

Actividades con Imagina TDR STEAM y ArduinoBlocks.



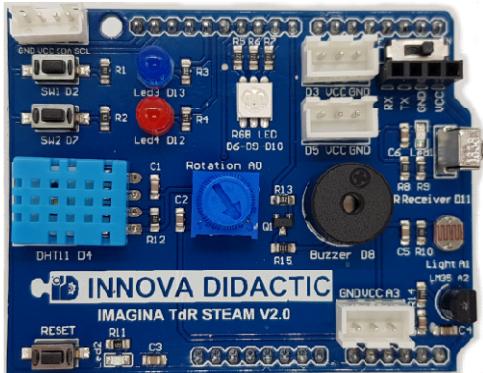
Índice

Introducción	3
¿Cómo funciona un sistema de control programado?	4
Componentes de Shield Imagina TDR STEAM.....	5
Placa de control y shield Imagina TDR STEAM	5
Especificaciones	7
Placa de control Arduino.....	7
ArduinoBlocks.....	8
ArduinoBlocks y la shield Imagina TDR STEAM	9
Preparativos: instalación de los drivers y programas	9
Programación Arduino con ArduinoBlocks	11
Actividades con la Shield Imagina TDR STEAM	15
A01. – El LED	15
A01.1.– On/OFF del LED Rojo (Pin 12)	16
A01.2.– On/OFF del LED Rojo y Azul (Pin 12 y 13).....	17
A01.3.– On/OFF del LED Rojo y Azul (Pin 12 y 13).....	18
A02. – El LED RGB PINes (D6-D9-D10)	19
A02.1.– Identificación de colores y Pines LED RGB.....	20
A02.2.– Múltiples colores con el LED RBG.....	21
A02.3.– Control PWM del LED RBG I	22
A02.4.– Control PWM del LED RBG II	23
A02.5.– Control PWM del LED RBG III.....	25
A03. – Generar notas con el Buzzer o Zumbador.....	26
A03.1.– Primeros sonidos con el Buzzer	27
A03.2.– Escala musical I	28
A03.3.– Escala musical II	30
A03.4.– Melodías RTTTL.....	31
A04. – Sensor pulsador	32
A04.1.– ON/OFF LED con Pulsador.....	33
A04.2.– ON/OFF LED con Pulsador I	34
A04.3.– ON/OFF LED con Pulsador II	36
A05. – Potenciómetro.....	38

A05.1.– Lectura valor Potenciómetro por el Puerto Serie.....	39
A05.2.– Uso del bloque MAPEAR y el Potenciómetro.	42
A05.3.– Control del LED RGB con el Potenciómetro.....	43
A06. – Lectura del valor de la fotocélula (LDR).	44
A06.1.– On/Off de LED según el nivel de luz.....	45
A07. – Medicción de temperatura de una habitación.....	46
A07.1.– Lectura del valor de la temperatura.	47
A07.2.– Alarma por exceso de temperatura.....	49
A08. – Temperatura y Humedad. Sensor DHT11	50
A08.1.– Confort higrométrico con DHT11.....	51
A09. – Receptor IR.....	54
A09.1.– Test de receptor IR.....	55
A10. – Puertos de expansión.....	56

Introducción

El presente manual pretende ser una herramienta base para iniciarse en el mundo de la programación, la electrónica y la robótica utilizando para la Shield de Arduino Imagina TDR STEAM.



En este documento no pretende ser únicamente un manual para aprender programación. El manual presenta una serie de actividades guiadas para aprender a programar de una manera entretenida y divertida mientras aprendemos conceptos relacionados con las **S.T.E.A.M.**

Para realizar la programación con la shield **IMAGINA TDR STEAM** utilizaremos un lenguaje de programación visual basado en bloques llamado [ARDUINOBLOCKS](#), Hay quien podría pensar que un lenguaje de este estilo es muy básico y limitado, pero ya veremos a lo largo del manual la gran potencialidad y versatilidad de este programa.

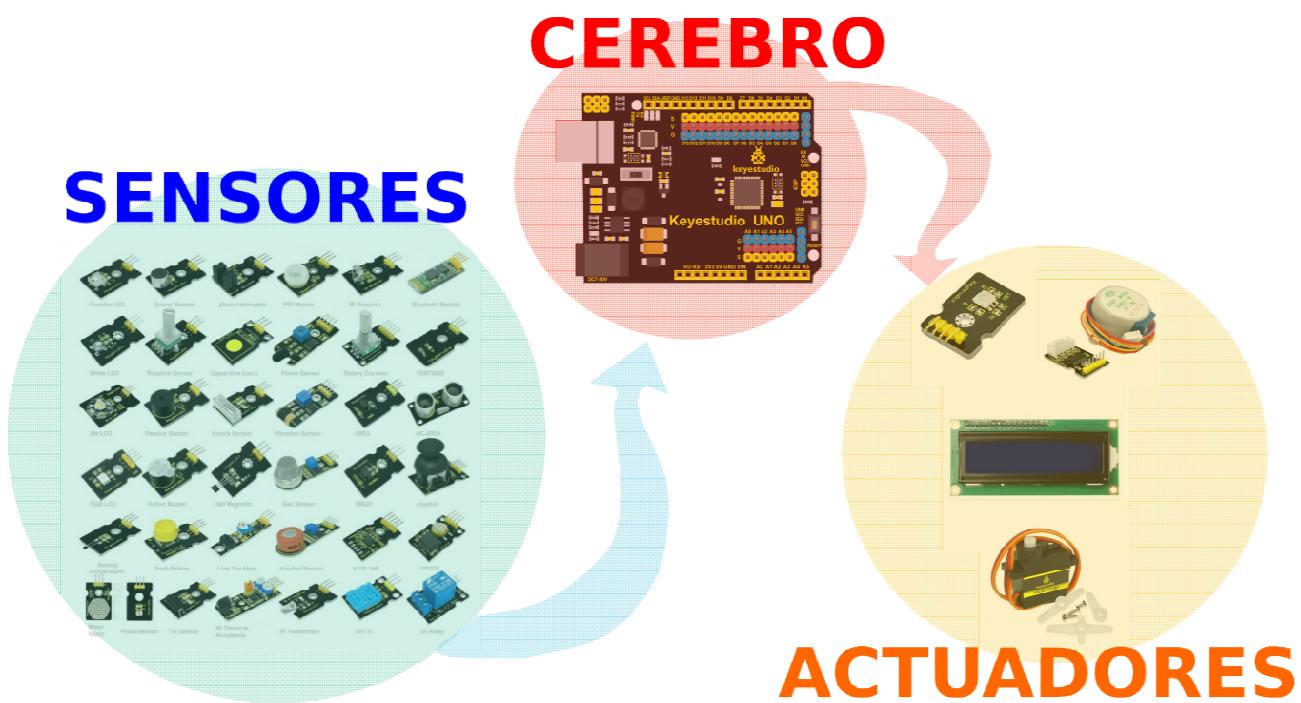
The screenshot shows the main interface of the ArduinoBlocks website. At the top, there's a navigation bar with the ArduinoBlocks logo, a search bar labeled "Buscar proyectos", and links for "Recursos", "Español", and "Iniciar sesión". Below the navigation, there are several cards with different content:

- Por donde empezar...**: Shows a video thumbnail for "ArduinoBlocks-Connector v4" and a download link for "DESCARGA ARDUINOBLOCKS-CONNECTOR".
- Libros & Documentación**: Features links to "DIDACTRONICA Blog" and projects like "Programando Sonoff Basic con ArduinoBlocks", "Proyecto EducaCont: Mide la contaminación del aire con Arduino", and "Nivel de volumen con Neopixel y juegos de luz con música (vúmetro)".
- Enlaces**: Displays a Twitter feed for @ArduinoBlocks with tweets about Sonoff devices and a link to a blog post.
- Plataformas soportadas...**: Lists supported boards: Arduino, Uno, Nano, Mega, Leonardo, Pro Micro, Espruino, etc.
- Colaborador & Distribuidor Oficial**: Shows the logos of INNOVA DIDACTIC and ROBOLOT.
- Promos**: Shows small promotional images for ArduinoBlocks.

El portal es www.arduinoblocks.com.

¿Cómo funciona un sistema de control programado?

Un sistema de control programado funciona de manera similar a la de un ser humano. Cuando nuestro cerebro recibe información de los sentidos; oído, olfato, gusto, vista y tacto; analiza esta información, la procesa y da órdenes a nuestros músculos para realizar movimientos o estímulos a las cuerdas vocales para emitir sonidos; los 5 sentidos equivalen a entradas de información y la voz los músculos serían las salidas sonoras y motrices.



En el caso de un **sistema de control programado**, un chip hace la función de cerebro. Este chip se llama microcontrolador y tiene unas entradas de información donde se conectan los sensores de luz (LDR), temperatura (NTC), sonido... y también tiene salidas, donde se conectan los motores, LEDs, buzzers...

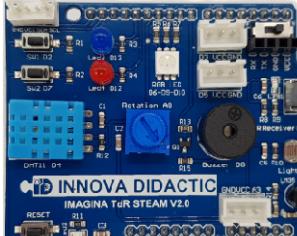
La diferencia principal es que, así como nuestro cerebro ha ido aprendido lo que tiene que hacer a lo largo de nuestra vida a base de estímulos positivos y negativos, el sistema programado tiene su memoria vacía, es decir, no sabe lo que debe hacer. Entonces nosotros tenemos que decirle como tiene que actuar en función de las señales que recibe de los sensores. ¡A esta acción se le llama **programar!**



Componentes de Shield Imagina TDR STEAM

La **Shield Imagina TDR STEAM** es una placa didáctica desarrollada por el equipo ROBOLOT que presenta la gran ventaja de tener una gran cantidad de sensores, actuadores y conexiones de expansión incorporados directamente en ella. Únicamente hay que conectar la shield a una placa Arduino UNO y ya está todo listo para empezar a programar.

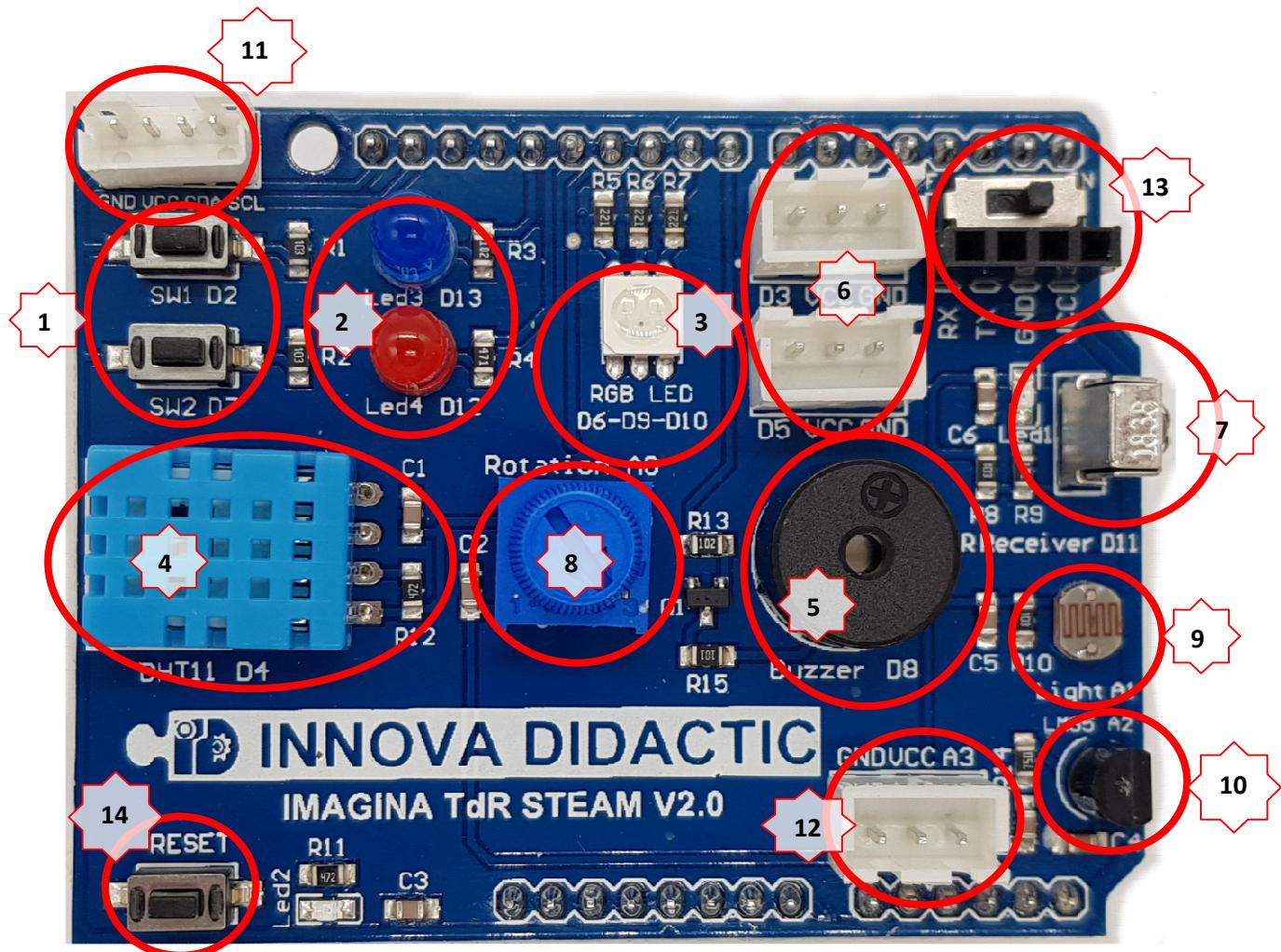
Placa de control y shield Imagina TDR STEAM

No.	Componente	Referencia	Imagen
PLACA DE CONTROL			
1	Placa Keyestudio Easy Plug	KS0001	
2	Cable USB	1	
SHIELD			
3	Imagina TDR STEAM		

La **Shield Imagina TDR STEAM** dispone de los siguientes módulos:

	Sensor/ Actuador/ Módulo	PIN(es) de contacto.
1	Dos pulsadores (SW1, SW2)	D2 y D7
2	Dos LEDs (Azul Led3 y Rojo Led4)	D13 y D12
3	LED RGB	D6-D9-D10
4	Módulo DHT11 Sensor de Temperatura y Humedad	D4
5	Buzzer o Piezoeléctrico	D8
6	Dos puertos (Entradas/Salidas) digitales	D3 y D5

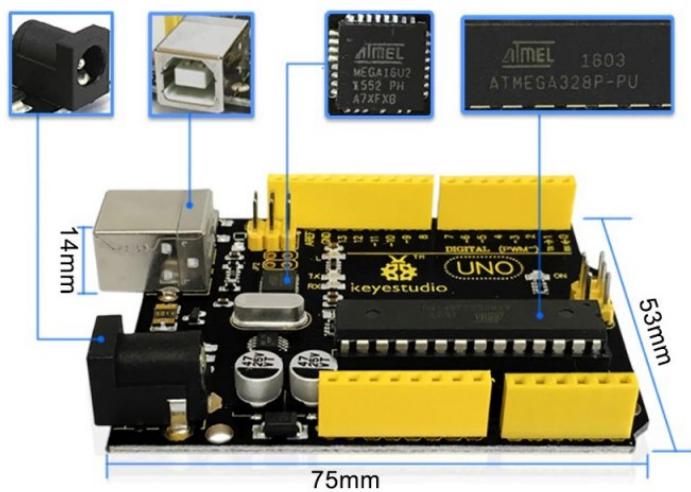
7	Módulo receptor de infrarrojos (IR)	D11
8	Módulo potenciómetro giratorio	A0
9	Sensor de luminosidad (LDR)	A1
10	Sensor de temperatura (LM35)	A2
11	Interface I2C compatible con sensores y módulos Keyestudio	SDA-A4, SCL-A5
12	Puerto Entrada Analógico	A3
13	Conexión de comunicaciones Bluetooth y Wifi (switch On/Off)	Rx,Tx
14	Botón de reinicio.	



Placa de control Arduino.

La placa de control necesaria para la Shield Imagina TDR STEAM está basada en una **Arduino UNO**. **Arduino** es una plataforma de prototipos electrónica de **código abierto** (Open-Source) basada en hardware y software flexibles y fáciles de usar. Está pensado para *artistas, diseñadores,...*, como hobby y en general para cualquiera interesado en crear objetos o entornos interactivos.

Arduino puede “sentir” el entorno mediante la recepción de entradas desde una variedad de sensores y puede “actuar” a su alrededor mediante el control de luces, motores y otros artefactos.



La placa posee 14 pines digitales de entrada / salida (de los cuales 6 se pueden usar como salidas PWM), 6 entradas analógicas, un cristal de cuarzo de 16 MHz, una conexión USB, un conector de alimentación, un encabezado ICSP y un botón de reinicio.

Al ser hardware libre existen multitud de fabricantes que han desarrollado versiones basadas en Arduino. Uno de esos fabricantes es **Keyestudio**.

La placa **Keyestudio UNO R3** lleva un microcontrolador que está basado en el ATmega328.



Especificaciones

- Voltaje de funcionamiento: + 5V
- Voltaje de entrada externo: + 7V ~ + 12V. (Límite: +6 V. <+ 20 V).
- Corriente de interfaz DCI / O: 20 mA
- FlashMemory: 32 KB (ATmega328) de los cuales 0,5 KB utilizados por el gestor de arranque
- Capacidad de almacenamiento EEPROM: 1KB
- Frecuencia del reloj: 16MHz

- Microcontrolador ATmega328P-PU
- Pines de E / S digital 14 (de los cuales 6 proporcionan salida PWM)
- Pines de E / S digitales PWM 6 (D3, D5, D6, D9, D10, D11)
- Pines de entrada analógica 6 (A0-A5)
- LED_BUILTIN D13

ArduinoBlocks

[ARDUINOBLOCKS](#) es un lenguaje de programación gráfico por “Bloques” creado por el profesor Juanjo López. Está pensado para que niños y niñas aprendan a programar con placas Arduino a partir de los 8 años.

Los distintos bloques sirven para leer y escribir las distintas entradas y salidas de la placa, así como programar funciones lógicas, de control, etc.

The screenshot shows the ArduinoBlocks software interface. On the left, there is a sidebar with a tree view of available blocks categorized by color:

- Lógica (Blue)
- Control (Green)
- Matemáticas (Yellow)
- Texto (Orange)
- Variables (Red)
- Listas (Purple)
- Funciones (Dark Blue)
- Entrada/Salida** (Grey, currently selected)
- Tiempo
- Puerto serie
- Bluetooth
- Sensores
- Actuadores
- Pantalla LCD
- Memoria
- Motor
- Motor-Shield
- Keypad
- Reloj RTC
- GPS
- Tarjeta SD
- MQTT
- NeoPixel
- RFID
- LedMatrix 8x8
- Domótica

The main workspace contains the following blocks:

- Leer digital Pin 2
- Escribir digital Pin 2 ON
- Leer analógica Pin A0
- Escribir analógica (PWM) Pin 3 Valor 0
- Leer digital Pin 2
- Escribir digital Pin 2 false
- Leer analógica Pin A 0
- Escribir analógica (PWM) Pin 3 Valor 0
0 - 255
- Leer pulso Pin 2 ON Tiempo de espera 1000
- Interrupción Pin 2 RISING
- Inicializar (green block)
- Bucle (yellow block)

ArduinoBlocks y la shield Imagina TDR STEAM

En este manual usaremos **ArduinoBlocks** dedicado al uso de la Shield Imagina TDR STEAM, con estos bloques podremos programar las entradas y salidas de nuestra placa para que realice las tareas que queramos.

Antes de comenzar necesitaremos instalar unos drivers y programas en nuestro ordenador.

Preparativos: instalación de los drivers y programas

Para programar la nuestra placa **arduino** con el programa online [ArduinoBlocks](#), necesitaremos instalar un pequeño programa disponible en la sección de “RECURSOS” en la página principal de [arduinoblocks.com](#). Se trata de [ArduinoBlocksConnector v.4](#)



También lo puedes descargar directamente de este enlace:

<http://www.arduinoblocks.com/web/site/abconnector>

Una vez clicado el link nos aparecerá esta ventana en la que podremos elegir nuestro sistema operativo; Windows, Ubuntu, MacOS o incluso RaspberryPi.



ArduinoBlocks-Connector v4

La aplicación que conecta ArduinoBlocks a tu placa Arduino!



Windows Ubuntu 32 bits Ubuntu 64 bits MacOS RaspberryPi

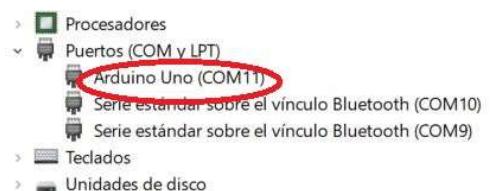
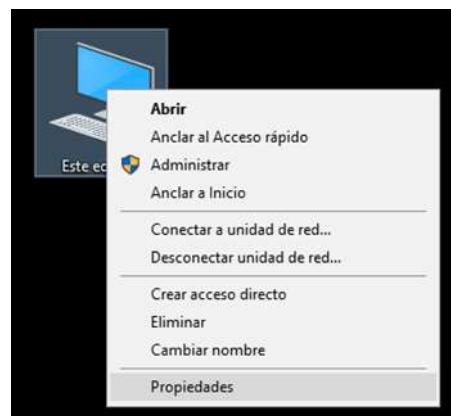
Windows
Probado en XP, 7, 10 (32/64 bits)

 Windows

Descarga para Windows (.exe)

¡Desactiva el antivirus si la descarga falla!

Una vez instalado *ArduinoBlocksConnector v.4*, conectaremos nuestra placa Keyestudio EasyPlug a nuestro ordenador utilizando el cable USB. Esperamos un breve tiempo y nos aseguraremos que se ha reconocido la placa. Para ello, con el botón derecho nos dirigiremos sobre el icono “Este equipo” y pulsaremos en “Propiedades”. Nos dirigiremos a “Administrador de dispositivos” y observaremos si en “Puertos (COM y LPT)” aparece nuestra placa Arduino UNO con un (COMXX) disponible. (Todo esto en Windows.)

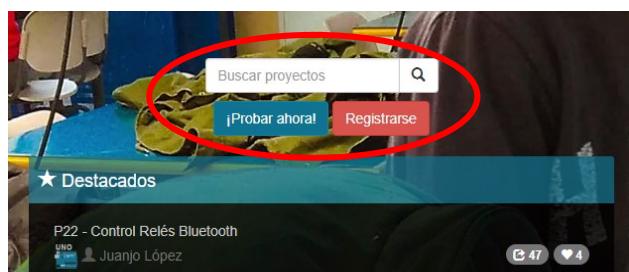


*En caso de que no nos reconociera la placa, como paso opcional podemos descargar e instalar el programa oficial [Arduino IDE](#) para que se instalen los drivers necesarios, ni tan si quiera es necesario ejecutarlo.

Programación Arduino con ArduinoBlocks

En el portal www.arduinoblocks.com, tan solo es necesario realizar un registro gratuito rellenando todos los campos y pulsando en “Nuevo usuario”.

Será necesario validar la cuenta en el correo recibido en nuestra dirección de correo (sin no aparece en “Bandeja de entrada”, revisar la carpeta “Spam” o “Correo Basura”) y validar clicando sobre el link recibido.



Nuevo usuario

*** Recommended **GMail** accounts (Review SPAM folder) *** (Hotmail,Msn,... may not work due to spam filters)

Correo electrónico	<input type="text"/>
Confirmación de correo electrónico	<input type="text"/>
Clave	<input type="text"/>
Confirmación de clave	<input type="text"/>
Nombre	<input type="text"/>
Apellidos	<input type="text"/>
País	SPAIN <input type="button" value="▼"/>
Ciudad	<input type="text"/>

Recibir información y novedades por email

Seguidamente ya podremos “Iniciar sesión”.

Iniciar sesión

Correo electrónico	<input type="text"/>
Password	<input type="text"/>

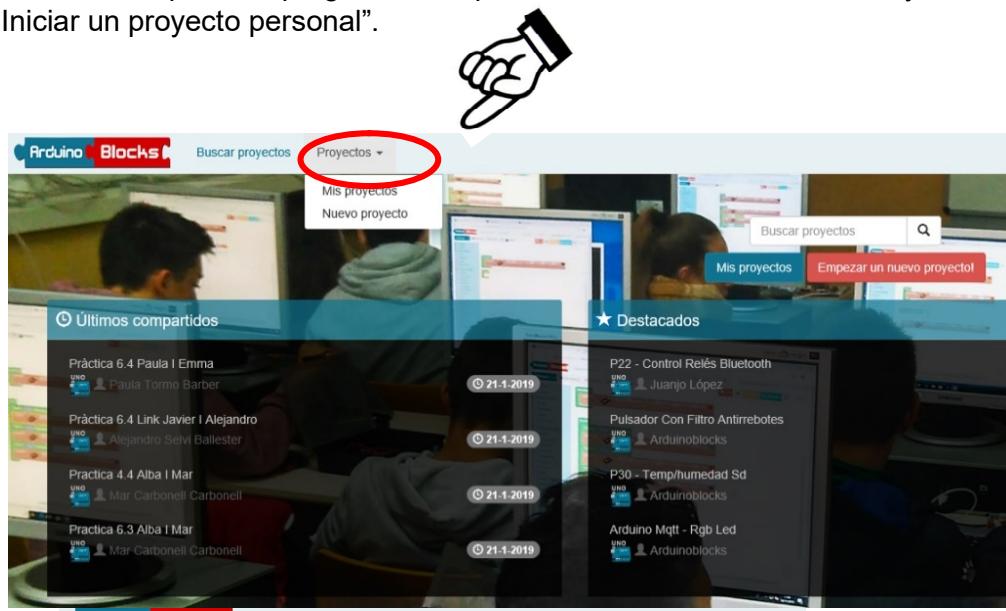
[Nuevo usuario](#)
[No recuerdo mi clave](#)

Una vez logueados nos aparece una ventana como esta, en la que tendremos guardados todos los proyectos que vayamos haciendo para poder acceder a ellos desde cualquier lugar.

Nombre	Fecha creación	Fecha modificación	Tipo de
uiio	2019-07-24 08:52:45	2019-07-24 09:16:46	Keyestudi
Teclado con leonardo	2019-07-24 07:44:33	2019-07-24 08:51:52	Arduino L

ArduinoBlocks permite crear proyectos personales, pero también permite crear proyectos como profesor para compartirlo con los alumn@s y poder supervisarlos y corregirlos. Otra aspecto muy destacable es que puedes hacer públicos tus proyectos por lo que hay una gran comunidad que comparte sus proyectos y cualquier usuario puede verlos y editarlos.

Pues vamos a empezar a programar...., para ello seleccionaremos “Proyectos”, “Nuevo proyecto” e “Iniciar un proyecto personal”.



Nuevo proyecto

Proyecto personal

Iniciar un proyecto personal

Empieza a trabajar en tu proyecto ahora mismo. Será totalmente privado para tí. Una vez finalizado, si lo deseas, lo puedes compartir con el resto del mundo!

Profeso

Crear un proyecto

¿Eres profesor? ¡Puedes crear tu proyecto y tú podrás...

En el menú de “Tipo de proyecto” ArduinoBlocks permite programar varios tipos de placas Arduino y diferentes Robots, en nuestro caso seleccionaremos la opción “**ARDUINO UNO**”.

Esta opción nos presentará los menús necesarios para poder programar nuestra placa de forma fácil y sencilla.

Una práctica muy recomendable es ir documentando los proyectos que se van haciendo, para ello la plataforma de ArduinoBlocks dispone de un menú de cada proyecto en el que se pueden anotar el **Nombre del proyecto**, una **Descripción**, los **Componentes del proyecto** y **Comentarios**.

Una vez rellenados todos los campos daremos al botón de “**NUEVO PROYECTO**”.

La siguiente imagen nos muestra la interface de programación con los bloques específicos para poder programar con mucha facilidad nuestra placa de control **Keyestudio Uno R3** y la **Shield Imagina TDR STEAM**.

Bloques ▾ Información Archivos Ejemplo Shield Imagina TDR STEAM

- Lógica
- Control
- Matemáticas
- Texto
- Variables
- Listas
- Funciones
- Entrada/Salida
- Tiempo
- Puerto serie
- Bluetooth
- Sensores**
- Actuadores
- Pantalla LCD
- Pantalla OLED
- Memoria
- Motor
- Motor-Shield
- Keypad
- Reloj RTC
- GPS
- Tarjeta SD
- MQTT
- NeoPixel
- RFID

Pin A0 %

Pin 2 Lógica invertida

Pin 2 se ha pulsado Lógica invertida

Pin 2

Pin 2

Temperatura °C Pin 2

Temperatura °C Pin 2

Pin A0 %

Pin A0

Por último, y una vez tengamos finalizado el programa, la forma de transferirlo a la placa siempre será la misma.

Primero tenemos que asegurarnos que **ArduinoBlocksConnector** está en marcha. (Haremos click en su ícono y lo dejaremos abierto en un segundo plano).



ArduinoBlocks-Connector

```
>>> ArduinoBlocks-Connector v3
>>> by Juanjo Lopez
>>> www.arduinoblocks.com
>>> Listening on port 9987
>>> (Ctrl+C to finish)
>>> Running...
```

Después, comprobaremos que nuestra placa está detectada en el puerto correspondiente. En el caso de la imagen en el **COM11** (1). Si no fuera así daríamos al botón de refrescar (2).

Por último, con el botón “**Subir**” (3) transferimos el programa del ordenador a la placa.

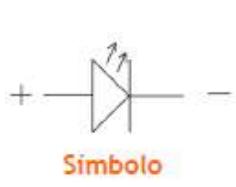
Actividades con la Shield Imagina TDR STEAM

A continuación, os proponemos una serie de actividades y prácticas para aprender a programar vuestras placas **Keyestudio UNO R.3** y la **Shield Imagina TDR STEAM** paso a paso para que realice infinidad de tareas.

A01. – El LED

Vamos a empezar con nuestro primer programa que será encender y apagar el LED Rojo correspondiente al Pin D12.

Un LED (Diodo emisor de luz, también "diodo luminoso") 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.



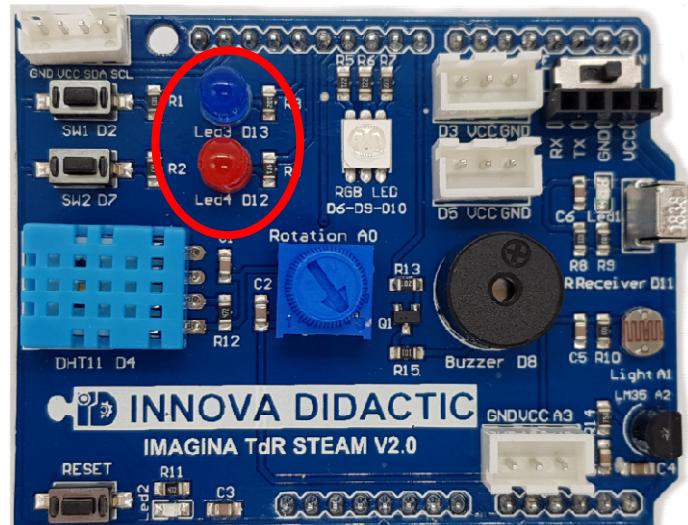
Símbolo



Componente

El LED tiene una polaridad, un orden de conexión, y al conectarlo al revés se puede quemar.

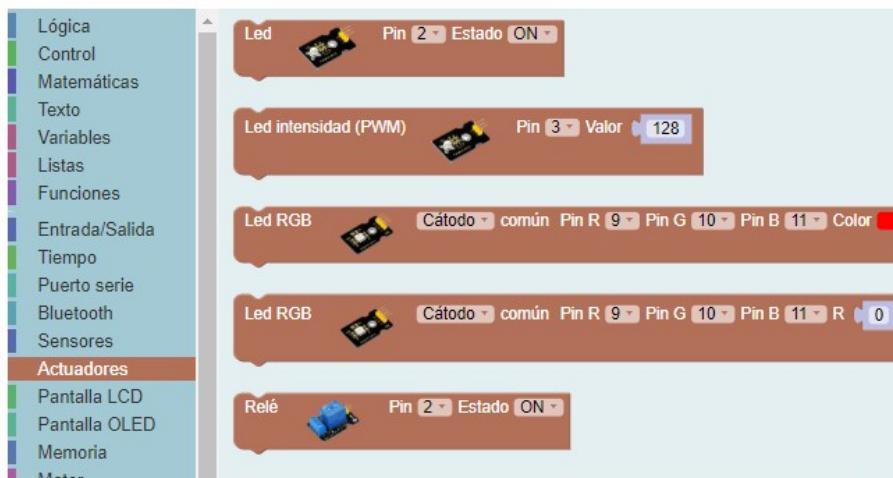
La **Shield Imagina TDR STEAM** dispone de dos LEDs (uno azul y otro rojo), conectados en los pines D13 (azul) y D12 (rojo)



En programación el primer programa que se realiza se le suele llamar “**Hola Mundo**”, así que vamos a realizar nuestro primer “Hola Mundo”.

A01.1.– On/OFF del LED Rojo (Pin 12)

Entramos en **ArduinoBlocks** en el *tipo de proyecto* el *Arduino UNO* y puesto el nombre (*Hola Mundo*), picharemos en el bloque de **ACTUADORES**.



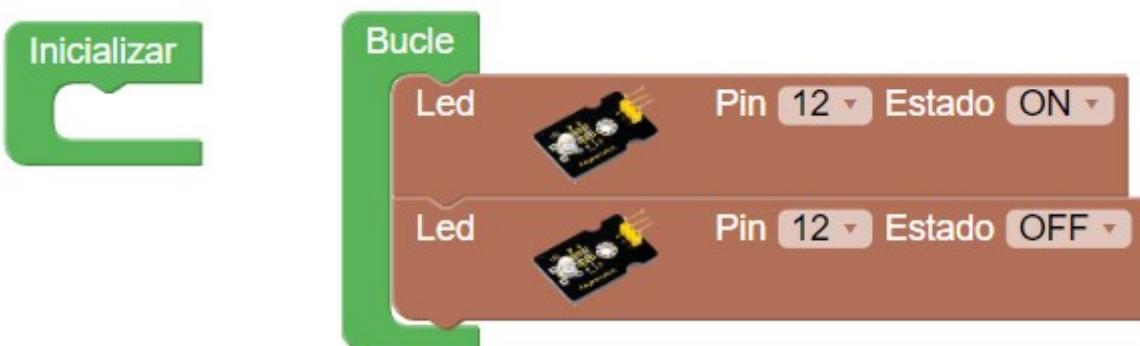
Clicamos sobre el bloque 1 y lo arrastramos sobre el área de programación. Fíjate que en el área de programación hay dos bloques verdes (*Inicializar* y *Bucle*) y que podríamos meter nuestro bloque del LED en cualquiera de ellos (*Los bloques cuando son compatibles encajan como un puzzle*). Pues bien, todo lo que se meta dentro del

bloque de ***Iniciar*** 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.

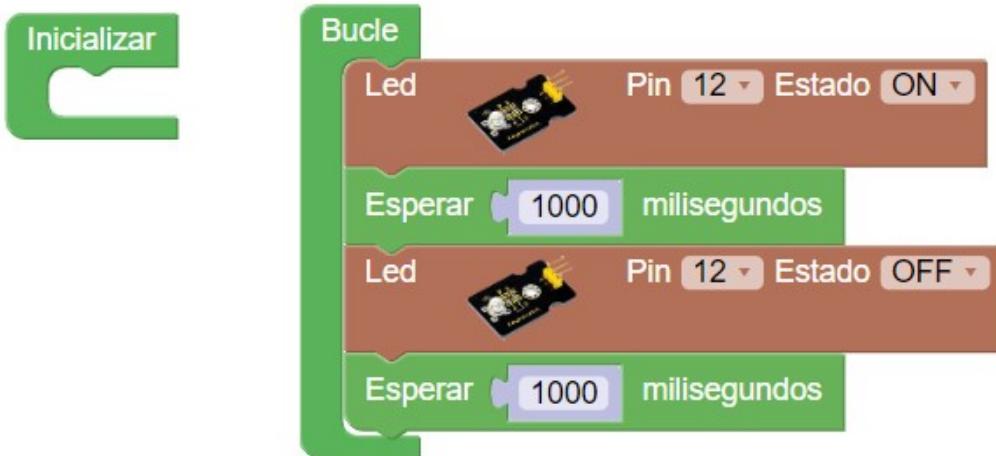
Vamos a meter nuestro bloque de LED en el *Bucle* y cambiamos al Pin D12. El LED puede tener dos estados; ON/OFF, que podemos cambiar en el menú despegable.



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



Pero este programa no es correcto del todo, no hay tiempos que indiquen cuánto tiempo tiene que estar encendido o apagado el LED. Necesitamos ir al bloque de **TIEMPO** y seleccionar *Esperar XXXX milisegundos* (recuerda; 1.000 milisegundos son 1 seg).



Ahora tenemos el LED encendido durante 1 seg. y apagado otro.

Ejercicio: Prueba a cambiar los valores del tiempo.

A01.2.– On/OFF del LED Rojo y Azul (Pin 12 y 13)

Como hemos comentado anteriormente la shield dispone de dos LEDs (Rojo y Azul), vamos a realizar un programa para que se vayan alternando su encendido y apagado.



Ejercicio: Haz que los dos leds se enciendan y apaguen a la vez.

A01.3.– On/OFF del LED Rojo y Azul (Pin 12 y 13)

Imagina que queremos hacer un ciclo de repetir 5 veces el encendido y apagado del LED rojo antes de que se encienda el azul, sería muy repetitivo, no?? Para ser más efectivo programando, en el menú de **CONTROL** existe el bloque “**Repetir ____ veces hacer...**”



En el siguiente programa fíjate como el LED rojo se enciende y apaga cada medio segundo 5 veces y después se queda el LED azul encendido durante 4 seg.



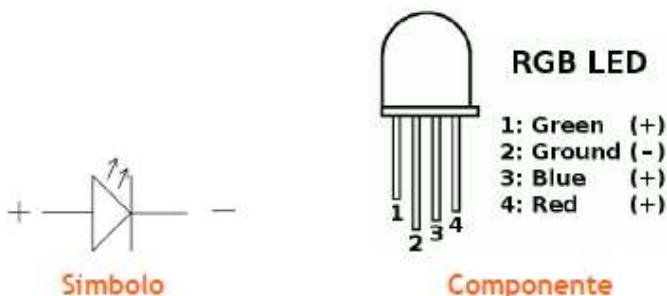
Ejercicio: Modificando ligeramente este programa, puedes conseguir que el LED azul se quede brillando indefinidamente.

Pista: Ve al menú TIEMPO y prueba con el bloque:

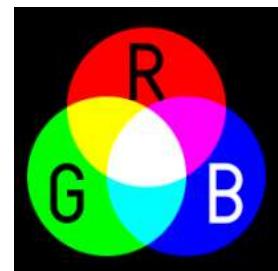
Esperar por siempre (fin)

A02. – El LED RGB PINes (D6-D9-D10)

Un **LED RGB** es un LED que incorpora en su mismo encapsulado tres LEDs. Es RGB porque R (red, rojo), G (green, verde) y B (blue, azul) así se pueden formar miles de colores ajustando de manera individual cada color. Los tres LEDs están unidos por el negativo o cátodo.



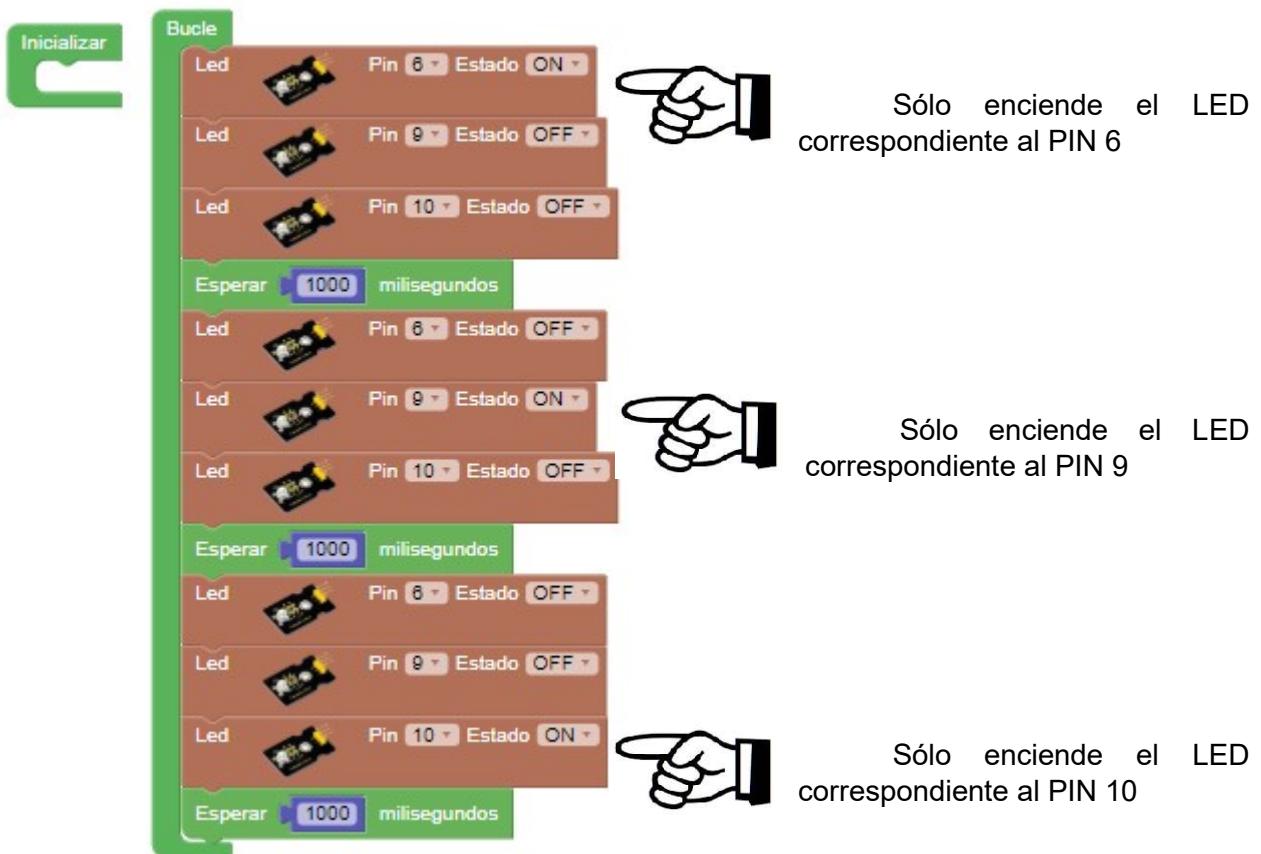
En Arduino cada uno de esos LEDs podría tomar 256 colores diferentes, es decir, el Rojo podría ir desde 0 hasta 255, el verde de 0 a 255 y el azul de 0 a 255, en total un LED RGB podría dar más de 16,5 millones de colores diferentes.



La **Shield IMAGINA TdR STEAM** dispone de un LED RGB conectado a los Pines (D6, D9 y D10)

A02.1.– Identificación de colores y Pines LED RGB

Con este sencillo programa vamos a identificar cada color del LED RGB con su pin correspondiente.



Ejercicio: ¿Ya has identificado las correspondencias de pines y colores?

Ejercicio: Prueba ahora a ir activando dos o tres pines a la vez y fíjate en los colores que emite el LED RGB.



A02.2.– Múltiples colores con el LED RGB.

Arduinoblocks tiene bloques específicos para facilitar al máximo la programación de los LEDs RGB. Si vamos al menú “**ACTUADORES**” nos encontramos con el bloque “**Led RGB**”



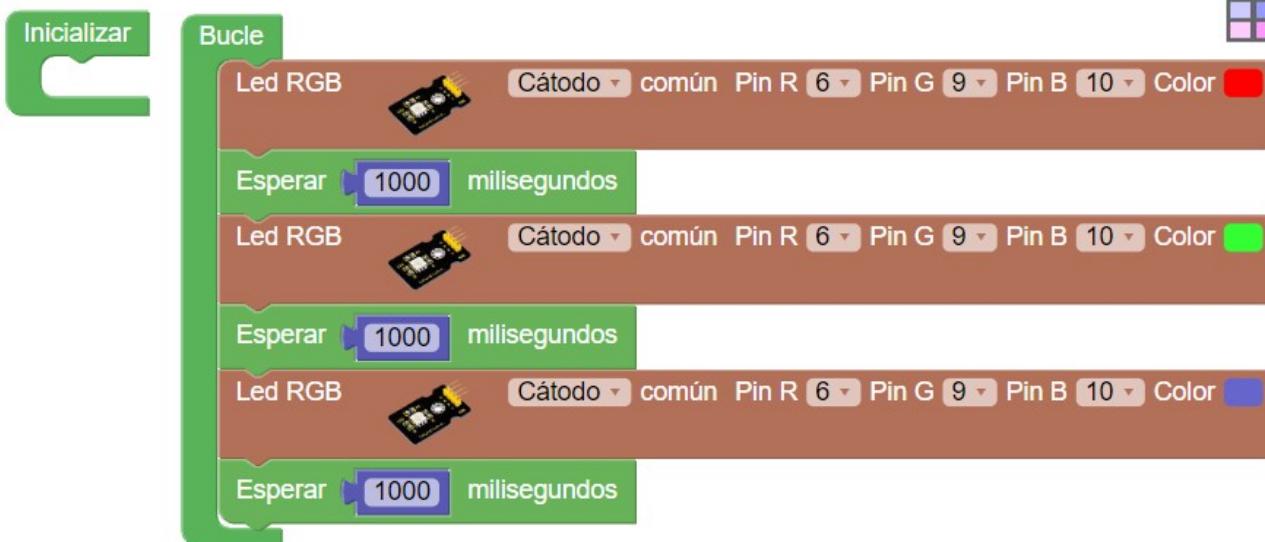
Al utilizar ese bloque debemos cambiar la numeración de los pines para hacerlo corresponder con los de la **Imagina TDR STEAM**. Deberán quedar así:



Comprueba como al clicar sobre el ícono del color se desplegará una paleta de colores para que puedas elegir cualquier color.



Ahora realiza el siguiente programa y comprueba su funcionamiento:



Ejercicio: Prueba a cambiar los colores y tiempos según tus preferencias.

A02.3.– Control PWM del LED RGB I.

Continuando con la práctica anterior, ahora vamos a controlar el brillo de un LED utilizando el PWM. Pero lo primero de todo, ¿Qué significa PWM? Bien, PWM es la abreviatura de ***pulse-width modulation*** (modulación de ancho de pulso).

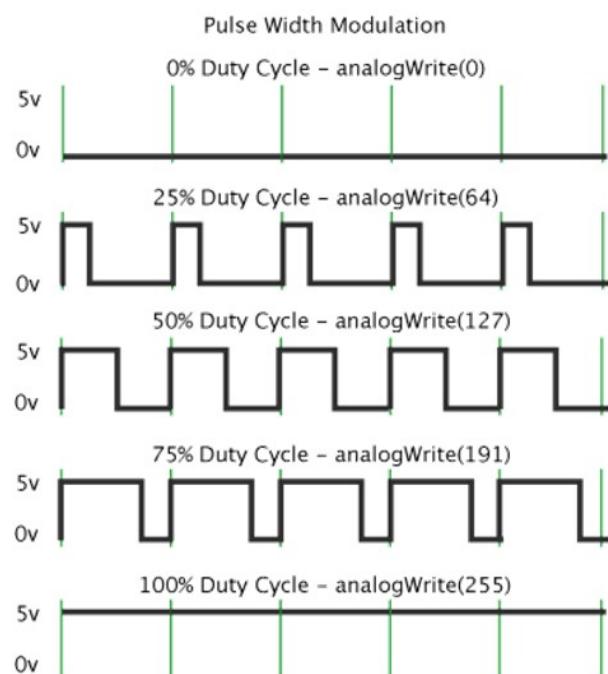
Las salidas de voltaje de Arduino sólo tienen dos estados ALTO/BAJO, ON/OFF, ENCENDIDO/APAGADO,... es decir, corresponden a una salida de 5 V (ON) y de 0 V (OFF). Con esto sólo podemos hacer actividades de encender y apagar un LED, no podríamos controlar su brillo de menos a más o viceversa. Sin embargo el PWM permite hacer un rango de valores de 0 a 255 entre el 0 y 5 V. De esta manera podemos controlar el brillo del LED y muchas cosas más.

La modulación de ancho de pulso, o PWM, es una técnica para obtener resultados analógicos con medios digitales.

El control digital se 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.

El PWM se utiliza mucho para controlar lámparas, velocidades de motores, producción de sonidos,...

El Keyestudio UNO R.3 tiene un total de 6 salidas PWM, que son digitales 3, 5, 6, 9, 10 y 11.



En la Shield **Imagina TDR STEAM** sólo se puede controlar por PWM el LED RGB, es decir, los Pines 6, 9 y 10.

El rango de **VALOR** lo podemos variar desde 0 hasta 255, de tal modo que si el valor es 0 el LED estará totalmente apagado y si el valor es 255 el LED brillará al máximo.

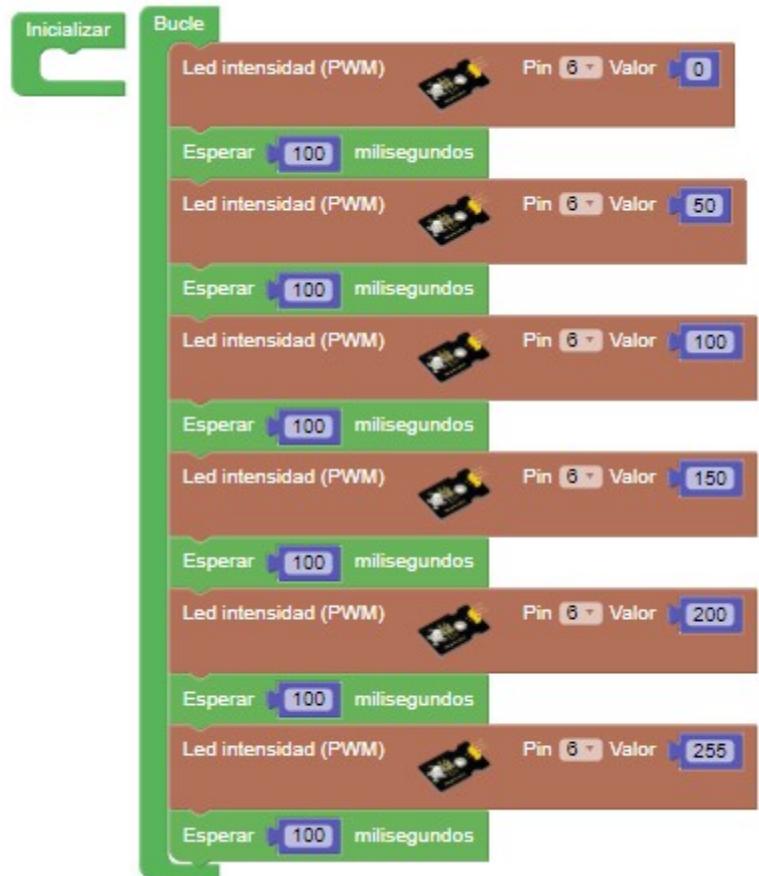
De esta manera podríamos encender y apagar un LED con un programa similar a este; pero *recuerda, los LEDs rojo y azul (Pin 12 y 13) no se pueden controlar por PWM*.



A02.4.– Control PWM del LED RGB II.

Ahora vamos a hacer que el brillo del LED varíe.

Empezaremos introduciendo un valor de 0 en el PWM para ir aumentándolo de 50 en 50 unidades hasta llegar a 255. Haremos una espera de 200 milisegundos entre un aumento de valor y otro. El programa quedaría como este:



Imagínate que en lugar de incrementar los valores de 50 en 50 lo quisieras hacer de 1 en 1, tendrías que tener 255 bloques más otros tantos de esperas... Como verás esta forma de programar no es muy efectiva, es muy repetitiva y tendría muchos bloques. Existe un bloque con el que podríamos hacer esto de una manera mucho más sencilla. Es el siguiente bloque:

contar con *i* desde 1 hasta 10 de a 1
hacer



Este bloque lo tenemos dentro de **CONTROL**.

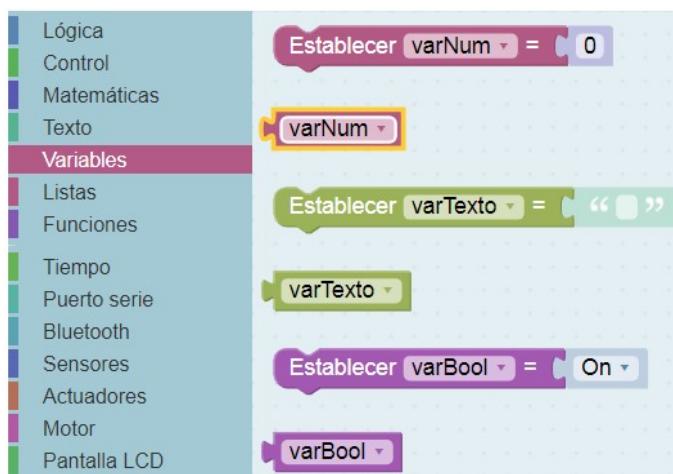
Este bloque tiene un concepto muy utilizado en programación que son las **VARIABLES**. En este bloque la variable se llama *i* (nombre que se puede cambiar). A grandes rasgos, una variable es una "cajita" en la cual se van introduciendo diferentes valores que el programa utilizará según necesite.



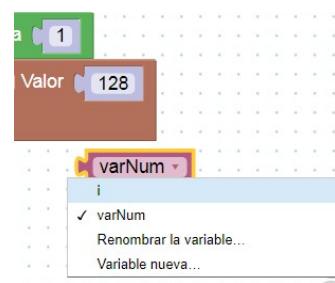
Este sería el programa de ir aumentando el brillo de un LED de 1 en 1 desde 0 hasta 255 con una espera de 25 milisegundos.



En el programa inicialmente el valor de la **VARIABLE i** vale 1, cada vez que repite el bucle su valor va aumentado de 1 en 1.

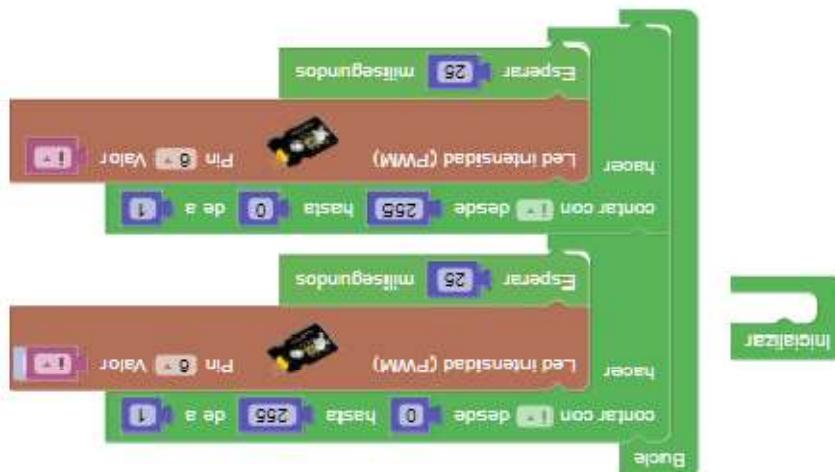


Para introducir el Valor del PWM, del bloque del LED, la variable **i**, debemos ir a **Variables**, seleccionar el segundo bloque y en el menú desplegable seleccionar **i**.



Ejercicio: Intenta completar el programa haciendo que el brillo del LED ascienda de 0 a 255 y que descienda de 255 a 0.

Solución:



A02.5.– Control PWM del LED RGB III.

Pero **ArduinoBlocks** aún nos guarda una sorpresa más, es un bloque específico dentro del menú ACTUADORES en el cual podemos programar los tres colores del LED RGB por PWM directamente.



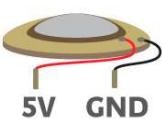
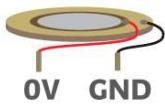
Basándote en el ejercicio anterior, prueba con este programa:



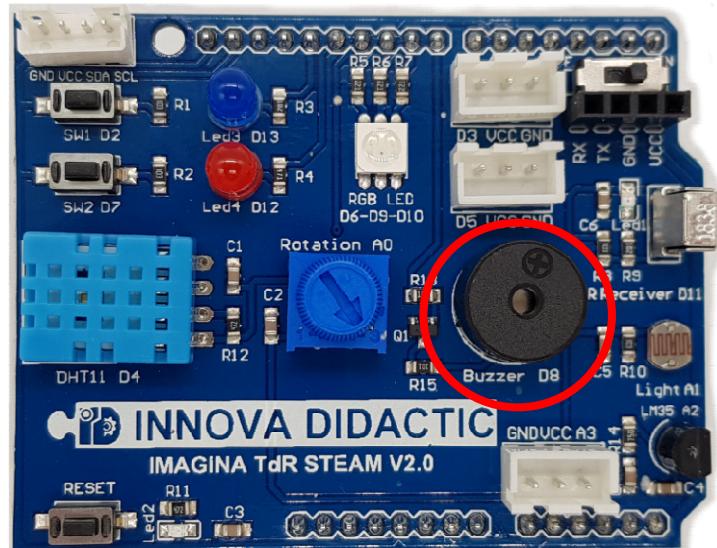
Ejercicio: Diviértete creando infinidad de colores con el LED RGB y el PWM.

A03. – Generar notas con el Buzzer o Zumbador.

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 buzzer Activo o Pasivo, este zumbido será del mismo tono o le podremos variar. Sirve como mecanismo de señalización o aviso y se utiliza en múltiples sistemas, como en automóviles o en electrodomésticos, incluidos los despertadores.



La **Shield Imagina TDR STEAM** tiene un Buzzer o Zumbador pasivo. Este buzzer está conectado en el Pin D8.



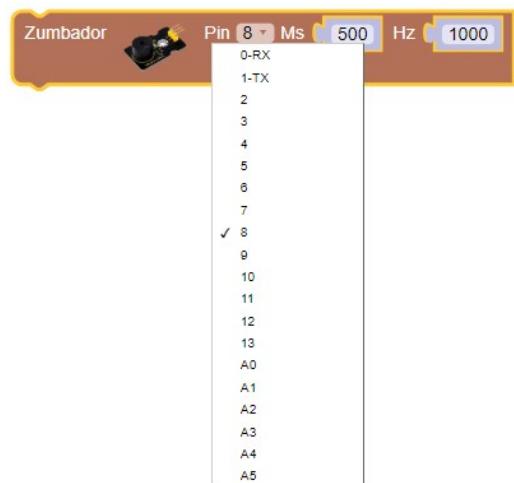
ArduinoBlocks tiene 4 bloques específicos para programar el buzzer están en el menú de ACTUADORES.

Sensores

Actuadores

- Pantalla LCD
- Pantalla OLED
- Memoria
- Motor
- Motor-Shield
- Keypad
- Reloj RTC
- GPS
- Tarjeta SD
- MQTT
- NeoPixel
- RFID

El Buzzer puede ser conectado en cualquiera de los pines digitales y también en los analógicos, pero esto último no es recomendable. En nuestro caso sólo es viable en el PIN D8.



A03.1.– Primeros sonidos con el Buzzer.

En el bloque Zumbador podemos variar dos parámetros; Ms (Milisegundos) (1) es el tiempo que dura cada sonido y Tono (2), que es la frecuencia a la que vibra la membrana para emitir el sonido.

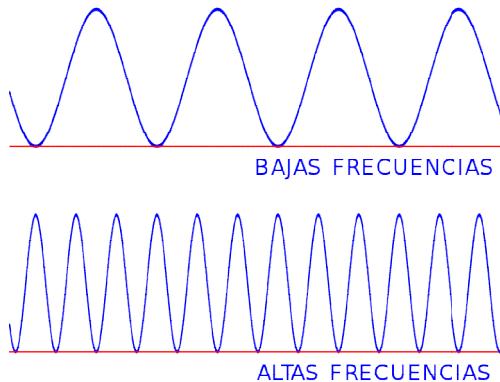


Prueba con este sencillo programa para ver cómo suena.



Cambia ligeramente el programa introduciendo unas esperas entre un sonido y otro. ¿Ves la diferencia?

A03.2.– Escala musical I.



¿Qué es la frecuencia? 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, dándonos una idea de la rapidez con que se producen.

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

Nota	Frecuencia (Hz)
do (control)	261.6
do#	277.2
re#	293.7
mi	329.6
fa	349.2
fa#	370
sol	392
sol#	415.3
la	440
la#	466.2
si	493.2
do	523.3



DO RE MI FA SOL LA SI DO DO SI LA SOL FA MI RE DO

<https://es.wikipedia.org/wiki/Frecuencia>

https://es.wikipedia.org/wiki/Escala_musical

Vamos a hacer una escala de DO a DO utilizando estos valores.

```

loop() {
    speaker(261.6, 500);
    speaker(277.2, 500);
    speaker(329.6, 500);
    speaker(349.2, 500);
    speaker(370, 500);
    speaker(440, 500);
    speaker(493.2, 500);
    speaker(523.3, 500);
}
  
```

Ejercicio: ¿Te atreves a programar esta melodía?



HIMNO DE LA ALEGRÍA (PARTE 1)

| MI MI FA SOL | SOL FA MI RE | DO DO RE MI | MI RE RE |
 (1) (2) (1) (2)

| MI MI FA SOL | SOL FA MI RE | DO DO RE MI | RE DO DO |

(1) (2) (1) (2)

A03.3.– Escala musical II.

Es un poco engoroso tener que estar introduciendo los valores de cada frecuencia en cada nota, ¿no te parece? Para solucionar esto **ArduinoBlocks** dispone de otro bloque en el menú **ACTUADORES**, el **TONO (Hz)**

De esa forma podremos programar el buzzer sin necesidad de usar las frecuencias.



Mucho más sencillo así, verdad??



A03.4.– Melodías RTTTL.

¿Qué es eso de melodías RTTTL? RTTTL Es un formato de codificación de notas creado por Nokia en *Melodías Móviles*. RTTTL Quiere decir "*Ringing Tones Text Transfer Language*" y fue el formato más común para melodías móviles en Internet.

¿Y cómo puedo usar melodías RTTTL con mi shield **Imagina TDR STEAM** y **Arduinoblocks**? Pues es muy sencillo, sólo necesitas estos dos bloques:



Haz este programa y fíjate la cantidad de melodías que tienes disponibles al clicar sobre el menú desplegable del bloque RTTTL;



- ✓ The Simpsons
- Indiana Jones
- Muppets
- Looney
- 20th Century Fox
- Star Wars
- A-Team
- Mission Impossible
- Gadget
- Bubble Bobble
- Arkanoid
- Donkey Kong
- Pac-Man
- Tetris
- Super Mario
- Addams Family
- Popeye
- Beethoven
- Ghostbusters

Pruébalo!!! Es muy divertido...

Si quieres crear o incorporar nuevas melodías RTTTL mira en este enlace:

<http://arduinoblocks.didactronica.com/2019/11/melodias-rtttl/>

A04. – Sensor pulsador.

En la siguiente actividad vamos a utilizar el pulsador. Previamente debemos recordar la diferencia 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 tocarlo. ¿Os imagináis que el timbre de vuestra casa fuera un interruptor y no un pulsador?

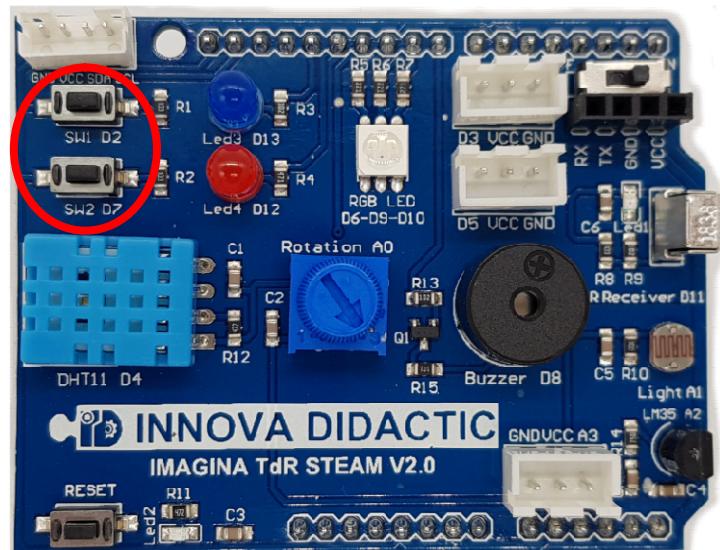


Hay dos tipos de pulsadores; los NA (normalmente abierto) o los NC (normalmente cerrado), con lo que al pulsarlo se activará la función inversa de la que en ese momento este realizando.

La Shield **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 no lo pulsemos.

En el menú de **SENSORES** encontramos los dos bloques correspondientes al pulsador y el pulsador filtrado.

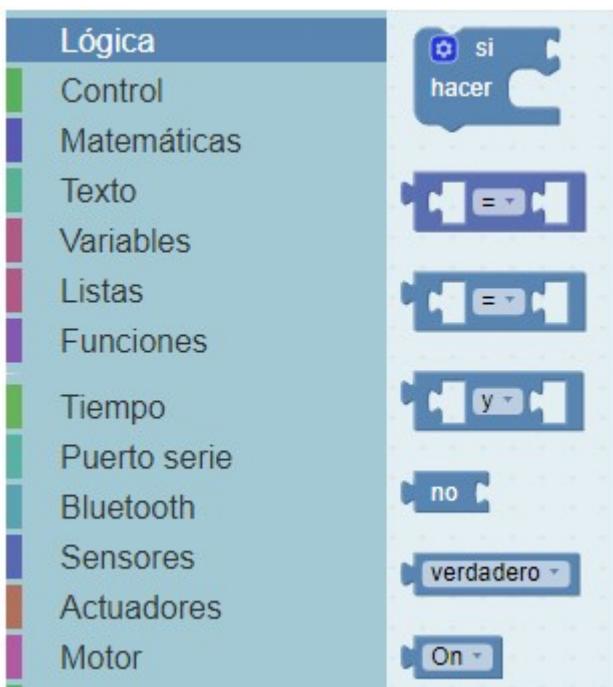


The screenshot shows the ArduinoBlocks IDE interface. On the left, there is a sidebar with categories: Texto, Variables, Listas, Funciones, Entrada/Salida, Tiempo, Puerto serie, Bluetooth, **Sensores**, Actuadores, Pantalla LCD, and Pantalla OI FD. A hand icon is also present. On the right, three sensor blocks are listed:

- Pulsador**: Pin 2, Lógica invertida
- Pulsador (filtrado)**: Pin 2, se ha pulsado, Lógica invertida
- Pulsador táctil**: Pin 2

A04.1.– ON/OFF LED con Pulsador.

Para realizar este programa necesitamos conocer unas de las funciones más utilizadas en programación, las funciones del menú “Lógica” con las funciones de condición.

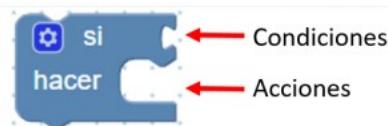


Condiciones “Si...hacer”.



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.

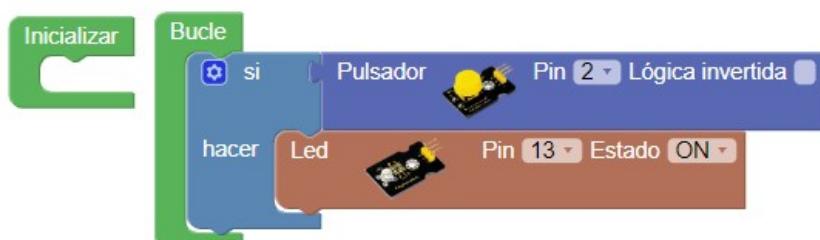
Funciona como una oración condicional en español, si se cumple la condición incluida en su primer apartado, entonces se realiza la acción incluida en su segundo apartado. En caso contrario, no se hace nada.



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, pero ¿qué va a ocurrir? Pruébalo.

No se apaga el LED azul nunca, ¿no?, es lógico, en ningún momento del programa decimos que el LED tenga que estar en la posición OFF.



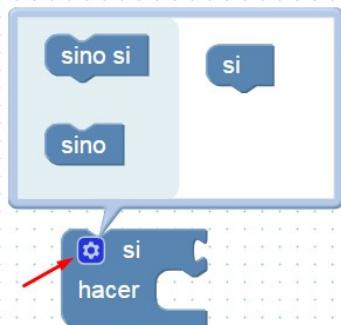
Con este otro programa al pulsar el SW1 se enciende el LED azul y al pulsar el SW2 se apaga.



A04.2.– ON/OFF LED con Pulsador I.

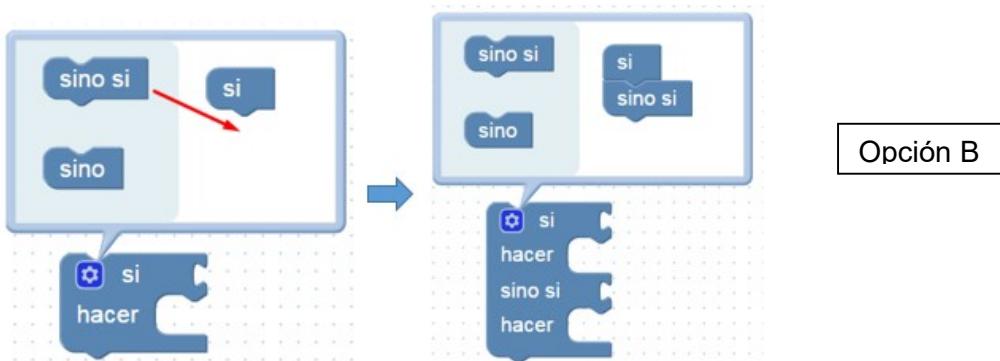
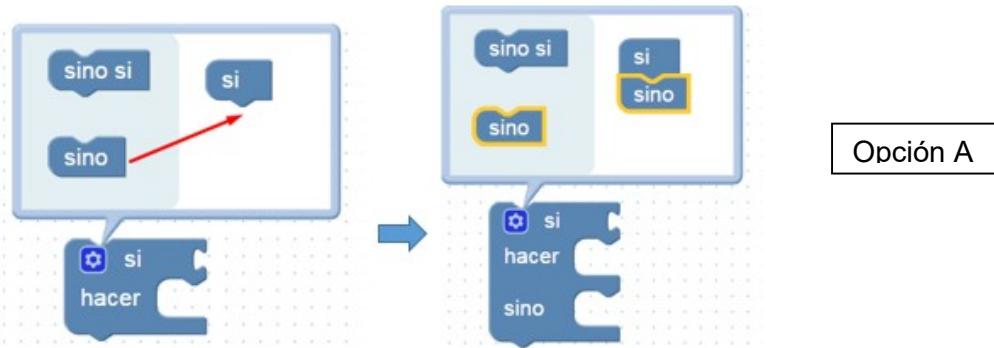
Necesitar dos pulsadores para tener que controlar un solo Led no parece la mejor opción, necesitamos introducir un “**SINO**”

Usando un único debemos poner una nueva condición; un **SINO** para en ella cambiar el estado del LED a OFF. Para ello debemos ampliar el condicional de la siguiente manera;

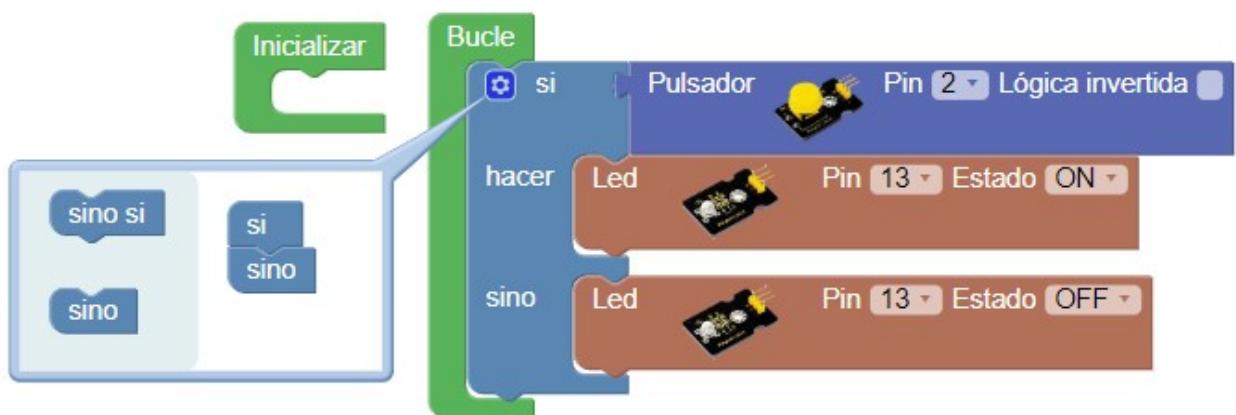


Haciendo clic sobre el símbolo del engranaje señalado con la flecha roja en la imagen, nos aparece un cuadro con funciones con las que podemos ampliar el condicional “Si”.

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



Veremos ejemplos del uso de estas variantes a lo largo de diferentes programas en este documento. Pero continuando con esta actividad, realizaremos un programa que al pulsar el pulsador se encenderá el LED y **sino**, se apagará.



¿Sabes lo que es 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 Buzzer.

Este es el código Morse:

Fuente:

https://es.wikipedia.org/wiki/C%C3%B3digo_morse

INTERNATIONAL MORSE CODE

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to five dots.

Ejercicio: Haz una máquina de código Morse acústica e intercambia mensajes con tus compañeros.



A • -	U • • -
B - - . .	V • • • -
C - - . -	W • - -
D - - .	X - - . • -
E •	Y - - • -
F • • - -	Z - - - . •
G - - -	
H	
I ..	
J . - - -	
K - - . -	1 . - - - -
L . - - .	2 . • - - -
M - -	3 . . . -
N - - .	4 -
O - - -	5
P . - - - .	6 -
Q - - - . -	7 - - . . .
R - - .	8 - - - . .
S . . .	9 - - - - .
T -	0 - - - -

Ahora hemos conseguido que cuando el pulsador no esté accionado el LED se apague y sólo se encienda cuando lo pulsemos. Pero ¿Cómo conseguir que se quede encendido sin tener que estar presionando el pulsador continuamente?

A04.3.– ON/OFF LED con Pulsador II.

En esta actividad vamos a hacer que el Led AZUL se quede en estado encendido o apagado pulsando una sola vez el pulsador SW1.

Empecemos viendo la diferencia entre estos dos bloques: “**PULSADOR**” y “**PULSADOR FILTRADO**”



Hagámonos las siguientes preguntas; ¿Cada vez que apretamos un pulsador lo hacemos durante el mismo tiempo? ¿Cada persona acciona el pulsador durante el mismo tiempo? La respuesta es no. Esa es la razón del uso del bloque de “**pulsador filtrado**”, gracias a este bloque el problema lo tenemos solucionado. Sólo detecta una pulsación bien al ser pulsado o al ser soltado.



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*”.



En INICIALIZAR establecemos la variable a 0 y la llamamos ESTADO



El contador consiste en ir sumando una unidad a la variable “ESTADO” cada vez que se dé al pulsador, esto se consigue de la siguiente manera; En un condicional al accionar pulsador, la variable tendrá su valor inicial más 1. Por ejemplo; al inicio del programa la variable “ESTADO” tiene un valor de 0, al dar al pulsador su nuevo valor será igual a 0+1; al volver a dar al pulsador su valor será 1+1; al volver a pulsar 2+1....

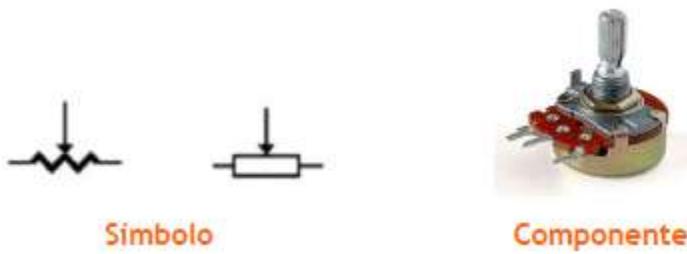


De esa manera, cuando “ESTADO” tiene un valor de 0 el LED estará apagado, cuando “ESTADO” tenga un valor de 1 el LED estará encendido y cuando se vuelva a dar al pulsador el valor de “ESTADO” será 2 y se le mandará volver a un valor de 0 por lo que el LED estará apagado nuevamente. Este sería el programa;



A05. – Potenciómetro

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 o brillo.



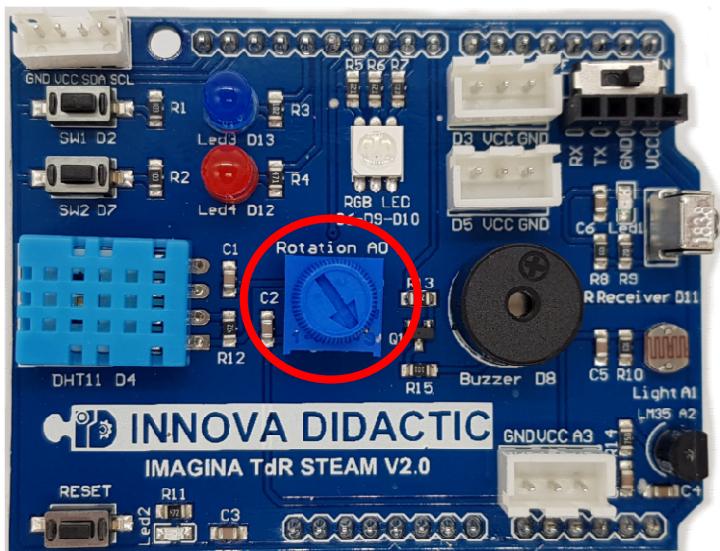
La Shield **Imagina TDR STEAM** tiene un potenciómetro denominado “Rotation” que van asociado al pin A0.

Empezamos con el uso de las entradas en los *pines Analógicos*.

La diferencia entre un sensor **analógico** y **digital** es que mientras este último, el digital, sólo permite dos tipos de entradas, 0 o 1, *alto o bajo, high o low, on o off*,... un sensor analógico puede tener infinidad de valores. En Arduino las entradas analógicas pueden tener 2^{10} valores, 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.

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



Vamos con el primer programa...

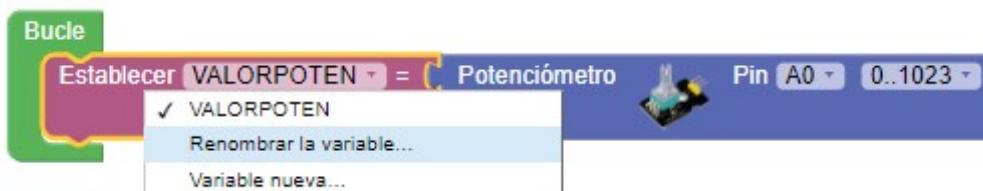
A05.1.– Lectura valor Potenciómetro por el Puerto Serie.

Para realizar una lectura de los valores del sensor es necesario utilizar la consola serie que nos ofrece **Arduinoblocks**, vamos a ver como se hace.

En primer lugar generamos una variable a la que llamaremos “ValorPoten”. Vamos al menú “**Variables**” y seleccionamos “**Establecer..... =...**”



Ahora fijaremos el valor de la variable al valor del potenciómetro tal y como está en la imagen.



Para cambiar el nombre de la variable clicaremos 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 aceptar.

www.arduinoblocks.com dice

Renombrar todas las variables «VALORPOTEN» a:

VALORPOTEN

Aceptar

Cancelar

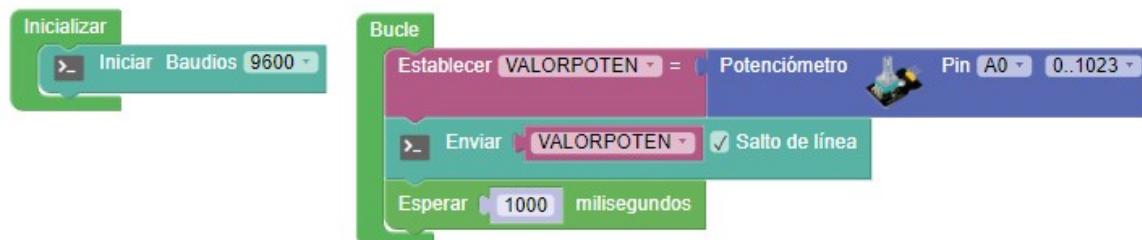
Es importante establecer la variable con el valor del potenciómetro dentro del BUCLE ya que si sólo se hace en INICIALIZAR el valor siempre será el mismo a lo largo de todo el programa. En otras ocasiones interesa establecer las variables en el inicio, pero es este caso no.

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....**





Fíjate como queda el programa:



Dentro del bloque **Enviar...** debemos colocar el valor de la variable creada (ValorPoten).



Sube el programa y después clica sobre el botón de **Consola**.



Se abrirá la siguiente ventana y clicaremos sobre el botón conectar. De esa manera podremos ver cada segundo el valor de nuestro potenciómetro. Ve girando el potenciómetro y mira como van cambiando los valores.

ArduinoBlocks :: Consola serie

Baudrate: Conectar Desconectar Limpiar

```
281.00
343.00
383.00
418.00
428.00
428.00
428.00
428.00
```

Es muy útil hacer lecturas por el puerto serie, a lo largo del manual lo seguiremos usando.

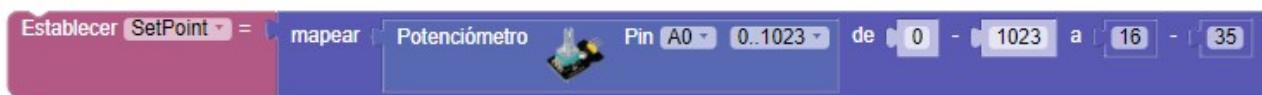


A05.2.– Uso del bloque MAPEAR y el Potenciómetro.

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.



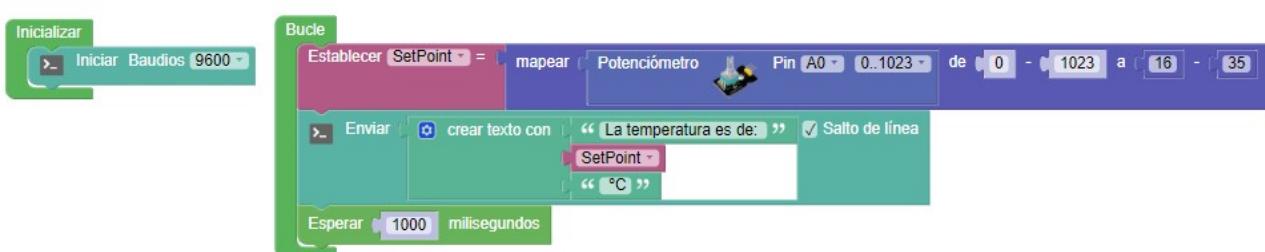
En esta actividad vamos a imaginar que con el potenciómetro queremos definir el SetPoint de un calefactor y sus límites están en un rango entre 16 y 35 °C. Para ello definiremos una variable, llamada SetPoint, al valor *mapeado* del potenciómetro.



Ampliaremos el programa anterior para realizar lecturas por el puerto serie utilizando el nuevo bloque de “**crear texto con...**”. Fíjate como al clicar sobre el símbolo del mecanismo podemos ampliar las líneas añadiendo “varNum” a la parte derecha.



Después añade el bloque y termina el programa como la imagen;

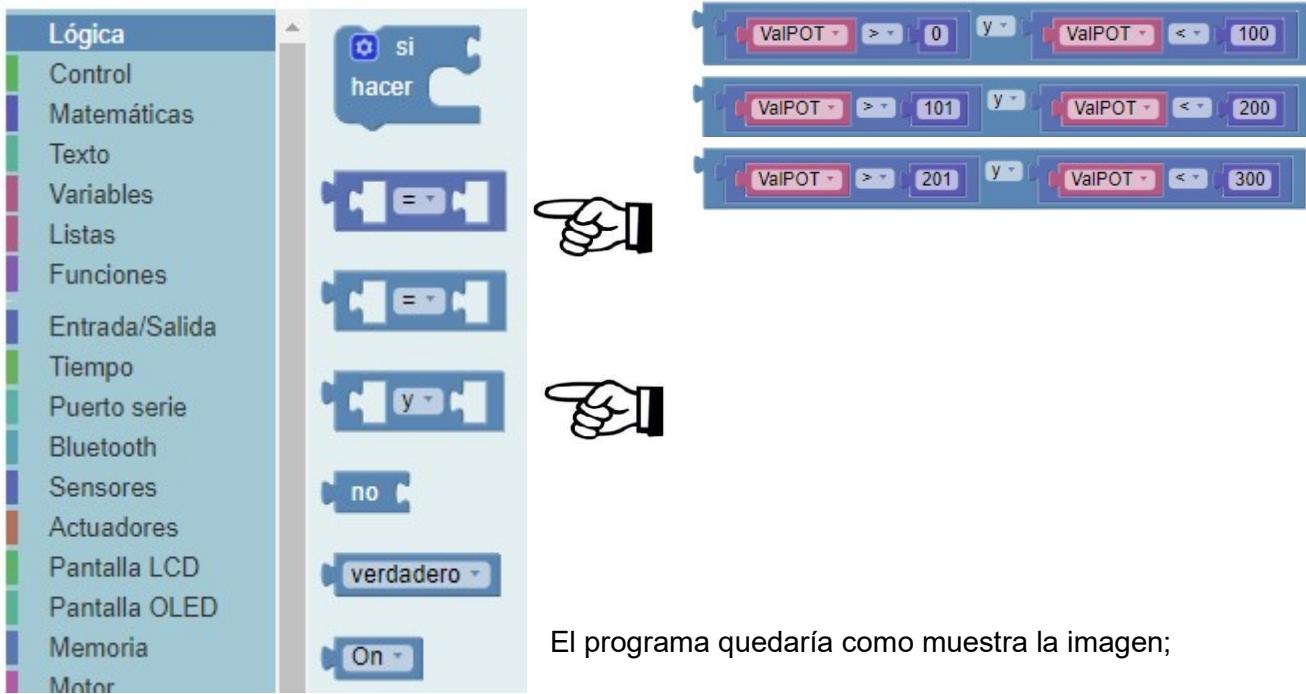


Por último, carga el programa, abre la consola serie y comprueba las lecturas moviendo el potenciómetro.

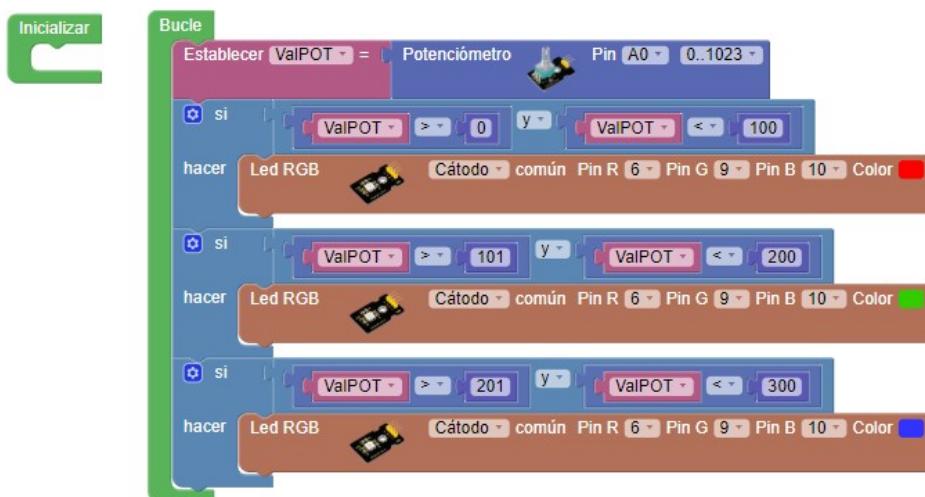
A05.3.– Control del LED RGB con el Potenciómetro.

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 300 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:



El programa quedaría como muestra la imagen;



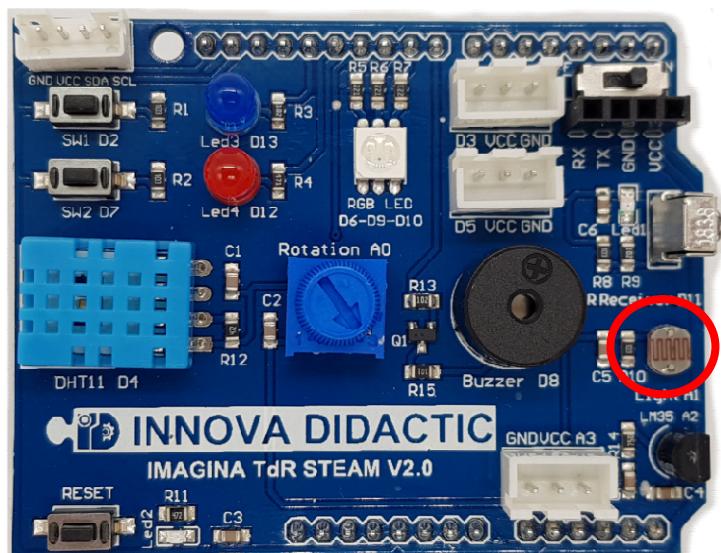
Ejercicio: Completa el programa con más condiciones y más colores hasta llegar a 1023.

A06. – Lectura del valor de la fotocélula (LDR).

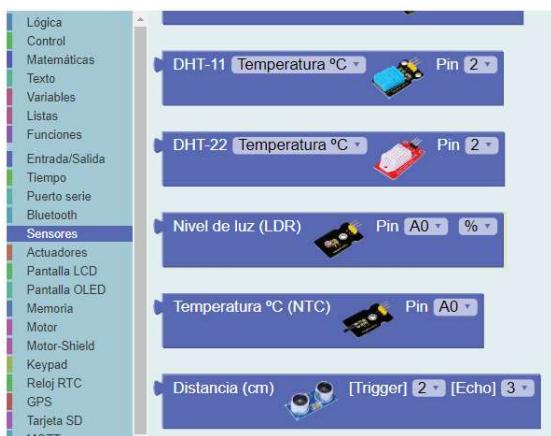
Ahora que ya sabemos usar el Puerto Serie para leer los valores de los sensores, vamos a utilizarlo para ver el valor de una fotocélula (LDR). ¿Qué es una LDR? Una LDR es un resistor que varía su valor de resistencia eléctrica dependiendo de la cantidad de luz que incide sobre él. El valor de la resistencia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



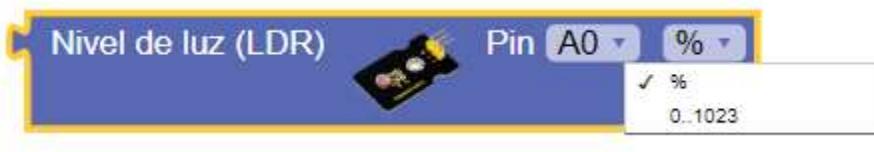
En la Shield **Imagina TDR STEAM** la fotorresistencia está denominada como “light” y viene conectada en el Pin analógico A1



En el menú “**Sensores**” de **ArduinoBlocks** hay un bloque específico para el uso de este sensor.



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



A06.1.– On/Off de 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 el Pin A1 y la lectura en %.

Por último un condicional en el cual cuando el valor sea menor de 20 que se encienda el LED, sino que permanezca apagado.

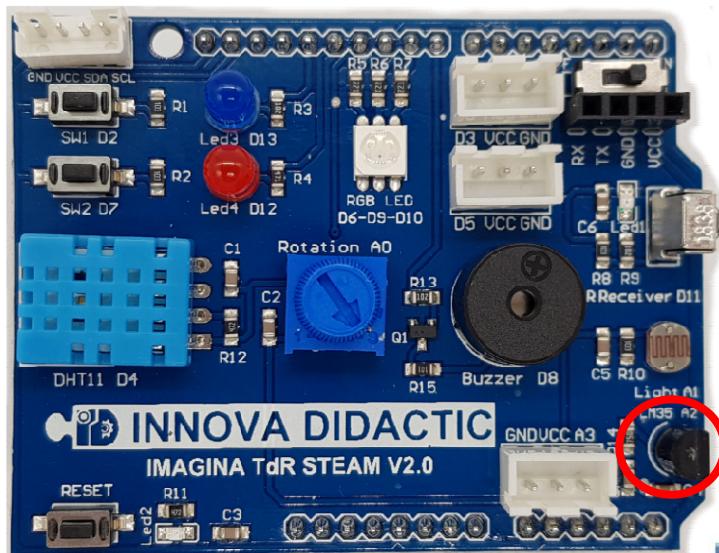


Ejercicio: Siempre es recomendable realizar una lectura por la consola serie de los valores de nuestros sensores. Repasando la actividad anterior en la que leímos el valor del potenciómetro por el puerto serie intenta hacer lo mismo con la LDR.

A07. – Medicción de temperatura de una habitación.

En la siguiente actividad vamos a medir la temperatura de una habitación utilizando el sensor de temperatura LM35D.

El sensor de temperatura LM35D tiene un rango de temperatura de 0º a 100º °C y una sensibilidad de 10 mV por grado °C.



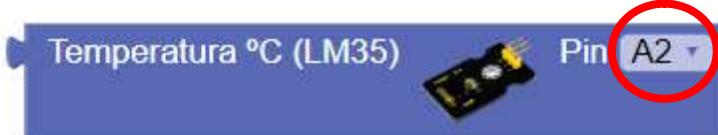
La Shield **IMAGINA TDR STEAM** dispone de este sensor LM35D y está conectado en el Pin analógico A2.

En el menú “**Sensores**” de **ArduinoBlocks** hay un bloque específico para el uso de este sensor.

The screenshot shows the ArduinoBlocks interface. On the left, there is a sidebar with a hand icon pointing to the "Sensores" section, which is highlighted in blue. The sidebar also lists other categories like Lógica, Control, Matemáticas, Texto, Variables, Listas, Funciones, Entrada/Salida, Tiempo, Puerto serie, Bluetooth, and Keypad. On the right, there is a list of sensors with their respective pins and data types:

- Calidad del aire (Pin A0, %)
- Nivel de luz (TEMT6000) (Pin A0, %)
- Temperatura °C (LM35) (Pin A0, °C)
- Temperatura °C (TMP36) (Pin A0, °C)

Recuerda que para programar el sensor hay que elegir el Pin A2.



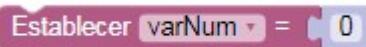
A07.1.– Lectura del valor de la temperatura.

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.

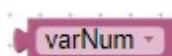
Las variables son elementos muy comunes en programación. Básicamente, crear una variable es darle un nombre a un dato o a una lectura. Por ejemplo, las mediciones de valores de temperatura las podemos guardar en una variable que se llame “Temperatura” o las del sensor de ultrasonidos en una llamada “Distancia”, 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:

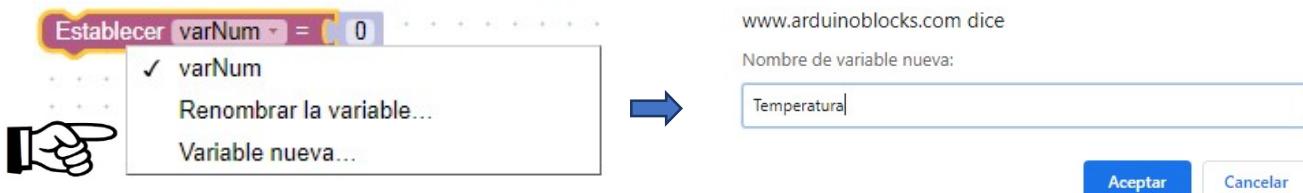
1. El bloque en el que le damos valor a la variable:



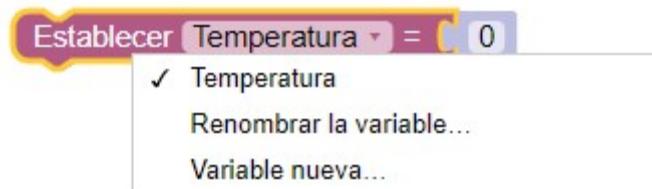
2. Y el bloque de la propia variable creada, para poder insertarla y combinarla con otros bloques:



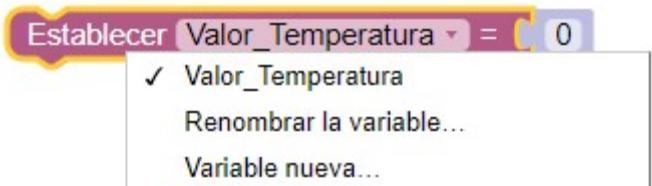
También podemos personalizar el nombre de la variable, de la siguiente forma:



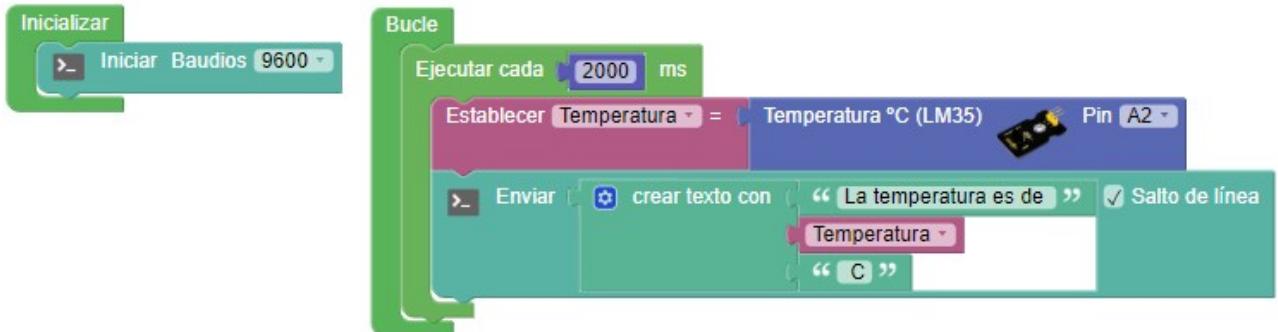
Una vez creada la nueva variable, podemos seleccionarla haciendo clic sobre el desplegable:



Ten 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”.



Ahora vamos a realizar el programa, recuerda el programa que hicimos para fijar el SetPoint con el Potenciómetro. Esto programa será igual pero cambiaremos el valor del potenciómetro por el valor del sensor de temperatura.



Sube el programa a la placa, conceta la consola serie y comprueba los resultados.

ArduinoBlocks :: Consola serie

Baudrate: 9600 Conectar Desconectar Limpiar

```

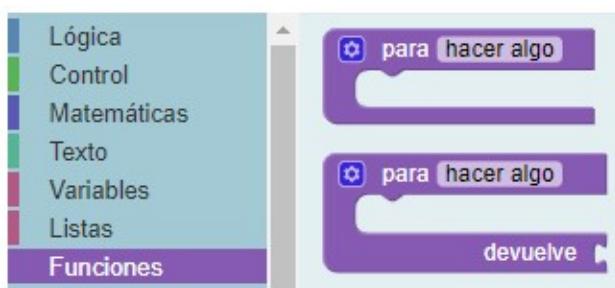
La temperatura es de 23.44 C
La temperatura es de 24.90 C
La temperatura es de 24.90 C
La temperatura es de 25.39 C
La temperatura es de 25.88 C
La temperatura es de 25.88 C
La temperatura es de 26.37 C
La temperatura es de 25.88 C
  
```



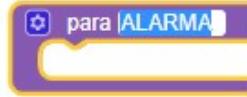
A07.2.– Alarma por exceso de temperatura.

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

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.



Debes escribir el nombre de ALARMA dentro de la función.



Al hacer esto automáticamente en el menú “Función” aparecerá un nuevo bloque llamado ALARMA.



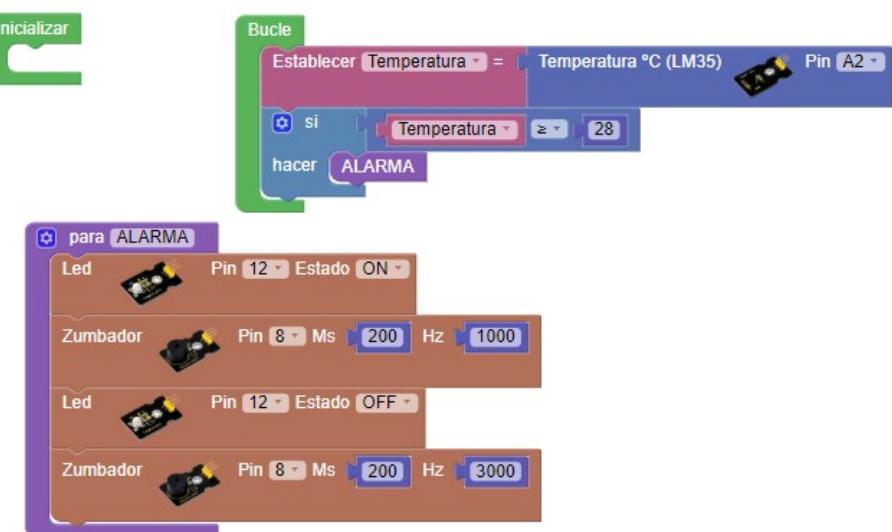
En el menú “Funciones” tenemos el bloque “para _____ hacer algo”



Lo utilizaremos para crear nuestra función como la siguiente imagen;



El programa final sería el siguiente:

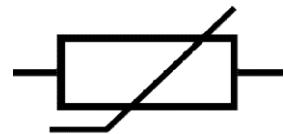


Ejercicio: Varía el programa anterior para que cuando la temperatura sea inferior a los 28°C el LED RGB esté verde.

A08. – Temperatura y Humedad. Sensor DHT11

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 0 a +50 °C con un error de +-2°C 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. Su funcionamiento se basa en la variación de la resistividad que presenta un semiconductor con la temperatura.



<https://es.wikipedia.org/wiki/Termistor>

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.

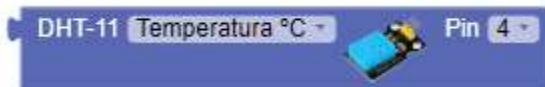
.- Los que tienen un coeficiente de temperatura positivo (en inglés *Positive Temperature Coefficient* o PTC), los cuales incrementan su resistencia a medida que aumenta la temperatura.

La Shield **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.



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



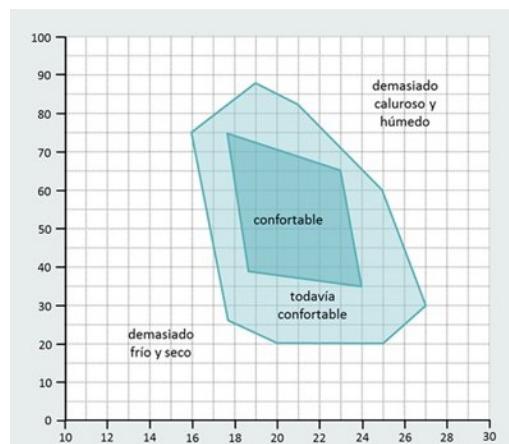
A08.1.– Confort higrométrico con DHT11.

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.

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 16 °C y superior a 26,5 °C.

Zona NARANJA:

- .- Humedad entre el 20 % y el 40% y entre el 65% y el 85%.
- .- Temperatura entre los 16 °C y los 18°C y entre los 24 y los 26,5 °C.

Zona VERDE:

- .- Humedad entre 40 % y el 65%.
- .- Temperatura entre 18 °C y 24 °C.

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.



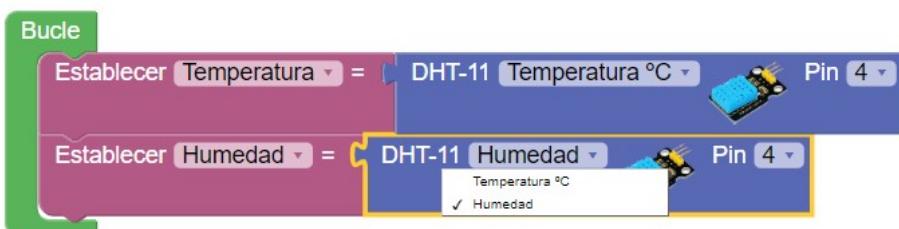
Para realizar este programa necesitaremos varios de los bloques del menú de Lógica. Necesitaremos evaluar una **condición Lógica** y utilizar **conjunciones** y **disyunciones**.

“Y” Se cumple si los dos operandos son verdaderos.

“O” Se cumple si alguno de los dos operandos es verdadero.



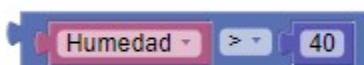
Antes de evaluar las condiciones debemos establecer las dos variables; la variable **Temperatura** y la variable **Humedad**. Recuerda que en el menú desplegable del sensor DHT-11 debes elegir la Temperatura °C o la Humedad y que está conectado al Pin D4;



Usando 3 bloques de conjunciones debes crear el siguiente bloque;



Después ir metiendo las condiciones en cada uno de ellos;



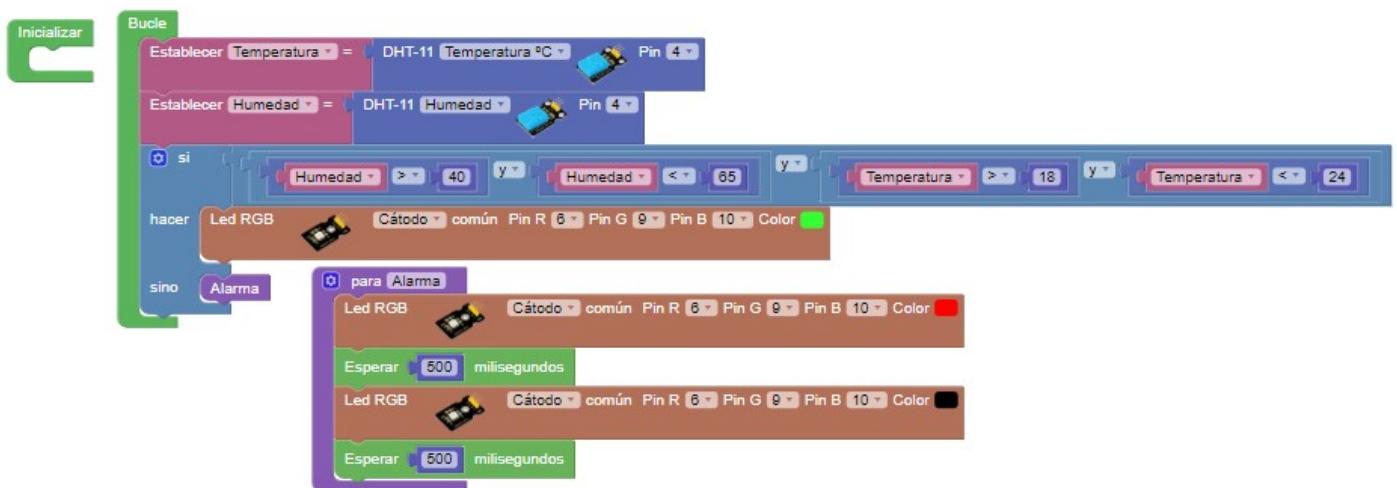
Y ve uniendo todo hasta conseguir esta condición final;



Por último debes crear una función, que la puedes llamar “Alarma”, mira como el dar un color negro al Led RGB es apagarlo.



Y con todo esto, el programa final quedaría así;



Ejercicio: ¿Te animas a realizar el programa final en el que estén reflejadas las tres zonas con sus colores correspondientes? Ínténtalo!!!

Ejercicio: Otra práctica que con los conocimientos que ya tienes podrías hacer el realizar una lectura por el puerto serie de los valores de Temperatura y Humedad del Sensor DHT-11.



A09. – Receptor IR.

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.



En el caso del receptor de infrarrojos (IR) de la shield **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, 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 uno o dos LED (*Light Emitting Diode*) que emiten 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 nuestra Arduino.

En la Shield 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.



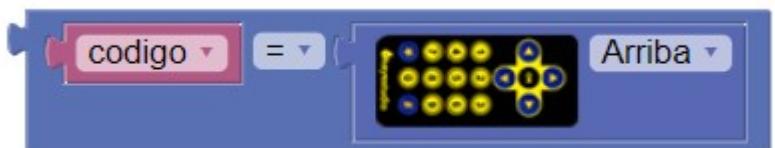
A09.1.– Test de receptor IR.

Vamos a realizar una pequeña actividad en la que utilicemos el receptor IR y el mando emisor de Keyestudio.

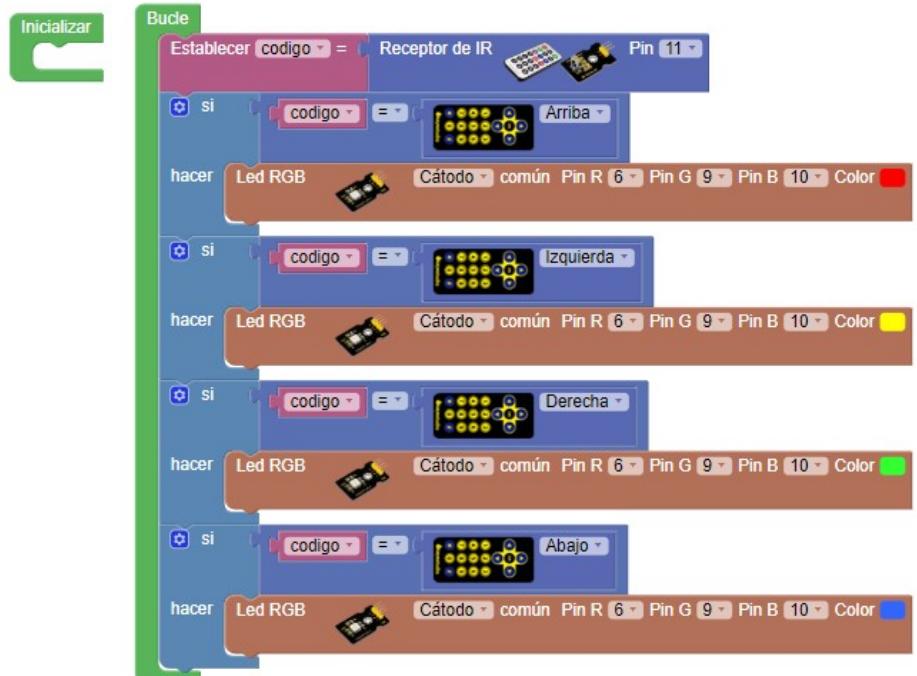
Primero crearemos una variable a la que llamaremos “Codigo” (*sin acento*) y la estableceremos con el bloque del Receptor IR;



Seguidamente estableceremos la siguiente igualdad y lo haremos como condición para encender el Led RGB en color Rojo.



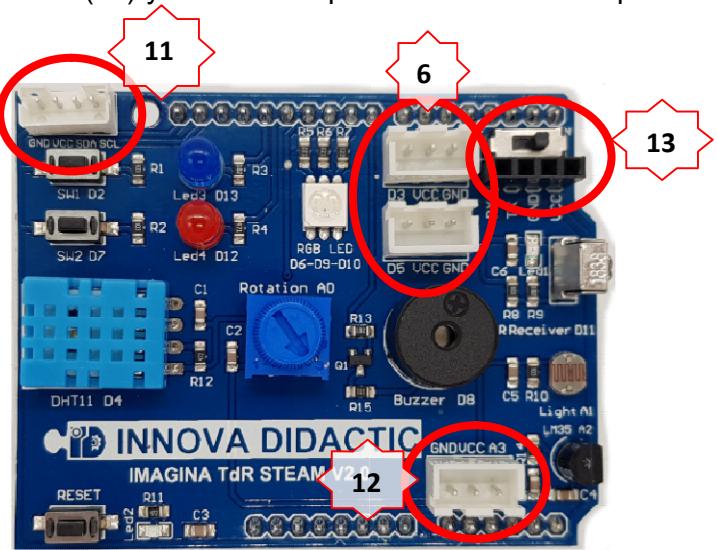
Podemos completar el programa dando distintas acciones a distintos botones, en este caso sólo usaremos las flechas, pero igualmente podrían ser los números.



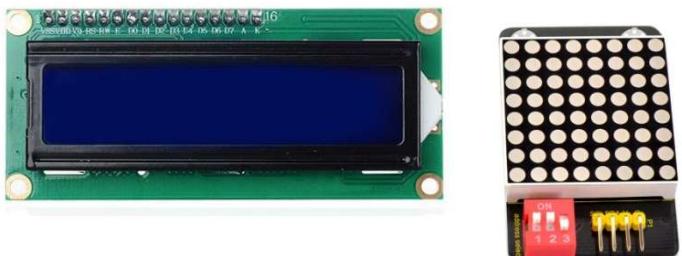
A10. – Puertos de expansión.

La shield **Imagina TDR STEAM** dispone de dos puertos de expansión digitales (6), uno analógico (12), un puerto de comunicaciones por I2C (11) y conexiones para comunicaciones por Bluetooth o wifi (13).

En los analógicos y digitales (tres en total) se pueden conectar infinidad de sensores y de actuadores, como servomotores, motores cc con driver, relés, micrófonos, sensores de ultrasonidos, medidores de humedad, sensores de gas, sensores de llama, de colisión, de detección de movimientos PIR, de efecto hall,



Por el puerto de comunicaciones I2C podemos conectar pantallas LCD, matriz de Leds 8x8, pantallas OLED, sensores acelerómetros, sensores de infrarrojos, módulo reloj, ...



Por último, en las conexiones para comunicaciones podemos conectar un módulo bluetooth o un módulo wifi ESP01 junto a su zócalo.



¿Te imaginas la cantidad de proyectos que podrás crear?

¡Hasta aquí el manual de iniciación con la shield IMAGINA TDR STEAM!!!

Lo que hemos visto es solo una pincelada del mundo Arduino. Hay infinidad de proyectos para crear con la gran cantidad de sensores y actuadores que existen. Consulta los Kits para primaria, secundaria y formación profesional disponibles en [Innova Didactic. Material Arduino](#)

