Retos

Copyright 2026

Reto A01. El led

Estos contenidos han sido elaboradas por Fernando Hernández García, Ingeniero Técnico Industrial Especialidad Electrónica, formador del profesorado y profesor del Institut Torre del Palau (Terrassa – Barcelona). *Enlace de los contenidos*. ***Licencia CC-BY-NC-ND***.

Permiso

Vamos a empezar con nuestro primer reto. Vamos a realizar un programa que va a encender y apagar el led rojo correspondiente al pin D12.

Un **LED** (Diodo Emisor de Luz) es un diodo semiconductor que emite luz. Se usan como indicadores en muchos dispositivos, y cada vez con mucha más frecuencia en iluminación. Los leds presentan muchas ventajas sobre las fuentes de luz incandescente como un consumo de energía mucho menor, mayor tiempo de vida, menor tamaño, gran durabilidad y fiabilidad.

El led tiene una polaridad, un orden de conexión, y al conectarlo al revés no funciona. Para evitar que se queme siempre debe llevar una resistencia en serie para limitar la corriente eléctrica que le atraviesa.

image-1654079285660.022.png

La placa Imagina TDR STEAM dispone de dos leds (uno azul y otro rojo), conectados en los pines D13 (azul) y D12 (rojo).

image-1654079314970.62T3M1.png

7.1.1 Reto A01.1. ON/OFF led rojo.

Entramos en ArduinoBlocks en el tipo de proyecto para la placa Imagina TDR STEAM. En la columna de la izquierda tenemos las agrupaciones de bloques clasificados en función de su tipología.

En el área de programación siempre hay dos bloques verdes (Inicializar y Bucle). Estos bloques siempre aparecen al iniciar un nuevo programa. Pues bien, todo lo que se meta dentro del bloque de *Inicializar* sólo se ejecutará la primera vez que se inicie el programa, mientras que si se colocan dentro del Bucle se ejecutarán una y otra vez hasta que apaguemos la placa.

image-1654079619989.png_

image-1654079635691.png

Seleccionaremos el bloque de *TDR STEAM*. Vamos a meter nuestro bloque de led en el Bucle y elegimos el color del led (Rojo o Azul). El led puede tener dos estados: ON/OFF (encendido/apagado), que podemos cambiar en el menú despegable.

Si sólo dejamos este bloque con el led en estado ON, este quedaría encendido para siempre, para que se apague deberemos cambiar el estado a OFF.

image-1654079654890.027.png

Pero este programa no es correcto del todo. No hay tiempos que indiquen cuanto tiempo tiene que estar encendido o apagado el led. Necesitamos ir al bloque de *Tiempo*** y seleccionar *Esperar (valor) milisegundos* (recuerda: 1.000 milisegundos es 1 segundo).

image-1654079684925.png

image-1654079701094.029.png

Ahora tenemos el led encendido durante 1 segundo y apagado otro. Esto ser repetirá por tiempo infinito hasta que quitemos la alimentación a la placa. El programa se quedará guardado en la memoria del microcontrolador, así que, si lo alimentamos con una fuente de alimentación externa, seguirá funcionando.

Actividad de ampliación: prueba ahora de hacer una intermitencia más rápida (500ms ON y 250ms OFF).

7.1.2 Reto A01.2. ON/OFF led rojo y azul

Como hemos comentado anteriormente la placa dispone de dos leds (rojo y azul). Ahora vamos a realizar un programa para que se vayan alternando su encendido y apagado.

image-1654079723058.030.png

> Actividad de ampliación: prueba ahora de hacer que los leds rojo y azul se enciendan a la vez y con los siguientes tiempos: 500ms ON y 250ms OFF.

Copyright 2026

7.1.3 Reto A01.3. ON/OFF led rojo y azul con repeticiones.

Imagina que queremos hacer un ciclo de repetición. Queremos repetir 5 veces el encendido y apagado del led rojo antes de que se encienda el azul. Para realizar esta acción lo podemos hacer de la siguiente forma: en el menú de *Control* existe el bloque *Repetir (valor) veces hacer...*

image-1654079804312.031.png

En el siguiente programa fíjate como el led rojo se enciende y apaga cada medio segundo (500ms) 5 veces y después se queda el led azul encendido durante 4 segundos (4000ms).

image-1654079839248.png

> Actividad de ampliación: prueba ahora de hacer que el led rojo se encienda 10 veces cada vez que el led azul se enciende 3 veces.

Copyright 2026

Reto A02. El led RGB

Estos contenidos han sido elaboradas por Fernando Hernández García, Ingeniero Técnico Industrial Especialidad Electrónica, formador del profesorado y profesor del Institut Torre del Palau (Terrassa – Barcelona). *Enlace de los contenidos*. ***Licencia CC-BY-NC-ND***.

Permiso

Un **led RGB** es un led que incorpora en su mismo encapsulado tres leds. Las siglas RGB corresponden a: R (Red=rojo), G (Green=verde) y B (Blue=azul). Con estos tres colores, en óptica, se puede formar cualquier color, ajustando de manera individual la intensidad de cada color. Los tres leds están unidos por el negativo o cátodo (RGB de cátodo común).

image-1654080228442.033.pngimage-1654080238074.034.png

En Arduino, cada uno de esos leds podría tomar 256 colores diferentes, es decir, el Rojo podría ir desde 0 hasta 255, el Verde de 0 a 255 y el Azul de 0 a 255, en total un led RGB podría dar más de 16,5 millones de colores diferentes.

image-1654080262518.png

La placa **Imagina TDR STEAM** dispone de un led RGB conectado a los pines (D6-Red, D9-Green y D10-Blue). Estos tres pines son PWM para permitir regular su intensidad.

La modulación PWM permite generar una señal analógica mediante una salida digital. Utiliza un sistema de codificación de 8 bits (en el sistema binario: 28 = 256, per tanto, del 0 al 255).

image-1654080900430.png

Continuando con la práctica anterior, ahora vamos a controlar la intensidad de un led utilizando la modulación PWM. Pero antes vamos a explicar cómo funciona la modulación PWM. PWM es la abreviatura de ***Pulse-Width Modulation*** (modulación de ancho de pulso*)*. Las salidas digitales de Arduino sólo tienen dos estados: **ALTO/BAJO**, **ON/OFF**, **ENCENDIDO/APAGADO**. Es decir, corresponden a una salida de 5 V (ON) y de 0 V (OFF). Con esto sólo podemos hacer actividades de encender y apagar un led, no podríamos controlar su brillo de menos a más o viceversa. Esto lo realiza por la proporción entre el estado alto (ON) y bajo (OFF) de la señal. El control digital se

Copyright 2026

utiliza para crear una onda cuadrada de ciclo de trabajo diferente, una señal conmutada entre encendido y apagado. Este patrón de encendido y apagado puede simular voltajes entre encendido total (5 voltios) y apagado (0 voltios) al cambiar la parte del tiempo que la señal pasa en comparación con el tiempo que la señal pasa. La modulación de ancho de pulso, o PWM, es una técnica para obtener resultados analógicos con medios digitales. Se utiliza mucho para controlar leds, velocidades de motores, producción de sonidos, etc.

image-1654080389894.IJ51M1.png

La placa Keyestudio UNO tiene un total de 6 salidas PWM, que son digitales 3, 5, 6, 9, 10 y 11, pero en la placa **Imagina TDR STEAM** sólo se puede controlar por PWM el led RGB (pines 6, 9 y 10). Tenemos dos bloques diferentes para regular el color:

- Asignaremos directamente el color en la paleta de colores:
  image-1654080400111.EB2AN1.png

- Introduciremos la cantidad de cada uno de los tres colores primarios (R rojo, G verde, B azul) con un valor comprendido entre 0 i 255.

image-1654080407936.4MV7M1.png

7.2.1 Identificación de colores RGB.

Con este sencillo programa vamos a identificar cada color del led RGB con su pin correspondiente. Para ello vamos a utilizar los bloques normales de ArduinoBlocks (no los específicos de la Imagina TdR STEAM).

Encendemos sólo el led correspondiente al Pin 6.

Encendemos sólo el led correspondiente al Pin 9.

Encendemos sólo el led correspondiente al Pin 10.

image-1654080497350.DHZXM1.png

> Actividad de ampliación: prueba ahora de ir activando varios leds a la vez para ver qué color aparece (puedes guiarte con la imagen siguiente).

7.2.2 Múltiples colores con el led RGB

ArduinoBlocks tiene bloques específicos para facilitar al máximo la programación de los leds RGB. Al utilizar ese bloque no debemos preocuparnos por saber las conexiones del led RGB porque ya están asignadas internamente en el bloque.

Comprueba como al pulsar sobre el icono del color se despliega una paleta de colores para que puedas elegir cualquier color.

image-1654080560701.Q05AN1.png

También se puede introducir directamente los números de cada uno de los colores RGB.

image-1654080570481.1HJDN1.png

Vamos a realizar ahora un programa que muestre sucesivamente los tres colores primarios.

image-1654080578279.32YTM1.png

Ahora realizaremos el mismo programa, pero poniendo la cantidad de cada color.

image-1654080588775.XYGAN1.png

> Actividad de ampliación: prueba ahora de hacer un programa que muestre los colores del arcoíris en orden.

Copyright 2026

Reto A03. El pulsador

Estos contenidos han sido elaboradas por Fernando Hernández García, Ingeniero Técnico Industrial Especialidad Electrónica, formador del profesorado y profesor del Institut Torre del Palau (Terrassa – Barcelona). *Enlace de los contenidos*. ***Licencia CC-BY-NC-ND***.

Permiso

En el siguiente reto vamos a utilizar el pulsador. Previamente debemos recordar que diferencia hay entre un pulsador y un interruptor. Un interruptor es un dispositivo que abre o cierra en paso de la corriente eléctrica, por ejemplo, los interruptores de la luz de nuestras casas, cada vez que los pulsamos cambian de estado y permanecen en él hasta ser pulsados de nuevo. Sin embargo, un pulsador sólo se activa mientras dure la pulsación volviendo a su estado inicial en el momento en el que se deje de pulsar.

image-1654081338195.M5BVM1.png

Hay dos tipos de pulsadores; los NA (normalmente abierto) o los NC (normalmente cerrado), con lo que al pulsarlo se activará la función inversa de la que en ese momento este realizando.

image-1654081373685.png

La placa Imagina TDR STEAM tiene dos pulsadores de nominados SW1 y SW2 que van asociados a los pines D2 y D7 respectivamente.

Ahora vamos a realizar un programa en el cual al pulsar sobre el pulsador se encienda el led y se apague cuando lo dejemos de pulsar. En el menú de *Sensores* encontramos los dos bloques correspondientes al pulsador y el pulsador filtrado.

image-1654081394030.png

7.4.1 Control ON/OFF de un led con un pulsador I

Para realizar este programa necesitamos conocer unas de las funciones más utilizadas en programación. Las funciones del menú *Lógica* con las funciones de condición (condicionales).

Condicionales:

image-1654081437761.31I3M1.png *si (condición) hacer (acciones)*

Se trata del famoso bucle *Si* (*if* en inglés) que es uno de los pilares de la programación, ya que permite evaluar estados y tomar decisiones en consecuencia.

El funcionamiento es el siguiente: si se cumple la condición incluida en su primer apartado, entonces se realiza la acción incluida en su segundo aparado. En caso contrario, no se hace nada.

image-1654081470378.png

En el apartado de condiciones se pueden introducir multitud de factores: estado de sensores (analógicos o digitales), comparaciones, igualdades, operaciones matemáticas, etc.

Usando el bloque lógico *condicional de Si.... hacer...* el programa quedaría como la imagen.

image-1654081532620.TR99M1.png

No se apaga el led azul nunca, esto no es lógico, en ningún momento del programa decimos que el led tenga que estar en la posición OFF. En el siguiente programa conseguiremos que el led rojo solamente esté encendido cuando apretemos el pulsador 1.

image-1654081543567.CKI0M1.png

Con este otro programa al pulsar el SW1 se enciende el led azul y al pulsar el SW2 se apaga.

image-1654081552529.OEUBN1.png

> Actividad de ampliación: prueba ahora de apaga los dos leds con el pulsador SW1 y encenderlos con SW2.

7.4.2 Control ON/OFF de un led con un pulsador II

Necesitar dos pulsadores para tener que controlar un solo Led no parece la mejor opción, necesitamos introducir un sino. Si usamos un único pulsador debemos poner una nueva condición: un *sino* para cambiar el estado del led a OFF. Para ello debemos ampliar el condicional. Pulsando

sobre el símbolo del engranaje (señalado con la flecha roja en la imagen) nos aparece un cuadro con funciones con las que podemos ampliar el condicional *Si*.

image-1654081609478.png

Hay dos opciones que se consiguen arrastrando los bloques como se aprecia en las siguientes imágenes:

image-1654081627304.png

Podemos ir encadenando varias estructuras (opción B).

Veremos ejemplos del uso de estas variantes a lo largo de diferentes programas en este documento. Pero, continuando con este reto, vamos a realizar un programa que al pulsar el pulsador 2 se encenderá el led azul y *sino*, se apagará.

image-1654081637415.QMM5M1.png

El código Morse es un sistema de representación de letras y números mediante señales emitidas de forma intermitente. Estas señales pueden ser luminosas como con nuestro Led o también podrían ser acústicas utilizando el zumbador.

Este es el código Morse:

image-1654081653294.2D6DN1.png

> Actividad de ampliación: haz una máquina de código Morse que envíe un mensaje de ayuda (SOS).

7.4.3 Control ON/OFF de un led con un pulsador III

En este reto vamos a hacer que el Led Azul se quede en estado encendido o apagado pulsando una sola vez el pulsador SW1.

En el programa que vamos a hacer a continuación necesitamos asegurarnos que cada vez que accionamos el pulsador sea detectado como una sola señal, ya que vamos a realizar un contador. Para hacer el contador vamos a utilizar una variable que la llamaremos *Estado*.

image-1654081692820.38E0M1.png

En *Inicializar* establecemos la variable a 0 y la llamamos *ESTADO*. Esta variable la utilizaremos como indicador de cambio de estado del pulsador.

image-1654081701654.YBCUM1.png

El contador consiste en ir sumando una unidad a la variable *ESTADO* cada vez que se dé al pulsador. Cada vez que pulsemos el pulsador, la variable tendrá su valor anterior más 1. Por ejemplo: si al inicio del programa la variable *ESTADO* tiene un valor de 0, al dar al pulsador su nuevo valor será igual a 0+1=1; al volver a dar al pulsador su valor será 1+1=2; al volver a pulsar 2+1=3, etc.

image-1654081713488.S8RAN1.png

Este programa irá incrementando el valor indefinidamente cada vez que apretemos el pulsador. Pero vamos a hacer que se reinicie la variable cada vez que llegue al valor 2, es decir, que vuelva a poner el valor de la variable a 0. De esa manera, cuando *Estado* tiene un valor de 0 el Led estará apagado, cuando *ESTADO* tenga un valor de 1 el Led estará encendido y cuando se vuelva a dar al pulsador el valor de *ESTADO* será 2 y se le mandará volver a un valor de 0 por lo que el Led estará apagado nuevamente. Este sería el programa:

image-1654081726457.88HDN1.png

image-1654081723480.61D7M1.png

> Actividad de ampliación: intenta hacer el mismo programa, pero más simplificado (utilizando menos instrucciones).

Copyright 2026

Reto A04. El potenciómetro

Estos contenidos han sido elaboradas por Fernando Hernández García, Ingeniero Técnico Industrial Especialidad Electrónica, formador del profesorado y profesor del Institut Torre del Palau (Terrassa – Barcelona). *Enlace de los contenidos*. ***Licencia CC-BY-NC-ND***.

Permiso

Un potenciómetro es una resistencia cuyo valor es variable ya que son un tipo de resistencias especiales que tienen la capacidad de variar su valor cambiando de forma mecánica su posición. Con ellos indirectamente, se puede controlar la intensidad de corriente que fluye por un circuito si se conecta en paralelo, o controlar el voltaje al conectarlo en serie. Son adecuados para su uso como elemento de control en los aparatos electrónicos como el control de volumen, brillo, etc.

image-1654081830843.79L3M1.png

La placa Imagina TDR STEAM tiene un potenciómetro denominado *Rotation* que van asociado al pin A0. Las entradas *Anúmero* son entradas analógicas, así que empezamos con el uso de este tipo de entradas. Este potenciómetro permite realizar un giro de unos 270º entre topes (3/4 de vuelta).

image-1654081842559.png

La diferencia entre un sensor **analógico** y **digital** es que mientras este último, el digital, sólo permite dos tipos de entradas, 0-1, *alto-bajo*, *high-low*, *on-off*, un sensor analógico puede tener infinidad de valores. En Arduino, las entradas analógicas pueden tener 210 valores (10 bits de resolución), es decir, valores comprendidos entre 0 y 1023.

En el menú de sensores de ArduinoBlocks, disponemos de un bloque específico para realizar programas utilizando el potenciómetro de nuestra placa.

image-1654081855950.F0XAN1.png

En el desplegable del bloque del sensor, podemos elegir su lectura en porcentaje (%) o en valor (de 0 a 1023).

image-1654081861633.BDB3M1.png

7.5.1 Lectura de valores con el puerto serie

Para realizar una lectura de los valores del sensor es necesario utilizar la *Consola* (lector de datos por el puerto serie) que nos ofrece ArduinoBlocks, vamos a ver como se hace.

En primer lugar, generamos una variable a la que llamaremos *pot*.

image-1654081913047.10MTM1.png

Para cambiar el nombre de la variable pulsaremos sobre el menú desplegable del bloque de la variable y elegiremos *Variable nueva...* nos aparecerá una ventana en la que escribiremos el nuevo nombre y daremos a *Aceptar*. Ahora fijaremos el valor de la variable al valor del potenciómetro, tal y como está en la imagen.

image-1654081926365.ZSVVM1.png

Es importante establecer la variable con el valor del potenciómetro dentro de *Bucle*, ya que si sólo se hace en *Inicializar* el valor siempre será el mismo a lo largo de todo el programa. En otras ocasiones interesa establecer las variables en el inicio, pero no es este el caso.

Continuando con el programa, ahora nos faltan los bloques del *Puerto Serie*. El primero que debemos utilizar es el *Iniciar Baudios 9.600* que siempre lo colocaremos en el Inicio y después el bloque *Enviar.*

image-1654081966347.png

Observa cómo queda el programa resultante:

image-1654081976418.MCPDN1.png

Sube ahora el programa y después pulsa sobre el botón de la *Consola*.

image-1654081986506.HCX6M1.png

Se abrirá la siguiente ventana y pulsaremos sobre el botón conectar. De esta manera podremos ver cada medio segundo el valor de nuestro potenciómetro. Gira el potenciómetro y observa cómo van cambiando los valores.

image-1654081991794.AGYYM1.png

> Actividad de ampliación: prueba ahora quitando el tic de *Salto de línea* a ver qué sucede.

Copyright 2026

7.5.2 Ajuste de valores de entrada y salida: mapear

Existe un pequeño “problema” entre las entradas y las salidas en Arduino. Las entradas trabajan con 10 bits (210 valores = 0 a 1023) y las salidas trabajan a 8 bits (28 valores = 0 a 255). Debido a esto, debemos realizar un cambio de escala. A este cambio de escala se le llama “mapear”. En el menú *Matemáticas* existe un bloque llamado *mapear*. Este bloque permite modificar el rango de un valor o variable desde un rango origen a un rango destino. Esta función es especialmente útil para adaptar los valores leídos de sensores o para adaptar valores a aplicar en un actuador.

image-1654082047764.9XRYM1.png

En esta actividad vamos a imaginar que con el potenciómetro queremos definir un rango de valores entre 0 a 255. Para ello definiremos una variable, llamada *consigna*, que será el valor *mapeado* del potenciómetro. En el potenciómetro cambiaremos su opción para obtener datos 0...1023.

image-1654082057689.WSEAN1.png

Continuando el programa para poder realizar lecturas por el puerto serie utilizaremos un nuevo bloque de *crear texto con...*. Fíjate como al pulsar sobre el símbolo del mecanismo podemos ampliar las líneas añadiendo *varNum* a la parte derecha.

image-1654082067919.EA8YM1.png

El programa resultante quedará de la siguiente forma:

image-1654082093466.png

Por último, carga el programa, abre la *Consola* y comprueba las lecturas moviendo el potenciómetro.

image-1654082113505.X7KUM1.png

> Actividad de ampliación: cambia ahora el rango de salida y el texto que envía por el puerto serie.

7.5.3 Control del led RGB con el potenciómetro

En la siguiente actividad vamos a controlar los colores del led RGB utilizando el potenciómetro. Vamos a hacer que cambie de color según varíe el valor del potenciómetro. Es decir, cuando el valor del potenciómetro se encuentre entre 0 y 100 que el color del led sea rojo, cuando se encuentre entre 101 y 200 que sea verde y cuando esté entre 201 y 255 que sea azul.

Del menú *Lógica* vamos a necesitar dos nuevos bloques; el bloque de *Evaluar condición* y el bloque de *Conjunción/Disyunción*. Con ellos crearemos estas condiciones:

image-1654082159180.F79BN1.pngimage-1654082172579.TJI7M1.png

Deberemos crear tres estructuras para hacer los tres rangos.

image-1654082185559.DZB7M1.png

El programa quedaría como muestra la imagen:

image-1654082195859.YP6TM1.png

También se puede hacer el mismo programa de la siguiente forma:

image-1654082208046.QPJTM1.png

> Actividad de ampliación: completa el programa con más condiciones.

Reto A05. El zumbador

Estos contenidos han sido elaboradas por Fernando Hernández García, Ingeniero Técnico Industrial Especialidad Electrónica, formador del profesorado y profesor del Institut Torre del Palau (Terrassa – Barcelona). *Enlace de los contenidos*. ***Licencia CC-BY-NC-ND***.

Permiso

El zumbador (Buzzer en inglés) es un transductor electroacústico que produce un sonido o zumbido continuo o intermitente. En función de si se trata de un zumbador *Activo* o *Pasivo*, este zumbido será del mismo tono o lo podremos variar. Sirve como mecanismo de señalización o aviso y se utiliza en múltiples sistemas, como en automóviles o en electrodomésticos.

image-1654080653175.C8ZSM1.png

La placa ***Imagina TDR STEAM*** tiene un zumbador pasivo que está conectado en el pin D8.

image-1654080843720.png

ArduinoBlocks tiene 4 bloques específicos para programar y trabajar con el zumbador.

image-1654080827607.png

El sonido que emite el zumbador depende de la frecuencia de emisión del sonido. La frecuencia es el número de repeticiones por unidad de tiempo de cualquier evento periódico. Sabemos que el sonido se transmite en forma de onda y la frecuencia de un sonido es el número de oscilaciones o variaciones de la presión por segundo, nos indica cuantos ciclos por segundo tiene una onda.

image-1654080709723.CCS7M1.png

En la siguiente tabla están las frecuencias del sonido de las notas musicales:

image-1654080725180.4GB6M1.png

7.3.1 Primeros sonidos con el zumbador.

En el bloque Zumbador podemos variar dos parámetros: Ms (1) es el tiempo que dura cada sonido en milisegundos y Hz (2) es la frecuencia a la que vibra la membrana del zumbador para emitir el sonido.

image-1654080962021.png

Prueba con este sencillo programa cómo suena el zumbador.

image-1654080970112.6R5DN1.png

> Actividad de ampliación: cambia ligeramente el programa introduciendo tiempos diferentes en la duración de las notas y en las esperas para observar las diferencias que aparecen en la ejecución del programa.

7.3.2 Escalas musicales con el zumbador.

Vamos a hacer una escala de DO_4 a DO_5 utilizando un bloque que nos permite introducir directamente la nota sin que tengamos que saber los valores de la tabla de notas y frecuencias.

image-1654081045964.2X25M1.png

Haremos una pequeña escala musical.

image-1654081069503.LZN4M1.png

> Actividad de ampliación: intenta tocar esta melodía utilizando el zumbador. Las notas negras deben tener una duración de 500ms, la negra con puntillo 750ms y la blanca 1000ms.
> image-1654081094076.5GHYM1.png

7.3.3 Melodías con RTTTL

Las melodías en formato RTTTL (Ring Tone Text Transfer Language) es un lenguaje muy simple, creado por Nokia, con el objetivo inicial de definir de forma sencilla partituras musicales en formato texto para móviles.

Estas melodías RTTTL se pueden introducir de forma sencilla desde ArduinoBlocks y sólo se necesitan dos bloques.

image-1654081148133.9EHZM1.png

Copyright 2026

Realiza este programa y elige una de las melodías que hay disponibles en el menú desplegable del bloque RTTTL.

image-1654081161967.GNW0M1.png

> Actividad de ampliación: prueba con diferentes melodías RTTL.

Reto A06. La fotocélula LDR

Estos contenidos han sido elaboradas por Fernando Hernández García, Ingeniero Técnico Industrial Especialidad Electrónica, formador del profesorado y profesor del Institut Torre del Palau (Terrassa – Barcelona). *Enlace de los contenidos*. ***Licencia CC-BY-NC-ND***.

Permiso

Ahora que ya sabemos usar el Puerto Serie para leer los valores de los sensores, vamos a utilizarlo para ver el valor de una fotocélula (LDR). Una LDR (Light Dependent Resistor) es un resistor que varía su valor de resistencia eléctrica dependiendo de la cantidad de luz que incide sobre él. El valor de la resistencia disminuye con el aumento de intensidad de luz incidente.

image-1654082431044.02ECN1.png

En la placa Imagina TDR STEAM la fotorresistencia está denominada como "*Light*" y viene conectada en el Pin analógico A1.

image-1654082444954.png

En el menú TDR STEAM de ArduinoBlocks hay un bloque específico para el uso de este sensor.

image-1654082462399.png

En este bloque también se puede seleccionar el tipo de lectura del valor del sensor en % o en unidades de 0 a 1023.

image-1654082470602.EKGDN1.png

7.6.1 Encender y apagar un led según el nivel de luz

En esta actividad vamos a simular en el encendido automático de una farola cuando se hace de noche. Utilizando la LDR y el led azul vamos a hacer que cuando la LDR esté a oscuras se ilumine el led azul.

El programa es muy sencillo. Hay que generar una variable que la llamaremos “nivel_luz” y la estableceremos al sensor LDR. Recuerda seleccionar valor 0...1023.

Por último, un condicional en el cual cuando el valor sea menor de 500 que se encienda el led azul y, sino que permanezca apagado.

image-1654082514928.D16WM1.png

> Actividad de ampliación: haz un programa que muestre los valores de la LDR por el puerto serie.

Reto A07 Sensor de temperatura LM35D

Estos contenidos han sido elaboradas por Fernando Hernández García, Ingeniero Técnico Industrial Especialidad Electrónica, formador del profesorado y profesor del Institut Torre del Palau (Terrassa – Barcelona). *Enlace de los contenidos*. ***Licencia CC-BY-NC-ND***.

Permiso

En el siguiente reto vamos a medir la temperatura de una habitación utilizando el sensor de temperatura LM35D. El LM35D tiene un rango de temperatura de 0º a 100º ºC y una sensibilidad de 10mV/ºC.

La placa Imagina TDR STEAM dispone de este sensor LM35D y está conectado en el Pin analógico A2.

image-1654082592908.png

En el menú TDR STEAM de ArduinoBlocks hay un bloque específico para el uso de este sensor.

image-1654082612944.png

7.7.1 Lectura del valor de la temperatura

Para ello vamos a repasar el concepto de variables. Lo vimos inicialmente con la práctica del LED en la que incrementábamos y disminuíamos su brillo utilizando una variable *i*. En esta ocasión vamos a profundizar un poco más.

Las variables son elementos muy comunes en programación. Básicamente, crear una variable es darle un nombre a un dato o a una lectura. Por ejemplo, las mediciones de valores de temperatura las podemos guardar en una variable que se llame “Temperatura” o las del sensor de ultrasonidos en una llamada “Distancia”, etc. No es obligatorio su uso, pero nos permiten trabajar más cómodamente, además, como podemos personalizar su nombre, ayudan a clarificar el código y utilizar un lenguaje más natural.

Al trabajar con variables vamos a tener dos tipos de bloques principales:

- El bloque en el que le damos valor a la variable:

image-1654082740800.2718M1.png

- Y el bloque de la propia variable creada, para poder insertarla y combinarla con otros bloques:

image-1654082746959.FZUWM1.png

También podemos personalizar el nombre de la variable, de la siguiente forma:

image-1654082753591.WGWAN1.png

Una vez creada la nueva variable, podemos seleccionarla pulsando sobre el desplegable:

image-1654082766545.RSOWM1.png

Hay que tener en cuenta que las variables solo pueden estar formadas por una palabra. Si quieres incluir varias palabras, puedes usar el truco de separarlas con una barra baja "_", como en el ejemplo, "*Valor_Temperatura*".

image-1654082780941.U9WXM1.png

Ahora vamos a realizar el programa que mida la temperatura y la muestre por la consola.

image-1654082788877.V0CYM1.png

Envía el programa a la placa, conecta la *Consola* y comprueba la información que aparece en el terminal.

image-1654082794609.I491M1.png

> Actividad de ampliación: haz un programa que muestre datos más rápido y que muestre otro texto.

7.7.2. Alarma por exceso de temperatura

Ahora vamos a hacer una alarma sonora y acústica. El programa consistirá en hacer que el led rojo y el zumbador se accionen a la vez cuando el sensor de temperatura detecte una temperatura superior a 28ºC.

Para ello vamos a utilizar las *Funciones*. Las funciones permiten agrupar bloques de código. Esto es útil cuando un bloque de código se repite en varias partes del programa y así evitamos escribirlo varias veces o cuando queremos dividir el código de nuestro programa en bloques funcionales para realizar un programa más entendible.

Las funciones nos permiten realizar tareas que se repiten a lo largo del programa. En el menú "Funciones" tenemos el bloque "*para () hacer algo*". Lo utilizaremos para crear nuestra función como la siguiente imagen.

image-1654082817485.3FQ2M1.png

Debes escribir el nombre de ALARMA dentro de la función. Al hacer esto automáticamente en el menú "Función" aparecerá un nuevo bloque llamado ALARMA.

image-1654082831589.D8YCN1.png

image-1654082834802.OQ77M1.png

El programa final sería el siguiente:

image-1654082851564.S92EN1.png

> Actividad de ampliación: haz un programa que varíe el programa anterior para que encienda de color verde el led RGB si la temperatura inferior a los 28ºC y rojo si la temperatura es superior a los 35ºC. En el rango intermedio se indicará de color azul.

Reto A08 Sensor de temperatura y humedad DHT11

Estos contenidos han sido elaboradas por Fernando Hernández García, Ingeniero Técnico Industrial Especialidad Electrónica, formador del profesorado y profesor del Institut Torre del Palau (Terrassa – Barcelona). *Enlace de los contenidos*. ***Licencia CC-BY-NC-ND***.

Permiso

En esta actividad vamos a leer los valores de temperatura y humedad utilizando el sensor DHT11. Este sensor mide temperaturas en un rango de acción de 0ºC a +50ºC con un error de +/- 2ºC y la humedad relativa entre 20% y 90% con un error de +/-5%. No es un sensor con una gran sensibilidad, pero cumple nuestros objetivos sobradamente.

El sensor de temperatura es un termistor tipo NTC. Un termistor es un tipo de resistencia (componente electrónico) cuyo valor varía en función de la temperatura de una forma más acusada que una resistencia común.

image-1654159612307.JPVWM1.png

Su funcionamiento se basa en la variación de la resistividad que presenta un semiconductor con la temperatura.

El término proviene del inglés “*thermistor*", el cual es un acrónimo de las palabras *Thermally Sensitive Resistor* (resistencia sensible a la temperatura). Existen dos tipos fundamentales de termistores:

- Los que tienen un coeficiente de temperatura negativo (en inglés *Negative Temperature Coefficient* o NTC), los cuales decrementan su resistencia a medida que aumenta la temperatura.
- Los que tienen un coeficiente de temperatura positivo (en inglés *Positive Temperature Coefficient* o PTC), los cuales incrementan su resistencia a medida que aumenta la temperatura.

La placa Imagina TDR STEAM dispone de un sensor DHT11 conectado al pin D4. En un principio podríamos pensar que se trataría de un sensor analógico o que tuviera dos entradas, una para la

temperatura y otra para la humedad. Pero las propias características de diseño del sensor, hace que se puedan realizar todas las lecturas por un solo puerto digital.

image-1654159646042.png

En ArduinoBlocks en el menú de sensores tenemos el bloque específico para programar este sensor:

image-1654159668693.png

7.8.1 Zona de confort con DHT11

Puede definirse *confort térmico*, o más propiamente *comodidad higrotérmica*, como la ausencia de malestar térmico. En fisiología, se dice que hay *confort higrotérmico* cuando no tienen que intervenir los mecanismos termorreguladores del cuerpo para una actividad sedentaria y con una indumentaria ligera. Esta situación puede registrarse mediante índices que no deben ser sobrepasados para que no se pongan en funcionamiento los sistemas termorreguladores (metabolismo, sudoración y otros).

Según la imagen adjunta vamos a marcar unos puntos de temperatura y humedad en los que estaremos dentro de la zona de confort térmico, dentro de una zona de medio confort y fuera de la zona de confort.

image-1654159723710.BKRDN1.png

Usando el Led RGB vamos a indicar esas zonas:

- Led RGB en ROJO; fuera de la zona de confort.
- Led RGB en NARANJA; dentro de la zona media.
- Led RGB en VERDE; en la zona de confort.

A grandes rasgos estos serían los valores de las tres zonas:

- *Zona ROJA:*
  - Humedad por debajo del 20% y superior al 85%.**
  - Temperatura por debajo de 16ºC o superior a 26,5ºC.**
- *Zona NARANJA:*
  - Humedad entre el 20% y el 40% y entre el 65% y el 85%.**
  - Temperatura entre los 16ºC y los 18ºC o entre los 24 y los 26,5ºC.**
- *Zona VERDE:*
  - Humedad entre 40% y el 65%.
  - Temperatura entre 18ºC y 24ºC.

Copyright 2026

Para no complicar en exceso el programa de ejemplo, vamos a quedarnos sólo con la zona verde, es decir, el LED brillará en VERDE dentro de los parámetros de la Zona VERDE, para el resto el Led estará parpadeando en color ROJO.

Para realizar este programa necesitaremos varios de los bloques del menú de Lógica. Necesitaremos evaluar una *condición Lógica* y utilizar *conjunciones y disyunciones.*

- *Y:* se cumple si los dos operandos son verdaderos.
- *O:* se cumple si alguno de los dos operandos es verdadero.

image-1654159767980.Q898M1.png

Antes de evaluar las condiciones debemos establecer las dos variables; la variable *Temperatura* y la variable *Humedad*. Recuerda que en el menú desplegable del sensor DHT-11 debes elegir la Temperatura o la Humedad.

image-1654159786704.8Z43M1.png

Usando 3 bloques de conjunciones debes crear el siguiente bloque:

_image-1654159830584.png

Después ir metiendo las condiciones en cada uno de ellos:

image-1654159843828.5KJYM1.png

Y ve uniendo todo hasta conseguir esta condición final:

image-1654159853312.A05EN1.png

image-1654159867161.P75BN1.png

Por último, debes crear una función, que la puedes llamar *ALARMA*. Para apagar el led RGB es ponerlo de color negro.

Y con todo esto, el programa final quedaría así:

image-1654159874173.3WO9M1.png

> Actividad de ampliación: realiza un programa con las tres zonas por colores. Ten cuidado con las zonas donde se unen los rangos, ya que si el valor es justo (por ejemplo 40) deberás poner el símbolo >= o <=. También puedes mostrar los valores de humedad y temperatura por el puerto serie.

Copyright 2026

Copyright 2026

Reto A09. Receptor de infrarrojos IR

Estos contenidos han sido elaboradas por Fernando Hernández García, Ingeniero Técnico Industrial Especialidad Electrónica, formador del profesorado y profesor del Institut Torre del Palau (Terrassa – Barcelona). *Enlace de los contenidos*. ***Licencia CC-BY-NC-ND***.

Permiso

Una gran parte de los electrodomésticos utilizan mandos a distancia de infrarrojos, como los televisores o equipos musicales. El sensor infrarrojo es un dispositivo optoelectrónico capaz de medir la radiación electromagnética infrarroja de los cuerpos en su campo de visión. Todos los cuerpos emiten una cierta cantidad de radiación, esta resulta invisible para nuestros ojos, pero no para estos aparatos electrónicos, ya que se encuentran en el rango del espectro justo por debajo de la luz visible.

image-1654159907376.3YPEN1.png

En el caso del receptor de infrarrojos (IR) de la placa IMAGINA TDR STEAM permite codificar los protocolos de señales de pulsos infrarrojos utilizados por los mandos a distancia. Los protocolos detectados son los siguientes: RC5, RC6, NEC, SONY, image-1654159927216.C8JXM1.png PANASONIC, JVC, SAMSUNG, WHYNTER, AIWA, LG, SANYO, MITSUBISHI y DENON. Es decir, detectaría cualquier señal emitida por uno de esos mandos.

El mando a distancia contiene un circuito interno, un procesador y un led (*Light Emitting Diode*) que emite la señal infrarroja.

La señal infrarroja transmite el código correspondiente al botón del mando a distancia pulsado y lo transmite al dispositivo en forma de una serie de impulsos de luz infrarroja. Pensemos en el código Morse, que ya vimos en una de las prácticas anteriores, y sus tonos cortos y largos. De forma análoga, los pulsos de luz infrarroja transmitidos son de dos tipos, los llamados 0 y 1. Los 0 podrían verse como los tonos cortos y los 1 como los tonos largos. El receptor IR recibe la serie de impulsos de infrarrojos y los pasa a un procesador que descodifica la serie de 0 y 1 en los bits digitales para después realizar la función que programemos.

En la placa IMAGINA TDR STEAM el sensor receptor IR se encuentra conectado al pin digital D11. En el menú de “Sensores” de ArduinoBlocks tenemos dos bloques para programar nuestro receptor IR, uno propio del sensor y otro para el mando.

image-1654159971298.png

image-1654159990020.png

7.9.1 Recepción de comandos por infrarrojos

Vamos a realizar una pequeña actividad en la que utilicemos el receptor IR y el mando emisor de Keyestudio. Primero crearemos una variable de texto a la que llamaremos “codigo” (*sin acento, en programación se debe evitar poner acentos y símbolos*) y la estableceremos con el bloque del Receptor IR.

image-1654160021344.V1C2M1.png

Con este bloque podemos saber el código que envía nuestro mando a la placa. Realizaremos un pequeño programa para ver que código envía cada botón del mando.

image-1654160028168.VAICN1.png

A continuación, vamos a hacer otro programa. Consiste en encender el led rojo si pulsamos el botón arriba y que se apague cuando pulsamos el botón abajo. Vigila la versión del mando porque tendrás que cambiar la versión en función del mando.

image-1654160040207.3I83M1.png

Ahora ya podemos completar el programa que queda de la siguiente forma.

image-1654160048371.VNL0M1.png

> Actividad de ampliación: realiza un programa con el que puedas controlar el color del led RGB con diferentes botones del mando y que se muestre el código enviado por el puerto serie.

Copyright 2026

Reto A10 Puertos I2C: La pantalla LCD.

Estos contenidos han sido elaboradas por Fernando Hernández García, Ingeniero Técnico Industrial Especialidad Electrónica, formador del profesorado y profesor del Institut Torre del Palau (Terrassa – Barcelona). *Enlace de los contenidos*. ***Licencia CC-BY-NC-ND***.

Permiso

El estándar I2C (Inter-Integrated Circuit) fue desarrollado por Philips en 1982 para la comunicación interna de dispositivos electrónicos en sus artículos. Posteriormente fue adoptado progresivamente por otros fabricantes hasta convertirse en un estándar del mercado.

I2C también se denomina TWI (Two Wired Interface) únicamente por motivos de licencia. No obstante, la patente caducó en 2006, por lo que actualmente no hay restricción sobre el uso del término I2C.

El bus I2C requiere únicamente dos cables para su funcionamiento, uno para la señal de reloj (SCL) y otro para el envío de datos (SDA), lo cual es una ventaja frente al bus SPI. Por contra, su funcionamiento es un poco más complejo, así como la electrónica necesaria para implementarla.

image-1654160078963.9NZZM1.png

En el bus, cada dispositivo dispone de una dirección, que se emplea para acceder a los dispositivos de forma individual. Esta dirección puede ser fijada por hardware (en cuyo caso, frecuentemente, se pueden modificar los últimos 3 bits mediante “jumpers” o interruptores, o por software.

En general, cada dispositivo conectado al bus debe tener una dirección única. Si tenemos varios dispositivos similares tendremos que cambiar la dirección o, en caso de no ser posible, implementar un bus secundario.

El bus I2C tiene una arquitectura de tipo maestro-esclavo. El dispositivo maestro inicia la comunicación con los esclavos, y puede mandar o recibir datos de los esclavos. Los esclavos no pueden iniciar la comunicación (el maestro tiene que preguntarles), ni hablar entre si directamente.

El bus I2C es síncrono. El maestro proporciona una señal de reloj, que mantiene sincronizados a todos los dispositivos del bus. De esta forma, se elimina la necesidad de que cada dispositivo tenga su propio reloj, de tener que acordar una velocidad de transmisión y mecanismos para mantener la transmisión sincronizada (como en UART)

image-1654160095805.50HEN1.png

A este bus de comunicaciones se le pueden conectar múltiples dispositivos:

- Pantalla LCD.
- Pantalla OLED.
- Matriz de leds.
- Acelerómetros.
- Giroscopios.
- Sensores de temperatura.
- Controladores de servomotores.
- Sensor de color.

Vamos a proponer una serie de retos con algunos elementos I2C que no vienen integrados en la placa Imagina TDR STEAM pero que se pueden incorporar.

7.10.1 Pantalla LCD 16x2

Vamos a realizar la conexión de una pantalla LCD (16x2). La pantalla LCD utilizada es una pantalla de 16 caracteres (por fila) y dos columnas. Esta pantalla tiene 4 conexiones, dos cables (SDA y SCL para el bus de comunicaciones I2C) y los dos cables de alimentación (VCC y GND).

Conectaremos la pantalla LCD a la placa Imagina TDR STEAM en el conector indicado:

image-1654160165282.png

Hemos de tener cuidado y respetar las conexiones, tal y como se indica en la siguiente tabla:

TdR STEAM	***Color Cable***	***LCD***
GND	Negro	GND
VCC	Rojo	VCC
SDA	Amarillo	SDA
SCL	Blanco	SCL

Conexión de los cables a la pantalla y a la placa Imagina TDR STEAM.

image-1654160191322.SSAVM1.png

En la pantalla aparecerán correctamente los datos cuando la coloquemos es esta posición, sino se verán al revés:

image-1654160206573.96K7M1.png

En la configuración de la pantalla hemos de seleccionar la dirección de configuración 0x27.

Vamos a realizar un programa que muestre una información por la pantalla LCD. El programa mostrará un texto por la pantalla LCD y un contador de 0 a 255 que se reiniciará en cada ciclo.

image-1654160251845.W00YM1.png

710.2 NO INCLUIDO EN EL KIT DE CATEDU La pantalla OLED

La pantalla OLED la conectaremos también en el puerto de comunicaciones I2C respetando las conexiones.

image-1654160265692.T2K7M1.png

La pantalla que vamos a utilizar es una pantalla OLED de 0,96 pulgadas de 128x64 pixeles. OLED es la abreviatura de diodo emisor de luz orgánico. En el nivel microscópico, una pantalla OLED es una matriz de leds que se iluminan cuando emiten energía. La tecnología antigua de las LCD (pantalla de cristal líquido) utiliza polarizadores controlados electrónicamente para cambiar la forma en que la luz pasa o no pasa a través de ellos. Esto requiere una luz de fondo externa que ilumine toda la pantalla por debajo. Esto supone un consume alto de energía porque en el momento en que la pantalla está encendida, se debe proporcionar suficiente luz para todos los píxeles. La nueva tecnología OLED solo usa electricidad por cada píxel que queramos encender. Esto hace que la tecnología OLED sea muy eficiente. Además, la forma en que se construyen estos tipos de OLED permite que sean muy delgadas en comparación con la pantalla LCD.

Vamos a realizar un programa para poder ver los datos del sensor de humedad y temperatura por la pantalla OLED. Existen una serie de bloques específicos para controlar la pantalla.

image-1654160371696.8A5XM1.png

Primero configuraremos la dirección I2C de la pantalla OLED. En el caso de la pantalla utilizada, tendremos que seleccionar 0x3C.

image-1654160399947.BDL2M1.png

Colocamos el bloque dentro de Inicializar y creamos las dos variables (Temperatura y Humedad).

image-1654160410952.C772M1.png

A continuación, podemos hacer diferentes cosas:

- Borrar toda la pantalla.
- Mostrar información, utilizando diferentes tamaños de texto.
- Hacer figuras geométricas.
- Encender diferentes leds.
- Cargar una imagen Bitmap.

El programa resultante para ver los datos en la pantalla es el siguiente:

image-1654160443354.O938M1.png

Copyright 2026

Reto A11. Serial plotter

Estos contenidos han sido elaboradas por Fernando Hernández García, Ingeniero Técnico Industrial Especialidad Electrónica, formador del profesorado y profesor del Institut Torre del Palau (Terrassa – Barcelona). *Enlace de los contenidos*. ***Licencia CC-BY-NC-ND***.

Permiso

Vamos a realizar un programa muy sencillo e interesante que nos permitirá ver los datos del potenciómetro en forma de gráfica y los podremos exportar en formato CSV para poder tratarlos posteriormente. Con este programa conseguiremos realizar un sistema de adquisición de datos.

Este es el programa en ArduinoBlocks que hemos confeccionado.

image-1654160471133.AX86M1.png

Enviamos el programa a la placa y activamos el Serial Plotter.

image-1654160481531.FT7DN1.png

Pondremos la velocidad de comunicación (baudrate) a 9600 y después pulsaremos ***Conectar*** para empezar a ver los datos.

image-1654160495857.UOGTM1.png

Para poder guardar los datos en CSV hemos de apretar el botón de grabación, adquirir los datos que queremos y apretar el botón de parar grabación.

image-1654160505870.5PK8M1.png

Podremos ver la cantidad de muestras recogidas.

image-1654160517921.M9EUM1.png

Ahora pulsamos en el botón CSV para guardar los datos en nuestro ordenador y poder trabajar con el fichero de datos creado.

image-1654160536253.P0YWM1.png

Copyright 2026

Actividad de ampliación: modifica el programa para que muestre los datos ahora de otro sensor.

Copyright 2026

Reto A12. El micrófono

> *El siguiente reto está extraído del* ***LIBRO ACTIVIDADES CON IMAGINA TR STEAM Y ARDUINOBLOCKS VERSION 4.0***
>
> *Descarga en : Repositorio original en Dropbox Repositorio alternativo Drive*
>
> *Concretamente de la página 89 capítulo 7.11.1*
>
> *Elaborado por el equipo de Innova Didáctic Robolot Team* ***LICENCIA CC- BY NC ND*** *Permiso*

Vamos a realizar un pequeño programa en el que veamos por el puerto serie el nivel de sonido. Se puede utilizar cualquiera de los dos bloques de ejemplo.

2023-02-03 11_56_38-Manual Actividades Imagina TdR STEAM versión 4 - PDF-XChange Viewer.jpg

También se pueden mostrar los datos por la pantalla LCD. En este programa vamos a hacer dos ciclos uno de 0 a 255 en el que iremos regulando la intensidad del led y viendo el valor. Después mostramos el nivel de sonido. Después haremos lo mismo pero el ciclo a la inversa, de 255 a 0 y regulando el nivel de iluminación del led a la inversa. Después volvemos a mostrar el nivel del micrófono.

micro2.jpg

> Actividad de ampliación: modifica el programa para que muestre los datos tanto por la pantalla LCD como por el puerto serie.

Descarga otros retos

En la página de ArduinoBlocks http://www.arduinoblocks.com/ en **Recursos - Libros&Documentación** tenemos la descarga de las actividades a realizar con los alumnos del kit **Imagina TdR STEAM**

retosTdRSteam.png

Esta documentación está preparada para trabajar con los alumnos. Recomendamos utilizarla menos la parte de comunicaciones IoT Wifi y Bluetooth que lo explicaremos aparte.

Son documentos muy vivos por lo tanto es conveniente fijarse en la última fecha y en la última versión. Tenemos varios documentos, parecidos y en la siguiente página comentamos las diferencias.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Comentarios de las descargas de otros retos

LIBRO ACTIVIDADES CON IMAGINA TR STEAM Y ARDUINOBLOCKS VERSION 4.0

Descarga en :

- Repositorio original en Dropbox
- Repositorio alternativo Drive

retosTdRSteam3.png

Copyright 2026

Nuestra opinión LIBRO ACTIVIDADES CON IMAGINA TR STEAM Y ARDUINOBLOCKS VERSION 4.0

Consideramos que es el documento más completo, trabaja tanto la pantalla I2C LCD, el módulo micrófono y luego actividades de ampliación.

La parte de sistemas de telecomunicación **Bluetooth y Wifi** lo trataremos aparte en este curso pues la parte de Bluetooth trabaja la pantalla OLED y módulo acelerómetro que no se suministra, y la parte de Wifi no trabaja Adafruits ni Blynk.

Copyright 2026

Aconsejamos pues
Trabajar con los alumnos las actividades**desde la A01 hasta la A14** de este LIBRO ACTIVIDADES CON IMAGINA TR STEAM Y ARDUINOBLOCKS VERSION 4.0.

LIBRO RETOS VERSION 1

Descarga en de retos https://drive.google.com/file/d/1uQdLDhT1UigSFbHkFH5BH_9cO4BrDj-S/view

retosTdRSteam1.png

- **1 Reto A01. El led.**
  - 1.1 Reto A01.1. ON/OFF led rojo.
  - 1.2 Reto A01.2. ON/OFF led rojo y azul.
  - 1.3 Reto A01.3. ON/OFF led rojo y azul con repeticiones.
- **2 Reto A02. El led RGB.**
  - 2.1 Reto A02.1. Identificación de colores RGB.
  - 2.2 Reto A02.2. Múltiples colores con el led RGB. .
- **3 Reto A03. El zumbador. .**
  - 3.1 Reto A03.1. Primeros sonidos con el zumbador.
  - 3.2 Reto A03.2. Escalas musicales con el zumbador.
  - 3.3 Reto A03.3. Melodías con RTTTL. .
- **4 Reto A04. El pulsador. .**
  - 4.1 Reto A04.1. Control ON/OFF de un led con un pulsador I. .
  - 4.2 Reto A04.2. Control ON/OFF de un led con un pulsador II.
  - 4.3 Reto A04.3. Control ON/OFF de un led con un pulsador III. .
- **5 Reto A05. El potenciómetro**
  - 5.1 Reto A05.1. Lectura de valores con el puerto serie. .
  - 5.2 Reto A05.2. Ajuste de valores de entrada y salida: mapear.
  - 5.3 Reto A05.3. Control del led RGB con el potenciómetro. .
- **6 Reto A06. La fotocélula (LDR – sensor de luz).**
  - 6.1 Reto A06.1. Encender y apagar un led según el nivel de luz.
- **7 Reto A0 Sensor de temperatura LM35D.**
  - 7.1 Reto A01. Lectura del valor de la temperatura.
  - 7.2 Reto A02. Alarma por exceso de temperatura.
- **8 Reto A08. Sensor de temperatura y humedad DHT11.**
  - 8.1 A08.1. Zona de confort con DHT11. .
- **9 Reto A09. Receptor de infrarrojos (IR).**
  - 9.1 Reto A09.1. Recepción de comandos por infrarrojos. .
- **10 Reto A10. Puertos de expansión I2C: pantalla LCD.**

Copyright 2026

Nuestra opinión LIBRO RETOS

Trabaja la pantalla I2C LCD pero no el módulo micrófono. Luego trabaja la pantalla OLED que no se suministra en el kit. La parte de sistemas de comunicación opinamos igual que en el anterior libro

LIBRO ACTIVIDADES Y RETOS

Descarga del libro de Actividades y retos

https://drive.google.com/file/d/1d0E5d3q5bRT2IFGZSeYi0npw4KE6XMtY/view

retosTdRSteam2.png

- **A01. – El LED**
  - A01.1.– On/OFF del LED Rojo (Pin 12)
  - A01.2.– On/OFF del LED Rojo y Azul (Pin 12 y 13)
  - A01.3.– On/OFF del LED Rojo y Azul (Pin 12 y 13).
- **A02. – El LED RGB PINes (D6-D9-D10)**
  - A02.1.– Identificación de colores y Pines LED RGB
  - A02.2.– Múltiples colores con el LED RBG.
  - A02.3.– Control PWM del LED RBG I.
  - A02.4.– Control PWM del LED RBG II.
  - A02.5.– Control PWM del LED RBG III.
- **A03. – Generar notas con el Buzzer o Zumbador.**
  - A03.1.– Primeros sonidos con el Buzzer.
    A03.2.– Escala musical I. .
  - A03.3.– Escala musical II. .
  - A03.4.– Melodias RTTTL
- **A04. – Sensor pulsador.**
  - A04.1.– ON/OFF LED con Pulsador.

Copyright 2026

  - A04.2.– ON/OFF LED con Pulsador I.
  - A04.3.– ON/OFF LED con Pulsador II. .
- **A05. – Potenciómetro**
  - A05.1.– Lectura valor Potenciómetro por el Puerto Serie.
  - A05.2.– Uso del bloque MAPEAR y el Potenciómetro.
  - A05.3.– Control del LED RGB con el Potenciómetro.
- **A06. – Lectura del valor de la fotocélula (LDR).**
  - A06.1.– On/Off de LED según el nivel de luz.
- **A07. – Medición de temperatura de una habitación.**
  - A07.1.– Lectura del valor de la temperatura.
  - A07.2.– Alarma por exceso de temperatura.
- **A08. – Temperatura y Humedad. Sensor DHT11**
  - A08.1.– Confort higrométrico con DHT11.
- **A09. – Receptor IR. .**
  - A09.1.– Test de receptor IR.
- **A10. – Puertos de expansión.**

Nuestra opinión LIBRO RETOS

No trabaja la pantalla I2C LCD ni el módulo micrófono ni la parte de comunicaciones.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Copyright 2026