

Proyectos de inicio con EchidnaML

Primeros pasos con la placa EchidnaBlack2



EchidnaML

Echidna Educación



Índice

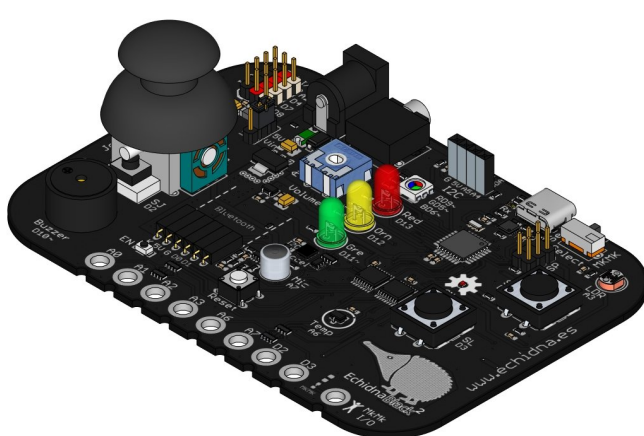
1. Introducción	3
2. Proyectos con EchidnaBlocks	4
2.1 Hola Mundo	7
2.2 Semáforo.....	10
2.3 Interruptor de luz	13
2.4 Timbre	16
2.5 Interruptor crepuscular	19
2.6 Piano de frutas	22
2.7 El echidna dice la temperatura	25
2.8 Termómetro de colores	28
2.9 Vúmetro	31
2.10 Telesketch	34
2.11 Movemos el echidna	38
3. Proyectos con IA.....	41
3.1 Asistente virtual	43
3.2 Clasificador de residuos.....	48
3.3 Mando de inclinación	53
4. Licencia	58



1. Introducción

EchidnaBlack2 (hardware) y **EchidnaML** (software) forman un **sistema integrado** pensado para aprender los fundamentos de la **programación**, la **robótica** y la **inteligencia artificial** (IA). Su objetivo es fomentar el **pensamiento computacional** en Primaria, Secundaria, Bachillerato y F.P., y estimular la creatividad con proyectos prácticos que conectan 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.



EchidnaML reúne dos herramientas que usaremos en estos proyectos:

- **EchidnaBlocks**: el entorno de programación por bloques, basado en Scratch, con el que programamos la placa.
- **LearningML**: la herramienta para crear modelos de **machine learning** que después usamos en nuestros programas.

Te proponemos una serie de **proyectos sencillos** para dar tus primeros pasos con la placa. No necesitas conocimientos previos de programación, aunque te resultará más fácil si ya conoces **Scratch**. Están organizados en dos bloques:

- **Proyectos con EchidnaBlocks**: cada proyecto presenta un componente de la placa y cómo programarlo.
- **Proyectos con IA**: añadimos inteligencia artificial con modelos de LearningML.

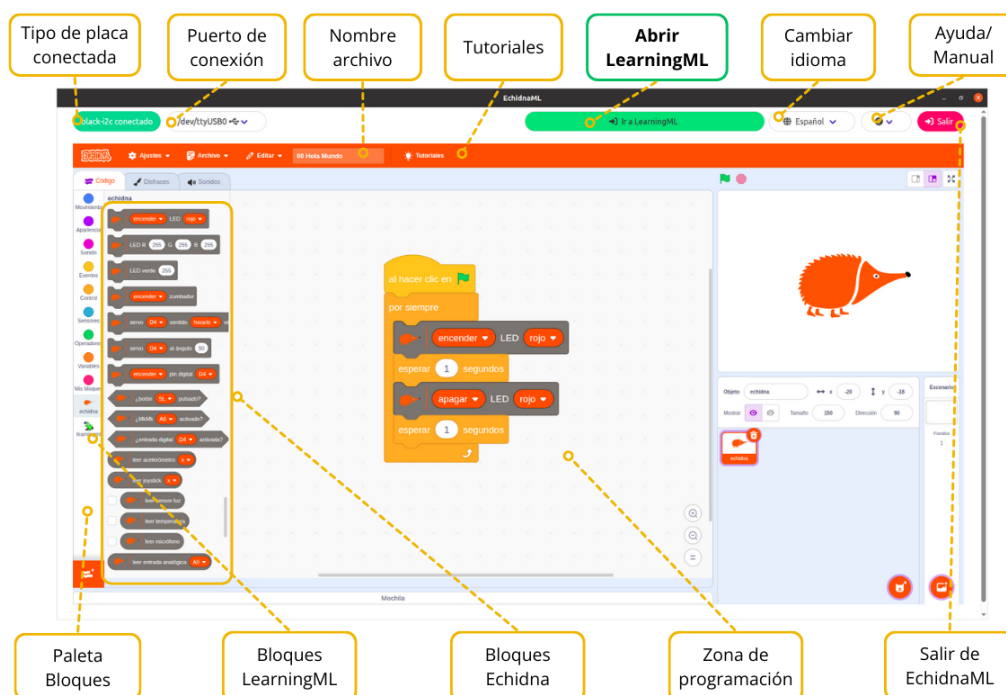
Si quieres ampliar información, consulta el [Manual de EchidnaML y EchidnaBlack](#) y la web del proyecto: www.echidna.es.



2. Proyectos con EchidnaBlocks

En estos proyectos programamos la placa **EchidnaBlack2** con los bloques de **EchidnaBlocks**. Cada proyecto presenta un componente nuevo de la placa y va creciendo en complejidad.

EchidnaBlocks es una versión de **Scratch** que incorpora **bloques** específicos para controlar la placa EchidnaBlack2 (y para usar modelos de machine learning, como veremos en los proyectos con IA).



EchidnaBlocks se **comunica** con la placa a través del **puerto serie**, mediante el cable USB: la placa envía el estado de sus **sensores** y tu programa lo procesa y le devuelve cómo deben estar sus **actuadores** (LED, zumbador...).

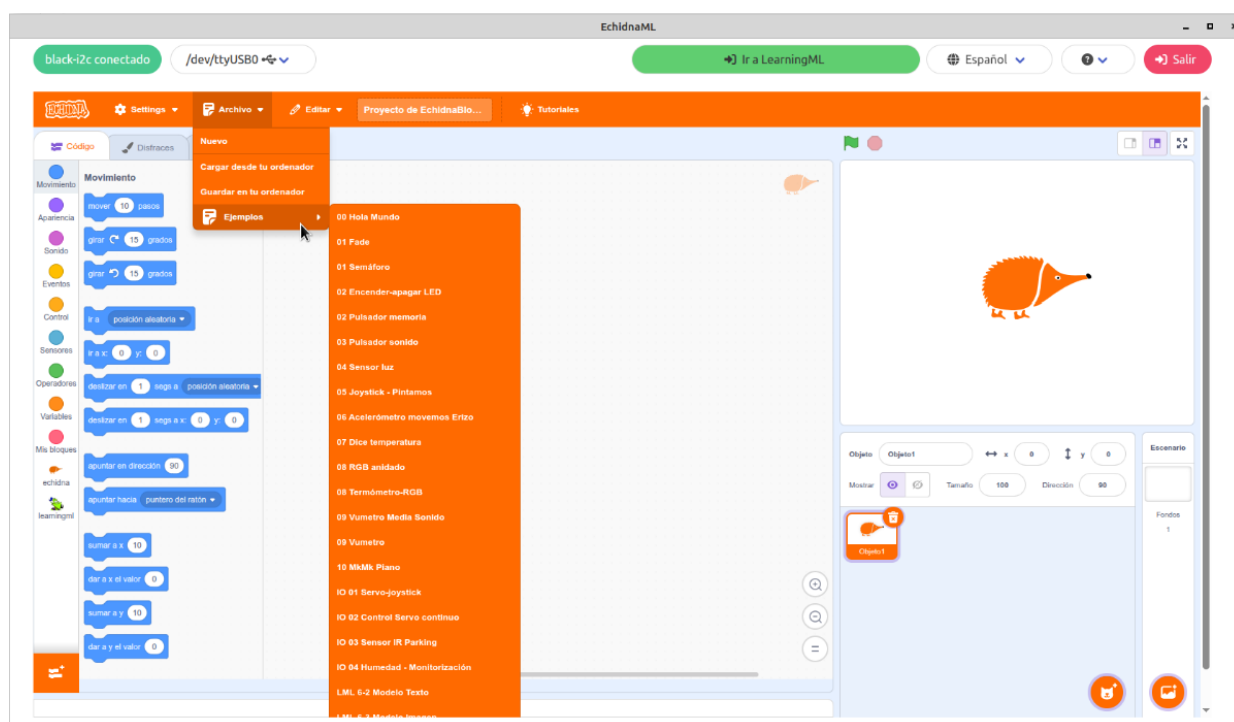
El **programa detecta automáticamente** la **placa** y la versión de la misma que estamos utilizando: EchidnaBlack o EchidnaBlack2. Para ello debemos conectar la placa antes de abrir EchidnaML.



El programa detecta automáticamente la placa y nos muestra el mensaje:



Los programas de todos los proyectos los puedes encontrar en EchidnaBlocks, en el menú **Archivo → Ejemplos**. Ábrelos para probarlos, estudiarlos y modificarlos.



Proyectos

1. [Hola Mundo](#)
2. [Semáforo](#)
3. [Interruptor de luz](#)
4. [Timbre](#)
5. [Interruptor crepuscular](#)
6. [Piano de frutas](#)
7. [El echidna dice la temperatura](#)
8. [Termómetro de colores](#)
9. [Vúmetro](#)
10. [Telesketch](#)
11. [Movemos el echidna](#)



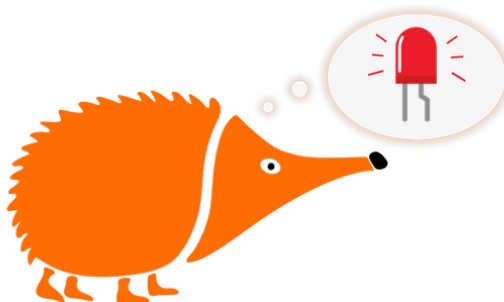
Estructura de los proyectos

Todos los proyectos siguen la misma estructura:

1. **Qué vamos a hacer:** descripción del proyecto, qué vamos a aprender y qué vamos a usar.
2. **Programamos:** el programa de bloques y cómo funciona.
3. **Mejóralo:** propuestas para ampliar el proyecto por tu cuenta.

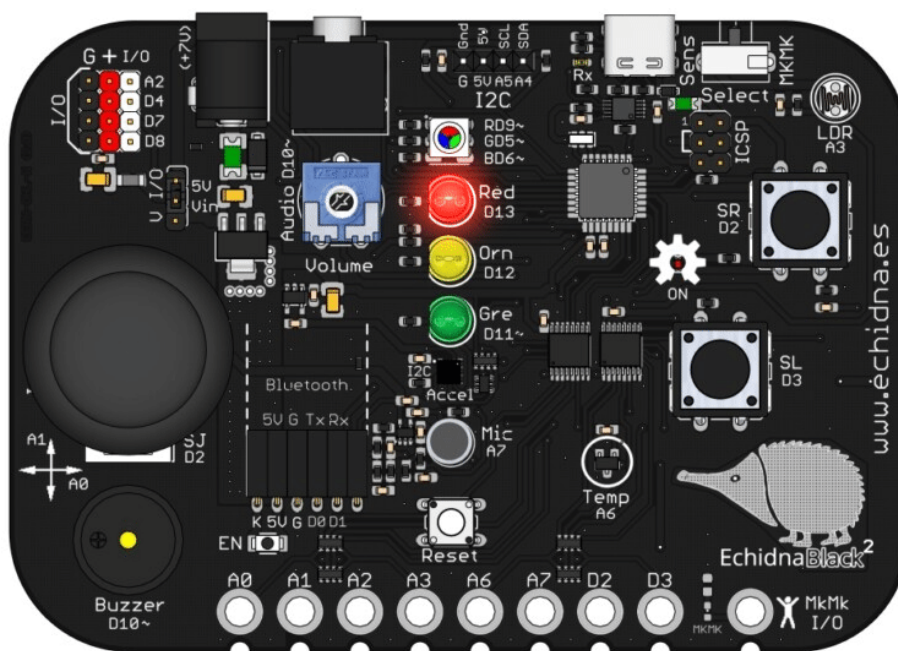


2.1 Hola Mundo



1. Qué vamos a hacer

En este primer proyecto haremos que el **LED rojo** de la placa EchidnaBlack2 se encienda y se apague de forma continua (parpadeo).



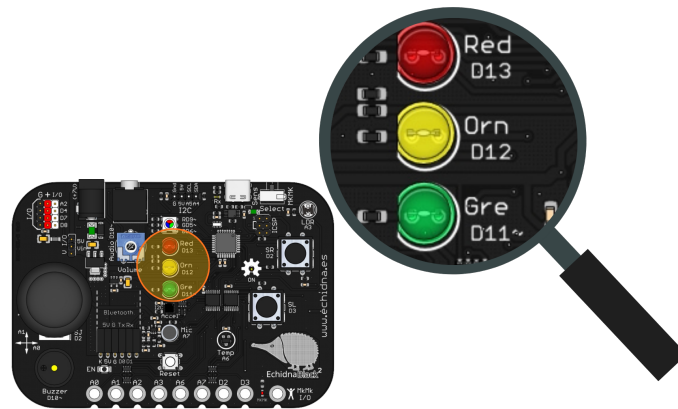
1.1 Qué vamos a aprender

- A crear nuestro primer programa en EchidnaML.
- A utilizar bucles infinitos para realizar una **programación cíclica** (tareas que se repiten para siempre).
- A controlar el estado (encendido/apagado) de un componente de la placa.

1.2 Qué vamos a usar

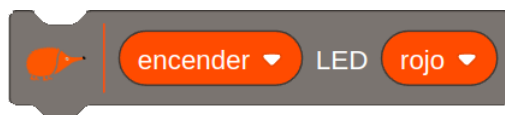
Componentes

- **LED rojo:** Se enciende y se apaga para crear el parpadeo.



Programación

Para encender y apagar los LED usamos el bloque encender LED:



En el bloque puedes elegir si quieres **encender** o **apagar** el LED y **qué LED**: verde, naranja o rojo.

2. Programamos

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

Construye el siguiente código arrastrando los bloques a tu área de trabajo:



Cómo funciona:



El programa empieza con el bloque **al hacer clic en** (bandera verde): todo lo que coloquemos debajo se ejecutará cuando pulsemos la bandera verde en EchidnaML.

El bloque **por siempre** crea un ciclo infinito que ejecuta los pasos en orden, de arriba a abajo:

1. **encender LED rojo**: Envía la señal para encender el LED.
2. **esperar 1 segundos**: Mantiene el LED encendido durante un segundo.
3. **apagar LED rojo**: Envía la señal para apagar el LED.
4. **esperar 1 segundos**: Mantiene el LED apagado durante un segundo antes de volver al paso 1.

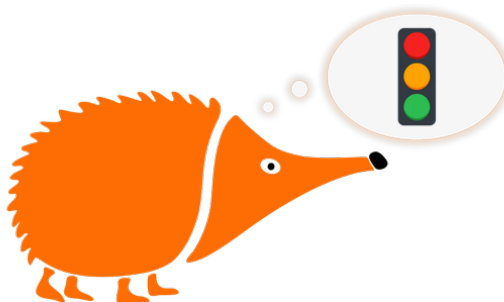
3. Mejóralo

Una vez que consigas hacer parpadear el LED, prueba a realizar estas modificaciones por tu cuenta:

1. **Ritmo rápido**: Cambia el tiempo de espera a 0.2 segundos. ¿Qué le ocurre al parpadeo? ¿Qué ocurre si sigues bajando el tiempo de espera?
2. **Sombra de señal**: Intenta que el LED esté encendido mucho tiempo (2 segundos) y apagado muy poco tiempo (0.1 segundos).
3. **LED virtual en pantalla**: Crea un objeto en EchidnaML que cambie de disfraz para simular en la pantalla el mismo parpadeo que ocurre en la placa real. **Pista**: para sincronizar el objeto LED, lo podemos hacer mediante mensajes.

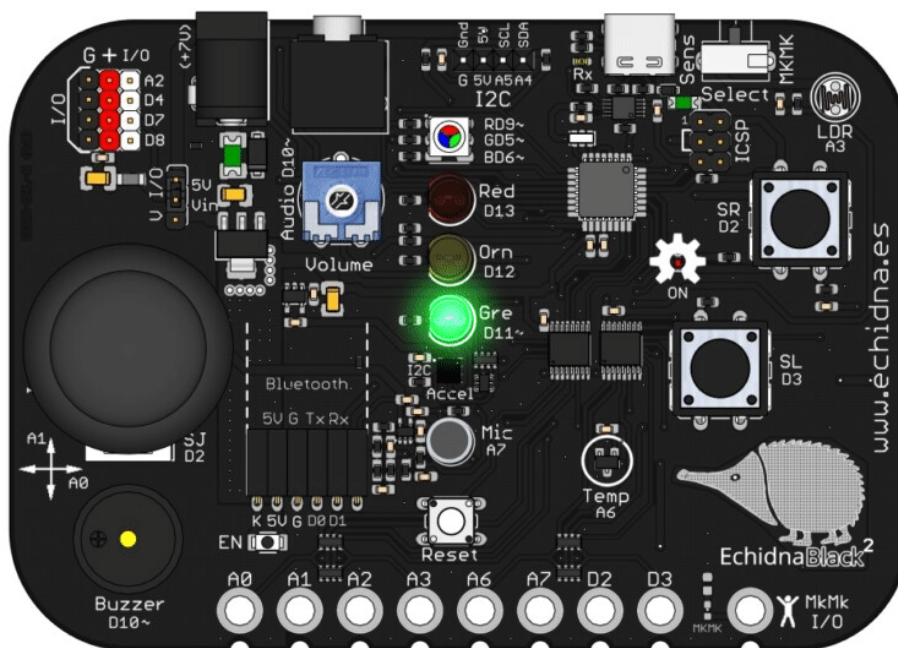


2.2 Semáforo



1. Qué vamos a hacer

Vamos a realizar un semáforo en el que el LED verde se enciende durante 5 segundos, luego se enciende el LED naranja durante 2 segundos y finalmente el LED rojo durante 5 segundos. El ciclo se repite continuamente.



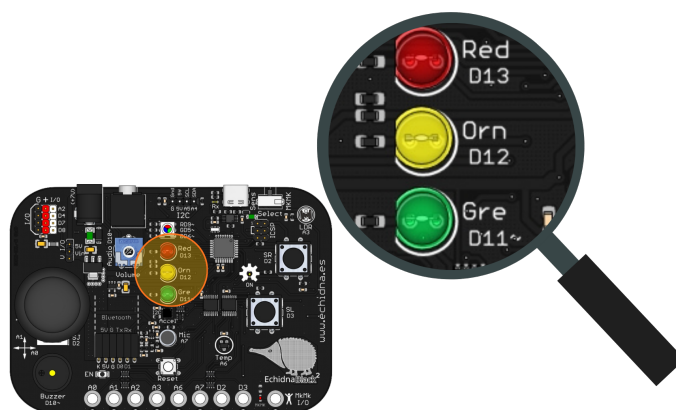
1.1 Qué vamos a aprender

- A diseñar **secuencias temporizadas** (eventos que ocurren en un orden y tiempo exactos).
- A controlar múltiples componentes digitales (LED) de forma coordinada.
- A estructurar un ciclo de estados infinito (**programación cíclica**).

1.2 Qué vamos a usar

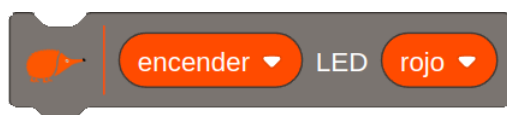
Componentes

- **LED verde:** Fase de paso.
- **LED naranja:** Fase de precaución.
- **LED rojo:** Fase de detención.



Programación

Para encender y apagar los LED usamos el bloque encender LED:



En el bloque puedes elegir si quieres **encender** o **apagar** el LED y **qué LED**: verde, naranja o rojo.

2. Programamos

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.



Cómo funciona:

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.

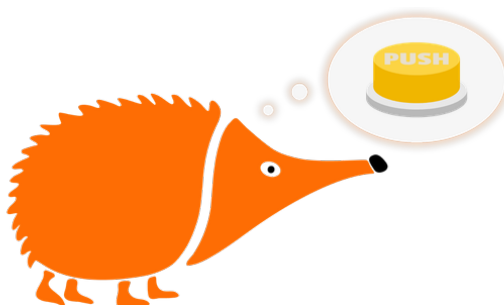
3. Mejóralo

Prueba a perfeccionar tu semáforo con algunas de estas tres mejoras que te proponemos:

1. **Naranja intermitente:** Modifica la fase intermedia para que el LED naranja no se quede fijo, sino que **parpadee 3 veces rápidas** (encendido 0.3 segundos / apagado 0.3 segundos) antes de pasar al rojo.
2. **Semáforo virtual en pantalla:** Diseña un objeto "Semáforo" en EchidnaML con tres disfraces (Verde, Naranja, Rojo). Programa el objeto para que cambie de disfraz en la pantalla al mismo tiempo que cambian los LED en la placa real.
3. **Semáforo sonoro adaptado:** Añade el **zumbador** de la placa para avisar a personas con discapacidad visual:
 - **Fase verde:** Sonido intermitente lento.
 - **Fase roja:** Sonido continuo o apagado.



2.3 Interruptor de luz



1. Qué vamos a hacer

Vamos a programar un sistema de encendido y apagado manual con dos botones:

- Al presionar el **pulsador derecho (SR)**, el **LED rojo** se encenderá.
- Al presionar el **pulsador izquierdo (SL)**, el **LED rojo** se apagará.

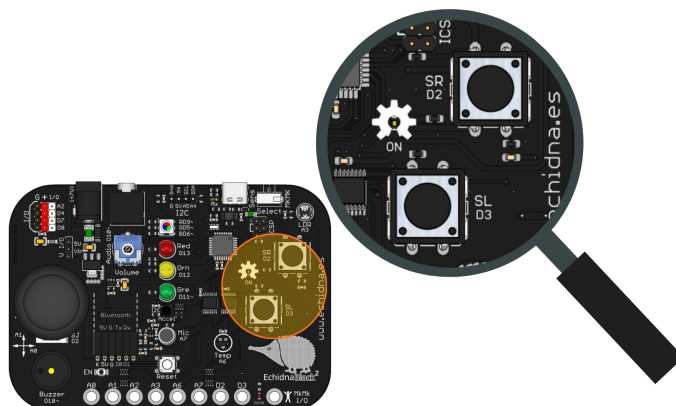
1.1 Qué vamos a aprender

- A leer **entradas digitales** (saber si un botón está presionado o no).
- A diferenciar claramente entre una **Entrada (Input)** y una **Salida (Output)** en robótica:
 - **Entrada (SR y SL):** Detecta la orden del usuario.
 - **Salida (LED rojo):** Produce la respuesta (luz).
- A tomar decisiones en el programa mediante el bloque condicional **si ... si no**: si se cumple la condición se ejecuta una parte del programa y, si no, la otra.
- A utilizar **condicionales anidados** (si ... si no y dentro otro si).
- A controlar el estado de un actuador (LED) mediante eventos físicos (pulsaciones).

1.2 Qué vamos a usar

Componentes

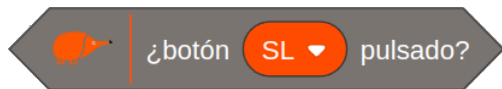
- **Pulsador SR (Switch Right / Derecho):** Para encender el LED.
- **Pulsador SL (Switch Left / Izquierdo):** Para apagar el LED.
- **LED rojo:** Componente que cambia de estado según el botón pulsado.





Programación

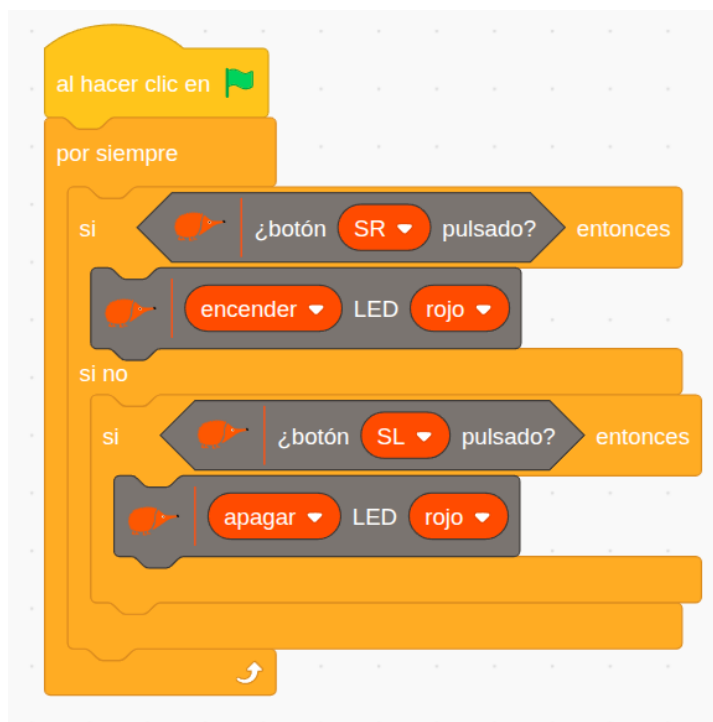
Para leer el estado de los pulsadores usamos el bloque ¿botón SL pulsado?:



En el bloque puedes elegir el pulsador **SL** o **SR**. Devuelve **verdadero** (1) si está pulsado y **falso** (0) si no lo está.

2. Programamos

El programa comprueba si el pulsador derecho está presionado; en ese caso, enciende el LED rojo. Si no lo está y presionamos el pulsador izquierdo, el LED se apaga.



Cómo funciona:

El programa revisa 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.

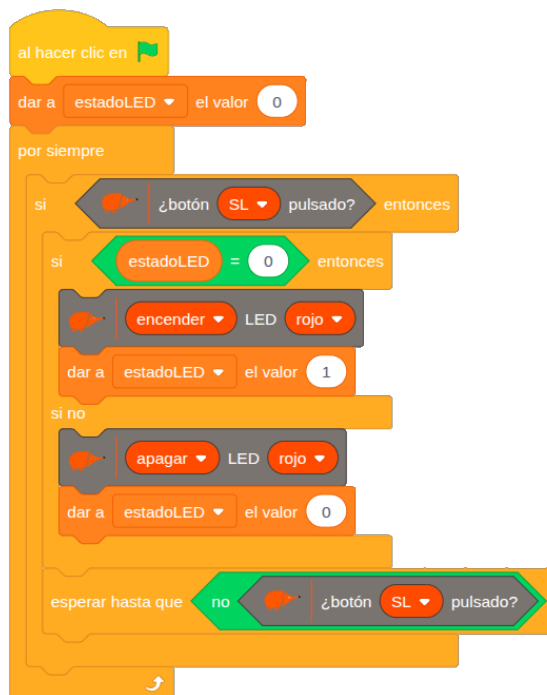


3. Mejóralo

Prueba a realizar algunas de las siguientes modificaciones al proyecto:

1. **Luz cruzada (biestable):** Añade el **LED verde** al programa para que funcionen de forma alterna:
 - Al pulsar **SR**: LED rojo encendido y LED verde apagado.
 - Al pulsar **SL**: LED rojo apagado y LED verde encendido.
2. **LED virtual en pantalla:** Crea un objeto en EchidnaML que cambie de disfraz (encendido/apagado) al mismo tiempo que cambia el LED físico de la placa.
3. **Pulsador con memoria (conmutador):** Programa un solo pulsador (por ejemplo, **SL**) para que funcione como el interruptor de la luz de tu habitación: la primera vez que lo pulsas enciende el LED, y al volverlo a pulsar lo apaga.

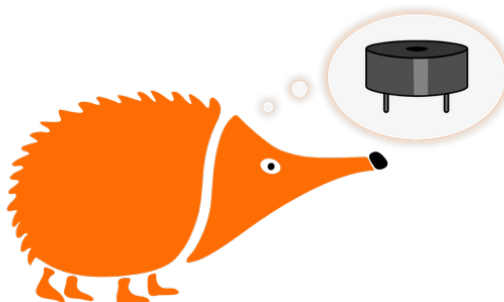
Ayuda: esta última mejora es más compleja, así que te dejamos una posible solución:



La clave es la variable **estadoLED**, que recuerda si el LED está apagado (0) o encendido (1). Cada vez que pulsamos **SL**, el programa cambia el LED al estado contrario y actualiza la variable. El bloque **esperar hasta que no ¿botón SL pulsado?** hace que el programa espere a que soltemos el pulsador; sin él, mientras lo mantenemos pulsado el LED se encendería y apagaría muchas veces seguidas.



2.4 Timbre



1. Qué vamos a hacer

Vamos a programar un timbre eléctrico: el **zumbador** emitirá un tono sonoro únicamente mientras mantengamos presionado el **pulsador izquierdo (SL)**, y se apagará inmediatamente al soltarlo.

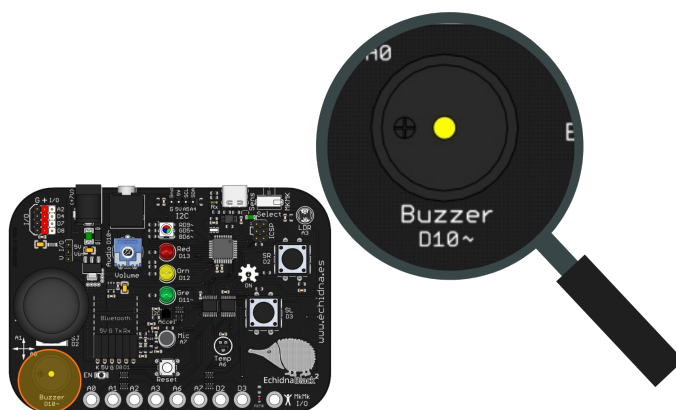
1.1 Qué vamos a aprender

- A controlar un actuador de sonido (**zumbador**) mediante un botón de entrada.
- A evaluar estados en tiempo real (presionado vs liberado).
- A usar el condicional **si ... si no** para que el zumbador suene solo mientras el pulsador está presionado.

1.2 Qué vamos a usar

Componentes

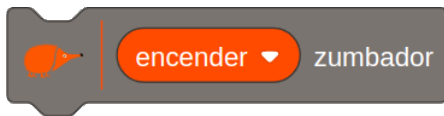
- **Pulsador SL (Switch Left / Izquierdo)**: Componente de entrada para activar el sonido.
- **Zumbador (Buzzer)**: Componente de salida que genera notas o pitidos.





Programación

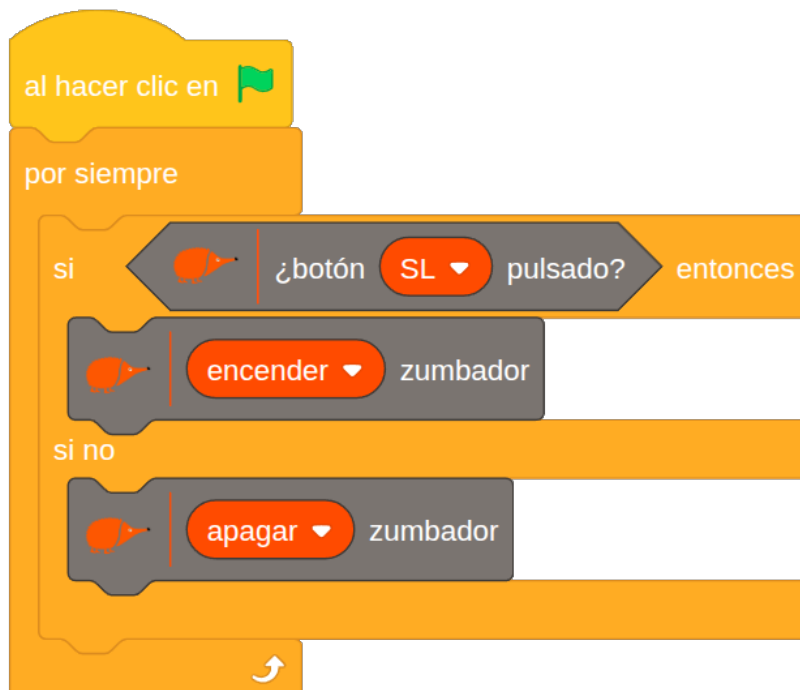
Para hacer sonar el zumbador usamos el bloque encender zumbador:



En el bloque puedes elegir si quieres **encender** o **apagar** el zumbador. El volumen se ajusta con el potenciómetro **Volume** de la placa.

2. Programamos

La programación se basa en revisar continuamente si el pulsador SL está presionado, si lo está se activa el zumbador y, si no, se apaga.



Cómo funciona:

El programa revisa continuamente:

SI el pulsador SL está presionado:
--> El zumbador suena.

SI NO (es decir, si SL está liberado):
--> El zumbador deja de sonar.

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



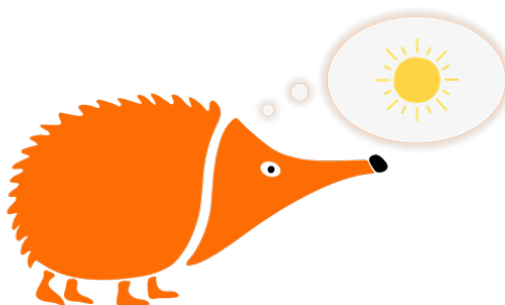
3. Mejóralo

Prueba a realizar alguna de estas mejoras en tu proyecto:

1. **Alarma intermitente:** Modifica el programa para que, al mantener pulsado **SL**, el sonido no sea continuo, sino que emita pitidos intermitentes tipo alarma (sonido 0.1 segundos, silencio 0.1 segundos).
2. **Emisor de código Morse:** Programa los dos pulsadores para que emitan pitidos de distinta duración: **SL**, un pitido corto (punto, 0.2 segundos), y **SR**, un pitido largo (raya, 0.6 segundos). ¡Combina puntos y rayas para enviar mensajes secretos en código Morse a tus compañeros!
3. **Efecto visual de altavoz (vibración en pantalla):** Crea o selecciona un objeto en EchidnaML con forma de altavoz o campana. Haz que el objeto cambie de tamaño ligeramente o gire de un lado a otro (simulando vibración) mientras el zumbador esté sonando.



2.5 Interruptor crepuscular



1. Qué vamos a hacer

Vamos a programar un sistema automático similar al de las farolas de la calle: el **LED verde** se encenderá automáticamente cuando la luz ambiental baje (de noche) y se apagará cuando haya suficiente luz (de día).

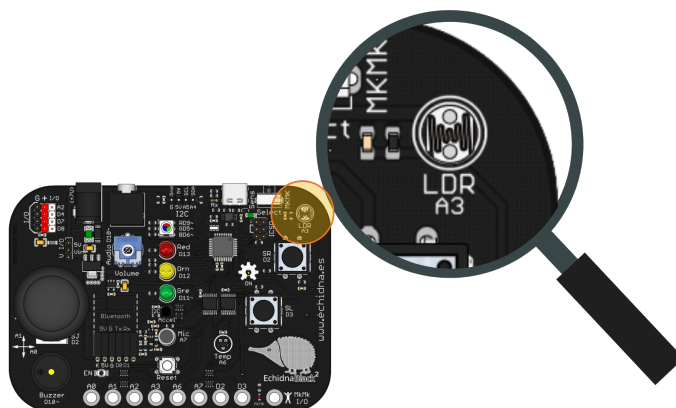
1.1 Qué vamos a aprender

- A programar un **sistema automático** que reaccione al entorno.
- A utilizar el **sensor de luz (LDR)** para medir la iluminación ambiental.
- A usar un **operador de comparación (<)** para que el condicional `si ... si no` decida a partir de la lectura de un sensor.
- A trabajar con **umbrales numéricos** para definir estados (día/noche).

1.2 Qué vamos a usar

Componentes

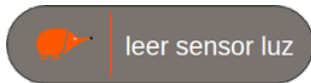
- **Sensor de luz (LDR):** Mide la cantidad de luz que recibe. Cuanto más oscuro esté el entorno, menor será el valor registrado.
- **LED verde:** Funcionará como nuestra luz automática.





Programación

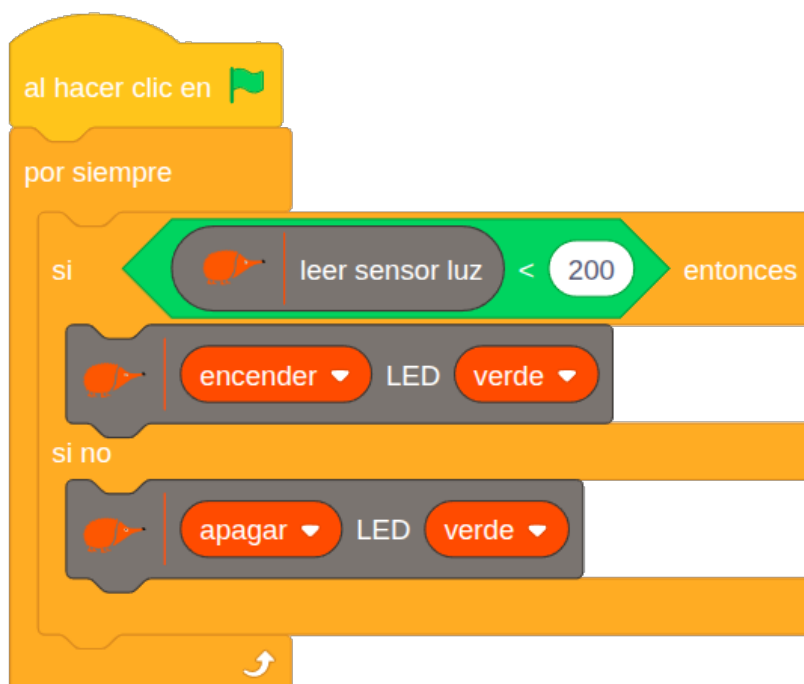
Para leer el sensor de luz usamos el bloque leer sensor luz:



Nos da valores entre **0** (no hay luz) y **1023** (mucha luz). Si marcas la casilla que hay junto al bloque, verás en el escenario el valor que mide en cada momento.

2. Programamos

Revisamos continuamente el valor del sensor de luz: si es menor que un cierto umbral, encendemos el LED; en caso contrario, lo apagamos.



Cómo funciona:

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 o iguales a 200):
--> Se apaga el LED verde.

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



3. Mejóralo

Prueba a realizar algunas de estas mejoras en tu proyecto de forma autónoma:

1. **Calibra tu aula:** Averigua qué valor lee el sensor de luz en tu mesa y ajusta el umbral exacto para que la luz se encienda solo cuando tapes el sensor con la mano. Para ver en pantalla el valor del sensor de luz, marca la casilla que hay junto al bloque `Leer sensor luz`.
2. **Fondo de día y noche:** Añade dos fondos al escenario de EchidnaML (uno soleado y otro nocturno). Haz que el fondo cambie en la pantalla al mismo tiempo que se enciende o apaga el LED en la placa.
3. **Luz blanca:** Sustituye el LED verde por el **LED RGB** para que ilumine más: cuando haya oscuridad, se encenderá en color **blanco**, y cuando haya mucha luz, se apagará. **Pista:** con el bloque `LED R G B`, el blanco se consigue poniendo los tres colores a 255, y para apagarlo los ponemos a 0.



2.6 Piano de frutas



1. Qué vamos a hacer

Vamos a convertir la entrada **MkMk A0** de nuestra placa en una tecla de piano táctil. Cada vez que toques la entrada (o un objeto conductor conectado a ella), el ordenador emitirá una nota musical.

1.1 Qué vamos a aprender

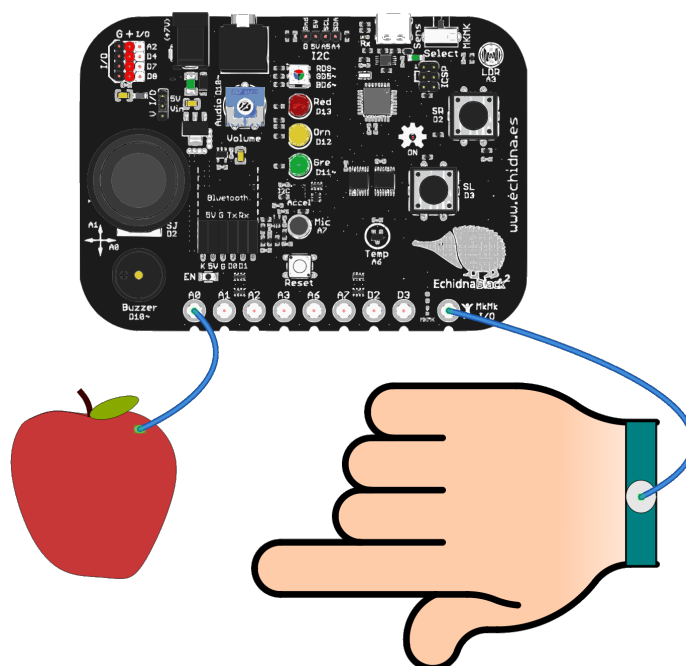
- A detectar la conductividad eléctrica de objetos cotidianos usando el **modo Makey Makey (MkMk)**.
- A conectar cables de cocodrilo en los conectores externos de la placa.
- A entender cómo funciona un **circuito cerrado** a través de nuestro propio cuerpo.
- A reproducir notas musicales y sonidos desde el software.

1.2 Qué vamos a usar

Componentes

- **Entrada MkMk A0:** Conector táctil para detectar pulsaciones o conductividad.
- **Conector común MkMk I/O (5V):** Indispensable para cerrar el circuito con tu cuerpo. Está en el extremo derecho de la fila de conectores y proporciona 5V.
- **Cables de cocodrilo:** Para conectar objetos externos (frutas, plastilina, papel de aluminio, etc.).

¡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.



En la imagen, la fruta está conectada a la entrada **A0** y la pulsera al conector común **MkMk I/O** (5V). Al tocar la fruta con la mano, una corriente muy pequeña pasa a través de tu cuerpo, el circuito se cierra y la placa detecta el contacto en la entrada A0.

Programación

Para leer una entrada MkMk usamos el bloque Leer MK MK:



En el bloque puedes elegir la entrada: **A0, A1, A2, A3, A6, A7, D2 o D3**. Devuelve **verdadero** (1) cuando detecta que el circuito se ha cerrado (la lectura de la entrada supera **350** en una escala de 0 a 1023) y **falso** (0) cuando el circuito está abierto.

2. Programamos

Primero elegimos el instrumento con el bloque **fijar instrumento a (1) Piano**. Después revisamos continuamente la entrada MkMk A0: **cundo detecta contacto**, se reproduce la **nota 60** (que corresponde a la nota Do central del piano) durante **0.25 tiempos** (con el tempo por defecto, 60 pulsos por minuto, equivale a un cuarto de segundo).



Cómo funciona:

El programa revisa continuamente:

SI la entrada MkMk A0 detecta contacto:
--> Suena la nota 60 (Do central) durante 0.25 tiempos.

Si no hay contacto, no ocurre nada y el programa vuelve a comprobar la entrada.

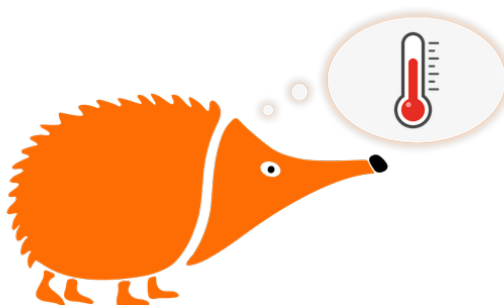
3. Mejóralo

Prueba a realizar algunas de las siguientes modificaciones al proyecto:

1. **Animación en pantalla:** Añade un objeto en EchidnaML (un instrumento o una nota musical) que cambie de disfraz, baile o salte en la pantalla cada vez que suene la nota.
2. **Luz y sonido:** Haz que el **LED RGB** se ilumine de un color distinto cada vez que toques una tecla de tu piano MkMk.
3. **Escala musical:** Utiliza los otros conectores disponibles (**A1, A2, A3**) con otros cables para programar diferentes notas (por ejemplo: Do=60, Re=62, Mi=64, Fa=65) y crea un mini teclado completo.



2.7 El echidna dice la temperatura



1. Qué vamos a hacer



Hola, hace una temperatura de 16.61°C

Vamos a convertir a nuestro echidna en un **hombre del tiempo**: cada vez que pulses la tecla **t** del teclado del ordenador, el echidna nos dirá en un bocadillo qué **temperatura** hace, medida con el sensor de la placa.

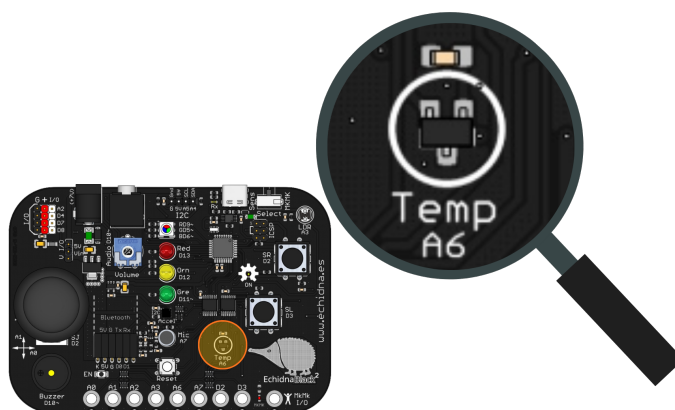
1.1 Qué vamos a aprender

- A utilizar el **sensor de temperatura** para medir la temperatura ambiente en **grados Celsius** (°C).
- A iniciar un programa al **pulsar una tecla** del ordenador.
- A **unir textos** y valores de un sensor para construir una frase.
- A hacer que un personaje **diga** mensajes en un bocadillo.

1.2 Qué vamos a usar

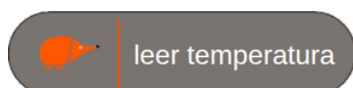
Componentes

- **Sensor de temperatura:** Mide la temperatura del ambiente. Entrega un voltaje que depende de la temperatura.



Programación

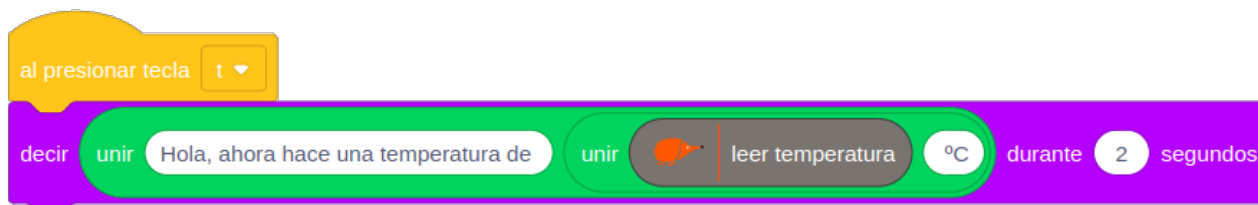
Para leer la temperatura usamos el bloque leer temperatura, que convierte ese voltaje directamente a grados Celsius (°C):



Si marcas la casilla que hay junto al bloque, verás la temperatura en el escenario.

2. Programamos

Usaremos el personaje de **Echidna** que aparece en el escenario. El programa empieza al pulsar la tecla **t**: el echidna dice durante 2 segundos una frase formada con el bloque unir, que junta tres partes: el texto "Hola, ahora hace una temperatura de ", el valor del sensor y el símbolo "°C".



Cómo funciona:

CUANDO se pulsa la tecla t:

--> Se lee la temperatura del sensor.

--> El echidna dice la frase con la temperatura durante 2 segundos.

Como dentro de un bloque unir solo caben dos partes, usamos un unir dentro de otro para juntar las tres.



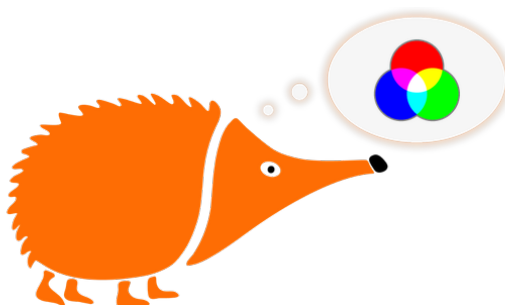
3. Mejóralo

Prueba a realizar algunas de estas mejoras en tu proyecto de forma autónoma:

1. **Sin tantos decimales:** El sensor da la temperatura con decimales (por ejemplo, 16.61). Usa el bloque redondear para que el echidna diga solo un número entero.
2. **Grados Fahrenheit:** Haz que el echidna diga también la temperatura en grados Fahrenheit (°F). Para calcularla, multiplica los grados Celsius por 1.8 y súmale 32.
3. **Fondo de frío y calor:** Añade dos fondos al escenario de EchidnaML, uno que represente el frío (por ejemplo, un paisaje nevado) y otro el calor (por ejemplo, una playa). Haz que se muestre uno u otro según si la temperatura está por debajo o por encima de un valor. Para probarlo, calienta el sensor tocándolo suavemente con el dedo.



2.8 Termómetro de colores



1. Qué vamos a hacer

Vamos a construir un **termómetro de colores**: el **LED RGB** de la placa cambiará de color automáticamente según la temperatura ambiente. Se encenderá en **azul** si hace frío, en **verde** si la temperatura es agradable y en **rojo** si hace calor.

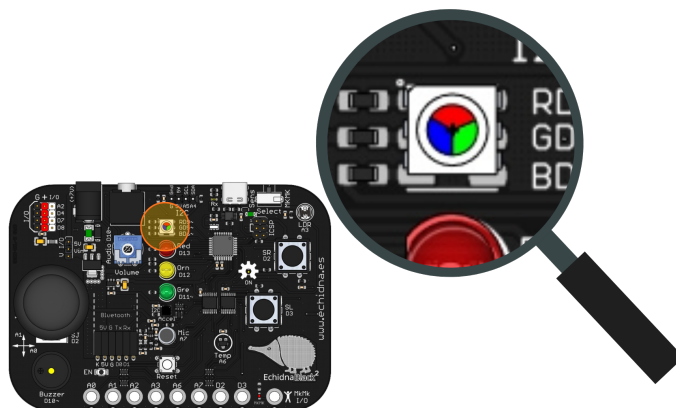
1.1 Qué vamos a aprender

- A utilizar el **LED RGB** y a **mezclar** sus tres colores (rojo, verde y azul) para conseguir otros.
- A ajustar la **intensidad** de cada color con valores de 0 a 255.
- A usar el **sensor de temperatura** como entrada para controlar un actuador.
- A usar un condicional **si ... si no dentro de otro** para distinguir tres zonas de temperatura.

1.2 Qué vamos a usar

Componentes

- **LED RGB**: Componente que tiene dentro tres LED: uno **rojo** (R, Red), uno **verde** (G, Green) y uno **azul** (B, Blue). Al mezclar la luz de los tres podemos conseguir más de 16 millones de colores.
- **Sensor de temperatura**: Mide la temperatura del ambiente en grados Celsius (°C).





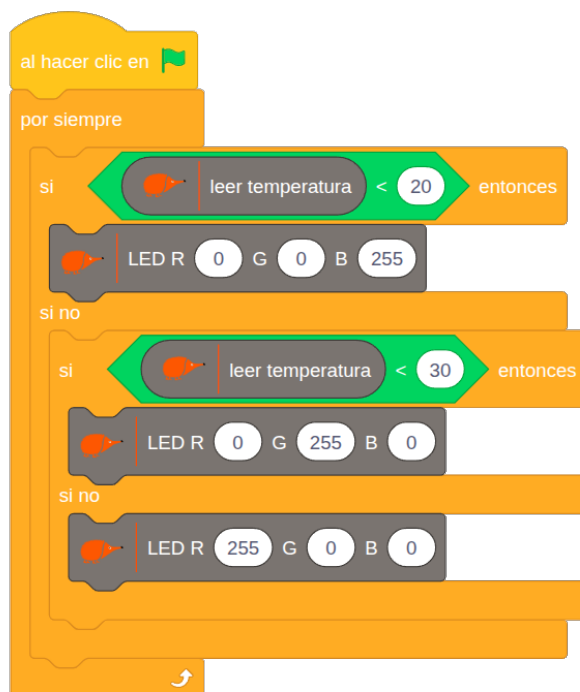
Programación

Para controlar el LED RGB usamos el bloque LED R G B, en el que damos a cada color un valor entre **0** (apagado) y **255** (máxima intensidad). Por ejemplo, para conseguir el **naranja Echidna** mezclamos mucho rojo (254), algo de verde (109) y casi nada de azul (4):



2. Programamos

Revisamos continuamente la temperatura y encendemos el LED RGB de un color según la zona en la que esté: para el azul solo damos valor al canal B, para el verde solo al G y para el rojo solo al R.



Cómo funciona:

El programa revisa continuamente:

SI la temperatura es menor de 20 °C (zona fría):

--> El LED RGB se ilumina en azul.

SI NO:

SI la temperatura es menor de 30 °C (entre 20 °C y 30 °C, zona agradable):

--> El LED RGB se ilumina en verde.

SI NO (30 °C o más, zona caliente):

--> El LED RGB se ilumina en rojo.



El segundo `si ... si` no solo se comprueba cuando la temperatura no es menor de 20 °C. Así, con dos condicionales, distinguimos tres zonas.

3. Mejóralo

Prueba a realizar algunas de estas mejoras en tu proyecto de forma autónoma:

1. **El echidna avisa:** Haz que el echidna del escenario diga "¡Qué frío!", "¡Qué bien se está!" o "¡Qué calor!" según el color que se encienda.
2. **Tu paleta de colores:** Prueba a mezclar colores en el bloque LED R G B y descubre qué valores necesitas para conseguir **amarillo**, **morado**, **blanco** o tu color favorito. Después úsalos para añadir una zona **muy fría** (menos de 10 °C) y otra **muy caliente** (más de 35 °C).
3. **Arcoíris:** Crea una variable y usa un bucle `repetir` para que el LED RGB pase poco a poco del azul al rojo, sumando 1 al rojo y restando 1 al azul en cada vuelta.



2.9 Vúmetro



1. Qué vamos a hacer

Vamos a construir un **vúmetro** o **semáforo de ruido**, que muestra con los LED de la placa cuánto ruido hay a nuestro alrededor. Con silencio se encenderá solo el **LED verde**; si el ruido aumenta, se encenderá también el **naranja**; y si hay mucho ruido, se encenderán los **tres LED**.

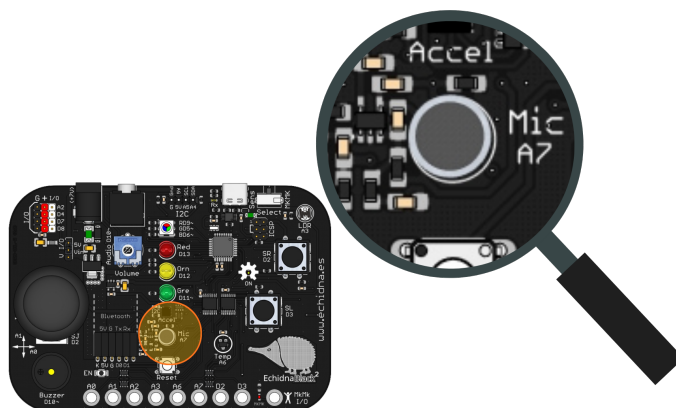
1.1 Qué vamos a aprender

- A utilizar el **micrófono** para medir la **intensidad del sonido**.
- A usar un condicional **si ... si no dentro de otro** para distinguir tres niveles.
- A trabajar con **umbrales numéricos** para definir estados (silencio, ruido moderado y mucho ruido).
- A entender por qué algunas señales, como el sonido, **cambian constantemente**.

1.2 Qué vamos a usar

Componentes

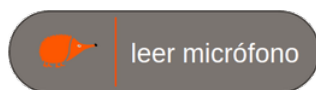
- **Micrófono:** Convierte las vibraciones del sonido en una señal eléctrica. Da valores bajos con silencio y valores más altos cuanto más intenso es el sonido, entre 0 y 1023.
- **LED verde, naranja y rojo:** Nos indicarán el nivel de ruido.





Programación

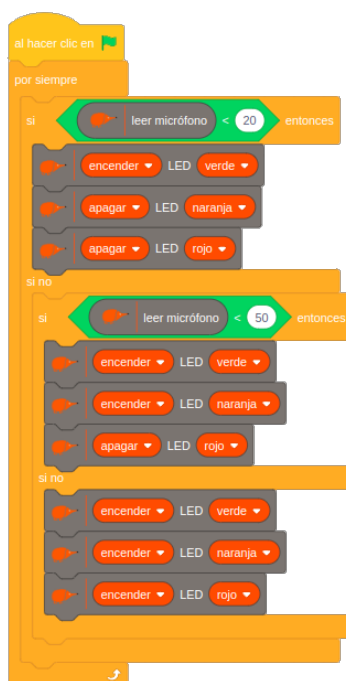
Para leer el micrófono usamos el bloque leer micrófono:



Si marcas la casilla que hay junto al bloque, verás en el escenario el valor que mide en cada momento.

2. Programamos

Revisamos continuamente el valor del micrófono y, según el nivel de ruido, encendemos o apagamos cada uno de los tres LED.



Cómo funciona:

El programa revisa continuamente:

SI el micrófono registra valores menores de 20 (silencio):

--> Se enciende el LED verde y se apagan el naranja y el rojo.

SI NO:

SI registra valores menores de 50 (entre 20 y 50, ruido moderado):

--> Se encienden los LED verde y naranja y se apaga el rojo.

SI NO (50 o más, mucho ruido):

--> Se encienden los tres LED.



Es probable que veas que los LED **parpadean** aunque el ruido sea constante. Esto ocurre porque la señal del sonido cambia muy deprisa: el micrófono capta las vibraciones de la onda sonora y cada lectura da un valor distinto. En la primera propuesta de Mejóralo aprenderás a solucionarlo.

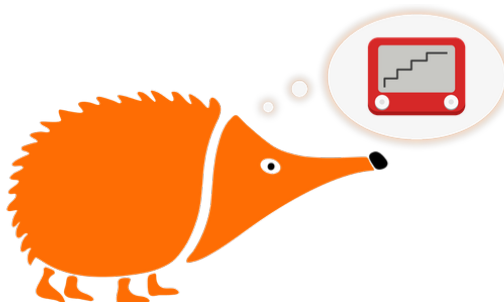
3. Mejóralo

Prueba a realizar algunas de estas mejoras en tu proyecto de forma autónoma:

1. **Una lectura más estable:** Para que los LED no parpadeen, en lugar de usar cada lectura del micrófono calcula la **media** de varias (por ejemplo, de 10) y usa esa media en los condicionales. **Pista:** necesitas una variable que vaya sumando las lecturas (suma) y otra que las cuente (numeroDatos); la media es $\text{suma} / \text{numeroDatos}$. **Ayuda:** puedes encontrar la programación en **Archivo → Ejemplos → Vumetro_media_sonido**.
2. **Calibra tu aula:** Observa qué valores mide el micrófono con silencio, hablando en voz baja y dando una palmada, y ajusta los umbrales (20 y 50) para que el vúmetro funcione bien en tu clase.
3. **Vúmetro virtual en pantalla:** Diseña un vúmetro en la pantalla: crea un objeto en EchidnaML con cuatro disfraces (sin barras, una, dos y tres barras) y haz que cambie de disfraz al mismo tiempo que se encienden los LED de la placa.



2.10 Telesketch



1. Qué vamos a hacer

Vamos a convertir la pantalla de EchidnaML en un **Telesketch**, la pizarra mágica que dibuja líneas con dos ruedas. En nuestro caso usaremos el **joystick** de la placa: al mover la palanca hacia un lado, el **lápiz** dibujará una línea en el escenario en esa misma dirección.

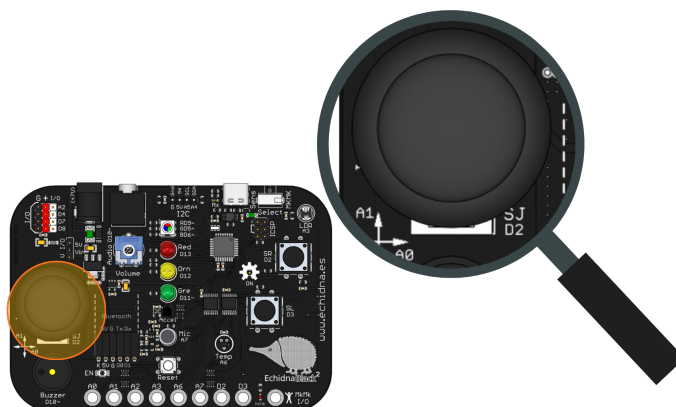
1.1 Qué vamos a aprender

- A utilizar el **joystick** y sus dos ejes: horizontal (**X**) y vertical (**Y**).
- A leer **valores analógicos** entre 0 y 1023.
- A usar la extensión **Lápiz** de EchidnaML para dibujar en el escenario.
- A combinar varios condicionales **si** con **umbrales** (**>** y **<**) para detectar hacia dónde movemos la palanca.
- A mover un objeto por el escenario usando **direcciones** y **coordenadas**.

1.2 Qué vamos a usar

Componentes

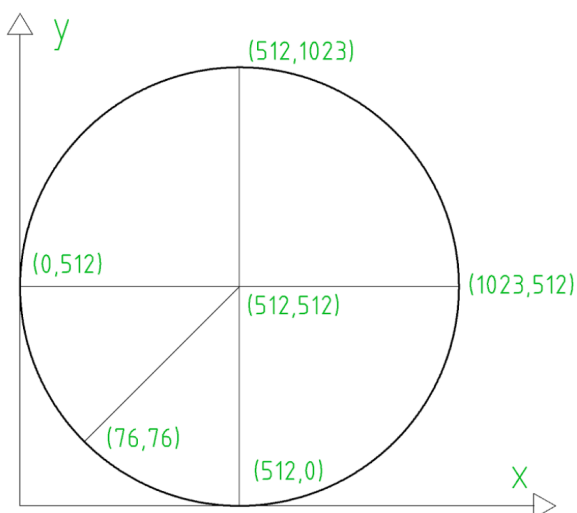
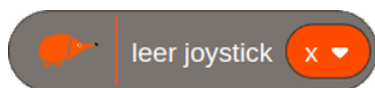
- **Joystick:** Palanca que se mueve en varias direcciones. Por dentro tiene dos potenciómetros, uno para cada eje, que indican hacia dónde y cuánto la has desplazado. Además, al presionar la palanca hacia abajo funciona como un pulsador, que en EchidnaBlack2 es el mismo que el pulsador **SR**.





Programación

Para leer el joystick usamos el bloque leer joystick, en el que puedes elegir el eje **x** o el eje **y**:



En **reposo**, el joystick da valores alrededor de **512** en los dos ejes. En el **eje X** da **0** si mueves la palanca a la izquierda y **1023** si la mueves a la derecha. En el **eje Y** da **0** si la mueves hacia abajo y **1023** si la mueves hacia arriba.

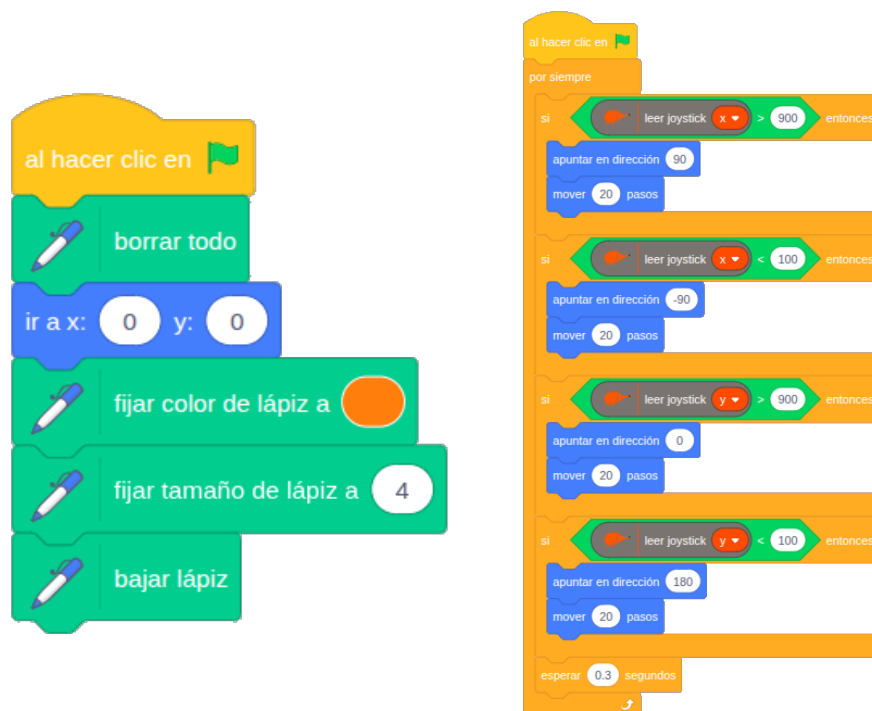
2. Programamos

Añade la extensión Lápiz: pulsa el botón de extensiones, en la esquina inferior izquierda de EchidnaML, y elige **Lápiz**. Así tendrás los bloques para dibujar en el escenario.

Configuración inicial: al hacer clic en la bandera verde preparamos el lápiz para que:

- Borre todo lo que haya dibujado en el escenario.
- Se coloque en el centro del escenario, el punto (0, 0).
- Fije el color y el grosor con el que va a pintar.
- Baje el lápiz, para que dibuje al moverse.

Control con el joystick: revisamos continuamente los dos ejes del joystick. Si la palanca está desplazada hacia un lado, el objeto apunta en esa dirección y avanza 20 pasos, dibujando una línea.



Cómo funciona:

El programa revisa continuamente:

SI el eje x es mayor de 900 (palanca a la derecha):

--> Apunta en dirección 90 y avanza 20 pasos.

SI el eje x es menor de 100 (palanca a la izquierda):

--> Apunta en dirección -90 y avanza 20 pasos.

SI el eje y es mayor de 900 (palanca hacia arriba):

--> Apunta en dirección 0 y avanza 20 pasos.

SI el eje y es menor de 100 (palanca hacia abajo):

--> Apunta en dirección 180 y avanza 20 pasos.

Los valores 900 y 100 actúan como umbrales: mientras la palanca está en reposo (alrededor de 512) no se cumple ninguna condición y el lápiz no se mueve. El bloque esperar 0.3 segundos evita que el lápiz se mueva demasiado rápido.

3. Mejóralo

Prueba a realizar algunas de estas mejoras en tu proyecto de forma autónoma:

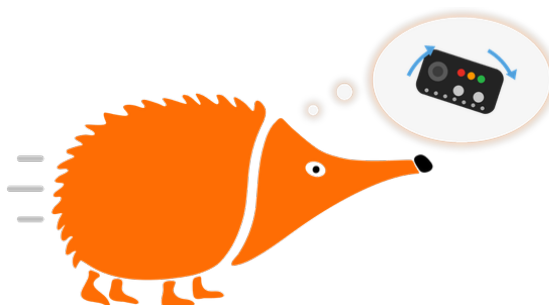
1. **Borra la pizarra:** Como en un Telesketch de verdad, haz que al presionar el joystick (pulsador **SR**) se borre el dibujo y el lápiz vuelva al centro del escenario.
2. **Cambia de color:** Usa el pulsador **SL** para cambiar el color del lápiz cada vez que lo pulses, y enciende el **LED RGB** del mismo color para saber con cuál estás pintando.



3. **Controla la velocidad:** Haz que el lápiz avance más o menos pasos según cuánto muevas la palanca del joystick. **Pista:** usa los bloques `cambiar x en y` y `cambiar y en` con el valor de cada eje menos 512 (para que en reposo valga 0) y divídelo, por ejemplo, entre 50.



2.11 Movemos el echidna



1. Qué vamos a hacer

Vamos a convertir la placa en un **mando de videojuego**: al **inclin**ar la **placa** hacia un lado, el personaje de Echidna se moverá por la pantalla en esa dirección, y si **levantas la placa de golpe**, el personaje dará un **salto**.

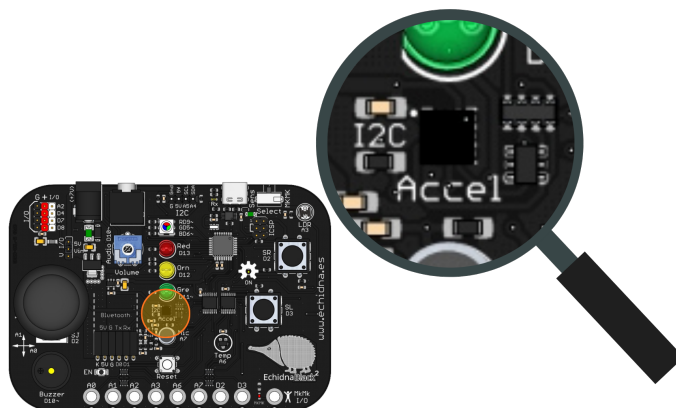
1.1 Qué vamos a aprender

- A utilizar el **acelerómetro** para detectar la **inclinación** y los **movimientos bruscos** de la placa.
- A leer los valores de los tres ejes: **X**, **Y** y **Z**.
- A mover un personaje por el escenario usando sus **coordenadas** (sumar a **x** y sumar a **y**).
- A animar un personaje cambiando de **disfraz**.
- A ejecutar dos programas a la vez usando **dos hilos** de ejecución.

1.2 Qué vamos a usar

Componentes

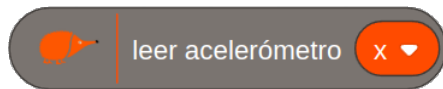
- **Acelerómetro**: Sensor que mide los movimientos de la placa en los tres ejes. Con los ejes **X** e **Y** detecta hacia dónde inclinas la placa, gracias a la fuerza de la gravedad. Con el eje **Z** detecta los movimientos bruscos hacia arriba o hacia abajo.





Programación

Para leer el acelerómetro usamos el bloque **leer acelerómetro**, en el que puedes elegir el eje **x**, **y** o **z**:



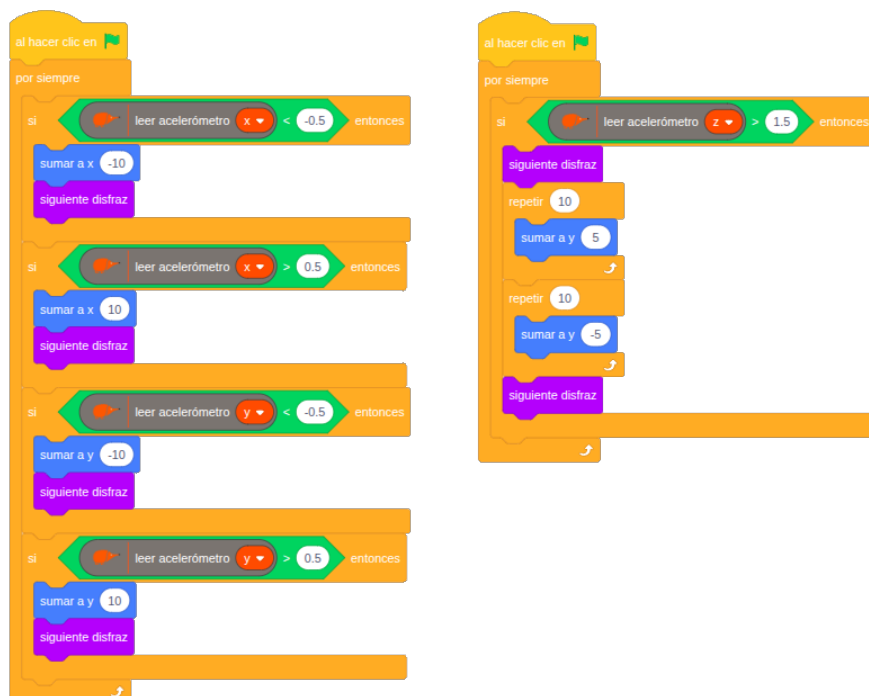
Estos son los valores que nos da:

- **En reposo:** alrededor de 0 en los ejes X e Y, y alrededor de 1 en el eje Z.
- **Ejes X e Y:** al inclinar la placa, el valor cambia de 0 hasta -1 hacia un lado y hasta 1 hacia el otro.
- **Eje Z:** al mover la placa bruscamente, el valor se aleja mucho de 1.

2. Programamos

Usaremos el personaje de **Echidna** que aparece en el escenario. El programa tiene **dos hilos** de ejecución, es decir, dos grupos de bloques que empiezan a la vez al hacer clic en la bandera verde y funcionan al mismo tiempo:

- **Hilo de movimiento:** revisa continuamente los ejes X e Y. Si la placa está inclinada, suma o resta 10 a la posición del personaje y cambia de disfraz para que parezca que camina.
- **Hilo de salto:** revisa continuamente el eje Z. Si detecta un movimiento brusco, el personaje sube 50 pasos (10 veces 5) y vuelve a bajar.



Cómo funciona:

Hilo de movimiento, el programa revisa continuamente:



SI el eje x es menor de -0,5 (placa inclinada a la izquierda):

--> El personaje se desplaza hacia la izquierda.

SI el eje x es mayor de 0,5 (placa inclinada a la derecha):

--> El personaje se desplaza hacia la derecha.

SI el eje y es menor de -0,5 (placa inclinada hacia atrás):

--> El personaje se desplaza hacia abajo.

SI el eje y es mayor de 0,5 (placa inclinada hacia delante):

--> El personaje se desplaza hacia arriba.

Hilo de salto, el programa revisa continuamente:

SI el eje z es mayor de 1,5 (placa levantada de golpe):

--> El personaje salta: sube y vuelve a bajar.

Los valores 0,5 y -0,5 actúan como umbrales: si la placa está casi plana, no se cumple ninguna condición y el personaje se queda quieto.

3. Mejóralo

Prueba a realizar algunas de estas mejoras en tu proyecto de forma autónoma:

1. **Sin salirse de la pantalla:** Haz que el personaje vuelva al centro del escenario al hacer clic en la bandera verde y que no pueda salirse por los bordes. **Pista:** usa el bloque rebotar si toca un borde.
2. **Salto con efectos:** Haz que, cada vez que el personaje salte, suene el **zumbador** y se encienda el **LED RGB**.
3. **Recoge la comida:** Añade un nuevo objeto (por ejemplo, una hormiga) en una posición al azar. Cuando el echidna lo toque, suma un punto a una **variable** y mueve la comida a otra posición al azar.



3. Proyectos con IA

EchidnaML integra **LearningML**, una **plataforma educativa** diseñada para aprender los fundamentos del **machine learning** (aprendizaje automático) de forma sencilla y visual. Esto nos permite incorporar **inteligencia artificial** a nuestros proyectos con la placa EchidnaBlack2.

En estos proyectos crearemos **modelos** que clasifican **textos, imágenes y números**, y después los usaremos en nuestros programas con los bloques de LearningML de EchidnaBlocks. Así entenderemos cómo se **entrenan** los modelos, qué **datos** necesitan, cómo influyen los **sesgos** y cómo se usan después para tomar **decisiones**.

Conceptos clave

- **Modelo:** lo que construye el ordenador a partir de los ejemplos para clasificar datos nuevos.
- **Clase:** cada una de las categorías en las que el modelo puede clasificar un dato (por ejemplo, Enciende y Apaga).
- **Ejemplo:** cada dato que añadimos a una clase para enseñar al modelo cómo es esa clase.
- **Confianza:** el porcentaje de seguridad que tiene el modelo en su respuesta.

El entorno de LearningML

Para abrir LearningML pulsamos el botón **Ir a LearningML** de EchidnaBlocks, y para volver, el botón **Ir a EchidnaBlocks**.



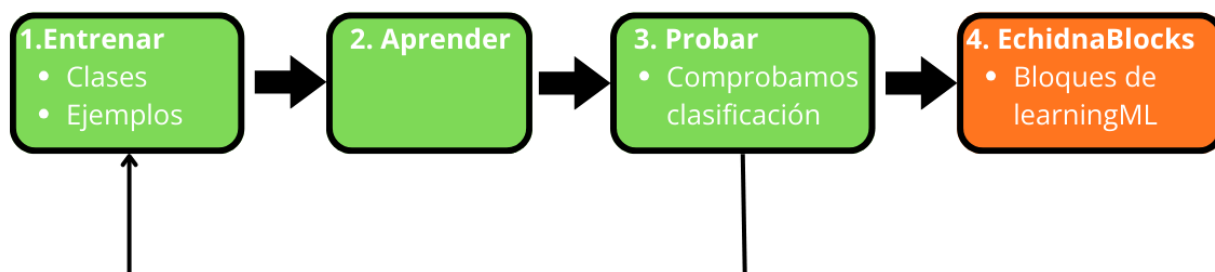
En estos proyectos usaremos sobre todo:

- **Nombre del archivo (6):** el nombre con el que guardamos el modelo.
- **Tipos de modelo (7):** elegimos si queremos reconocer **textos, imágenes o números**.
- **Ir a EchidnaBlocks (3):** volvemos a EchidnaBlocks para usar el modelo en nuestro programa.



Fases para crear un modelo

Fases LearningML



Para crear un modelo seguimos siempre estas fases:

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

Proyectos

1. **Asistente virtual**: un modelo de **texto** que entiende nuestras órdenes para encender y apagar la luz.
2. **Clasificador de residuos**: un modelo de **imágenes** que reconoce con la cámara a qué contenedor va cada residuo.
3. **Mando de inclinación**: un modelo de **números** que reconoce, con el acelerómetro, hacia dónde inclinamos la placa.

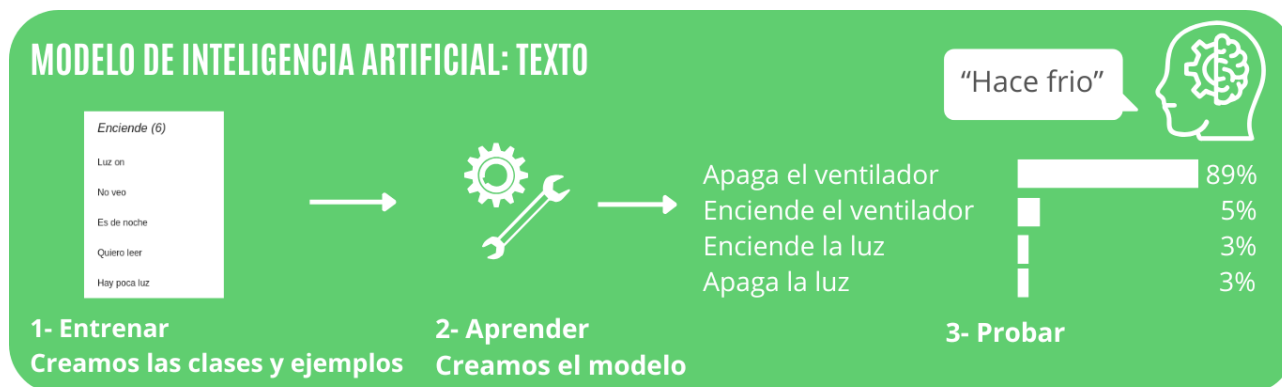
Estructura de los proyectos

Todos los proyectos con IA siguen la misma estructura:

1. **Qué vamos a hacer**: descripción del proyecto, qué vamos a aprender y qué vamos a usar.
2. **Entrenamos el modelo**: creamos las clases y los ejemplos, el modelo aprende y lo probamos.
3. **Programamos**: el programa de bloques que usa el modelo y cómo funciona.
4. **Mejóralo**: propuestas para ampliar el proyecto por tu cuenta.



3.1 Asistente virtual



1. Qué vamos a hacer

Vamos a crear un **asistente virtual** que entiende lo que le escribimos. Le pediremos con nuestras propias palabras que encienda o apague la luz, y el asistente encenderá o apagará el **LED RGB** de la placa EchidnaBlack2.

Para que el asistente entienda frases que nunca ha visto, crearemos con **LearningML** un **modelo de texto**: le enseñaremos ejemplos de frases para encender y para apagar la luz, y él aprenderá a reconocerlas.

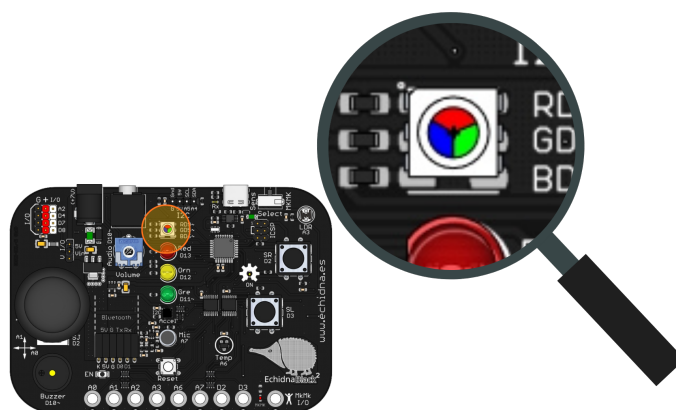
1.1 Qué vamos a aprender

- A crear un **modelo de machine learning** que clasifica **textos**.
- A organizar los datos en **clases** y a añadir **ejemplos** a cada una.
- A **entrenar, probar** y **mejorar** un modelo.
- A usar la **confianza** del modelo para decidir si nos fiamos de su respuesta.
- A usar el modelo en EchidnaBlocks para controlar la placa.

1.2 Qué vamos a usar

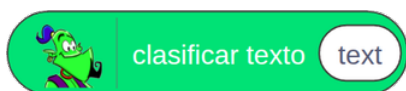
Componentes

Usaremos el **LED RGB** de la placa y el **teclado** del ordenador para escribir las órdenes.



Programación

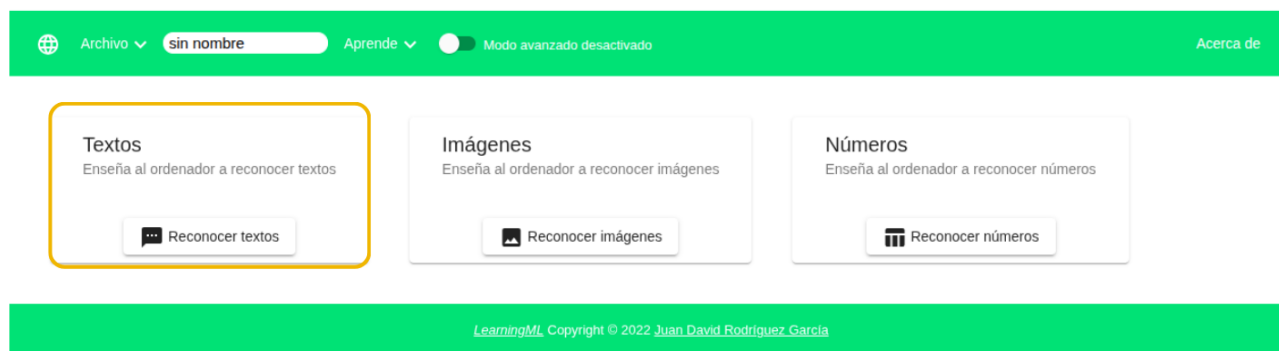
Para usar el modelo en nuestro programa usamos el bloque clasificar texto:



Le damos un **texto** y nos devuelve la **clase** que el modelo considera más probable; en nuestro caso, **Enciende** o **Apaga**.

2. Entrenamos el modelo

Abrimos LearningML y elegimos trabajar con datos de tipo **texto**.



2.1 Entrenar: clases y ejemplos

Creamos dos **clases** y añadimos a cada una ejemplos de frases:

- **Enciende:** frases que indican que queremos luz, por ejemplo "Luz on", "No veo", "Es de noche", "Quiero leer" o "Hay poca luz".
- **Apaga:** frases que indican que no la necesitamos, por ejemplo "Luz off", "Apaga", "Hay mucha luz", "Voy a dormir" o "Es de día".



Enciende (6)

Luz on

No veo

Es de noche

Quiero leer

Hay poca luz

Apaga (5)

Hay mucha luz

Luz off

Voy a dormir

Apaga

Es de día



La **calidad** del modelo depende de los **datos**: cuantos más ejemplos, y mejor elegidos, añadas a cada clase, mejor clasificará las frases nuevas.

2.2 Aprender

Pulsamos el botón **Aprender a reconocer textos**. El algoritmo analiza los ejemplos y construye un **modelo** capaz de clasificar textos que no ha visto antes. A esto lo llamamos **aprendizaje a partir de datos**.

2. Aprender

Llegó el momento de aprender a clasificar textos

Lenguaje de los textos Español

2.3 Probar

Escribimos frases **distintas** de los ejemplos y comprobamos en qué clase las coloca el modelo y con qué **confianza** (en %). Así comprobamos que el modelo ha aprendido y no solo ha memorizado los ejemplos.

3. Probar

Introduce términos nuevos y comprueba si se clasifican correctamente

Expresión

Me voy a la cama

Creo que pertenece a la clase Apaga, aunque no estoy muy segura

- Apaga (65.29 %)
- Enciende (34.71 %)

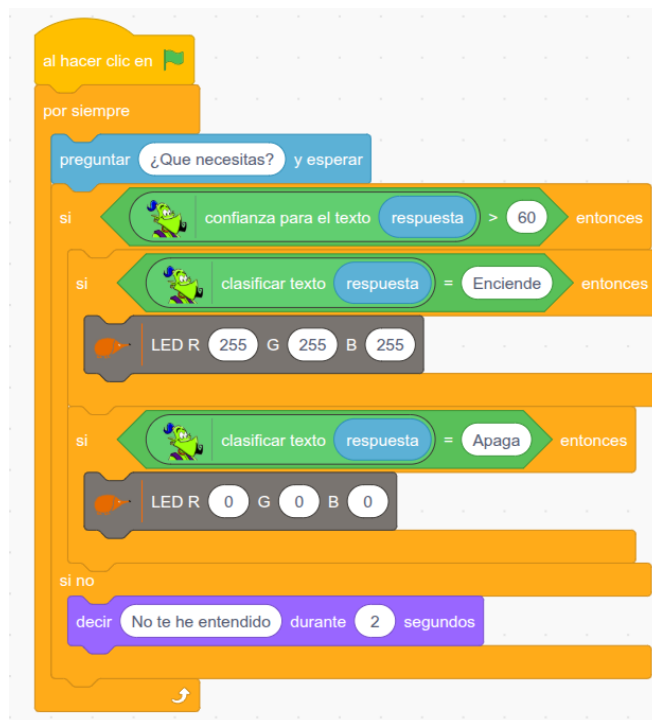


En el ejemplo, el modelo clasifica la frase "Me voy a la cama", que no está entre los ejemplos de entrenamiento, como **Apaga** con un 65 % de confianza.

Si el modelo se equivoca o la confianza es baja, volvemos a **2.1 Entrenar**: revisamos los ejemplos, añadimos otros nuevos y aprendemos de nuevo.

3. Programamos

Cuando el modelo funcione bien, abrimos **EchidnaBlocks** y construimos el programa con los bloques de LearningML y los de la placa:



Cómo funciona:

El echidna pregunta "¿Qué necesitas?" y espera nuestra respuesta. Antes de actuar, comprueba la **confianza** del modelo para esa respuesta: solo nos hace caso si es mayor de 60.

```

SI la confianza es mayor de 60:
  SI la clase es "Enciende":
    --> El LED RGB se enciende en blanco (R 255, G 255, B 255).
  SI la clase es "Apaga":
    --> El LED RGB se apaga (R 0, G 0, B 0).
SI NO:
  --> El echidna dice "No te he entendido".
  
```

Como todo está dentro de un bucle `por siempre`, el echidna vuelve a preguntar "¿Qué necesitas?", listo para una nueva orden.



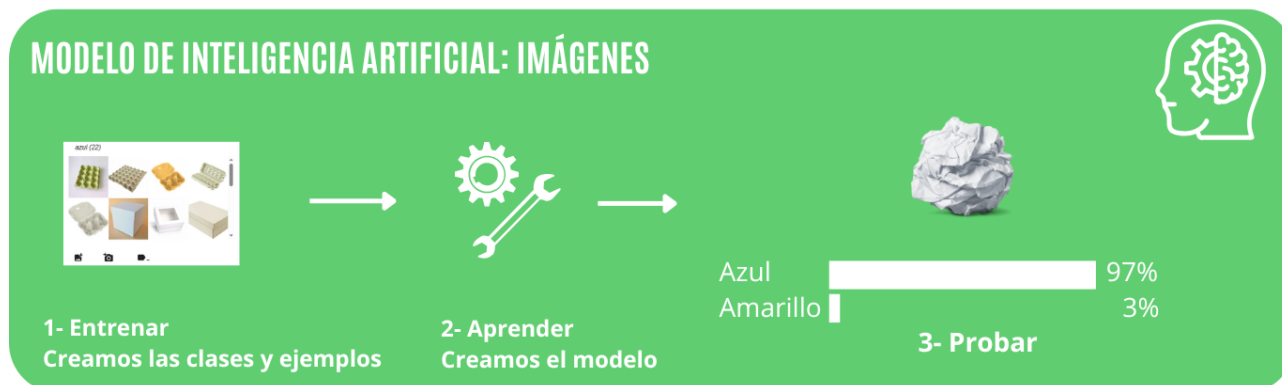
4. Mejóralo

Prueba a realizar algunas de estas mejoras en tu proyecto de forma autónoma:

1. **Más exigente:** Cambia el 60 de confianza por 80. ¿Qué ocurre con las frases que antes entendía con dudas?
2. **Entrévalo mejor:** Busca frases que el asistente no entienda o clasifique mal, añádelas como ejemplos a su clase y vuelve a entrenar el modelo. ¿Sube la confianza?
3. **Elige el color:** Añade las clases **Rojo**, **Verde** y **Azul** para que, además de encender y apagar, el asistente cambie el color del LED RGB según lo que le pidas.



3.2 Clasificador de residuos



1. Qué vamos a hacer

Vamos a construir un **clasificador de residuos** que nos ayuda a reciclar. Enseñamos un residuo a la **cámara** y el sistema decide si va al contenedor **amarillo** (envases) o al **azul** (papel y cartón). Después enciende el **LED RGB** del color del contenedor y **abre** su tapa con un **servomotor**.

Para que reconozca los residuos, crearemos con **LearningML** un **modelo de imágenes**: le enseñaremos fotos de envases y de cartones, y él aprenderá a distinguirlos.

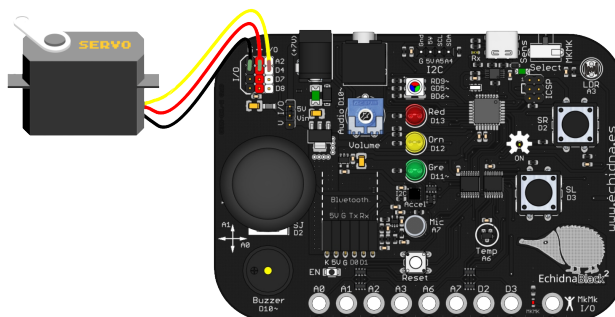
1.1 Qué vamos a aprender

- A crear un **modelo de machine learning** que clasifica **imágenes**.
- A preparar buenas **imágenes de entrenamiento** para cada **clase**.
- A usar la **cámara** (webcam) en EchidnaBlocks para clasificar lo que ve.
- A mover un **servomotor de posición** a un ángulo concreto.

1.2 Qué vamos a usar

Componentes

Usaremos la **cámara** del ordenador, el **LED RGB** de la placa y dos **servomotores de posición** conectados a los pines **D7** (contenedor azul) y **D8** (contenedor amarillo).

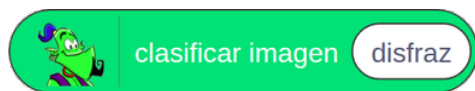




Presta atención al conectar los cables: Vcc, GND y señal se indican con los colores rojo, negro y amarillo. Como vamos a conectar **dos servomotores**, usa alimentación externa y coloca el selector de alimentación en la posición **Vin**.

Programación

Para usar el modelo en nuestro programa usamos el bloque clasificar imagen:



Le damos una **imagen** (en nuestro caso, la de la cámara, con el bloque imagen de vídeo) y nos devuelve la **clase** más probable: **azul** o **amarillo**.

2. Entrenamos el modelo

Abrimos LearningML y elegimos trabajar con datos de tipo **imágenes**.



2.1 Entrenar: clases y ejemplos

Creamos dos **clases**, **azul** y **amarillo**, y añadimos a cada una imágenes de residuos, desde un archivo o haciendo fotos con la cámara:

- **azul:** papel y cartón, por ejemplo hueveras y cajas de cartón.
- **amarillo:** envases, por ejemplo briks y latas.



Para que el modelo aprenda bien, añade **muchos residuos distintos** a cada clase (en el ejemplo, 22 imágenes por clase), vistos desde varias posiciones.



2.2 Aprender

Pulsamos el botón **Aprender a reconocer imágenes**. El algoritmo analiza las imágenes y construye un modelo capaz de reconocer imágenes parecidas, pero distintas. Con imágenes este proceso puede tardar **varios segundos**, y más cuantas más imágenes hayamos añadido.

2. Aprender

Llegó el momento de aprender a clasificar imágenes

 Aprender a reconocer imágenes

2.3 Probar

Enseñamos a la cámara residuos **nuevos**, que no hemos usado en el entrenamiento, y comprobamos en qué clase los coloca el modelo y con qué **confianza**.

3. Probar

Introduce términos nuevos y comprueba si se clasifican correctamente

imagen de prueba



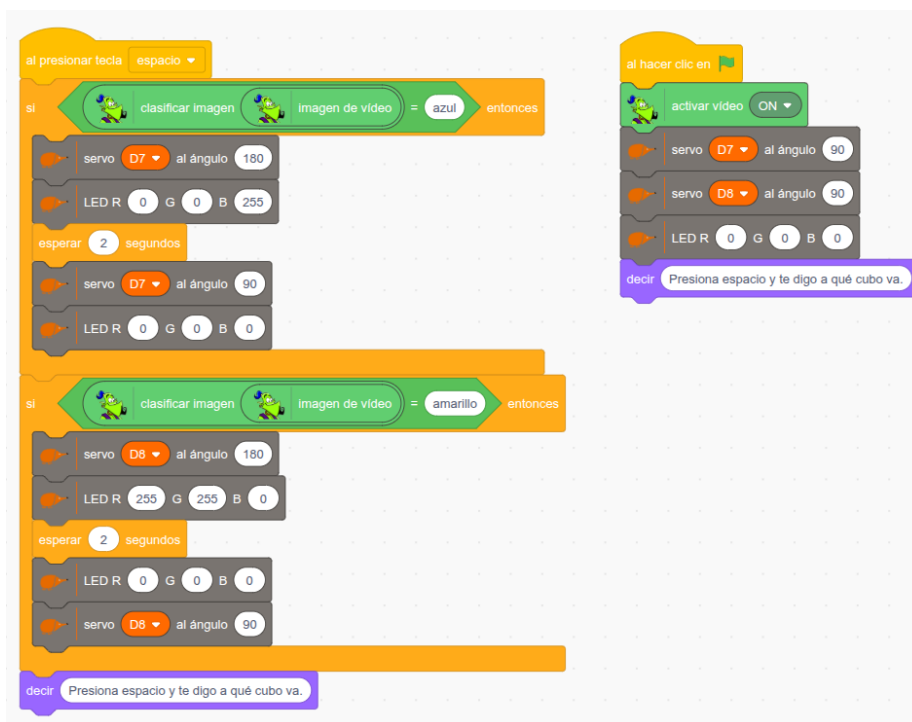
- amarillo (91.68 %)
- azul (8.32 %)

En el ejemplo, el modelo clasifica la imagen de una botella como **amarillo** con un 91,68 % de confianza.

Si el modelo se equivoca, volvemos a **2.1 Entrenar**: añadimos más imágenes o revisamos las que tenemos y aprendemos de nuevo.

3. Programamos

Cuando el modelo funcione bien, abrimos **EchidnaBlocks** y construimos el programa:



Cómo funciona:

Configuración inicial: al hacer clic en la bandera verde, activamos la cámara, colocamos los dos servos a 90° (tapas cerradas) y apagamos el LED RGB. El echidna dice "Presiona espacio y te digo a qué cubo va."

Al pulsar la tecla **espacio**, el programa clasifica la imagen de la cámara:

SI la clase es "azul":

- > El servo D7 se mueve a 180° (abre el contenedor azul).
- > El LED RGB se ilumina en azul (R 0, G 0, B 255).
- > Espera 2 segundos, cierra el contenedor (servo D7 a 90°) y apaga el LED.

SI la clase es "amarillo":

- > El servo D8 se mueve a 180° (abre el contenedor amarillo).
- > El LED RGB se ilumina en amarillo (R 255, G 255, B 0).
- > Espera 2 segundos, cierra el contenedor (servo D8 a 90°) y apaga el LED.

Después, el echidna vuelve a decir "Presiona espacio y te digo a qué cubo va."

4. Mejóralo

Prueba a realizar algunas de estas mejoras en tu proyecto de forma autónoma:

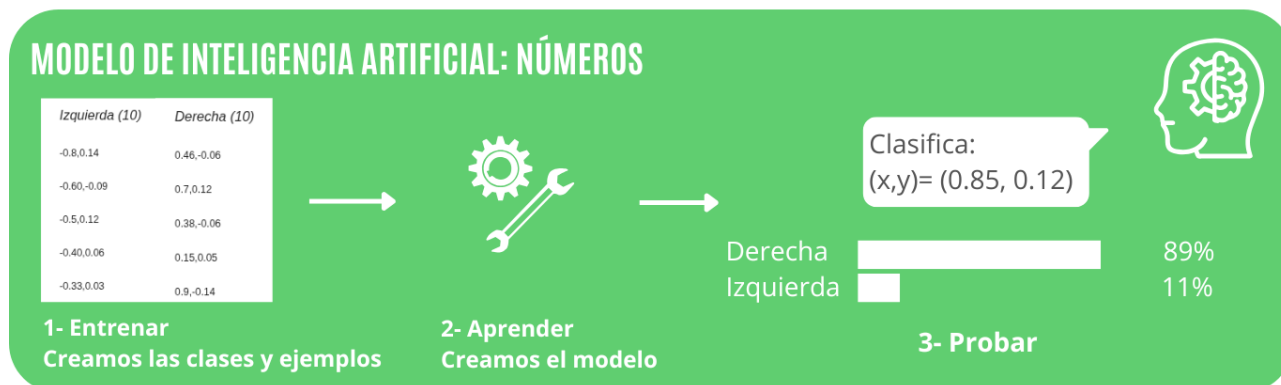
1. **Dímelo en pantalla:** Haz que el echidna diga a qué contenedor va el residuo ("¡Al amarillo!" o "¡Al azul!").



2. **Solo si está seguro:** Usa el bloque confianza para la imagen para que solo abra un contenedor si la confianza es mayor de 70; si no, el echidna dirá "No estoy seguro, prueba otra vez".
3. **Contenedor verde:** Añade la clase **verde** (vidrio) con imágenes de botellas y tarros, un tercer servomotor en el pin **D4** y el LED RGB en verde.



3.3 Mando de inclinación



1. Qué vamos a hacer

Vamos a convertir la placa EchidnaBlack2 en un **mando**: al inclinarla hacia la **derecha** o hacia la **izquierda** y pulsar **SL**, el echidna se desplazará por el escenario en esa dirección.

Para saber hacia dónde inclinamos la placa, crearemos con **LearningML** un **modelo de números**: le enseñaremos los valores que da el **acelerómetro** en cada posición, y él aprenderá a reconocerlas.

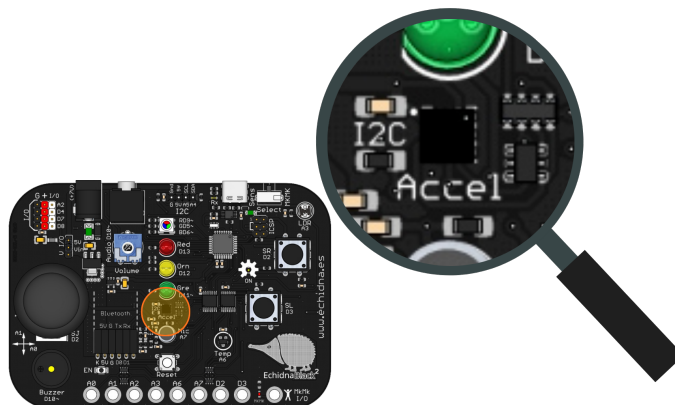
1.1 Qué vamos a aprender

- A crear un **modelo de machine learning** que clasifica **números**.
- A recoger **datos** de un sensor para entrenar un modelo.
- A introducir ejemplos con **varias columnas** (en nuestro caso, los ejes X e Y del acelerómetro).
- A usar el modelo en EchidnaBlocks para controlar un personaje.

1.2 Qué vamos a usar

Componentes

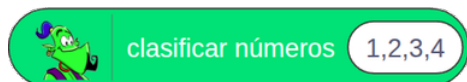
Usaremos el **acelerómetro** de la placa, que mide su inclinación, y el pulsador **SL**.





Programación

Para usar el modelo en nuestro programa usamos el bloque clasificar números:



Le damos un **conjunto de números separados por comas** y nos devuelve la **clase** más probable; en nuestro caso, **Derecha** o **Izquierda**.

2. Entrenamos el modelo

Abrimos LearningML y elegimos trabajar con datos de tipo **números**.



2.1 Entrenar: clases y ejemplos

Primero elegimos el **número de columnas** de cada ejemplo. Elegimos **2**, porque cada ejemplo tendrá los valores del acelerómetro en los ejes **X** e **Y**.

Creamos dos **clases**, **Derecha** e **Izquierda**, y añadimos a cada una ejemplos. Cada ejemplo es una pareja de números separados por una **coma**: primero el eje X y después el eje Y.

<i>Izquierda (10)</i>	<i>Derecha (10)</i>
-0.8,0.14	0.46,-0.06
-0.60,-0.09	0.7,0.12
-0.5,0.12	0.38,-0.06
-0.40,0.06	0.15,0.05
-0.33,0.03	0.9,-0.14



Para conseguir los ejemplos, inclina la placa y anota los valores del acelerómetro. Puedes verlos en el escenario marcando la casilla que hay junto al bloque `Leer acelerómetro`. Al inclinar hacia la izquierda, el eje X da valores **negativos** (hasta -1), y hacia la derecha, **positivos** (hasta 1). Recoge ejemplos con inclinaciones **suaves** y **fuertes**.

2.2 Aprender

Pulsamos el botón **Aprender a reconocer números**. El algoritmo analiza los ejemplos y construye un modelo capaz de clasificar números nuevos.

2. Aprender

Llegó el momento de aprender a clasificar números



Aprender a reconocer números

2.3 Probar

Escribimos parejas de números **nuevas**, parecidas a los ejemplos pero distintas, y comprobamos en qué clase las coloca el modelo y con qué **confianza**.

3. Probar

Introduce términos nuevos y comprueba si se clasifican correctamente

Números

0.3,0.05

Comprobar

Probablemente pertenezca a la clase Derecha

- Derecha (82.08 %)
- Izquierda (17.92 %)

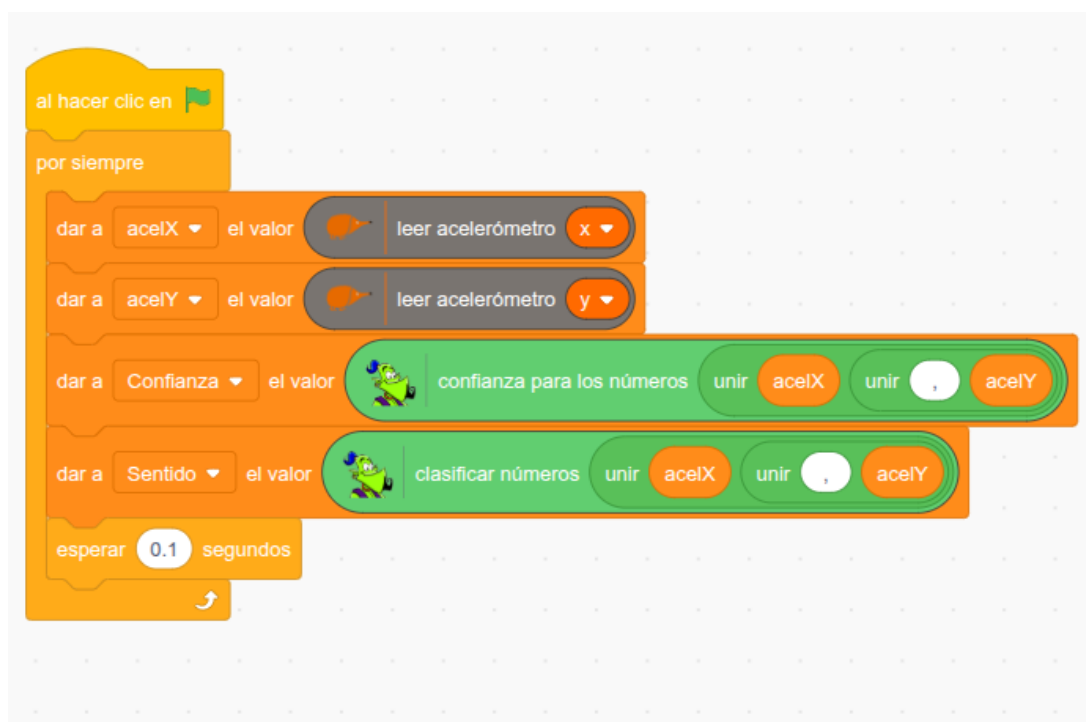
En el ejemplo, el modelo clasifica 0.3, 0.05 como **Derecha** con un 82,08 % de confianza.

Si el modelo se equivoca, volvemos a **2.1 Entrenar**: añadimos más ejemplos o revisamos los que tenemos y aprendemos de nuevo.

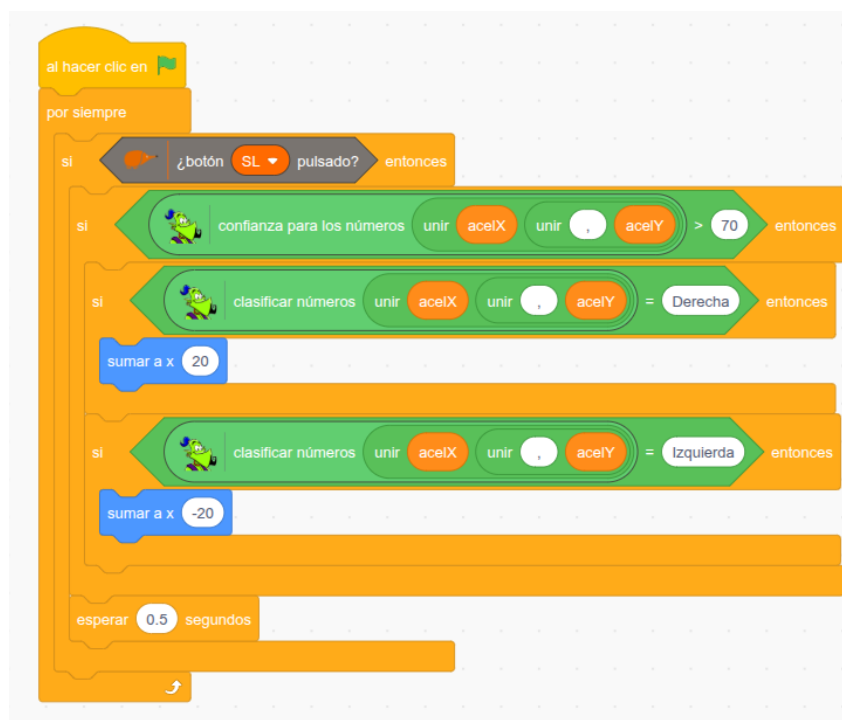
3. Programamos

Cuando el modelo funcione bien, abrimos **EchidnaBlocks** y construimos el programa, que tiene dos scripts.

El primero lee continuamente el **acelerómetro** y guarda los ejes X e Y en las variables `aceX` y `aceY`. También guarda en `Confianza` y `Sentido` la confianza y la clase que da el modelo, para verlas en el escenario mientras inclinamos la placa:



El segundo mueve el echidna según la clase:



Cómo funciona:

El programa revisa continuamente el pulsador **SL**. Cuando lo pulsamos, une los valores `acelX` y `acelY` del acelerómetro con una coma (bloque `unir`) y se los da al modelo para que los clasifique. Antes de mover el echidna, comprueba la **confianza**:



SI está pulsado SL:

SI la confianza es suficiente:

SI la clase es "Derecha":

--> El echidna se desplaza 20 pasos a la derecha (sumar a x 20).

SI la clase es "Izquierda":

--> El echidna se desplaza 20 pasos a la izquierda (sumar a x -20).

--> Espera 0.5 segundos.

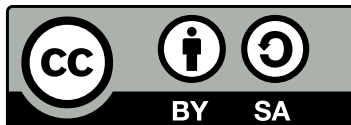
4. Mejóralo

Prueba a realizar algunas de estas mejoras en tu proyecto de forma autónoma:

1. **Pasos de gigante:** Cambia los 20 pasos y el tiempo de espera para que el echidna se mueva más rápido o más despacio.
2. **Placa quieta:** Añade la clase **Quieta** con ejemplos de la placa en horizontal, para que el echidna no se mueva cuando no la inclinas.
3. **Cuatro direcciones:** Añade las clases **Adelante** y **Atrás** (inclinando la placa en el eje Y) para que el echidna también se mueva hacia arriba y hacia abajo.



4. Licencia



Proyectos de inicio con EchidnaML: proyectos sencillos para iniciarse con **EchidnaML** y la placa **EchidnaBlack2**.

Autoría: Echidna Educación.

Este recurso se publica bajo la licencia [Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional \(CC BY-SA 4.0\)](https://creativecommons.org/licenses/by-sa/4.0/). Esto significa que puedes:

- **Compartir:** copiar y redistribuir el material en cualquier medio o formato.
- **Adaptar:** remezclar, transformar y crear a partir del material con cualquier finalidad, incluso comercial.

Siempre que cumplas estas condiciones:

- **Reconocimiento:** debes citar la autoría (Echidna Educación), enlazar la licencia e indicar si has hecho cambios.
- **Compartir igual:** si remezclas, transformas o creas a partir del material, debes difundir tu obra con la misma licencia.