Retos

Imagina TDR STEAM y ArduinoBlocks.

Versión 1.0
(junio 2021)

INNOVA DIDACTIC

Esta guía ha sido elaborada por Fernando Hernández García, Ingeniero Técnico Industrial Especialidad Electrónica, formador del profesorado y profesor del Institut Torre del Palau (Terrassa – Barcelona).

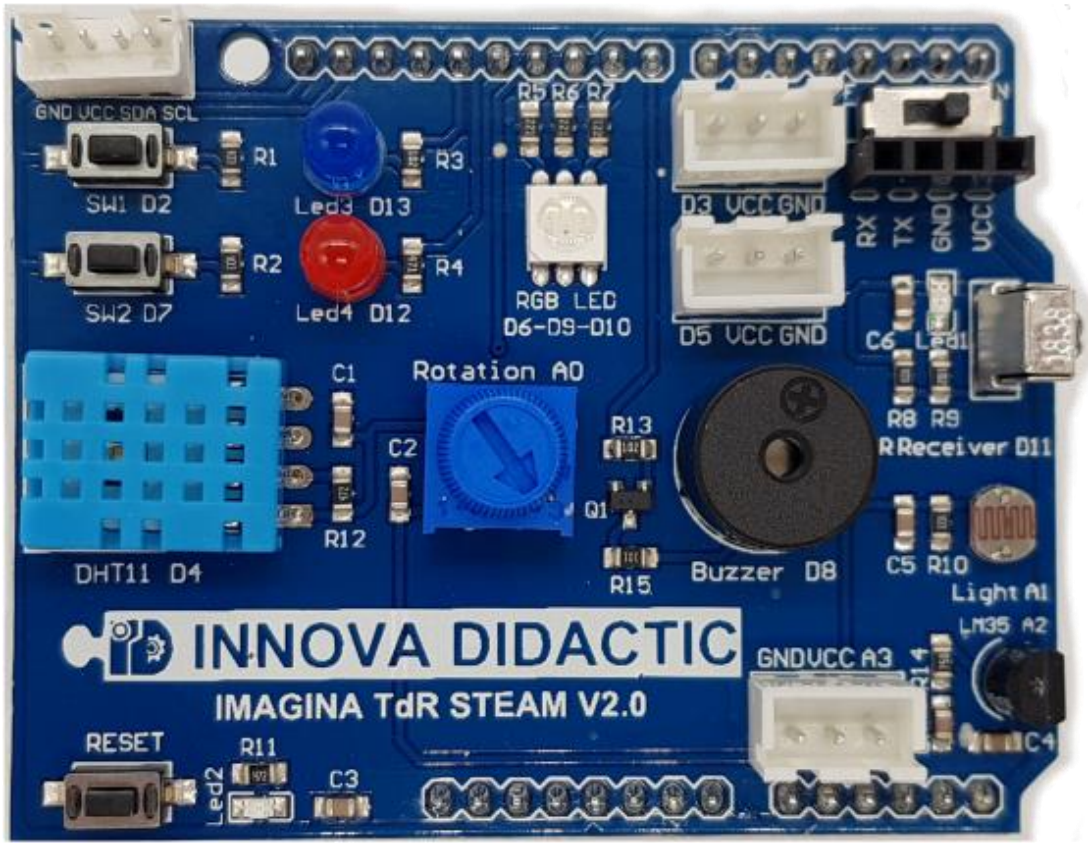
Se basa en documentaciones anteriores y ha sido desarrollado para el concurso RETOS CON IMAGINA TDR STEAM de Innova Didactic y Robolot.

Agradecer el soporte de ArduinoBlocks, Robolot Team e Innova Didàctic.

Índice

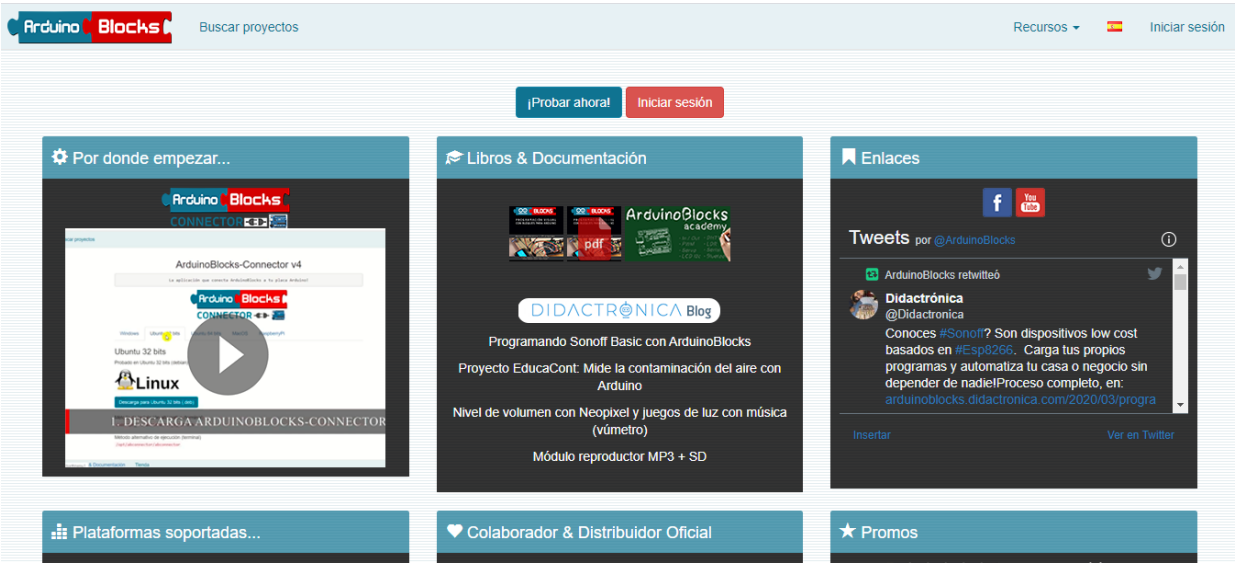
1 Introducción.

El presente manual pretende ser una herramienta base para iniciarse en el mundo de la programación, la electrónica y la robótica utilizando para ello la placa basada en Arduino: ***Imagina TDR STEAM*** (sobre una placa ***Keyestudio UNO***) y el entorno de programación ArduinBlocks. La placa Imagina TDR STEAM es una ***Shield*** (placa/escudo). Significa que es una placa que tiene que ir colocada sobre otra que contiene el sistema de control. En este caso, sobre una placa Keyestudio UNO. La placa Imagina TDR STEAM tiene numerosos sensores y actuadores que permitirán hacer infinidad de proyectos.



En este documento no pretende ser únicamente un manual para aprender programación. El manual presenta una serie de actividades guiadas y retos para aprender a programar de una manera entretenida y divertida mientras aprendemos conceptos relacionados con las S.T.E.A.M. (Science, Technology, Engineering, Arts, Math).

Para realizar la programación con la placa 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 que tiene este programa.



Con la placa Imagina TDR STEAM tenemos la mayoría de sensores y actuadores que necesitamos para poder introducirnos en el mundo de la programación de entornos físicos (Physical Computing). También tenemos conexiones de expansión para poner conectar más elementos externos.

Página del entorno de programación: ArduinoBlocks.

www.arduinoblocks.com

Página de información: Didactrónica.

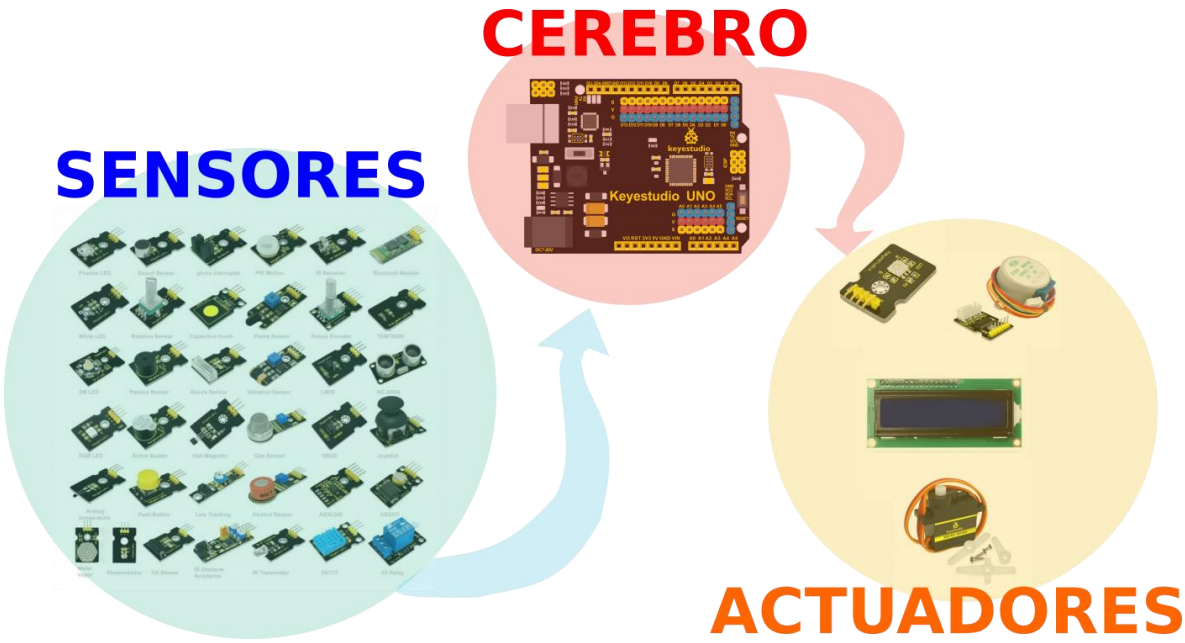
www.didactronica.com

Página para comprar material: Innova Didactic S.L.

shop.innovadidactic.com

2 Funcionamiento de un sistema de control programado.

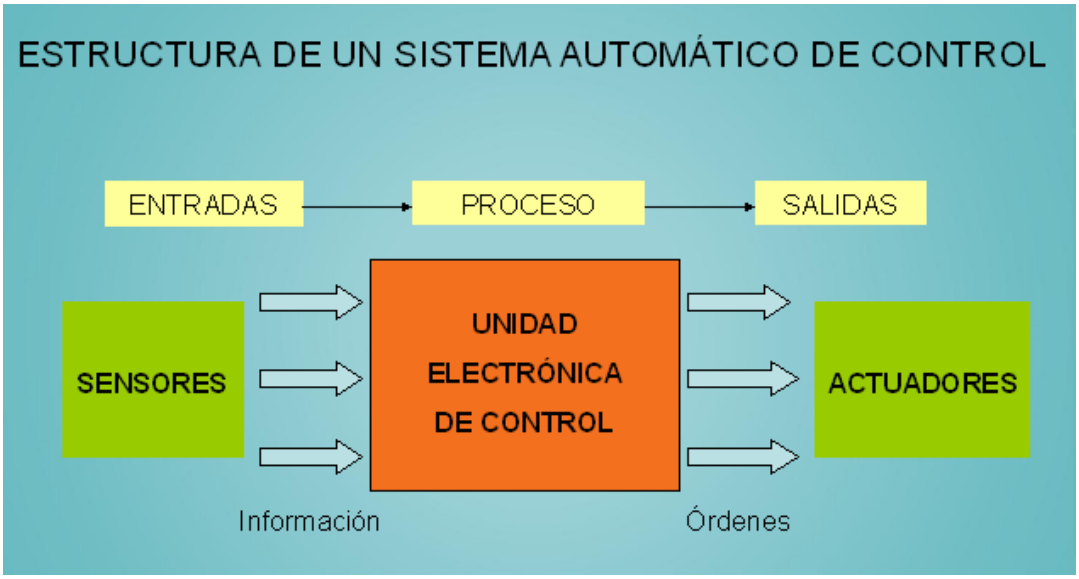
Un sistema de control programado funciona de manera similar a la de un ser humano. Cuando nuestro cerebro recibe información de los sentidos; oído, olfato, gusto, vista y tacto; analiza esta información, la procesa y da órdenes a nuestros músculos para realizar movimientos, dar estímulos a las cuerdas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



En el caso de un sistema de control programado, un microcontrolador hace la función de cerebro. Este componente electrónico 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, zumbadores, etc.

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, 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. Estas órdenes se generan mediante el código de programa que introducimos en nuestro sistema de control.

En todo sistema de control automático, los sensores (entradas) convierten las magnitudes físicas, químicas, etc. en eléctricas, creando así la información. Esta información se envía a la unidad de control (proceso) que procesa dicha información y genera las órdenes mediante señales eléctricas que serán convertidas en los actuadores (salidas) en otro tipo de magnitudes.



3 Componentes de la placa Imagina TDR STEAM.

La placa **Imagina TDR STEAM** es una placa didáctica desarrollada por el equipo ***ROBOLOT TEAM*** 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 esta placa (*shield*) a una placa Arduino UNO (en nuestro caso, una placa compatible llamada Keyestudio UNO) y ya está todo listo para empezar a programar.

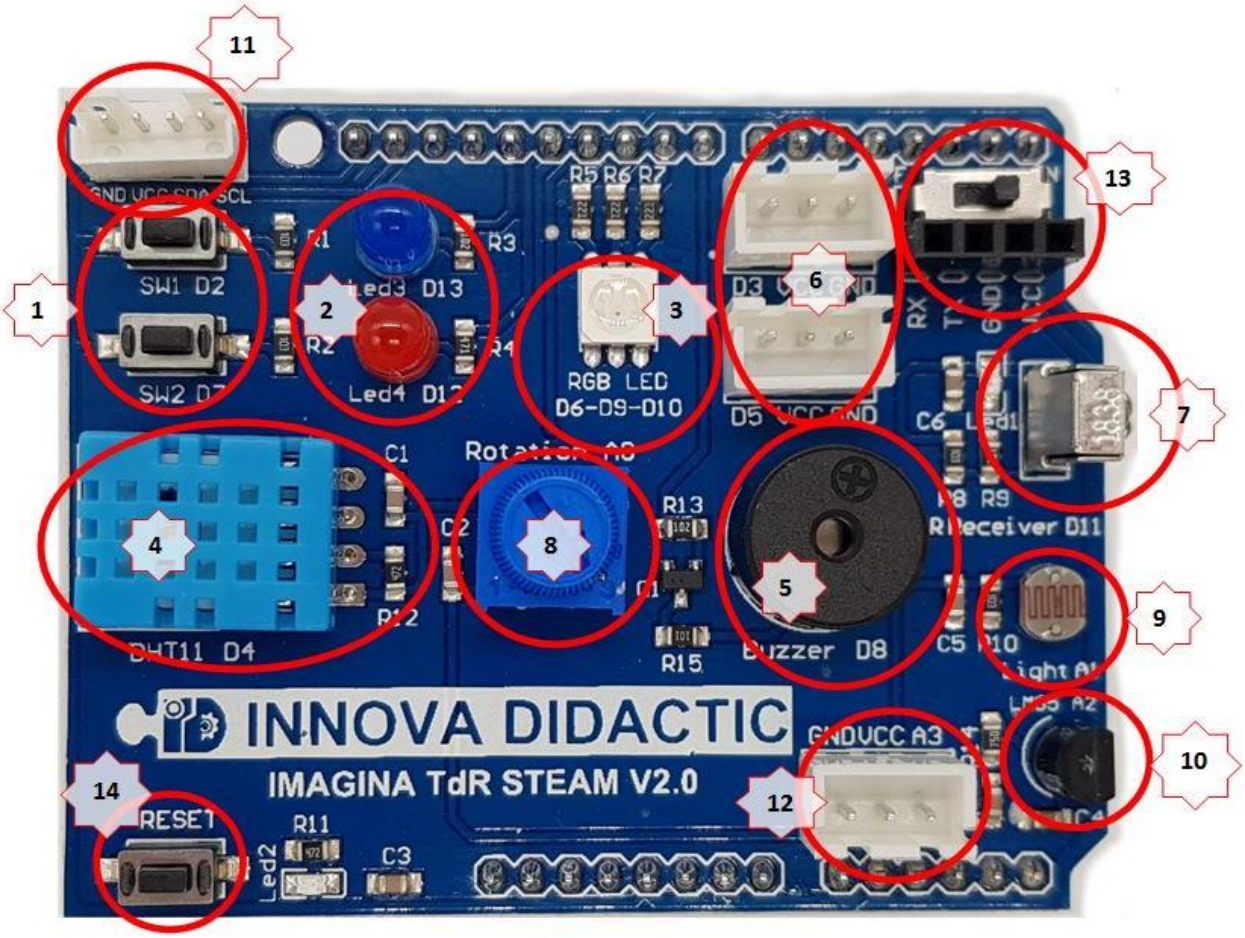
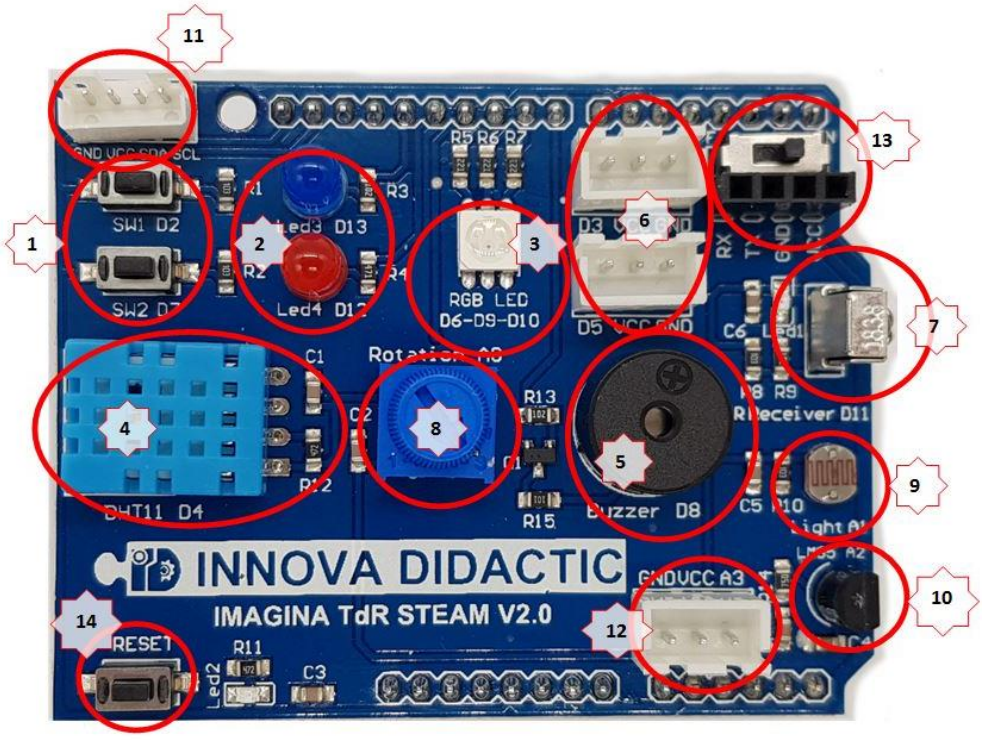


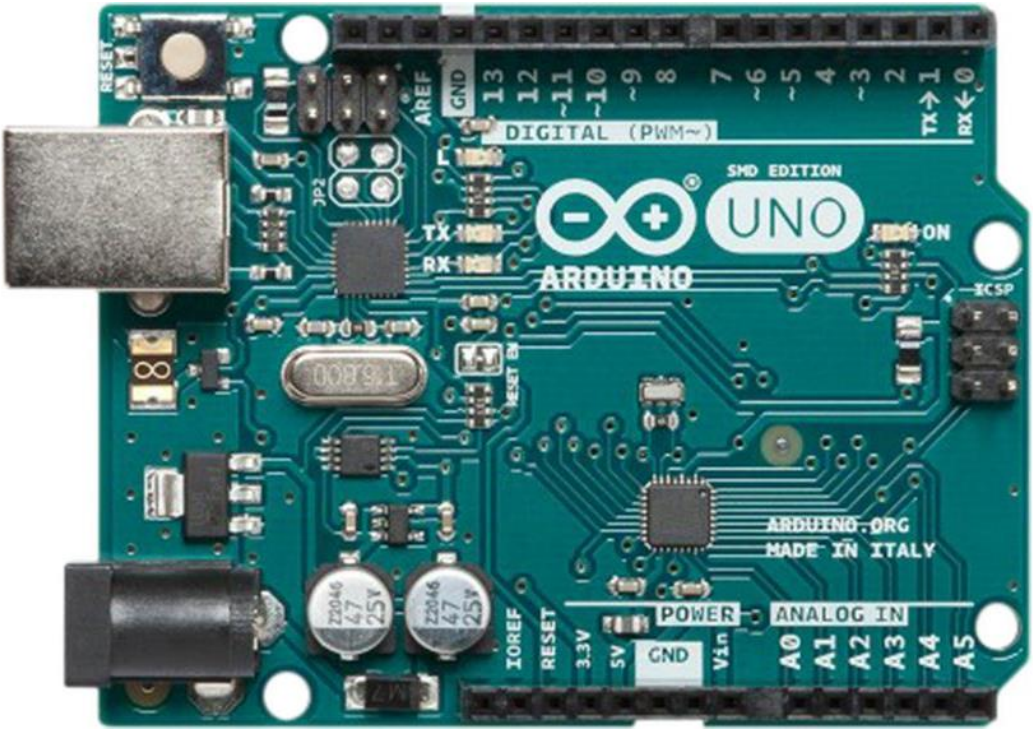
Tabla con la relación de elementos que hay en la placa Imagina TDR STEAM y sus conexiones:

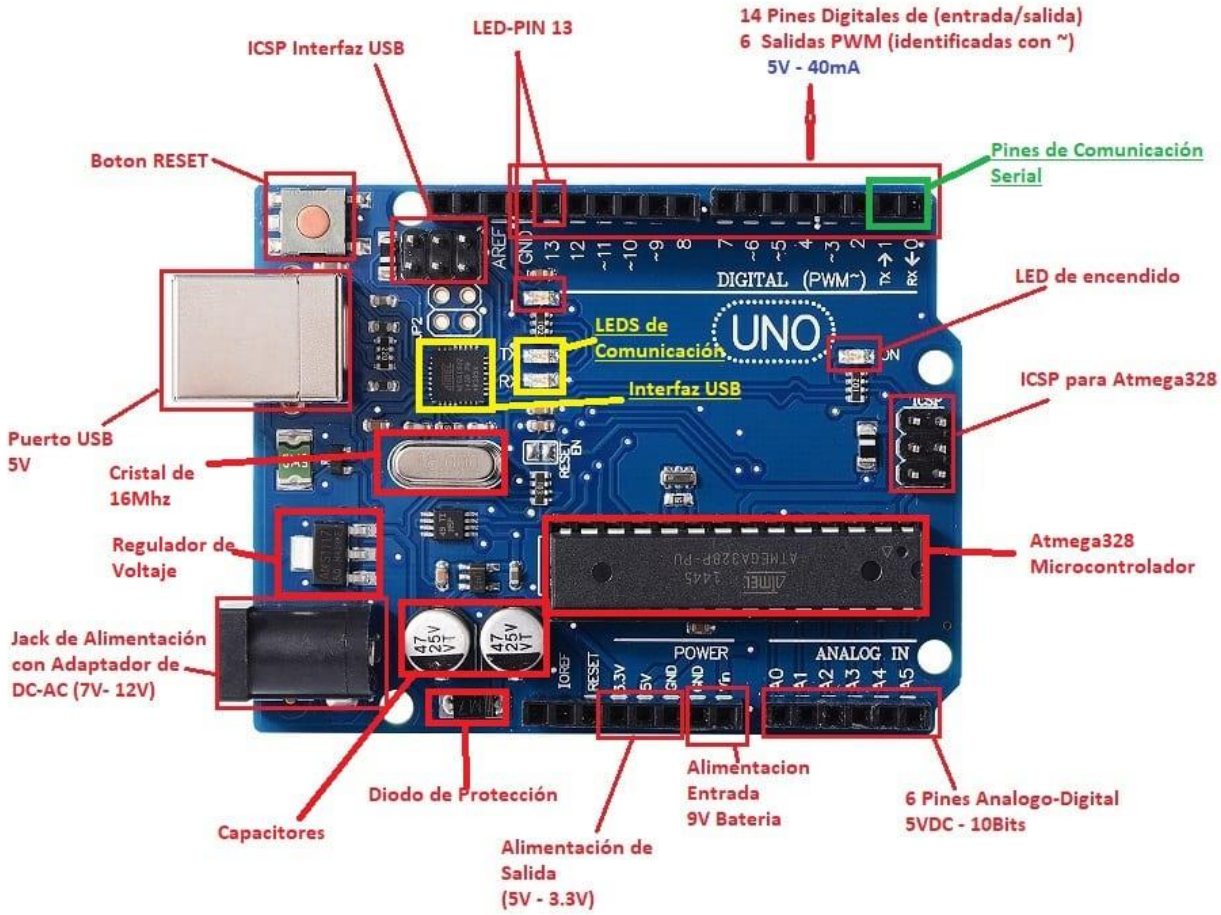
	Sensor/ Actuador/ Módulo	Pin de conexión
1	Dos pulsadores (SW1, SW2)	D2 y D7
2	Dos leds (Azul Led3 y Rojo Led4)	D13 y D12
3	Led RGB	D6-D9-D10
4	Módulo DHT11 Sensor de Temperatura y Humedad	D4
5	Zumbador o Piezoeléctrico	D8
6	Dos puertos (Entradas/Salidas) digitales	D3 y D5
7	Módulo receptor de infrarrojos (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 (Swich On/Off)	Rx - Tx
14	Botón de reinicio.	-



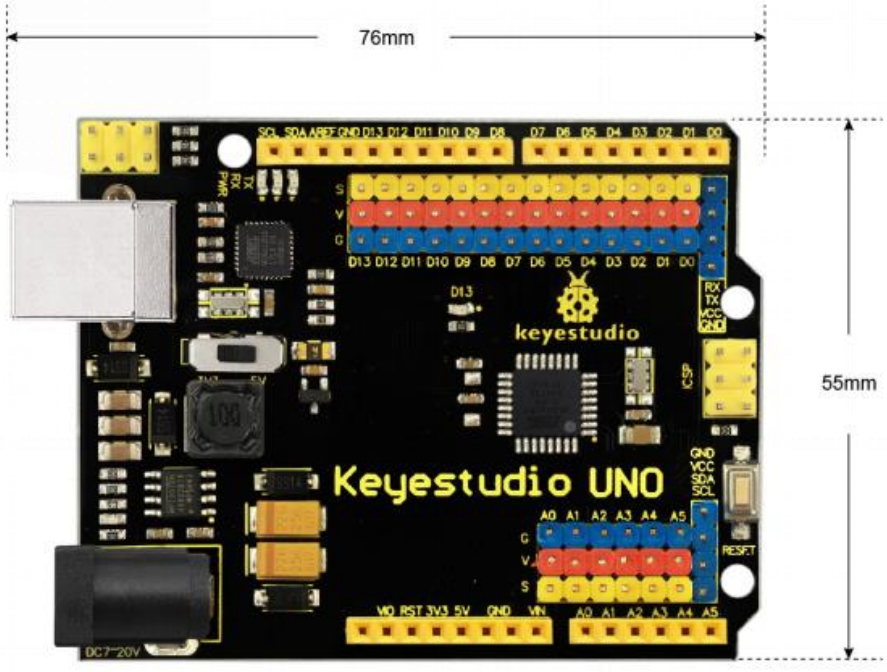
4 Placa de control Keyestudio UNO (compatible con Arduino UNO).

La placa de control que gestiona la placa Imagina TDR STEAM está basada en una placa ***Arduino UNO***. Arduino es una plataforma de prototipos electrónicos de código abierto (Open-Source) basada en hardware y software libre, flexible y fácil de usar, con una comunidad muy grande que genera muchísima información. Esta plataforma permite crear diferentes tipos de sistema para diferentes usos. Puede ser empleada por *artistas, diseñadores*, ingenieros, profesores, alumnos, etc., y en general, cualquier persona que esté interesada en crear objetos o entornos interactivos, prototipos, sistemas robóticos, etc.

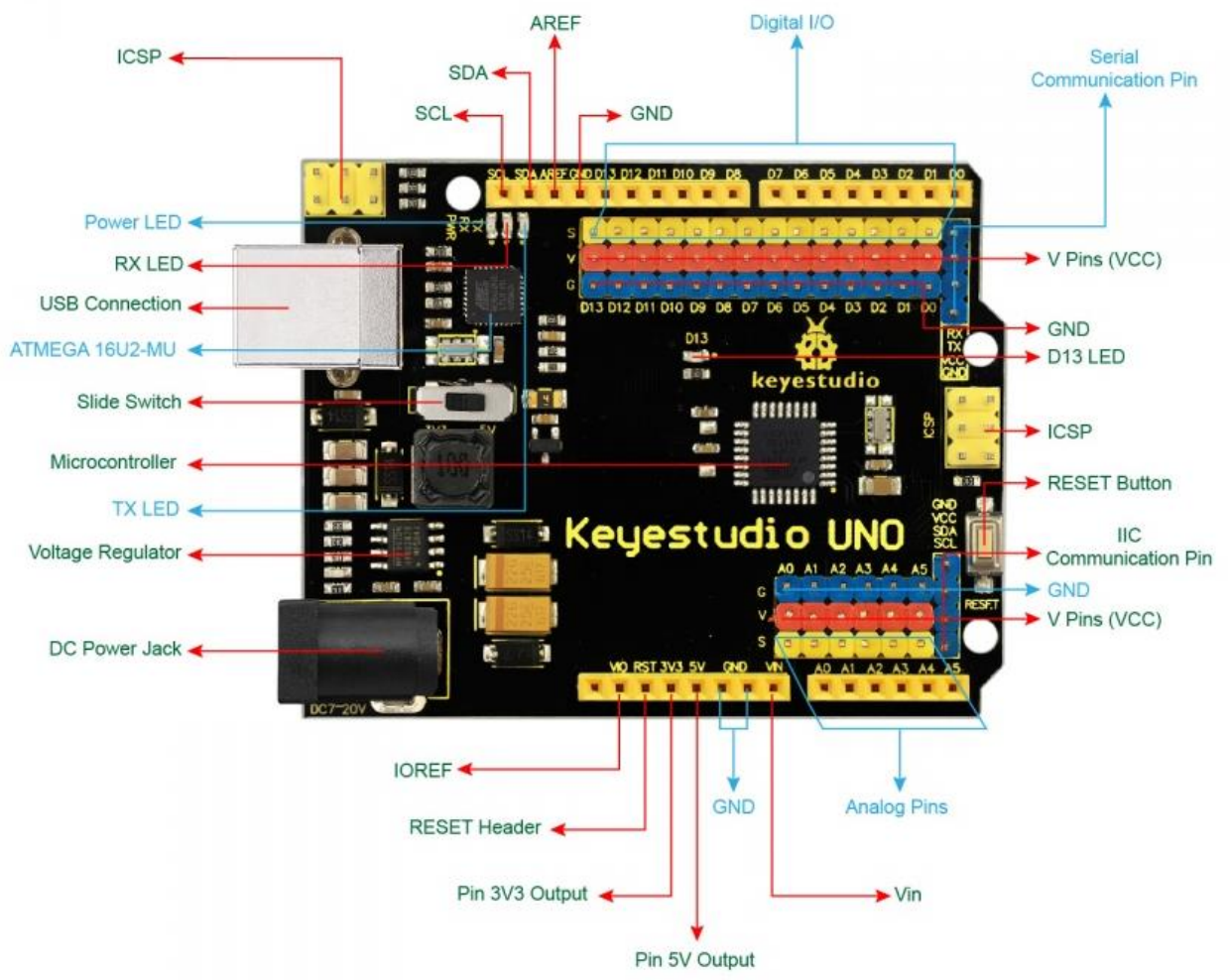




Al ser hardware libre existen multitud de fabricantes que han desarrollado versiones basadas en Arduino. Uno de esos fabricantes es ***Keyestudio***.



La placa **Keyestudio UNO** lleva un microcontrolador que está basado en el ATmega328. Tiene 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 conector ICSP y un botón de reinicio (reset).



Especificaciones:

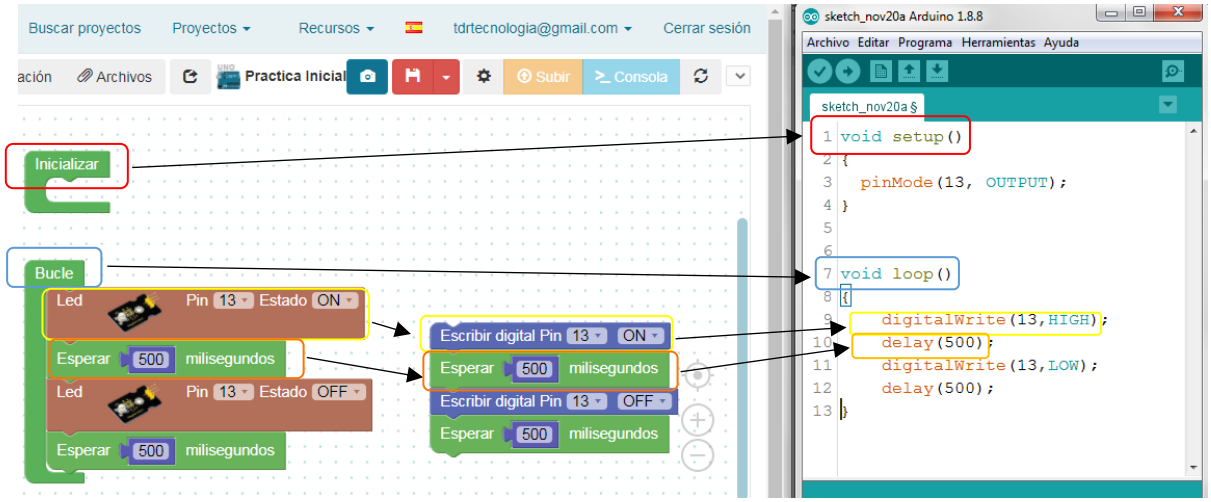
- Voltaje de funcionamiento: + 5V.
- Voltaje de entrada externo: + 7V ~ + 12V. (Límite: +6 V. <+ 20 V).
- Corriente de interfaz DCI / O: 20mA.
- Flash Memory: 32 KB (ATmega328) de los cuales 0,5 KB utilizados por el gestor de arranque.
- Capacidad de almacenamiento EEPROM: 1KB.

- 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.

Las conexiones de la placa TDR STEAM con la placa Keyestudio UNO son las siguientes:

Conexión	Descripción	Tipo
D0	Rx	Comunicación PC/Bluetooth/Wifi
D1	Tx	
D2	Pulsador SW1	Entrada digital
D3	Libre	E/S digital
D4	DHT11 (sensor de humedad y temperatura)	Entrada digital
D5	Libre	E/S digital
D6	RGB (rojo)	Salida digital
D7	Pulsador SW2	Entrada digital
D8	Zumbador	Salida digital
D9	RGB (verde)	Salida digital
D10	RGB (azul)	Salida digital
D11	Sensor infrarrojos	Entrada digital
D12	Led rojo	Salida digital
D13	Led azul	Salida digital
A0	Potenciómetro	Entrada analógica
A1	LDR (sensor de luz)	Entrada analógica
A2	LM35 (sensor de temperatura)	Entrada analógica
A3	Libre	Entrada analógica
A4	SDA	I2C
A5	SCL	

Para realizar la programación la podemos hacer mediante la IDE de Arduino o mediante ArduinoBlocks. Como podemos ver son dos sistemas diferentes. En la IDE de Arduino la programación se realiza mediante instrucciones (derecha de la imagen) y en ArduinoBlocks se realiza mediante bloques (izquierda de la imagen). En la siguiente imagen se hace una comparación de código entre ArduinoBlocks y Arduino IDE.

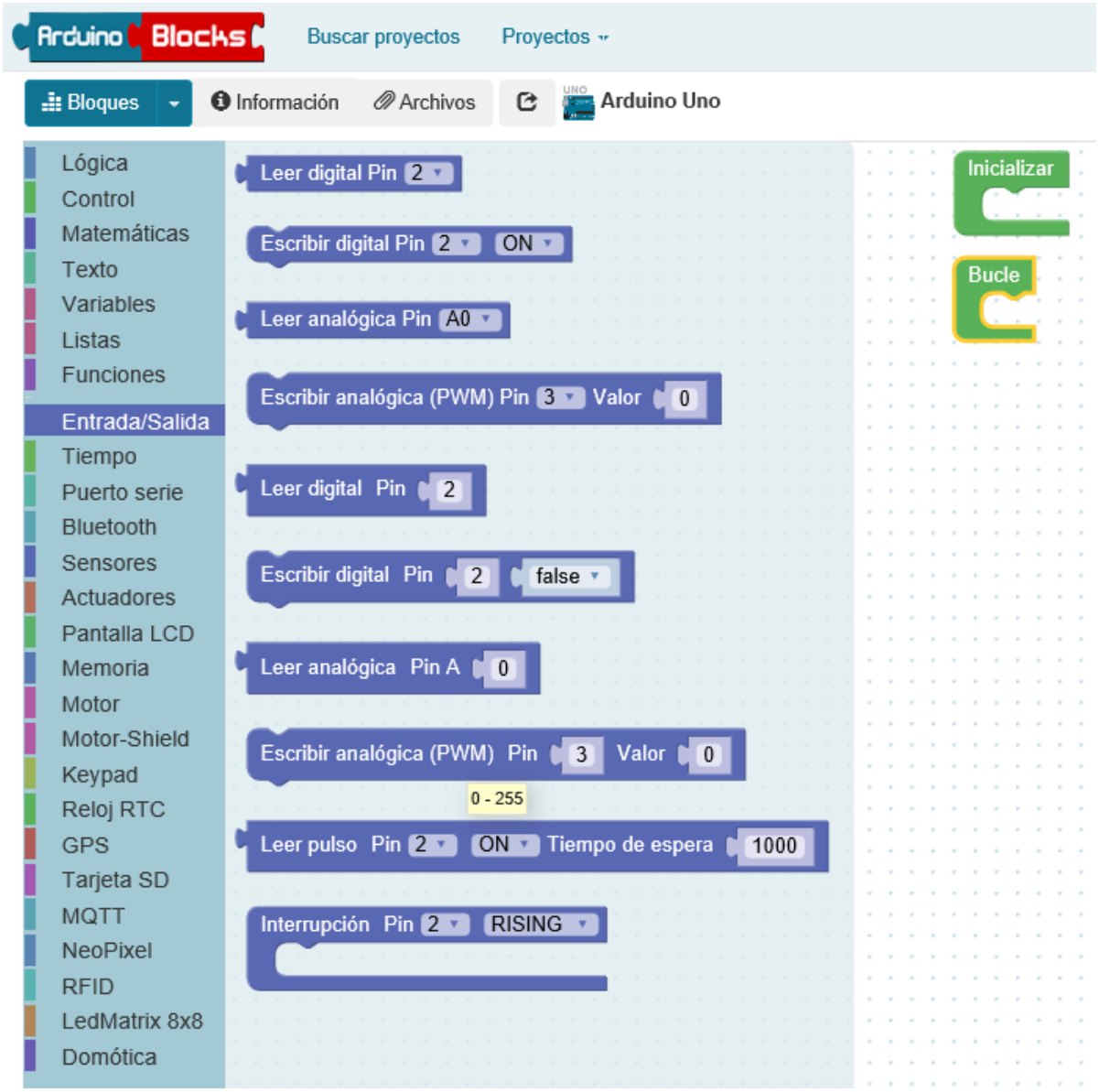


Utilizar ArduinoBlocks simplifica y hace más inteligible el código, lo que permite iniciarse en el mundo de la programación de un modo más amigable. ArduinoBlocks también permite programar de diversas formas, ya que tiene bloques específicos que realizan las mismas funciones pero que se pueden entender de forma más sencilla.

5 Programación con ArduinoBlocks.

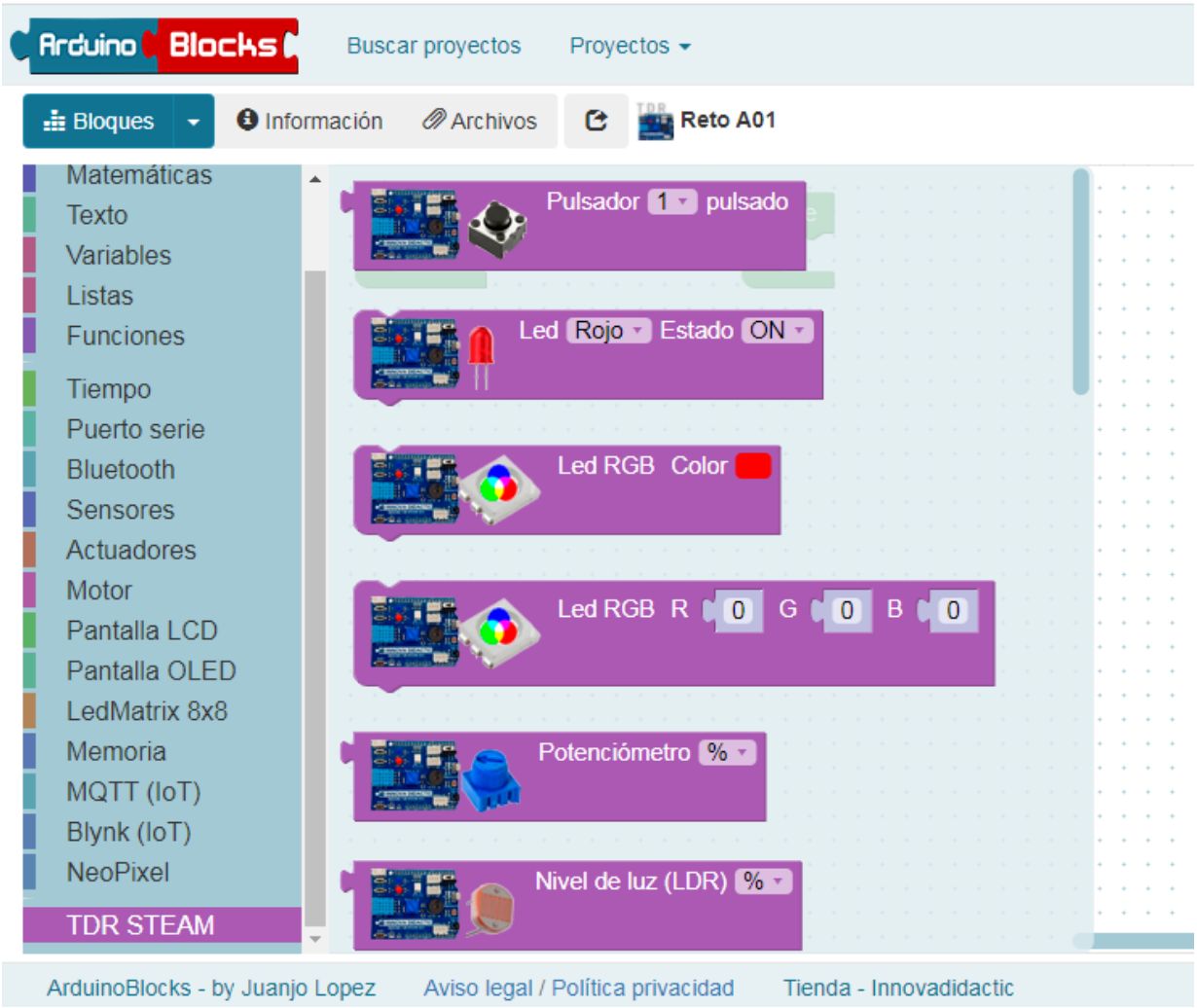
ArduinoBlocks es un lenguaje de programación gráfico por “Bloques” creado por el profesor Juanjo López. Está pensado para que niñas y niños aprendan a programar con placas Arduino a partir de unos 8-9 años.

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



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

Podemos programar ArduinoBlocks de diferentes maneras ya que tiene múltiples bloques. También permite exportar el código para la IDE de Arduino. Pero para este manual utilizaremos, principalmente, la programación que se muestra en medio (bloques específicos para Imagina TDR STEAM), ya que es más fácil de entender.

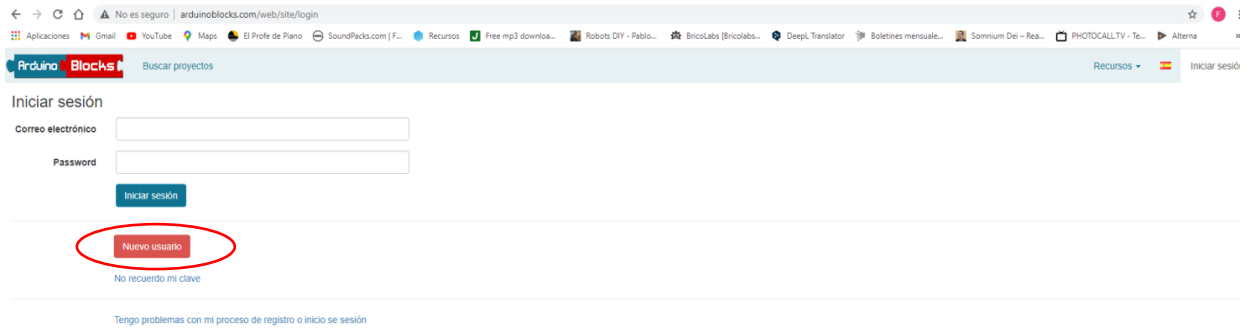


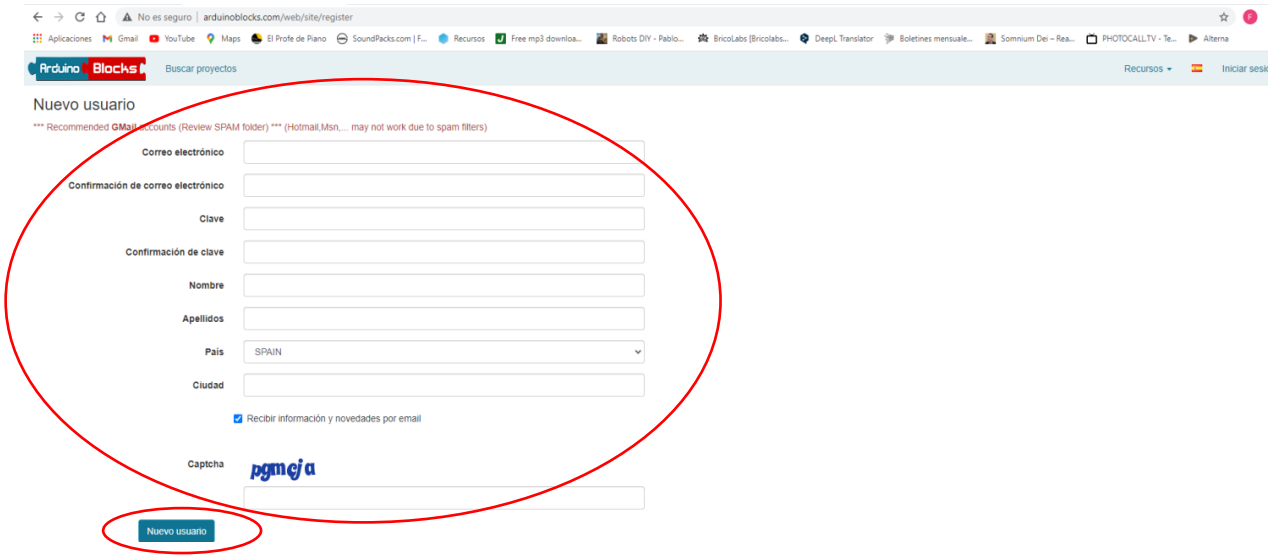
6 Instalación de ArduinoBlocks.

ArduinoBlocks trabaja on-line pero tenemos que instalar un pequeño programa que será el encargado de conectar nuestro programa con la placa Keyestudio UNO. Este programa basado en Python se llama *Connector*.

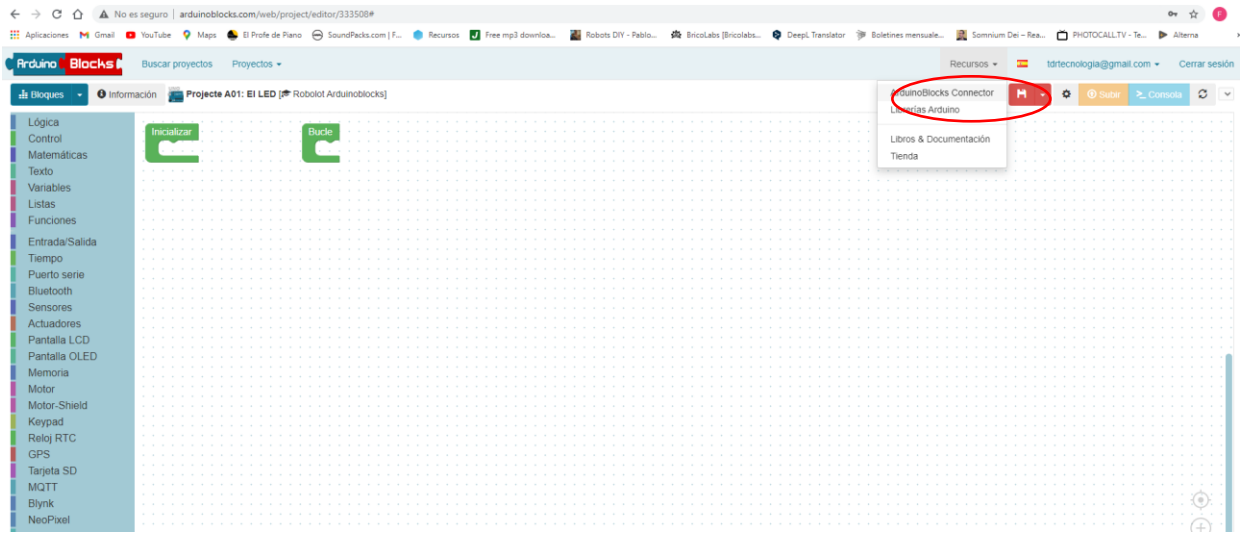
Primero deberemos crear una cuenta en ArduinoBlocks y después instalar el software ***Connector***.

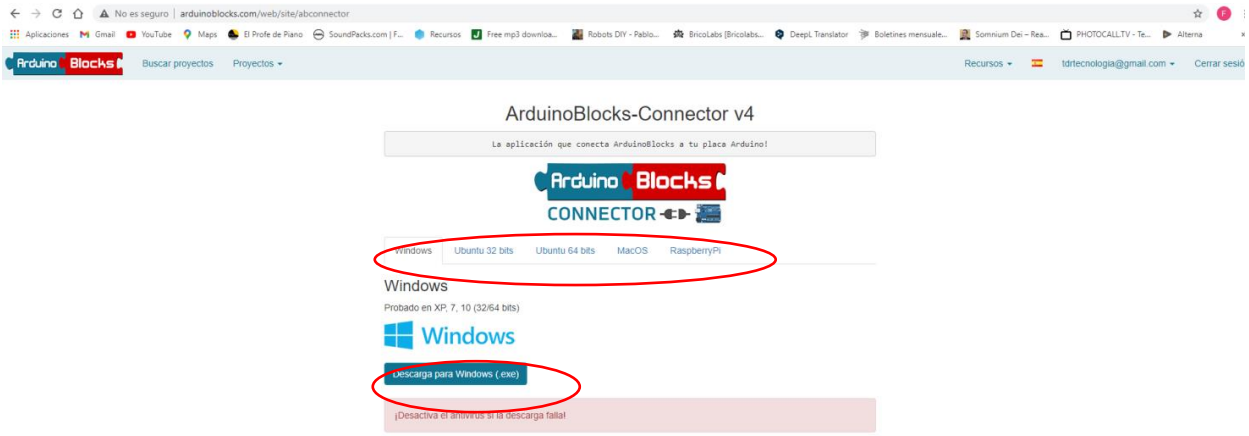
- Para crear una ***cuenta*** en ***ArduinoBlocks***:

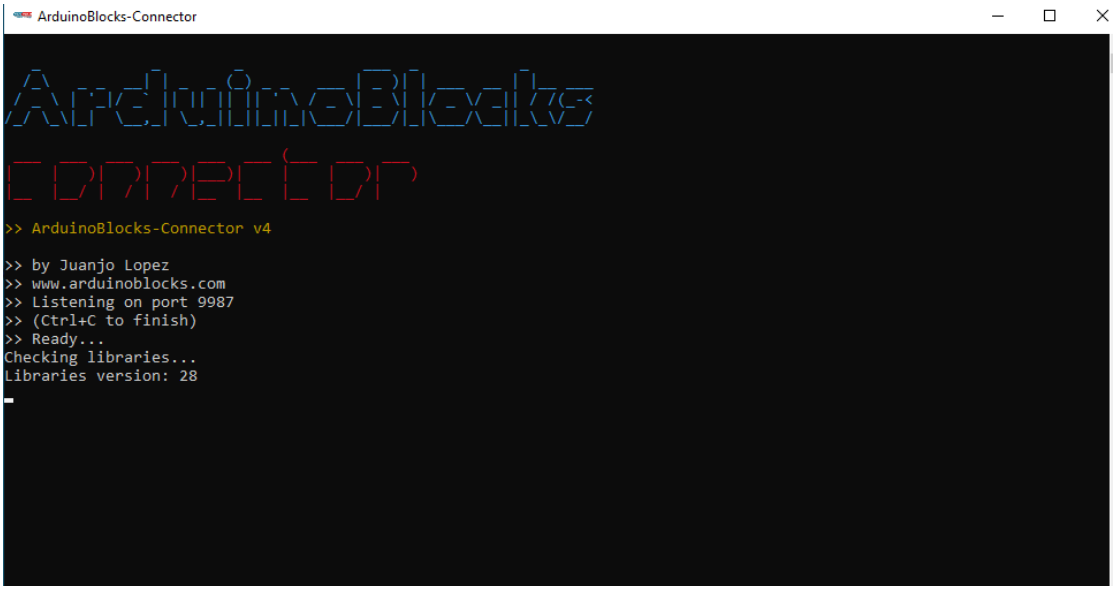




- Para instalar ***Connector***:







7 Retos con la placa Imagina TDR STEAM.

A continuación, os proponemos una serie de actividades y retos para aprender a programar la placa ***Kesyestudio UNO R.3*** y la placa ***Imagina TDR STEAM*** con ArduinoBlocks paso a paso para que realicen infinidad de tareas.

- Reto A01. El led.
- Reto A02. El led RGB.
- Reto A03. El zumbador.
- Reto A04. El pulsador.
- Reto A05. El potenciómetro.
- Reto A06. La fotocélula (LDR – sensor de luz).
- Reto A07. Sensor de temperatura LM35D.
- Reto A08. Sensor de temperatura y humedad DHT-11.
- Reto A09. Sensor de infrarrojos (IR).
- Reto A10. Puertos de expansión I2C.
    - LCD.
    - OLED.
- Reto A11. Serial Plotter.
- Reto A12. Comunicaciones.
    - Bluetooth.
    - Wifi.

En las actividades y retos, primero hay una explicación del componente electrónico que se va a utilizar y, a continuación, una descripción de la práctica a realizar (puede ser más de una práctica). Por último, se propone una actividad de ampliación.

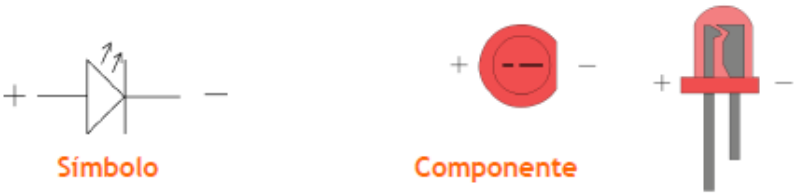
7.1 Reto A01. El led.

Reto A01. El led.

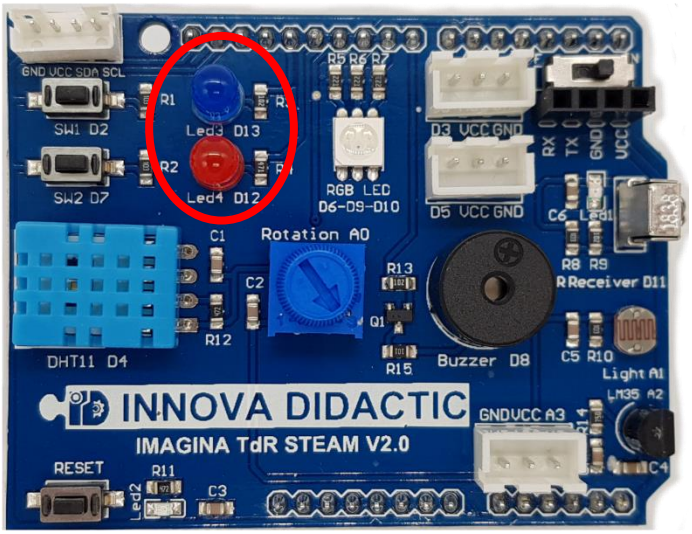
Vamos a empezar con nuestro primer reto. Vamos a realizar un programa que va a encender y apagar el led rojo correspondiente al pin D12.

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.



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

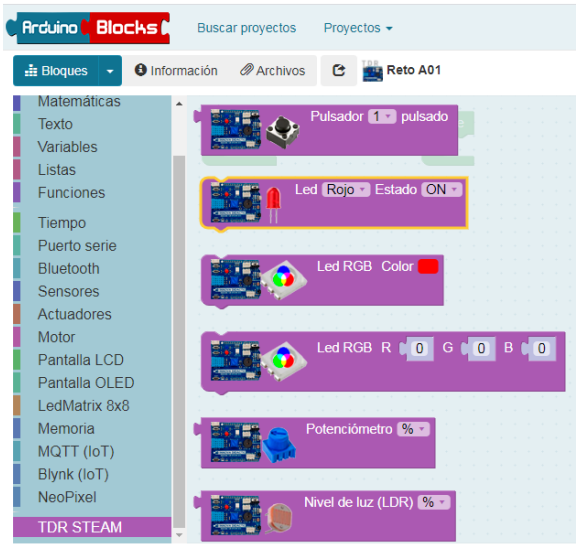


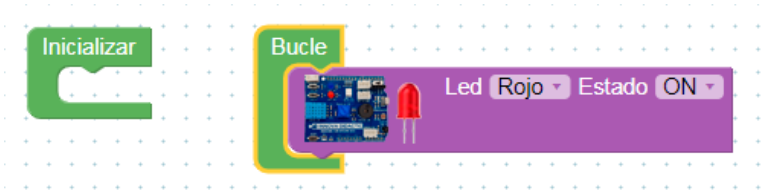
7.1.1 Reto A01.1. ON/OFF led rojo.

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.

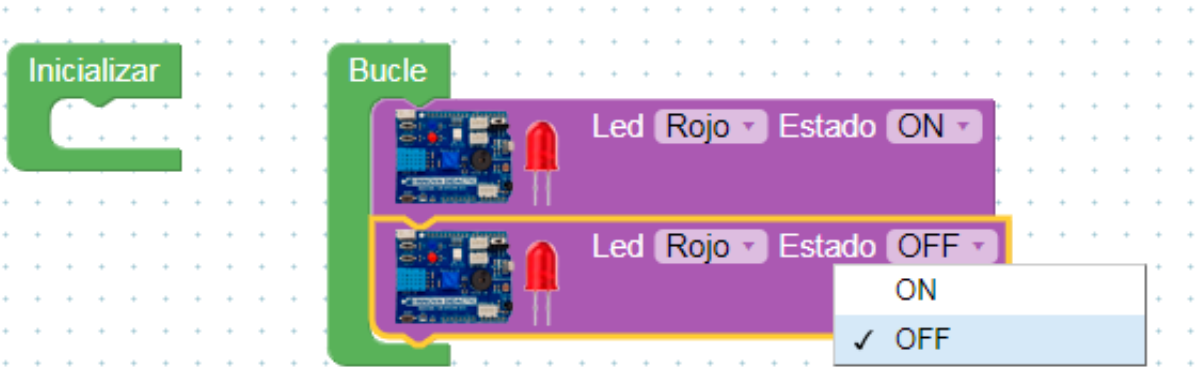
En el área de programación siempre hay dos bloques verdes (Inicializar y Bucle). Estos bloques siempre aparecen al iniciar un nuevo programa. Pues bien, todo lo que se meta dentro del bloque de *Inicializar* sólo se ejecutará la primera vez que se inicie el programa, mientras que si se colocan dentro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



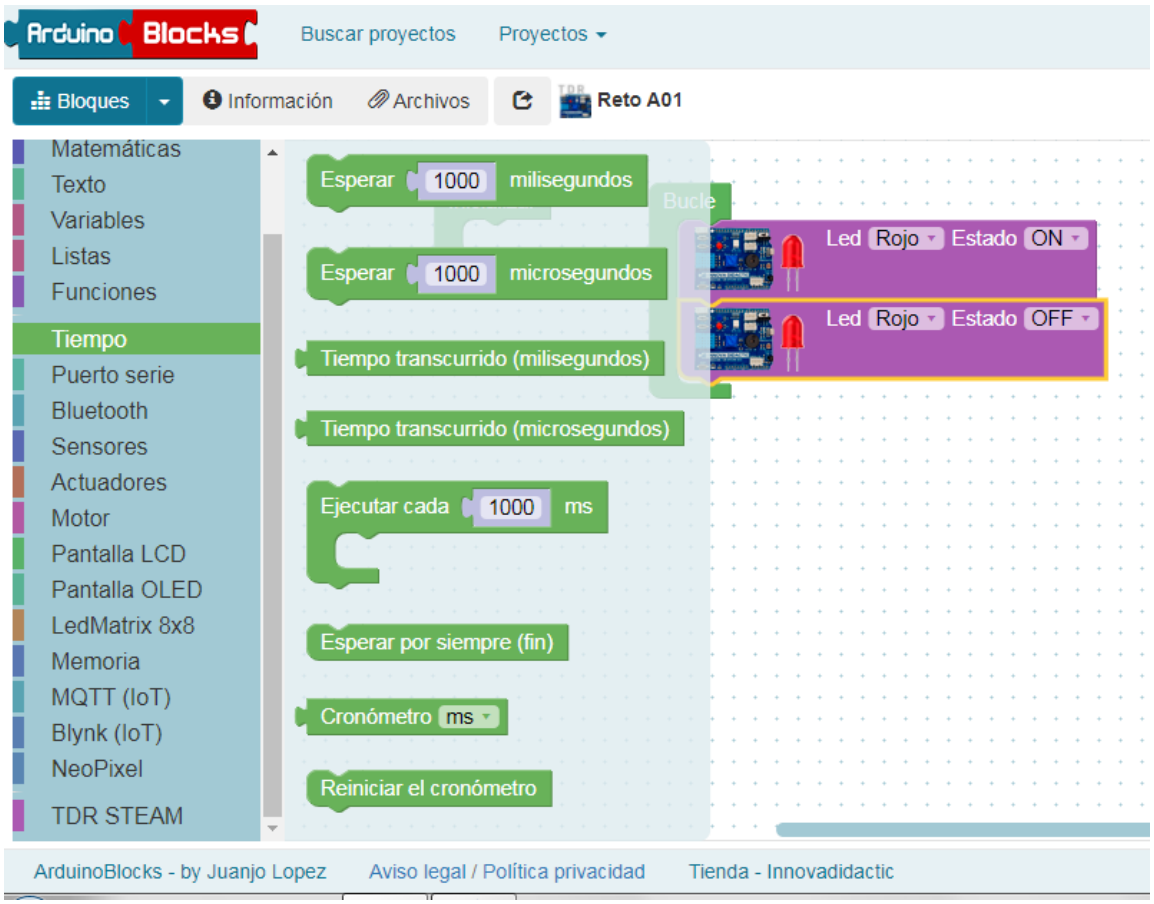


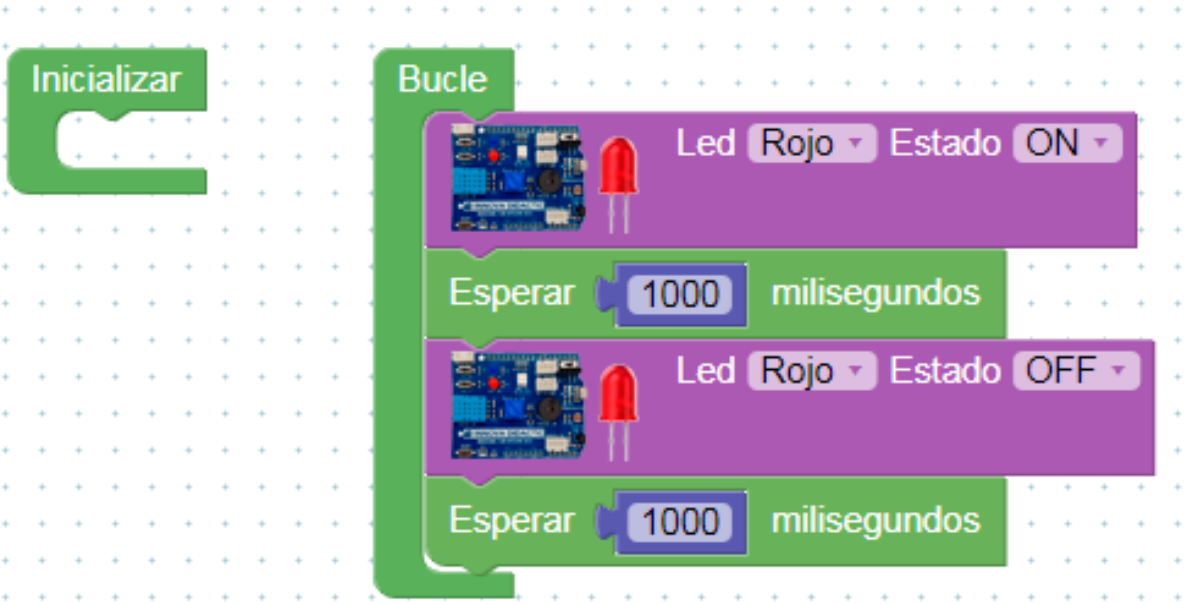
Seleccionaremos el bloque de *TDR STEAM*. Vamos a meter nuestro bloque de led en el Bucle y elegimos el color del led (Rojo o Azul). El led puede tener dos estados: ON/OFF (encendido/apagado), que podemos cambiar en el menú despegable.

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.



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).





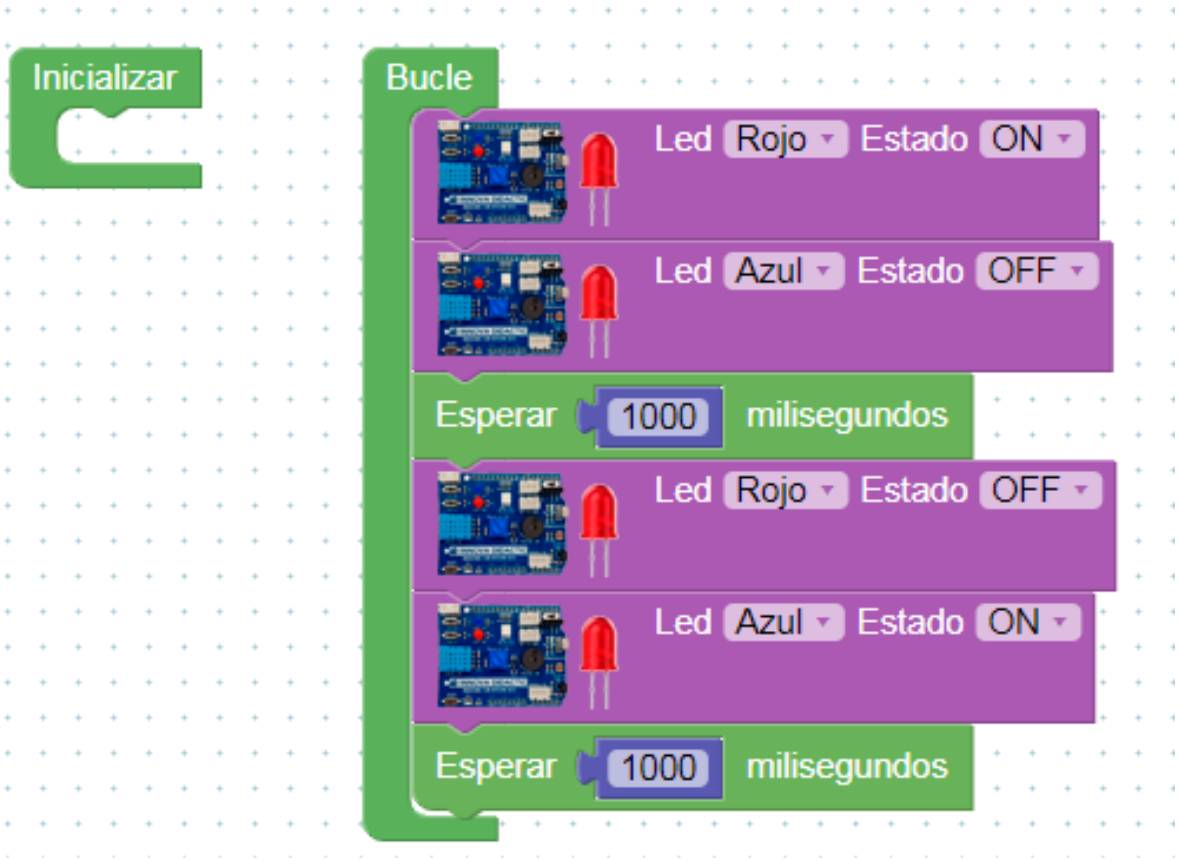
Ahora tenemos el led encendido durante 1 segundo y apagado otro. Esto ser repetirá por tiempo infinito hasta que quitemos la alimentación a la placa. El programa se quedará guardado en la memoria del microcontrolador, así que, si lo alimentamos con una fuente de alimentación externa, seguirá funcionando.

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.

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.

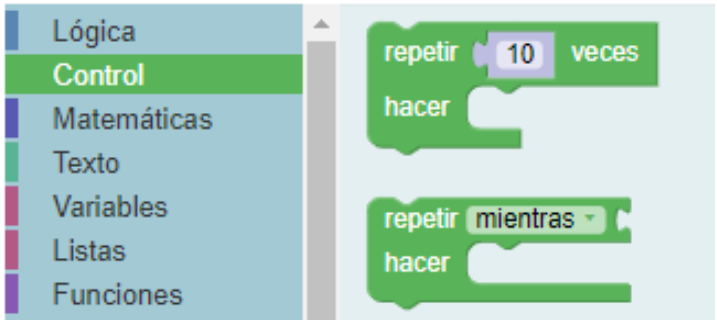


Actividad de ampliación: prueba ahora de hacer que los leds rojo y azul se enciendan a la vez y con los siguientes tiempos: 500ms ON y 250ms OFF.

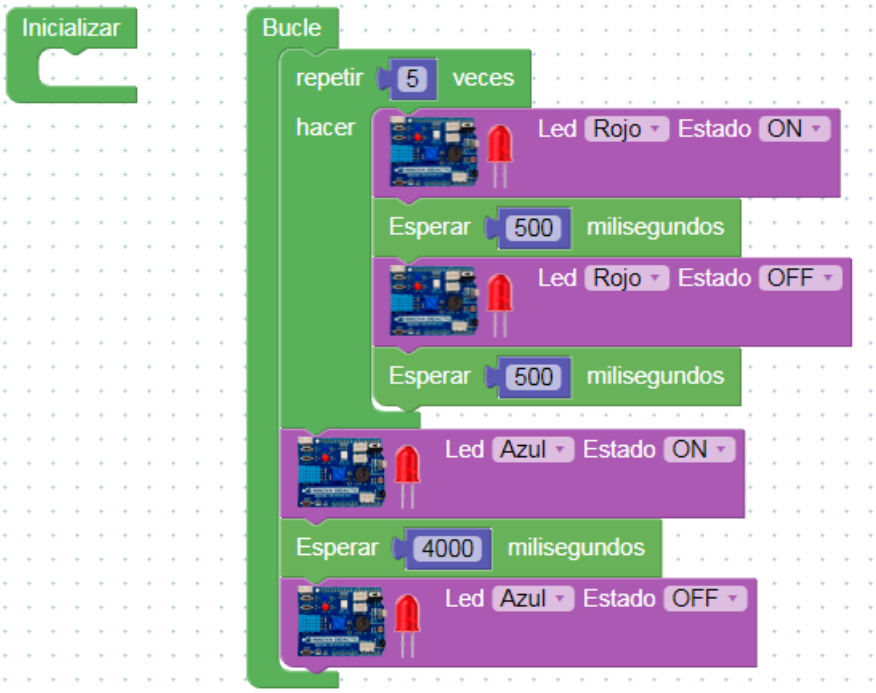
7.1.3 Reto A01.3. ON/OFF led rojo y azul con repeticiones.

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…*



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).



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.

7.2 Reto A02. El led RGB.

Reto A02. El led RGB.

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).

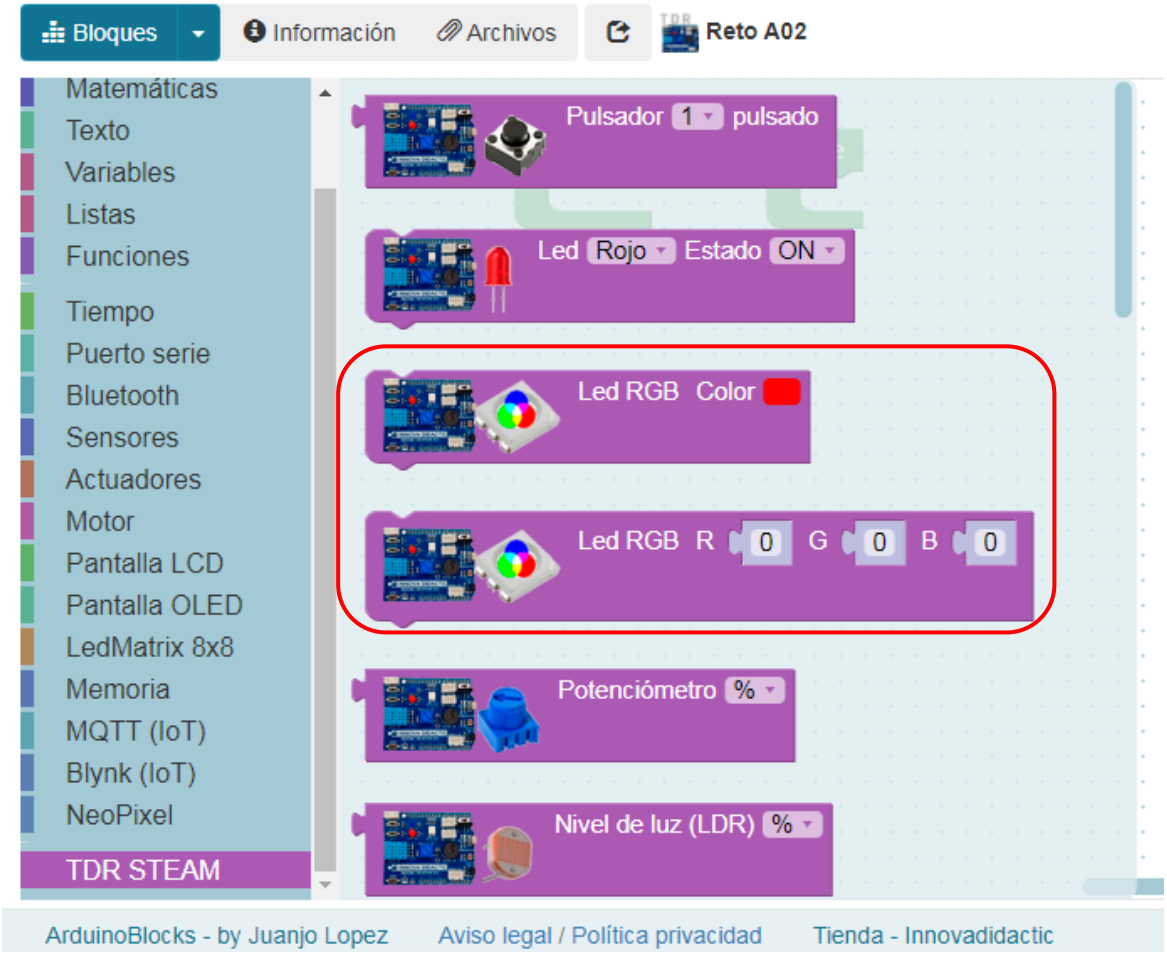
Símbolo

Componente

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.

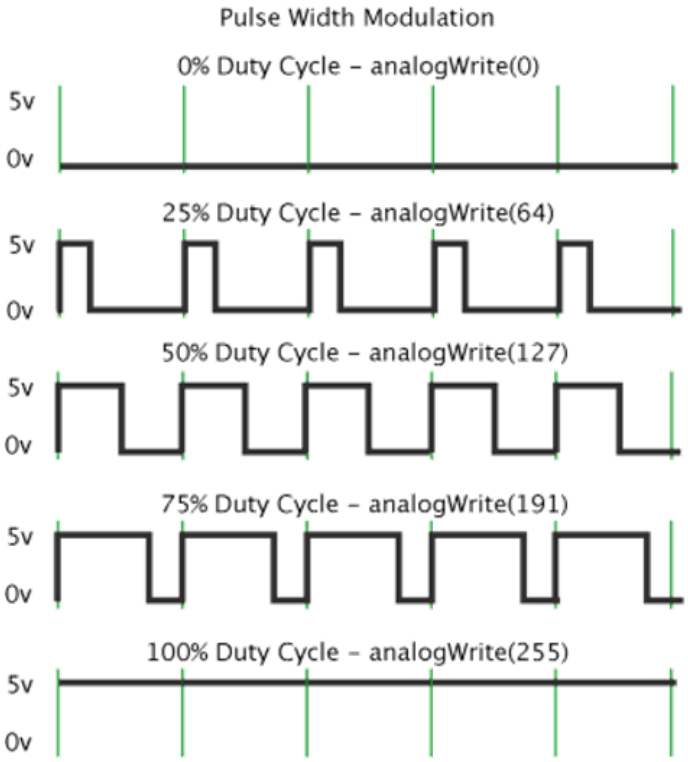
La placa **Imagina TDR STEAM** dispone de un led RGB conectado a los pines (D6-Red, D9-Green y D10-Blue). Estos tres pines son PWM para permitir regular su intensidad.

La modulación PWM permite generar una señal analógica mediante una salida digital. Utiliza un sistema de codificación de 8 bits (en el sistema binario: $2^8 = 256$, per tanto, del 0 al 255).



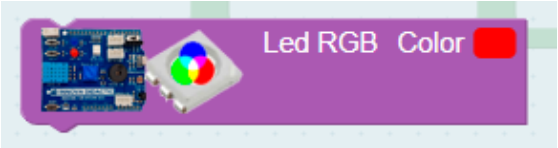
Continuando con la práctica anterior, ahora vamos a controlar la intensidad de un led utilizando la modulación PWM. Pero antes vamos a explicar cómo funciona la modulación PWM. PWM es la abreviatura de ***Pulse-Width Modulation*** (modulación de ancho de pulso). Las salidas digitales de Arduino sólo 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 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.

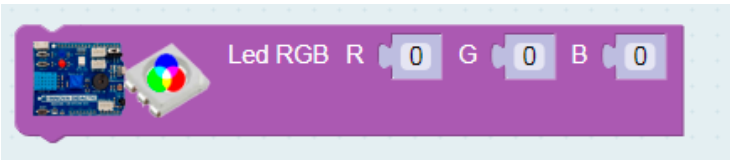


La placa Keyestudio UNO tiene un total de 6 salidas PWM, que son digitales 3, 5, 6, 9, 10 y 11, pero en la placa **Imagina TDR STEAM** sólo se puede controlar por PWM el led RGB (pines 6, 9 y 10). Tenemos dos bloques diferentes para regular el color:

- Asignaremos directamente el color en la paleta de colores:



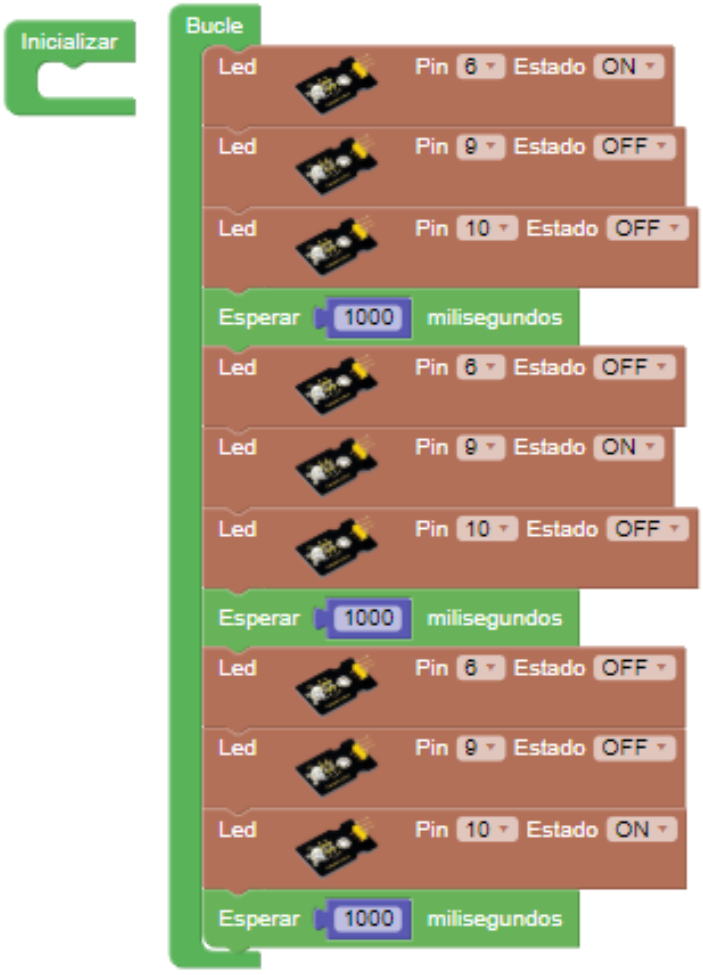
- Introduciremos la cantidad de cada uno de los tres colores primarios (R rojo, G verde, B azul) con un valor comprendido entre 0 i 255.



7.2.1 Reto A02.1. Identificación de colores RGB.

Reto A02.1. Identificación de colores RBG.

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.

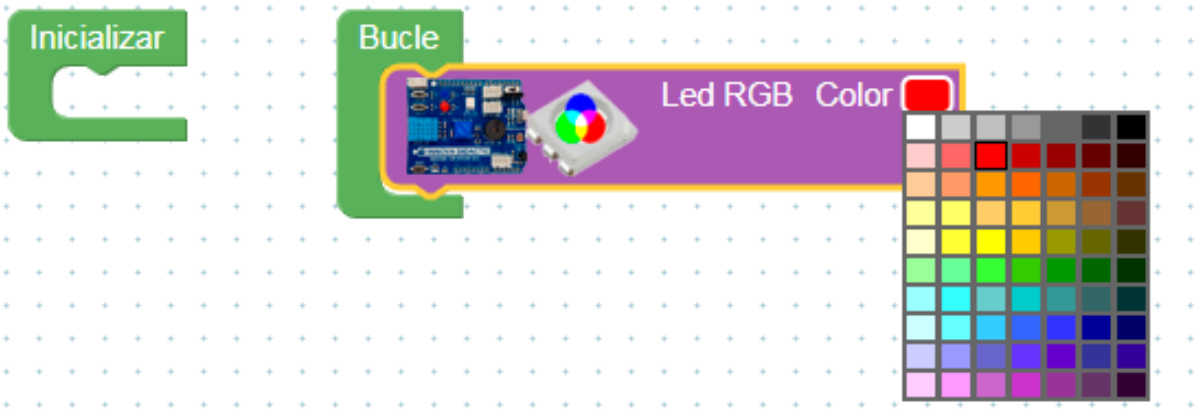
Actividad de ampliación: prueba ahora de ir activando varios leds a la vez para ver qué color aparece (puedes guiarte con la imagen siguiente).

7.2.2 Reto A02.2. Múltiples colores con el led RGB.

Reto A02.2. Múltiples colores con el led RBG.

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

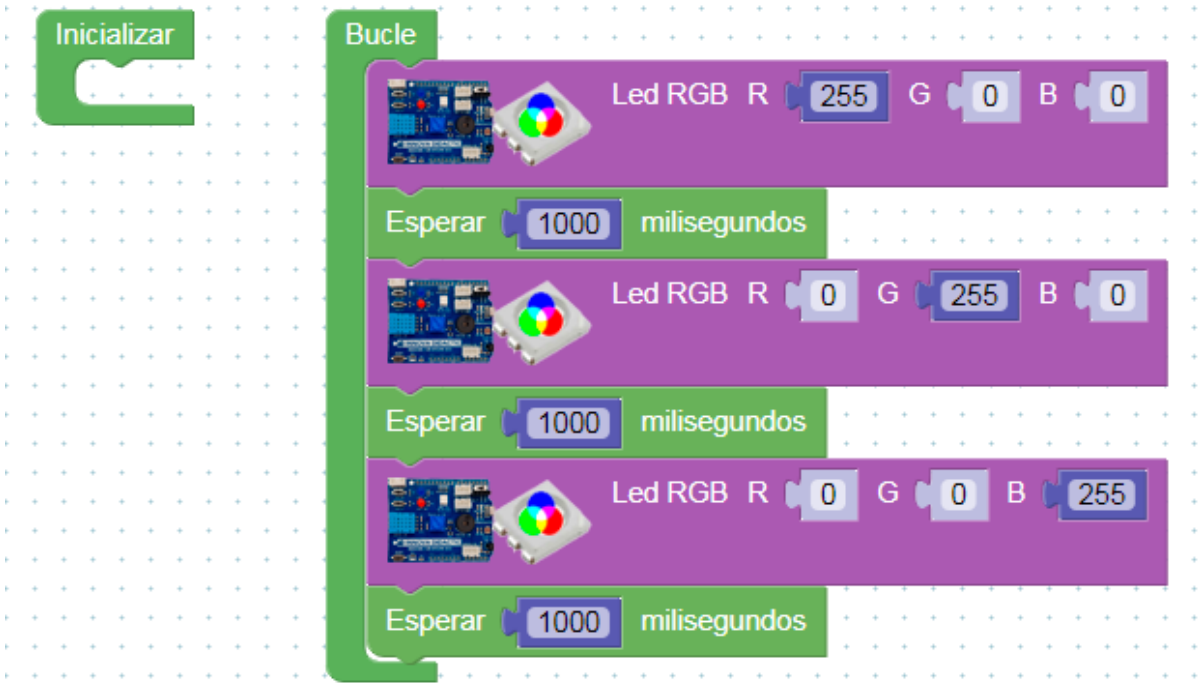
Comprueba como al pulsar sobre el icono del color se despliega una paleta de colores para que puedas elegir cualquier color.



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

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

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

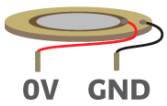


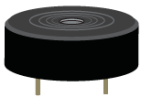
Actividad de ampliación: prueba ahora de hacer un programa que muestre los colores del arcoíris en orden.

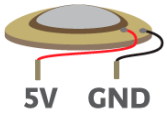
7.3 Reto A03. El zumbador.

Reto A03. El zumbador.

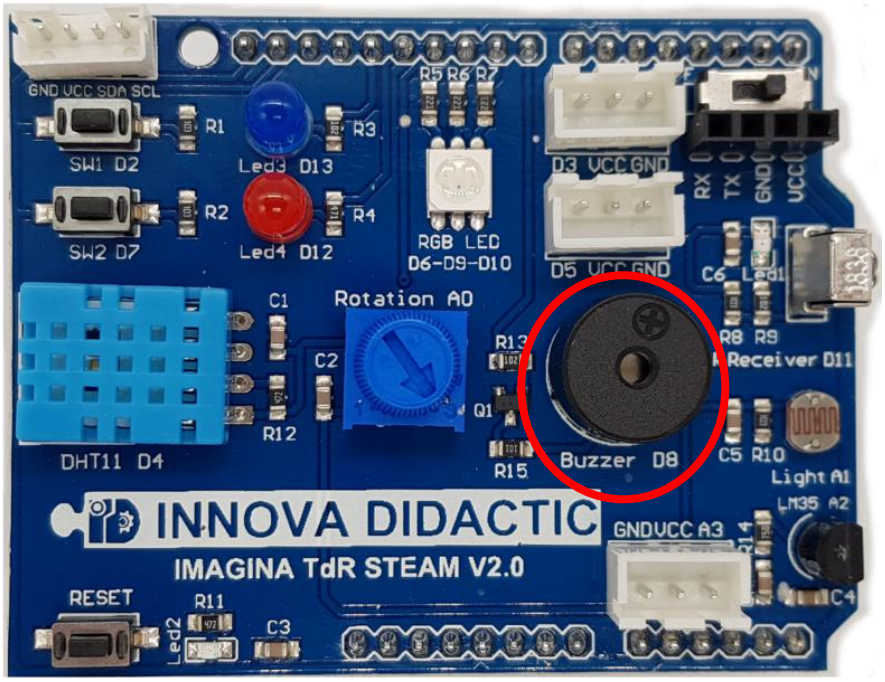
El zumbador (Buzzer en inglés) es un transductor electroacústico que produce un sonido o zumbido continuo o intermitente. En función de si se trata de un zumbador *Activo* o *Pasivo*, este zumbido será del mismo tono o lo podremos variar. Sirve como mecanismo de señalización o aviso y se utiliza en múltiples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





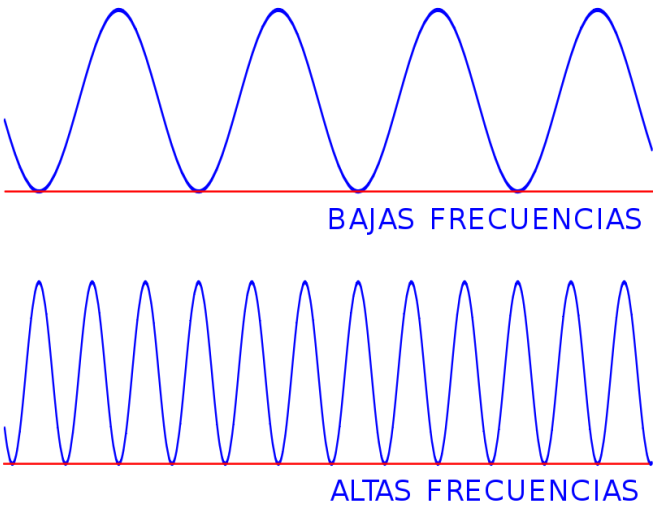


La placa ***Imagina TDR STEAM*** tiene un zumbador pasivo que está conectado en el pin D8.



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

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.



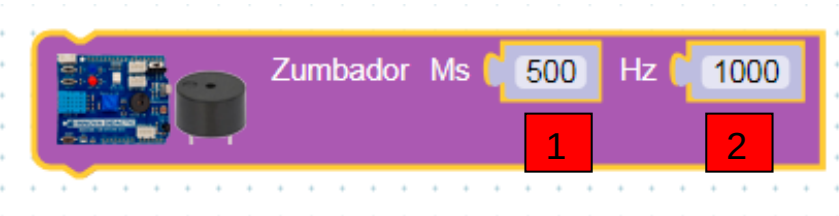
En la siguiente tabla están las frecuencias del sonido de las notas musicales:

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

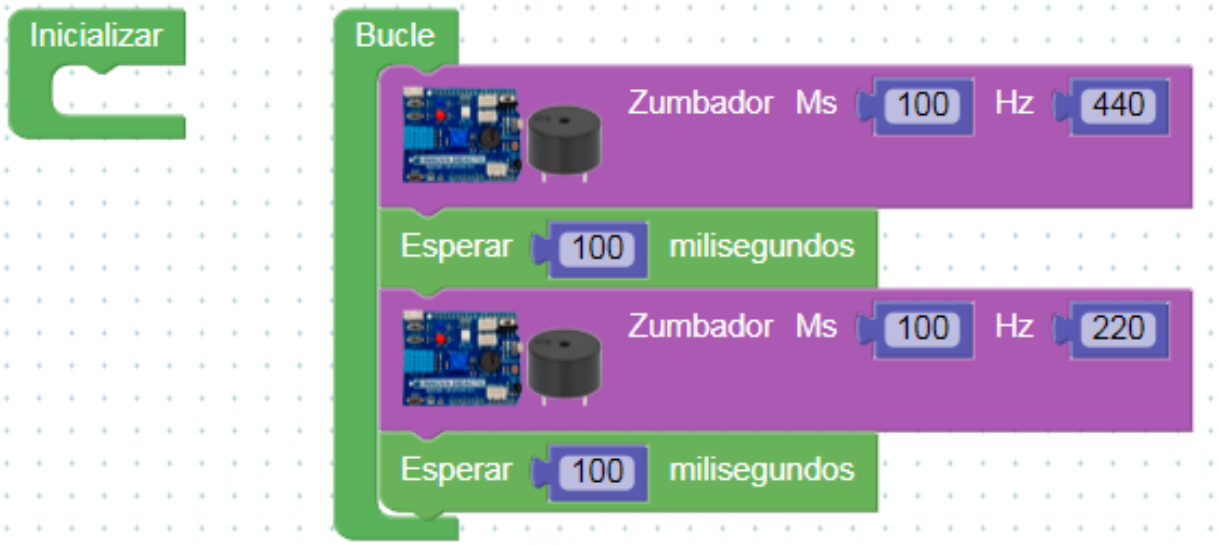
7.3.1 Reto A03.1. Primeros sonidos con el zumbador.

Reto A03.1. 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.



Prueba con este sencillo programa cómo suena el zumbador.



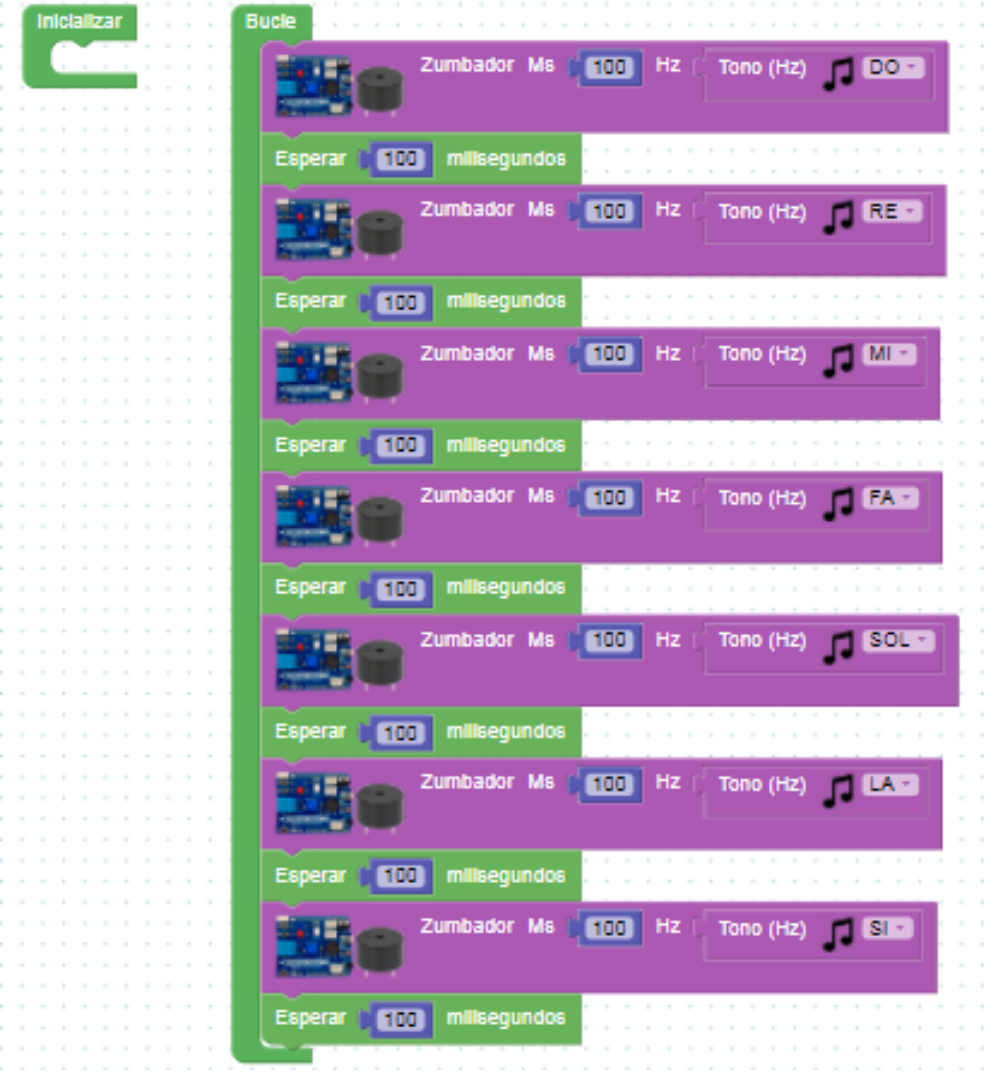
Actividad de ampliación: cambia ligeramente el programa introduciendo tiempos diferentes en la duración de las notas y en las esperas para observar las diferencias que aparecen en la ejecución del programa.

7.3.2 Reto A03.2. Escalas musicales con el zumbador.

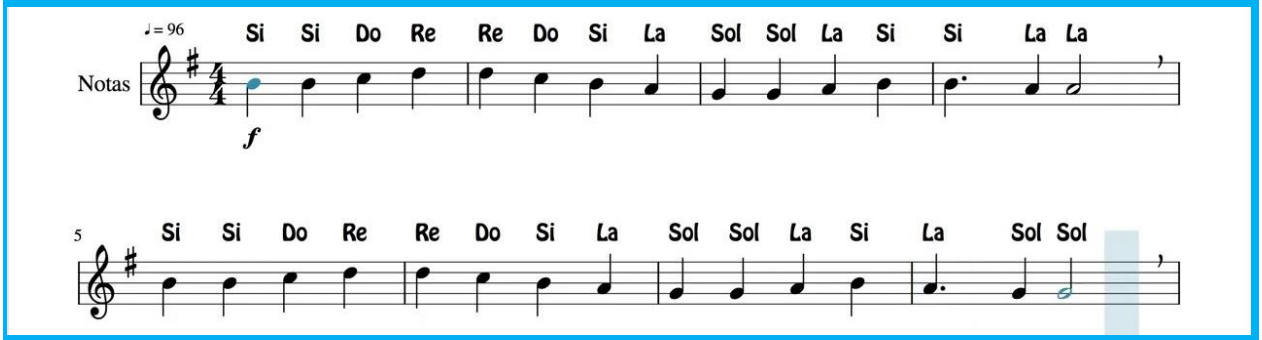
Reto A03.2. Escalas musicales con el zumbador.

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.

Haremos una pequeña escala musical.



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.



7.3.3 Reto A03.3. Melodías con RTTTL.

Reto A03.3. 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.

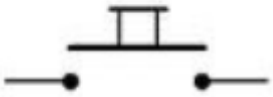
Realiza este programa y elige una de las melodías que hay disponibles en el menú desplegable del bloque RTTTL.

Actividad de ampliación: prueba con diferentes melodías RTTL.

7.4 Reto A04. El pulsador.

Reto A04. El pulsador.

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.



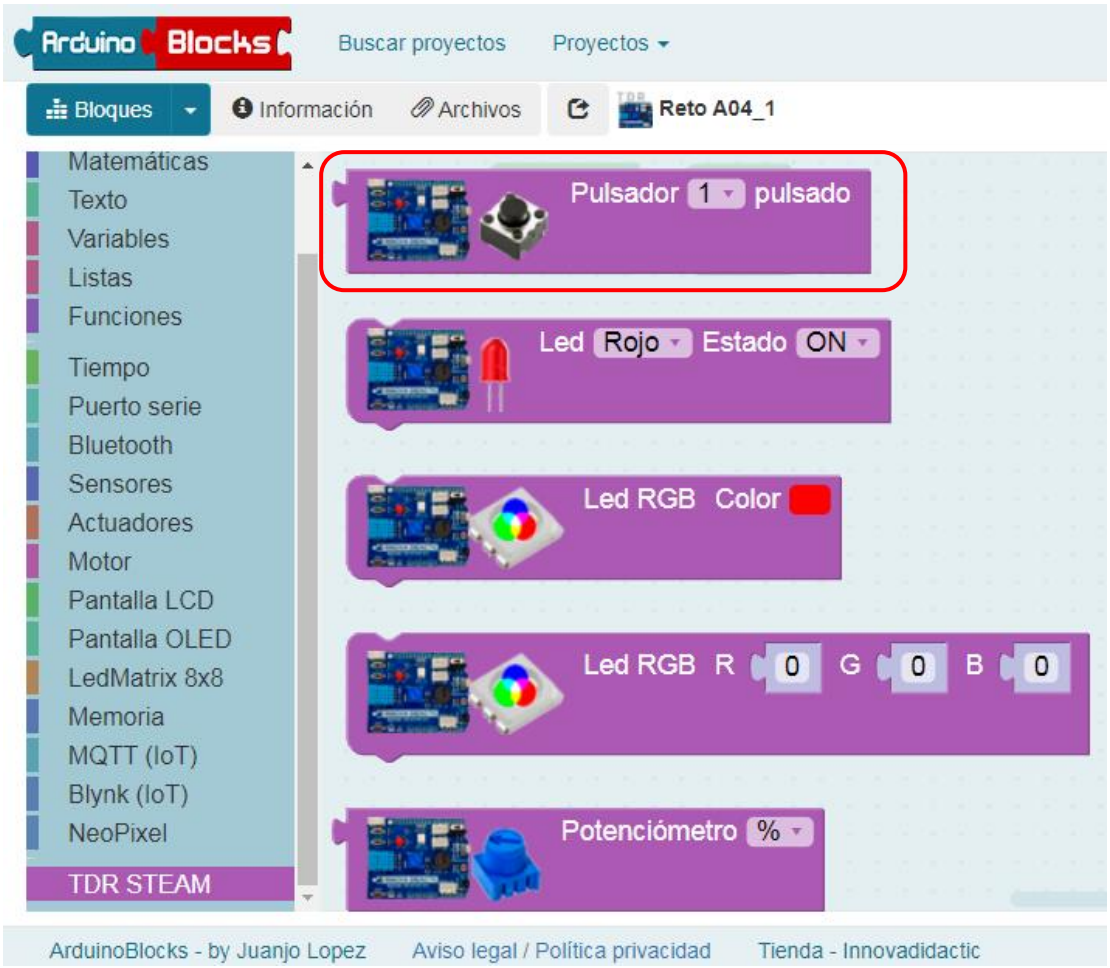
Símbolo

Componente

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 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.


Arduino Blocks
Buscar proyectos
Proyectos
Bloques
Información
Archivos
Reto A04_1
Matemáticas
Texto
Variables
Listas
Funciones
Tiempo
Puerto serie
Bluetooth
Sensores
Actuadores
Motor
Pantalla LCD
Pantalla OLED
LedMatrix 8x8
Memoria
MQTT (IoT)
Blynk (IoT)
NeoPixel
TDR STEAM
Pulsador 1 pulsado
Led Rojo Estado ON
Led RGB Color
Led RGB R 0 G 0 B 0
Potenciómetro %
ArduinoBlocks - by Juanjo Lopez
Aviso legal / Política privacidad
Tienda - Innovadidactic

7.4.1 Reto A04.1. Control ON/OFF de un led con un pulsador I.

Reto A04.1. 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:

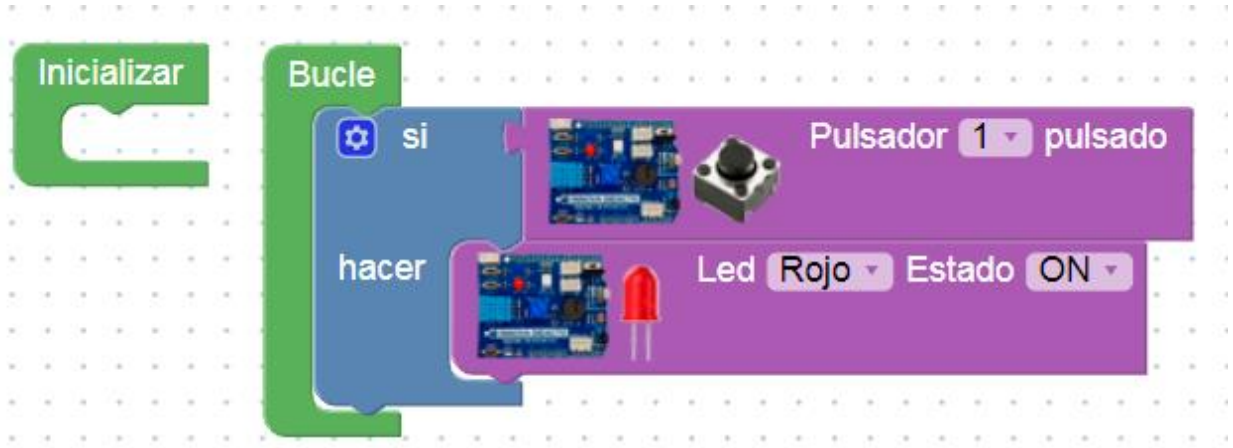
si (condición) hacer (acciones)

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.

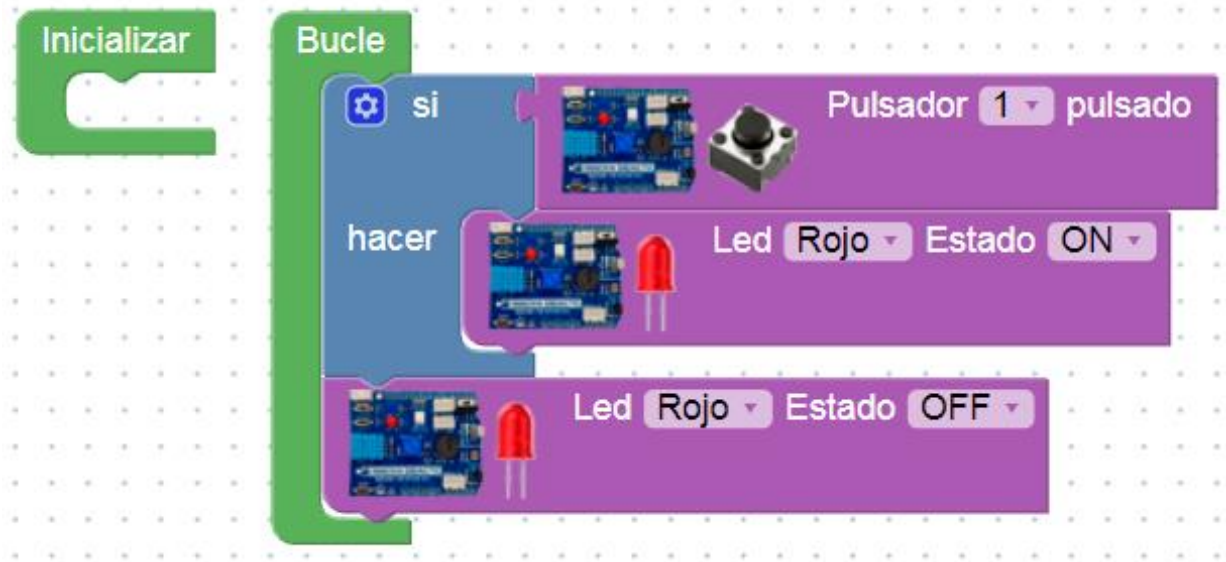
El funcionamiento es el siguiente: si se cumple la condición incluida en su primer apartado, entonces se realiza la acción incluida en su segundo aparado. En caso contrario, no se hace nada.

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

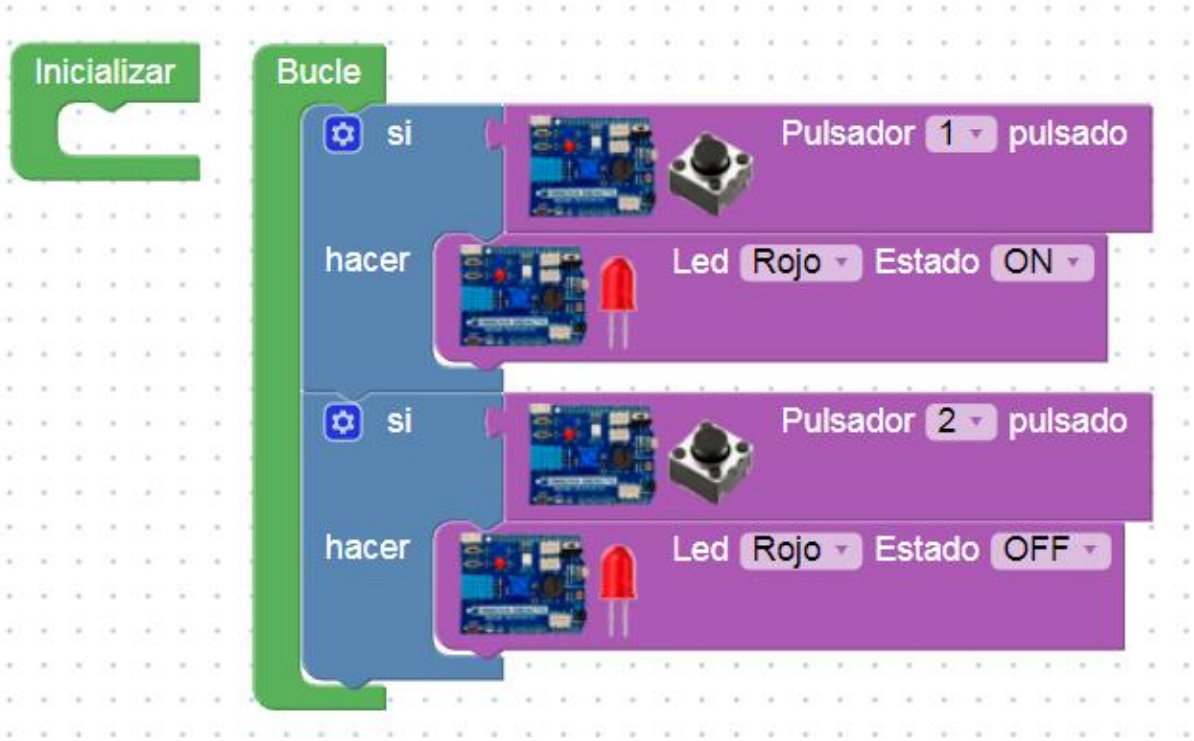
Usando el bloque lógico *condicional de Si…. hacer…* el programa quedaría como la imagen.



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.



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

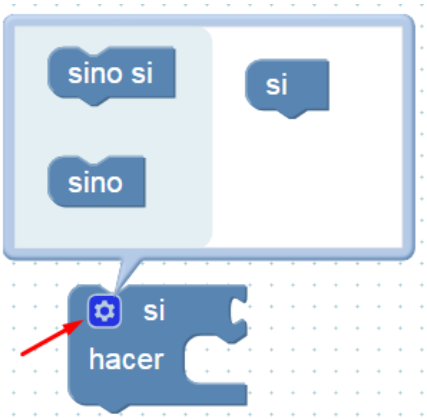


Actividad de ampliación: prueba ahora de apaga los dos leds con el pulsador SW1 y encenderlos con SW2.

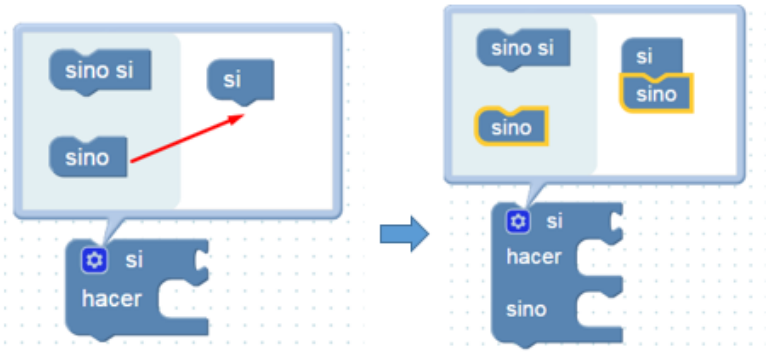
7.4.2 Reto A04.2. Control ON/OFF de un led con un pulsador II.

Reto A04.2. Control ON/OFF de un led con un pulsador II.

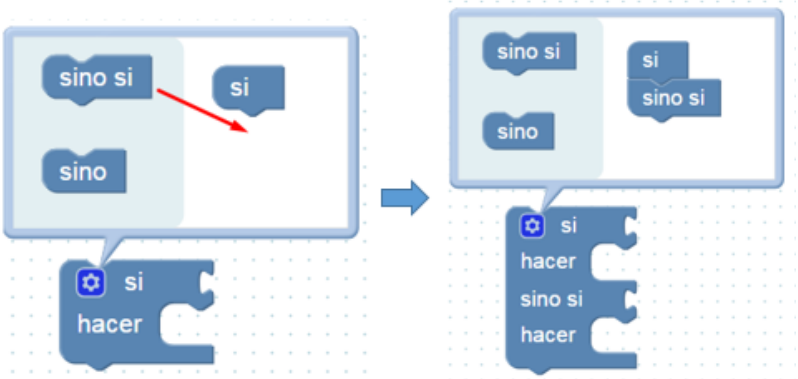
Necesitar dos pulsadores para tener que controlar un solo Led no parece la mejor opción, necesitamos introducir un sino. Si usamos un único pulsador debemos poner una nueva condición: un *sino* para cambiar el estado del led a OFF. Para ello debemos ampliar el condicional. Pulsando sobre el símbolo del engranaje (señalado con la flecha roja en la imagen) nos aparece un cuadro con funciones con las que podemos ampliar el condicional *Si*.



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



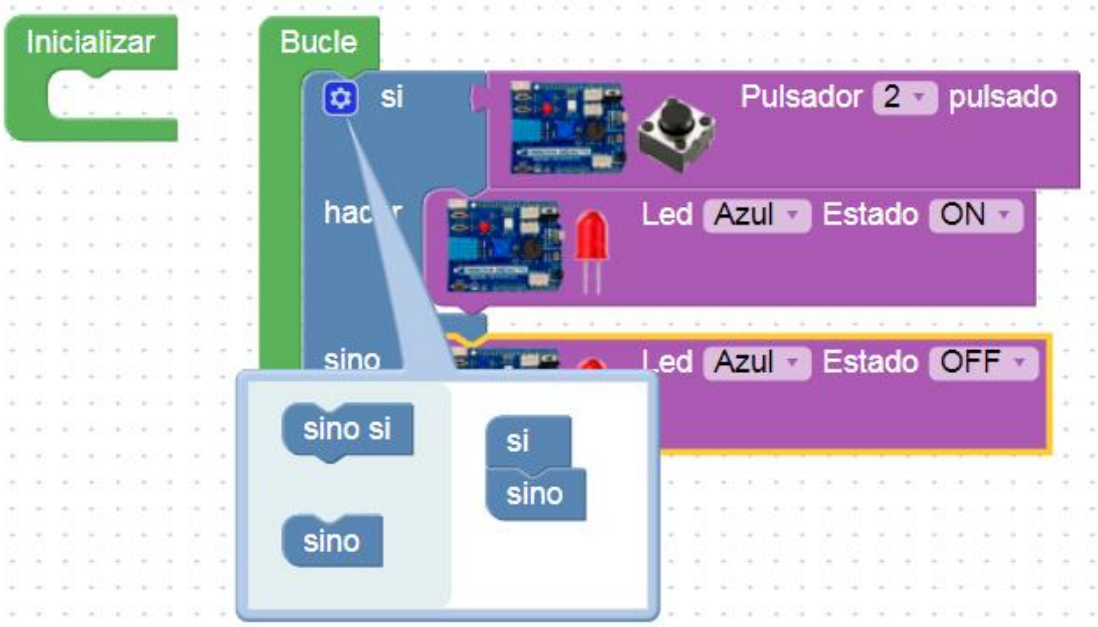
Opción A



Opción B

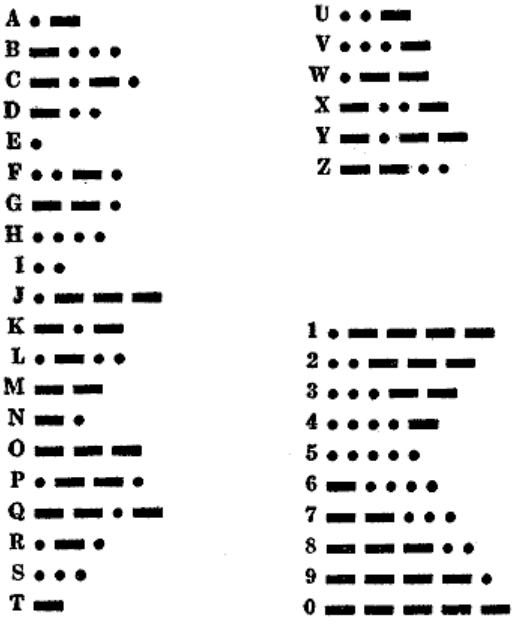
Podemos ir encadenando varias estructuras (opción B).

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.



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:



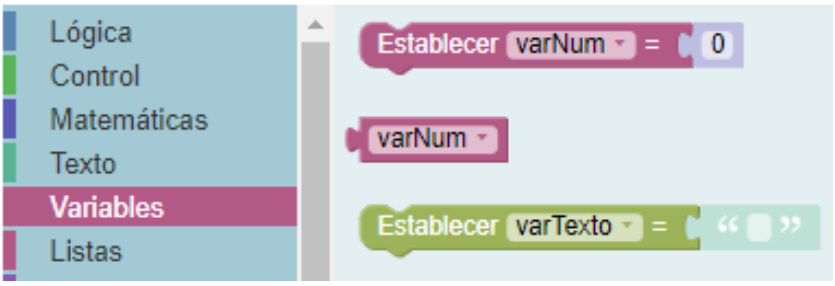
Actividad de ampliación: haz una máquina de código Morse que envíe un mensaje de ayuda (SOS).

7.4.3 Reto A04.3. Control ON/OFF de un led con un pulsador III.

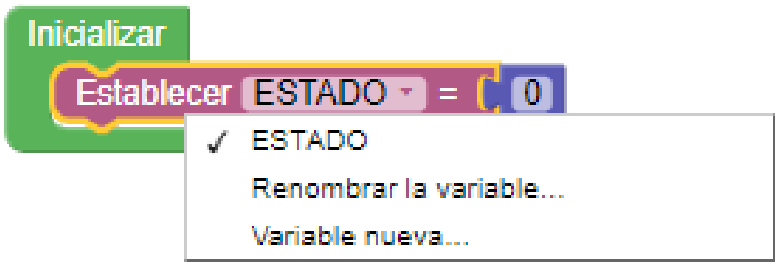
Reto A04.3. Control ON/OFF de un led con un pulsador III.

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*.

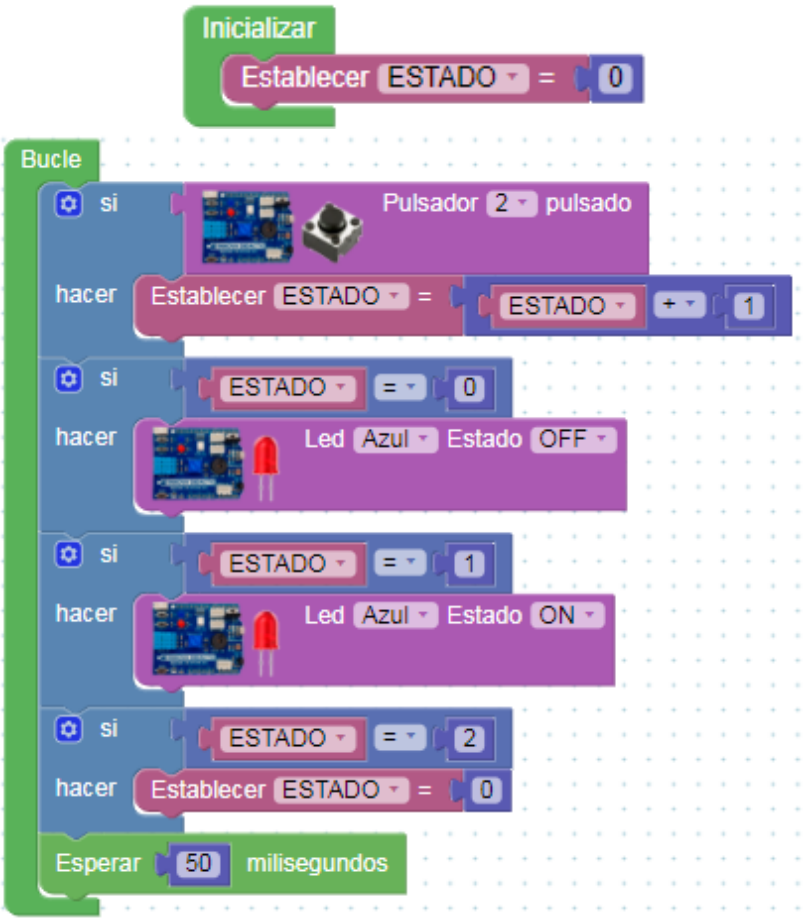


En *Inicializar* establecemos la variable a 0 y la llamamos *ESTADO*. Esta variable la utilizaremos como indicador de cambio de estado del pulsador.



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.

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:

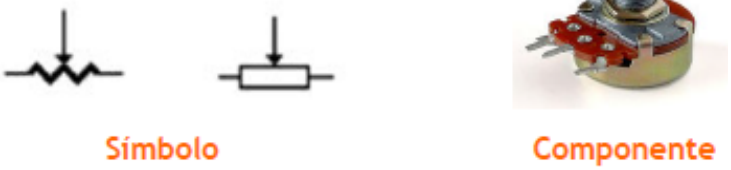


Actividad de ampliación: intenta hacer el mismo programa, pero más simplificado (utilizando menos instrucciones).

7.5 Reto A05. El potenciómetro.

Reto A05. El potenciómetro.

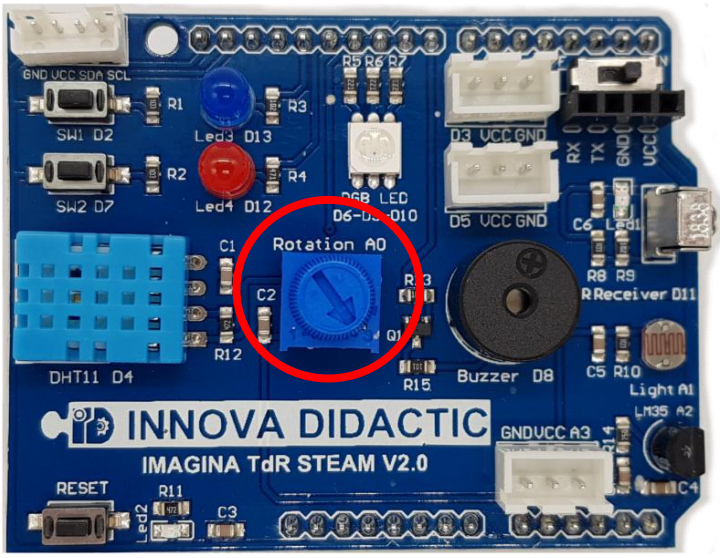
Un potenciómetro es una resistencia cuyo valor es variable ya que son un tipo de resistencias especiales que tienen la capacidad de variar su valor cambiando de forma mecánica su posición. Con ellos indirectamente, se puede controlar la intensidad de corriente que fluye por un circuito si se conecta 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, brillo, etc.



Símbolo

Componente

La placa Imagina TDR STEAM tiene un potenciómetro denominado *Rotation* que van asociado al pin A0. Las entradas $A_{número}$ 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).



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 2^{10} 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.

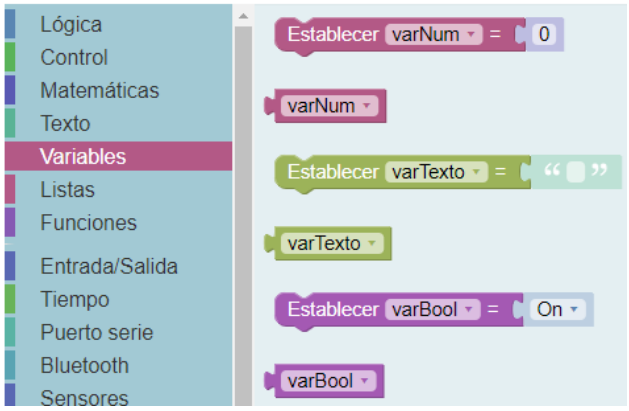
En el desplegable del bloque del sensor, podemos elegir su lectura en porcentaje (%) o en valor (de 0 a 1023).

7.5.1 Reto A05.1. Lectura de valores con el puerto serie.

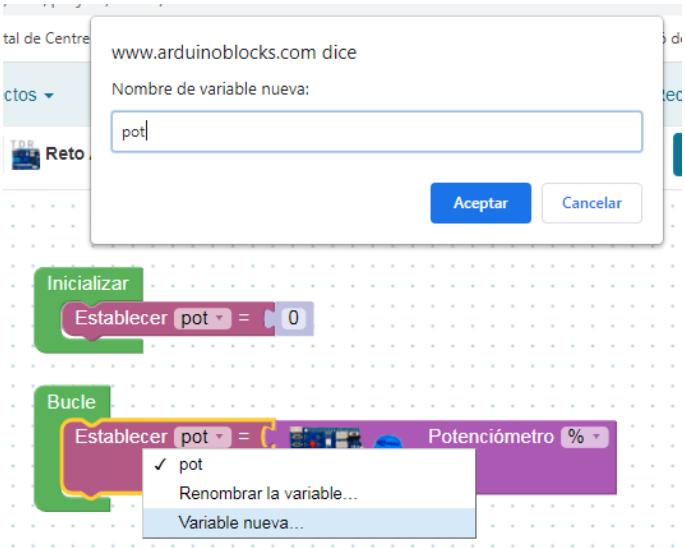
Reto A05.1. Lectura de valores con el puerto serie.

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*.

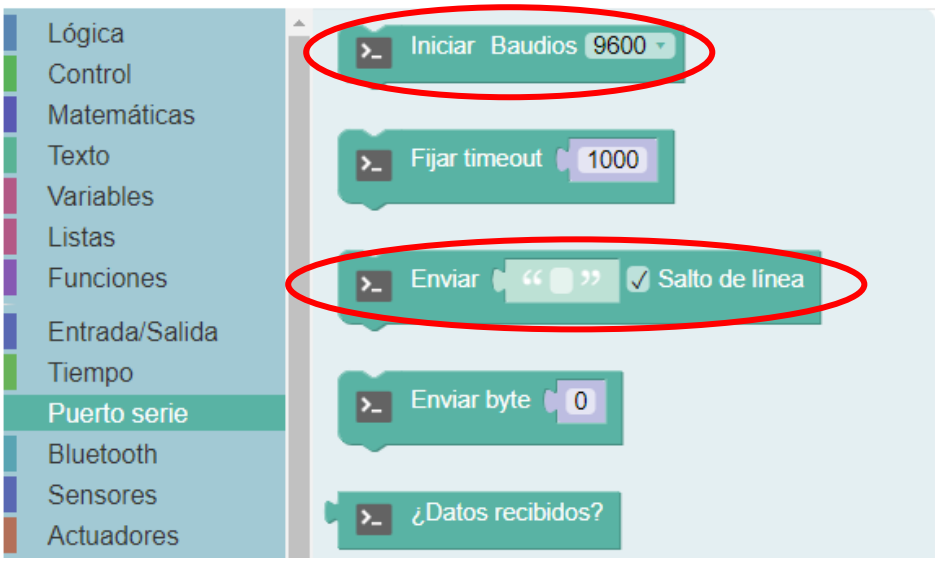


Para cambiar el nombre de la variable pulsaremos sobre el menú desplegable del bloque de la variable y elegiremos *Variable nueva…* nos aparecerá una ventana en la que escribiremos el nuevo nombre y daremos a *Aceptar*. Ahora fijaremos el valor de la variable al valor del potenciómetro, tal y como está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

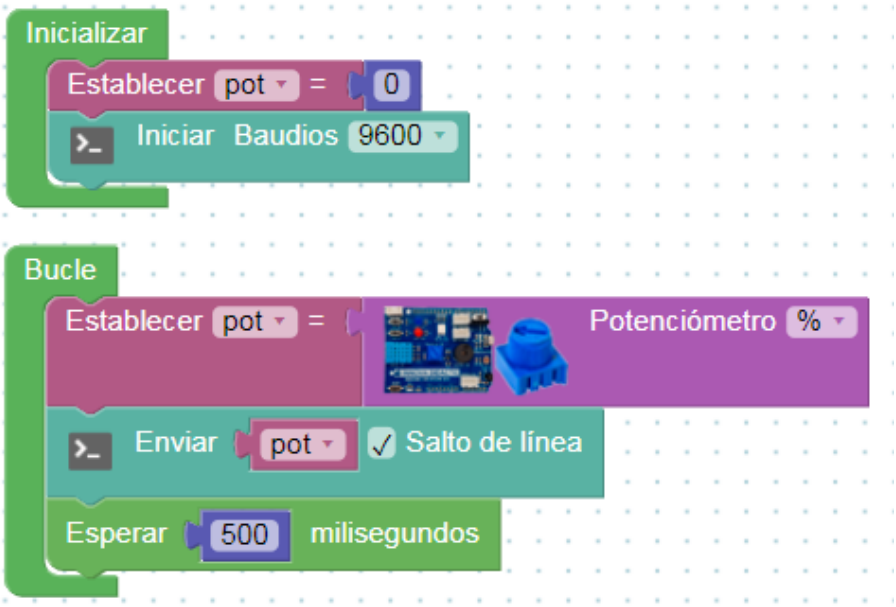


Es importante establecer la variable con el valor del potenciómetro dentro de *Bucle*, ya que si sólo se hace en *Inicializar* el valor siempre será el mismo a lo largo de todo el programa. En otras ocasiones interesa establecer las variables en el inicio, pero no es este el caso.

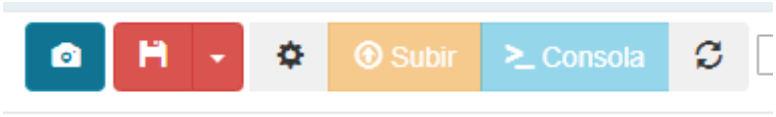
Continuando con el programa, ahora nos faltan los bloques del *Puerto Serie*. El primero que debemos utilizar es el *Iniciar Baudios 9.600* que siempre lo colocaremos en el Inicio y después el bloque *Enviar.*



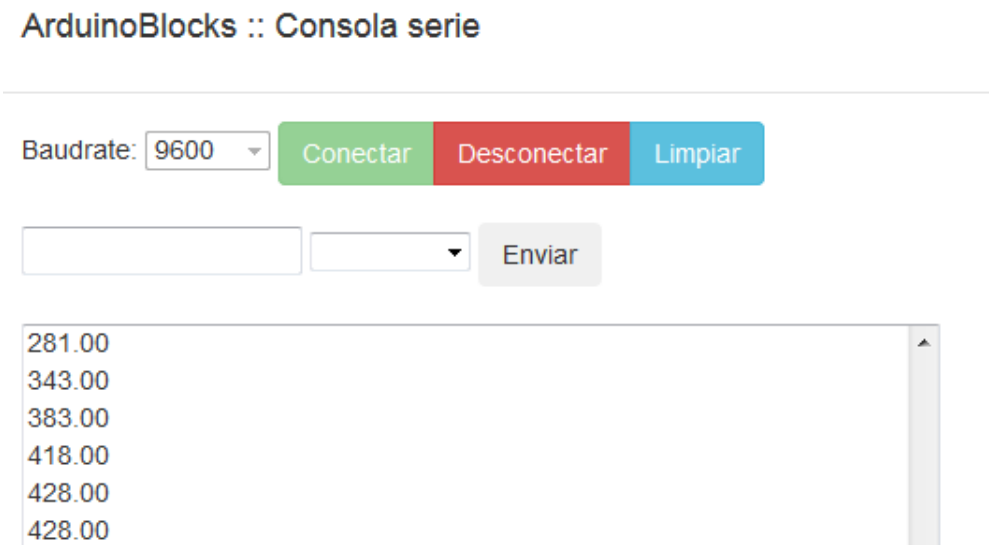
Observa cómo queda el programa resultante:



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



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.



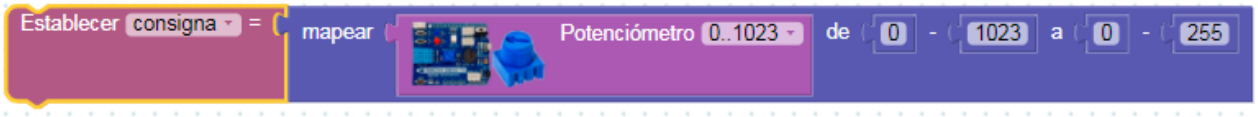
Actividad de ampliación: prueba ahora quitando el tic de *Salto de línea* a ver qué sucede.

7.5.2 Reto A05.2. Ajuste de valores de entrada y salida: mapear.

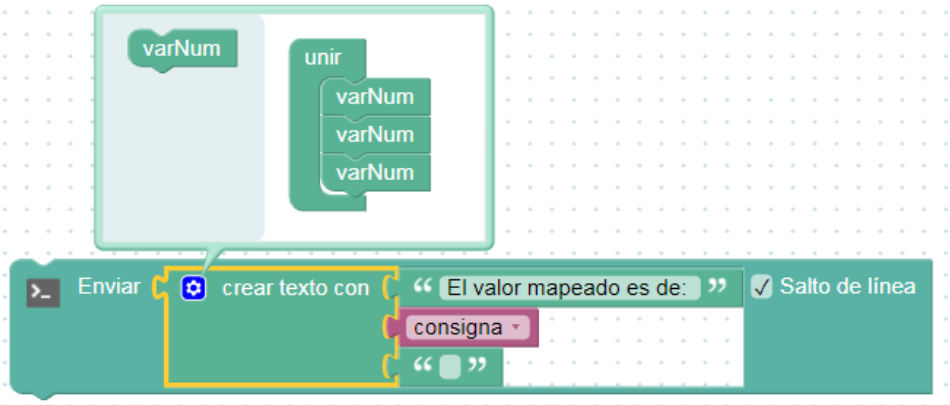
Reto A05.2. Ajuste de valores de entrada y salida: mapear.

Existe un pequeño “problema” entre las entradas y las salidas en Arduino. Las entradas trabajan con 10 bits (2^{10} valores = 0 a 1023) y las salidas trabajan a 8 bits (2^8 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.

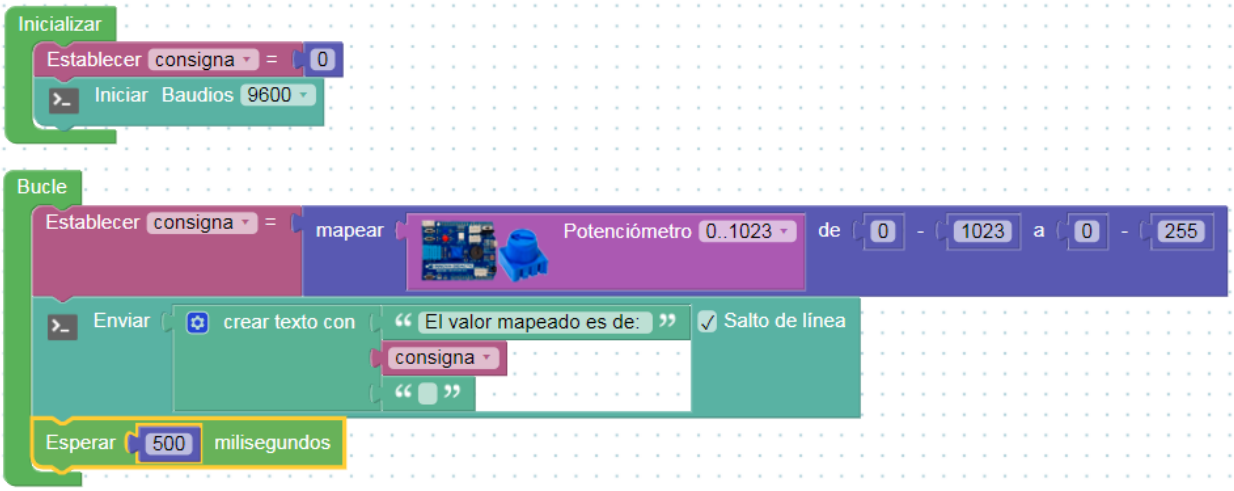
En esta actividad vamos a imaginar que con el potenciómetro queremos definir un rango de valores entre 0 a 255. Para ello definiremos una variable, llamada *consigna*, que será el valor *mapeado* del potenciómetro. En el potenciómetro cambiaremos su opción para obtener datos 0...1023.



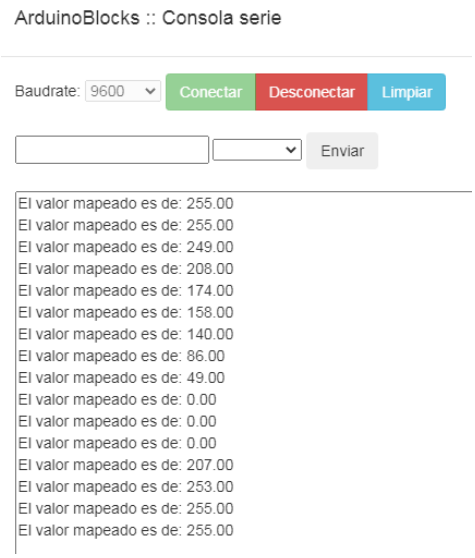
Continuando el programa para poder realizar lecturas por el puerto serie utilizaremos un nuevo bloque de *crear texto con...*. Fíjate como al pulsar sobre el símbolo del mecanismo podemos ampliar las líneas añadiendo *varNum* a la parte derecha.



El programa resultante quedará de la siguiente forma:



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



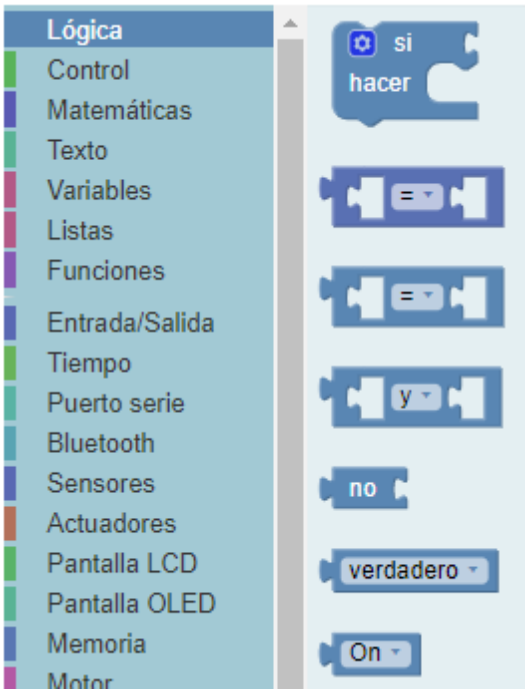
Actividad de ampliación: cambia ahora el rango de salida y el texto que envía por el puerto serie.

7.5.3 Reto A05.3. Control del led RGB con el potenciómetro.

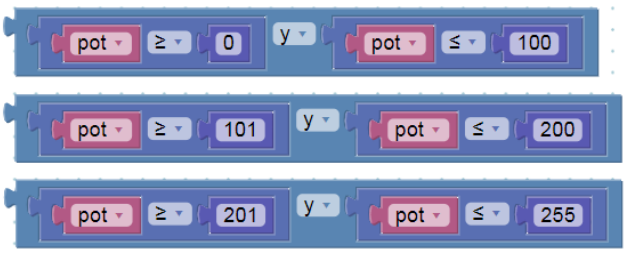
Reto A05.3. Control del led RGB con el potenciómetro.

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 255 que sea azul.

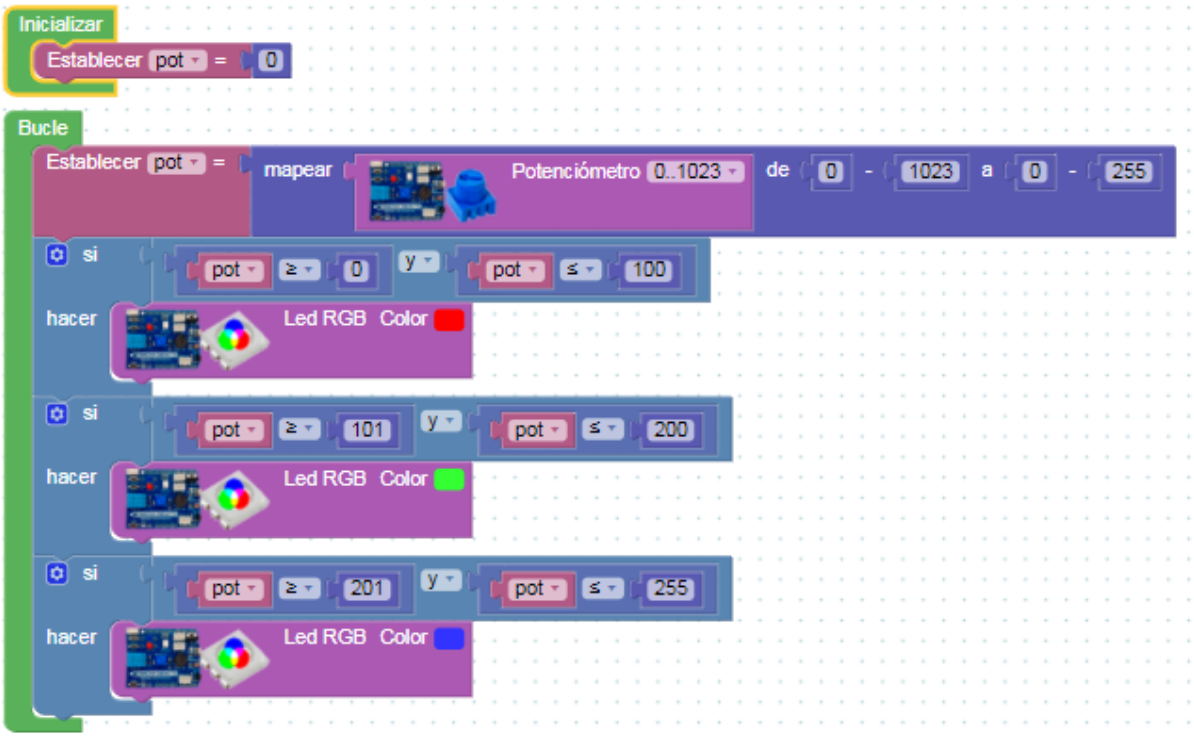
Del menú *Lógica* vamos a necesitar dos nuevos bloques; el bloque de *Evaluar condición* y el bloque de *Conjunción/Disyunción*. Con ellos crearemos estas condiciones:



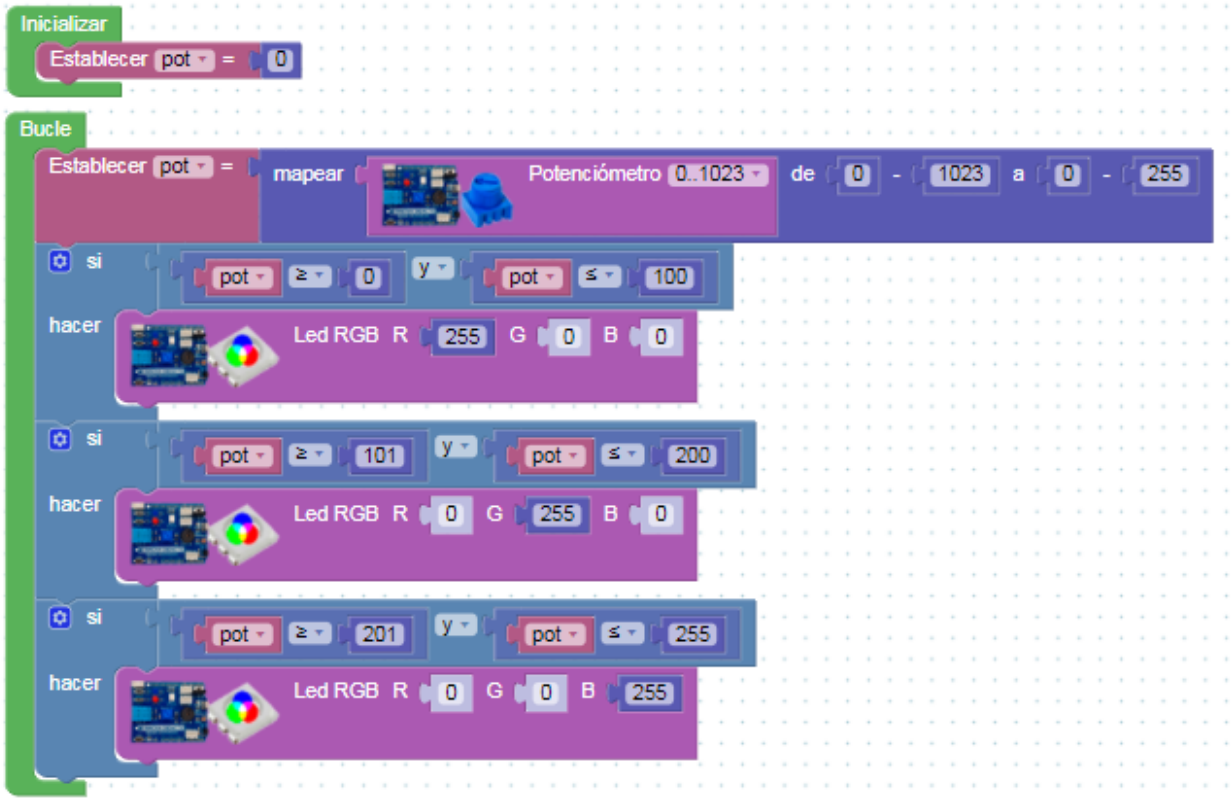
Deberemos crear tres estructuras para hacer los tres rangos.



El programa quedaría como muestra la imagen:



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



Actividad de ampliación: completa el programa con más condiciones.

7.6 Reto A06. La fotocélula (LDR – sensor de luz).

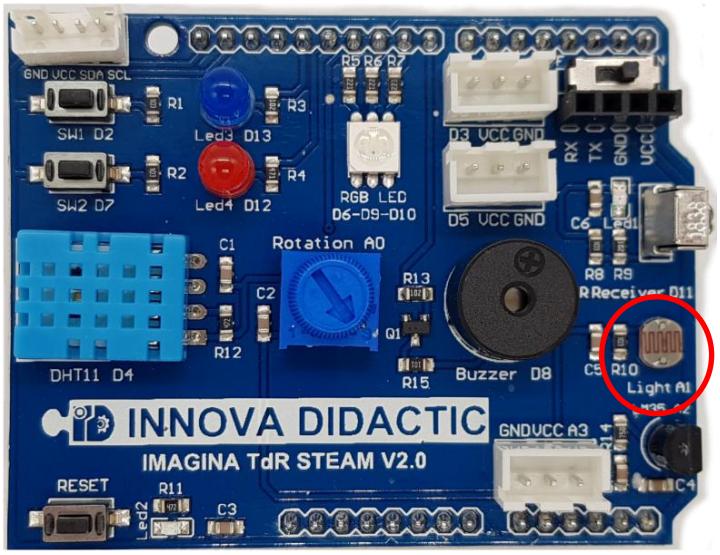
Reto A06. La fotocélula (LDR – sensor de luz).

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.

Símbolo

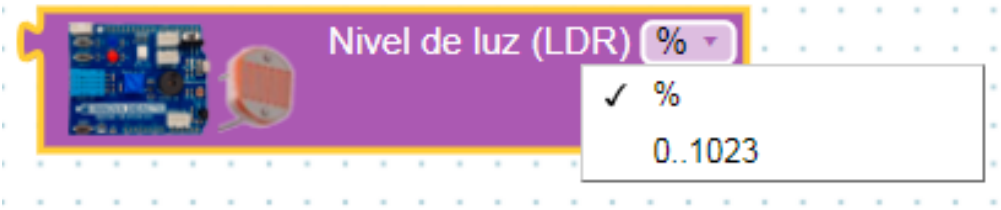
Componente

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



En el menú TDR STEAM 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.



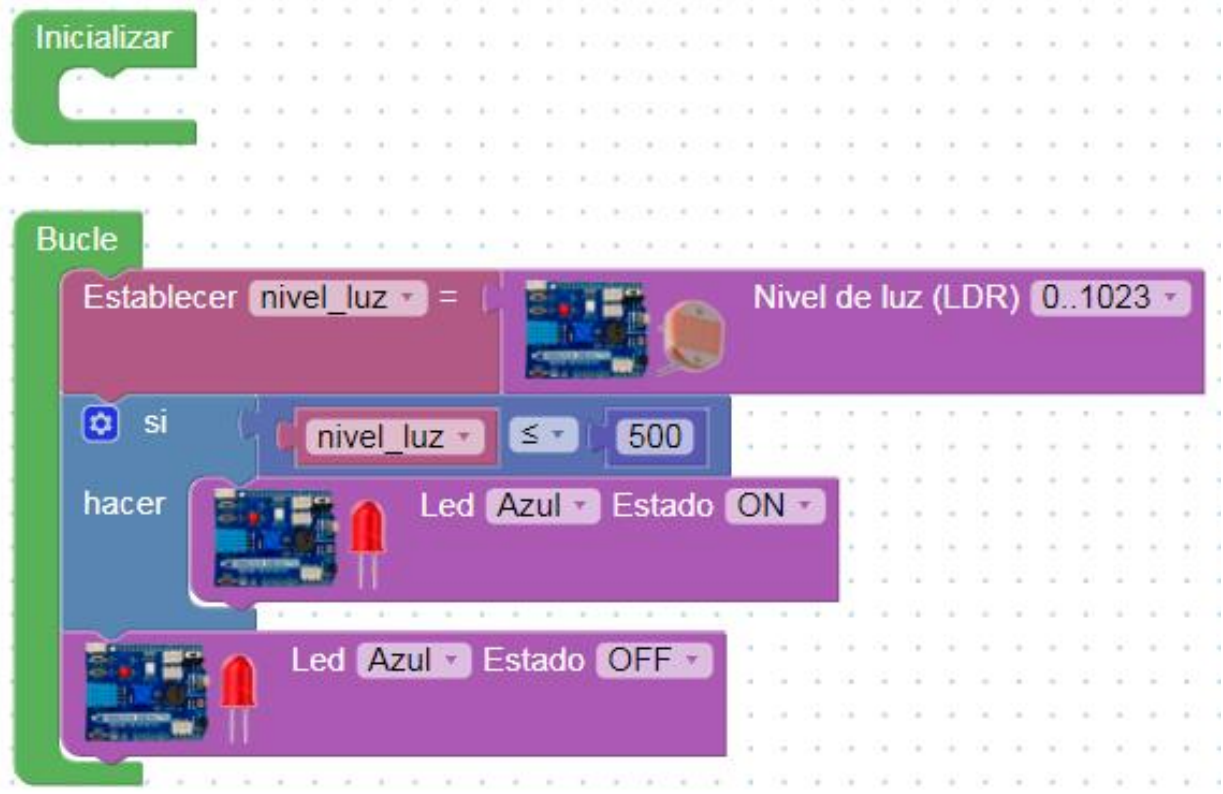
7.6.1 Reto A06.1. Encender y apagar un led según el nivel de luz.

Reto A06.1. 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.



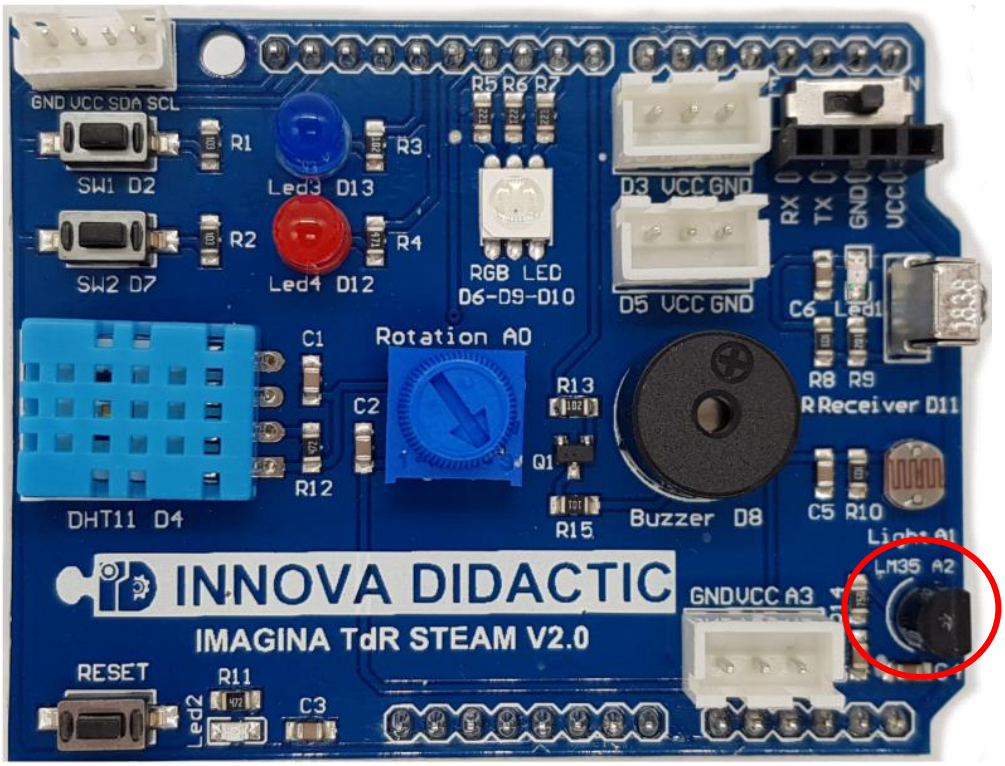
Actividad de ampliación: haz un programa que muestre los valores de la LDR por el puerto serie.

7.7 Reto A07. Sensor de temperatura LM35D.

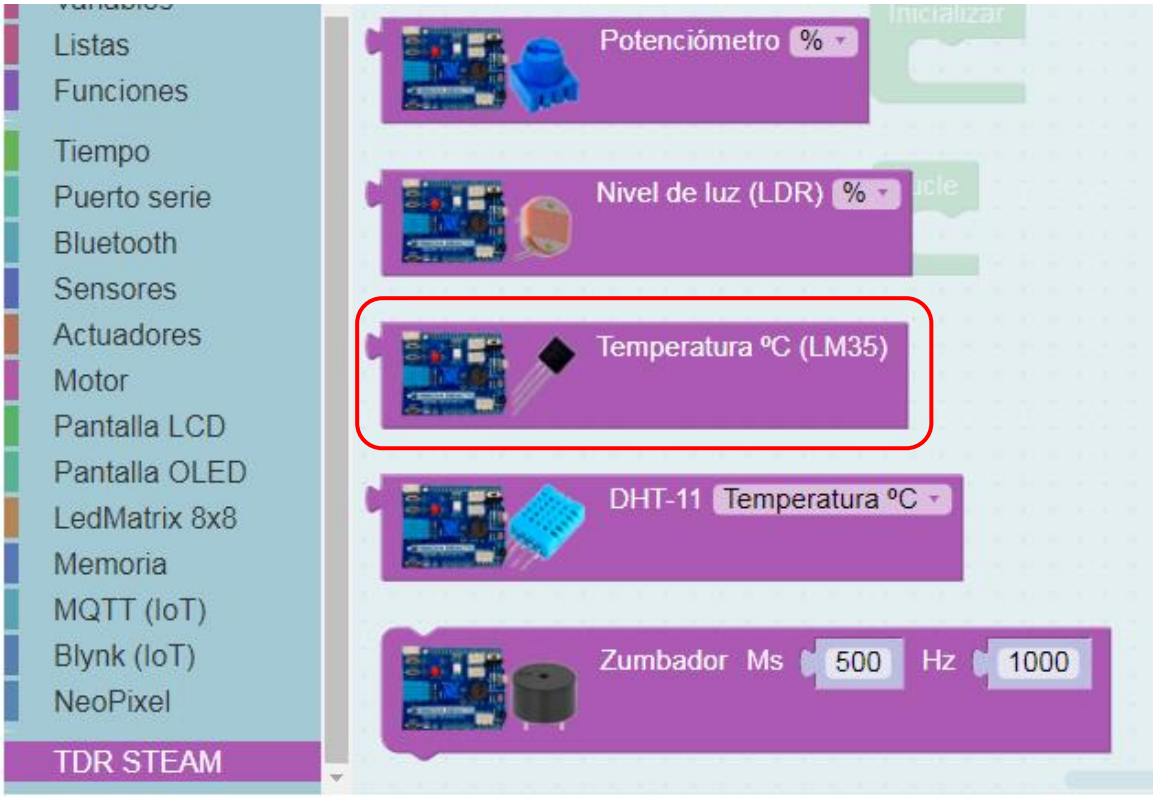
Reto A07. Sensor de temperatura LM35D.

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 ℃ y una sensibilidad de 10mV/ºC.

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



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


Listas
Funciones
Tiempo
Puerto serie
Bluetooth
Sensores
Actuadores
Motor
Pantalla LCD
Pantalla OLED
LedMatrix 8x8
Memoria
MQTT (IoT)
Blynk (IoT)
NeoPixel
TDR STEAM
Potenciómetro %
Nivel de luz (LDR) %
Temperatura ºC (LM35)
DHT-11 Temperatura ºC
Zumbador Ms 500 Hz 1000

7.7.1 Reto A07.1. Lectura del valor de la temperatura.

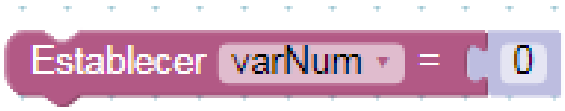
Reto A07.1. Lectura del valor de la temperatura.

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.

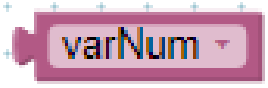
Las variables son elementos muy comunes en programación. Básicamente, crear una variable es darle un nombre a un dato o a una lectura. Por ejemplo, las mediciones de valores de temperatura las podemos guardar en una variable que se llame “Temperatura” o las del sensor de ultrasonidos en una llamada “Distancia”,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

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

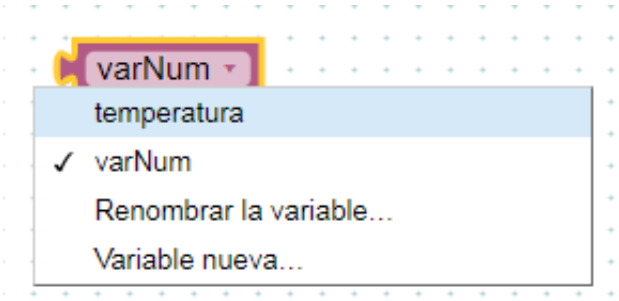


- Y el bloque de la propia variable creada, para poder insertarla y combinarla con otros bloques:

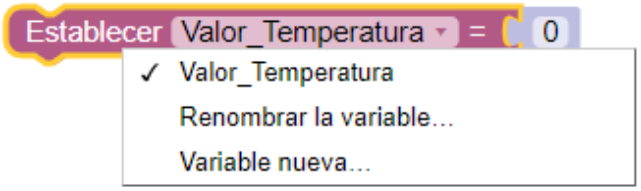


También podemos personalizar el nombre de la variable, de la siguiente forma:

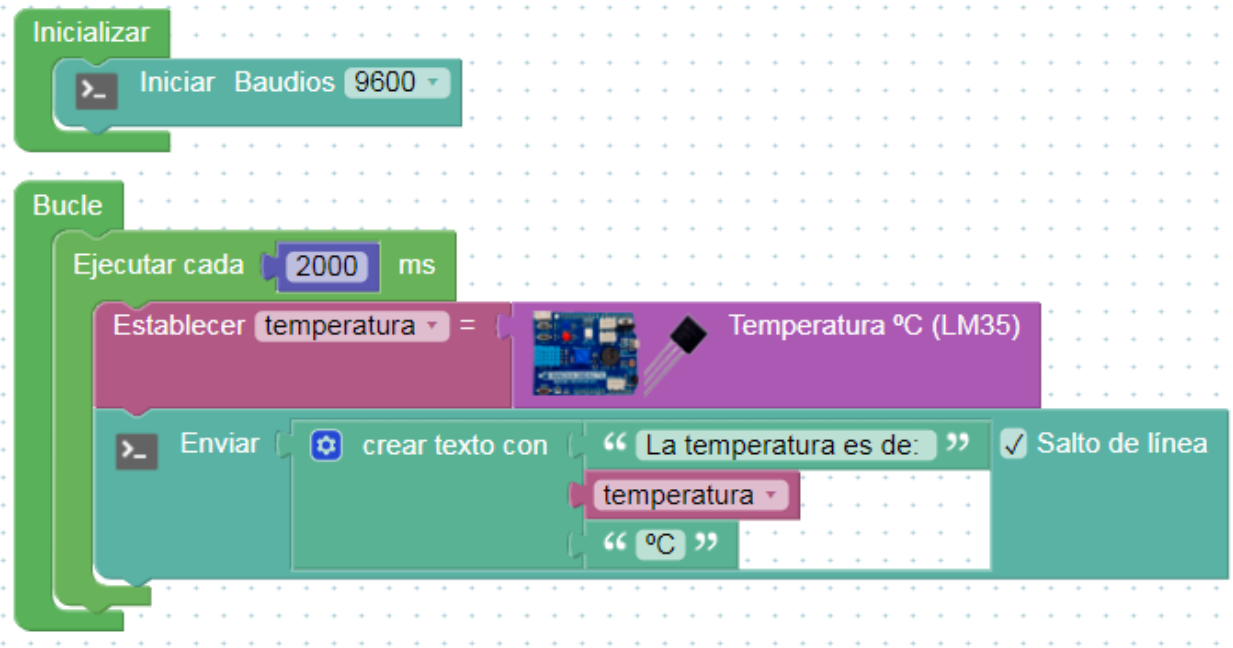
Una vez creada la nueva variable, podemos seleccionarla pulsando sobre el desplegable:



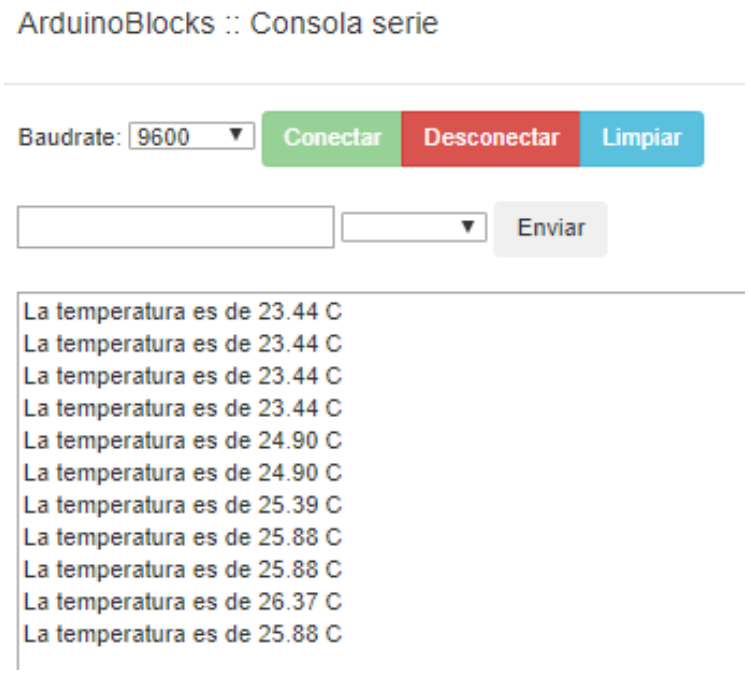
Hay que tener en cuenta que las variables solo pueden estar formadas por una palabra. Si quieres incluir varias palabras, puedes usar el truco de separarlas con una barra baja "_", como en el ejemplo, "*Valor_Temperatura*".



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



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



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

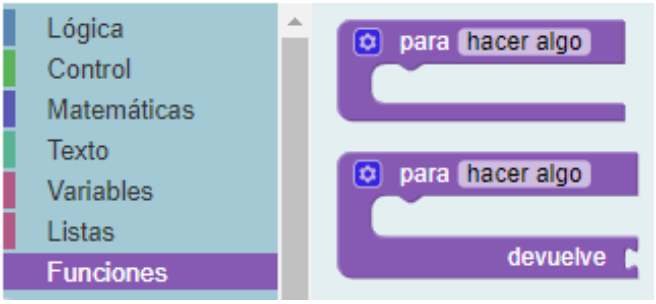
7.7.2 Reto A07.2. Alarma por exceso de temperatura.

Reto A07.1. Alarma por exceso de temperatura.

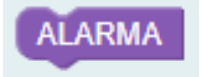
Ahora vamos a hacer una alarma sonora y acústica. El programa consistirá en hacer que el led rojo y el zumbador se accionen a la vez cuando el sensor de temperatura detecte una temperatura superior a 28ºC.

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.

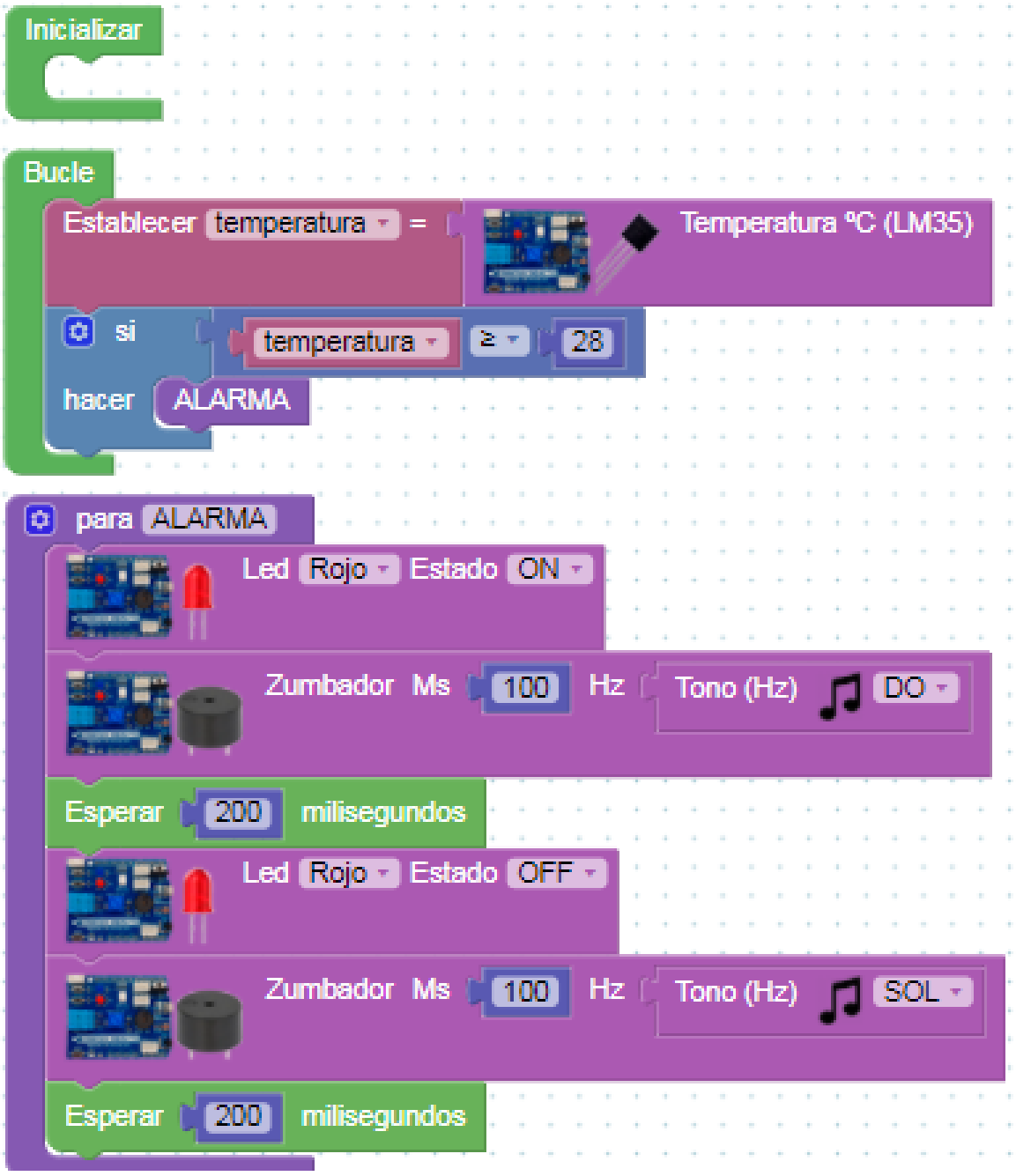
Las funciones nos permiten realizar tareas que se repiten a lo largo del programa. En el menú "Funciones" tenemos el bloque "*para () hacer algo*". Lo utilizaremos para crear nuestra función como la siguiente imagen.



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



El programa final sería el siguiente:

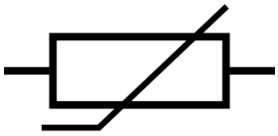


Actividad de ampliación: haz un programa que varíe el programa anterior para que encienda de color verde el led RGB si la temperatura inferior a los 28ºC y rojo si la temperatura es superior a los 35ºC. En el rango intermedio se indicará de color azul.

7.8 Reto A08. Sensor de temperatura y humedad DHT11.

Reto A08. Sensor de temperatura y humedad 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 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.



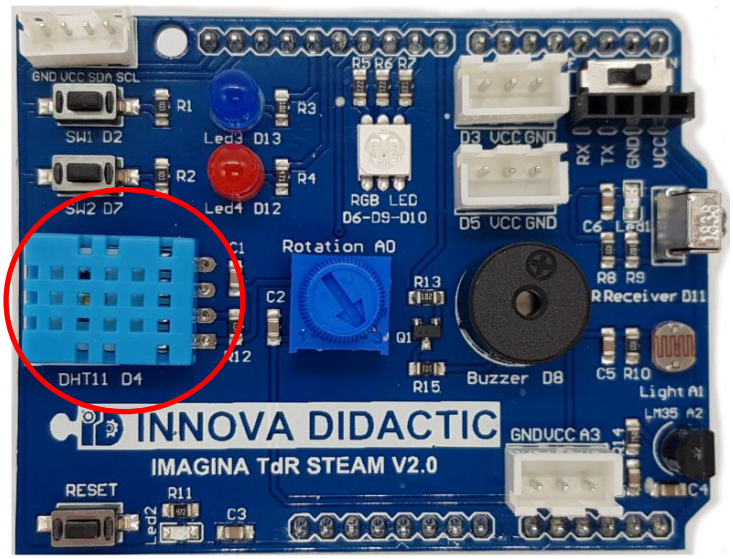
El sensor de temperatura es un termistor tipo NTC. Un termistor es un tipo de resistencia (componente electrónico) cuyo valor varía en función de la temperatura de una forma más acusada que una resistencia común. Su funcionamiento se basa en la variación de la resistividad que presenta un semiconductor 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.



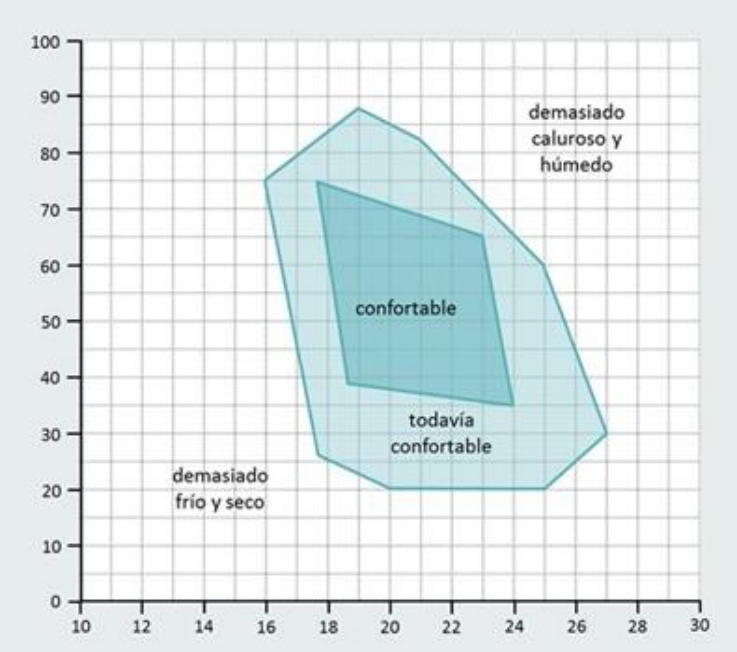
En ArduinoBlocks en el menú de sensores tenemos el bloque específico para programar este sensor:

7.8.1 A08.1. Zona de confort con DHT11.

Reto A08.1. 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.



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:

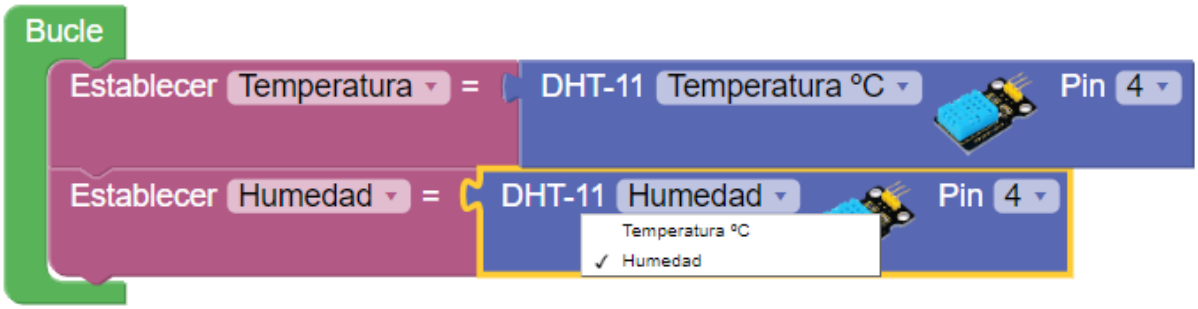
- *Zona ROJA:*
  - Humedad por debajo del 20% y superior al 85%.
  - Temperatura por debajo de 16ºC o superior a 26,5ºC.
- *Zona NARANJA:*
  - Humedad entre el 20% y el 40% y entre el 65% y el 85%.
  - Temperatura entre los 16ºC y los 18ºC o entre los 24 y los 26,5ºC.
- *Zona VERDE:*
  - Humedad entre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

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.*

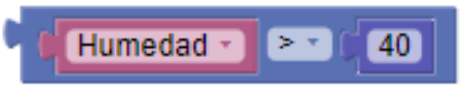
- *Y*: se cumple si los dos operandos son verdaderos.
- *O*: se cumple si alguno de los dos operandos es verdadero.

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 o la Humedad.

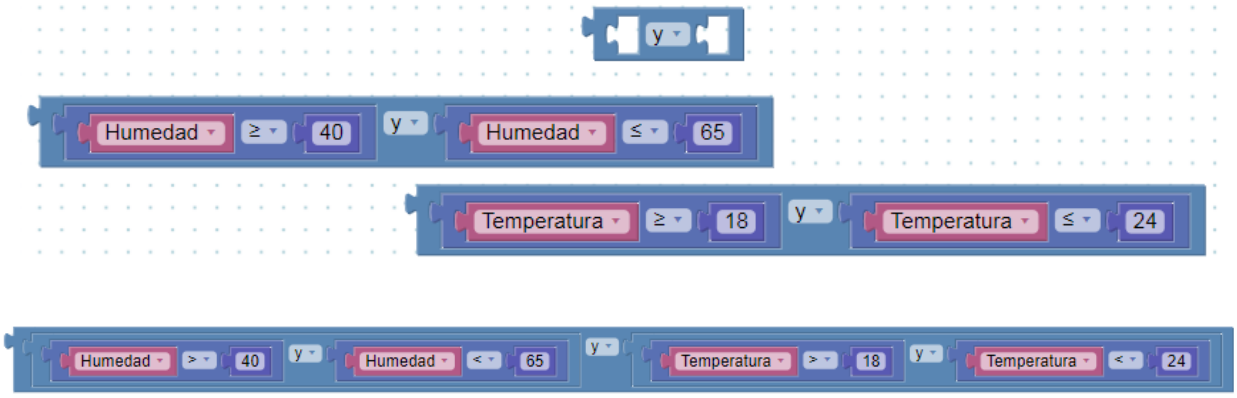


Usando 3 bloques de conjunciones debes crear el siguiente bloque:

Después ir metiendo las condiciones en cada uno de ellos:

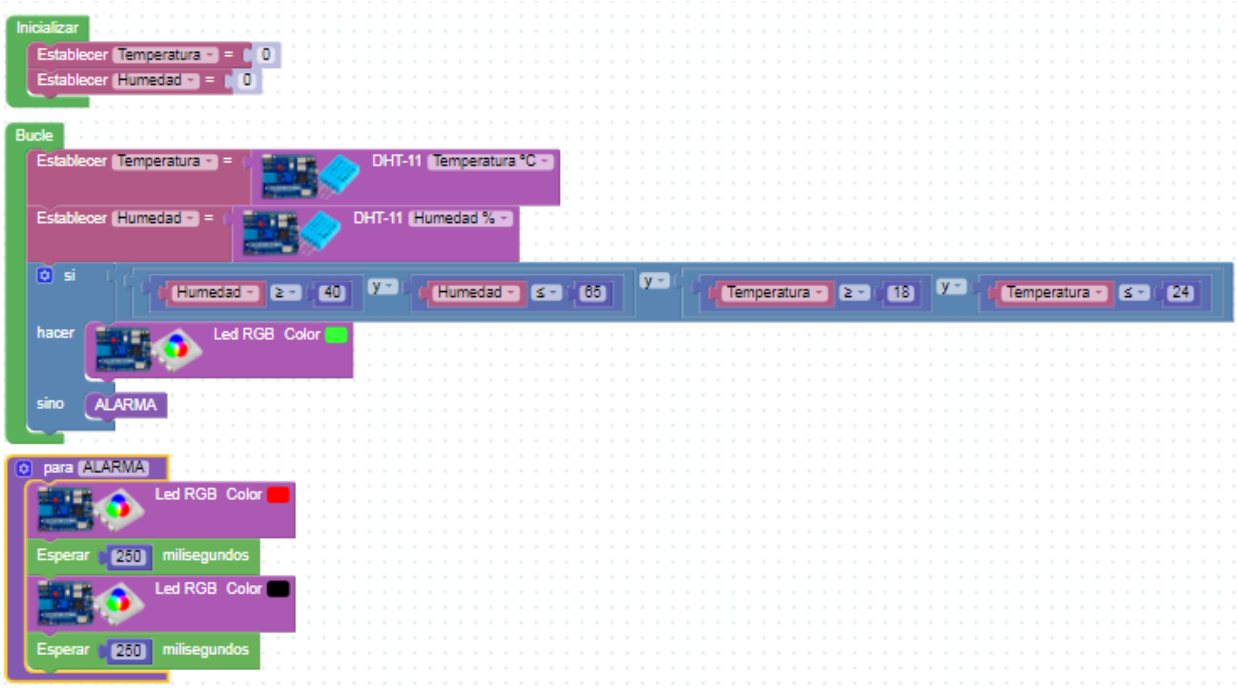


Y ve uniendo todo hasta conseguir esta condición final:



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

Y con todo esto, el programa final quedaría así:



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.

7.9 Reto A09. Receptor de infrarrojos (IR).

Reto A08. Receptor de infrarrojos (IR).

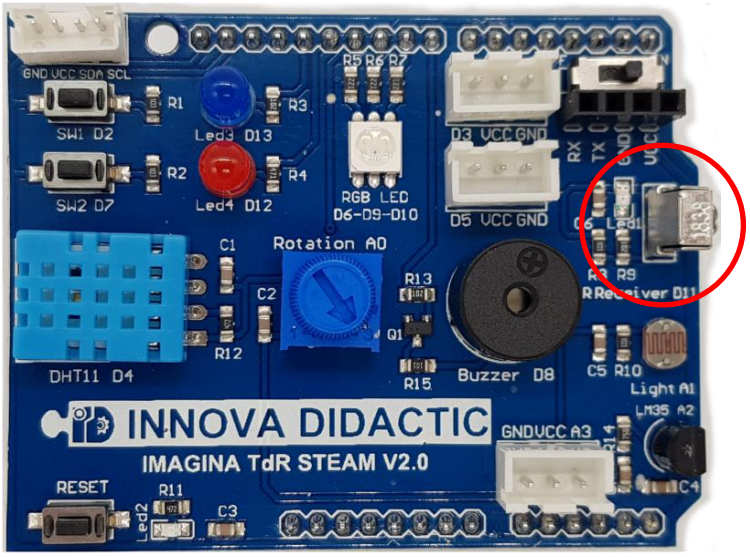
Una gran parte de los electrodomésticos utilizan mandos a distancia de infrarrojos, como los televisores o equipos musicales. El sensor infrarrojo es un dispositivo optoelectrónico capaz de medir la radiación electromagnética infrarroja de los cuerpos en su campo de visión. Todos los cuerpos emiten una 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.

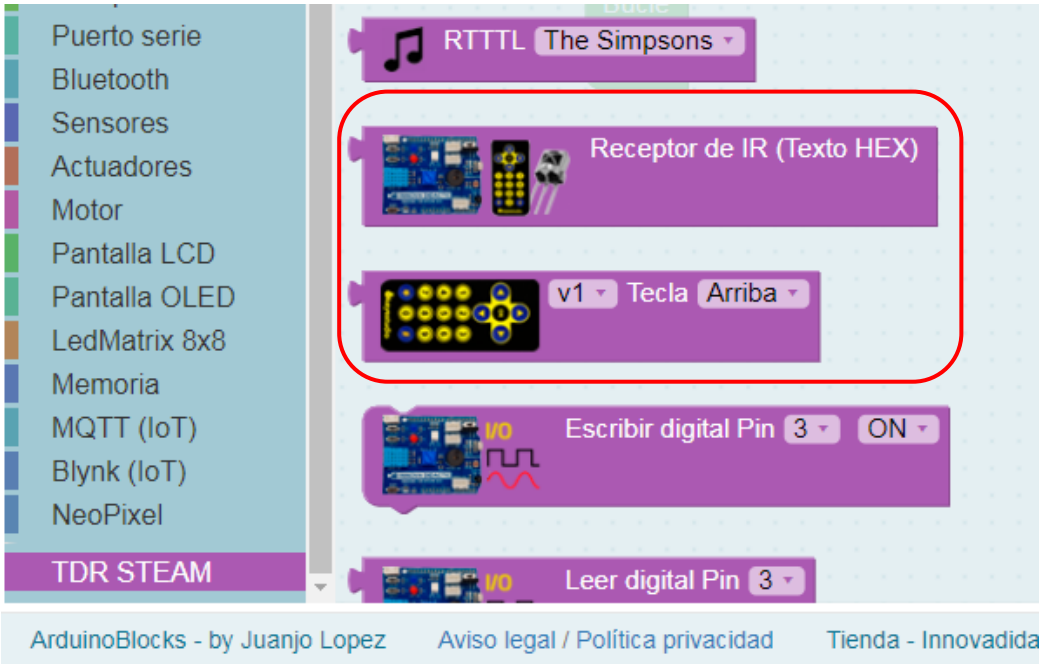
En el caso del receptor de infrarrojos (IR) de la placa IMAGINA TDR STEAM permite codificar los protocolos de señales de pulsos infrarrojos utilizados por los mandos a distancia. Los protocolos detectados son los siguientes: RC5, RC6, NEC, SONY, PANASONIC, JVC, SAMSUNG, WHYNTER, AIWA, LG, SANYO, MITSUBISHI 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.





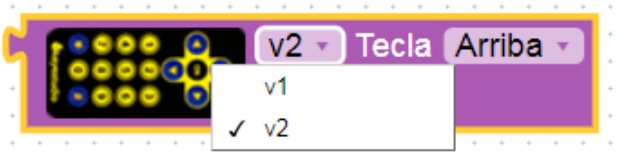
7.9.1 Reto A09.1. Recepción de comandos por infrarrojos.

Reto A09.1. Recepción de comandos por infrarrojos.

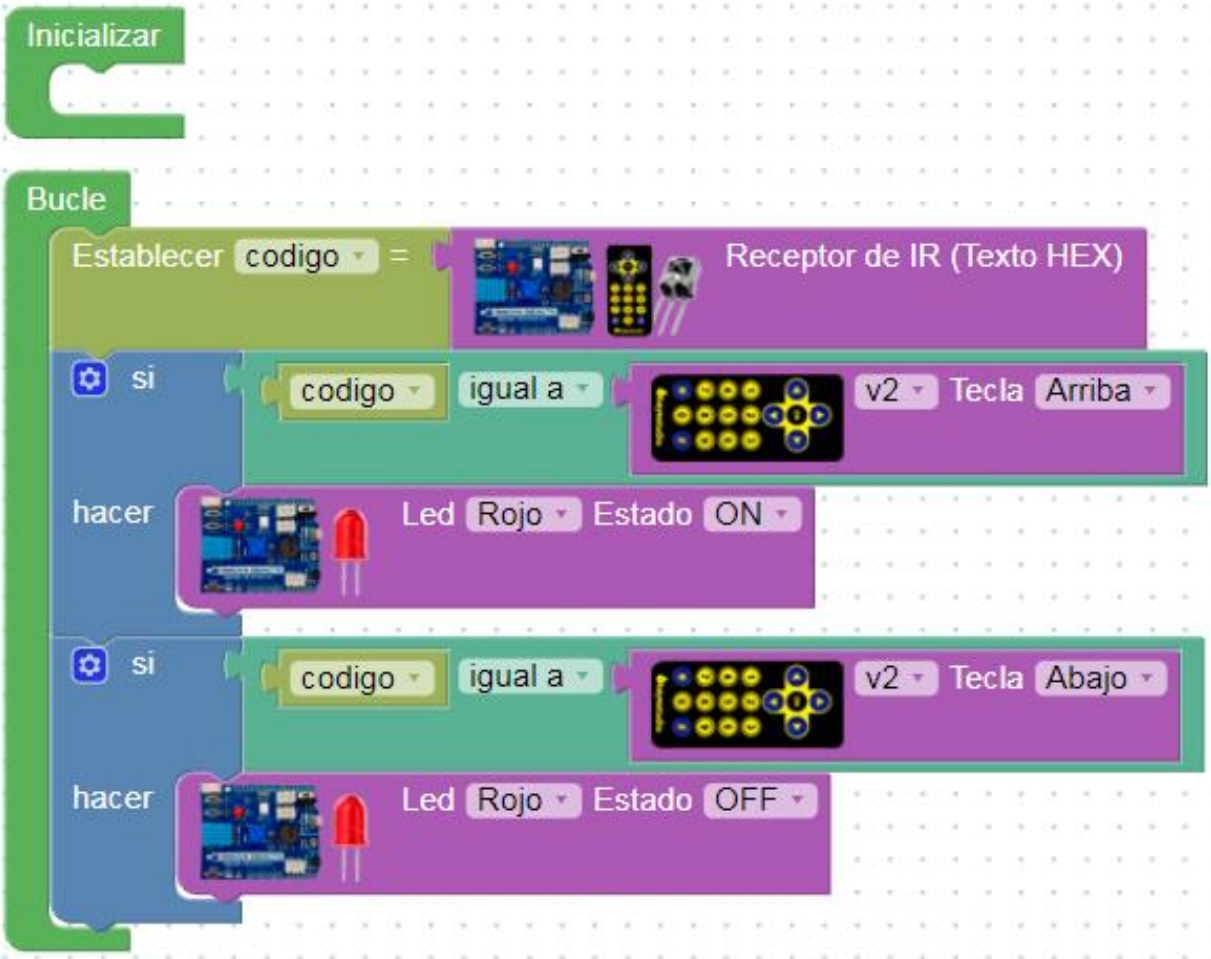
Vamos a realizar una pequeña actividad en la que utilicemos el receptor IR y el mando emisor de Keyestudio. Primero crearemos una variable de texto a la que llamaremos “codigo” (*sin acento, en programación se debe evitar poner acentos y símbolos*) y la estableceremos con el bloque del Receptor IR.

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.

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.



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



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.

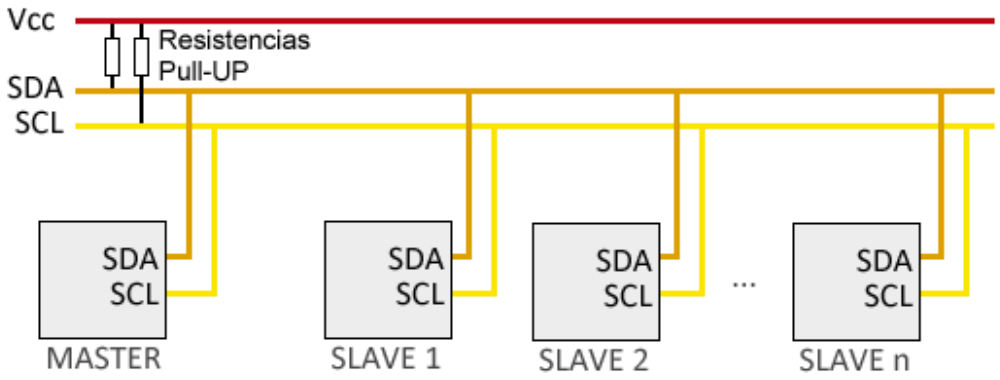
7.10 Reto A10. Puertos de expansión I2C: pantalla LCD.

Reto A10. Puertos de expansión I2C: pantalla LCD.

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.

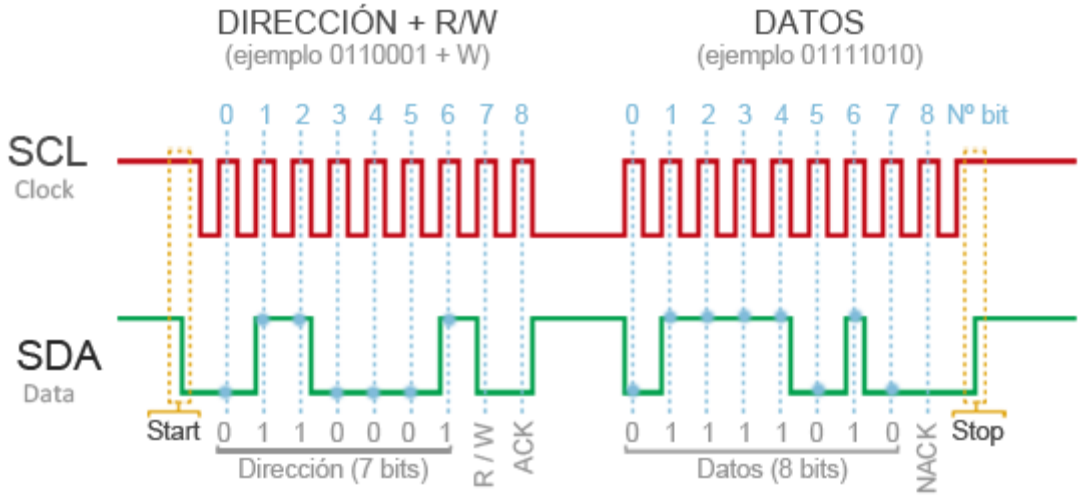


En el bus, cada dispositivo dispone de una dirección, que se emplea para acceder a los dispositivos de forma individual. Esta dirección puede ser fijada por hardware (en cuyo caso, frecuentemente, se pueden modificar los últimos 3 bits mediante “jumpers” o interruptores, o por software.

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)



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 Reto A10.1. Pantalla LCD 16x2.

Reto A10.1. Pantalla LCD 16x2.

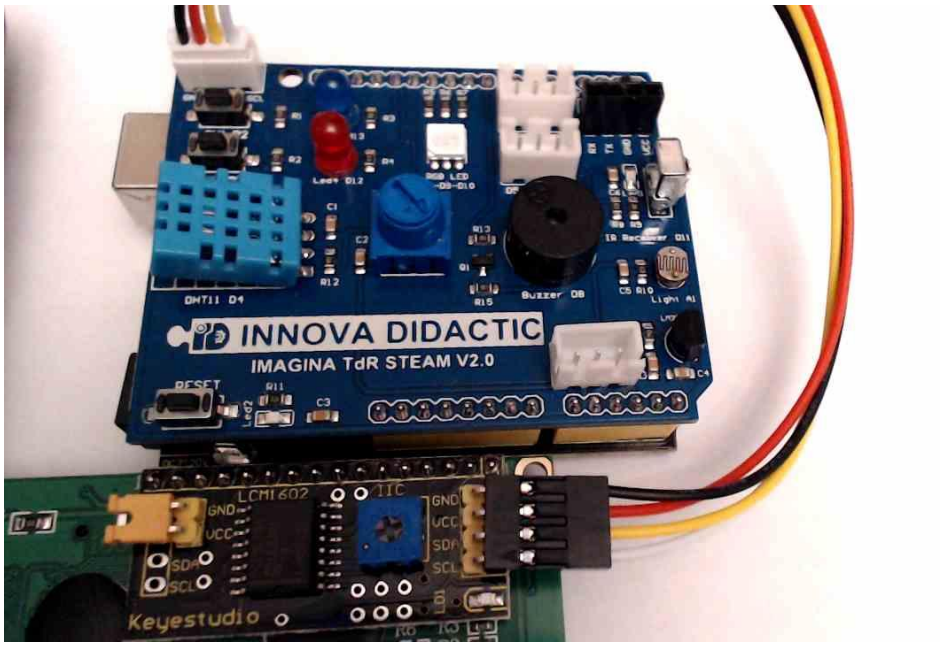
Vamos a realizar la conexión de una pantalla LCD (16x2). La pantalla LCD utilizada es una pantalla de 16 caracteres (por fila) y dos columnas. Esta pantalla tiene 4 conexiones, dos cables (SDA y SCL para el bus de comunicaciones I2C) y los dos cables de alimentación (VCC y GND).

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

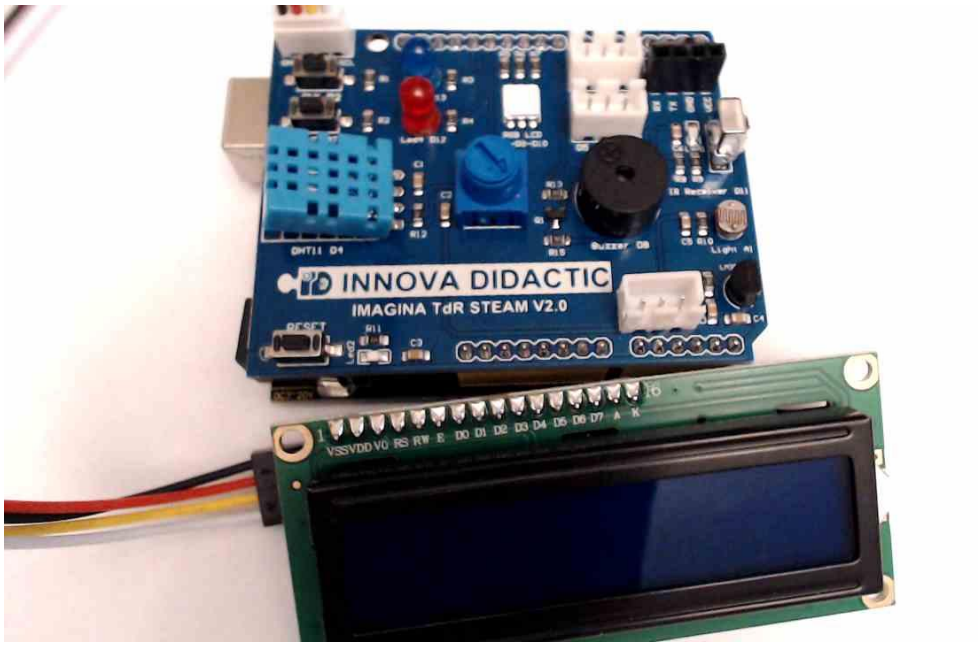
Hemos de tener cuidado y respetar las conexiones, tal y como se indica en la siguiente tabla:

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.

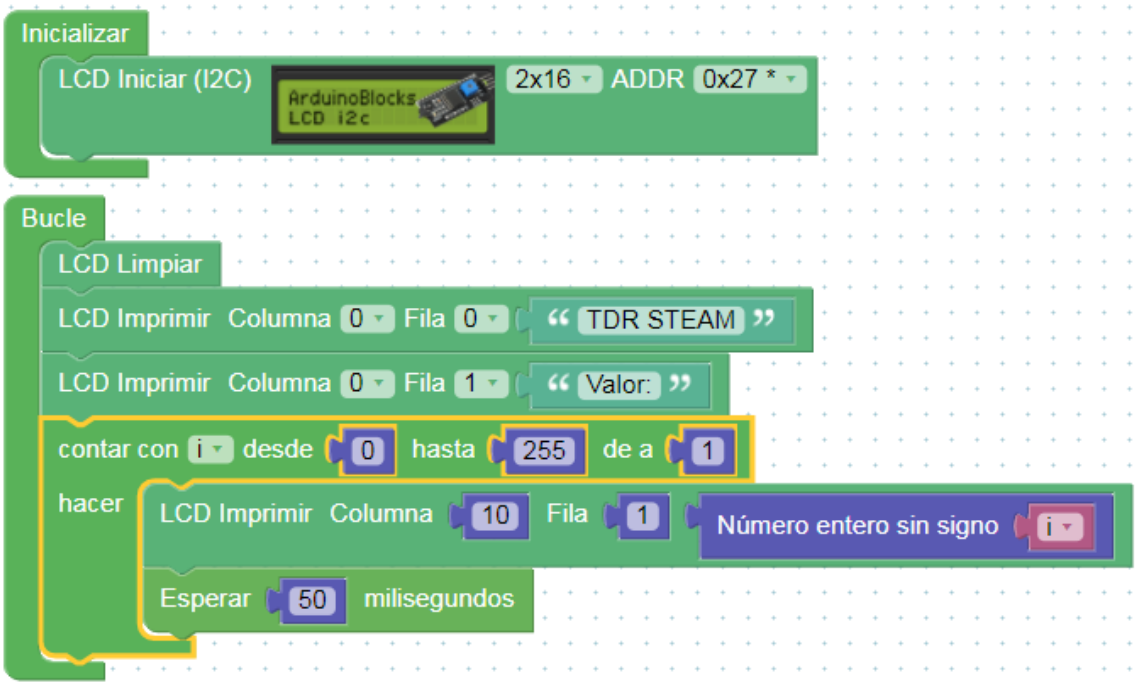


En la pantalla aparecerán correctamente los datos cuando la coloquemos es esta posición, sino se verán al revés:



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.



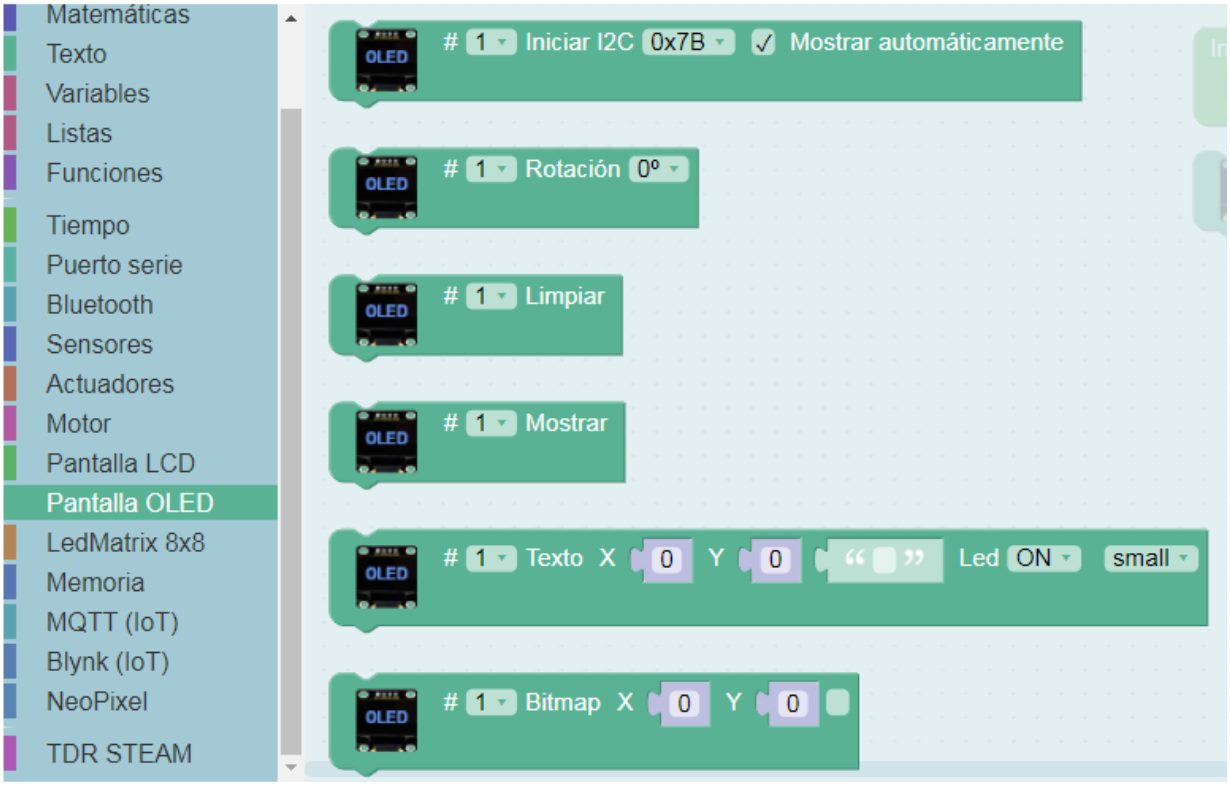
7.10.2 Reto A10.2. La pantalla OLED.

Reto A10.2. La pantalla OLED.

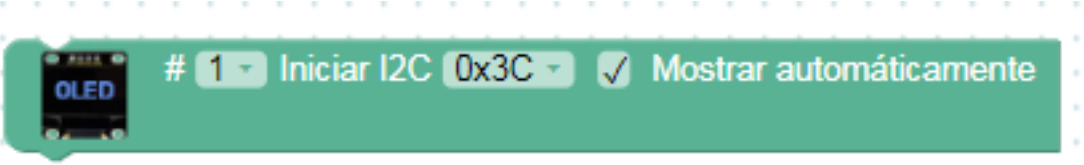
La pantalla OLED la conectaremos también en el puerto de comunicaciones I2C respetando las conexiones.

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.

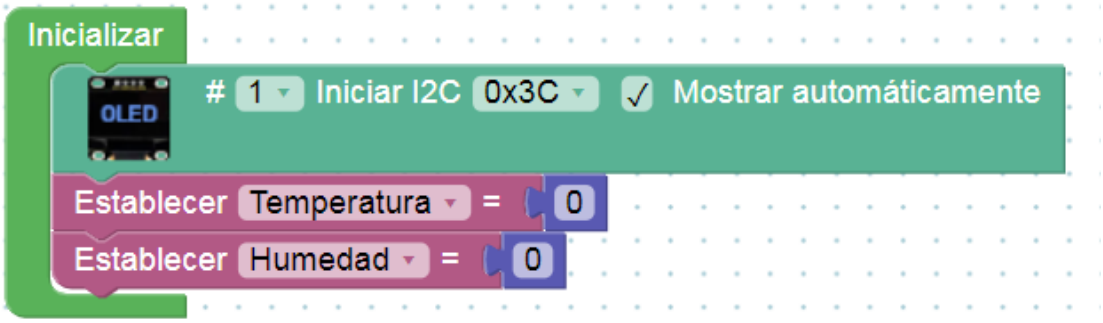
Vamos a realizar un programa para poder ver los datos del sensor de humedad y temperatura por la pantalla OLED. Existen una serie de bloques específicos para controlar la pantalla.



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



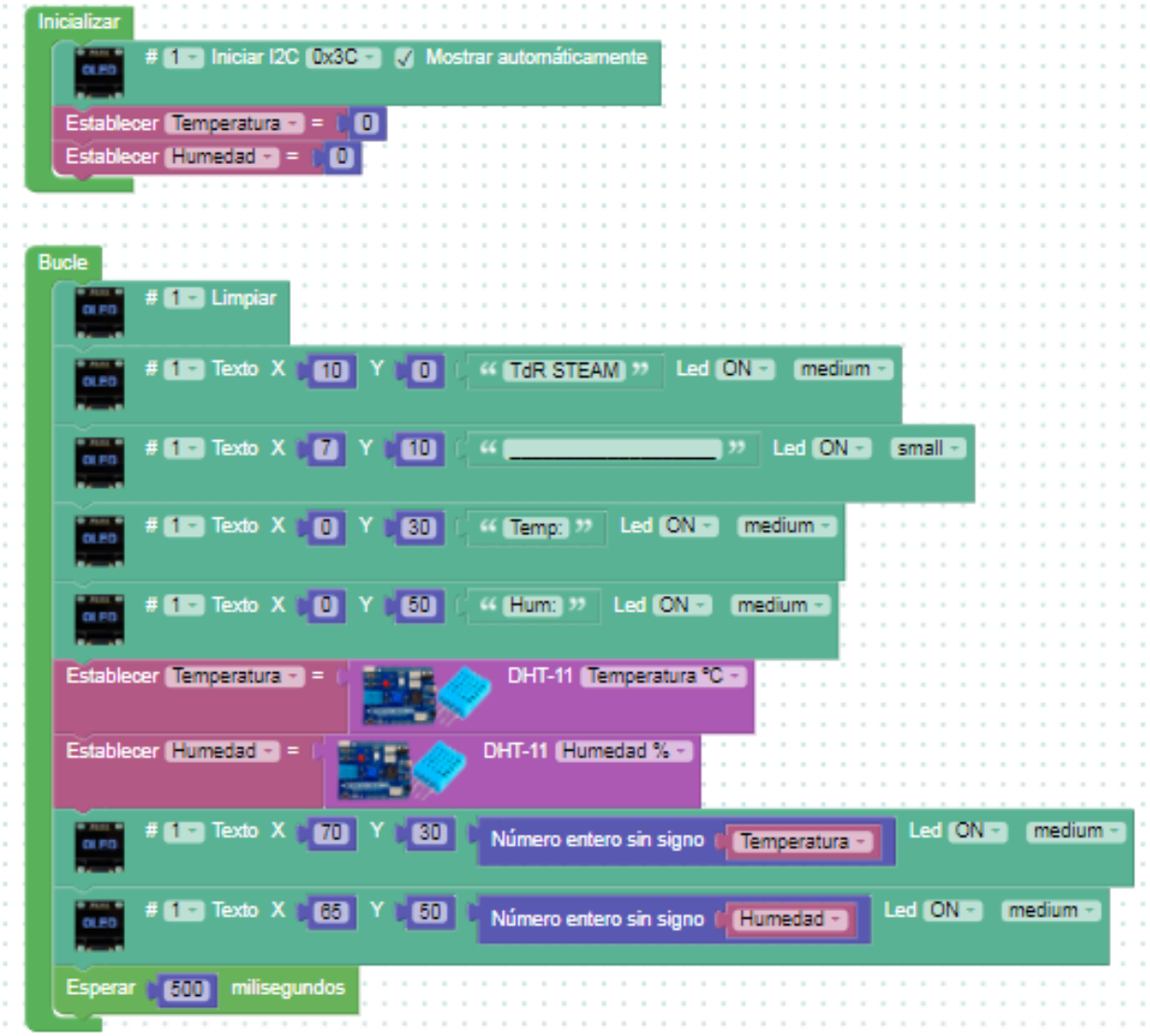
Colocamos el bloque dentro de Inicializar y creamos las dos variables (Temperatura y Humedad).



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:

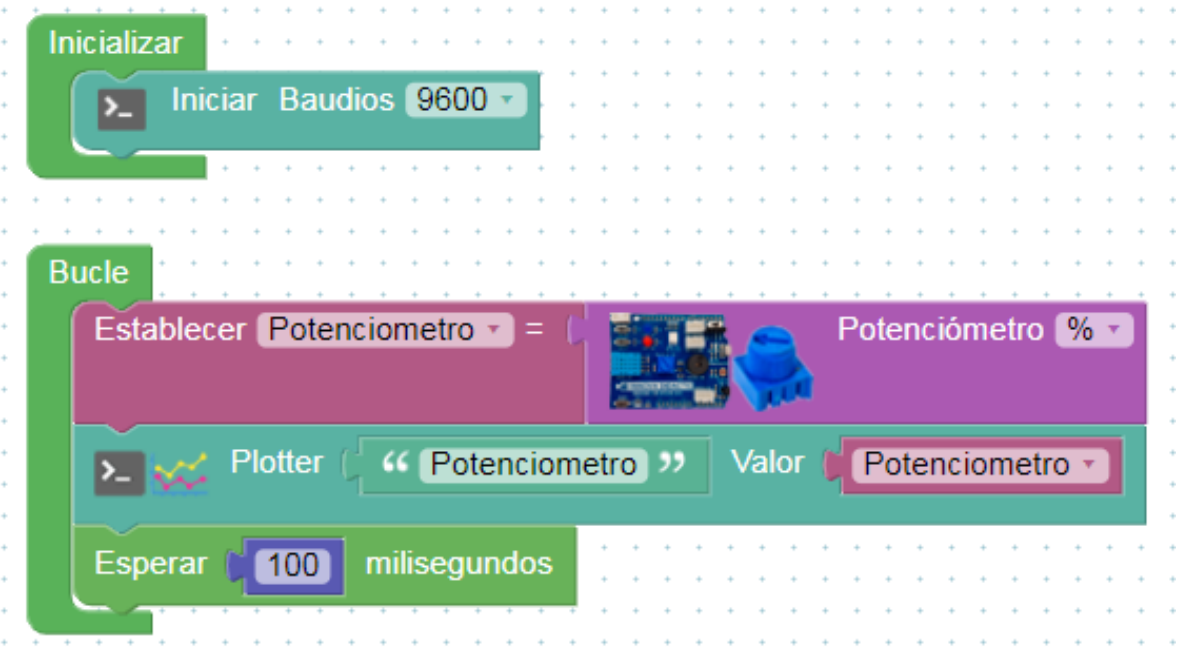


7.11 Reto A11.1. Serial plotter.

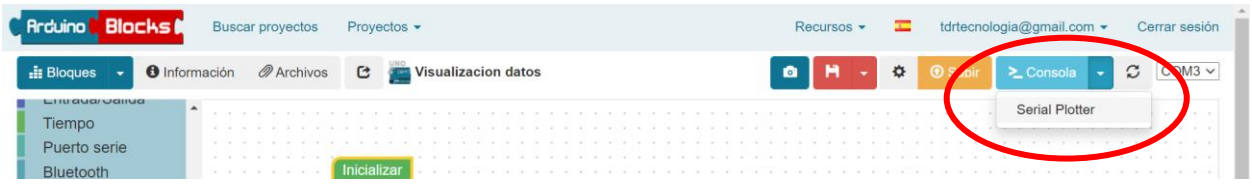
Reto A11.1. Serial plotter.

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.

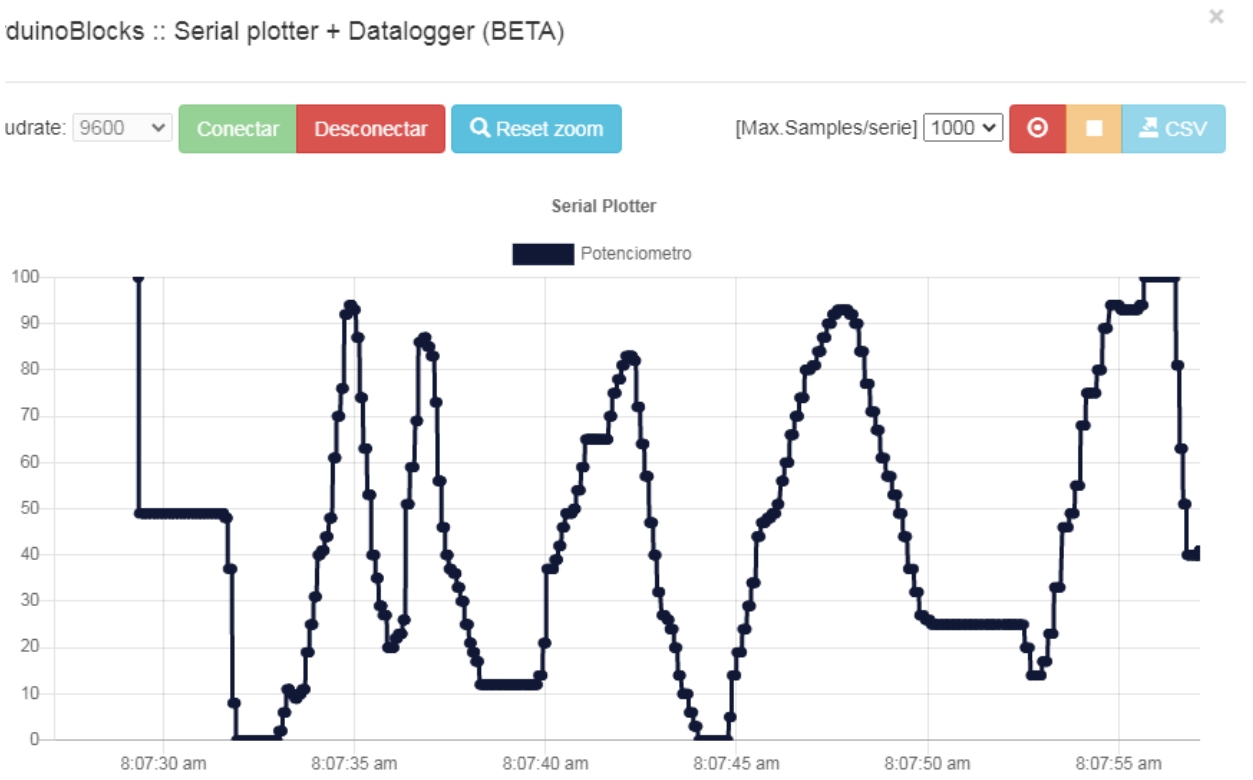
Este es el programa en ArduinoBlocks que hemos confeccionado.



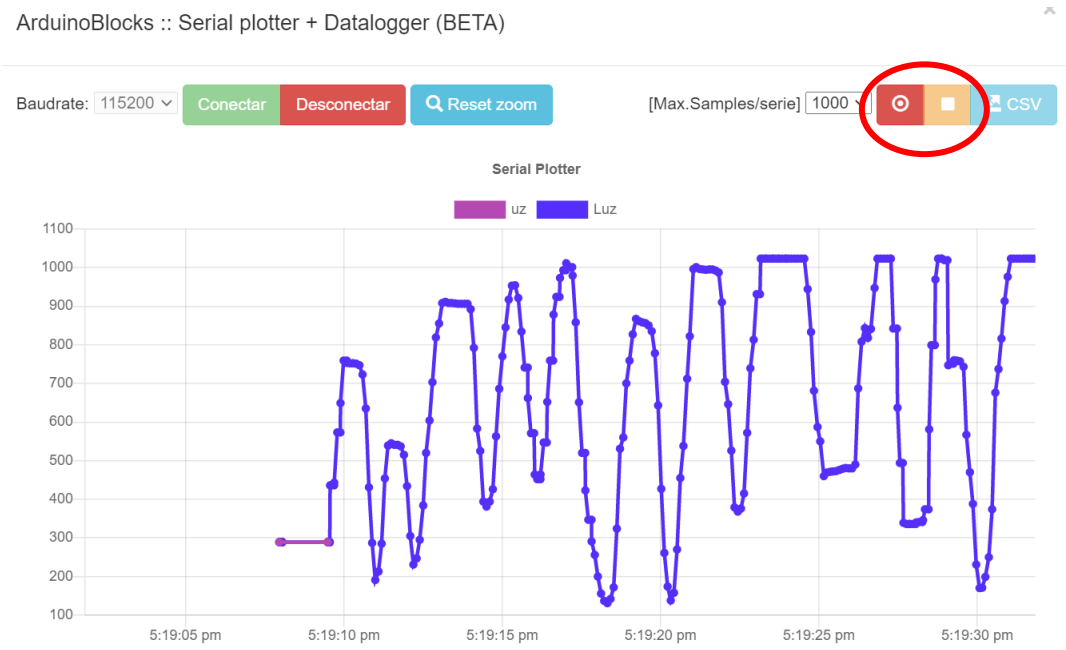
Enviamos el programa a la placa y activamos el Serial Plotter.



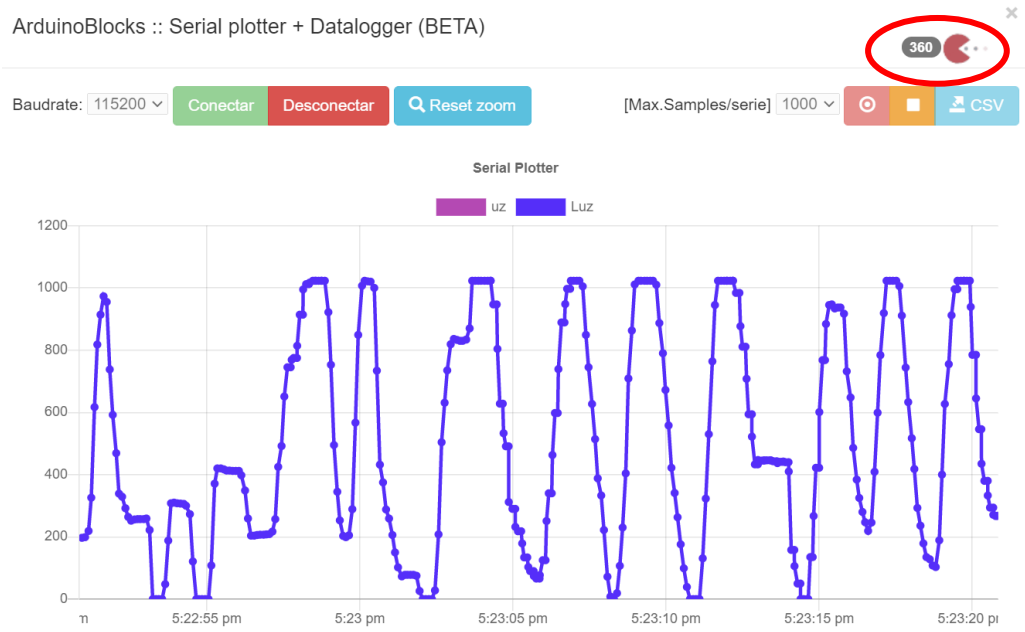
Pondremos la velocidad de comunicación (baudrate) a 9600 y después pulsaremos ***Conectar*** para empezar a ver los datos.



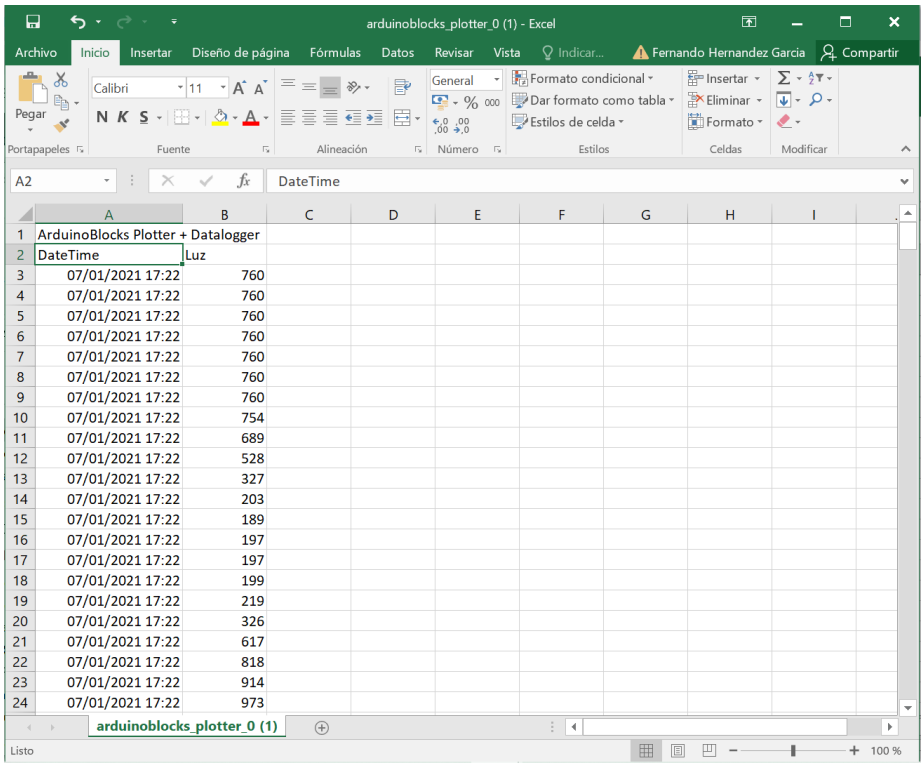
Para poder guardar los datos en CSV hemos de apretar el botón de grabación, adquirir los datos que queremos y apretar el botón de parar grabación.



Podremos ver la cantidad de muestras recogidas.



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



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

7.12 Sistemas de comunicaciones: Bluetooth y Wifi.

En la placa TDR STEAM disponemos de un puerto de comunicaciones serie que nos permite conectar módulos Bluetooth o Wifi. Además, dispone de un interruptor para poder conectarlos o desconectarlos ya que utiliza los mismos pines Rx/Tx que para comunicarse con el ordenador).

Estado del interruptor:

- OFF: comunicación con el ordenador.
- ON: comunicación con los diferentes módulos Bluetooth o Wifi.

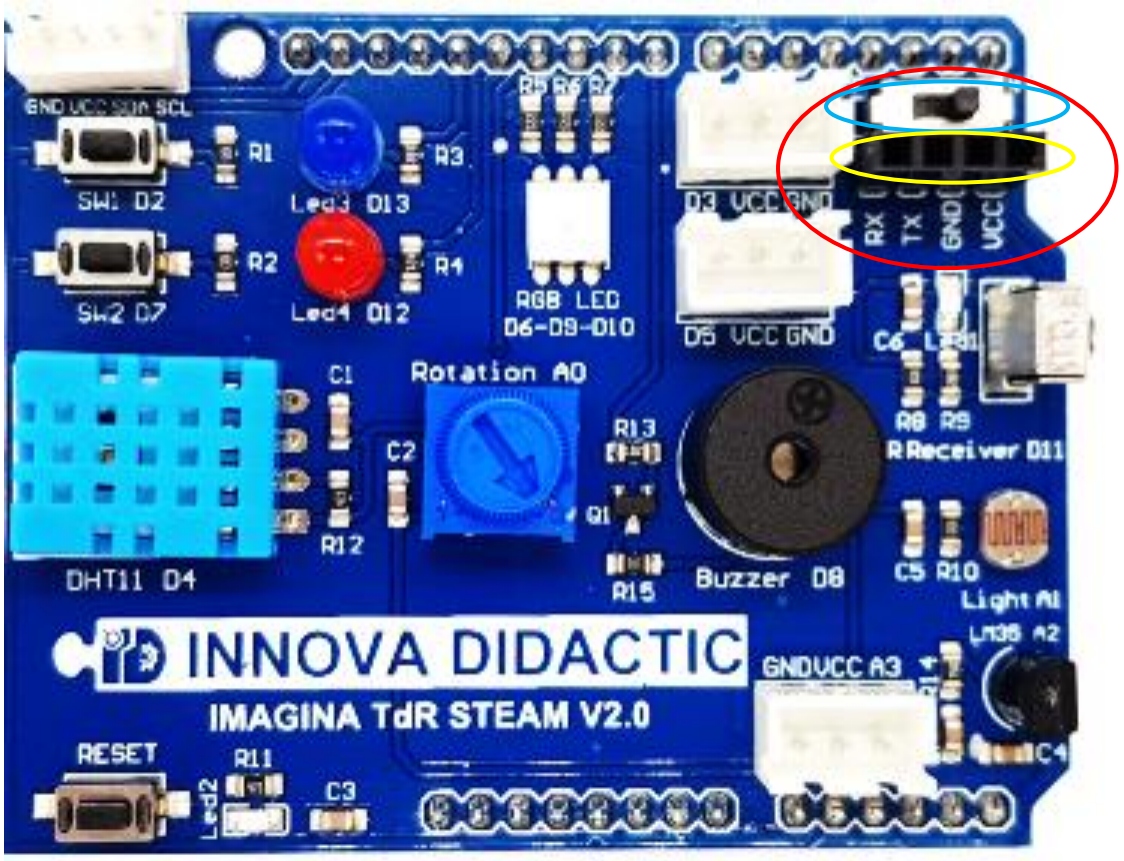
7.12.1 Reto A12.1. Módulo Bluetooth.

Reto A12.1. Módulo Bluetooth.

Para poder hacer la comunicación por Bluetooth con nuestra placa Imagina TDR STEAM hemos de conectar un módulo Bluetooth (en nuestro caso utilizaremos un módulo HC-06).

Primero conectaremos el módulo Bluetooth al conector marcado de nuestra placa. Hemos de tener en cuenta la posición del interruptor:

- OFF: transmisión del programa desde ArduinoBlocks.
- ON: funcionamiento en modo Bluetooth.



Necesitaremos dos programes pera poder trabajar:

- ArduinoBlocks: programa que funcionar en la placa Keyestudio.
- AppInventor2: aplicación que funcionará en el teléfono móvil.

7.12.1.1 Reto A12.1.1. AppInventor2.

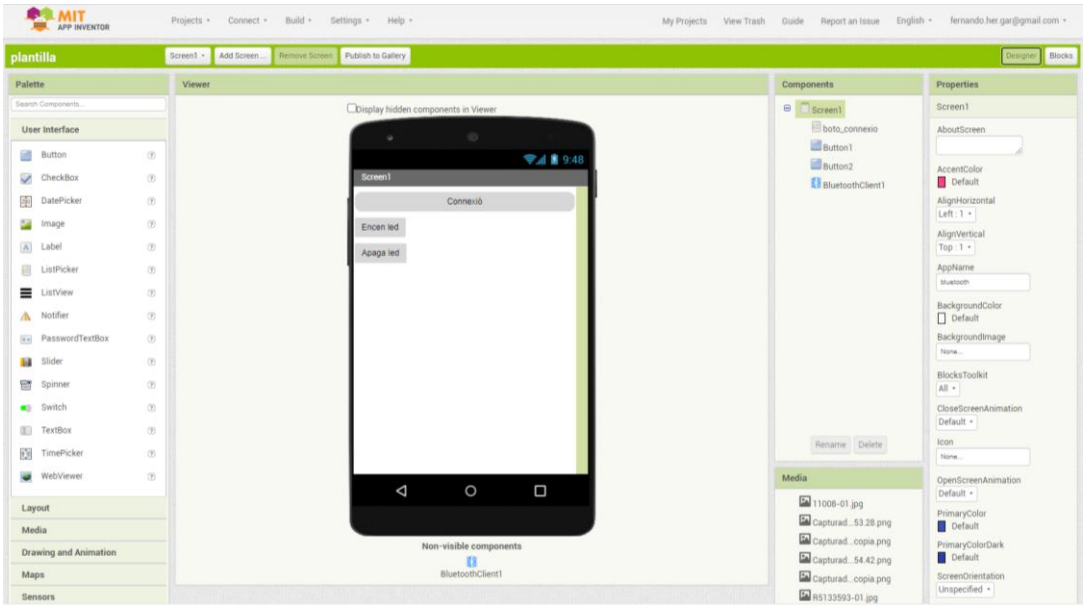
Reto A12.1.1. AppInventor2.

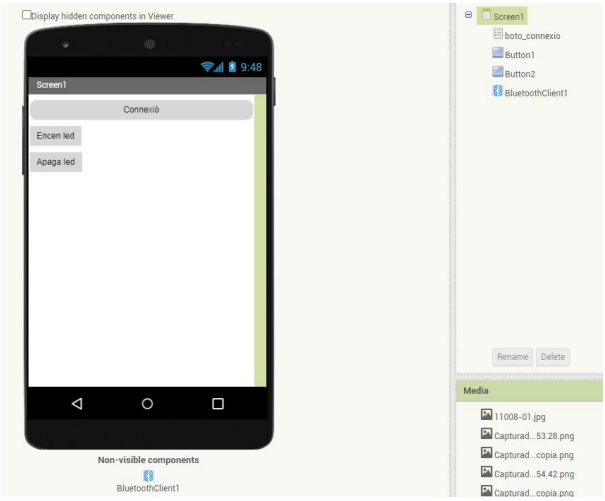
Primero deberemos crear una cuenta para AppInventor2 a través del siguiente enlace:

http://ai2.appinventor.mit.edu/

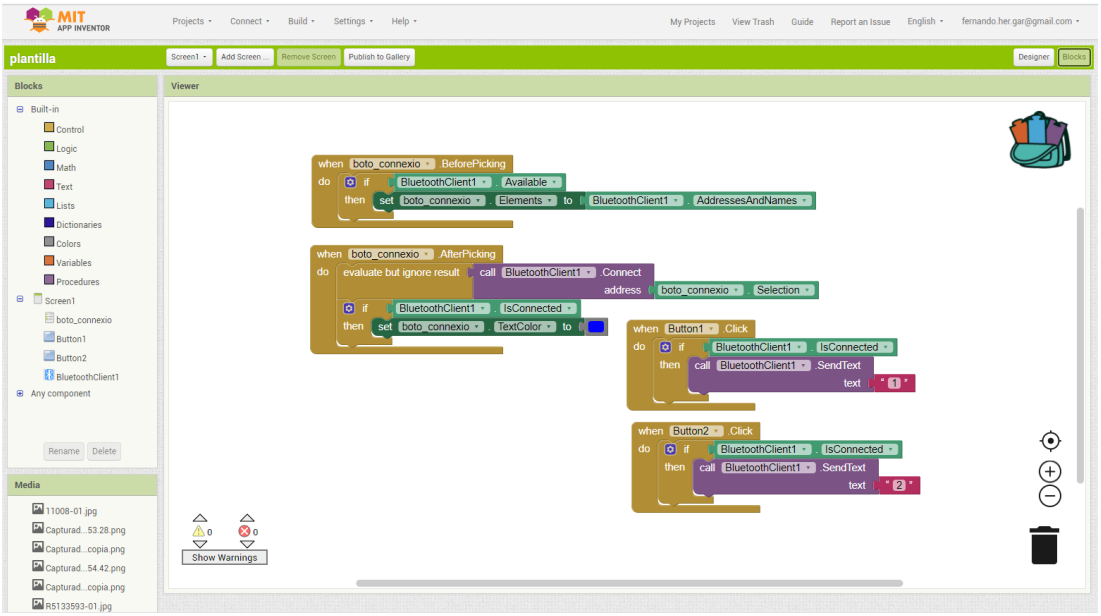
Una vez creada la cuenta procederemos a realizar la programación, tanto en “Designer” como en “Blocks”. A continuación, se detallan las dos partes del programa.

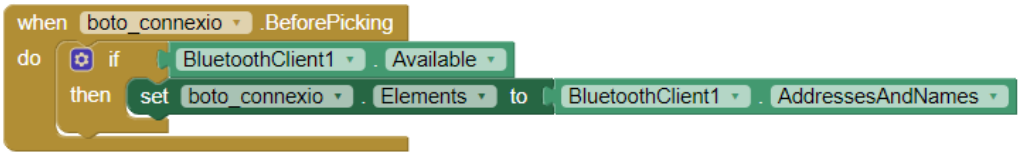
- Designer:

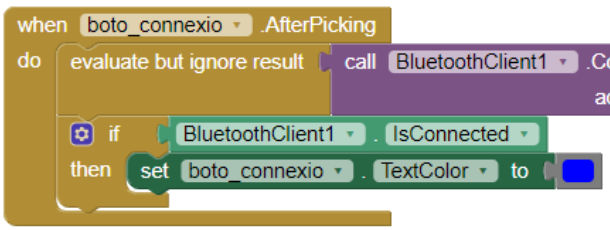


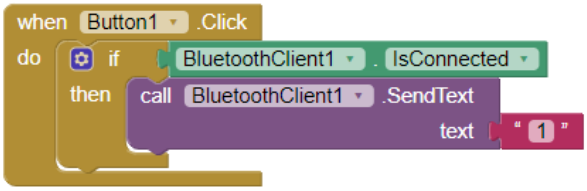


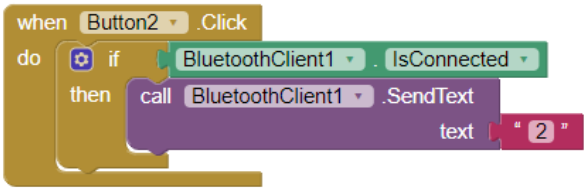
- Blocks

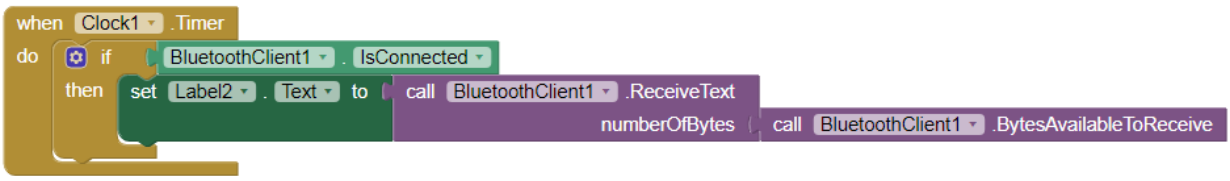

MIT APP INVENTOR
plantilla
Blocks
Viewer
when boto_connexio .BeforePicking
do if BluetoothClient1 . Available
then set boto_connexio . Elements to BluetoothClient1 . AddressesAndNames
when boto_connexio .AfterPicking
do evaluate but ignore result call BluetoothClient1 .Connect
address boto_connexio . Selection
if BluetoothClient1 . IsConnected
then set boto_connexio . TextColor to
when Button1 .Click
do if BluetoothClient1 . IsConnected
then call BluetoothClient1 .SendText
text "1"
when Button2 .Click
do if BluetoothClient1 . IsConnected
then call BluetoothClient1 .SendText
text "2"
Show Warnings


when boto_connexio .BeforePicking
do if BluetoothClient1 . Available
then set boto_connexio . Elements to BluetoothClient1 . AddressesAndNames


when boto_connexio .AfterPicking
do evaluate but ignore result call BluetoothClient1 .Connect
address boto_connexio . Selection
if BluetoothClient1 . IsConnected
then set boto_connexio . TextColor to


when Button1 .Click
do if BluetoothClient1 . IsConnected
then call BluetoothClient1 .SendText
text "1"

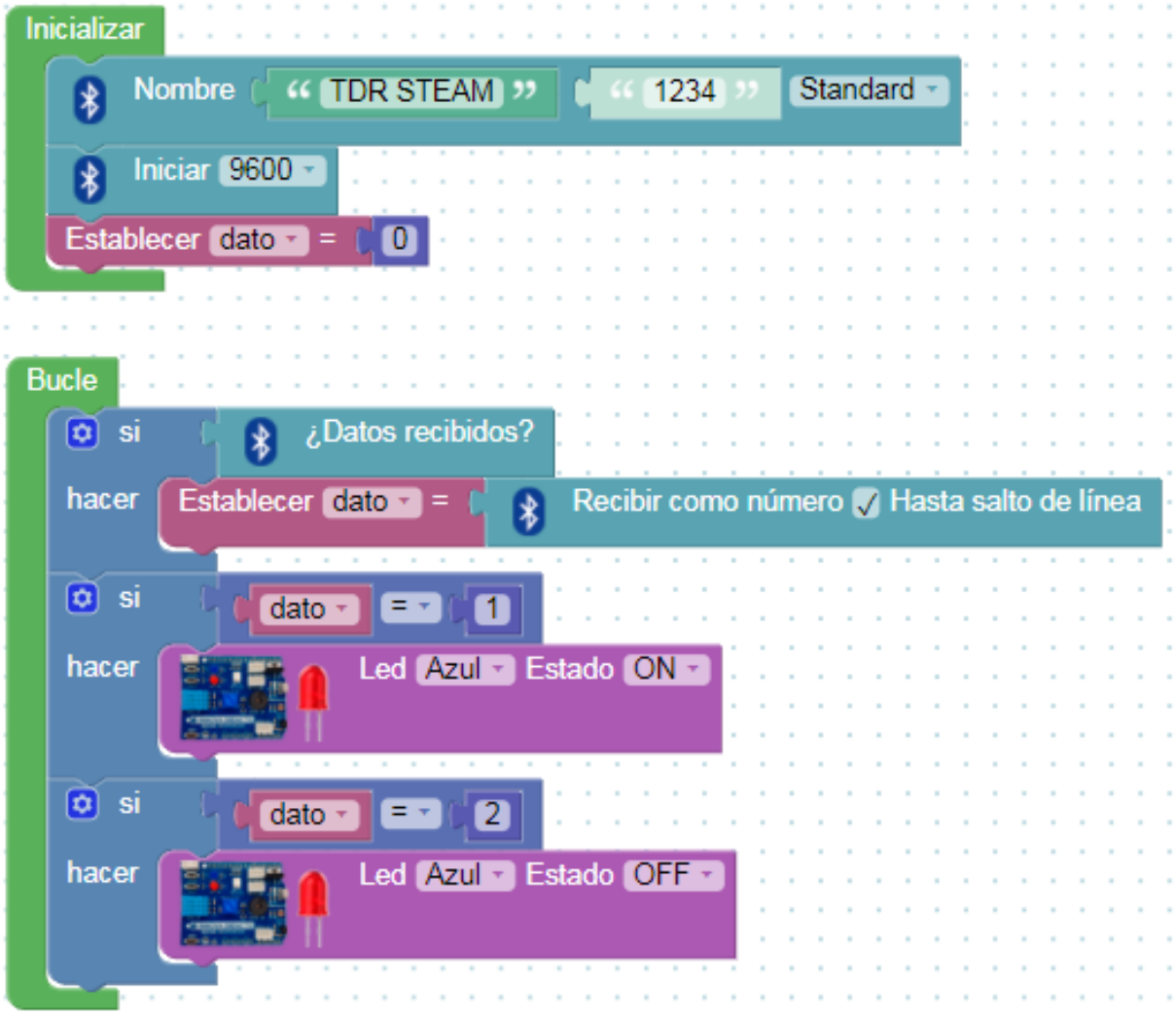

when Button2 .Click
do if BluetoothClient1 . IsConnected
then call BluetoothClient1 .SendText
text "2"


when Clock1 .Timer
do if BluetoothClient1 . IsConnected
then set Label2 . Text to call BluetoothClient1 .ReceiveText
numberOfBytes call BluetoothClient1 .BytesAvailableToReceive

7.12.1.2 Reto A12.1.2. ArduinoBlocks.

Reto A12.1.2. ArduinoBlocks.

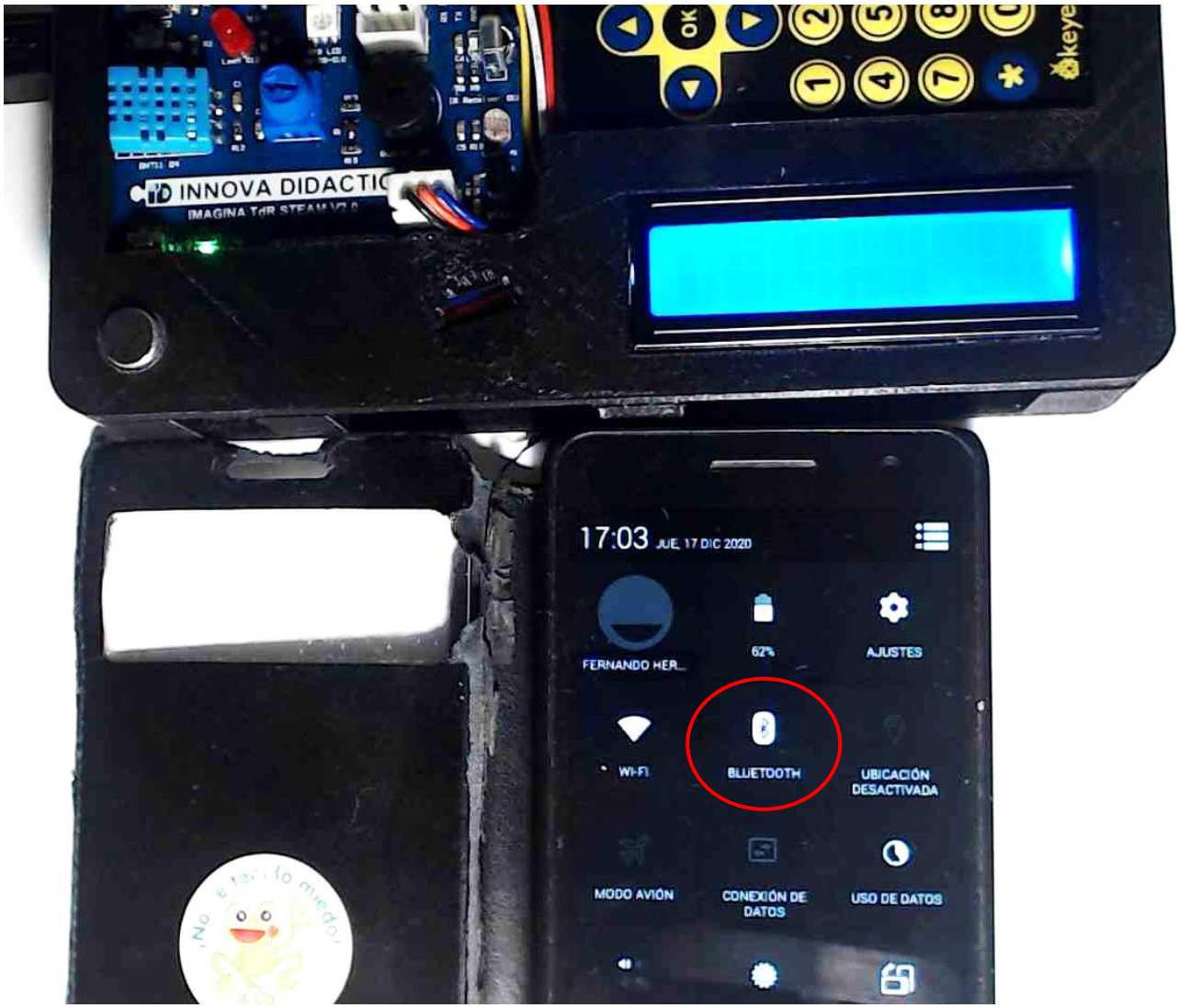
A continuación, procederemos a realizar el programa en ArduinoBlocks y que irá en la placa Imagina TDR STEAM.



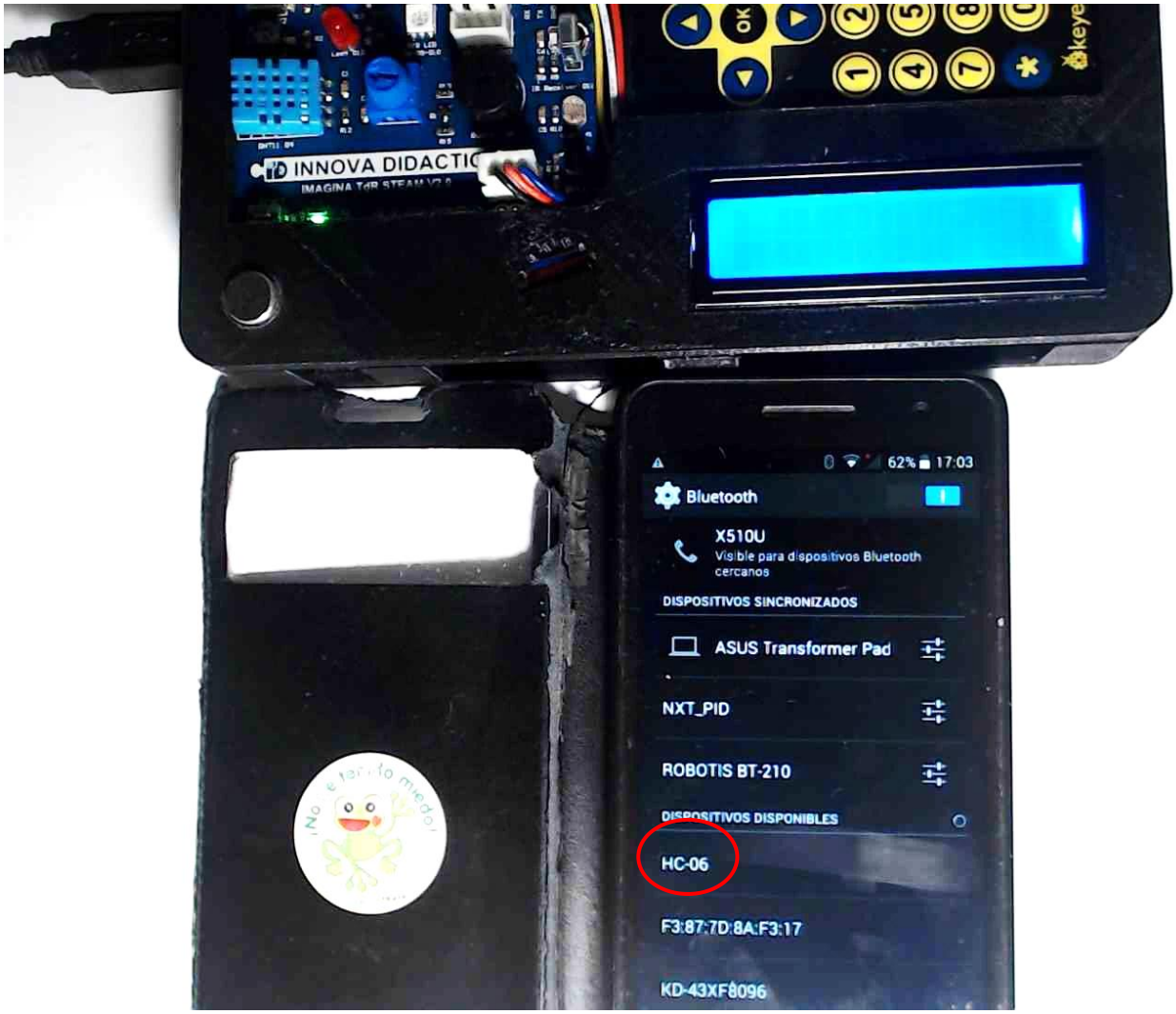
Cuando tenemos preparados los dos programas, seguiremos los siguientes pasos:

- Enviar el programa de ArduinoBlocks a la placa Keyestudio (interruptor en la posición OFF).
- Activar el módulo Bluetooth mediante el interruptor y ponerlo en la posición ON.
- Instalar la aplicación en el teléfono móvil.
- Sincronizar el Bluetooth del móvil con el módulo Bluetooth de la placa Keyestudio.
- Probar el programa.

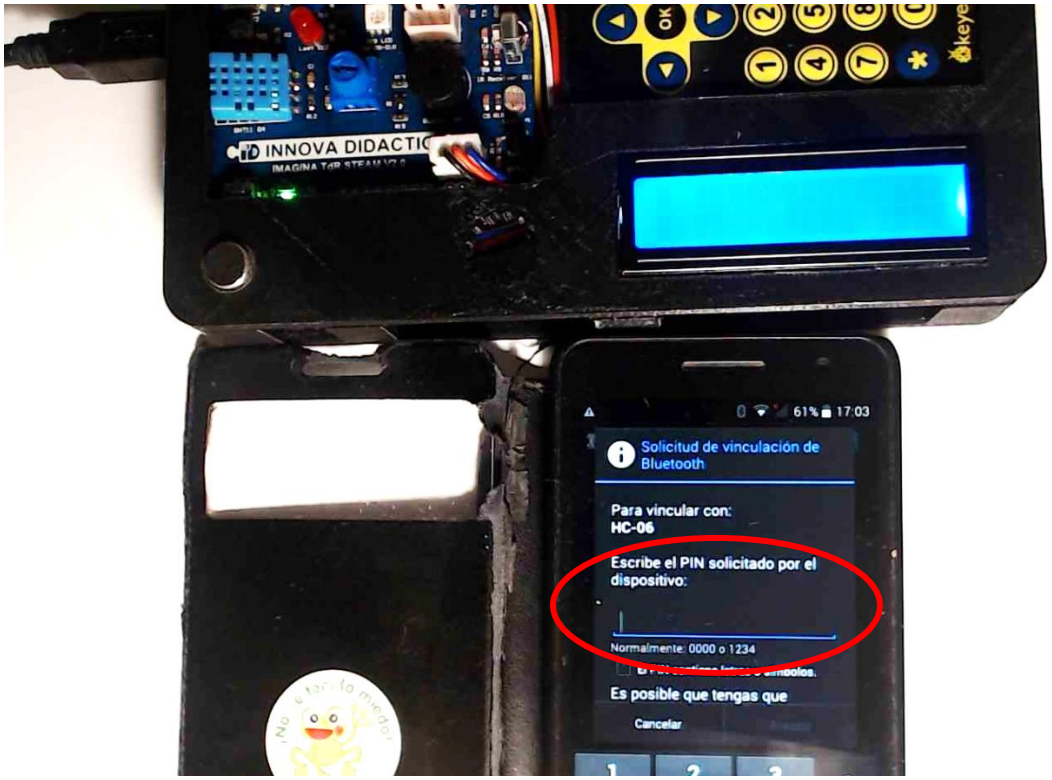
Activamos el módulo Bluetooth del teléfono móvil.

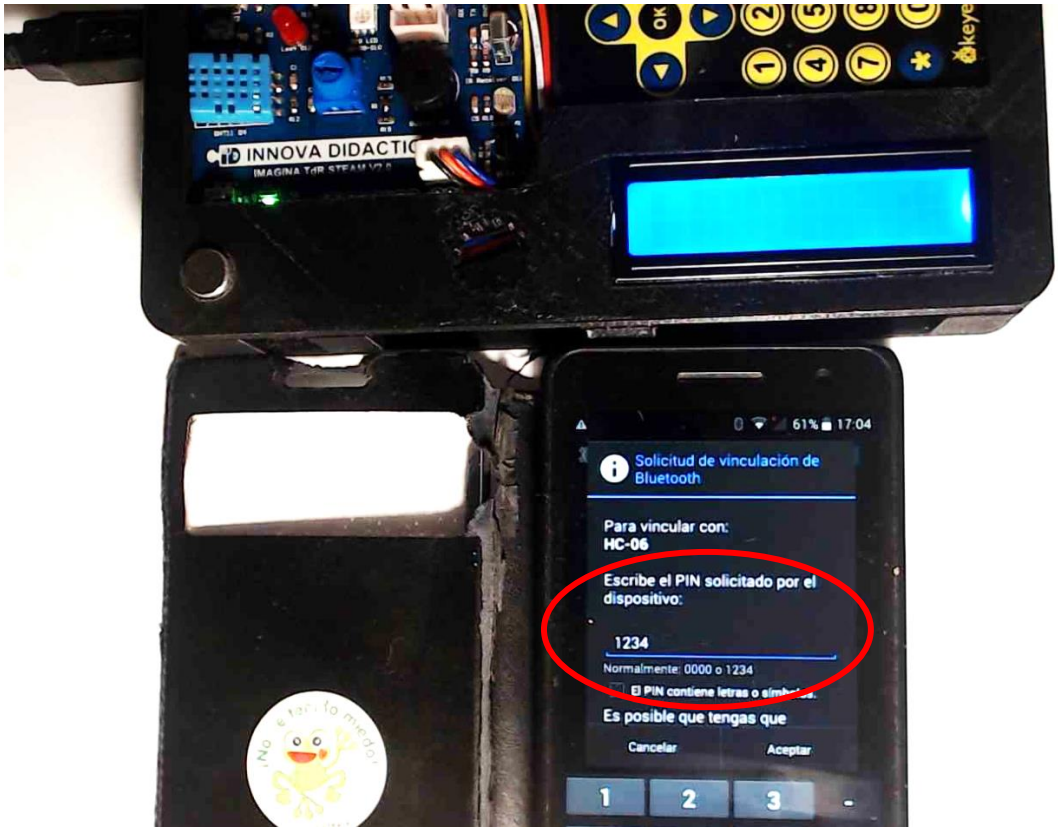


Abrimos el servicio Bluetooth del móvil y buscamos el nuevo dispositivo. Aparecerá un dispositivo llamado **HC-06**.

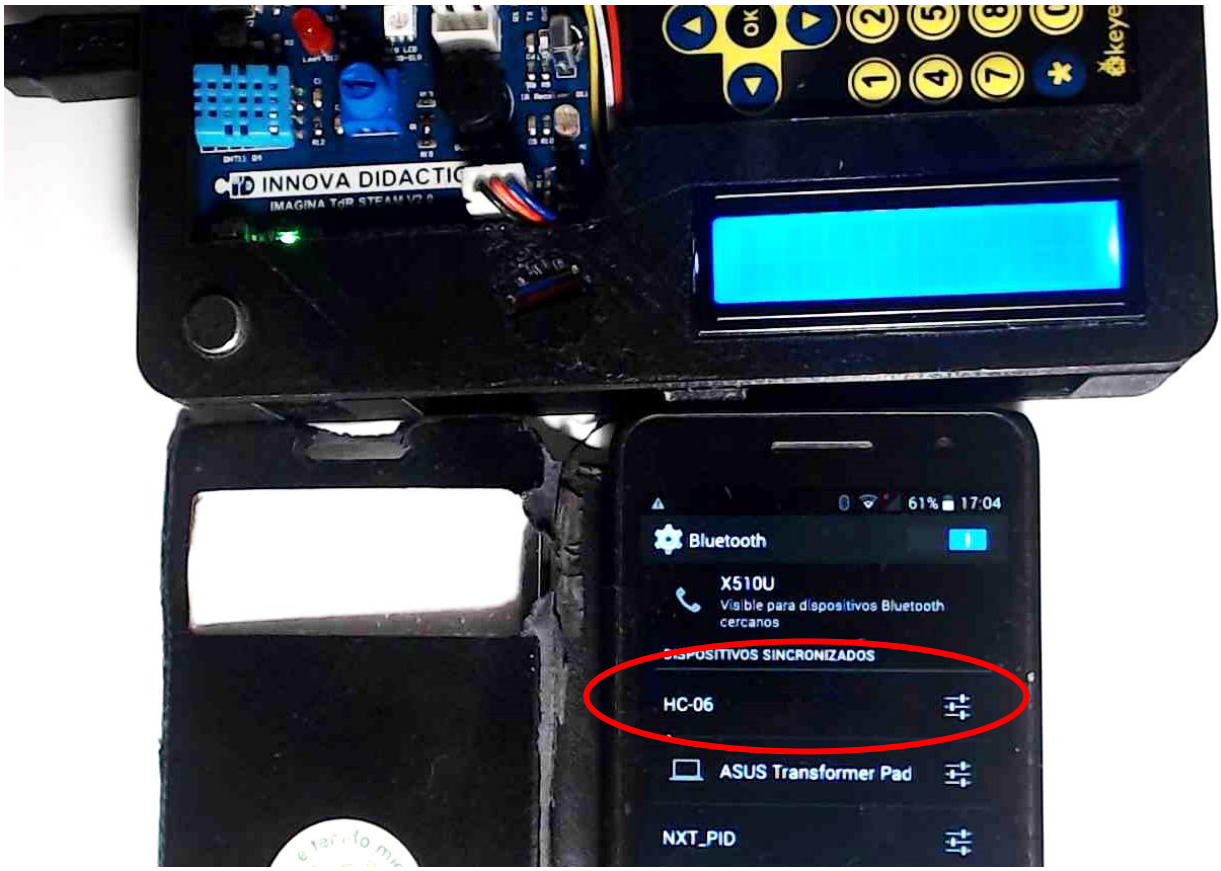


Elegimos este módulo e introducimos la contraseña: 1234 (opcionalmente puede ser 0000).

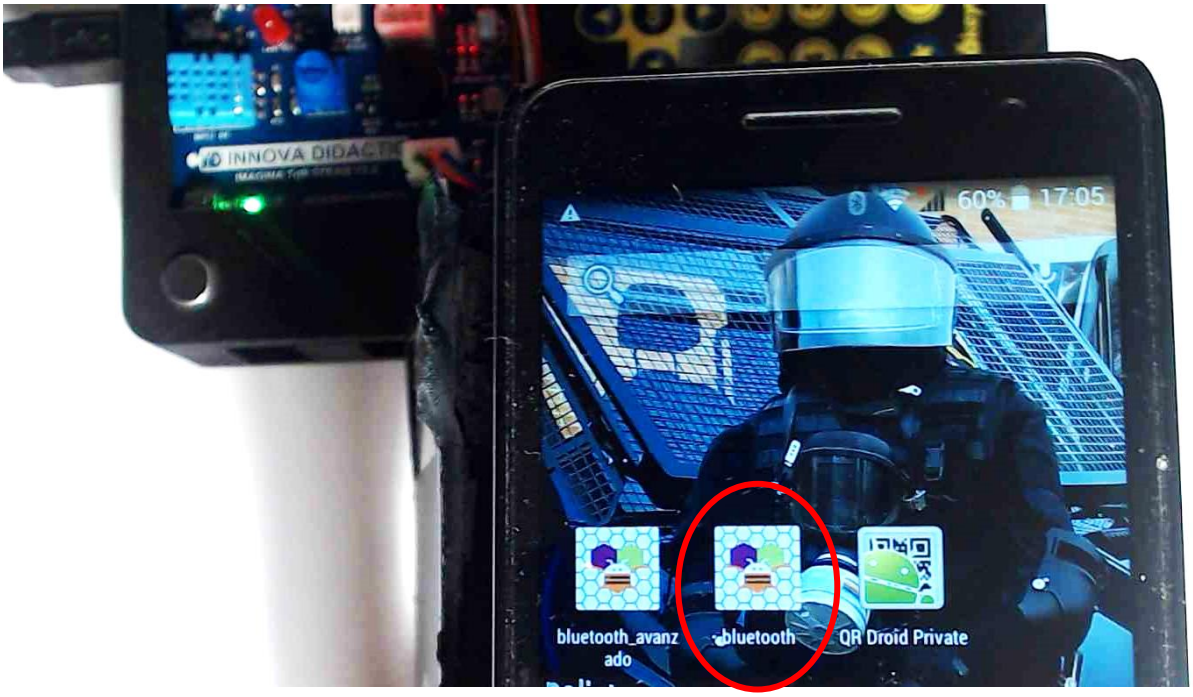




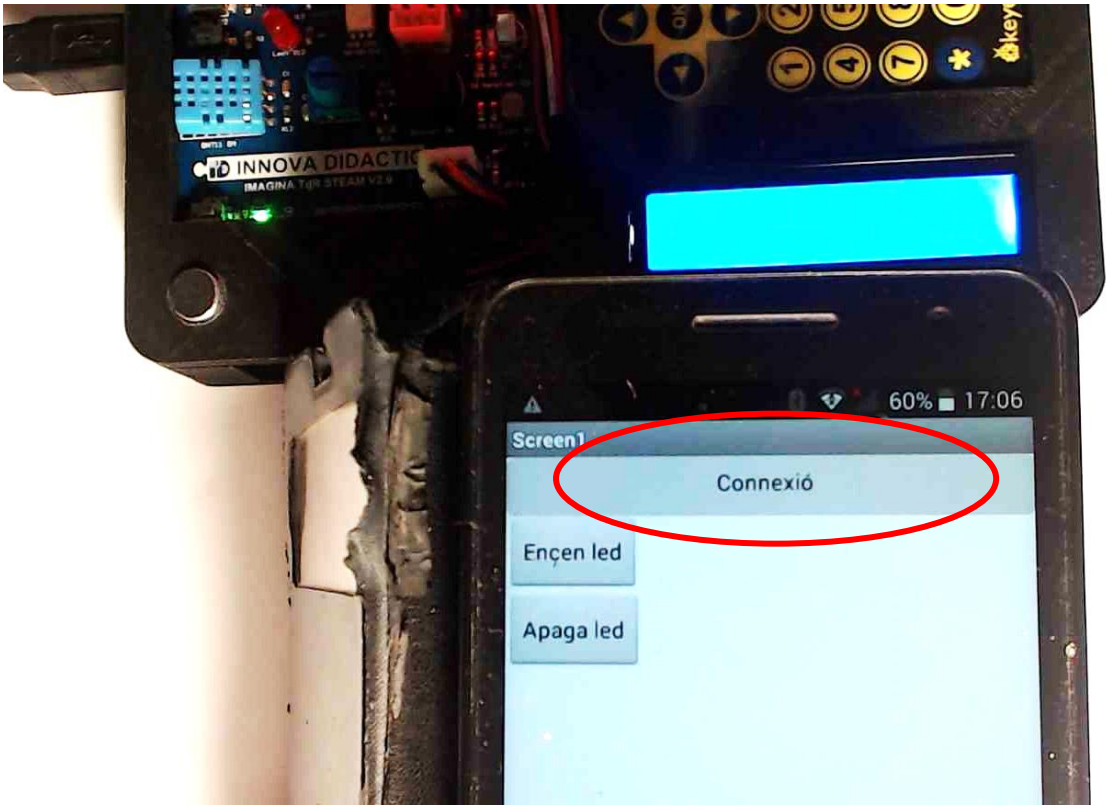
Ahora ya tenemos sincronizados los dos dispositivos Bluetooth.



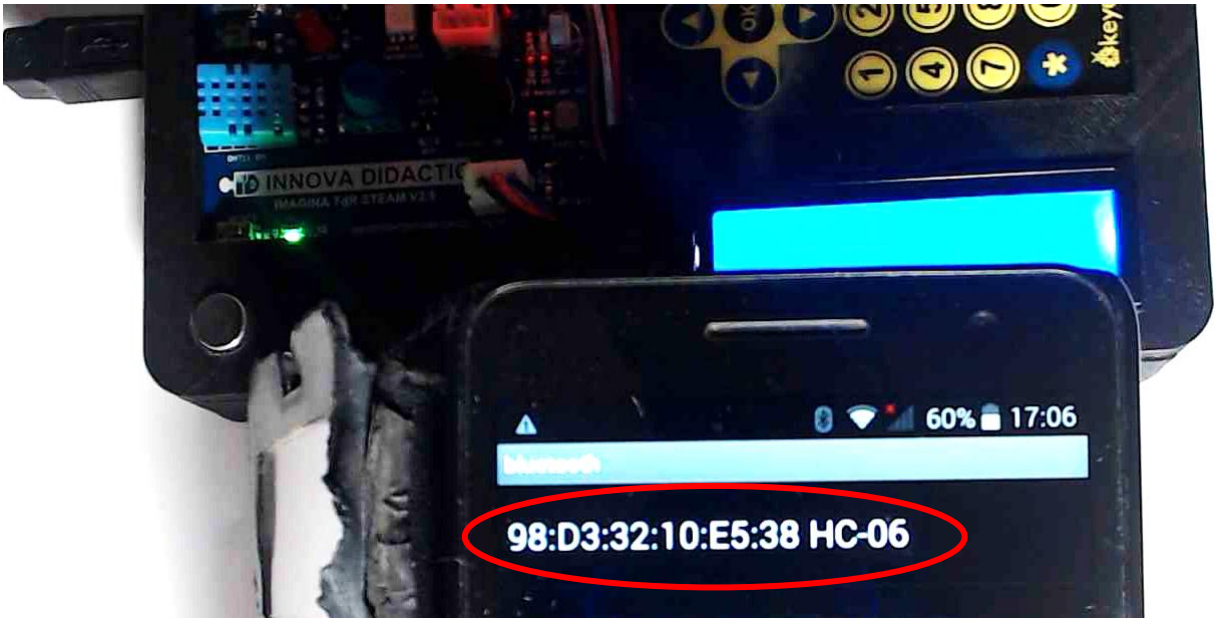
A continuación, abrimos la aplicación (en este caso se llama “Bluetooth).



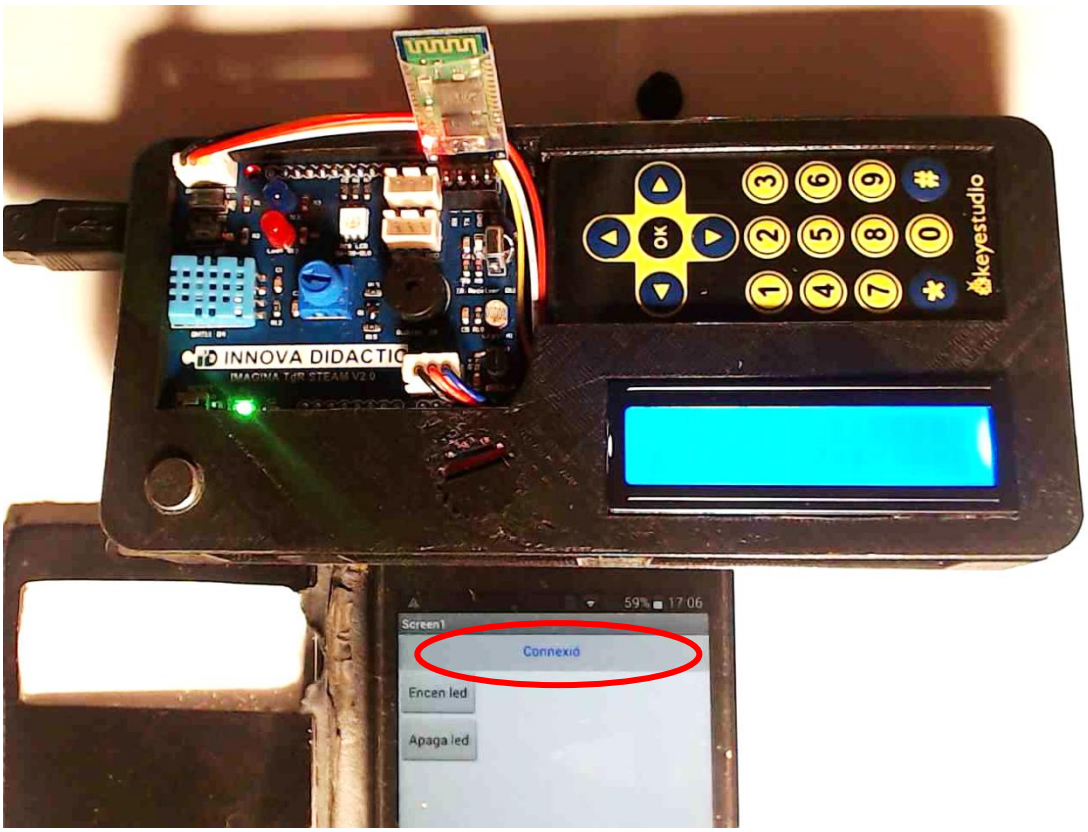
Apretamos el botón *Connexió* para poder hacer la conexión (está en letras de color negro). Recordamos que el interruptor ahora está en la posición *ON* en la placa Imagina TDR STEAM.



Seleccionamos nuestro módulo Bluetooth (mediante la dirección MAC):

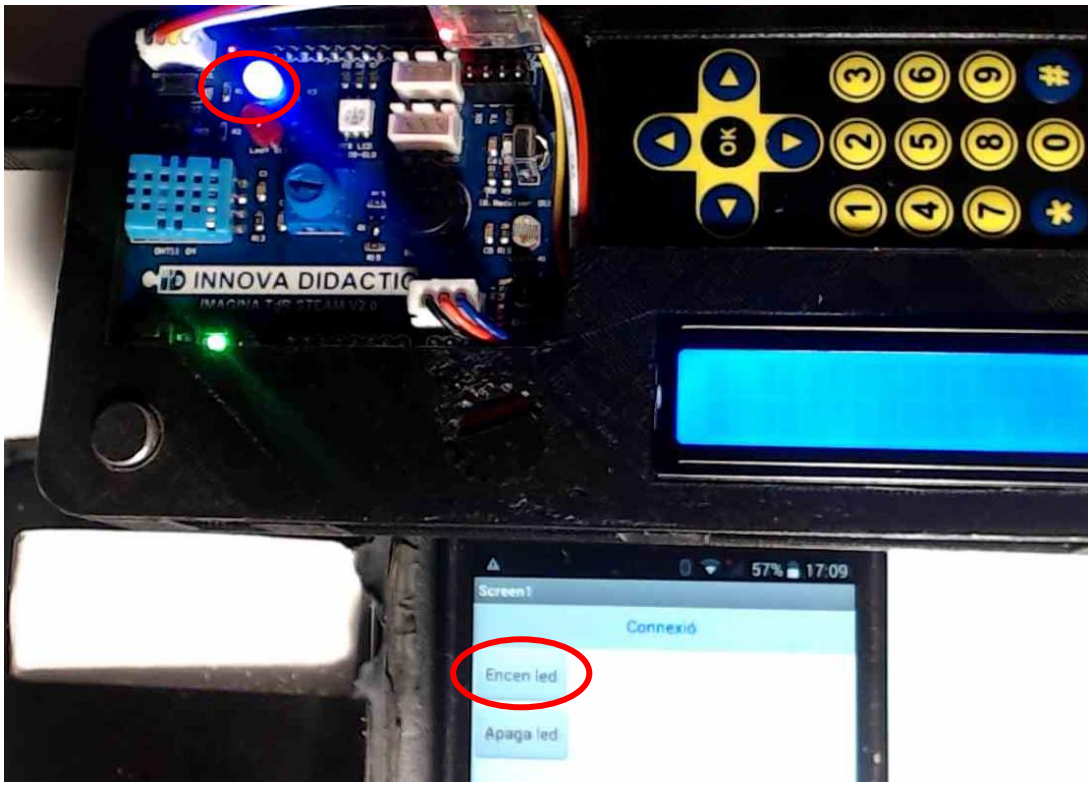


Ahora el botón *Connexió* aparecerá con las letras de color azul.

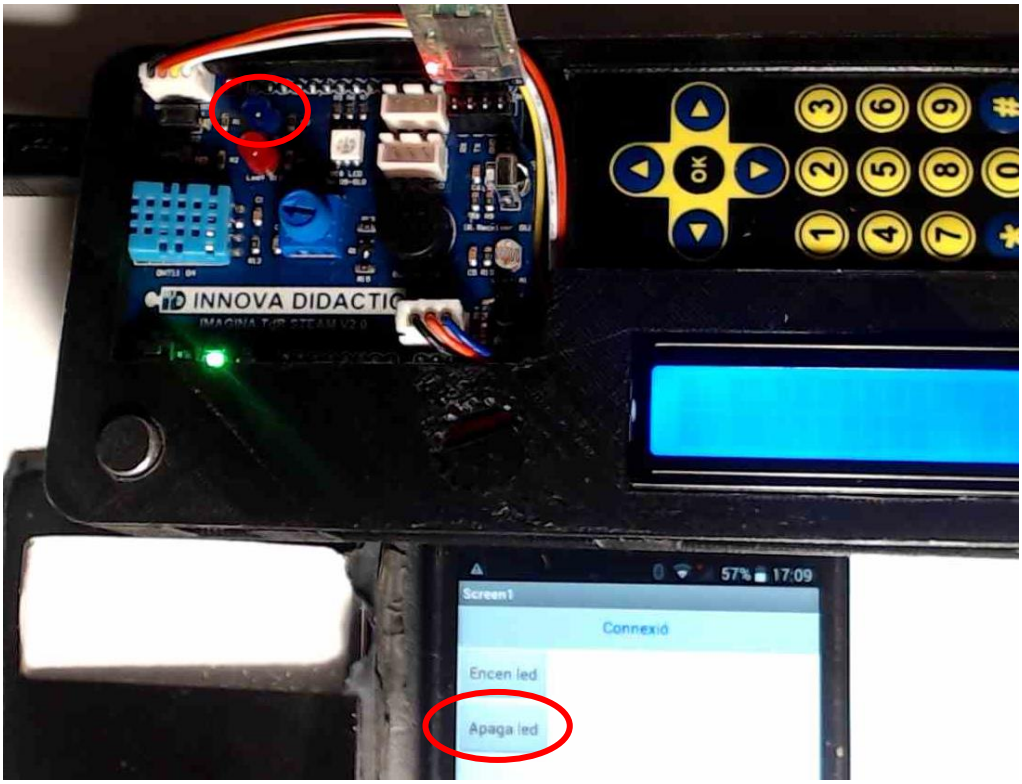


Y la luz del módulo Bluetooth de la placa Imagina TDR STEAM quedará fija en el color rojo.

Si apretamos el botón *Encen led* se encenderá el led azul.



Si apretamos el botón *Apaga led* se apagará el led azul.



7.12.2 Reto A12.2. Módulo Wifi.

Reto A12.2. Módulo Wifi.

Para poder realizar la comunicación Wifi hemos de realizar unos pasos similares a la comunicación Bluetooth.

Para poder visualizar los datos enviados desde la placa Imagina TDR STEAM utilizaremos el programa ThingSpeak y la aplicación ThingView. Por tanto, hemos de preparar los siguientes programas:

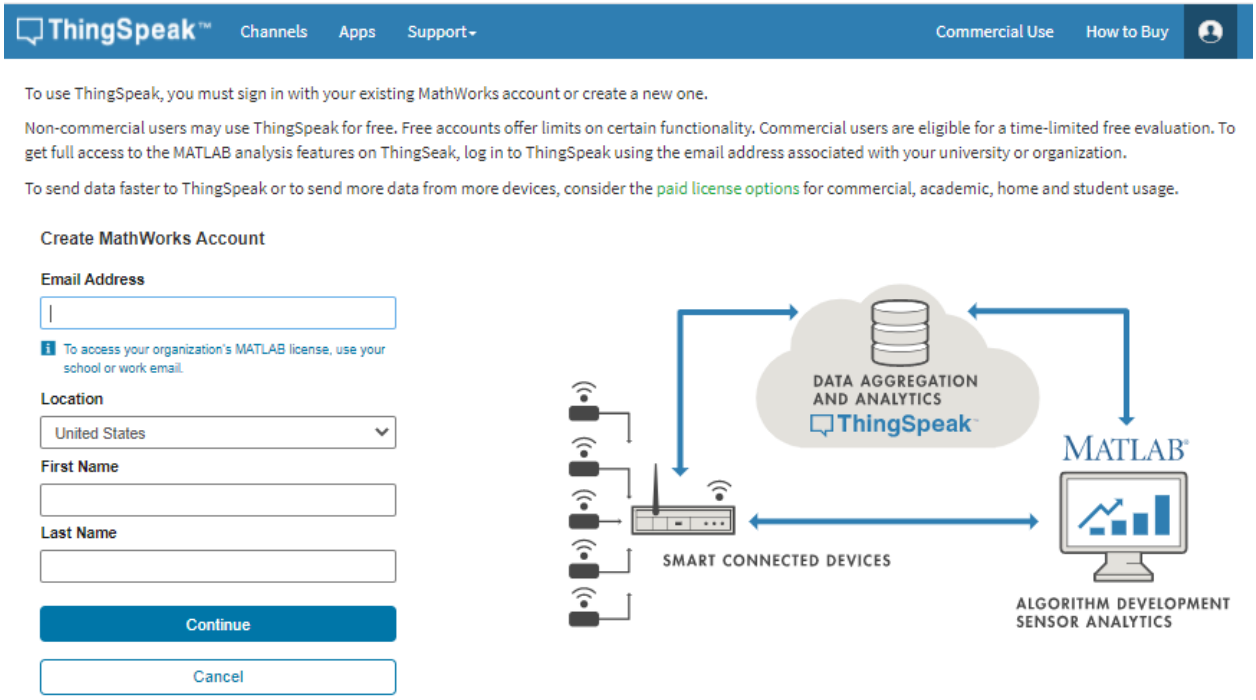
- ArduinoBlocks: programa de recogida y envío de datos.
- ThingSpeak: programa per ver los datos en el ordenador a través de Internet.
- ThingView: aplicación para ver los datos en el teléfono móvil.

Para configurar los tres elementos hemos de seguir los pasos descritos a continuación.

7.12.2.1 Reto A12.2.1. ThingSpeak.

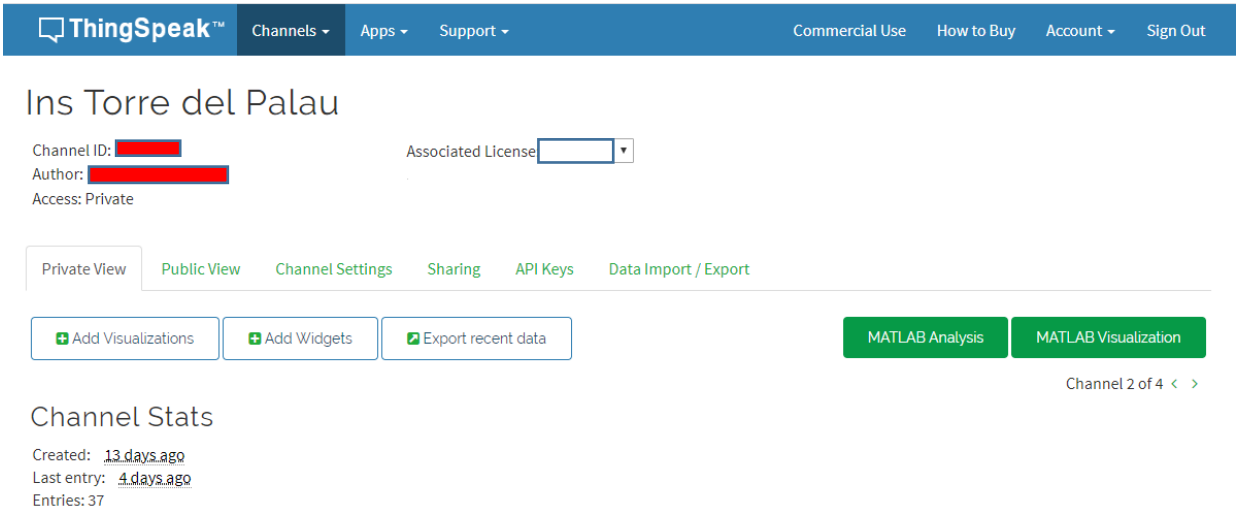
Reto A12.2.1. ThingSpeak.

Crear una cuenta en ThingSpeak: https://thingspeak.com/login



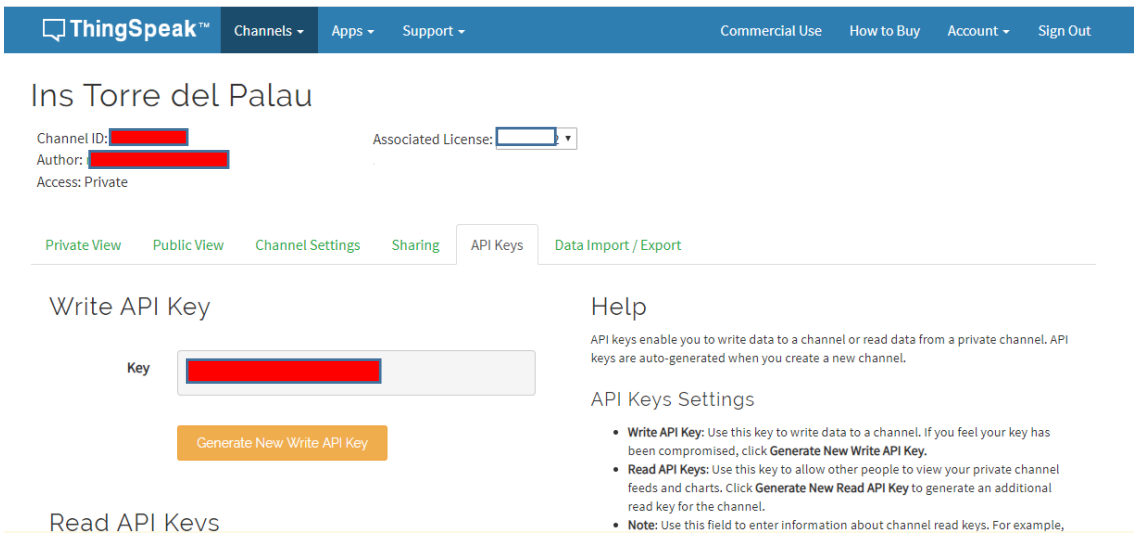
Una vez que hemos creado la cuenta ThingSpeak hemos de apuntar los siguientes datos:

- Channel ID: referencia de nuestro dispositivo.
- Author: referencia del autor del dispositivo.

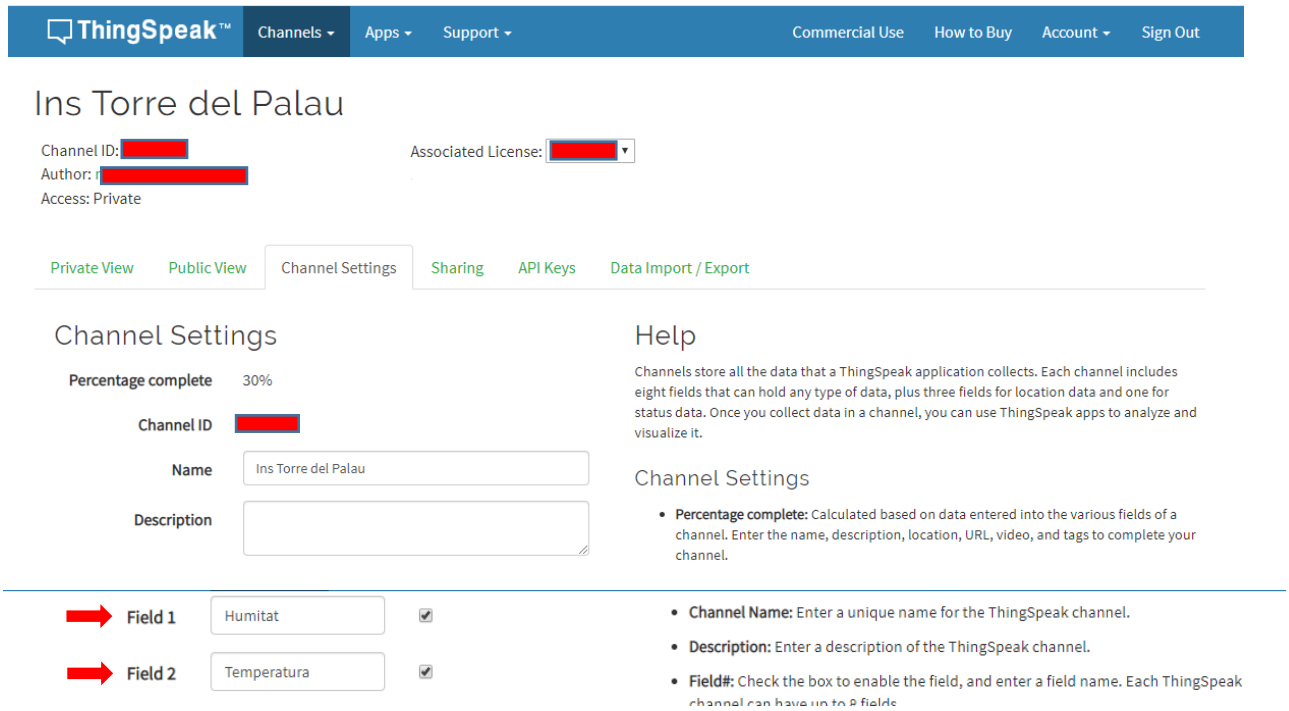


Otros datos importantes están en la pestaña “API Keys”.

- Write API Key: código identificativo para enviar los datos.



Por último, hemos de hacer la configuración de los canales dónde recibiremos los datos en *Channel Settings*.



¡Ya hemos configurado ThingSpeak!

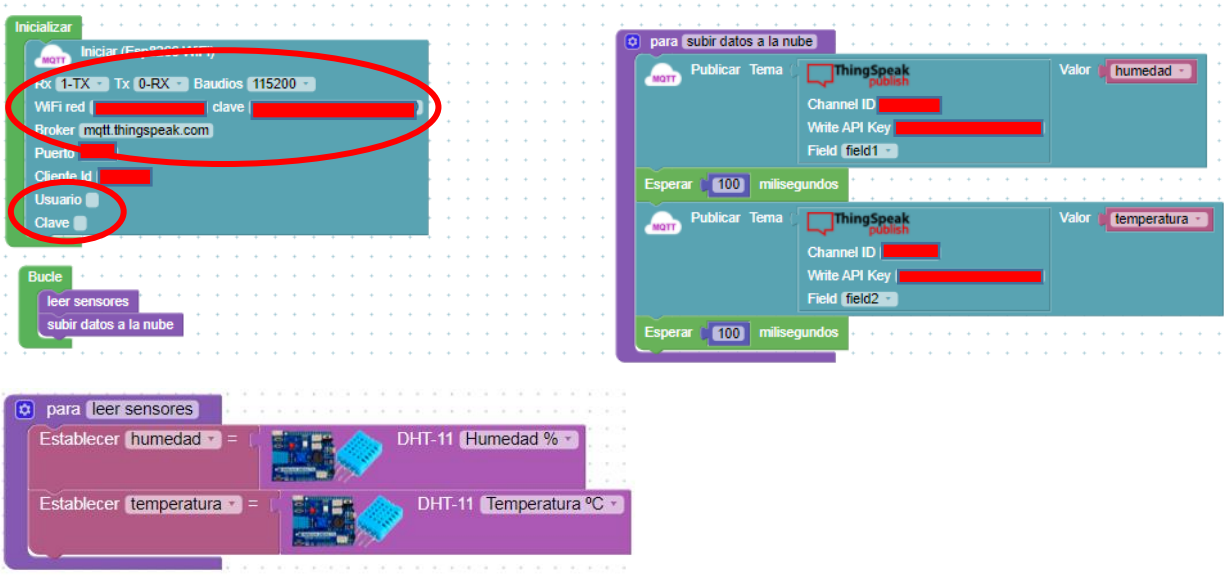
7.12.2.2 Reto A12.2.2. ArduinoBlocks.

Reto A12.2.2. ArduinoBlocks.

Una vez configurado ThingSpeak, prepararemos el programa de ArduinoBlocks para leer los datos de los sensores y enviarlos vía Wifi.

En el bloque “Inicializar”, configuraremos:

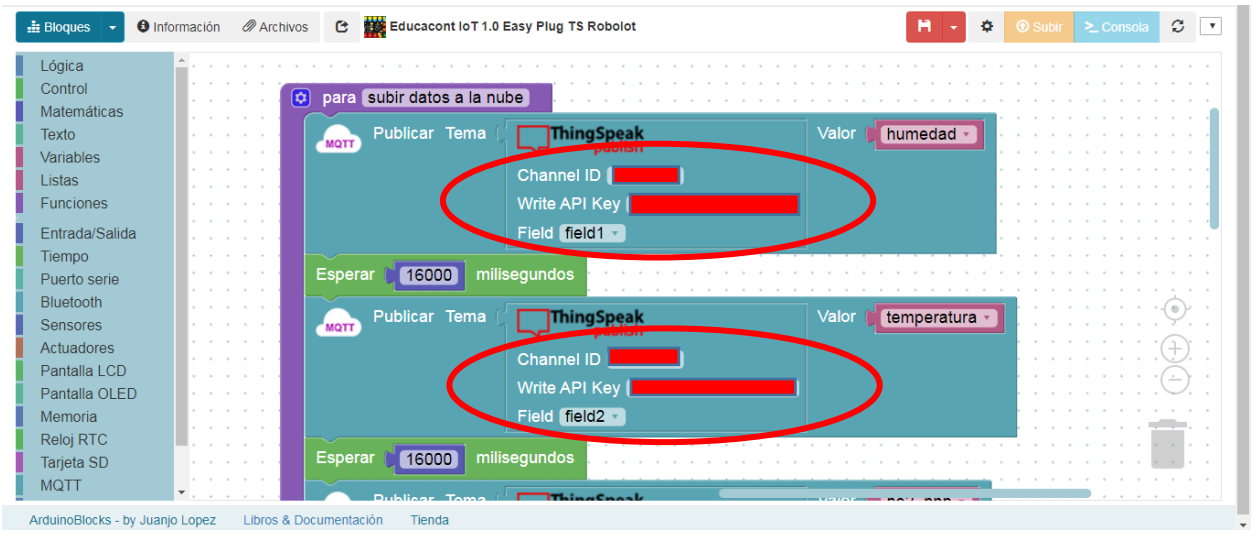
- WIFI red: nombre de la red Wifi donde nos queremos conectar.
- Clave: contraseña de nuestra red Wifi.
- Usuario: nombre de usuario que tenemos en ThingSpeak (mwa0xxx).
- Clave: contraseña de nuestra cuenta de ThingSpeak.



A continuación, configuraremos la publicación de los datos con la función *para: subir datos a la nube*:

- Channel ID: identificador de nuestro canal en ThingSpeak.
- Write API Key: código identificativo para enviar los datos a ThingSpeak.

Estos datos los hemos de poner en cada campo de: *Publicar Tema*.

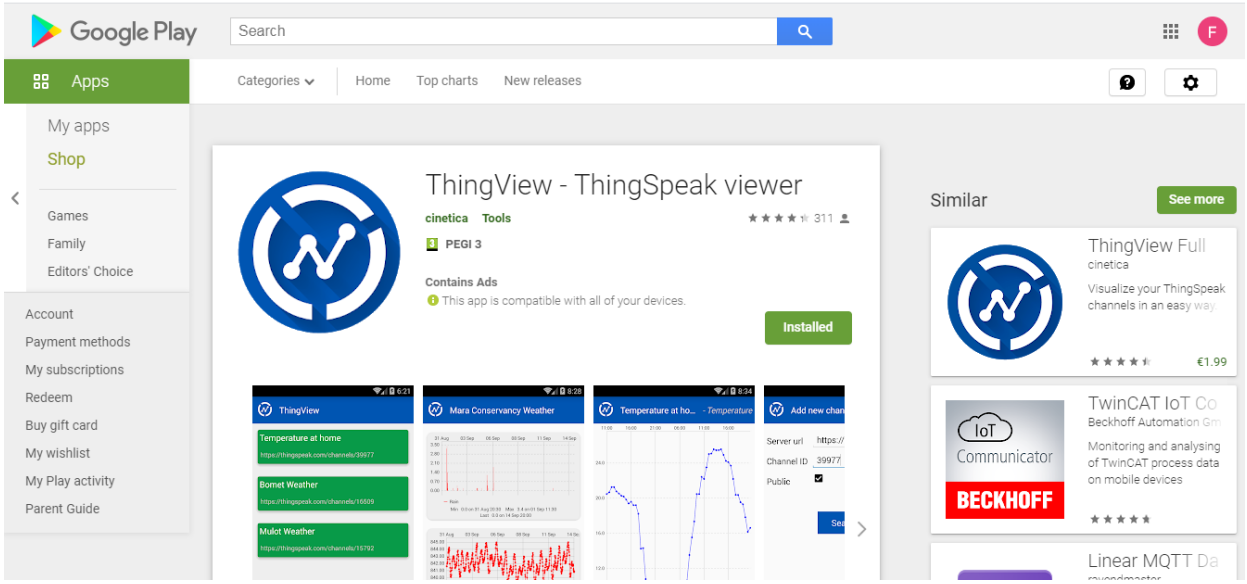


¡Ya hemos configurado ArduinoBlocks!

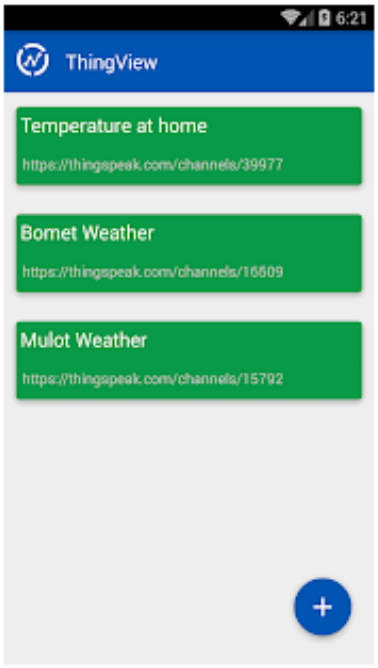
7.12.2.3 Reto A12.2.3. ThingView.

Reto A12.2.3. ThingView.

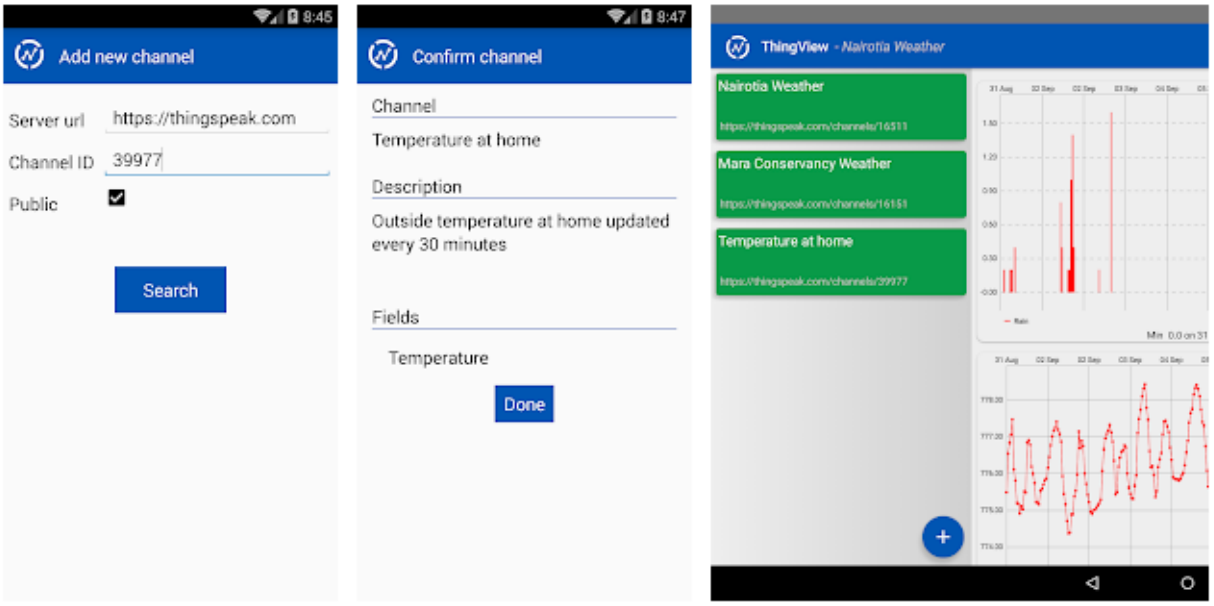
Si queremos ver los datos en el móvil, podemos instalar la aplicación ThingView. Para hacer la instalación hemos de seguir los siguientes pasos:



Add channel → *Channel ID*: ponemos el código de nuestro canal de ThingSpeak (Channel ID).



Visualización de datos en ThingView.



Ya está configurado, ¡a ver datos!

8 Proyectos con la placa Imagina TDR STEAM.

A continuación, os proponemos una serie de proyectos finales con las placas ***Kesyestudio UNO R.3*** e ***Imagina TDR STEAM***. Están realizados en unas fichas que ocupan una hoja para que sean más fáciles de visualizar. Cada proyecto tiene dos tipos de actividades muy similares, pero programadas de dos formas diferentes: utilizando los bloques propios de la Imagina TDR STEAM o utilizando los bloques convencionales.

Aquí tenéis un enlace a un recopilatorio de fichas de profesores con proyectos realizados en el curso "Reptes TdR STEAM" impartido por CESIRE y Robolot Team (*nota: están pendientes de verificar y comprobar).*

https://www.dropbox.com/sh/m04oqajozo4oh1x/AADvpopOeWnjF048q-wS5hkBa?dl=0

También se puede integrar todo el kit de la placa Imagina TDR STEAM dentro de una caja 3D diseñada por Eduard Casadevall. Aquí tenéis las fotografías de este práctico montaje.