# 2 ARDUINOBLOCKS

# Introducción

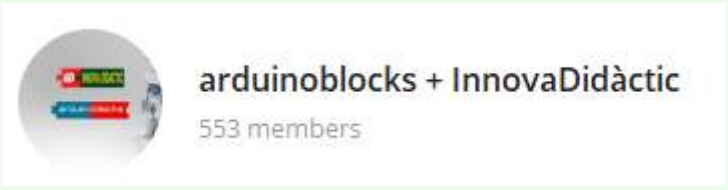
Esto no pretende ser un tutorial exhaustivo de STEAMAKERSBLOCKS, sino una guía rápida. STEAMAKERSBLOCKS es un programa que tiene muchas posibilidades. Si quieres saber más sobre ARDUINOBLOCKS tutoriales, ejemplos, foro.... te recomendamos

http://arduinoblocks.didactronica.com/ o el libro Arduino blocks - libros y tutoriales

**IMPORTENTE SABERLO: TENEMOS UN CHAT DE Ayuda en STEMAKERSBLOCKS**

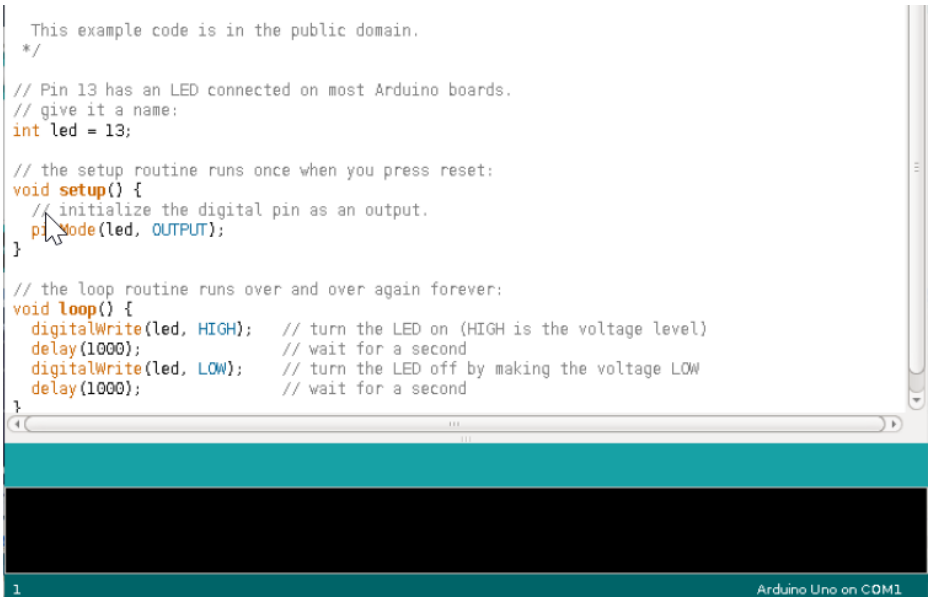
hay chat de **Telegram** con una comunidad de profesores y técnicos de la empresa que apoya Arduinoblocks donde puedes encontrar proyectos, enlaces interesantes y lo más importante:  Puedes preguntar tus dudas o problemas

https://t.me/innovadidactic_comunidad




## ¿Por qué una programación con bloques?

Arduino se programa en lenguaje C++ (con algunas variaciones para simplificarlo). Para programar normalmente se utiliza el IDE ("Integrated Development Environment"/"Entorno de Desarrollo Integrado") de Arduino, que permite escribir el código, compilar el programa (crear el programa binario para el procesador Arduino) y grabarlo en la placa Arduino a través del puerto USB. El IDE de Arduino se puede descargar desde la web oficial. Es totalmente libre *(José Andrés Echevarría @cantabRobots CC-BY-NC-SA)*



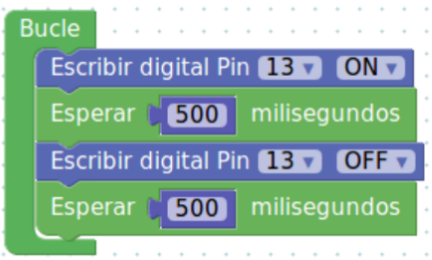

*(José Andrés Echevarría @cantabRobots CC-BY-NC-SA)*

Sin embargo pensando en edades más tempranas se han desarrollado formas más sencillas e intuitivas de programar Arduino como son los **lenguajes de programación por bloques**. De todos estos lenguajes cabe destacar **STEAMAKERSBLOCKS**.

Gracias a este lenguaje visual podemos programar las placas Arduino sin necesidad de escribir ni una sola línea de código, de esta forma podemos empezar a realizar proyectos con Arduino de una forma muy rápida y a edades más tempranas.  La única desventaja es que el lenguaje por código tiene todo el potencial que requiere la programación de un experto.

Mismo programa creado con el IDE de Arduino en C++ (imagen de la izquierda) y con Arduinoblocks (imagen de la derecha).
*(José Andrés Echevarría @cantabRobots CC-BY-NC-SA)*

```
void setup()
{
pinMode(13, OUTPUT);
}
void loop()
{
digitalWrite(13, HIGH);
delay(500);
digitalWrite(13, LOW);
delay(500);
}
```




*José Andrés Echevarría @cantabRobots CC-BY-NC-SA)*

Para trabajar con Arduinoblocks debemos ir a su página web http://www.arduinoblocks.com/ desde cualquier navegador y para cualquier sistema operativo (Windows, Linux, Mac). *(José Andrés Echevarría @cantabRobots CC-BY-NC-SA)*

## STEAMAKERSBLOCKS (antes Arduinoblocks)

steamakersblocks antes ArduinoBlocks

**STEMAKERSBLOCKS** es un programa creado por el profesor Juanjo López. Gracias a su entorno gráfico facilita la programación de placas Arduino a todos los niveles. Esta herramienta permite programar a personas sin conocimientos previos de programación, pero su versatilidad y potencia es tan grande que expertos programadores también pueden utilizarlo. *(José Andrés Echevarría @cantabRobots CC-BY-NC-SA)*

De Juan José López Almendros CC-BY-SA

La programación en **STEMAKERSBLOCKS** se realiza con bloques al estilo AppInventor o Scratch, *se puede utilizar a partir de 8 años*. No tenemos que escribir líneas de código y no nos permitirá unir bloques incompatibles evitando así posibles errores de sintaxis. La plataforma **STEMAKERSBLOCKS** genera, compila y sube el programa a la placa Arduino por medio de la conexión USB. Una vez subido el programa, la placa el Arduino no necesitará de la conexión al PC para funcionar pudiendo alimentarla con baterías o una fuente de alimentación para que funcione de forma autónoma.

**STEMAKERSBLOCKS** actualmente funciona con todos los navegadores de última generación: Mozilla Firefox, Google Chrome, Opera, Safari,... *(José Andrés Echevarría @cantabRobots CC-BY-NC-SA)*

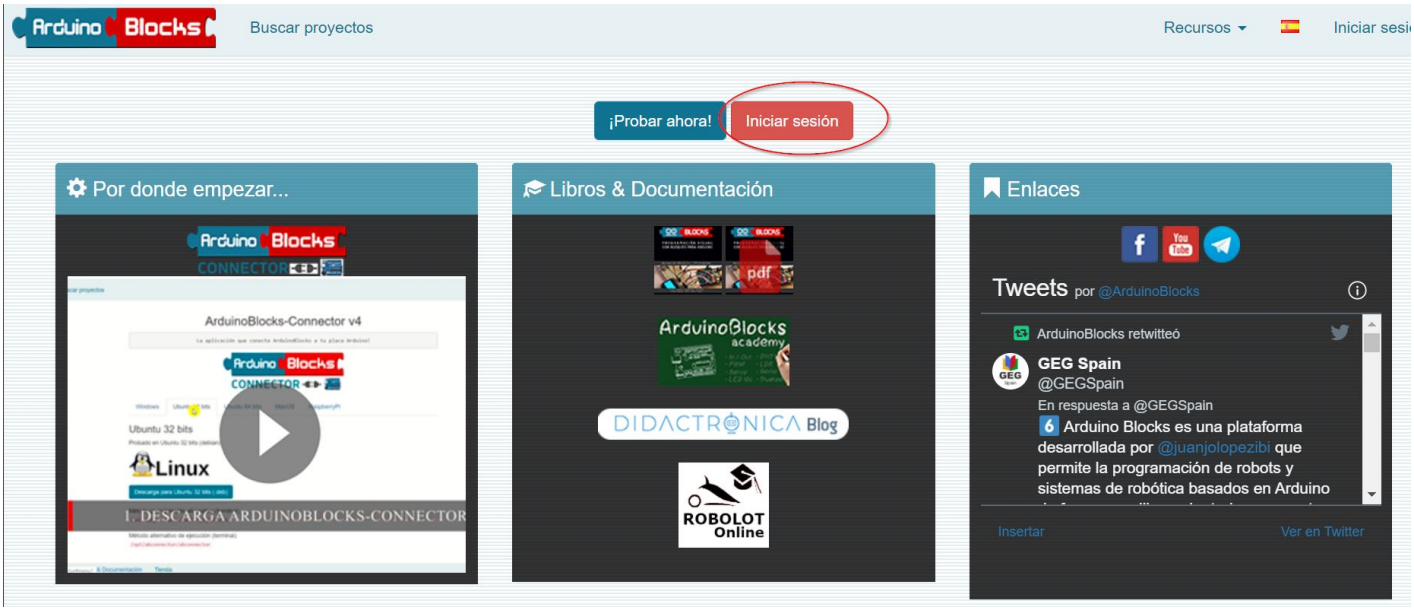
Por otro lado, tal y como se describe en la Wiki de **Vitalinux**, **STEMAKERSBLOCKS** funciona perfectamente con todos los sistemas operativos, pudiendo ser fácilmente instalable en equipos individuales y a nivel de centro dentro del soporte de **Vitalinux**.

# Crear cuenta

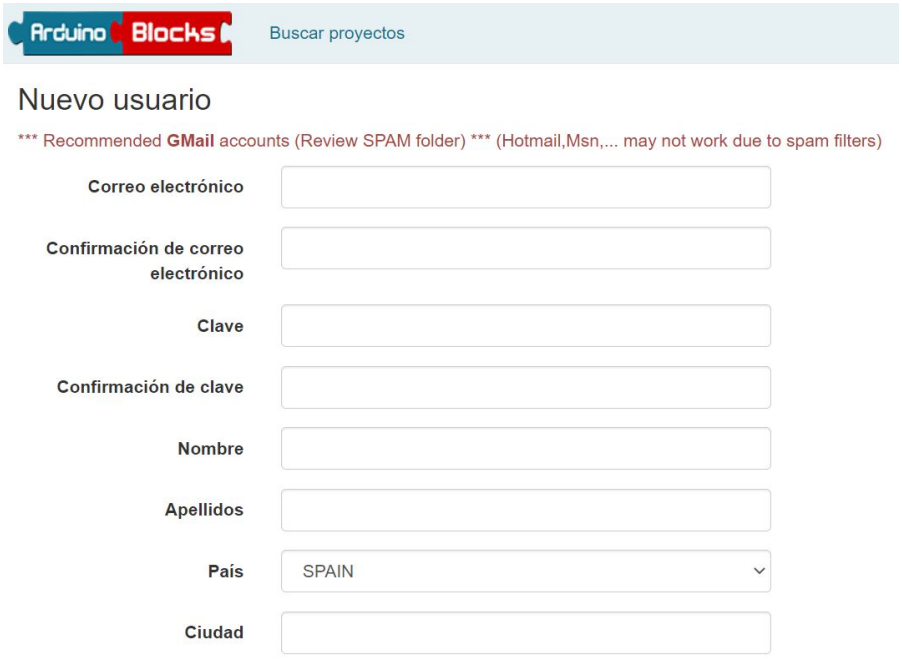
Registrándonos como usuarios de la plataforma **Steamakerblocks** podemos aprovechar todas estas posibilidades:

- Guardar tus proyectos en la nube de **Steamakerblocks** .
- Añadir información al proyecto: descripción, componentes utilizados, imágenes, etc.
- Añadir archivos adjuntos relacionados con el proyecto: esquemas, fotos, archivos para impresión 3D, aplicaciones, etc.
- Compartir proyectos con el resto del mundo.
- Importar proyectos compartidos por otros usuarios.
- Valorar y comentar proyectos.
- Programar directamente Arduino desde el propio navegador (con la aplicación: **Steamakerblocks -Connector**).
- Utilizar la consola serie desde el propio navegador.

Entramos en https://www.steamakersblocks.com/ e iniciamos sesión




Y rellenamos el formulario




*Autor José Andrés Echevarría @cantabRobots CC-BY-NC-SA*

# Cuentas alumnado

Tal y como dice el tutorial de Juanjo López : *Permite a un usuario registrado con email, crear y administrar nuevas cuentas de usuario dentro de una organización, centro educativo o institución.*

- ❤Permite crear usuarios alumnos sin necesidad de ceder ningún tipo de datos
- □□Puedes pasar proyectos a los alumnos, vacíos o empezados
- □□Tú puedes ver los proyectos de los alumnos y ponerles comentarios

https://www.youtube.com/embed/zdzKX0NX60Y

- Si lo quieres en papel, te recomendamos el tutorial de Juanjo López son 12 diapositivas muy bien explicados
  - Repositorio Github
  - Repositorio en steamakerbloks.com
- Prueba con Catedu:
  - https://www.steamakersblocks.com/
  - usuario alumnox.catedu donde x de 1 a 20
  - contraseña [donde esta catedu en minúsculas]

# ArduinoBlocks connector

> **Espera !!!** Aún no conectes tu placa (ESP32, Arduino, TDR STEAM...)

## PRIMER PASO Descargar e instalar Connector

Para poder usar la herramienta **Steamakerblocks** tenemos que ejecutar antes **Connector**. Lo descargamos de la misma página de **Steamakerblocks** según el sistema operativo que usemos: Windows (**W7 E INFERIORES NO FUNCIONA**), Linux ....

Entra en https://www.steamakersblocks.com/ y en Recursos, tienes la web para descargar este programa:


Lo descargamos y lo **instalamos.**

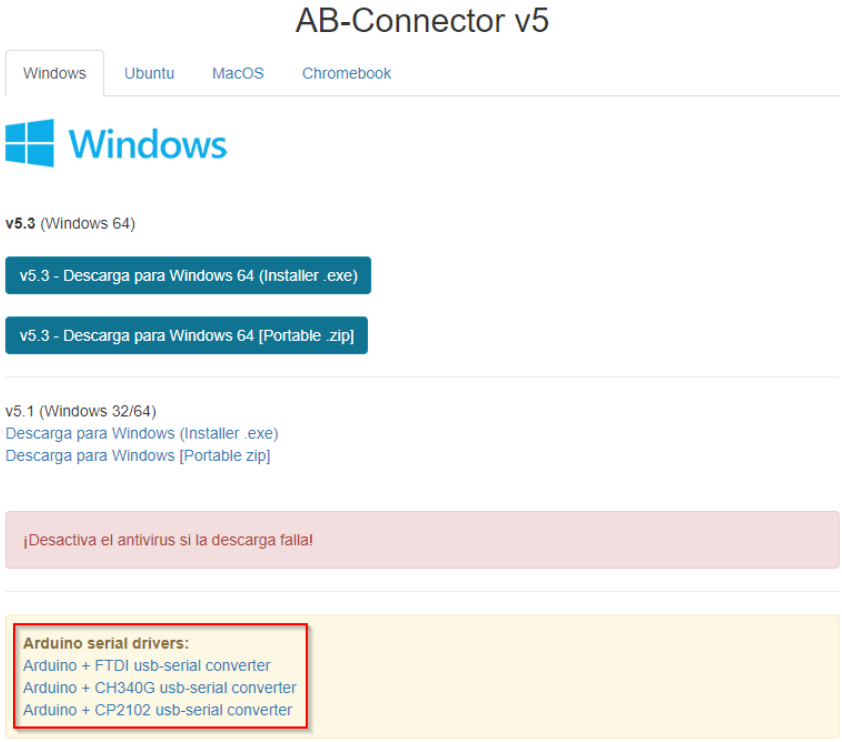
En el caso de tener equipos Vitalinux, es fácilmente accesible e instalable desde la aplicación **Vitalinux Play** o si se desea una instalación masiva en el centro a través de su página de soporte:


## SEGUNDO PASO: INSTALAR LOS DRIVERS

> Si no hacemos estos pasos, cuando conectamos la placa, siempre sale en el COM1, le damos a subir y sale error

En http://www.arduinoblocks.com/web/site/abconnector5 tenemos abajo ARDUINO SERIAL DRIVERS

> RECOMENDAMOS EL PRIMER ENLACE Y EL TERCERO



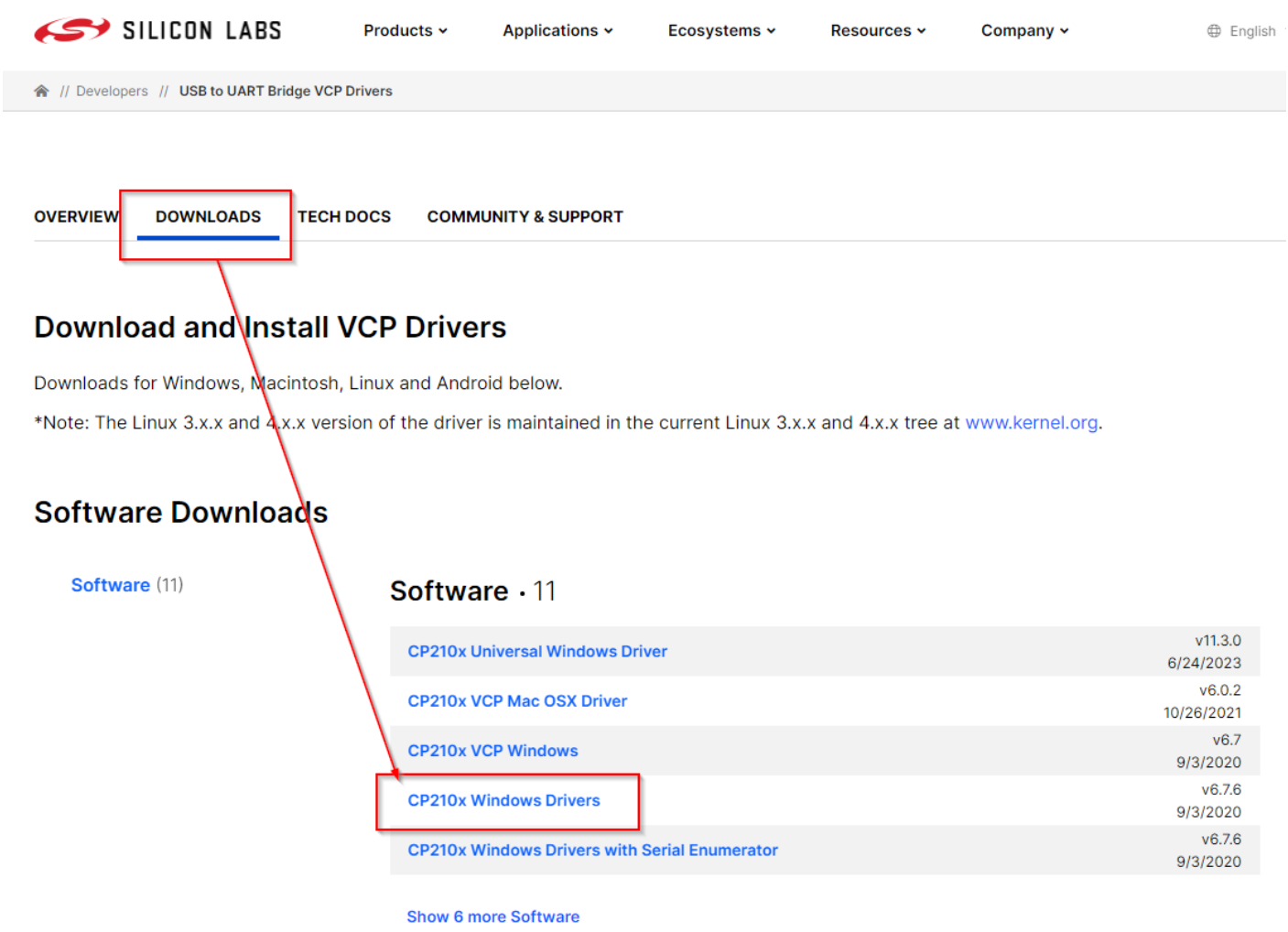

**En el primero** el instalador está en este enlace

https://cdn.sparkfun.com/assets/learn_tutorials/7/4/CDM21228_Setup.exe

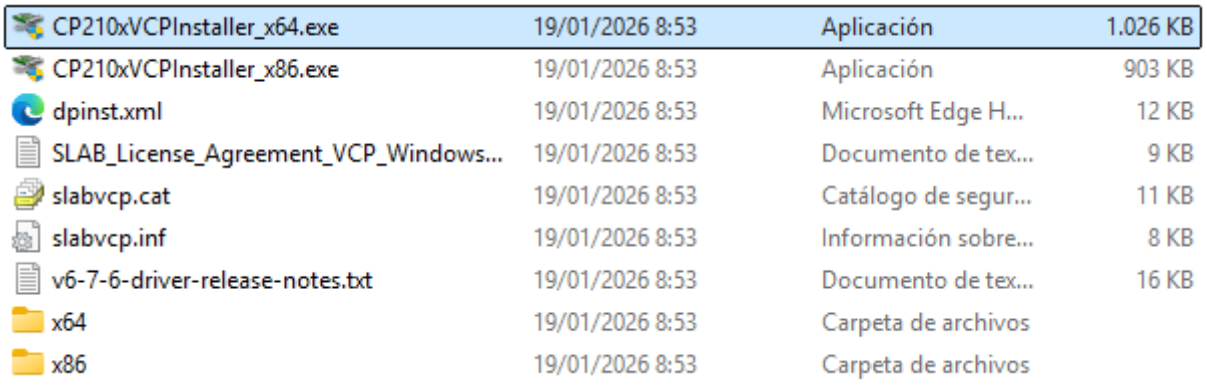

**El segundo** sólo si quieres utilizar Arduinos no oficiales, de fabricantes chinos, que tiene el CH340g y hay que leerse la página, paciencia

**El tercero** es necesario el 2102 si utilizas el ESP32 el instalador esta en este enlace, es una carpeta comprimida, la descomprimes y está el ejecutable instalador

https://www.silabs.com/documents/public/software/CP210x_Windows_Drivers.zip

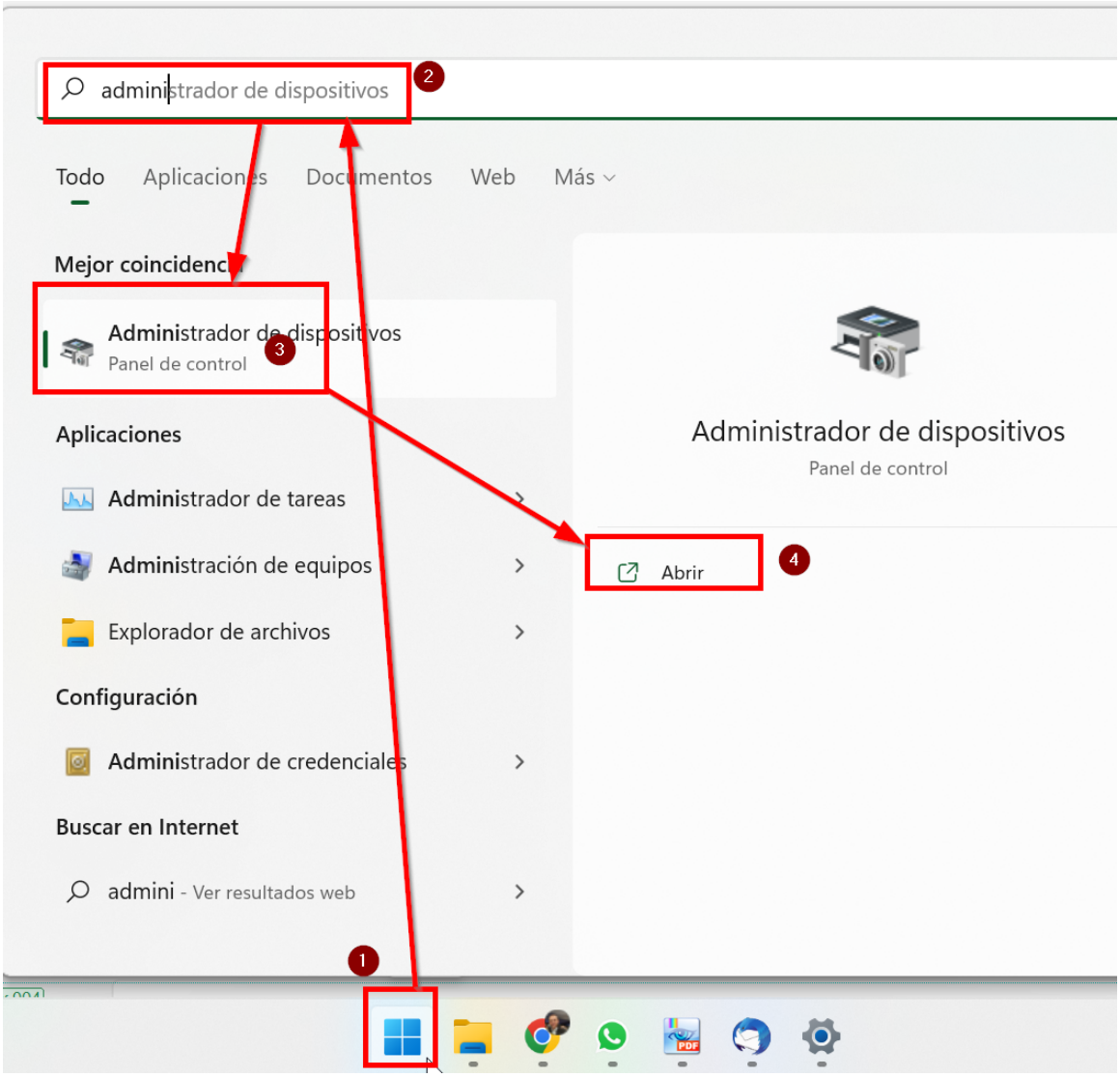



concretamente hay que ejecutar este (al menos que el equipo sea muy viejo de 32bits)

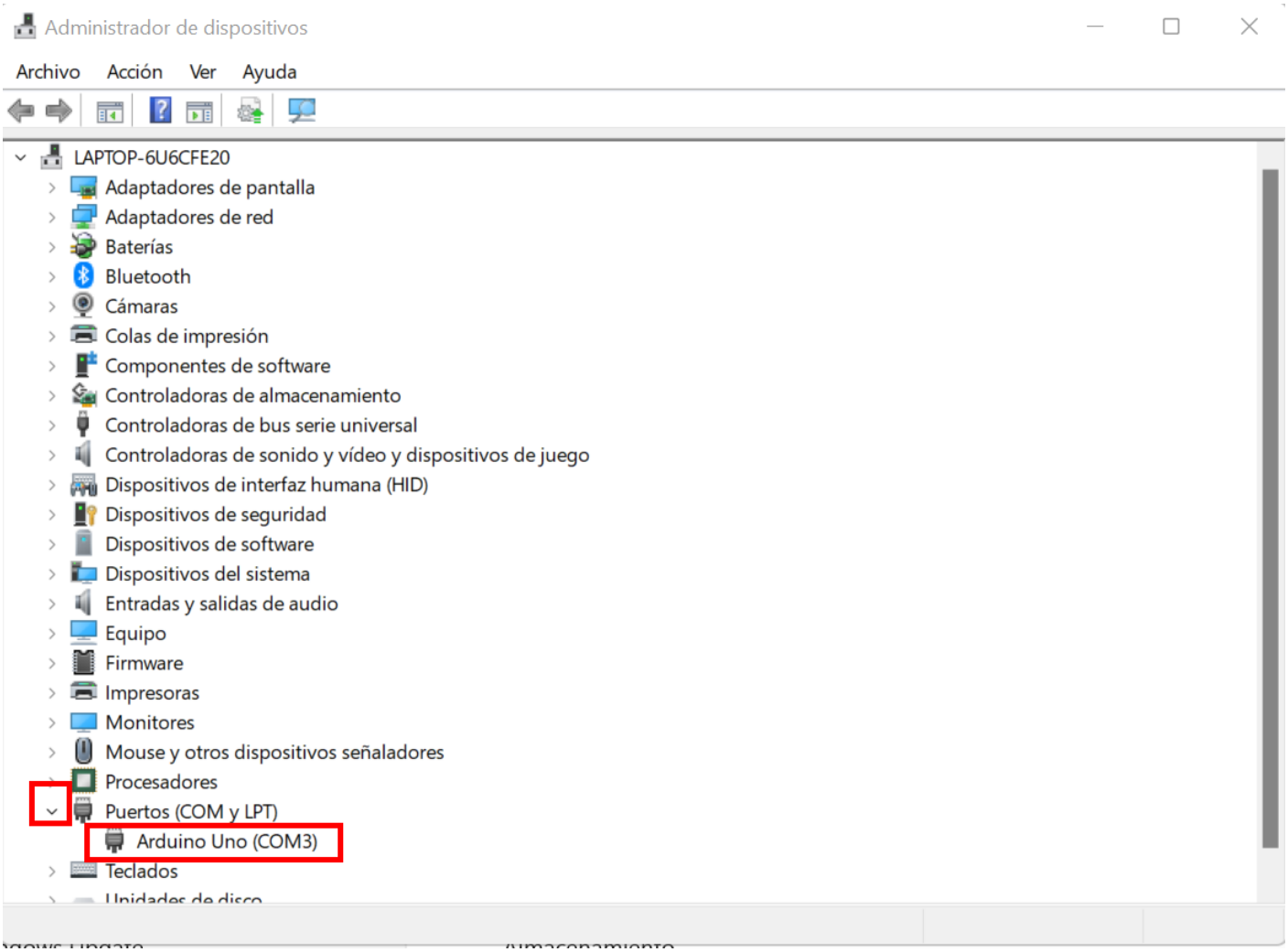



## COMPROBAR QUE DETECTA LA PLACA

Ahora **conectamos la placa** (ESP32, Arduino, NodeMCU, KeyStudio TDR STEAM...) a nuestro ordenador, y observamos si lo detecta, en Windows entramos en Administrador de dispositivos:




Y vemos que en los puertos COM se ha detectado correctamente la placa:




> En el caso de que no aparezca, es que no se han instalado correctamente los *drivers* de Arduino. Entonces vamos a la página oficial de Arduino y descargamos el programa
>
> **ARDUINO IDE** : https://www.arduino.cc/en/software y lo instalamos. Al instalar este programa se instalan los drivers en nuestro ordenador. No hace falta ejecutarlo.

En el caso de equipos con sistema operativo Linux (como Vitalinux) el puerto serie tiene la forma **/dev/XXXX**

## YA PUEDES EJECUTAR ARDUINOBLOCKS CONNECTOR

Ahora buscamos el programa ArduinoBlocks connector que hemos descargado e instalado en el primer paso y lo **ejecutamos**.


```
12:36:51> AB-Connector v5.3
12:36:51> Path: C:\Program Files\abconnector\bin
12:36:51> Port: 9987
12:36:51> Arduino-CLI: 0.35.3
12:36:51> ['arduino:avr', 'esp32:esp32', 'esp8266:esp8266']
12:36:51> Checking/updating libs...
12:36:51> Libraries version: 56
```

# **ATENCIÓN** No podemos cerrar la ventana mientras utilizamos *Arduinoblocks*, la minimizamos simplemente.

En caso contrario, Arduinoblocks no se puede comunicar con nuestra placa Arduino, NodeMCU, KeyStudio, etc ....

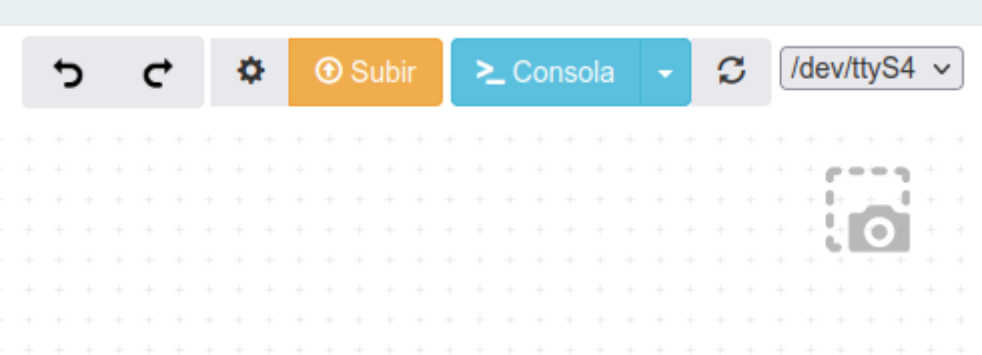
## YA PUEDES EJECUTAR ARDUINOBLOCKS

Entramos en la web ARDUINOBLOCKS http://www.arduinoblocks.com/ nos logueamos e iniciciamos un proyecto, Vemos que en el editor que aparece ya los puertos COM (si no te aparece, dale a la rueda actualizar)

> Aparecen varios COM, **elegir el último que tiene que coincidir con el que has visto en el segundo paso**, no necesariamente es el COM más alto.
> Si se queda una ruleta de espera demasiado tiempo, entonces, actualizar la página o darle a actualizar el botón 1 de la figura :

Una vez elegido el COM ya puedes darle al botón amarillo **SUBIR** cuando has realizado tu proyecto pero antes de subir, por si acaso dale a **guardar** el proyecto que has realizado.

En el caso de equipos con Linux veremos algo así:

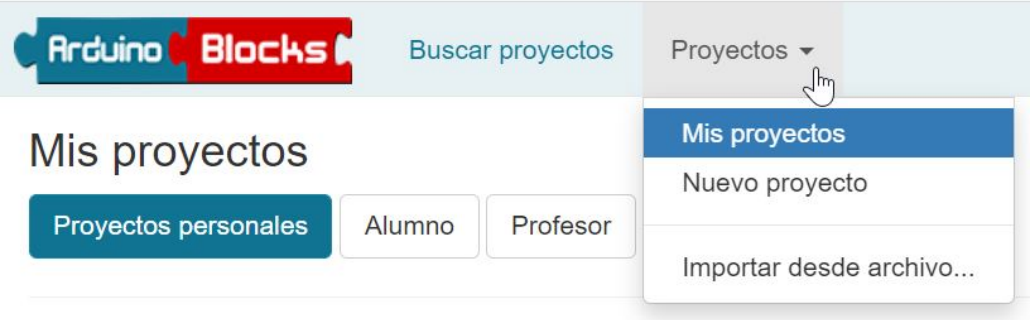



**¿Tengo que hacer los cuatro pasos cada vez?**
No, sólo la primera vez para asegurar los drivers del Arduino, las siguientes veces que te conectes lo único que tienes que hacer  es el tercer y cuarto paso

**IMPORTANTE: TENER EL SOFTWARE ARDUINOBLOCKS ACTUALIZADO** para que funcionen los nuevos bloques que se incorporan en Arudinoblocks

# Empezando

Entramos en Proyectos y podemos ver nuestros proyectos creados como también empezar uno.




Y nos aparece tres opciones :

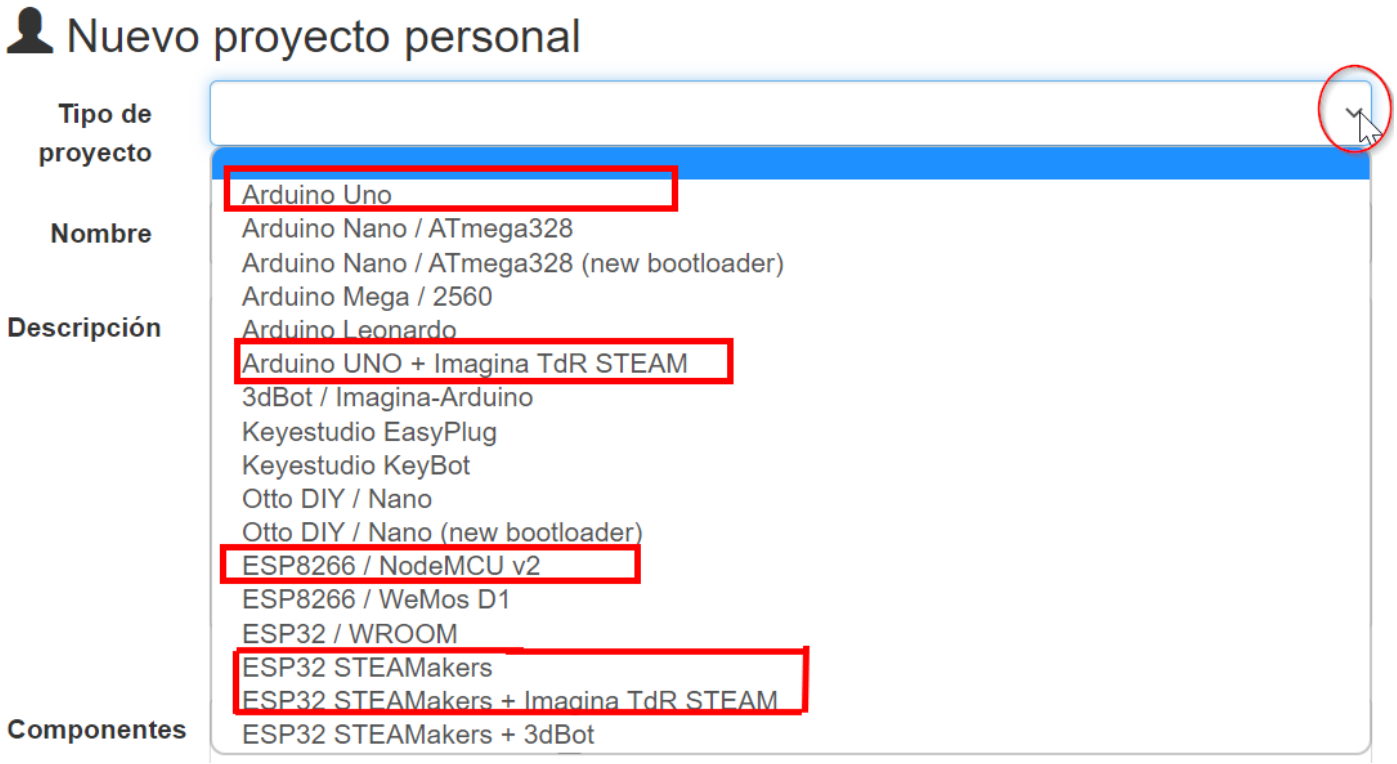

En esta ventana podremos elegir que tipo de proyecto vamos a realizar:

- **Proyecto Personal**: Iniciar un nuevo proyecto que sólo será accesible para el usuario. Posteriormente se puede compartir al resto de la comunidad si se desea.
- **Proyecto Profesor**: Iniciar un proyecto como profesor. De esta forma no se inicia un proyecto como tal, sino que se especifican los datos del proyecto y se genera un código para que los alumnos se puedan suscribir al proyecto. El profesor podrá supervisar y valorar los proyectos de sus alumnos.
- **Alumno**: De esta forma nos unimos a un proyecto planteado por el profesor. Nosotros realizaremos el proyecto como si de un proyecto personal se tratara, pero el profesor podrá supervisar y valorar nuestro trabajo.

> *Adaptado de este enlace. José Andrés Echevarría @cantabRobots CC-BY-NC-SA*

Lo primero que tenemos que elegir es para qué tipo de placa se hace el proyecto.

- En el caso de que estés con el kit de CATEDU **Rover marciano con Arduinoblocks** el tipo de proyecto es para **ESP8266 / NodeMCU**
- En el caso de que estés con el kit de CATEDU **Arduino con Arduinoblocks** el tipo de proyecto es para **Arduino UNO**
- En el caso de que estés con el kit de CATEDU **STEAMAKERSBLOCKS en el aula** tienes dos opciones totalmente válidas:
  - **ArduinoUno**
  - **ArduinoUno + Imagina TdR STEAM**
- En el caso de que estés con el kit de CATEDU **ESP32 en el aula** tienes dos opciones totalmente válidas:
  - **ESP32 STEAMakers**
  - **ESP32 STEMakers + Imagina TdR STEAM**
- En el caso de que estés con el kit de CATEDU **SMARTHOME ESP32** tienes que elegir:
  - **ESP32 STEAMakers**
-



> **ATENCIÓN** luego **NO** se puede cambiar. Es decir, un proyecto realizado para un tipo de placa, no se puede cambiar a otro tipo de placa (la razón es simpe: las instrucciones cambian)

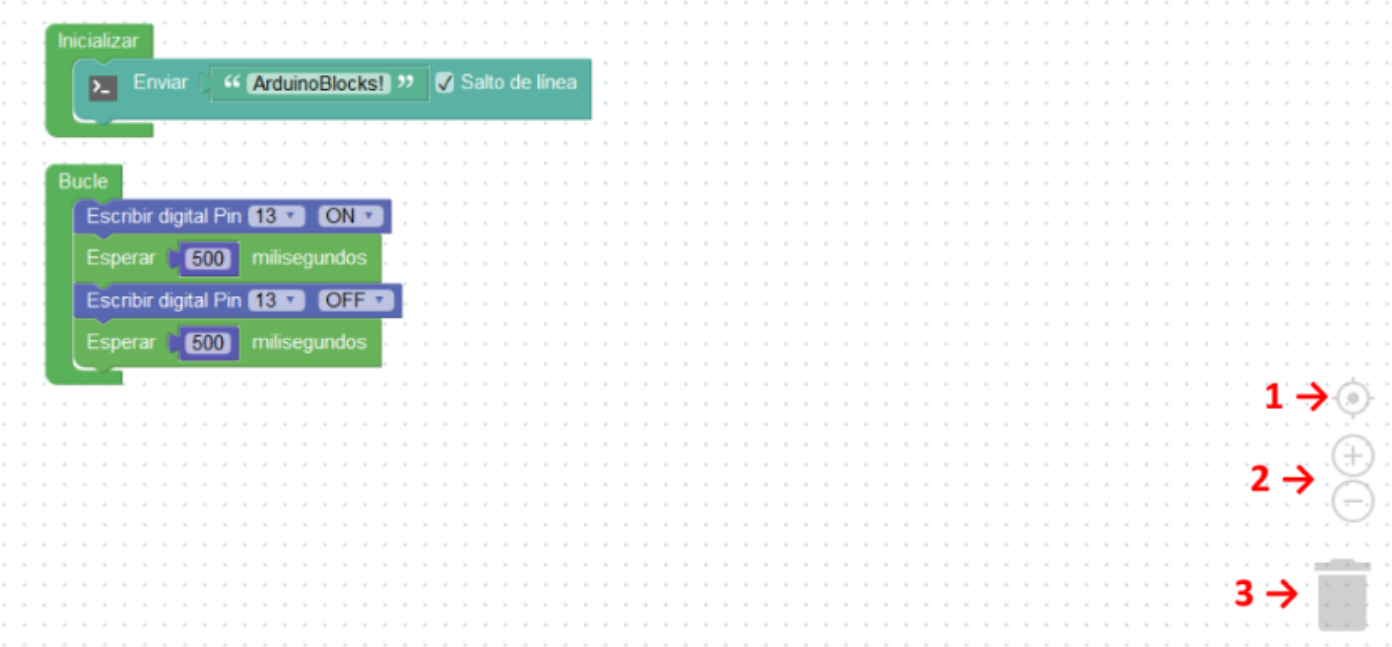
Luego el nombre y el resto de campos es optativo pero **importante y buena costumbre** rellenarlos, sobre todo si el proyecto lo **compartimos**:

- Descripción
- Componentes
- Comentarios

## Área de programación del proyecto

Este es el área sobre el que se trabaja en Arduinoblocks. En esta área arrastraremos y colocaremos los bloques que vamos a utilizar para crear nuestro programa.

En el área de trabajo hay un Zoom (2) para ampliar o reducir la imagen, un icono para centrar (1) y un icono donde podremos borrar los bloques que no utilicemos (3).



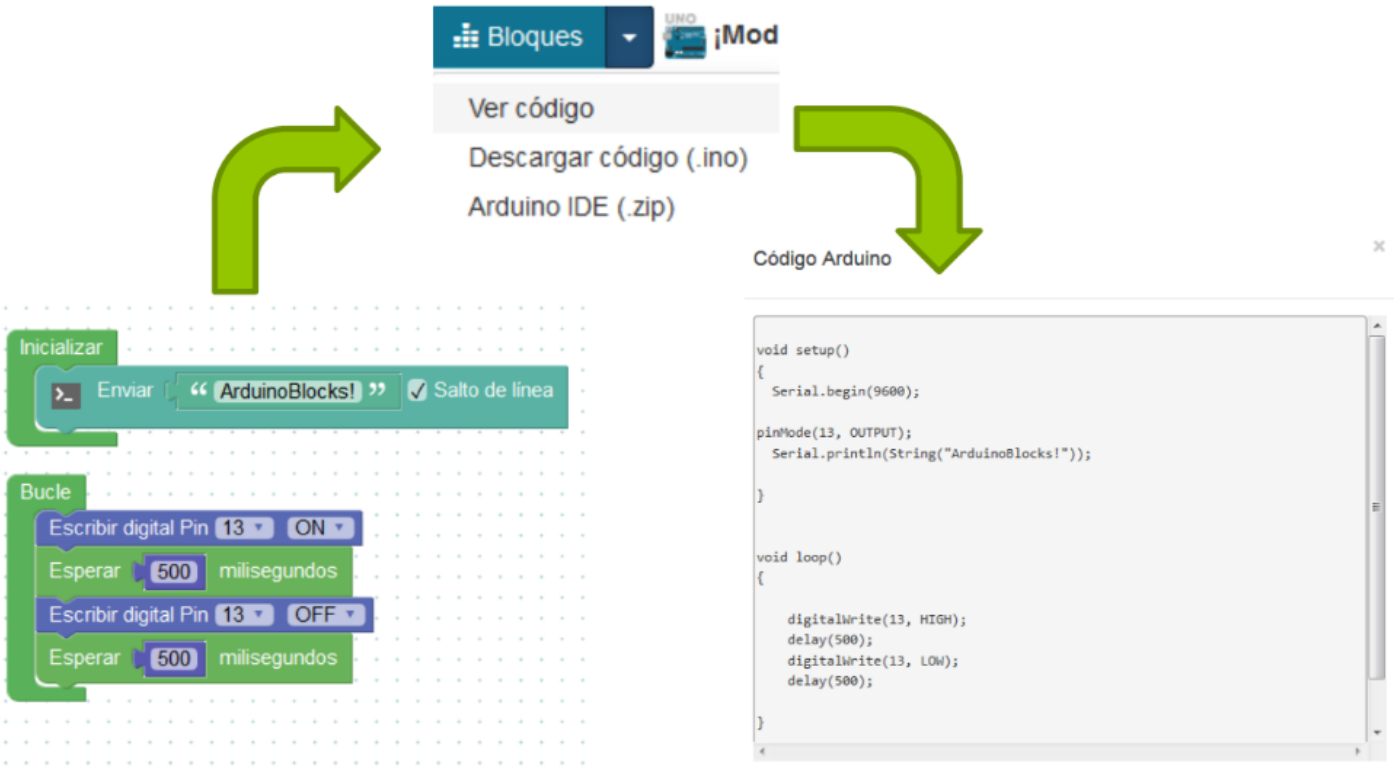

> *Adaptado de este enlace. José Andrés Echevarría @cantabRobots CC-BY-NC-SA*

Las principales secciones del área de programación son las siguientes :


> *Adaptado de este enlace. José Andrés Echevarría @cantabRobots CC-BY-NC-SA*

## *Ver el código*

**ArduinoBlocks** genera el código de Arduino a partir de los bloques. El programa se puede compilar y subir directamente a la placa Arduino gracias a la aplicación **ArduinoBlocks-Connector**, sin embargo si deseamos ver o descargar el código podemos realizarlo desde el área de bloques.




*Adaptado de este enlace. José Andrés Echevarría @cantabRobots CC-BY-NC-SA*

Siempre, desde un **lenguaje de programación en bloques** podemos obtener su equivalente a **Código de Arduino IDE (**de hecho es lo que hacen los programas), y luego con las funciones de **Código de Arduino IDE** el software lo pasa a **lenguaje máquina** que es la que se graba el Arduino, **pero no al revés** es decir, no existen programas que dado un código máquina o código Arduino IDE lo pasen a bloques gráficos, (igual que no hay programas que lean el código máquina que hay grabado en un Arduino y lo pasen a código Arduino IDE). Esto no es del todo 100% verdadero pues la Ingeniería inversa en informática trata pues de eso: obtener la fuente aunque sea parcial, pues si obtienes el código legible, puedes alterar lo que quieras.
Cuando compras un programa comercial, te dan el lenguaje máquina ilegible. Mientras que los programas de software libre se publica el código fuente legible para que todo el mundo pueda mejorarlo.

Por ejemplo en la siguiente figura, el programa gráfico **mBlock** que se utiliza en Arduino, mBot, etc... pasa sus instrucciones de lenguaje de programación de bloques parecido a Scratch a lenguaje de **Código de Arduino IDE** y Arduino IDE graba instrucciones binarias de **lenguaje máquina** al Arduino.


**¡¡A disfrutar!!**

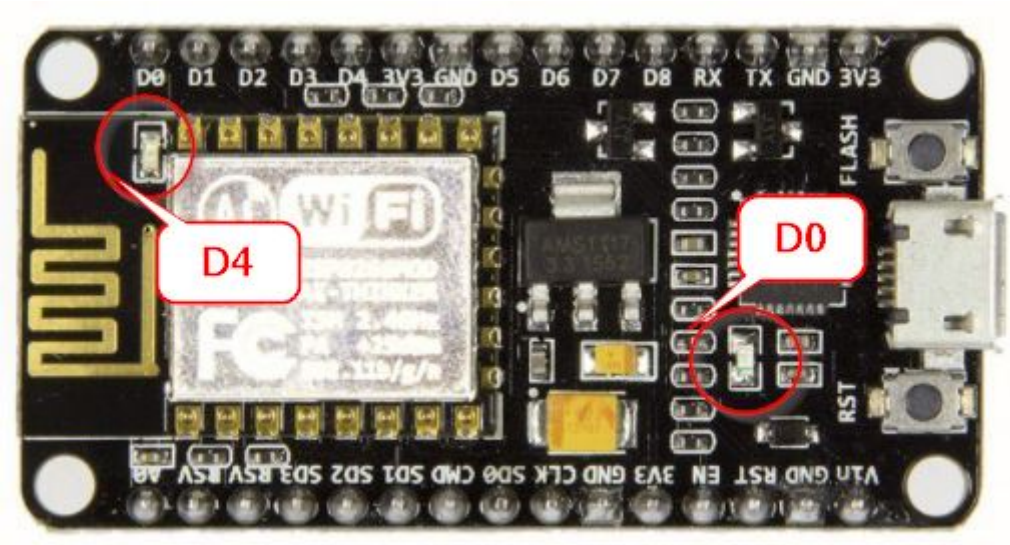
Consejo: Te recomendamos visitar el canal de Youtube de Arduinoblocks

https://www.youtube.com/c/ArduinoBlocks

# Intermitente

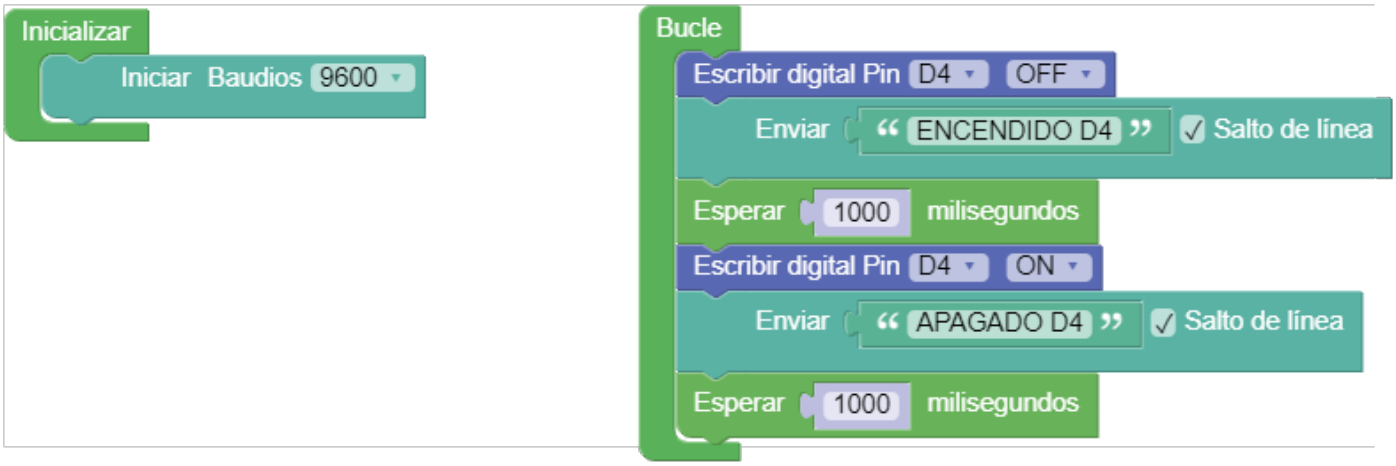
Nuestro primer programa que vamos a hacer es un intermitente en D4

> **tip** ¿Por qué con D4 y no con otro? Porque D4 tiene un led integrado en la placa, también D0.




> **danger**
>
> ## ATENCIÓN
>
> Estos leds van al reves, (configuración pop-down) luego **para encender el led, D4 tiene que estar a nivel bajo**, y al revés para apagarlo, D4 en nivel alto.

El programa es muy sencillo y lo completamos que se pueda visualizar por la salida puerto serie, por eso al iniciar ponemos el puerto serie a 9600 baudios.




Le damos a **subir** (teniendo el programa Arduinoblocks conector minimizado, eso lo podemos ver enseguida pues detecta en que COM está conectado, en la figura sale COM5) y en **Cónsola** podemos ver el contenido del puerto serie.

# ¿Qué es Arduino?

## ¿Qué es Arduino?

Arduino es una tarjeta electrónica que integra básicamente a un microcontrolador y un conjunto de pines de conexión de entradas y salidas que permiten, mediante un determinado programa, interaccionar con el medio físico mediante sensores y actuadores electrónicos. De esta forma podrás crear tus propios proyectos tecnológicos, dotarlos de sensores que detecten magnitudes físicas como luz, calor, fuerza, etc... y en base a esa información, escribiendo un programa, activar otros dispositivos (actuadores) como pequeñas bombillas, ledes, servomotores, pequeños motores DC, relés, etc... Los sensores se conectan a los pines de entrada y los actuadores a los de salida.

> **¿Sabías que.... ?** Uno de los co-creadores de Arduino es Español, de Zaragoza: **David Cuartielles** +info

## ¿Qué es un microcontrolador?

Es un circuito integrado que se puede programar, o sea que puede ejecutar las órdenes que tenga almacenadas en su memoria. Tiene las tres funciones principales de un computador: la unidad central de proceso, memoria y entradas y salidas.

Arduino utiliza la marca ATMEL, y el modelo de microcontrolador depende del tipo de tarjeta, por ejemplo la tarjeta Arduino Uno utiliza el micro ATMEL MEGA 328P. Si quieres saber las entrañas de esta placa aquí

## ¿Qué se puede hacer con Arduino? ¿Algún ejemplo?

Realmente el límite lo marca tu imaginación pero por dar alguna pista, podrías diseñar un sistema para la apertura y cierre de la puerta de un garaje, hacer un robot móvil que detecte objetos o que siga una línea negra, crear un detector de luz y oscuridad, implementar un termómetro, controlar un cilindro neumático, etc...

En este manual tienes múltiples ejemplos de pequeños proyectos para el aula, aunque Arduino es una herramienta que también se utiliza en el ámbito profesional para monitorización de sensores y automatización a pequeña escala por su flexibilidad, fiabilidad y precio.

## ¿Qué son las entradas y salidas?

Mediante los conectores de Arduino correspondientes a las entradas y salidas podemos comunicar nuestros programas con el “mundo exterior”. Si queremos leer el valor de la magnitud física medida por un sensor, por ejemplo una LDR que detecta el nivel de luminosidad, lo tendremos que hacer conectando el sensor a uno de los pines de entrada (en este caso analógicas) de la tarjeta.

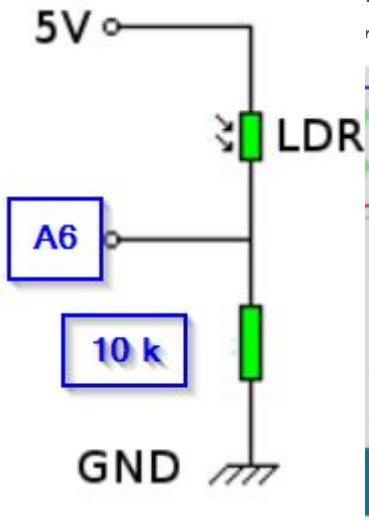
De esta forma con una simple instrucción de lectura en el programa, podremos obtener el valor de la magnitud física. Si nuestra intención es actuar o “hacer algo” una vez leído el valor del sensor, por ejemplo encender un led si el sensor de luminosidad detecta oscuridad, tendremos que conectar el actuador (en este caso el led) a un pin de salida que proporcionará la corriente necesaria para activarlo.

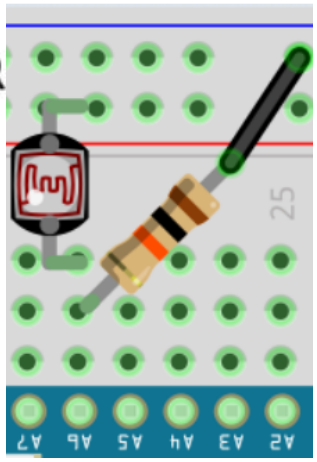
En Arduino las entradas pueden ser analógicas o digitales y las salidas sólo digitales. Cada pin digital tiene doble función entrada o salida. En la zona de configuración del programa hay que indicar explícitamente mediante una instrucción cuál es función desempeña un determinado pin.

## ¿Dónde se conectan los sensores? ¿A las entradas analógicas o digitales?

La mayoría de sensores miden señales analógicas y proporcionan una variación de voltaje dentro de un rango (normalmente de 0 a +5V) dependiendo de lo que varíe la magnitud física medida. Muchos sensores son resistivos a algo (luz, temperatura, humedad,...), es decir que varían su resistencia eléctrica con la magnitud física, pero mediante un sencillo montaje de divisor de tensión conseguimos una variación de voltaje apta para Arduino. Estos montajes los veremos en las prácticas.

Veamos este ejemplo:
**El sensor LDR** es una resistencia que cambia según la intensidad de la luz. La estrategia es colocar el LDR en *un divisor de tensión* con otra resistencia de valor parecido al promedio del que l valor del punto de unión proporciona una tensión entre 0 y 5V. Como es nectamos a una entrada analógica (en la figura al A6)



Una vez realizadas las conexiones, si midieramos la salida del sensor con un voltímetro nos daría un valor decimal, por ejemplo un nivel de luz “intermedio” (rango de 0 a 5V) de un sensor de luz podría dar 3,3 voltios. Este tipo de información el microcontrolador **no la entiende tal cual**, sólo es capaz de interpretar números binarios (“0” ó “1”) por lo que para traducir los valores analógicos dispone internamente de un conversor analógico – digital que hará la conversión entre los dos sistemas, de forma que podremos tener valores discretos de la medida de los sensores analógicos. En el Arduino **las entradas analógicas** leen valores analógicos entre 0V y la alimentación (normalmente 5V) y los convierten en números entre **0 y 1023** (porque lo codifica en 10 dígitos binarios proporcionan $2^{10}$ = 1024 combinaciones).

> Por ejemplo, si la entrada analógica lee un valor de 3,3V y la fuente de alimentación es 5V, la señal analógica que lee Arduino, haciendo una regla de 3, tiene un valor de 3,3 * 1023 / 5 = 675,18 = **675**

> **Mapeo**
> Para convertir estos valores 0 -1023 a valores más legibles, por ejemplo 0 - 100 para representarlo en % o 0-5 para que represente la medida en voltios ... veremos en programación la función **mapear**

**La mayoría de los sensores nos lo venden ya preparados** montados en una pequeña placa electrónica y con circuitos integrados auxiliares para no tener que estar haciendo divisores de tensión. Pueden tener salida analógica o **salida digital**, que en este caso lo tenemos que conectar a cualquier entrada digital D0 hasta D13.

Veamos el mismo ejemplo del LDR: Podemos comprar este módulo:

Estos módulos proporcionan 3 pines: dos que son la alimentación, (0V, GND o - )  y (+5V V+o Vcc) y el pin que proporciona la lectura (Vout o D0 o I/O). En el caso de que sea un sensor que mida una magnitud analógica como en este caso la luz, suelen proporcionar un potenciómetro para determinar qué luminosidad se considera un 0 o un 1.

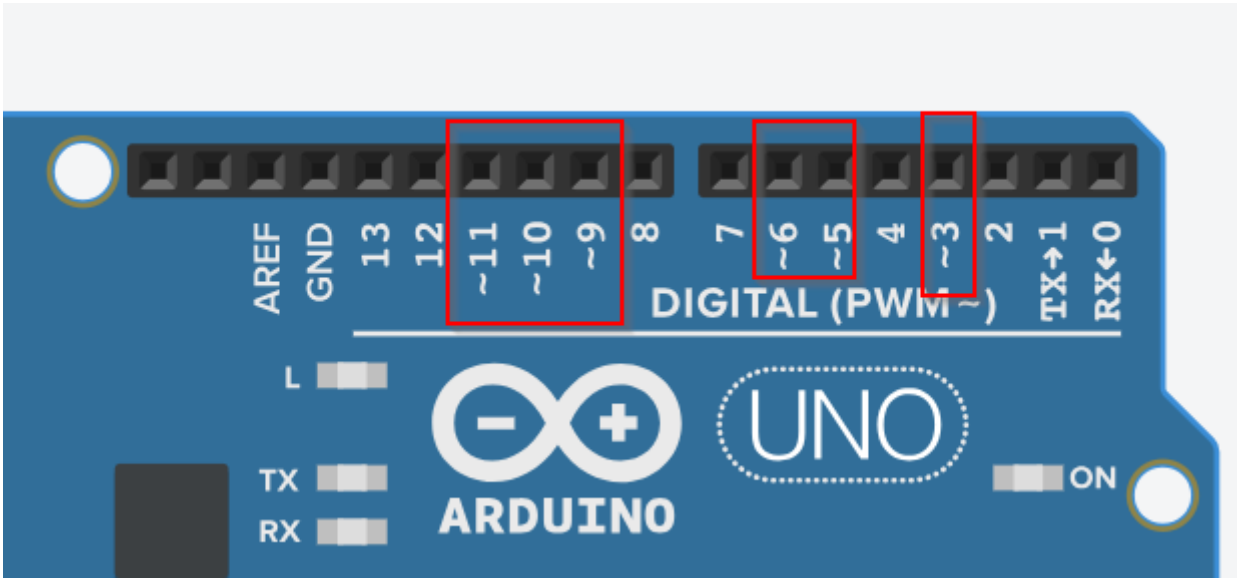
## ¿Hay sensores digitales que no estén en una placa electrónica?

Las entradas digitales sin una placa electrónica son cuando las señales a leer son valores discretos. Por ejemplo queremos poner un pulsador o un interruptor que encienda un led. Hacemos un montaje que cuando se pulse, entren 5 voltios en el pin digital de entrada y cuando no se pulse que “entren” 0 voltios. De esta manera la lectura del pin digital de entrada será “HIGH” con 5 voltios o “LOW” con 0 voltios.

Veremos más adelante que un interruptor no es un simple cable que conectamos a +5V o a 0V pues ¿Qué valor lee Arduino mientras levantamos el cable de un sitio a otro?, para ello veremos configuraciones Pull-up o Pull-down que se repiten en muchos sensores.

## ¿Qué son las salidas digitales etiquetadas con PWM (~)?

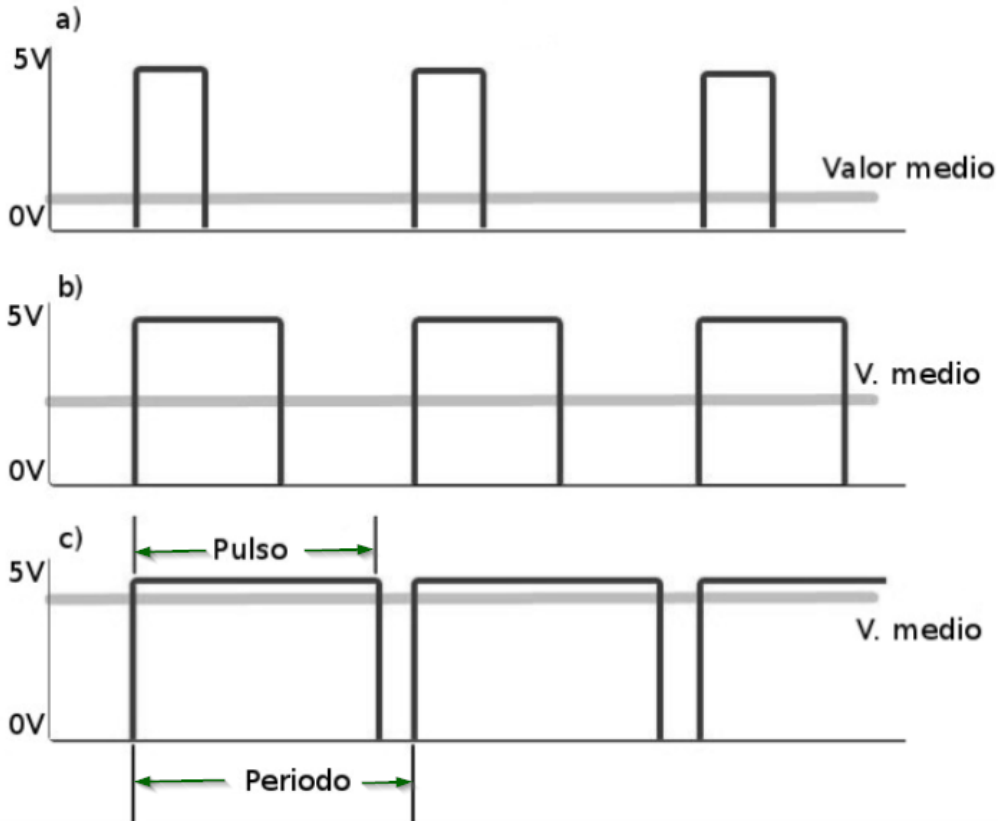
Son salidas digitales que **simulan una salida analógica**. Las siglas significan *Modulación por Ancho de Pulso (Pulse Width Modulation)* es decir, proporcionan una onda cuadrada con un nivel alto (+5V) de “cierta” duración.




**Los valores PWM** que podemos proporcionar pueden ir desde **0 a 255.**

- Un valor bajo es una señal cuadrada donde el pulso es pequeño comparado con el periodo (ver figura primera señal) por lo tanto el Valor medio es bajo.

- Un valor mitad, es decir 255/2 = 127 o 128, corresponde a una señal cuadrada perfecta (ver figura segunda señal) por lo que el Valor medio es la mitad, si 5V es la tensión de alimentación, sería 2,5V.
- Un valor alto es una señal cuadrada donde el pulso es grande comparado con el periodo (ver figura la tercera señal) por lo tanto el Valor medio es alto
- Un valor 0 corresponde a un 0V analógico y sería una señal sin pulso
- Un valor 255 corresponde a la máxima tensión (la de alimentación (normalmente 5V).
- La frecuencia es de 490Hz para los pines 3, 9, 10, 11 y de 980Hz para los pines 5 y 6 en un Arduino UNO o Nano




De esta manera podemos simular señales analógicas, esto es muy útil para activar servomotores y llevarlos a una posición determinada o variar la luminosidad de un led o en los motores de los robots que vayan más deprisa o más despacio

## ¿Puedo accionar motores DC con Arduino?

Si son motores muy, muy pequeños sí sería posible aunque no es recomendable, pueden dañar la placa. Los motores necesitan un consumo alto de corriente, sobre todo si tienen que mover cierta carga, por lo que se recomienda o bien utilizar una tarjeta Shield o extensión de Arduino que dispone de circuitería apta para proporcionar dicha corriente (transistores).

- En el curso Arduino con código utiliza una Shield llamada *Edubásica* que dispone de un transistor y un circuito integrado LM293 para realizar esta función, además de otras ventajas para el aprendizaje de Arduino.
- En el curso Rover con Arduino utiliza Shield motor para NodeMCU

# Software del Arduino

Las placas electrónicas educativas (Echidna, Micro:bit, Arduino, ESP32 ...) se programan mediante **varias opciones** :

https://docs.google.com/presentation/d/e/2PACX-1vQb1Dv9wN9QK-F6V7yvwDoyzquqwWlGvIyVJr83Yk56kAoYD7bXLnYDm_tCQkeAgg/pubembed?start=falseloop=false&delayms=3000

## OPCIÓN LENGUAJE GRÁFICO POR BLOQUES

Recomendado para primaria. Tenemos muchas posibilidades de lenguajes gráficos. Destacamos dos:

- **ECHIDNASCRATCH**
  - **Específico para Echidna e integra la IA CURSO DE ECHIDNA**
- **MBLOCK** Basado en Scratch. Aunque es un programa especializado en el robot comercial mBot, (basado en Arduino), el mismo programa está adaptado para programar Arduino.
  - **CURSO ARDUINO CON MBLOCK** se utiliza Arduino y placa Protoboard
  - **CURSO DE ECHIDNA** se utiliza la Shield Echidnam y EchidnaBlack
  - **CURSO DE MBOT** se utiliza el robot mBot
- **STEAMAKERBLOCKS (antes ARDUINOBLOCKS)** se trabaja online, muy visual y muy amigable. Está adaptado tanto para trabajar tanto Arduino como muchas placas controladoras y en el aula. Podemos verlo en los siguientes cursos:
  - **CURSO ROVER CON ARDUINO** aunque no se utiliza un Arduino, sino un NodeMCU pero la programación es exactamente igual
  - **CURSO DE ARDUINO CON ARDUINOBLOCKS** donde se utiliza el Arduino con una placa protoboard
  - **CURSO ARDUINOBLOCKS EN EL AULA** donde se utiliza la Shield TDR-STEAM
  - **CURSO ESP32 EN EL AULA** donde también utiliza la Shiedl TDR Steam pero la placa no es un Arduino, sino ESP32, la programación es exactamente igual.

- **Microbloks** https://microblocks.fun/ placas: Arduino, Microbit, ESP32, RaspberryPico, .... ver minitutorial

Otros softwares para programar con bloques

- **Snap4Arduino** https://snap4arduino.rocks/run/ Online, libre... ver compartiva vs mBlock
- **S4A** https://s4a.cat/

**EN VIVO ¿Qué es eso?**

Existe una posibilidad de utilizar la placa "en vivo" frente a "cargar" el programa en la placa. Es decir, interactuando con el ordenador. El programa está en el PC. En la placa hay un **firmware** que le dice que este a las órdenes del PC. De esta manera podemos por ejemplo:

- Enviar órdenes desde el ordenador a la placa.
  Por ejemplo que al pulsar la tecla espacio que se encienda el led D13
- Enviar información desde la placa al ordenador
  Por ejemplo que muestre por pantalla nos muestre la cantidad de luz, que registra el sensor LDR, etc...

Que nosotros sepamos, estos programas permiten la programación en vivo :

- **mBlock** placas: Arduino, Microbit, Raspberry Pi, ... robots de Makeblock: mBot, Cyberpi...
- **EchidnaScratch** CURSO DE ECHIDNA
- **Microblocks**

**VENTAJAS** LA PROGRAMACIÓN EN VIVO PERMITE MUCHO JUEGO Y POSIBILIDADES A LA HORA DE ELABORAR PROYECTOS
**INCONVENIENTES**: Necesitas el ordenador encendido y conectado al robot.

**TAMBIÉN ES EN VIVO PERO ....**

Hay otros softwares que técnicamente trabajan en vivo, es decir, que el programa se ejecuta desde el ordenador, no se ejecuta en la placa, son :

- **Snap4Arduino** para placas Arduino
- **Picobriks** blocks para Picobrick board

**Pero no permiten trabajar utilizando los elementos del ordenador** (teclado, webcam, pantalla, sprite o objetos,,,)

Es **importante** que entiendas que para trabajar en vivo, la placa tiene que tener cargado un "**firmware**" para que interactúe con el ordenador.

**P: ¿Qué es eso de "firmware"?**
R: No es más que un software que se graba en los chips de la placa.
**P ¿Y por qué se llama así, y no se llama software o programa y en paz?**
R: Digamos que como se graba en los chips, es un medio camino entre software y hardware, para diferenciarlo del software habitual.

**EN CARGA ¿Qué es eso?**

Simplemente el programa que estas haciendo **se carga** en la placa
VENTAJAS: Eres independiente del ordenador, tu robot funciona independiente
DESVENTAJAS Pierdes todas las posibilidades de utilizar los recursos de un ordenador, teclado, pantalla, webcam, altavoces...

Es **importante** que si **cargas** tu programa en la placa, **pierdes** lo que había antes
Es decir, si quieres volver a trabajar EN VIVO tienes que cargar el firmware correspondiente.

# OPCIÓN LENGUAJE POR CÓDIGO

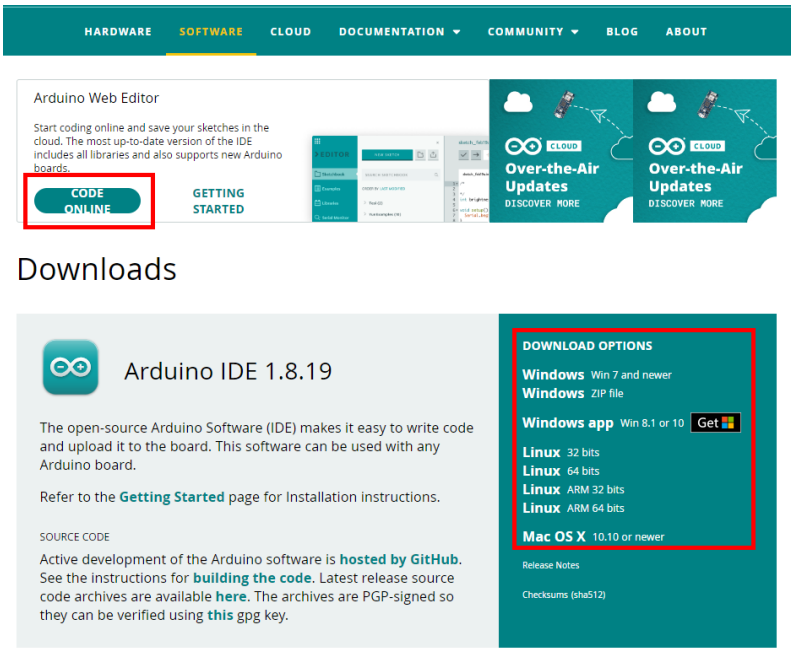
Recomendable a partir de secundaria.

## LENGUAJE POR CÓDIGO ARDUINO IDE

Es un lenguaje basado en Wiring y permite la programación del Arduino en un entorno de desarrollo (basado en Processing). El programa se llama **ARDUINO IDE** y se puede descargar desde la página oficial de Arduino: https://www.arduino.cc/en/software.

Hay otra posibilidad que es utilizarlo online, con la ventaja de tener tus proyectos "en la nube" y no depender del equipo. OJO, TIENES QUE TENER INSTALADO EL SOFTWARE **CREATE AGENT**

https://create.arduino.cc/getting-started/plugin/welcome




En los cursos de CATEDU se ha utilizado el lenguaje por código empezando desde cero en:

- **CURSO ARDUINO CON CÓDIGO** donde se trabaja con el Arduino y con diferentes sensores y actuadores, con o sin placa Shield Edubásica.
- **CURSO DE DOMOTICA CON ARDUINO** donde se realiza una maqueta de una casa controlada con domótica. También el curso ofrece la versión de hacer la misma maqueta utilizando lenguaje gráfico por bloques.

Recomendamos estas hojas resumen si vas a trabajar con código:

- En Español: enlaceDrive, enlaceGithub
- En Inglés: enlaceDrive, enlaceGithub, enlaceSpakrfun

## LENGUAJE POR CÓDIGO PYTHON

Es un lenguaje más amigable que ArduinoIDE y tiene muchos campos de aplicación aparte de la robótica

- Arduino tiene su propia versión para trabajar con sus placas compatibles: **Arduino Lab for Micropython**
  - Ver curso Arduino ALVIK
- Un compilador universal **Thonny**
  - Ver curso Pico-briks

# VENTAJAS E INCONVENIENTES
# LENGUAJE GRÁFICO POR BLOQUES vs CÓDIGO

El lenguaje gráfico por bloques es un lenguaje **sencillo de utilizar**, nos evita tener en cuenta muchas **librerías** y **cálculos.**

> Otra ventaja, es que el lenguaje por bloques es el único que permite programación "en vivo"

Por ejemplo, la instrucción leer valor distancia el sensor ultrasonidos, mediante programación por bloques es


mientras que en código es

```
double distancia;


double fnc_ultrasonic_distance(int _t, int _e){
  unsigned long dur=0;
  digitalWrite(_t, LOW);
  delayMicroseconds(5);
  digitalWrite(_t, HIGH);
  delayMicroseconds(10);
  digitalWrite(_t, LOW);
  dur = pulseIn(_e, HIGH, 18000);
      // devuelve cuanto tarda el pulso alto en microseg; 18000 es el tiempo a esperar limite
  if(dur==0)return 999.0;
  return (dur/57);
      // la velocidad del sonido es 344m/s = 34400 cm/seg = 0,0344 cm/microseg
     // como v=e/t luego e = v*t y como cuenta la ida y la vuelta distancia = v*t/2
     // luego distancia = 0,0344/2 * dur = dur/57
}
```

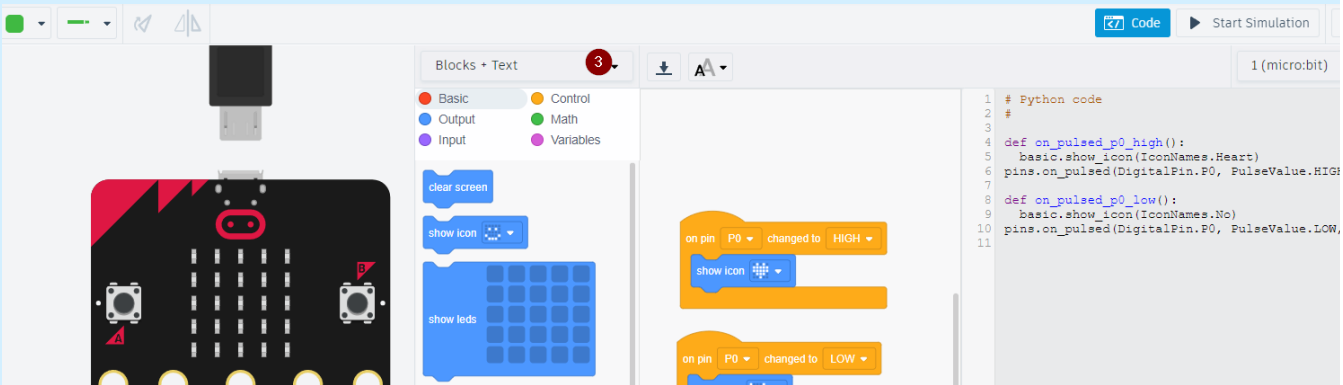
```
void setup()
{
  pinMode(6, OUTPUT);
pinMode(5, INPUT);

}


void loop()
{

  distancia = fnc_ultrasonic_distance(6,5);

}
```

Como se puede ver en **código**, tiene que calcular la distancia haciendo cálculos del tiempo de rebote del eco, mientras que el **gráfico** es sumamente sencillo y se centra en el objetivo del algoritmo a crear, no en lo accesorio. Esto hace que un lenguaje gráfico por bloques se puede aplicar **desde los 8 años**.

Por otra parte, el lenguaje **código tiene todo el potencial,** es decir, no todo está en los lenguajes gráficos. Si se quiere cosas más avanzadas, hay que recurrir al código.

> Un lenguaje **gráfico** se convierte en lenguaje **código**, pero **al revés no se puede,** debido a que el código es más depurado y no tiene la información necesaria para volver a su origen en bloques, ya lo has visto con el anterior ejemplo, el código tiene más información.

> **¿No te lo crees?** Haz la prueba, métete en https://www.tinkercad.com/ crea un programa con bloques, dale a la pestaña de código y te aparecerá una advertencia que perderás el programa con bloques ! **no puedes volver atrás**!
>
> Curiosamente, tiene una opción *bloques+código* que traduce cada bloque con un código, es decir, traduce cada bloque sin perder información, sólo de esa manera se puede pasar de bloques a código y viceversa.



> Hay herramientas para pasar de bloques a código pero no al revés

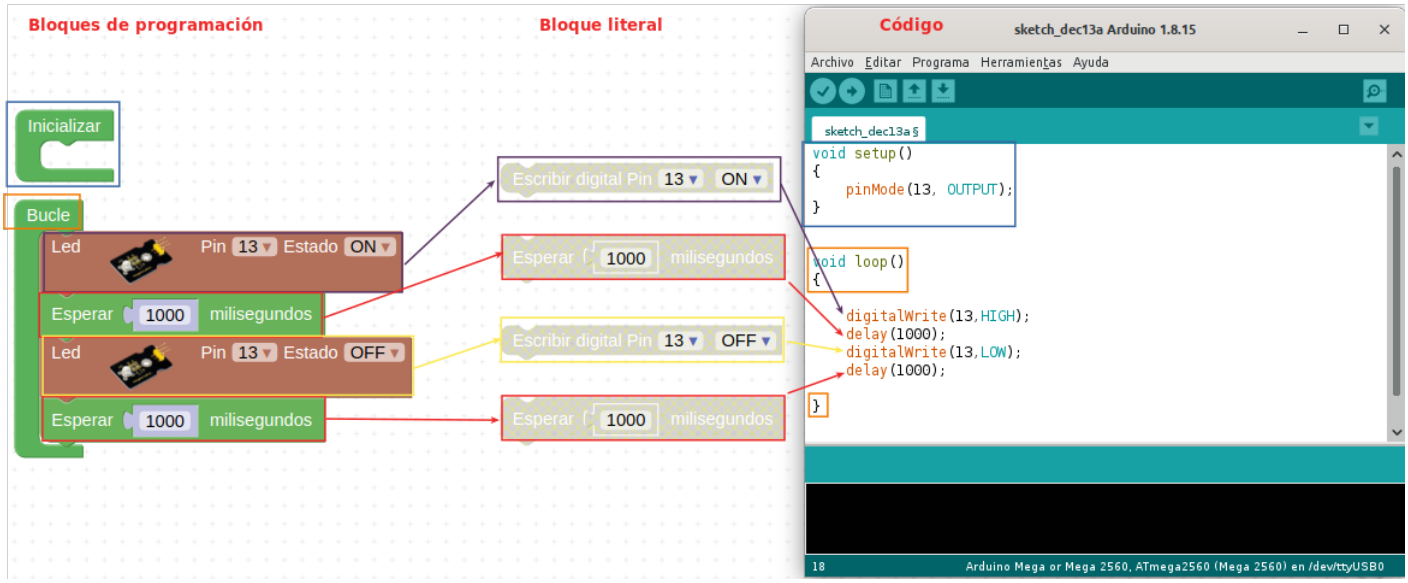



*Imagen Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA*

> El lenguaje **código** se traduce en lenguaje **máquina** (ceros y unos) entendible para el microprocesador, **pero al revés no se puede**.

En este vídeo, en mi opinión se olvida de mBlock, Snap4Arduino, S4A pero puedes ver un vistazo de los diferentes editores

# OPCIÓN SIMULACIÓN

Incluimos dentro del apartado de Software los diferentes programas que hay para simular placas electrónicas como Arduino, ESP32, Micro:bit etc...

## Tinkercad

Esta herramienta https://www.tinkercad.com aparece en el **Curso Arduino con código** en la práctica Comunicación entre dos Arduinos, pero también es una plataforma que sirve para hacer los diseños de elementos 3D, ver curso **Impresión 3D con Tinkercad**

Tiene la ventaja que es aplicación **online**, muy **visual** y buscan un reflejo de la práctica **real,** además de estar la herramienta **adaptada al aula** (gestión de alumnos y proyectos). Como desventajas podemos decir que no tiene mucha variedad de componentes electrónicos y la simulación es algo lenta.

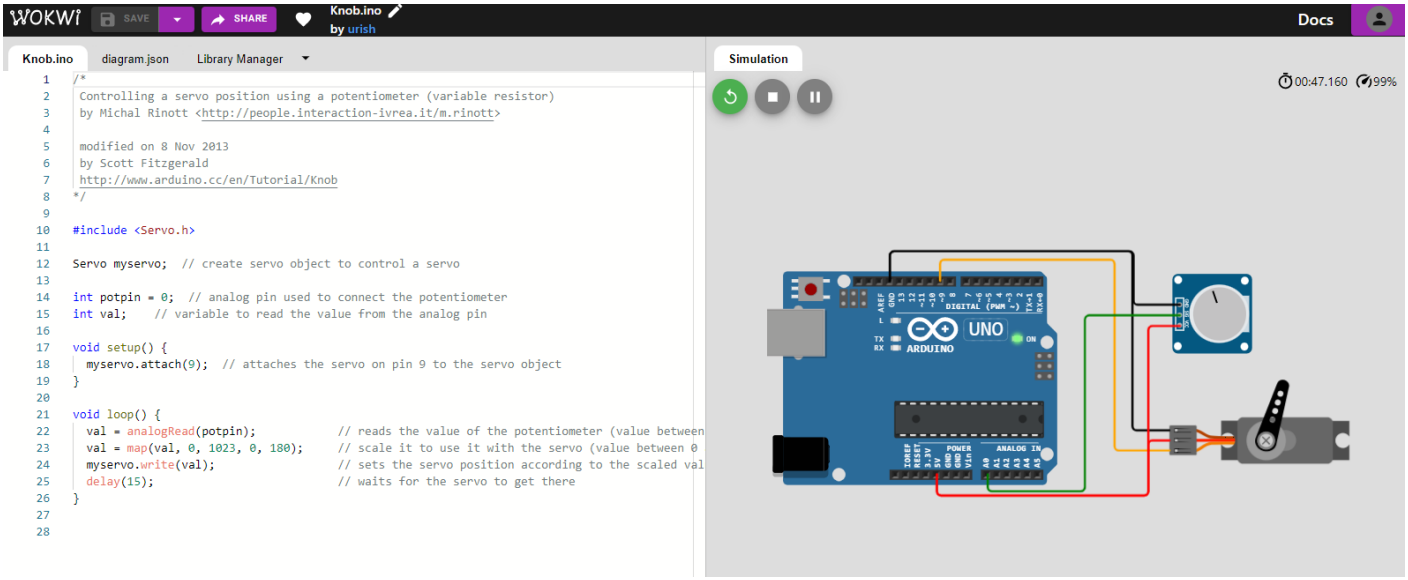
https://www.youtube.com/embed/pXEv0wxW9Jo?rel=0

Además permite embeber, **pruébalo !!!** □□□□□ Pulsa **Iniciar simulación** y luego pulsa el **botón A** de cualquiera de los dos micro:bits

https://www.tinkercad.com/embed/8vBL2E8wWDx?editbtn=1

## Wokwi

Si Tinkercad se queda corto, puedes probar esta plataforma https://wokwi.com/ con muchas posibilidades. Es **online** y puede trabajar con **multitud de placas**: ArduinoUno, ESP32, Raspberry,,,,
Como única desventaja que encontramos, es que echamos de menos la realidad de Tinkercad, por ejemplo no puedes poner una placa protoboard para realizar las conexiones, pero a cambio se gana simplicidad de cableado.



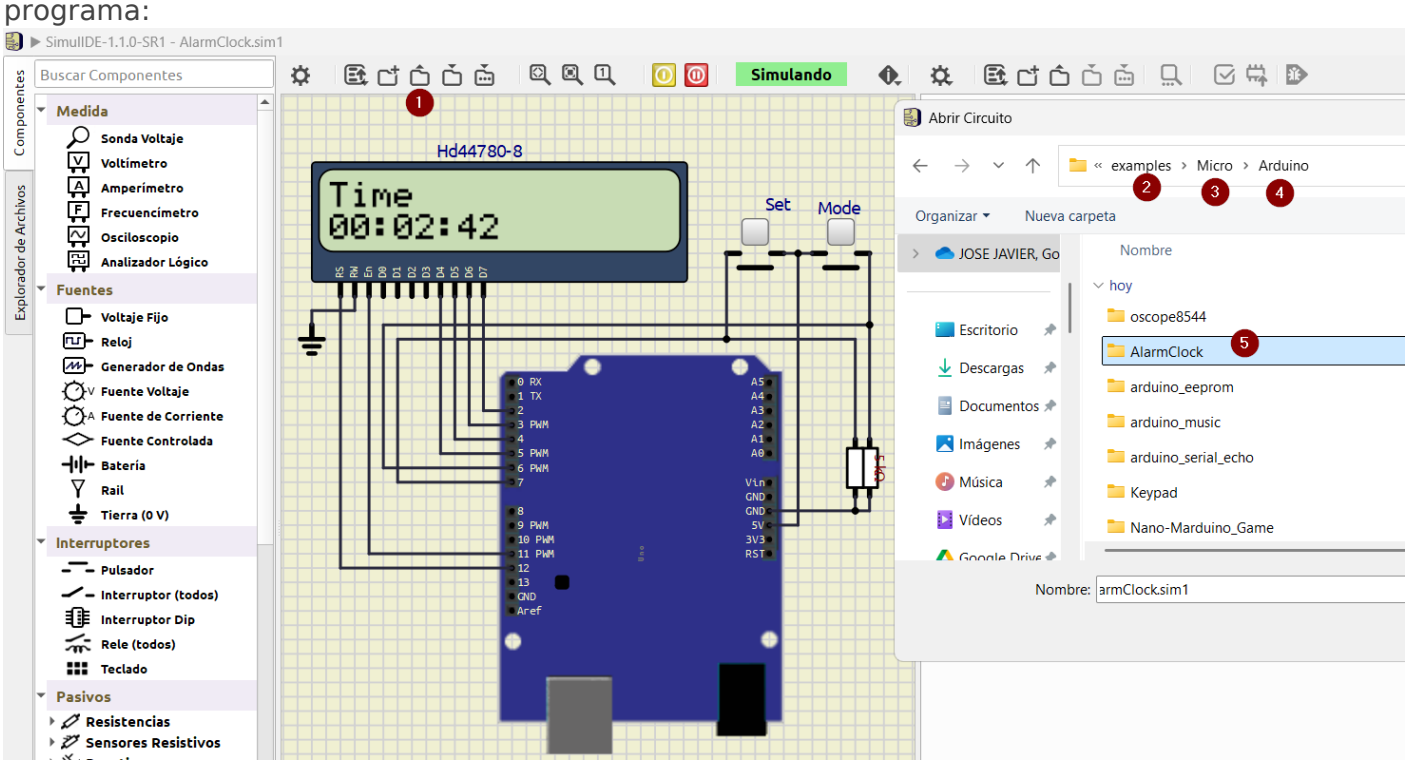

## UnoArduSim

Es una **aplicación local.** **UnoArduSim** además es una aplicación portable fácil de instalar y con los elementos de leds, motores servos ya preparados, ideal para ejemplos sencillos y para examinar señales y no depender de Internet, pero no es tan versátil.

https://www.youtube.com/embed/WLJ_l4uGjXg?rel=0

## SIMULIDE

En https://simulide.com/ podemos encontrar un programa local de software libre genérico para electrónica, incluido Arduino. En esta captura se puede ver uno de los ejemplos que incorpora el programa:




## OPCIÓN SÓLO DIBUJAR

- **TinkerCad** es un buen programa para dibujar los planos
  - ✅ permite también la simulación
  - ✅ permite embeber y compartir
  - ❌ no tiene muchos componentes
- **SimulIDE** es software libre. Es un programa portable.
  - ✅ Tiene muchos componentes
  - ✅ permite también la simulación
  - ❌ le faltan algunos sensores, pero van incorporando
- **Fritzing** es un clásico. Es un programa portable.
  - ✅ Tiene muchos componentes
  - ❌ no es gratis, hay que pagar 8€
- **Circuit canvas**
  - ✅ puede compartir por ejemplo

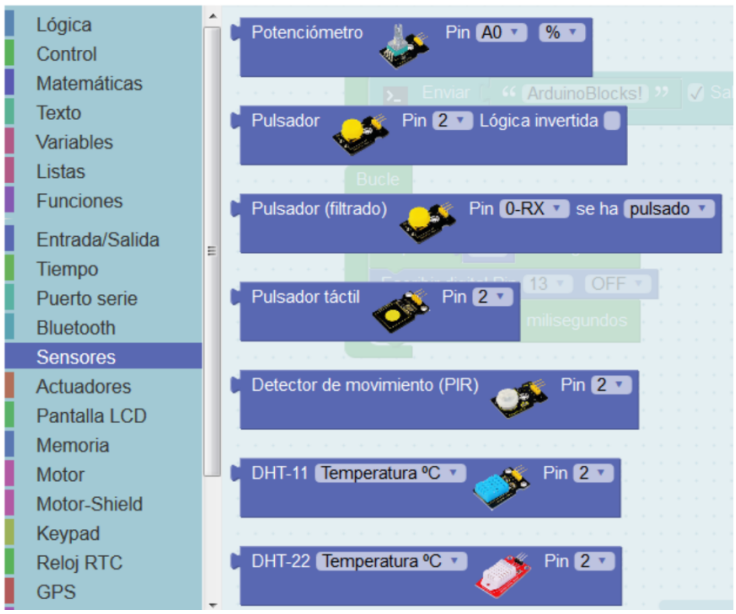
- tiene buenos tutoriales sobre electrónica
- todo en inglés

# Sensores

> Esta sección es una visión rápida de las posibles **entradas** de los robots.
> **NO LEAS TODOS SINO LOS QUE TIENE TU ROBOT**

## Un poco de teoría...

Cualquier sistema de control podríamos decir que funciona de una manera similar a un ser humano, salvando las distancias. Nosotros recibimos la información del mundo exterior gracias a nuestros sentidos (oído, olfato, gusto, vista y tacto), nuestro cerebro procesa esa información y a través de nuestros músculos o de nuestra voz realizamos diferentes acciones. Pues lo mismo sucede con los sistemas de control, reciben información del exterior gracias a los diferentes **SENSORES**, procesan esa información en sus PLACAS CONTROLADORAS (sus cerebros) tales como Arduino y dan una respuesta utilizando sus diferentes **ACTUADORES**.




Un sensor es un objeto capaz de detectar magnitudes físicas o químicas y transformarlas en variables eléctricas. Los sensores o periféricos de entrada nos permiten obtener información del mundo real para utilizarla desde el programa de Arduino.

En la actualidad la cantidad de sensores disponibles es tan extensa como las variables que queramos medir, desde sensores de temperatura, humedad, luminosidad,... hasta acelerómetros,

giroscopios, GPS,... pasando por detectores de gases, de pulsos cardiacos, sensores de efecto HALL,...

## Tipos de sensores

- **DIGITAL**: un sensor digital sólo tiene dos estados: activado/desactivado, ON/OFF, 1/0, Alto/Bajo, ... En este caso conectaremos el sensor a una de las entradas digitales de Arduino para leer el estado.

> Ejemplo: un pulsador es un tipo de sensor sencillo que sólo nos da dos estados, “pulsado o no pulsado”. Conectado a la placa Arduino debe generar 0v en reposo y 5v al pulsarlo. De esta forma desde el programa de Arduino podremos leer el estado del botón.

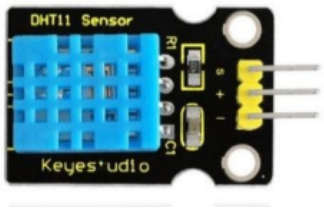

- **ANALÓGICO**: el sensor nos puede dar un rango de valores, normalmente se traduce en un valor de tensión o de corriente variable en función de la señal captada al sensor. En este caso conectaremos el sensor a una de las entradas analógicas de Arduino (A0,..., A5). El rango de entrada será una tensión entre 0v (GND) y 5v.

> Ejemplo: Una fotorresistencia es un componente electrónico cuya resistencia disminuye con el aumento de intensidad de luz incidente. Su valor varía entre 0 y 5 v. la cantidad de valores que pueden leer las entradas analógicas de Ardunio son de 10 bits es decir 1024 valores. De tal modo que 0 = 0 v. y 1023 = 5V.


- **DATOS**: el sensor ofrece su información a través de una interfaz de comunicación. La forma de comunicación puede ser por sistemas estándar como **I2C** o SPI o algunos sensores usan su propio protocolo para codificar la información y debemos realizar desde el software la decodificación correcta para interpretar los datos del sensor (normalmente

los desarrolladores de este tipo de sensores ofrecen una librería software para Arduino que hace todo el trabajo).

> Ejemplo: el sensor DHT11. Por un solo pin envía los datos de temperatura y humedad.




# Sensores modulares.

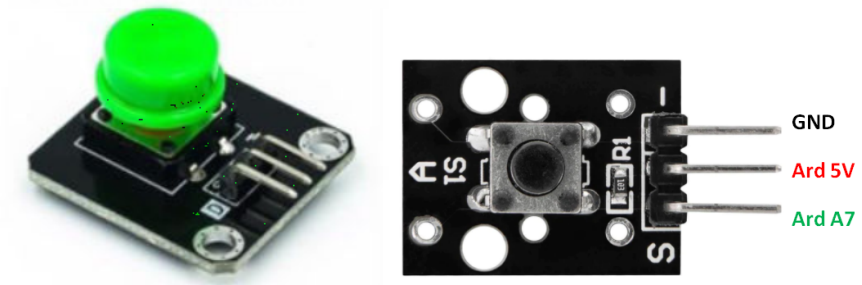
En la actualidad existen infinidad de sensores que los fabricantes presentan en forma modular. Esto hace que su conexión y utilización sea mucho más sencilla que la tradicional, olvidándonos de resistencias, polaridades, cableados,... para su correcto funcionamiento.

## Sensor pulsador

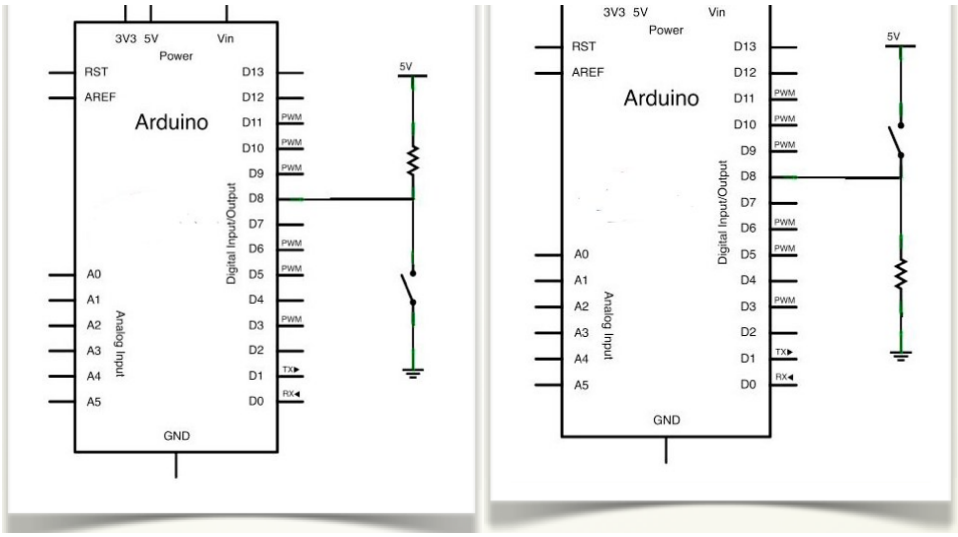
Es un sensor digital, que presenta dos estados; cuando se presiona el botón, emite una señal de bajo (0V), cuando suelta el botón, emite una señal de bajo alto (5V). Datasheet

Un ejemplo de uso

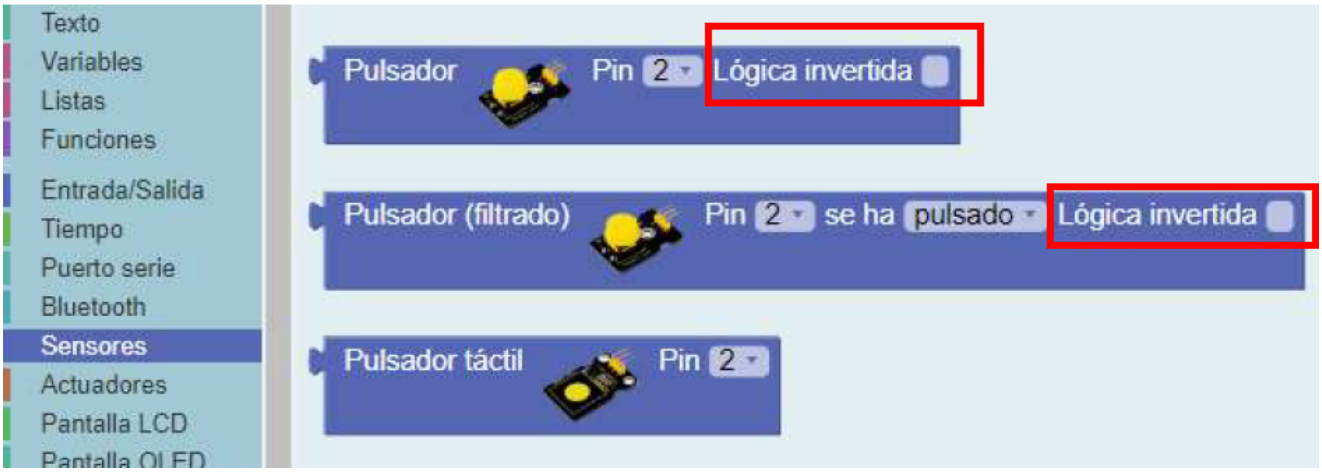
- en el robot mClon




Otra manera más "barata" de sustituir este módulo pulsador es poner un pulsador normal y una resistencia (±10k), al pulsar se produce una entrada en el Arduino, hay dos configuraciones, que al pulsar se emita un 0 lógico (configuración **Pull up**) o que al pulsar emita un 1 lógico (configuración **Pull down**) ¿Por qué hay que poner una resistencia?




Lo "normal" es que al pulsar se emita un '1' configuración **Pull down**, pero hay pulsadores que funcionan **Pull up** y los llaman **lógica invertida**, por eso en la programación por bloques podemos encontrar esto:




## Sensor Táctil Capacitivo.

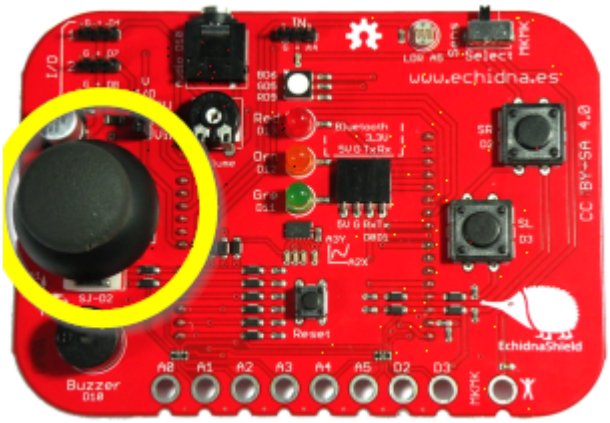
Este pequeño sensor puede "sentir" a las personas y el tacto y la retroalimentación de metales a un nivel de voltaje alto / bajo. Incluso aislado por alguna tela y papel, todavía puede sentir el tacto. Su sensibilidad disminuye a medida que la
capa de aislamiento se hace más gruesa. En nuestra opinión lo preferimos frente al *Sensor pulsador* pues es muy económico, duradero y fiable.

Un ejemplo de uso en

- Disparo láser en Arduino con ArduinoBlocks
- Apertura de puerta en Domotica con Arduino


## Potenciómetro y joystick

Un potenciómetro es una resistencia variable, es decir, cambia de valor mecánicamente, lo tenemos en multitud de dispositivos. El **joystick** es internamente dos potenciómetros con un pulsador integrado en un solo mando.

Este sensor es **analógico**, su salida puede ser cualquier valor entre Vcc y GND (si está en divisor de tensión como en la placa Edubásica no llega a esos valores extremos), por lo tanto hay que conectarlo a una entrada analógica de Arduino y como cualquier entrada analógica, proporcionará valores entre 0 y 1023.

Ejemplos de uso:

- Arduino con código: Mapeo del potenciómetro
- Arduino con código: Regular la luz con potenciómetro
- Arduinoblocks en el aula
- En Arduino con Echidna, con joystick
- Domótica con Arduino con joystick

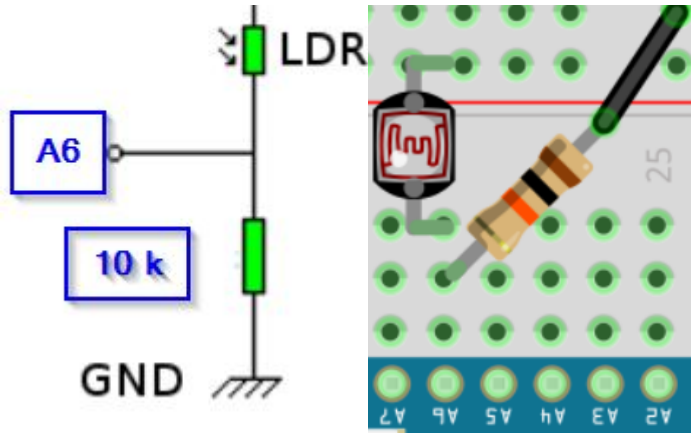
## Sensor Fotocélula LDR.

El uso de fotocélulas es muy común en nuestras vidas, las encontramos en el encendido automático de farolas, apertura de puertas,... La fotocélula es un semiconductor. Es ampliamente utilizado en campos de interruptores de control automático como cámaras, luces solares de jardín, lámparas de césped, detectores de dinero, relojes de cuarzo, tazas de música, cajas de regalo, mini luces nocturnas, interruptores de control de luz y sonido, etc. Es un sensor analógico dando valores entre 0 y 5V y como entrada analógica de un Arduino se traduce en un rango de 0 a 1023 valores.
Un ejemplos de uso :

- el interruptor crepuscular del curso Arduino con ArduinoBlocks
- Medir la luz en Rover con Arduino
- Medir la oscuridad en Arduino con mBlock
- Hinchar un balón en Arduino con mBlock

Una manera más económica de montar este sensor es utilizar una resistencia y un LDR:

- El LDR cuando más oscuridad, más resistencia
- En una configuración **PULL DOWN,** cuanto **más** luz, la resistencia del LDR baja, por lo tanto **más** tensión en A6

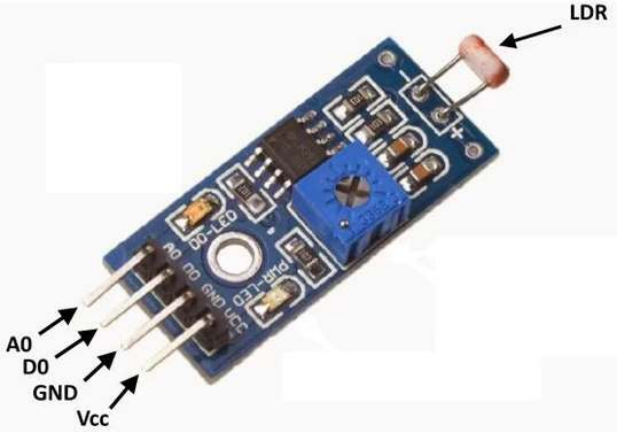



Los módulos LDR que se venden suelen esta configuración Pull down, es decir, cuanto **más** luz, **más** tensión:


La instrucción con Arduinoblocks ya cuenta con esta configuración Pull downUp de que cuando **más** luz, **más** valor tiene la entrada analógica.


Hay módulos LDR ya montados,  que tienen componentes **activos** es decir, llevan circuitos electrónicos, transistores que amplifican etc... y dan la salida **digital** con un potenciómetro para definir el rango de luz que cambia de estado lógico. Puedes ver en la figura que tiene una salida digital **D0**.


O hay algunos que tienen 4 pines como en la figura que ofrecen las dos cosas: salida analógica **A0** y digital **D0**.




Nosotros aconsejamos el divisor de tensión por tres razones: más barato, no implica gran circuitería y es visible su funcionamiento frente a estos encapsulados.

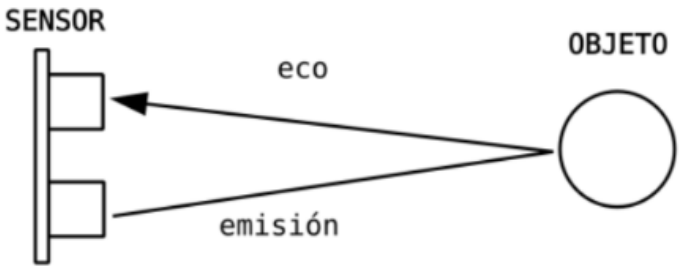
## Sensor de Ultrasonidos.

Es un sensor digital de distancias por ultrasonidos capaz de detectar objetos y calcular la distancia a la que se encuentra en un rango de 2 a 350 cm. Su uso es tan sencillo como enviar el pulso de arranque y medir la anchura del pulso de retorno.
**No es un sensor preciso**, con una ligera inclinación de la superficie ya da lecturas erróneas pero es muy barato

El más común es el **HC-SR04** que tiene 4 pines de conexión: **VCC** **Trig** (Disparo del ultrasonido) **Echo** (Recepción del ultrasonido) y **GND** aunque en algunos modelos como el de Elecfreaks tiene 3 pines. Integra Trig y Echo en uno sólo.
La distancia se calcula con esta fórmula:
**Distancia en cm = {(Tiempo en segundos entre Trig y el Echo) * (V.Sonido 34000 en cm/s)} / 2**
Si programas en código, tienes que utilizar la fórmula anterior, previamente tienes que programar

el cálculo del tiempo entre una emisión de un pulso en Trg y la respuesta en Echo.

Si no quieres programar bloques, no es necesario, seguro que hay un bloque que lo hace todo


Ejemplos de uso:

- Alarma en Domótica con Arduino
- Piano invisible en Arduino con ArduinoBlocks,
- Sensor parking en Arduino con ArduinoBlocks
- Piano invisible en Arduino con mBlock
- Sensor parking en Arduino con mBlock
- Sensor de distancia de ultrasonidos con Picobricks

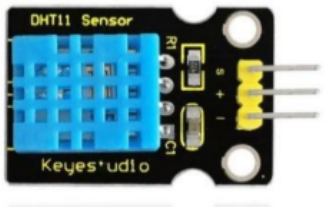
## Sensor DHT11 (Temperatura y Humedad).

Este sensor de temperatura y humedad **DHT11** nos permite determinar las zonas de confort para un rango de temperaturas entre 0ºC y 50ºC con un error de ± 2ºC y un rango de humedad entre 20 y 90 % ± 5% . Una salida digital para dos variables cómo lo hace? Tiene dentro un pequeño microprocesador que lanza por el bit de datos 40 bits en serie, los 16 primeros son la humedad (en BCD) y los 16 restantes es la temperatura (en BCD) los 8 restantes son de comprobación **Checksum** (en binario) como la letra del DNI. Por ejemplo 0100 0111 0000 0011 0001 1001 0000 0000 0001 1000 es 0100 0111 0000 0011 = 47.03% de humedad y 0001 1001 0000 0000 = 19.00ºC y la comprobación es la suma de 4+7+0+3+1+9+0+0=24=11000

Ejemplos de uso:

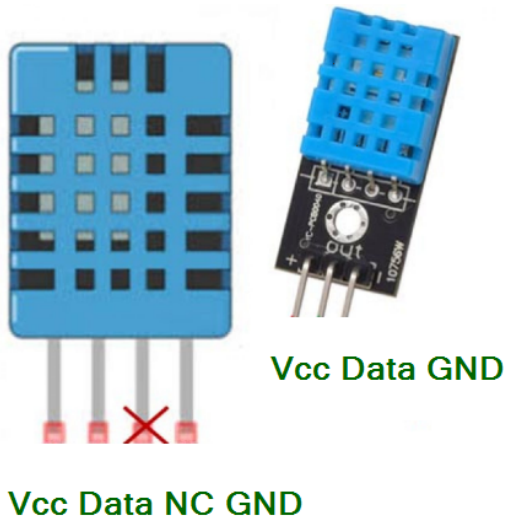
- Medir H y T con Blink en Rover con Arduino
- Estación meteorológica Arduino con Arduinoblocks

- Arduinoblocks en el aula
- SMART HOME con Micro:bit

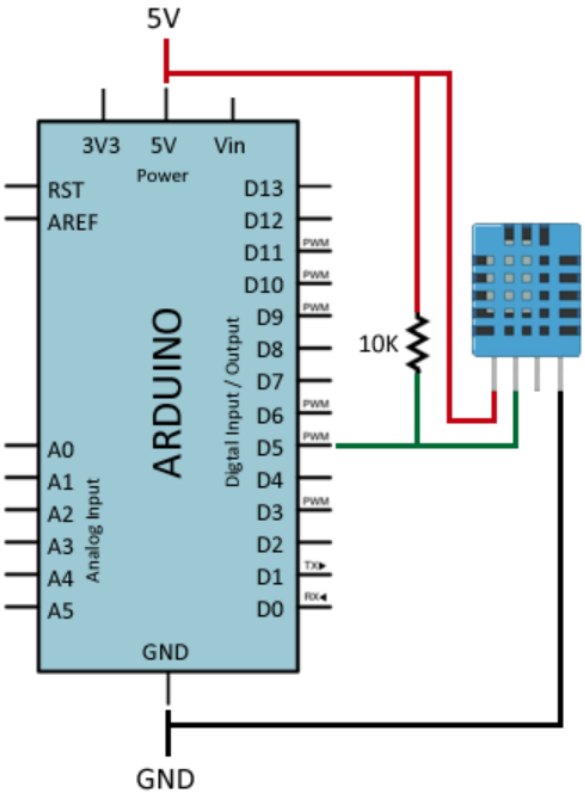



No es un sensor con gran sensibilidad, pero para propósitos educativos cumple sus funciones. Por dentro tiene una resistencia NTC que decrementa su resistencia si aumenta la temperatura. Hay otros que van al revés, los PTC. Tanto los NTC como los PTC se llaman ***thermistores.*** Para la humedad, mide la capacidad de un condensador que es sensible a la humedad, o sea, un **sensor capacitivo.**

Tenemos dos opciones comerciales: **Encapsulado** que lo tienes preparado para conectar la alimentación y leer por el pin de datos, o **sin encapsular**, que hay que colocar una resistencia de aproximadamente 10k entre Vcc y Data




Ejemplo de uso de un DHT11 sin encapsular:




Fuente Luis LLamas CC-BY-NC-SA https://www.luisllamas.es/arduino-dht11-dht22/

Si queremos mejorar la sensibilidad, podemos utilizar el **DHT22** que es igual pero de color blanco y más caro. Si lo que queremos es sólo la temperatura es mejor utilizar el **LM35D** que tiene un rango de temperaturas desde 0ºC a 100ºC con una sensibilidad de 2mV/ºC


> Es un sensor bastante mediocre, si necesitas una precisión el doble, te recomendamos el DHT22 que funciona exáctamente igual pero es de color blanco y más caro ~3€. Ver https://www.luisllamas.es/arduino-dht11-dht22/

## Sensor IR

Es un sensor para distancias cortas hasta 2cm y no da la distancia, simplemente si hay o no hay obstáculo, pero son muy baratos, unos 0.30€. Aquí tienes un ejemplo de evita obstáculos en un rover marciano con Raspberry Para saber más te recomendamos esta página de Luis Llamas

https://sketchfab.com/models/6ad4f3afb83940fea95cd3846aa68a18/embed

**IR Sensor Module for Arduino Projects | 3D Model** by **Veer AI** on **Sketchfab**

## Sensor llama

Este sensor de llama se puede utilizar para detectar fuego u otras luces cuya longitud de onda se encuentra entre 760 nm ~ 1100nm.

Un ejemplo de su uso:

- Alarma por fuego en Domótica con Arduino

| Radiación | | Longitud de onda λ |
|---|---|---|
| Ultravioleta 100-400 nm | ultravioleta C | 100 nm – 280 nm |
| | ultravioleta B | 280 nm – 315 nm |
| | ultravioleta A | 315 nm – 400 nm |
| Visible 400-780 nm | violeta | 400 nm – 455 nm |
| | azul | 455 nm – 490 nm |
| | verde | 490 nm – 570 nm |
| | amarillo | 570 nm – 590 nm |
| | anaranjado | 590 nm – 620 nm |
| | rojo | 620 nm – 780 nm |
| Infrarroja 780nm-1mm | infrarroja A | 780 nm – 1400 nm |
| | infrarroja B | 1400 nm – 3000 nm |
| | infrarroja C | 3000 nm – 1 mm |

## Sensor de Gas (MQ2).

Detecta gases inflamables : GLP, I-butano, propano, metano, alcohol, hidrógeno, humo... con más sensibilidad en algunos que en otros. Siempre detecta el conjunto. Son usados en electrónica de consumo y mercados industriales.

- **Sensibilidad** Tiene alta sensibilidad y se puede ajustar girando el potenciómetro.
- **Tiempo de respuesta**: Internamente posee un calentador para aumentar su temperatura y que estos gases reaccionen con la resistencia interna que tiene, por lo

tanto tardan algo en responder la primera vez que se conectan, incluso horas en algunos modelos. Una vez calentados son rápidos en la respuesta.

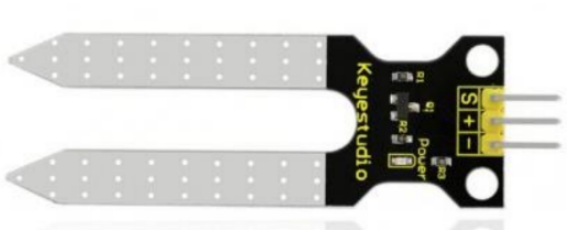
- **Tipo de salida**: Analógico pero si tiene 4 pines como el de la figura, incorpora un pin digital.
- Ejemplos de uso:
  - Smart Home Microbit
  - Smart Home ESP32

## Sensor de humedad de suelo.

> Un higrómetro de suelo FC-28 es un sensor que mide la humedad del suelo. Son ampliamente empleados en sistemas automáticos de riego para detectar cuando es necesario activar el sistema de bombeo. El FC-28 es un sensor sencillo que mide la humedad del suelo por la variación de su conductividad. No tiene la precisión suficiente para realizar una medición absoluta de la humedad del suelo, pero tampoco es necesario para controlar un sistema de riego. Los valores obtenidos van desde 0 sumergido en agua, a 1023 en el aire (o en un suelo muy seco). Un suelo ligeramente húmero daría valores típicos de 600-700. Un suelo seco tendrá valores de 800-1023.
>
> Luis Llamas CC-NC-BY-SA https://www.luisllamas.es/arduino-humedad-suelo-fc-28/

Se puede utilizar este sensor para hacer un dispositivo de riego automático, puede detectar si las plantas “tienen sed” y evitar que se marchiten.
La corriente de trabajo del sensor es menor de 20mA. El voltaje de salida es de 0V (en el aire) a 2,3V (totalmente sumergido en agua).

- Smart Agriculture Kit micro:bit




## Sensor de humedad.

Este sensor **analógico** está diseñado para identificar y detectar la presencia de agua y su cantidad. Puede servir para detectar el nivel de agua, para disparar una alarma en caso de una fuga de agua, también para hacer un limpiapalabrisas automático.... puedes ver un ejemplo de uso en :

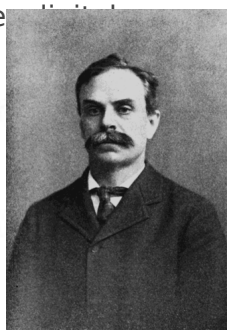
- Curso de Domótica con Arduino
- Smart Agriculture Kit micro:bit

Mide el volumen de agua caída a través de una serie de rastros de cables paralelos expuestos.



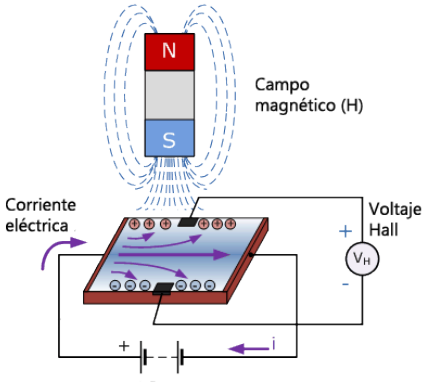
## Sensor de efecto Hall.

Este es un sensor de inducción magnética. Detecta los materiales magnéticos dentro de un rango de detección de hasta 3 cm. El  rango de detección y la fuerza del campo magnético son proporcionales. La salida e



Sensor Hall. *Edwin Helber Hall De Desconocido - Popular Science Monthly Volume 64, Dominio público*

Edwin Helbert Hall descubrió en 1879 que en presencia de un campo magnético, un conductor que conduzca una corriente se le producía un campo eléctrico porque las cargas eléctricas se desviaban de su trayectoria principal, nuestro sensor simplemente mide ese campo eléctrico:




De Luis Llamas CC-BY-NC

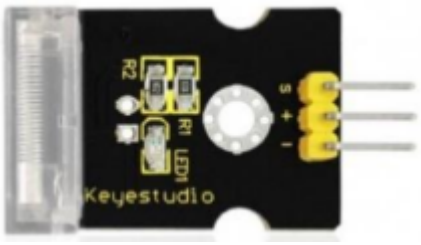
El sensor tiene un led de color rojo que indica que hay una lectura de campo magnético. Un ejemplo de uso lo puedes ver aquí: medir rocas magnéticas con el Rover con Arduino

## Sensor inclinación

Este sensor funciona al hacerle vibrar, emitiendo una señal digital de todo o nada. El módulo del sensor viene provisto de un potenciómetro para poder regularlo.

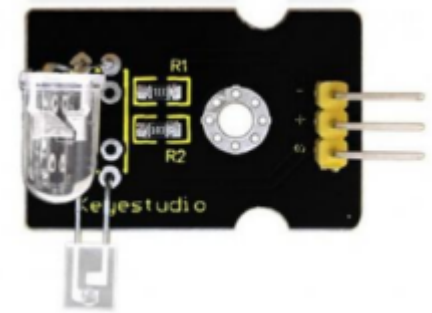

## Sensor de golpe

Es un sensor digital que al ser golpeado este sensor envía una señal momentánea.




## Sensor de pulso cardíaco.

Este módulo utiliza un LED infrarrojo (IR) ultrabrillante y un fototransistor para detectar el pulso en el dedo. Principio de funcionamiento: Se debe colocar el dedo entre el LED infrarrojo ultrabrillante (parte superior) mientras que el fototransistor, que queda en el otro lado, recoge la cantidad de luz transmitida. La resistencia del fototransistor variará levemente a medida que la sangre pase a través de su dedo.

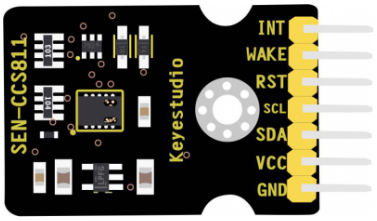



## Sensor de Alcohol.

Este sensor de gas analógico MQ-3 es adecuado para detectar alcohol. Se puede usar en un analizador de aliento. También tiene una alta sensibilidad al alcohol y baja sensibilidad a la bencina (éter de petróleo). La sensibilidad se puede ajustar con el potenciómetro.

## Sensor de CO2

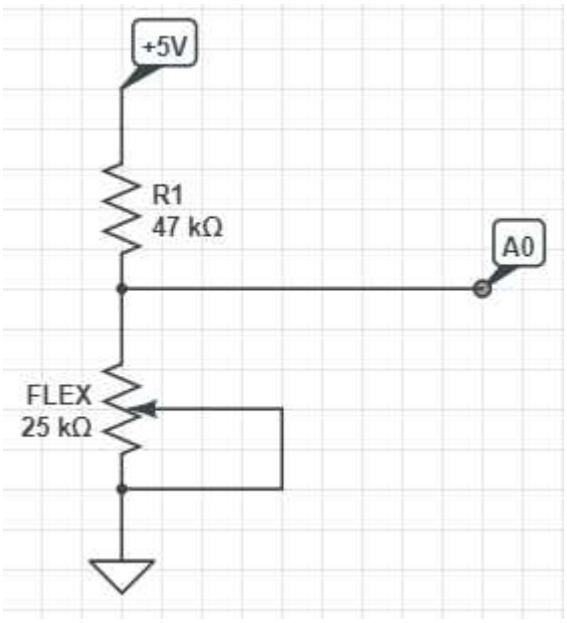
Hay sensores que utilizan **el protocolo I2C,** este protocolo permite conexiones serie y pueden compartir el mismo cable pues cada elemento tiene una dirección diferente. Esto lo veremos en el Display LCD. Se identifican por los pinen SDA y SCL




## Resistencia Flex

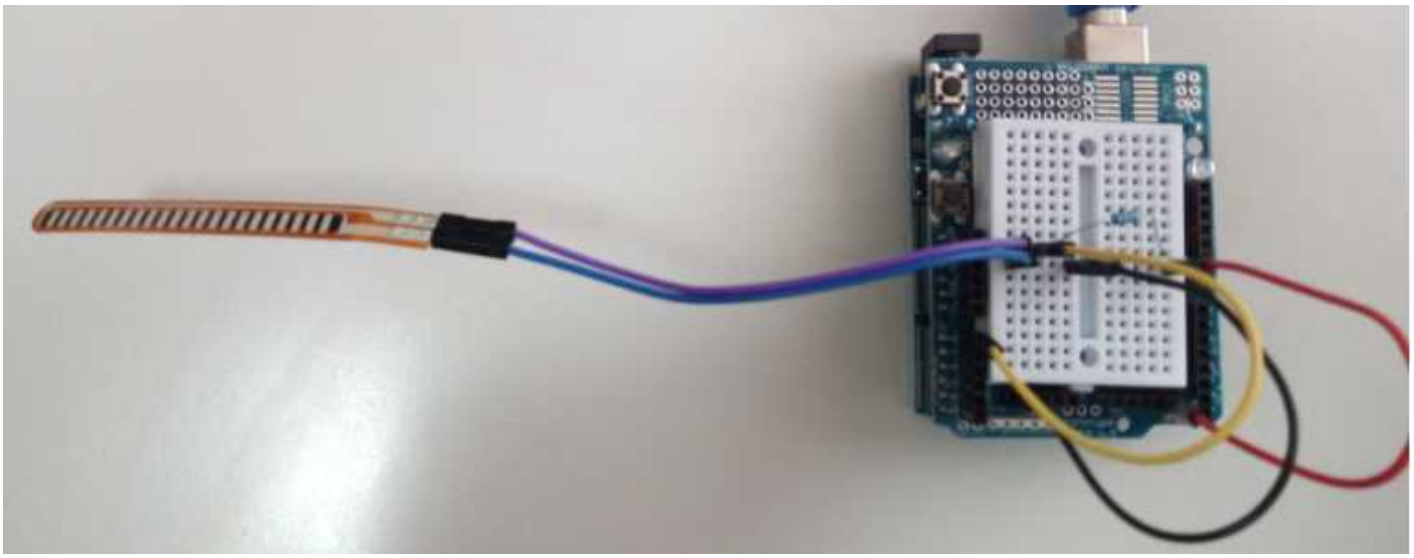
Es una resistencia que cuanto más se dobla más resistencia ofrece, desde 25k hasta 125k

Para utilizar esta resistencia haremos un **DIVISOR DE TENSIÓN** que consistirá en poner dos resistencias en serie y repartirá la tensión total entre 0V y 5V en las dos resistencias, **el punto medio** será un punto que tendrá una tensión variable en función de las dos resistencias, como la es variable, esa tensión es variable y ya tenemos la entrada **analógica**:




Es decir:

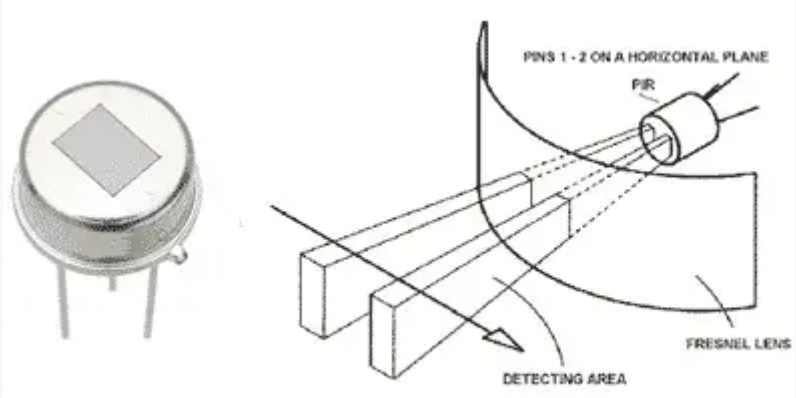
- La resistencia entre masa GND del ARDUINO (cable negro) y un punto en la placa protoboard
- ese punto medio conectarlo a una entrada analógica, por ejemplo A0 (cable amarillo)
- Una resistencia de valor parecida a la Flex de decenas de K entre ese punto y +5V (cable rojo en la foto)



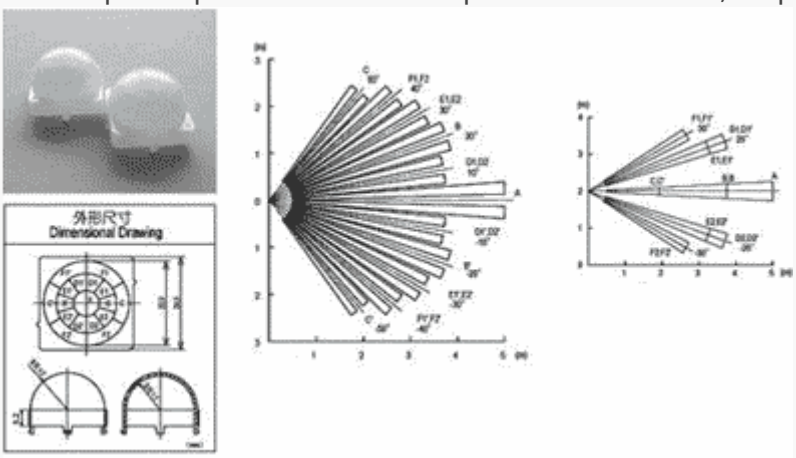
Este sensor tiene posibilidades para usarlo en "ropa inteligente".

## Sensor de movimiento

> Los sensores infrarrojos pasivos (PIR) son dispositivos para la detección de movimiento. Son baratos, pequeños, de baja potencia, y fáciles de usar. Por esta razón son frecuentemente usados en juguetes, aplicaciones domóticas o sistemas de seguridad.
> Los sensores PIR **se basan en la medición de la radiación infrarroja**. Todos los cuerpos (vivos o no) emiten una cierta cantidad de energía infrarroja, mayor cuanto mayor es su temperatura. Los dispositivos PIR disponen de un sensor piro eléctrico capaz de captar esta radiación y convertirla en una señal eléctrica.
> En realidad **cada sensor está dividido en dos campos** y se dispone de un circuito eléctrico que compensa ambas mediciones. Si ambos campos reciben la misma cantidad de infrarrojos la señal eléctrica resultante es nula. Por el contrario, si los dos campos realizan una medición diferente, se genera una señal eléctrica.
> De esta forma, si un objeto atraviesa uno de los campos se genera una señal eléctrica diferencial, que es captada por el sensor, y se emite una señal digital.



El otro elemento restante para que todo funcione es **la óptica del sensor**. Básicamente es una cúpula de plástico formada por lentes de fresnel, que divide el espacio en zonas, y enfoca la radiación infrarroja a cada uno de los campos del PIR.
De esta manera, cada uno de los sensores capta un promedio de la radiación infrarroja del entorno. Cuando un objeto entra en el rango del sensor, alguna de las zonas marcadas por la óptica recibirá una cantidad distinta de radiación, que será captado por uno de los campos del sensor PIR, disparando la alarma.



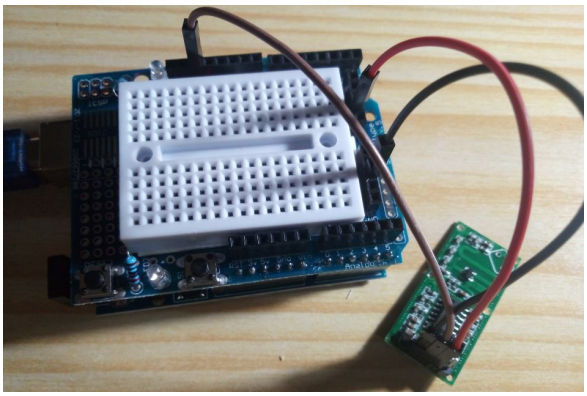

Luis Llamas CC-BY-NC-SA https://www.luisllamas.es/detector-de-movimiento-con-arduino-y-sensor-pir/

Puedes ver ejemplos de uso en robótica en :

- Smart Home para Microbit
- Smart Agriculture Kit para Microbit


Más sensibles son los **sensores de microondas**. Son un radar que por efecto Doppler pueden captar cualquier objeto en movimiento dentro de un alcance de 5-7 metros en cualquier dirección e independiente de su temperatura. Es un buen sensor para alarmas, activación de luz por presencia..... Para saber más ver la página de Luis Llamas

Su conexión es muy sencilla, es un detector digital que hay que alimentarlo como el resto de sensores.

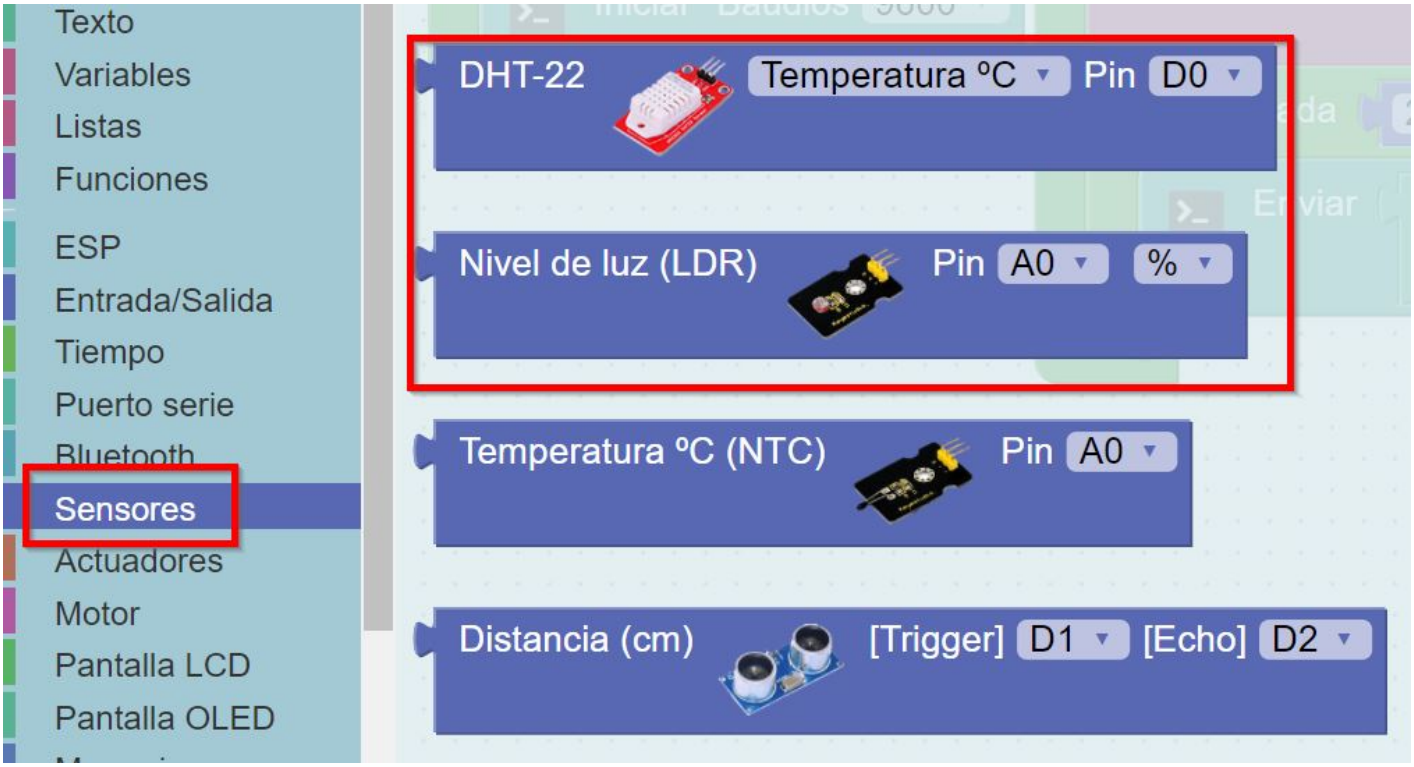


> Curiosamente, la gran ventaja técnica de los de microondas **es un gran inconveniente para usarlo en el aula**, con cualquier movimiento se dispara, luego **para clase es mejor el sensor PIR**

> Esta página esta adaptada de *este enlace. José Andrés Echevarría @cantabRobots CC-BY-NC-SA.*

# Sensores del rover Arduino

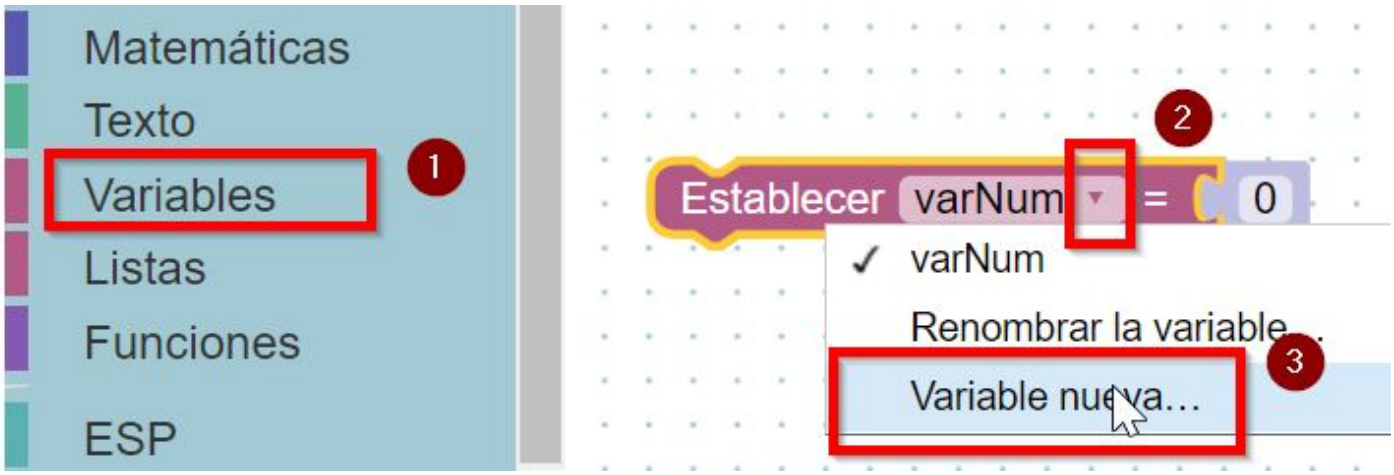
La ventaja de ARDUINOBLOCKS es que tiene ya incorporado una cantidad de bloques con sus librerías y funciones integradas por lo que el alumno no tiene que pelear con códigos y situaciones especiales, simplemente arrastrar el bloque y funcionar:




# Probando el sensor LDR.

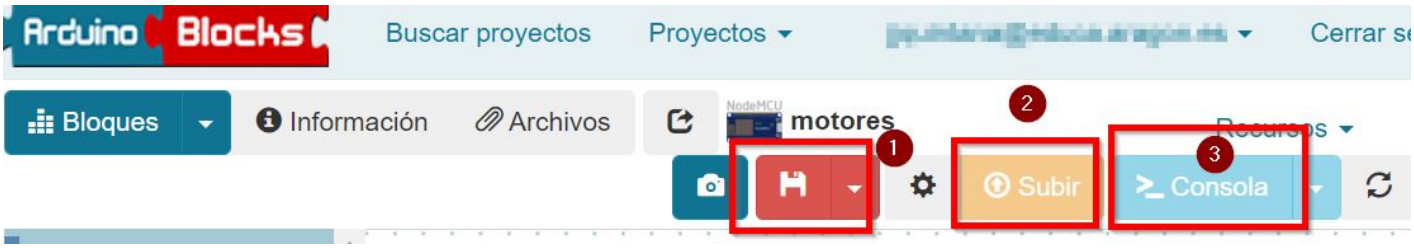
Vamos a probar este sensor que está conectado en una entrada analógica, para ello simplemente que escriba su valor en el puerto, y veremos cómo va cambiando.

Es recomendable para la lectura de los sensores, crear variables que almacenen los valores leídos, para luego utilizar estos valores:



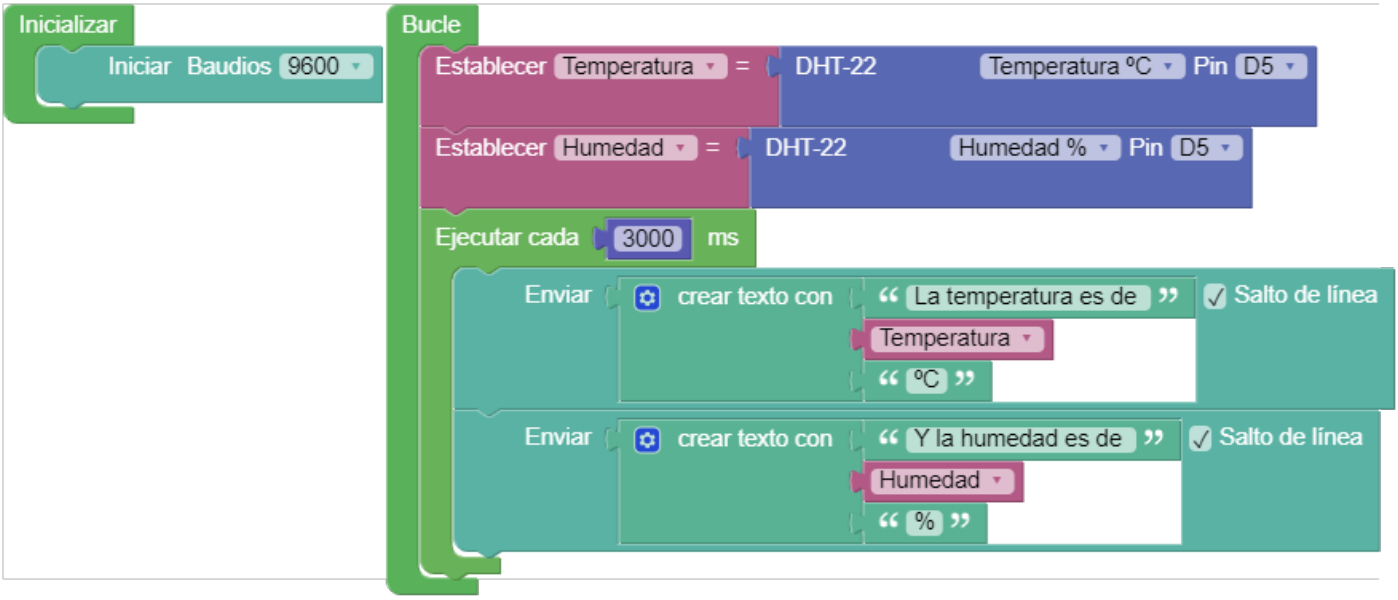

El siguiente código lee el sensor cada 2 seg. y lo vemos por el puerto serie. Podemos comprobar que **cuanto más luz, la lectura es menor.**


Guardamos el proyecto, lo subimos al Arduino y pulsamos en consola para ver cómo cambia los valores.




# Probando el sensor DHT22.

Vamos a probar este sensor, que mide a la vez temperatura y humedad. Con el bloque correspondiente, nos ahorramos bastante código.




# Probando el sensor Hall.

Este sensor tiene también su bloque :


Podemos construir un programa análogo al anterior y veremos que su salida es simplemente 0 o 1 depende si acercamos o no un imán.

# Actuadores y otras salidas

# Motores en el rover Arduino

Los motores **derechos** se gobiernan con la siguiente instrucción :


- Activando el pin D1 encendemos los motores
- El pin D3 en ON van hacia delante, si queremos que vayan hacia atrás, ponemos D3 en OFF
- El valor PWM del pin D1 es la potencia que transferimos al motor, puede ser desde 0 hasta 1023

Recomendamos leer esta página sobre el significado de las salidas PWM.

Para los motores **izquierdos** se gobiernan con la siguiente instrucción :


- Activando el pin D2 encendemos los motores
- El pin D4 en ON van hacia delante, si queremos que vayan hacia atrás, ponemos D4 en OFF
- El valor PWM del pin D2 es la potencia que transferimos al motor, puede ser desde 0 hasta 1023.

Recomendamos ejecutar estas instrucciones y jugar con estos valores para ver si están bien conectados los motores :