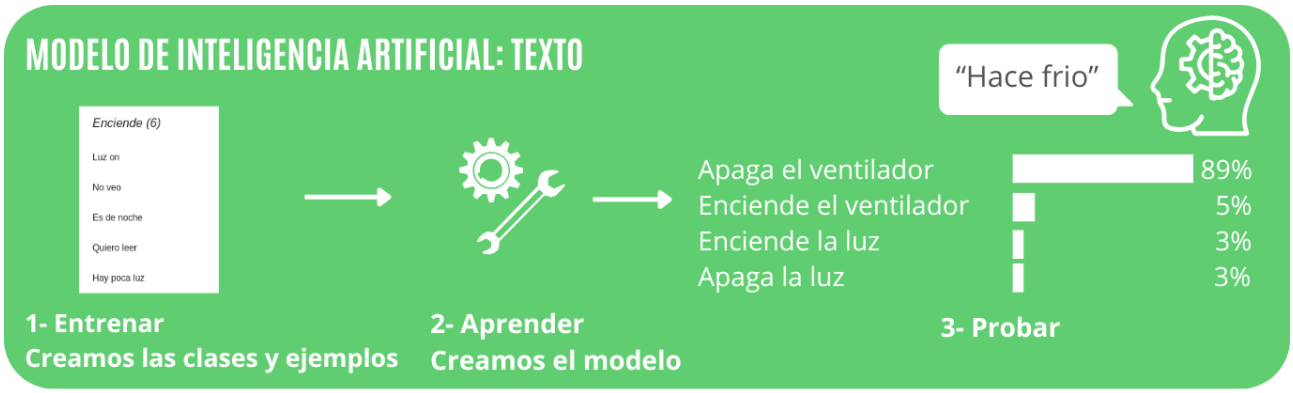
Una vez seleccionado el tipo de modelo textos, imágenes o números, iniciamos su construcción. A continuación, encontrará instrucciones para crear modelos de reconocimiento de **textos**, **imágenes** y **números**.

Para ampliar la información sobre su uso, puede consultar la web del proyecto web.learningml.org.

## 5.2 Modelo de texto

Vamos a ver los pasos para crear un **modelo** de **texto** en **LearningML** y como **usarlo** con **EchidnaBlocks**.

En este caso crearemos un **asistente virtual** que controla la iluminación de una vivienda y el ventilador.




### Abrir LearningML

Una vez hemos abierto EchidnaML abrimos la aplicación: Modelos de Machine Learning.

### Elegir tipo de datos

Lo primero que tenemos que hacer es **elegir** con qué **tipo** de **datos** vamos a trabajar: podemos hacerlo con textos, imágenes y números. En nuestro caso vamos a trabajar con datos de tipo **texto**.

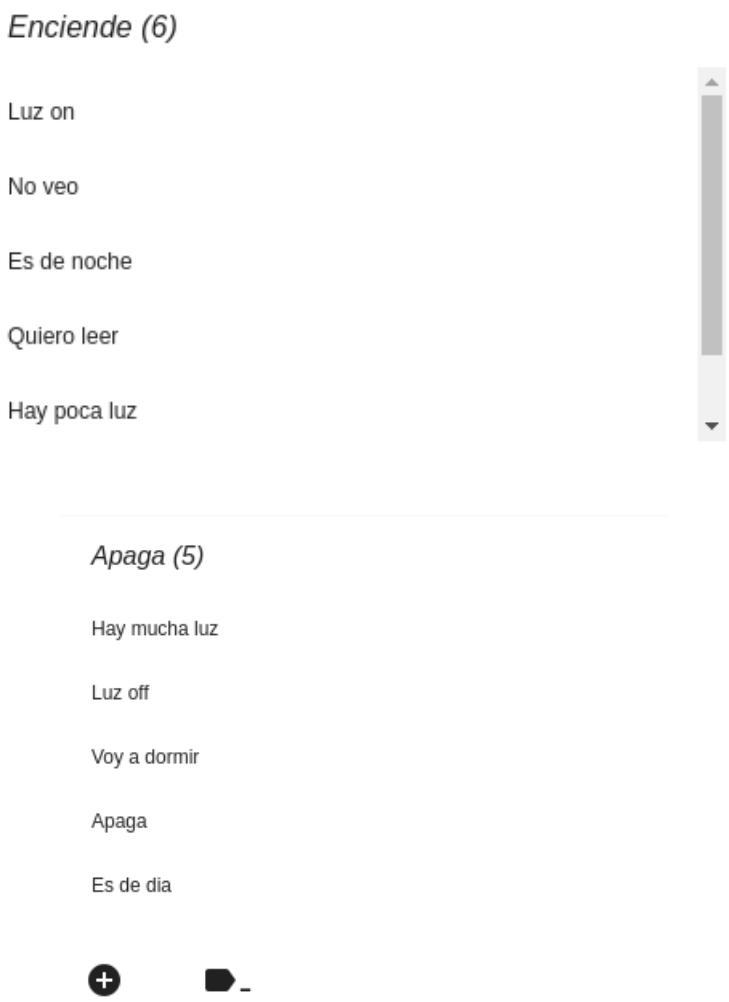

### 1- Entrenar: Crear las clases y añadir los ejemplos

Una vez elegido el tipo de datos (en este caso, texto), procedemos a la **creación** de las **clases** y a la **introducción** de los **datos** (ejemplos de texto) que el algoritmo usará para aprender a clasificarlos.

En nuestro caso, vamos a crear dos clases para controlar el LED: Enciende y Apaga.

**Clase "Enciende":** Introduciremos órdenes y frases que indiquen al sistema la intención de encender la luz (p. ej., "luz on", "enciende el LED", "enciende").

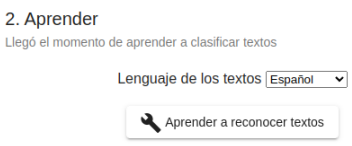
**Clase "Apaga":** Introduciremos órdenes que indiquen la intención de apagar la luz (p. ej., "apagar", "luz off", "corta la luz").




La **calidad** del **modelo** construido está directamente ligada a los datos de entrenamiento. Cuantos más textos, y mejor elegidos, se añadan a cada clase, mayor será la precisión del modelo al clasificar nuevos textos.

## 2- Aprender

Una vez que hemos definido las clases e introducido los ejemplos de entrenamiento, se procede a pulsar el botón **Aprender a reconocer textos**.



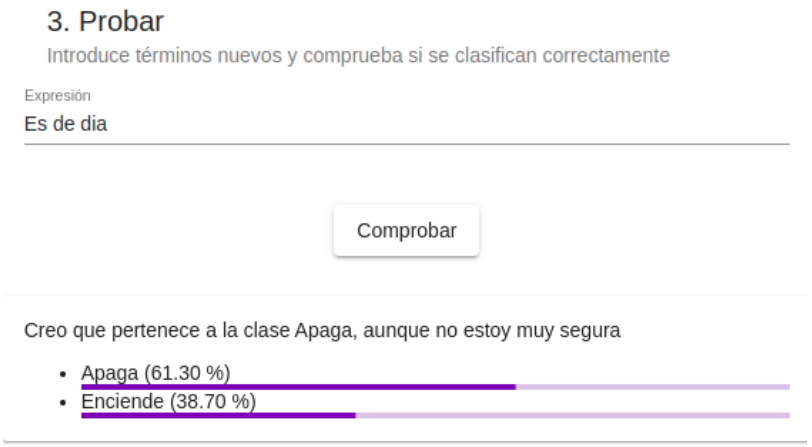
En ese momento, el algoritmo de Machine Learning analiza los datos proporcionados para construir un **modelo** de **inteligencia artificial**. Este modelo resultante es capaz de clasificar y reconocer nuevos textos que no ha visto antes.

Este proceso se denomina **aprendizaje a partir de datos** y constituye el fundamento esencial del Machine Learning.

## 3- Probar

Una vez construido el modelo, debemos realizar **pruebas** para **comprobar** si el modelo **clasifica correctamente** y con qué % de confianza lo hace.

Para esta fase de verificación, es crucial **utilizar textos** que sean **distintos** a los ejemplos que se introdujeron durante el proceso de entrenamiento. Esto garantiza que estamos probando la capacidad de generalización del modelo, y no simplemente su "memoria".




En este caso comprobamos que clasifica la frase “Es de día” con un 61% de probabilidad en la categoría Apaga, lo cual es correcto.

## 1- ¿Volver a entrenar?

Si los resultados de la prueba no son satisfactorios o si se desea incrementar la precisión del modelo, es necesario entrar en una fase de iteración y mejora.

**Revisión**: Examina y corrige los datos introducidos en los ejemplos.

**Ampliación**: Añade más ejemplos de entrenamiento.

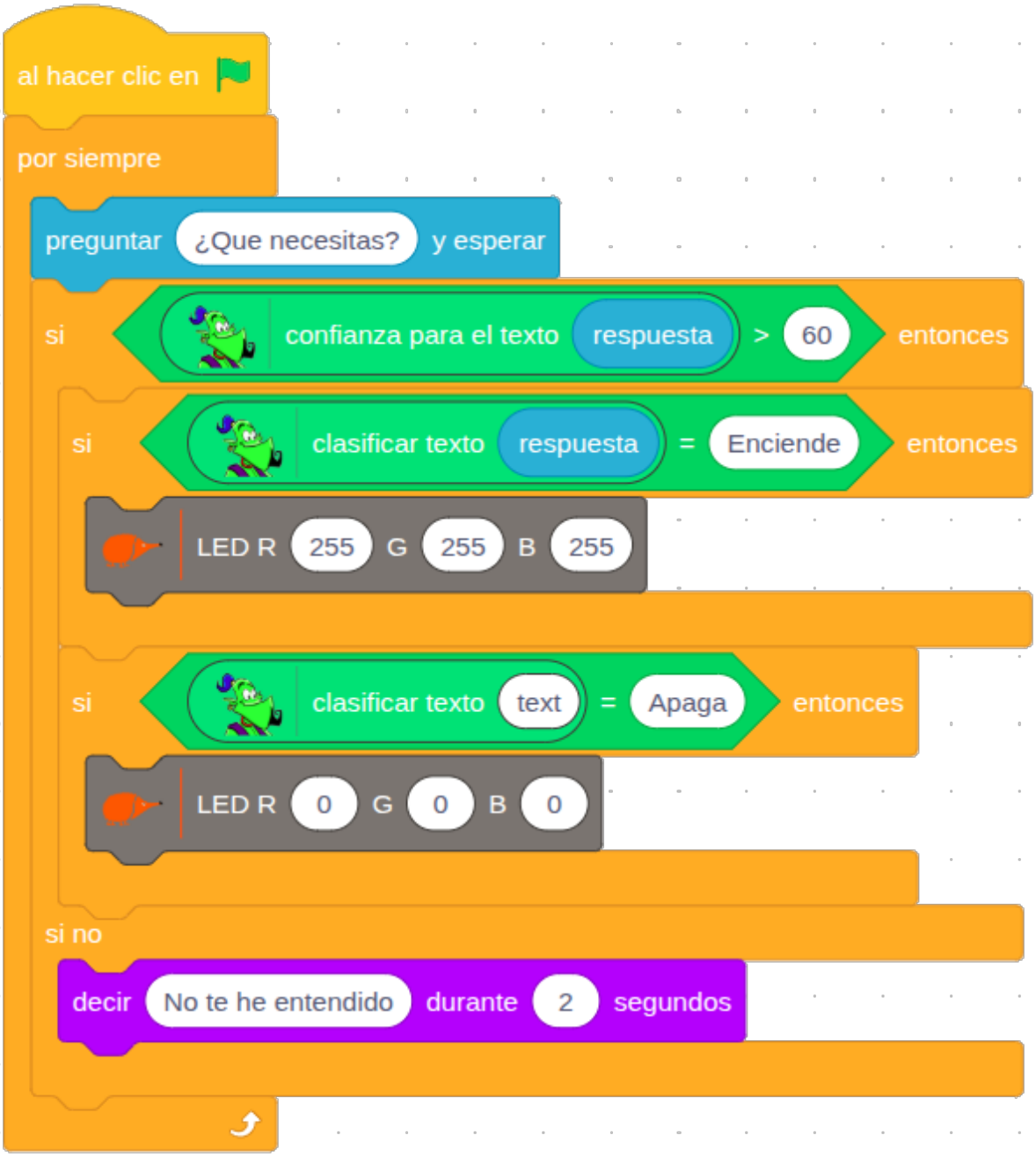
La **calidad** del **modelo** está directamente relacionada con la **cantidad** y la **calidad** de los **datos** de **entrenamiento**. Cuantos más datos de calidad se utilicen, siempre y cuando estén correctamente etiquetados, mayor será la precisión y la calidad de las clasificaciones futuras del modelo

## 4- Programamos en EchidnaBlocks

Una vez que las pruebas de entrenamiento hayan sido satisfactorias, abrimos **EchidnaBlocks**.

A continuación, la aplicación se programará combinando los bloques de Scratch, con los bloques de robótica y los de Machine Learning (LearningML), utilizando el modelo que acabamos de generar.

Este **ejemplo** muestra cómo **usar** el **modelo** de Machine Learning que hemos creado para **clasificar** una **entrada** de **texto** y, con el resultado, tomar una **decisión** en la placa robótica: **encender** o **apagar** el **LED RGB**.




**Consulta**: El personaje pregunta qué necesitamos y envía la respuesta que introduce el usuario al modelo de machine learning para su clasificación.

**Verificación de Confianza:** El sistema comprueba el índice de confianza que el modelo asigna a su clasificación. La acción sólo se ejecuta si este índice es superior al 60%. En caso de que la Confianza sea menor del 60% el programa dice “No te he entendido”.

**Clasificación**: Si la confianza es suficiente, el sistema determina la clase:

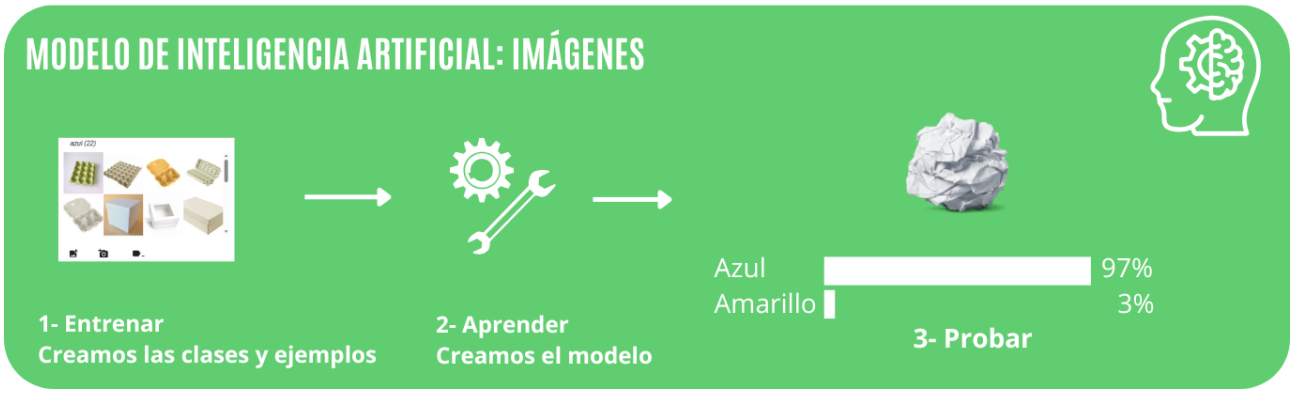
```
SI la clase es "Enciende":
    --> El programa envía la instrucción para encender el LED.

SI la clase es "Apaga":
    --> El programa envía la instrucción para apagar el LED.
```

Al finalizar, el personaje vuelve a decir: “¿Qué necesitas?”, indicando que está listo para una nueva clasificación.

## 5.3 Modelo de imágenes

En este segundo ejemplo con **LearningML** veremos los pasos para crear un **modelo** de **imágenes** y cómo usarlo con **EchidnaBlocks**.

Vamos a crear un modelo que nos clasifique los envases y papeles en las categorías amarillo y azul. En EchidnaBlocks vamos a mostrar la clasificación mediante un LED RGB y vamos a abrir el contenedor con un servomotor.




### Abrir LearningML

Una vez hemos abierto EchidnaML abrimos la aplicación: Modelos de Machine Learning.

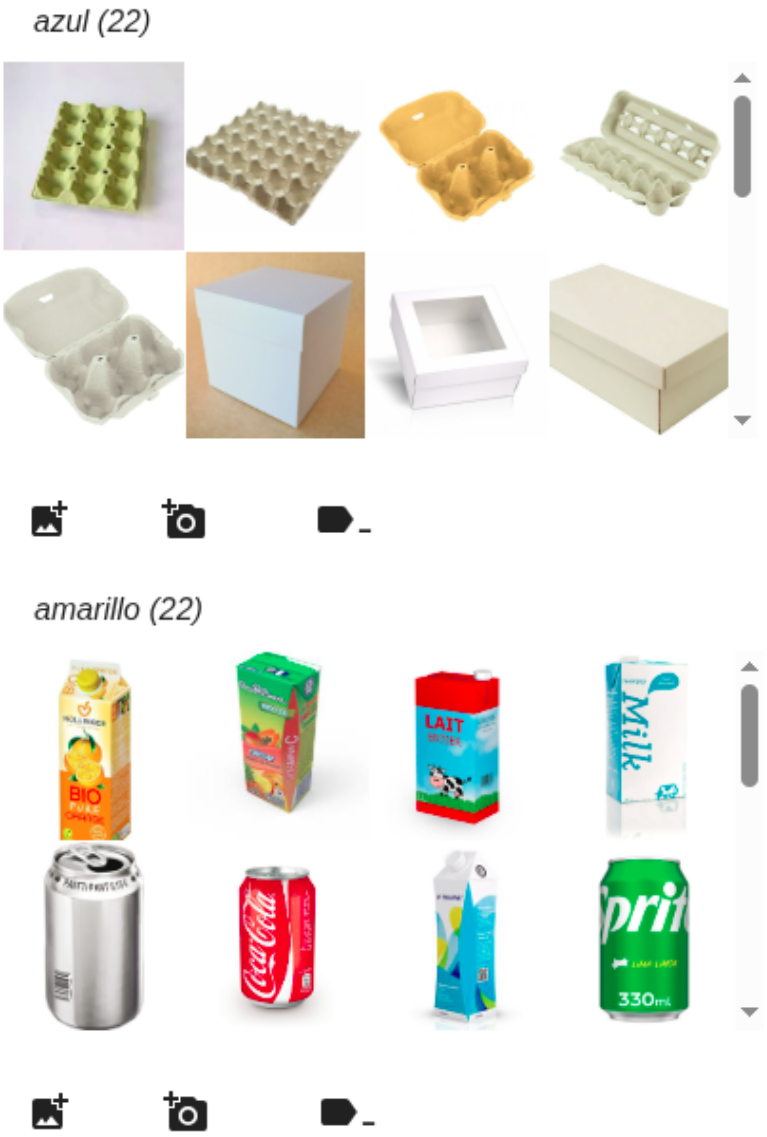
### Elegir tipo de datos

Es el momento de elegir con qué tipo de datos vamos a trabajar, en este caso trabajaremos con datos de imágenes.

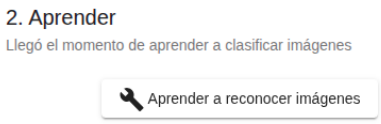

### 1- Entrenar: Crear las clases y añadir los ejemplos

Una vez hemos elegido el tipo de datos creamos las clases y le proporcionamos datos para que el algoritmo aprenda a reconocerlas.

Creamos dos clases: **azul** y **amarillo**, que nos permitirá reconocer residuos que se reciclan en el contenedor amarillo y residuos que se reciclan en el contenedor azul, para controlar dos servomotores que abran el contenedor adecuado.




## 2- Aprender



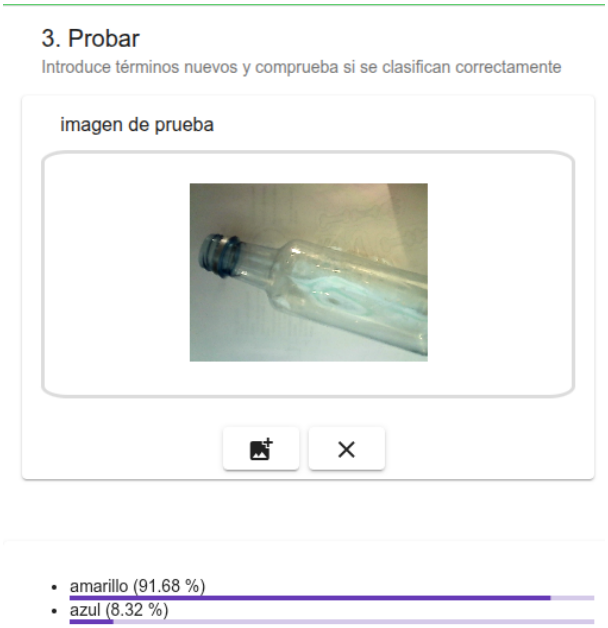

Pulsamos el botón **2. Aprender a reconocer imágenes**, para que el algoritmo de machine learning analice las imágenes y construya un modelo de inteligencia artificial capaz de reconocer imágenes similares pero distintas a las que hemos usado durante el entrenamiento.

La construcción del modelo, que se denomina aprendizaje, puede tardar varios segundos. Si las imágenes de ejemplo de la fase de entrenamiento son muchas, este proceso puede tardar bastante. Esto es una característica de los algoritmos de machine learning, ya que requieren mucho tiempo de proceso y bastante memoria.

## 3- Probar

Cuando el algoritmo finaliza ya tenemos disponible el modelo de Machine Learning. Es el momento de probar su funcionamiento.

Añadimos imágenes nuevas, no usadas en el entrenamiento y hacemos pruebas para comprobar si el modelo clasifica correctamente y con qué porcentaje de confianza lo hace.




En el ejemplo que mostramos, comprobamos que clasifica la imagen con más de un 91% de probabilidad en la categoría amarillo, lo cual refleja un alto grado de confianza en la respuesta dada.

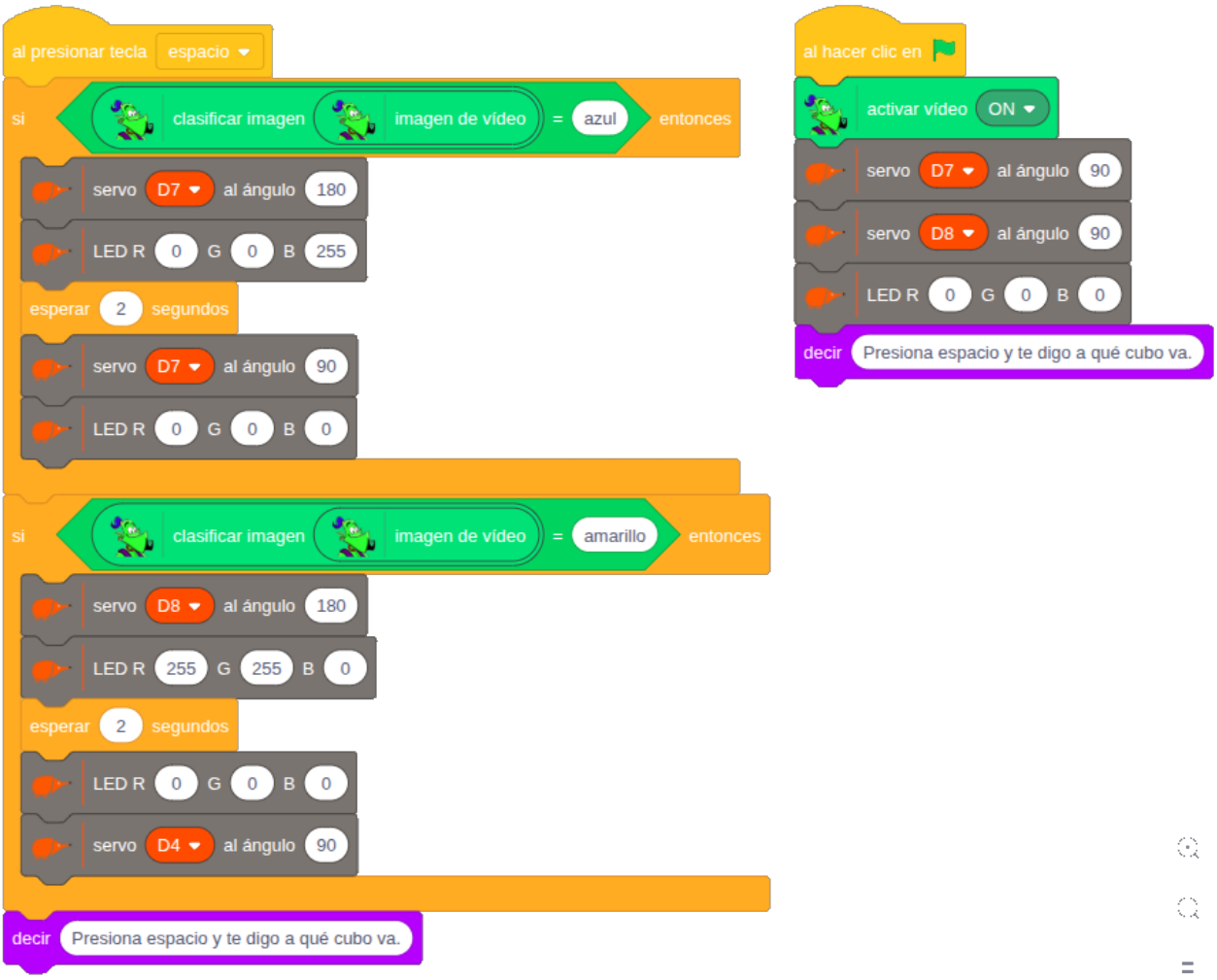
## 1- ¿Volver a entrenar?

Si el modelo no clasifica como queremos o deseamos mejorarlo hay que añadir y revisar los datos de la fase de entrenamiento.

## 4- Programamos en EchidnaBlocks

Una vez que las pruebas de entrenamiento hayan sido satisfactorias, podemos acceder a **EchidnaBlocks**. Desde donde ya podremos utilizar los bloques de machine learning con el modelo que acabamos de generar y programar nuestra aplicación robótica.

Este ejemplo implementa un sistema automatizado que utiliza un modelo de machine learning para **clasificar** una **imagen** capturada por la **cámara** y, en función del resultado, **activa** los **actuadores** del robot (servos y LED RGB).




**Configuración Inicial:**

Al inicio del programa, el sistema se prepara para recibir la clasificación:

- Se activa la cámara para la captura de vídeo.
- Los servos D7 y D8 se colocan en la posición de 90° (posición de cerrado o reposo).
- El LED RGB se asegura de estar apagado.

**El personaje indica**:

"Presiona espacio y te digo a qué cubo va", señalando que el sistema está listo para clasificar.

**Lógica de Clasificación:**

Al presionar la tecla Espacio, el programa clasifica el fotograma de vídeo capturado y ejecuta la acción robótica correspondiente:

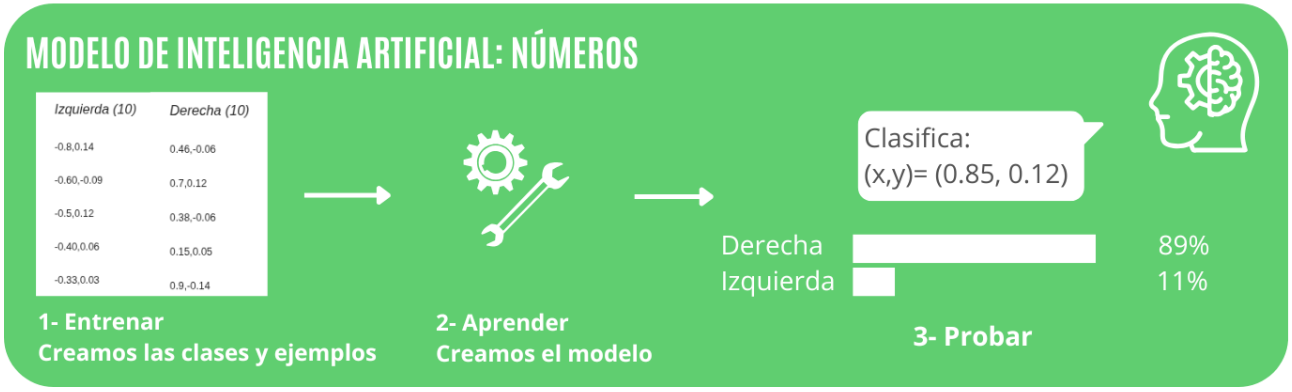
```
SI clasifica en clase "azul":
    --> El servo D7 se mueve a 0° (abre el cubo azul).
    --> El LED se ilumina en color azul (RGB 0, 0, 255).

SI clasifica en clase "amarilla":
    --> El servo D8 se mueve a 0° (abre el cubo amarillo).
    --> El LED se ilumina en color amarillo (RGB 255, 255, 0).
```

Al finalizar, el personaje vuelve a decir: “Presiona espacio y te digo a qué cubo va”, indicando que está listo para una nueva clasificación.

## 5.4 Modelo de números

En este tercer ejemplo con **LearningML** veremos los pasos para crear un **modelo** de **números** y cómo usarlo con **EchidnaBlocks**.

En este caso vamos a crear un modelo que nos **clasifique** la **inclinación** de la **placa** detectando: **derecha** e **izquierda**. En EchidnaBlocks vamos a desplazar el personaje en la dirección que nos clasifique.




### Abrir LearningML

Una vez hemos abierto EchidnaML abrimos la aplicación: Modelos de Machine Learning.

### Elegir tipo de datos

Es el momento de elegir con qué **tipo** de **datos** vamos a trabajar, en este caso con datos de tipo **números**.


### 1- Entrenar: Crear las clases y añadir los ejemplos

Una vez hemos elegido el tipo de **datos números**, seleccionamos el número de columnas del dato a introducir. En nuestro caso elegimos números con 2 columnas, ya que vamos a introducir datos de los valores del acelerómetro en coordenadas x e y.

Creamos las clases y le proporcionamos datos para que el algoritmo aprenda a reconocerlas.

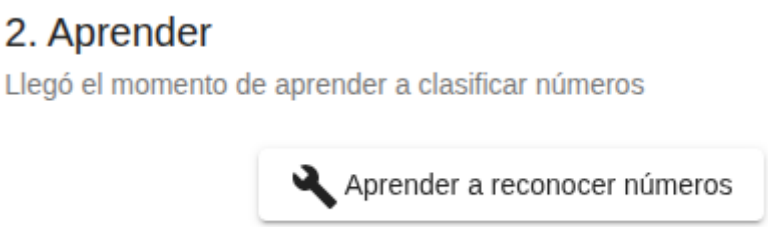
En este caso vamos a crear dos **clases**:

- **Derecha**
- **Izquierda**

Los números se deben introducir separándolos con una coma. Como hemos elegido un número de columnas igual a 2, debemos introducir un par de números separados por una coma.


## 2- Aprender

Al hacer clic en el botón **Aprender a reconocer números**, como ya hemos visto en los otros tipos, el algoritmo de machine learning aprende a reconocer números a partir de nuestros datos.

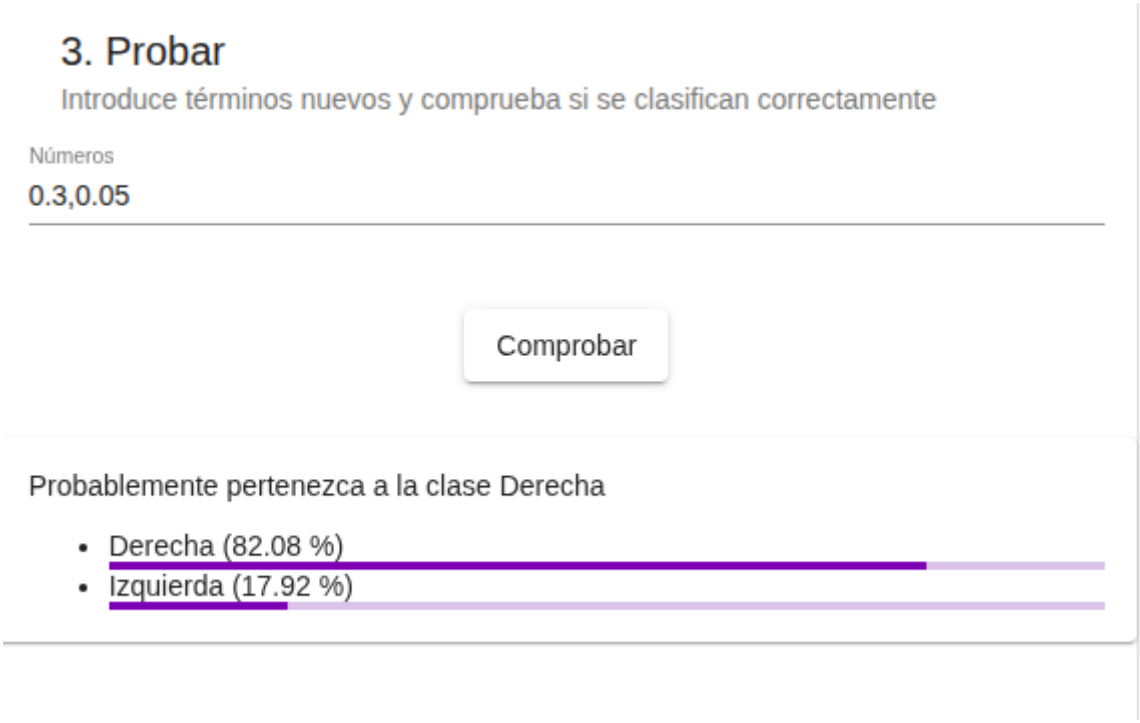



## 3- Probar

Es el momento de **probar** que el **modelo** que hemos creado **clasifica correctamente**. Para lo cual introducimos en la caja de texto de la fase 3-Probar, nuevos números separados por coma, similares pero distintos a los de la fase de entrenamiento.

El modelo arrojará su predicción y comprobaremos si clasifica correctamente y con qué porcentaje de confianza lo hace.

En este caso comprobamos que clasifica números de diferentes sectores con más de un 80% de probabilidad en la categoría que le corresponde.



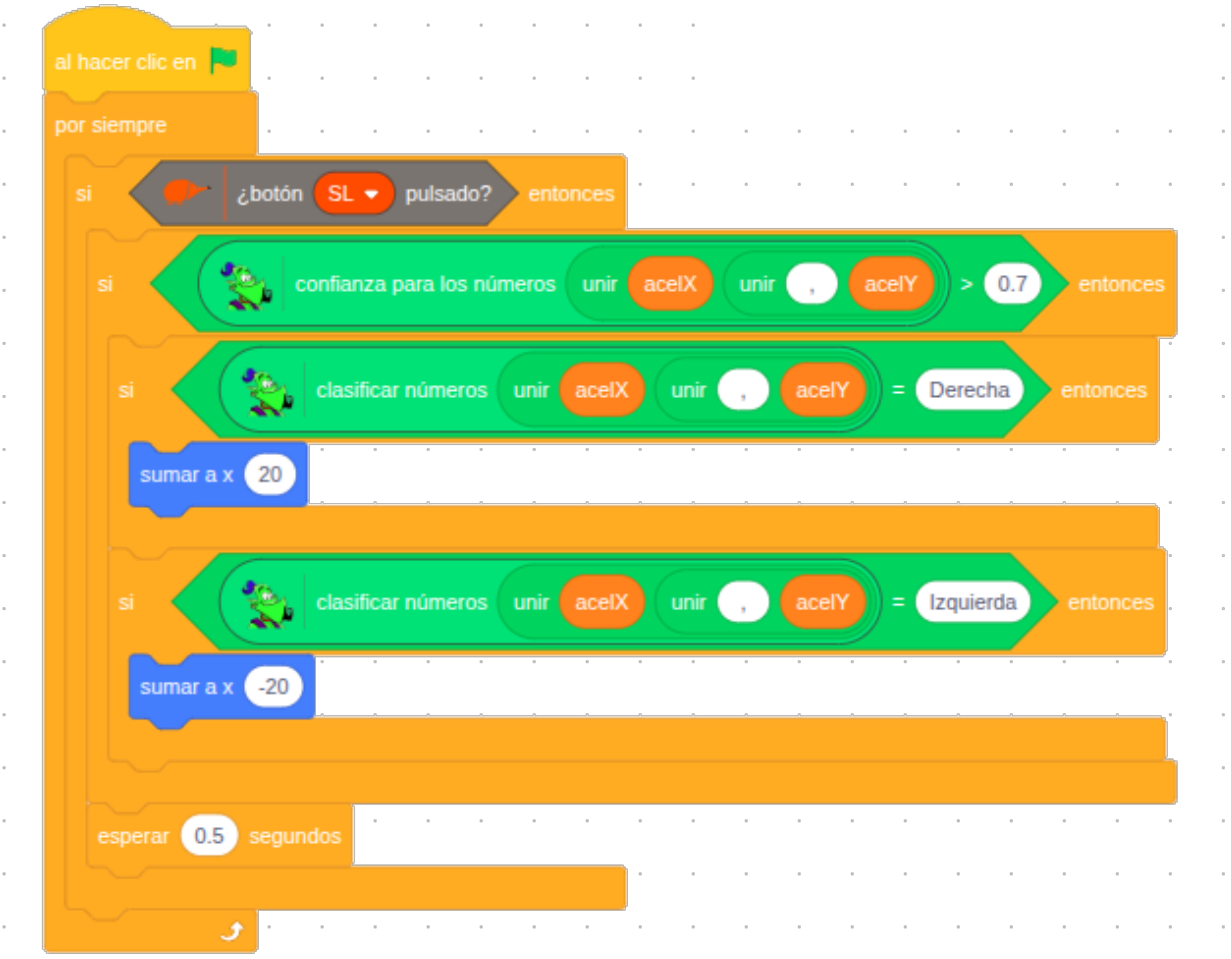

## 1- ¿Volver a entrenar?

Si no clasifica como queremos, tendremos que añadir y revisar los datos de la fase de entrenamiento.

## 4- Programamos en EchidnaBlocks

Una vez que las pruebas del modelo hayan sido satisfactorias, podemos acceder a **EchidnaBlocks**. Desde allí, ya podremos utilizar los **bloques** de **learningml** con el modelo que acabamos de generar y programar nuestra aplicación robótica.

En este ejemplo, cuando **presionamos** el botón **SL**, el programa **clasifica** la **inclinación** de la **placa** a partir de los **valores** del **acelerómetro**.

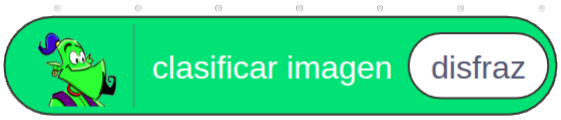
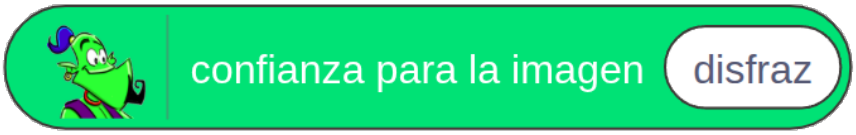


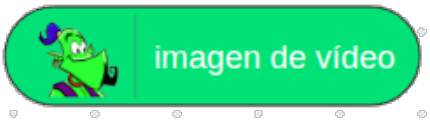
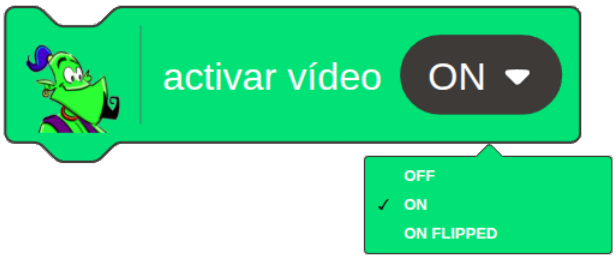
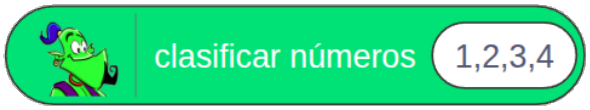
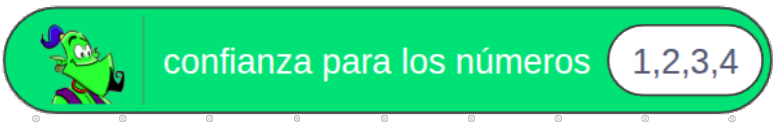

**Lógica de programación:**

```
SI la clasifica como "derecha":
    --> El personaje se desplaza a la derecha.

SI la clasifica como "izquierda":
    --> El personaje se desplaza a la izquierda.
```

## 5.5 Bloques LearningML en EchidnaBlocks

**EchidnaBlocks** cuenta con los siguientes **bloques** para usar los modelos de machine learning creados en **LearningML**.

| | |
|---|---|
|  | **Clasificar texto:** devuelve la etiqueta clasificada como más probable del texto que se introduzca como argumento. |
|  | **Confianza para el texto:** devuelve la probabilidad en porcentaje (0-100) de la clasificación propuesta por el modelo. |
|  | **Clasificar imagen:** devuelve el valor de la clasificación dada por el modelo de Machine Learning a la imagen que se aporta como primer argumento. Dicho argumento puede ser:<br>• El nº de disfraz cuya imagen se quiere clasificar<br>• El disfraz actual dado por el reporter “disfraz actual”<br>• La imagen tomada por la webcam y representada por el reporter |
|  | **Confianza para la imagen:** devuelve la probabilidad asignada por el modelo a la clasificación más probable (es decir a la que devuelve el reporter anterior) de la imagen que se aporta en su argumento, que igual que antes puede ser:<br>• El nº de disfraz cuya imagen se quiere clasificar<br>• El disfraz actual dado por el reporter “disfraz actual”<br>• La imagen tomada por la webcam y representada por el reporter |
| | **Imagen de vídeo:** devuelve la imagen tomada por la webcam. |

| | |
|---|---|
|  | |
|  | **Activar vídeo:** Un comando con el que se puede activar la webcam, activar la webcam en modo invertido o desactivar la webcam. |
|  | **Disfraz actual:** devuelve el disfraz actual activo. |
|  | **Clasificar números:** es un bloque que devuelve la etiqueta clasificada como más probable del conjunto de números que se introduzca como argumento. |
|  | **Confianza para los números:** devuelve la probabilidad en % (0-100) de la clasificación propuesta por el modelo. |

## 5.6 Exportar e importar datos de entrenamiento

La versión de **LearningML** en EchidnaML es un programa de **escritorio**, por lo que si quieres **guardar** los **datos** de **entrenamiento** de los modelos debes **descargarlos** y **almacenarlos** en tu **ordenador**. Para ello debes pulsar en Archivo → Guardar en tu ordenador y almacenarlo en la carpeta que elijas.


Para recuperar el proyecto, pulsa en Archivo → Cargar desde tu ordenador y busca el archivo en la carpeta donde lo almacenaste.

Los datos de entrenamiento de los modelos se almacenan en **formato json** y son compatibles con los realizados en LearningML escritorio y online. Una vez cargado un archivo json en LearningML será necesario hacer clic en el botón Aprender para poder usar el modelo.

# 6. PROGRAMA INSTALADO EN EL MICROCONTROLADOR

En este apartado vamos a ver el **programa** que hay **instalado** en el **microcontrolador** de la **placa** y **cómo** podemos **instalarlo** en caso de que se haya sobreescrito por otro programa.

## 6.1 ¿Qué es Firmata?

Para trabajar con **EchidnaML** en la placa **EchidnaBlack** tenemos que tener **instalado** un **programa** denominado **Firmata** que permite la comunicación entre la placa y el ordenador.

**Firmata** es un **protocolo** que facilita la **comunicación** entre **microcontroladores** y **ordenadores** de forma sencilla. Permite que el programa ejecutado en EchidnaML interactúe con la placa en tiempo real a través del puerto serie. De este modo, se establece un flujo bidireccional: la placa reporta constantemente las lecturas de sus sensores y el programa, tras procesarlas, envía las instrucciones necesarias para controlar los actuadores.


En EchidnaBlack trabajamos con **StandardFirmata**, que viene **instalado** de **serie** por lo que lo normal es que no necesites hacer nada. Si necesitas volver a instalar el programa StandardFirmata, en el siguiente apartado te explicamos cómo hacerlo.

## 6.2 Cómo instalar StandardFirmata

Vamos a ver cómo instalar StandardFirmata usando el **IDE de Arduino**, para lo cual debes seguir los siguientes pasos:

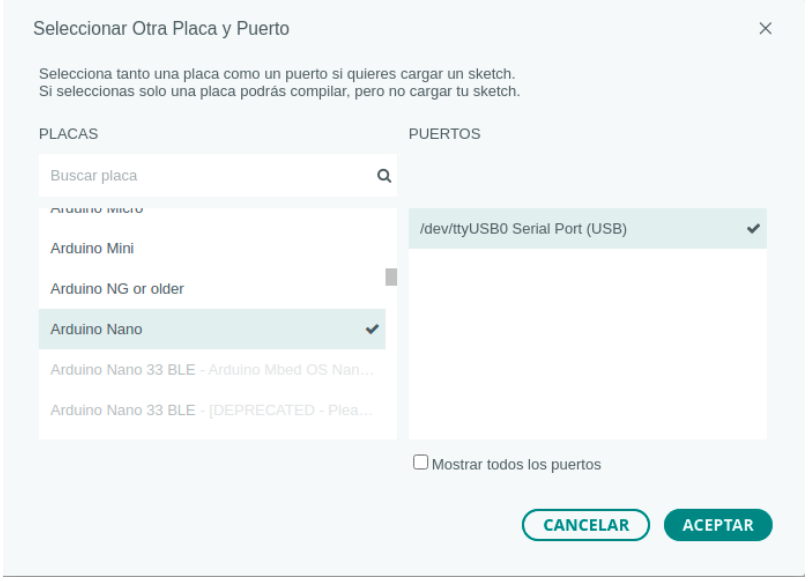
**1- Instalar IDE Arduino:**

El primer paso será tener instalado en nuestro ordenador el IDE de Arduino. Está disponible para Linux, MacOs y Windows y te lo puedes descargar desde la propia página de Arduino, donde también tienes una guía para instalarlo.

**2- Conectamos la placa EchidnaBlack** a nuestro ordenador a través del puerto USB.

**3- Abrimos el IDE de Arduino**.

**4- Seleccionamos la placa Arduino** que estemos usando y el **puerto USB** al que se conecta.




**Placa Arduino:** si tu placa es EchidnaBlack o EchidnaBlack2, debes escoger Arduino **Nano**.

En Herramientas → Placa → Arduino Nano.

**Puerto USB:** tenemos que seleccionar el puerto USB al que se conecta la placa.

Seguramente aparezca una indicación del puerto USB al que está conectado tu Echidna.

En función del sistema operativo que tengas te aparecerá un nombre para el puerto:

- GNU Linux: /dev/ttyUSB0
- Windows: COM21
- macOS: /dev/cu.usbserial-1410 (puerto USB)

En el que el número asignado puede variar en función de los dispositivos que tengamos conectados.

**5- Seleccionamos el programa que vamos a cargar, es decir el StandardFirmata.**

Lo hacemos desde el menú Archivo → Ejemplos → Firmata → StandardFirmata.

Atención si no hemos seleccionado la placa en el paso anterior no nos aparecen los programas Firmata.

**6- Cargamos el programa en la placa.**

Para ello clica en el botón “Subir”, que indica al IDE Arduino que cargue el programa en la placa.

Una vez cargado tu Echidna ya está lista para ser programada con EchidnaML. Aunque desconectes la placa y la guardes, el programa StandardFirmata seguirá instalado en la placa. Cuando la vuelvas a conectar a la computadora, se ejecutará dicho programa y será capaz de comunicarse con EchidnaML. Así que la carga del programa StandardFirmata solo la tendrás que hacer una vez.

## ¿Qué otros programas puedo usar?

Hemos explicado cómo usar el IDE de Arduino, pero también puedes usar otros programas como:

- PlatformIO
- Eclipse Arduino IDE
- Codebender
- ArduinoDroid
- Programino

# 7. CARACTERÍSTICAS TÉCNICAS DE LA PLACA

**Microcontrolador**: AtMega 328P, 16MHz. Compatible con Arduino Nano.

**Sensores incorporados:**

- Joystick X,Y lineal
- Sensor de luz LDR sensible 400 - 600LUX a 540nm
- Sensor de temperatura -40°C a +125°C 10mV/°C
- Acelerómetro X-Y-Z ±2g (I2C 0x18)
- Micrófono omnidireccional 100 - 20KHZ
- Pulsadores 12mm 1.27 - 2.55N
- Entradas MkMk Conductivas

**Actuadores incorporados:**

- LEDes 5mm Rojo, naranja y Verde
- LED RGB 65535 colores
- Rojo (619 - 624nm)
- Verde (520 - 540nm)
- Azul (460 - 480nm)
- Audio:
- Zumbador 2300Hz, 85dB a 10cm
- Conexión Jack (audio) 3.5mm
- Control de volumen con potenciómetro lineal

**Conectores incorporados:**

- USB-C (Programación/comunicación)
- Conector 6 pines para módulo Bluetooth HC-05: K, 5V, GND, RX, TX, NC
- Conector 4 pines I2C: GND, 5V, SCL, SDA
- Conector I/O:
- GND, + V, A2
- GND, + V, D4
- GND, + V, D7
- GND, + V, D8
- Conexión Jack de alimentación 2,5mm, 6,4mm: (+V), GND
- Conector 6 pines ICSP

**Consumo**: Conexión por USB 0,150W, Alimentación externa 0,160W.

# 8. INSTALACIÓN DE ECHIDNAML

**EchidnaML** permite trabajar **programación**, **robótica** e **inteligencia artificial** en un único entorno educativo. En este apartado encontrarás las versiones disponibles para GNU/Linux, Windows y macOS, junto con instrucciones paso a paso para instalar el programa en cada sistema operativo.

## 8.1 GNU/Linux

**EchidnaML** funciona en distribuciones **GNU/Linux** de **64 bits** basadas en **Debian**, como Ubuntu, Linux Mint o MAX.

**Descargas disponibles:**

- echidnaml_1.6.0_amd64.deb
- echidnaml_1.6.0-ubuntu-24.04_amd64.deb
- EchidnaML-1.6.0.AppImage

**¿Qué archivo debo descargar?**

- Si utilizas Ubuntu (20.04, 22.04), MAX o Linux Mint, usa `echidnaml_1.6.0_amd64.deb`.
- Si tu sistema operativo es Ubuntu 24.04, usa `echidnaml_1.6.0-ubuntu-24.04_amd64.deb`.
- Si utilizas otra distribución o prefieres una versión portátil, la mejor opción es `EchidnaML-1.6.0.AppImage`.

### Instrucciones de instalación

#### A- Instalación mediante paquete .deb

**Opción gráfica:**

1. Haz doble clic sobre el archivo descargado.
2. Pulsa «Instalar».
3. Introduce tu contraseña cuando se solicite.

**Opción mediante terminal de comandos:**

Abre una terminal de comandos y colócate en el directorio donde esté el archivo .deb. La instalación se hace así:

```
sudo apt install ./echidnaml_1.6.0_amd64.deb
```

#### B- Ejecución de la versión AppImage

1. Descarga el archivo.
2. Activa el permiso de ejecución (esto solo tienes que hacerlo la primera vez).
3. Haz doble clic para iniciar la aplicación.

Puedes activar el permiso de ejecución de dos maneras:

**Opción gráfica:**

- Haz clic derecho sobre el archivo y selecciona Propiedades.
- Dirígete a la pestaña Permisos y marca la casilla "Permitir ejecutar el archivo como un programa" (o similar, dependiendo de tu distribución).
- Cierra la ventana y haz doble clic sobre el icono para iniciar EchidnaML.

**Opción mediante terminal de comandos:**

Abre una terminal de comandos y colócate en el directorio donde esté el archivo AppImage. Ejecuta la siguiente instrucción:

```
sudo chmod +x EchidnaML-1.6.0.AppImage
```

## 8.2 Windows

**EchidnaML** está disponible para **Windows 10** y **Windows 11** de **64 bits**.

**Descarga:**

- EchidnaML 1.6.0.msi

### Instalación

1. Descarga el archivo de instalación.
2. Haz doble clic sobre el archivo .msi.
3. Sigue las instrucciones del asistente.

**Aviso de seguridad de Windows:**

Es posible que Windows muestre una ventana con el mensaje «Windows protegió su PC». Esto ocurre porque la aplicación no está firmada por una gran empresa de software.

Para continuar:

1. Haz clic en «Más información».
2. Pulsa «Ejecutar de todas formas».
3. Después podrás completar la instalación normalmente.

**Instalación del controlador CH341:**

En Windows es necesario instalar el driver del controlador CH341, que gestiona el puerto serie de la EchidnaBlack2.

## 8.3 macOS

**EchidnaML** está disponible para **macOS 14** o superior.

Actualmente se ofrece la versión para ordenadores con **Apple Silicon** (M1, M2, M3 y posteriores). La versión para procesadores Intel se publicará próximamente.

**Descarga:**

- EchidnaML-1.6.0-arm64.dmg

### Instalación

1. Descarga el archivo .dmg.
2. Haz doble clic para abrirlo.
3. Arrastra EchidnaML a la carpeta Aplicaciones.
4. Expulsa la imagen de disco cuando finalice la copia.

**Primer inicio:**

La primera vez que abras la aplicación, macOS puede indicar que el desarrollador no está verificado.

Para abrirla:

1. Ve a la carpeta Aplicaciones.
2. Haz clic derecho sobre EchidnaML.
3. Selecciona «Abrir».
4. Confirma pulsando nuevamente «Abrir».

Solo tendrás que hacerlo una vez.

# 9. PREGUNTAS FRECUENTES

En este apartado respondemos algunas preguntas que se dan con cierta frecuencia y que pueden resultar de ayuda.

## 9.1 Qué requisitos técnicos necesito

Para trabajar con EchidnaML se requiere un ordenador de escritorio o portátil con prestaciones básicas. En algunas distribuciones de software libre educativo viene instalada por defecto. Este software no es compatible con Chromebooks, dispositivos móviles ni tablets.

Se recomienda disponer, como mínimo, de un ordenador con las siguientes características: procesador x86_64 o ARM64 de 4 núcleos a 2GHz, 8 GB de memoria RAM y 5 GB de espacio libre en disco para la instalación y ejecución del software.

EchidnaML es compatible con los principales sistemas operativos de escritorio:

- Windows: Windows 10 (64 bits) y Windows 11 (64 bits)
- macOS (versiones 14 en adelante)
- Linux (distribuciones basadas en Debian, 64 bits)

Se recomienda mantener el sistema operativo actualizado y disponer de permisos de instalación para garantizar el correcto funcionamiento del programa y la comunicación con la placa EchidnaBlack.

## 9.2 El programa no detecta la placa

Si al abrir EchidnaML el programa no detecta la placa se puede deber a:

**1. EchidnaBlack no tiene instalado el Firmware StandardFirmata:**

Aunque el firmware viene instalado de serie, alguien puede haber escrito otro programa. Si es así debemos instalar el programa StandardFirmata tal como se especifica en el apartado 6 de esta guía.

**2. Nuestro ordenador no reconoce el puerto USB al que se conecta EchidnaML:**

Para que EchidnaBlack se pueda comunicar con nuestro PC es necesario que nuestro sistema operativo (SO) le dé permiso de acceso al puerto serie (USB) y que tenga el driver del controlador de comunicación (CH340) instalado.

**2.1 Driver de comunicación:**

EchidnaBlack utiliza el chip CH340, por lo que dependiendo del SO necesitarás instalar el controlador "Driver CH341"

- GNU Linux: En caso de que seas usuario Linux, no debería ser necesario instalar el driver. Si necesitas el driver para GNU Linux.
- macOS: Aquí tienes acceso al driver para Mac.
- Windows: Aquí tienes acceso al driver para Windows.

**2.2 Permiso de acceso al puerto serie:**

Dependiendo del SO es necesario dar o no permisos de acceso al puerto serie. Si ya has instalado el IDE de Arduino, estos permisos deberían estar ya dados.

**GNU Linux:** Si no tuvieras acceso al puerto serie puede que tengas que darle permiso desde una terminal usando el comando: sudo usermod -a -G dialout tu_usuario. Luego es necesario salir de la sesión y volver a entrar para que los cambios se hagan efectivos.

**macOS**: por defecto ya tenemos acceso al puerto serie.

**Windows**: por defecto ya tenemos acceso al puerto serie.

## 9.3 Puedo conectar la placa una vez abierto el programa

Lo más recomendable es conectar la placa EchidnaBlack2 al ordenador antes de abrir el entorno de programación EchidnaML. Esto asegura una detección rápida y que no perdamos el trabajo realizado.

Si ya has abierto el programa EchidnaML y quieres conectar la placa posteriormente:

1. Conecta la placa al ordenador mediante el cable USB.
2. Para iniciar la detección, haz clic en el icono USB (o el icono de conexión) dentro de la interfaz.

**Advertencia: Guardar el Trabajo**

Debes tener en cuenta que el proceso de detección y conexión reinicia el entorno de trabajo, provocando la pérdida de cualquier proyecto no guardado. Por esta razón, guarda siempre tu proyecto en el ordenador antes de iniciar el proceso de detección para evitar la pérdida de trabajo y poder recuperarlo.

## 9.4 El joystick registra valores mínimos muy altos y/o máximos muy bajos

En ocasiones el capuchón del joystick viene insertado muy profundamente en el eje, lo que provoca que choque contra la base y no haga todo el recorrido. Prueba a sacarlo un poco hacia arriba.

## 9.5 ¿Qué diferencias hay entre EchidnaBlack y EchidnaBlack2?

Si tienes una EchidnaBlack debes saber que casi todo lo que se cuenta en este manual es válido para tu placa.

El software EchidnaML detecta que versión de placa estamos usando y ajusta los bloques a las características de la misma.

La principal diferencia entre las placas es la incorporación del acelerómetro por I2C en la nueva placa que ha hecho que algunos pines se reajusten:

| | EchidnaBlack | EchidnaBlack2 |
|---|---|---|
| **Acelerómetro** | Conectado a A2, A3<br>Eje x, y | Conectado a A4, A5<br>Comunicación I2C<br>Eje x, y, z |
| **LDR** | Conectada a A5 | Conectada a A3 |
| **Pines MkMk** | A0, A1, A2, A3, A6, A7, D2, D3 | A0, A1, A2, A3, A6, A7, D2, D3 |
| **Pines I/O** | A4, D4, D7, D8 | A2, D4, D7, D8 |
| **Pines I2C** | No tiene | A4,A5 |

## 9.6 Qué tensiones soporta la placa

### Alimentación por USB-C (Recomendada para la mayoría de usos)

Esta es la forma más sencilla y común de usar la placa, mediante un cable USB-C conectado a un ordenador o a un cargador.

Características:

- La placa recibe una tensión de 5 V y suficiente energía para las funciones básicas.
- Tensión de uso: Toda la placa recibe una tensión regulada y estable de 5V.
- Límite de energía: Es suficiente para las funciones básicas y la mayoría de sensores pequeños (generalmente con un límite de 500 mA).
- Protección: Cuenta con un fusible rearmable que se desconecta si hay un consumo excesivo.

### Alimentación por Jack (Recomendada para proyectos grandes o autónomos)

La usamos cuando queremos conectar elementos con mucha potencia, como varios servomotores, o para proyectos autónomos.

Características:

- Tensión de uso: La placa recibe una tensión regulada y estable de 5V.
- Límite de energía: Es suficiente para las funciones básicas y la mayoría de sensores (generalmente con un límite de 1000 mA).
- Protección: Cuenta con un fusible rearmable que se desconecta si hay un consumo excesivo.

### Selector de Alimentación I/O

Este jumper te permite elegir qué tensión quieres enviar a los pines de entrada/salida de la placa (I/O, concretamente A2, D4, D7, D8) que usarán los componentes externos.

**Selector alimentación 5V:**

En el caso de querer alimentar las I/O desde los 5 Volts procedentes del regulador integrado, el selector tiene que estar colocado como indica la imagen de la izquierda. Aconsejado para sensores externos que necesiten una tensión estabilizada.

¡No utilices la alimentación 5 Volts cuando los dispositivos conectados consuman más de 500 mA! De lo contrario sobrepasaríamos la capacidad del regulador.

**Selector alimentación Vin (Alimentación externa):**

Aconsejado para alimentar servos u otros dispositivos conectados a I/O que necesiten una tensión mayor de 5 Volts.

¡ATENCIÓN!

Si tu alimentador externo tiene una tensión superior a la que soportan tus componentes conectados a I/O (que a menudo son 5 Volts), ¡puedes quemarlos inmediatamente!

# 10. REFERENCIAS

En este apartado encontrarás **enlaces** y **recursos adicionales** para quienes deseen **ampliar** la **información** sobre los temas tratados en esta Guía.

**Echidna**: https://echidna.es/

- EchidnaBlack2: características del Hardware.
- Descargar el programa EchidnaML
- Empieza a trabajar con EchidnaML y EchidnaBlocks: vídeos explicativos
- Manual en versión eXelearning y PDF
- Imprimir la carcasa.
- REA: Recursos educativos abiertos realizados con eXelearning.
- Actividades para su uso en Primaria y Secundaria

**LearningML**: https://web.learningml.org/

- Manual de learningML
- LearningML online

**Scratch**: https://scratch.mit.edu/

- Tutorial de introducción
- Explorar proyectos de iniciación

**Arduino**:

- Descargar IDE Arduino
- Instalar IDE de Arduino

# 11. LICENCIA

| | |
|---|---|
| Título | Manual EchidnaBlack2 y EchidnaML |
| Descripción | Manual sobre el uso de la placa EchidnaBlack2 y el software EchidnaML. |
| Autoría | Echidna Educación |
| Licencia | Creative Commons BY-SA 4.0 |