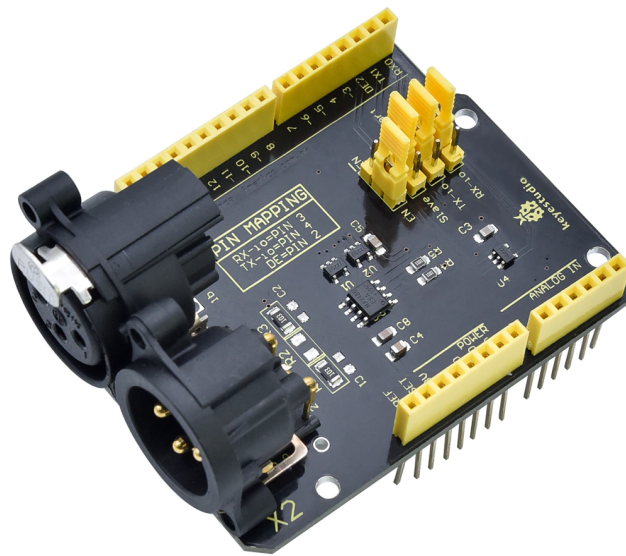


# DMX



## [Keyestudio DMX-Shield](#)

Arduino UNO  
ESP32 STEAMakers

# ÍNDICE

[1.-Introducción](#)

[2.-Configuración de la DMX-Shield](#)

[2.1.-Configuración para Arduino UNO:](#)

[2.2.-Configuración para ESP32 STEAMakers:](#)

[3.-Programación con ArduinoBlocks](#)

[3.1.-Inicializar shield](#)

[3.2.-DMX-Slave \(receptor\) -> Leer datos de los canales DMX](#)

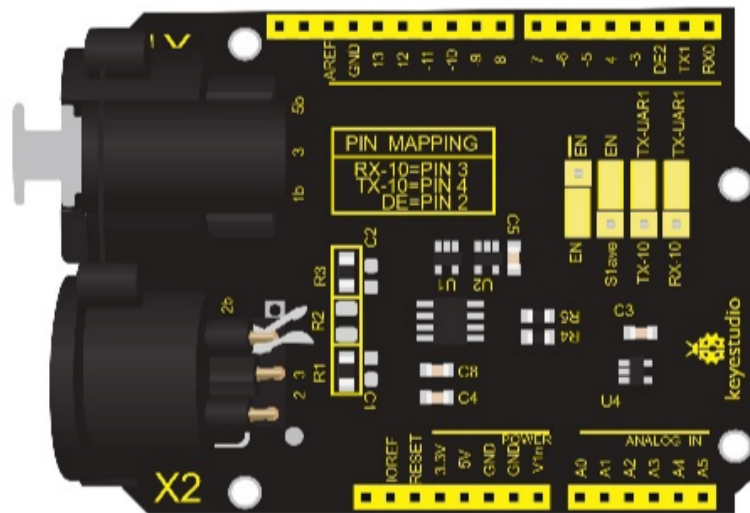
[3.3.-DMX-Controller \(emisor\) -> Escribir datos en los canales DMX](#)

[4.-Proyecto de ejemplo](#)

# 1.-Introducción

La DMX-Shield de keystudio nos permite añadir comunicación compatible con el DMX (DMX512) basada en el protocolo de comunicación RS485.

La shield puede utilizarse en ArduinoBlocks con proyectos tipo Arduino UNO y ESP32 STEAMakers.

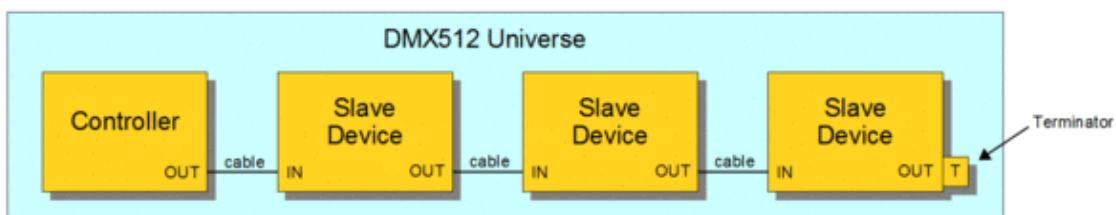


Información técnica sobre la shield:

[https://wiki.keystudio.com/KS0529\\_Keyestudio\\_DMX%EF%BC%88RDM%EF%BC%89Shield](https://wiki.keystudio.com/KS0529_Keyestudio_DMX%EF%BC%88RDM%EF%BC%89Shield)

El protocolo DMX permite comunicar dispositivos (normalmente utilizado para espectáculos visuales, iluminación, robótica, etc.) en la que un único dispositivo será el controlador DMX y el resto trabajarán como esclavos DMX.

El protocolo DMX envía 512 canales de datos, cada canal es un byte (0 a 255). La shield DMX conectada a Arduino o ESP32 podrá configurarse como controlador o como esclavo.



La propia shield incorpora las resistencias de terminación del bus de forma que no es necesario añadirlas.

En caso de trabajar como controlador podrá escribir en cada uno de los 512 canales y generará el envío periódico de la trama de datos con todos los canales según el protocolo DMX para el resto de dispositivos.

En caso de trabajar como esclavo recibirá los 512 canales y se podrán obtener todos los valores del bus DMX. Será el diseñador el que fija que canales son los responsables de actuar sobre el dispositivo en cuestión.

Más información sobre DMX-512

<https://hmong.es/wiki/DMX512-A>

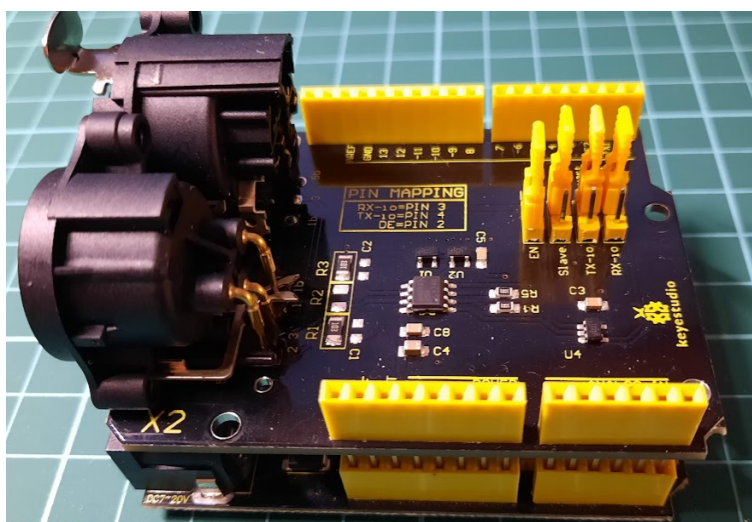
<https://en.wikipedia.org/wiki/DMX512>

## 2.-Configuración de la DMX-Shield

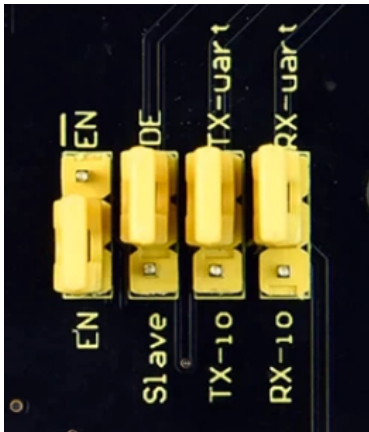
La shield dispone de unos “jumpers” de configuración para indicar los pines que se utilizarán de la placa (Arduino o ESP32) para la comunicación (RX/TX) y para la habilitación de la placa (EN).

### 2.1.-Configuración para Arduino UNO:

En Arduino UNO sólo podemos usar los pines RX,TX hardware (pines 0,1) pues la velocidad necesaria de comunicación no podemos implementarla por software en otros pines y obligatoriamente necesitamos usar el único puerto serie hardware disponible.



Por tanto los “jumpers” de la shield DMX se quedarán así:



**IMPORTANTE:**

A la hora de programar la placa , ya que la shield interferirá en el proceso de subida del programa (por estar conectada a los pines 0,1) debemos o quitar la shield, o simplemente quitar el jumper “RX” durante el proceso de subir el programa para que no interfiera en la comunicación serie. Una vez programada se vuelve a poner el “jumper” y funciona.

Debido a esta limitación nuestros programas de Arduino UNO no podrán usar tampoco los bloques de puerto serie, pues éste se estará usando para la comunicació con el bus DMX.

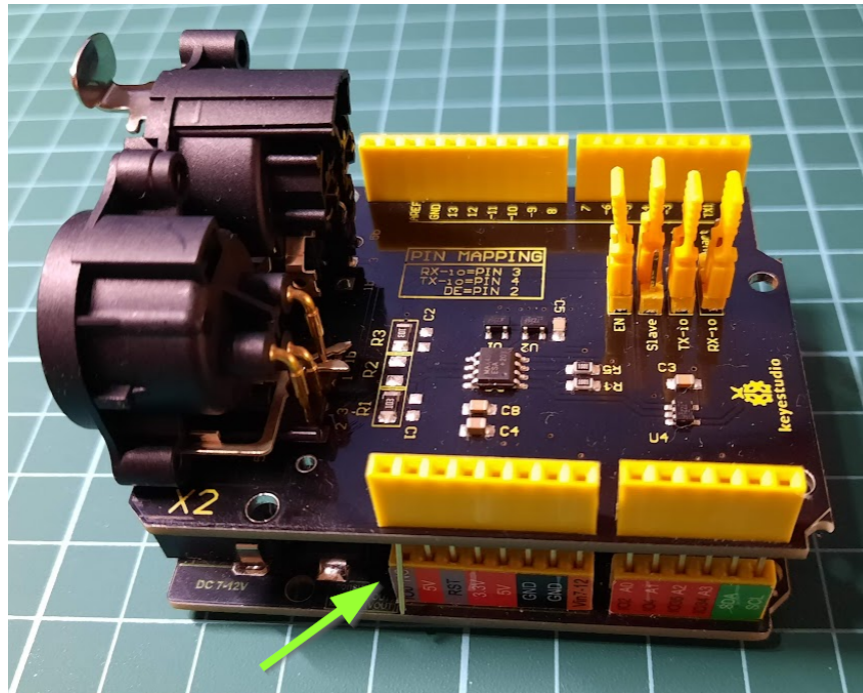
**2.2.-Configuración para ESP32 STEAMakers:**

En la placa ESP32 STEAMakers podemos usar la configuración de “jumpers” alternativa que utiliza otros pines (gpio) para la comunicación dejando libre los pines RX/TX del puerto serie y así evitando el conflicto a la hora de subir el programa.

**IMPORTANTE:**

Debemos tener precaución con la shield respecto al pin IO0 de la ESP32 STEAMakers, pues ese pin configura la placa ESP32 en modo BOOT y la placa se queda “colgada” después de subir el programa.

Para evitar este problema podemos o bien “cortar” la patilla (no tiene ninguna utilidad si no vamos a usar el pin IOREF en Arduino UNO) o bien “dejarla fuera” de la placa para evitar que el micro ESP32 se configure en modo BOOT.

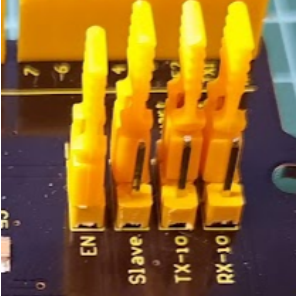

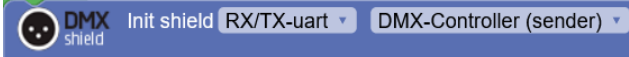
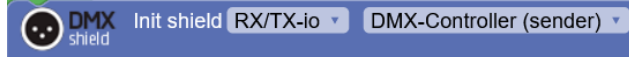


### 3.-Programación con ArduinoBlocks



### 3.1.-Inicializar shield

El bloque inicializar deberá seleccionar el modo en que hemos configurado los “jumpers”, en el caso de Arduino UNO solo tenemos una posibilidad y con ESP32 STEAMakers es recomendable usar los pines gpio alternativos para no interferir con el puerto serie a la hora de programar la placa y también poder usar la consola serie en nuestros programas.

<i>Pines puerto serie hardware (única opción en Arduino UNO)</i>	<i>Pines gpio alternativos (opción recomendada en ESP32)</i>
	
	

Modo:

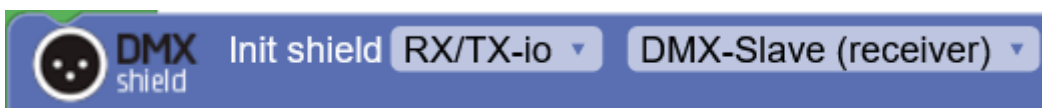
**DMX-Controller:** Envía los datos DMX para controlar otros dispositivos, solo debe haber un dispositivo controlador en el bus,



*(habitualmente el controlador del bus DMX es una mesa de control DMX o un PC con interfaz DMX)*

[https://www.thomann.de/es/consolas\\_de\\_iluminacion\\_dm.html](https://www.thomann.de/es/consolas_de_iluminacion_dm.html)

**DMX-Slave:** Dispositivo que recibe datos DMX del bus, pueden haber múltiples dispositivos esclavos en el bus.





### 3.2.-DMX-Slave (receptor) -> Leer datos de los canales DMX

En caso de iniciar la shield en modo DMX-Slave podremos acceder a los datos recibidos del bus DMX.

Debemos especificar el número de canal (de 1 a 512) y obtendremos un valor (byte) del canal correspondiente (0..255)



### 3.3.-DMX-Controller (emisor) -> Escribir datos en los canales DMX

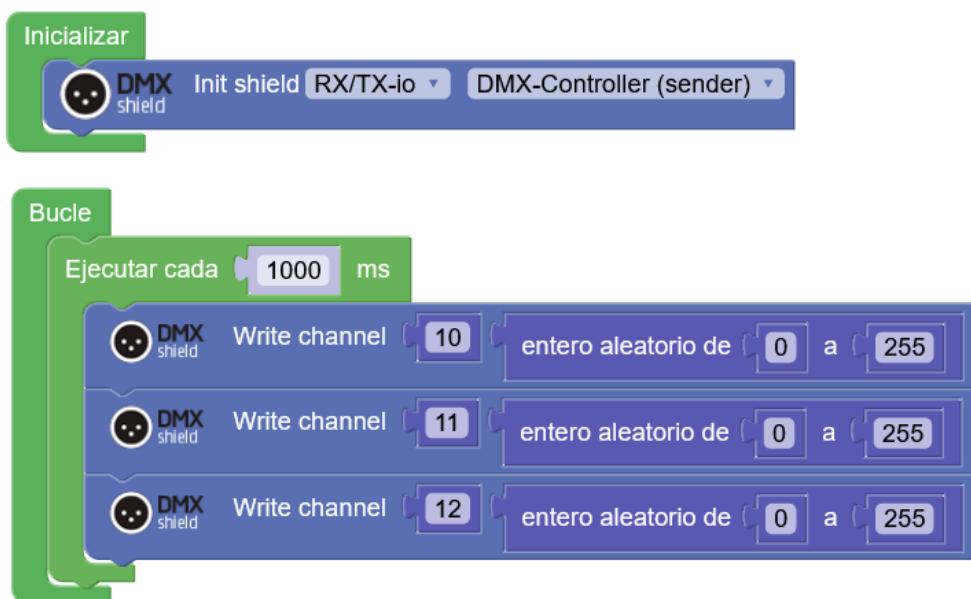
En caso de iniciar la shield en modo DMX-Controller podremos escribir los datos que se enviarán a través del bus al resto de dispositivos DMX.

Debemos especificar el número de canal (de 1 a 512) y un valor (byte) para el canal (0..255)



## 4.-Proyecto de ejemplo

-Controlador: envía valores aleatorios en los canales 10,11,12





**-Esclavo (Receptor):** controla un led RGB con los valores recibidos en los canales 10,11,12  
(el led RGB se conecta a los pines 9,10,11)

