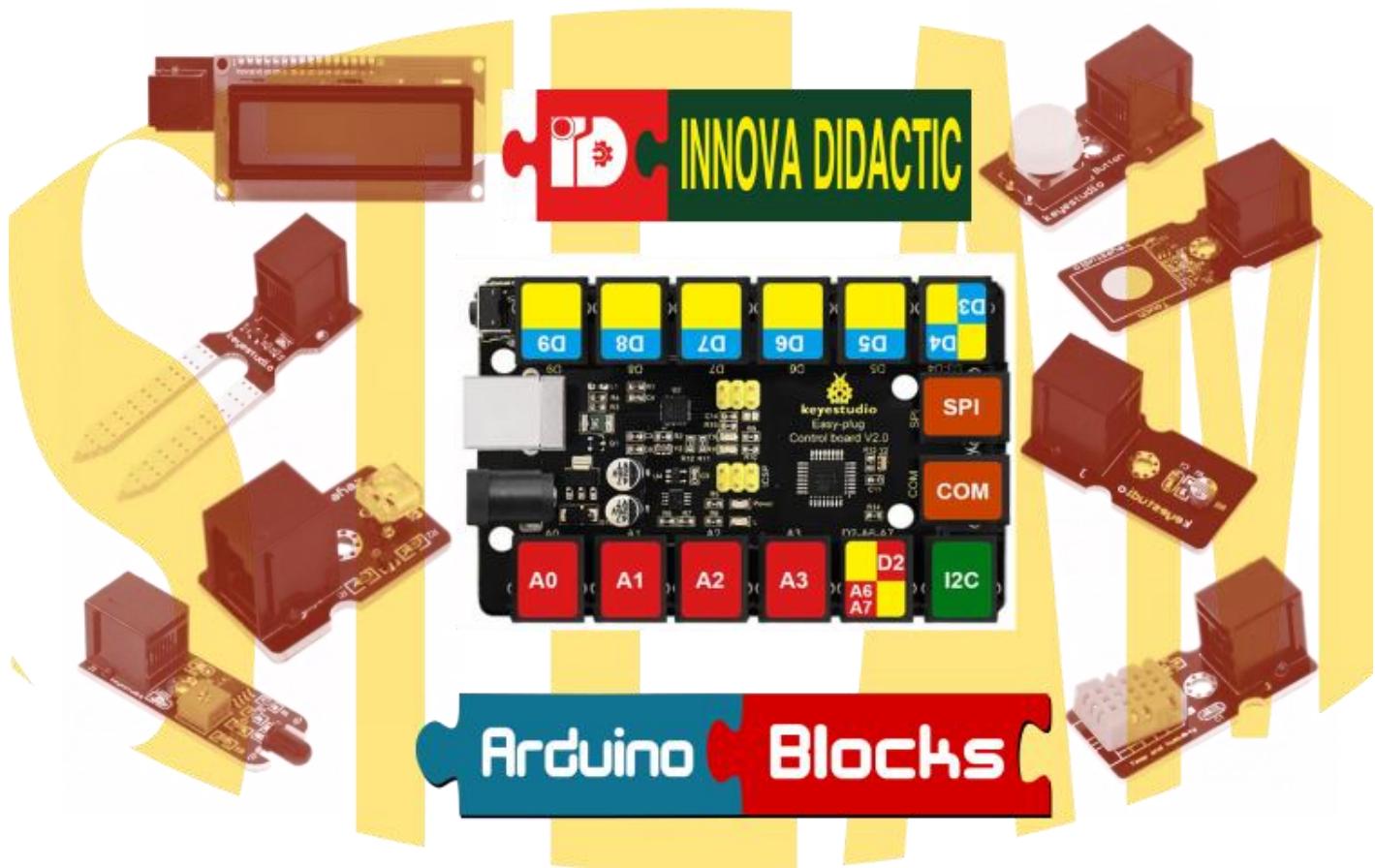


Actividades con ArduinoBlocks y el Kit Keyestudio para Secundaria



Índice

Introducción	3
¿Cómo funciona un sistema de control programado?	4
Componentes de Kit Keyestudio EasyPlug Secundaria	5
Placa de control, Sensores y Actuadores.....	5
Placa de control	10
Especificaciones	10
ArduinoBlocks.....	11
ArduinoBlocks y el Kit Keyestudio EasyPlug Secundaria	11
Preparativos: instalación de los drivers y programas	11
Programación Arduino con ArduinoBlocks	11
Actividades con el KIT Keyestudio Easy Plug Secundaria	11
A01. – Encender/Apagar un LED	11
A02. – Control de brillo de un LED usando el PWM.	11
A03. – Generar notas con el Buzzer o Zumbador.....	11
A04. – Sensor pulsador capacitivo	11
A05. – Sensor de colisión.....	11
A06. – Control de un Servomotor.....	11
A07. – Sensor infrarrojo para evitar obstáculos.....	11
A08. – Receptor infrarrojo IR.....	11
A09. – Sensor de llama y robot bombero.	11
A10. – Potenciómetro deslizante.....	11
A11. – Lectura del valor de la fotocélula (LDR).....	11
A12. – Encendido de una farola. Sensor luz ambiental.....	11
A13. – Sensor de Ultrasonidos I.....	11
A14. – Sensor de Ultrasonidos II.....	11
A15. – Control de temperatura de una habitación.....	11
A16. – Sensor de presencia y Funciones.....	11
A17. – Sensor de gas y mediciones por Consola.....	11
A18. – Calidad del aire y Matriz de Led I	11
A18. – Calidad del aire y Matriz de Led II	11
A19. – Pantalla LCD y Sensor DHT11	11

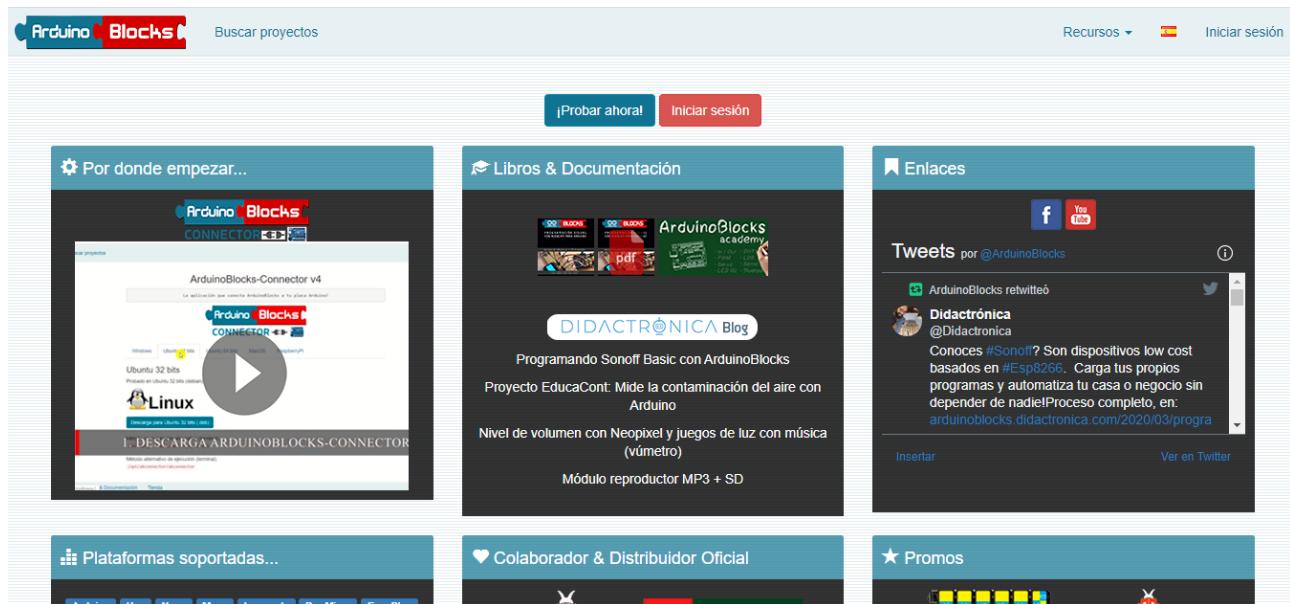
A20. – Módulo Bluetooth	11
A21. – La matriz LED 8x8 y el módulo Joystick.....	11
A22. – Pantalla OLED	11
A23. – Reloj DS3231 + Pantalla LCD.....	11
A24. – Control del servo con Acelerómetro.....	11
A25. – LED RGB	11
A26. – Comunicaciones por wifi, enviando información.....	11
A27. – Comunicaciones por wifi, activando un elemento.....	11

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 diferentes sensores y actuadores.

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

Para realizar la programación con el **Kit Keyestudio Easy Plug Secundaria** utilizaremos un lenguaje de programación visual basado en bloques llamado [ArduinoBlocks](#). Hay quien podría pensar que un lenguaje de este estilo es muy básico y limitado, pero ya veremos a lo largo del manual la gran potencialidad y versatilidad de este programa.

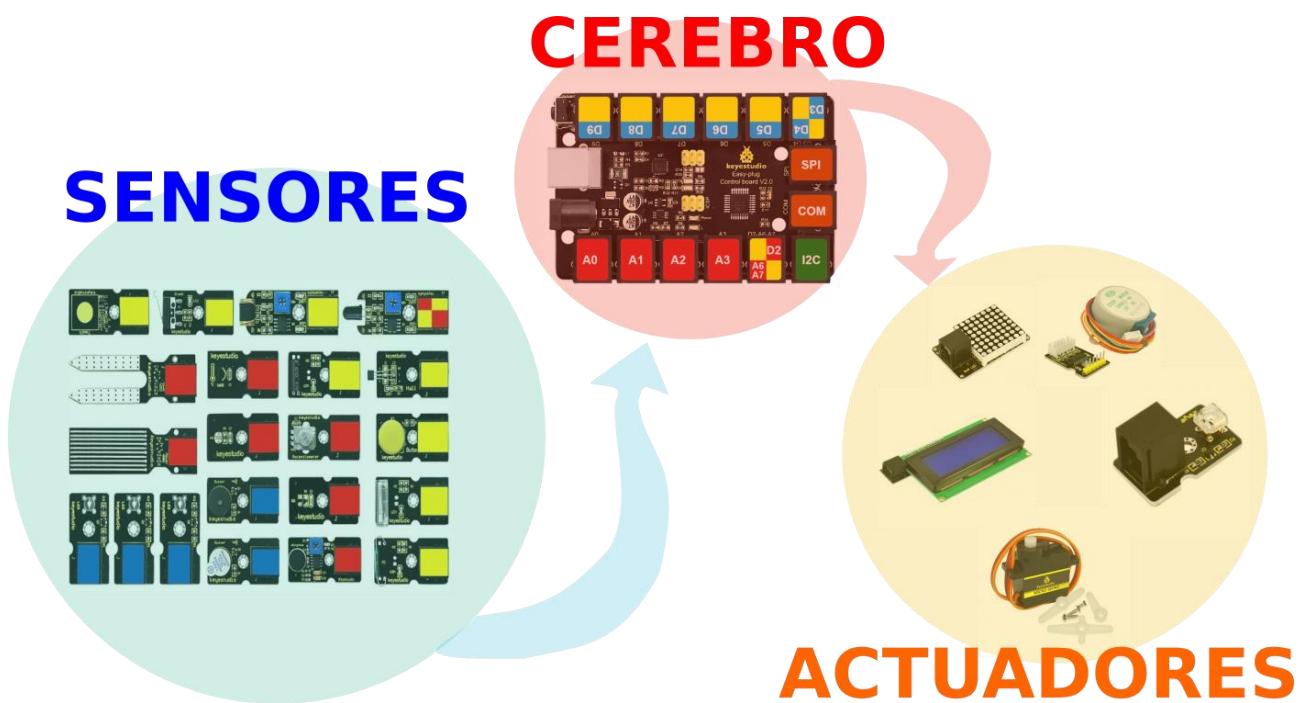


The screenshot shows the main interface of the ArduinoBlocks website. At the top, there's a navigation bar with the ArduinoBlocks logo, a search bar labeled "Buscar proyectos", and links for "Recursos", "Español", and "Iniciar sesión". Below the header, there are several sections: "Por donde empezar..." featuring a video player with a play button; "Libros & Documentación" listing various resources like "DIDACTRÓNICA Blog", "Programando Sonoff Basic con ArduinoBlocks", and "Proyecto EducaCont: Mide la contaminación del aire con Arduino"; "Enlaces" showing a Twitter feed from @ArduinoBlocks; "Plataformas soportadas..." listing supported platforms (Arduino, Uno, Nano, Micro, Leonardo, Due, Micro, GemmaBlue); "Colaborador & Distribuidor Oficial" featuring the Keyestudio logo; and "Promos" showing small promotional images.

El portal es www.arduinoblocks.com.

¿Cómo funciona un sistema de control programado?

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



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

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



Componentes de Kit Keyestudio Easy Plug Secundaria

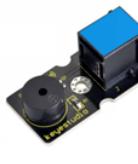
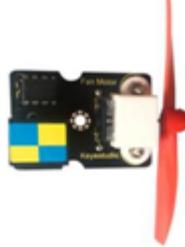
El **Kit Keyestudio Easy Plug Secundaria** se basa en una plataforma Arduino de código abierto, flexible y fácil de usar. La placa de control del kit viene con conectores RJ11 para conectar los sensores y actuadores, esto hace que las conexiones sean sencillas y no se necesite trabajar con protoboards. Más adelante se verán como son estas conexiones.

Placa de control, Sensores y Actuadores

No.	Componente	Referencia	Imagen
PLACA DE CONTROL			
1	Placa Keyestudio Easy Plug	KS0240	
2	Cables RJ11	KS9001 KS9011	
3	Cable USB	CABLEARDU	
SENSORES			
4	Módulo pulsador capacitivo	KS0113	

5	Módulo Joystick	KS0245	
6	Módulo Fotocélula (LDR)	KS0106	
7	Luz ambiental TEMT6000	KS0244	
8	Módulo DHT-11	KS0129	
9	Temperatura DS18B20	KS0124	
10	Analógico de gas MQ-2	KS0131	
11	Calidad del aire	KS0133	
12	Potenciómetro deslizante	KS0373	

13	Sensor de colisión	KS0111	
14	Sensor de llama	KS0116	
15	Receptor de infrarrojos IR	KS0125	
16	Infrarrojo para evitar obstáculos	KS0120	
17	Módulo Sensor PIR	KS0122	
18	Módulo SR01 Ultrasonidos	KS0376	
19	Sensor de Aceleración	KS0128	

ACTUADORES			
20	Módulo LED Verde, Amarillo y Rojo.	KS0225 KS0226 KS0224	
21	Módulo LED RGB	KS0370	
22	Módulo zumbador pasivo (Buzzer)	KS0103	
23	Servomotor 9g.	KS0194	
24	Driver con motor y hélice	KS0241	
MÓDULOS			
25	Pantalla LCD 1602 I2C	KS0137	
26	Matriz direccionable de leds 8x8	KS0395	

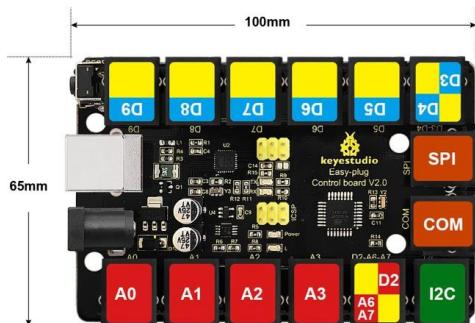
27	Oled 128 x 64	KS0136	
28	Reloj DS3231	KS0130	
29	Módulo Easy Plug Servo	KS9006	
30	Zócalo para Wifi y Bluetooth	KS0393	
31	Wifi	KS0339	
32	Bluetooth 2.0	KS0135	
33	Módulo HUB I2C	KS0390	

Placa de control

La placa de control del **Kit Keyestudio Easy Plug Secundaria** está basada en una **Arduino UNO**. **Arduino** es una plataforma de prototipos electrónica de **código abierto** (Open-Source) basada en hardware y software flexibles y fáciles de usar. Está pensado para **artistas, diseñadores, ...** como hobby y en general para cualquiera interesado en crear objetos o entornos interactivos.

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

Al ser hardware libre existen multitud de fabricantes que han desarrollado versiones basadas en Arduino. Uno de esos fabricantes es **Keyestudio** que entre todas sus placas destaca la Keyestudio Easy Plug con conectores RJ11:



La placa lleva un microcontrolador que está basado en el ATmega328.

Una de las principales diferencias entre Arduino UNO original y Keyestudio Easy Plug es que esta última presenta conexiones RJ11 para el conexionado de sensores, actuadores y módulos, lo que evita cableados incorrectos que puedan dañar la placa o los sensores.

La placa Easy Plug tiene puertos **digitales** de entrada / salida (incluidas tres salidas PWM), cuatro interfaces de entrada **analógica**, una interfaz dual digital, una interfaz de comunicación **SPI**, una interfaz de comunicación de puerto serie, una interfaz de comunicación **I2C** y una interfaz de módulo de joystick.

Especificaciones

- Voltaje de funcionamiento: + 5V
- Voltaje de entrada externo: + 7V ~ + 12V. (Límite: +6 V. <+ 20 V).
- Corriente de interfaz DCI / O: 20 mA
- 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

ArduinoBlocks

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

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

The screenshot shows the ArduinoBlocks software interface. On the left, there is a vertical palette of blocks categorized into groups: Lógica, Control, Matemáticas, Texto, Variables, Listas, Funciones, Entrada/Salida, Tiempo, Puerto serie, Bluetooth, Sensores, Actuadores, Pantalla LCD, Memoria, Motor, Motor-Shield, Keypad, Reloj RTC, GPS, Tarjeta SD, MQTT, NeoPixel, RFID, LedMatrix 8x8, and Domótica. The workspace contains several blocks: "Leer digital Pin 2", "Escribir digital Pin 2 ON", "Leer analógica Pin A0", "Escribir analógica (PWM) Pin 3 Valor 0", "Leer digital Pin 2", "Escribir digital Pin 2 false", "Leer analógica Pin A 0", "Escribir analógica (PWM) Pin 3 Valor 0 (0 - 255)", "Leer pulso Pin 2 ON Tiempo de espera 1000", and "Interrupción Pin 2 RISING". To the right, there is a visual representation of two green C-shaped blocks labeled "Inicializar" and "Bucle".

ArduinoBlocks y el Kit Keyestudio Easy Plug Secundaria

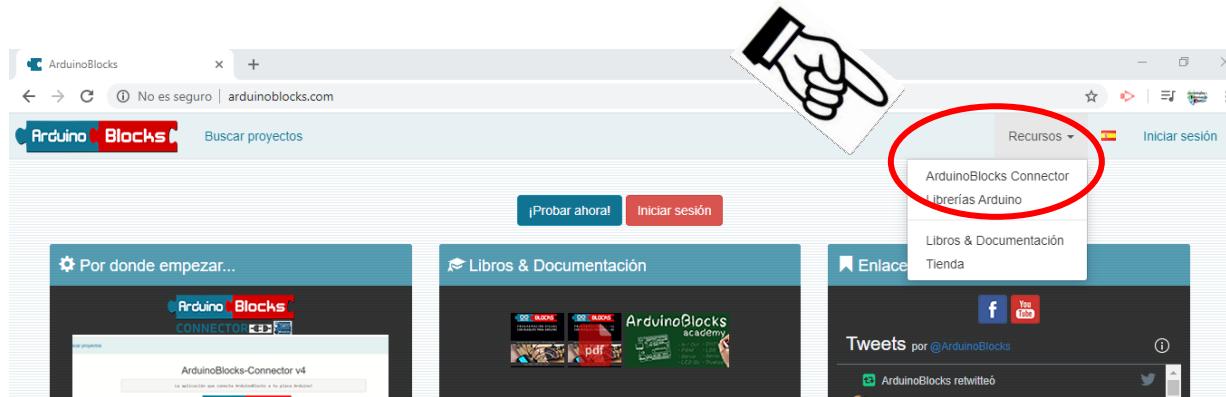
En este manual usaremos **ArduinoBlocks** dedicado al uso de la placa **Keyestudio Easy Plug**, con estos bloques podremos programar las entradas y salidas de nuestra placa para que realice las tareas que queramos.

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

Preparativos: instalación de los drivers y programas

Para programar nuestra placa **Keyestudio Easy Plug** con el programa online **ArduinoBlocks**, necesitaremos instalar un pequeño programa disponible en la sección de “**Recursos**” en la página principal de arduinoblocks.com.

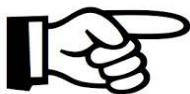
Se trata de [ArduinoBlocksConnector v.4](#)



También lo puedes descargar directamente de este enlace:

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

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



ArduinoBlocks-Connector v4

La aplicación que conecta ArduinoBlocks a tu placa Arduino!



Windows

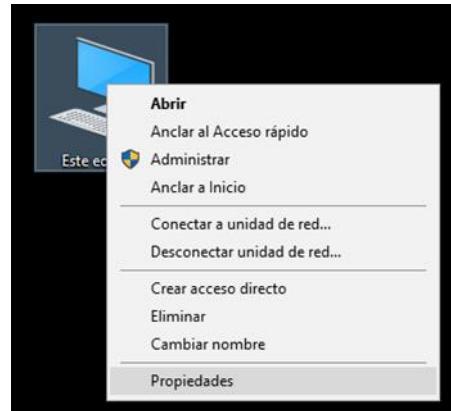
Probado en XP, 7, 10 (32/64 bits)



Descarga para Windows (.exe)

¡Desactiva el antivirus si la descarga falla!

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



Ventana principal del Panel de control

Ver información básica acerca del equipo

Edición de Windows

Windows 10 Pro
© 2018 Microsoft Corporation. Todos los derechos reservados.

Sistema

Fabricante: ASUSTek Computer Inc.
Procesador: Intel(R) Core(TM) i5-8250U CPU @ 1
Memoria instalada (RAM): 8,00 GB (7,86 GB utilizable)

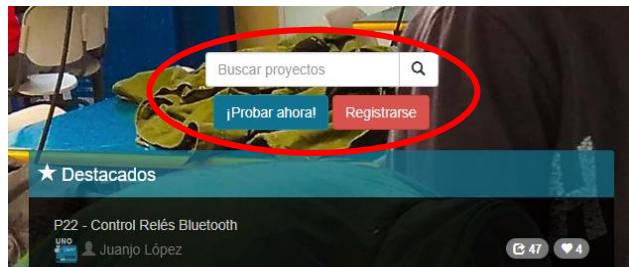
- ▶ Procesadores
- ▶ Puertos (COM y LPT)
 - ▶ **Arduino Uno (COM11)** (circulado en rojo)
 - ▶ Serie estándar sobre el vínculo Bluetooth (COM10)
 - ▶ Serie estándar sobre el vínculo Bluetooth (COM9)
- ▶ Teclados
- ▶ Unidades de disco

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

Programación Arduino con ArduinoBlocks

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

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



Nuevo usuario

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

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

Recibir información y novedades por email

Seguidamente ya podremos “Iniciar sesión”.

Iniciar sesión

Correo electrónico	<input type="text"/>
Password	<input type="text"/>
<input type="button" value="Login"/>	

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

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

Mis proyectos

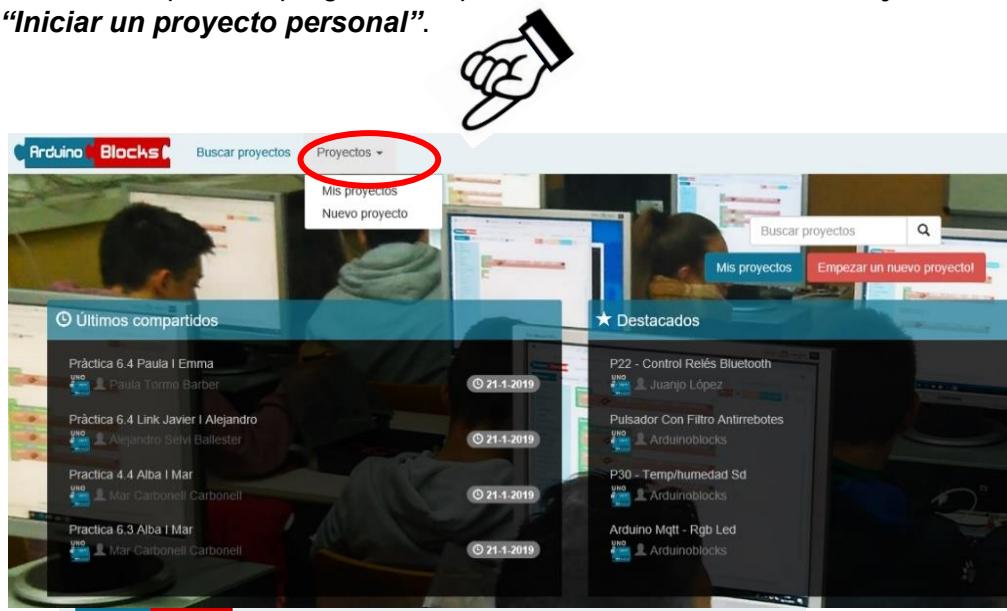
Proyectos personales [93] Alumno [12] Profesor [13] Compartido conmigo [0]

Mostrando 1-93 de 93 elementos.

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

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

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



Proyecto personal

Iniciar un proyecto personal

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

Profeso

Crear un proyecto

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

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



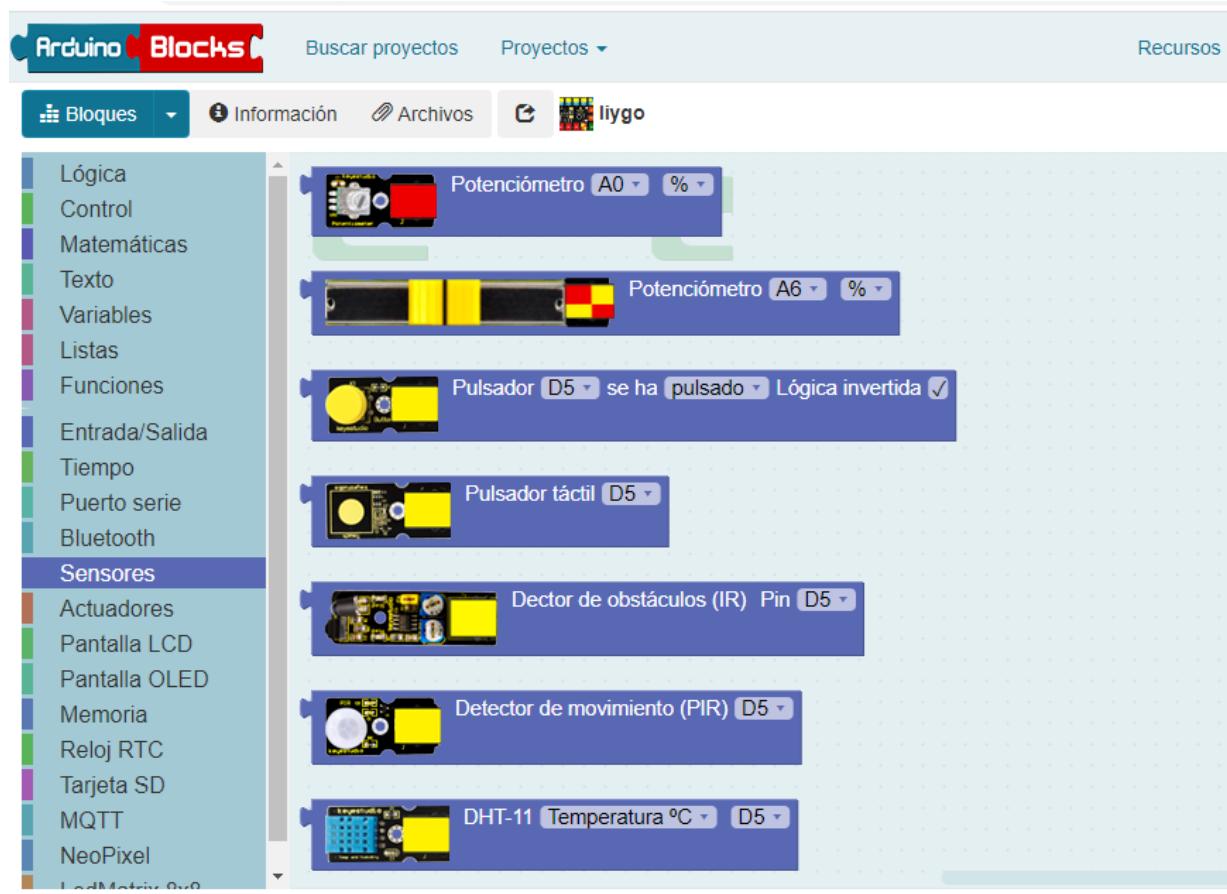
Esta opción nos presentará los menús necesarios para poder programar nuestra placa de forma fácil y sencilla con las funciones preparadas expresamente para la Easy Plug.

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

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

Nuevo proyecto personal
Tipo de proyecto Keyestudio KeyBot
Nombre
Descripción Normal A B I U S E M %
Componentes
Normal A B I U S E M %
Comentarios
Normal A B I U S E M %
Nuevo proyecto

La siguiente imagen nos muestra la interfaz de programación con los bloques específicos para poder programar con mucha facilidad nuestra placa de control Easy Plug.



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

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



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

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

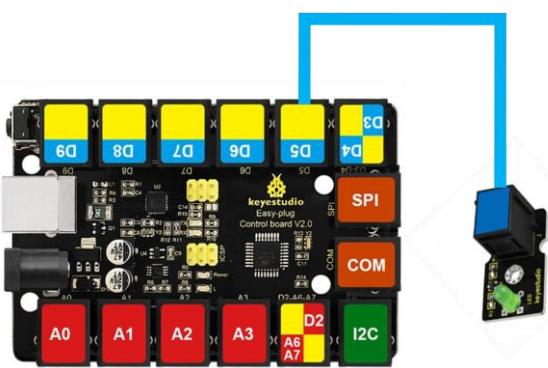
Por último, con el botón “Subir” (3) transferimos el programa del ordenador a la placa.

Actividades con el KIT Keyestudio Easy Plug Secundaria

A continuación, os proponemos una serie de actividades y prácticas para aprender a programar vuestro Kit de Keyestudio Easy Plug Secundaria paso a paso para que realice infinidad de tareas.

A01. – Encender/Apagar un LED

Vamos a empezar con nuestro primer programa que será encender y apagar un LED variando los tiempos de encendido y apagado.

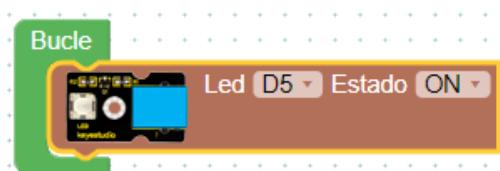
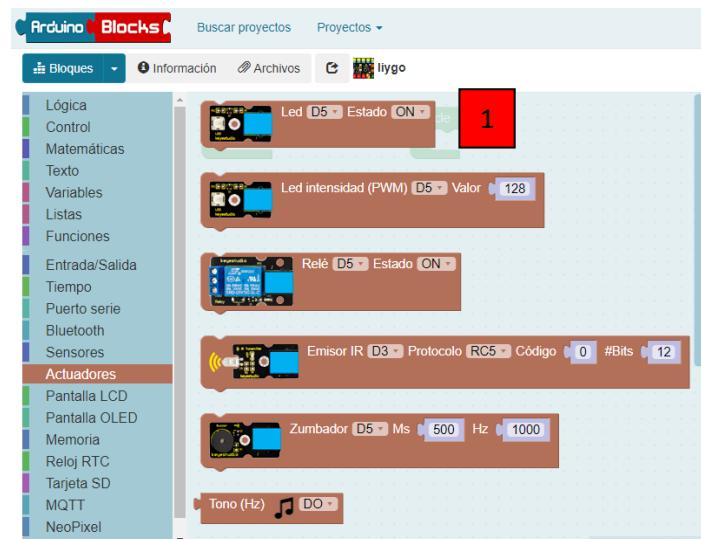


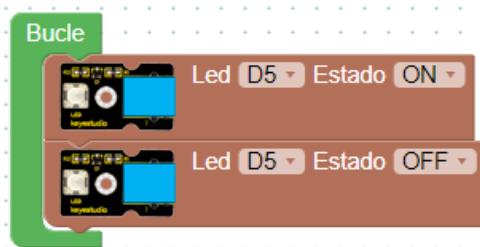
Para ello vamos a necesitar un LED y un cable RJ11. Conectaremos el LED al pin D5 de la placa de control.

En programación el primer programa que se realiza se le suele llamar “*Hola Mundo*”, así que este será nuestro “*Hola Mundo*” personal. Una vez seleccionado en **ArduinoBlocks** el “**Tipo de proyecto**” “**Keyestudio Easy Plug**” y puesto el nombre “***Hola Mundo***”, picharemos en el bloque de “**Actuadores**”.

Clicamos sobre el bloque 1 y lo arrastramos sobre el área de programación. Fíjate que en el área de programación hay dos bloques verdes “**Iniciar**” y “**Bucle**” y que podríamos meter nuestro bloque del LED en cualquiera de ellos (*Los bloques cuando son compatibles encajan como un puzzle*). Pues bien, todo lo que se meta dentro del bloque de “**Iniciar**” sólo se ejecutará la primera vez que se inicie el programa, mientras que si se colocan dentro del “**Bucle**” se ejecutarán una y otra vez hasta que apaguemos la placa.

Vamos a meter nuestro bloque de LED en el “**Bucle**” dejando seleccionado el Pin D5. El LED puede tener dos estados; ON/OFF, que podemos cambiar en el menú despegable.





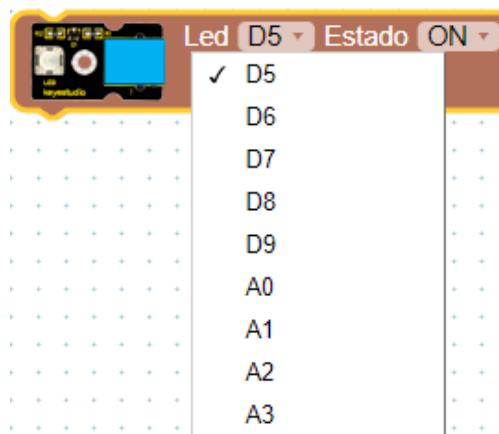
Si sólo dejamos este bloque con el LED en estado ON, este quedaría encendido para siempre, para que se apague deberemos meter el estado OFF.

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

Ahora tenemos el LED encendido durante 1 segundo y apagado otro. Prueba a cambiar los valores del tiempo.

El kit tiene tres LEDs; Verde, Amarillo y Rojo. Con lo que acabas de aprender seguro que se te ocurren muchas aplicaciones utilizándolos solos o juntos.

Recuerda que puedes conectar los LEDs en los **pines digitales D 5, 6, 7, 8 y 9**, aunque también los podrías conectar en los **pines analógicos A 1, 2 y 3**, esta opción no es recomendable ya que estos pines están reservados para otros fines.



A02. – Control de brillo de un LED usando el PWM.

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

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

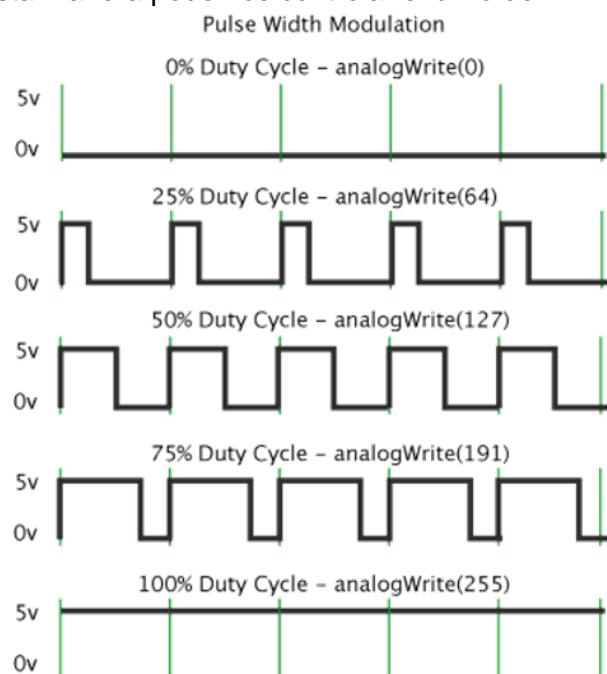
La modulación de ancho de pulso, o PWM, es una técnica para obtener resultados analógicos con medios digitales.

El control digital se utiliza para crear una onda cuadrada de ciclo de trabajo diferente, una señal conmutada entre encendido y apagado. Este patrón de encendido y apagado puede simular voltajes entre encendido total (5 voltios) y apagado (0 voltios) al cambiar la parte del tiempo que la señal pasa en comparación con el tiempo que la señal pasa.

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

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

La Placa Easy Plug y ArduinoBlocks tienen un bloque específico para controlar un LED por PWM y sólo permite utilizar los pines D5, 6 y 9.



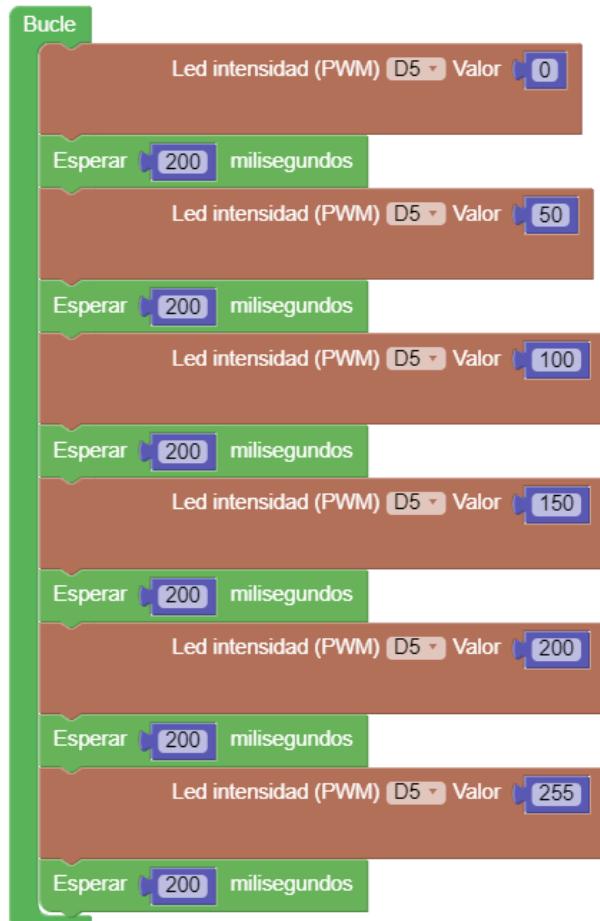
El rango de “**Valor**” lo podemos variar desde 0 hasta 255, de tal modo que si el valor es 0 el LED estará totalmente apagado y si el valor es 255 el LED brillará al máximo.

Podríamos hacer la actividad anterior con este programa:



Ahora vamos a hacer que el brillo del LED varíe.

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



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



Este bloque lo tenemos dentro de “**Control**”.



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

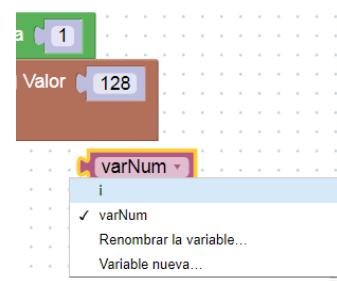
Este sería el programa de ir aumentando el brillo de un LED de 1 en 1 desde 0 hasta 255 con una espera de 20 milisegundos.



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

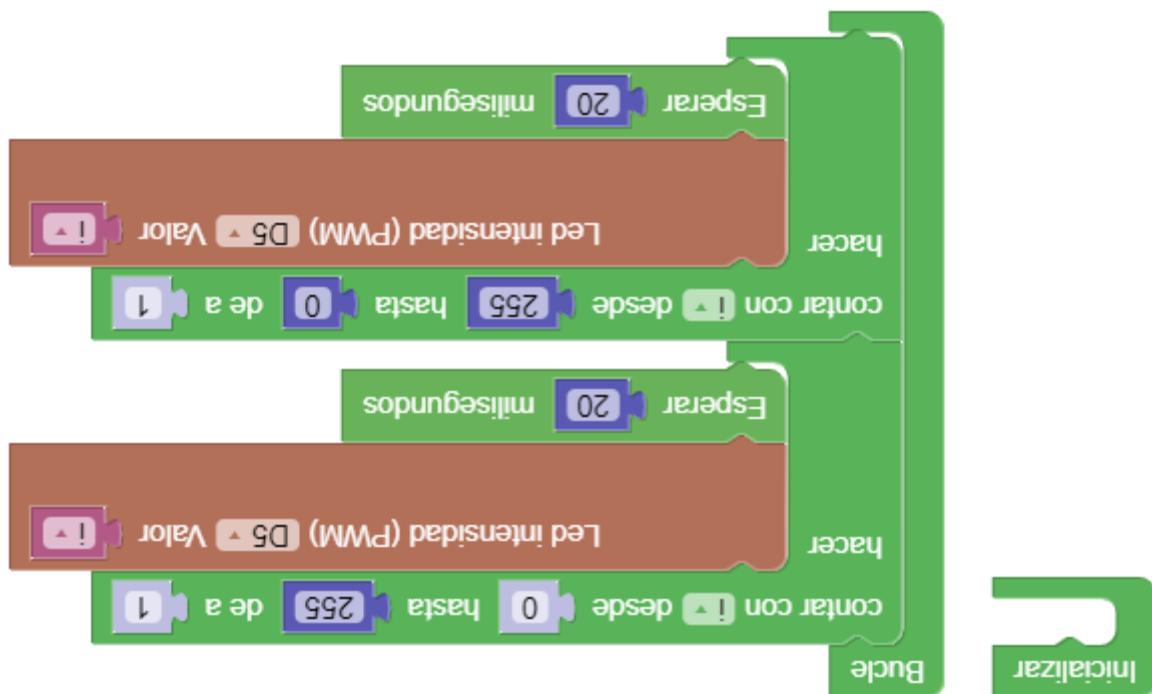


Para introducir el valor del PWM, del bloque del LED, la variable *i*, debemos ir a “**Variables**”, seleccionar el segundo bloque y en el menú desplegable seleccionar *i*.



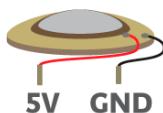
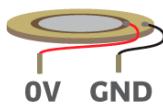
Intenta completar el programa haciendo que el brillo del LED ascienda de 0 a 255 y que descienda de 255 a 0.

Solución:



A03. – Generar notas con el Buzzer o Zumbador.

Zumbador, Buzzer en inglés, es un transductor electroacústico que produce un sonido o zumbido continuo o intermitente. En función de si se trata de un Buzzer Activo o Pasivo, este zumbido será del mismo tono o le podremos variar. Sirve como mecanismo de señalización o aviso y se utiliza en múltiples sistemas, como en automóviles o en electrodomésticos, incluidos los despertadores.



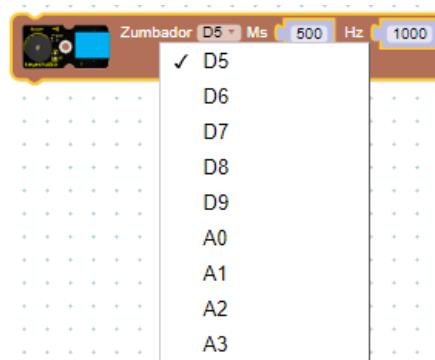
El **Kit Easy Plug Secundaria** tiene un Buzzer o Zumbador pasivo:



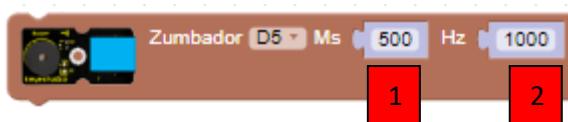
ArduinoBlocks tiene un bloque específico para el Zumbador, está en el menú de **“Actuadores”**.

The screenshot shows the ArduinoBlocks software interface. At the top, there are tabs for Bloques, Información, Archivos, and Iygo. On the left, a sidebar lists categories: Lógica, Control, Matemáticas, Texto, Variables, Listas, Funciones, Entrada/Salida, Tiempo, Puerto serie, Bluetooth, Sensores, and Actuadores (which is highlighted). The main workspace displays several blocks: Led D5 Estado ON, Led intensidad (PWM) D5 Valor 128, Relé D5 Estado ON, Emisor IR D3 Protocolo RC5 Código 0 #Bits 12, Zumbador D5 Ms 500 Hz 1000, Tono (Hz) DO, Zumbador D5 Reproducir RTTTL song: "", RTTTL The Simpsons, and Girar izquierda.

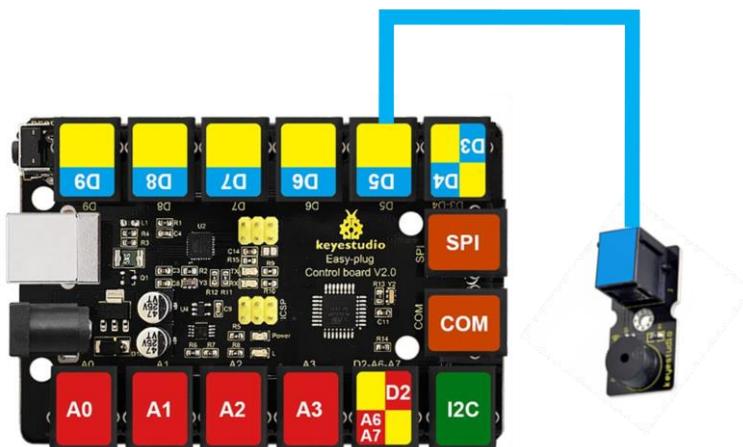
El Buzzer puede ser conectado en los pines digitales 5, 6, 7, 8 y 9 ya también en los analógicos, pero esto último no es recomendable.



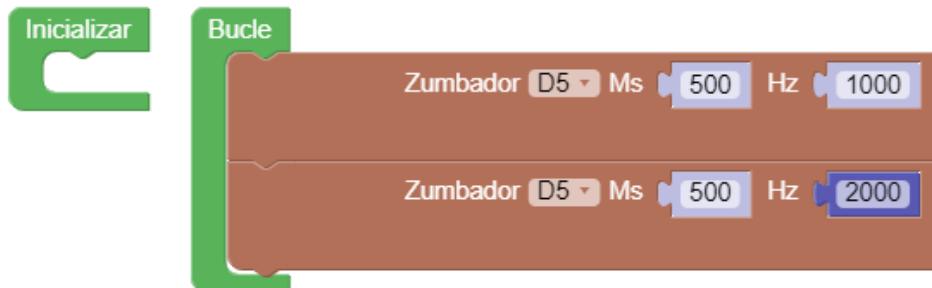
En el bloque **“Zumbador”** podemos variar dos parámetros: Ms (Milisegundos) (1) es el tiempo que dura cada sonido y Tono (2), que es la frecuencia a la que vibra la membrana para emitir el sonido.



Conecta el zumbador en el Pin D5 como en la imagen:



Prueba con este sencillo programa para ver cómo suena.



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

En la tabla de la derecha tienes las frecuencias de las notas musicales.

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

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



Iniciar

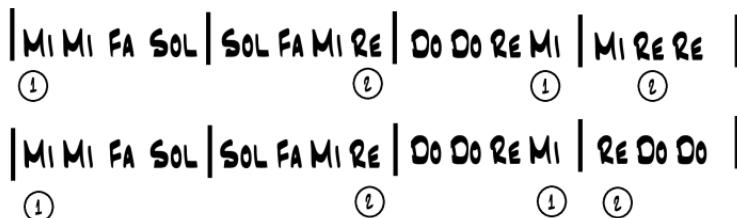
Bucle

- Zumbador D5 Ms 500 Hz 261.6
- Zumbador D5 Ms 500 Hz 277.2
- Zumbador D5 Ms 500 Hz 329.6
- Zumbador D5 Ms 500 Hz 349.2
- Zumbador D5 Ms 500 Hz 370
- Zumbador D5 Ms 500 Hz 440
- Zumbador D5 Ms 500 Hz 493.2
- Zumbador D5 Ms 500 Hz 523.3

¿Te atreves a programar esta melodía?

HIMNO DE LA ALEGRÍA

(PARTE 1)



En el menú de “**Actuadores**” existe el bloque de Tono (Hz).

Tono (Hz)

- DO
- DO#
- RE
- RE#
- MI
- FA
- FA#
- SOL
- SOL#
- LA
- LA#
- SI

Bloques

Información

Archivos

Iygo

Lógica

Control

Matemáticas

Texto

Variables

Listas

Funciones

Entrada/Salida

Tiempo

Puerto serie

Bluetooth

Sensores

Actuadores

Inicializar Bucle

Emisor IR D3 Protocolo RC5 Código 0 #Bits 12

Zumbador D5 Ms 500 Hz 1000

Tono (Hz) DO

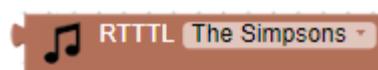
Zumbador D5 Ms Reproducir RTTTL song: "The Simpsons"

RTTTL The Simpsons

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



Mucho más sencillo así, ¿verdad?? Pero fíjate en este otro bloque RTTTL. Con él vas a poder hacer sonar infinidad de melodías en tu zumbador:



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

¡¡¡Pruébalo!!! Mira la cantidad de melodías que tienes disponibles.



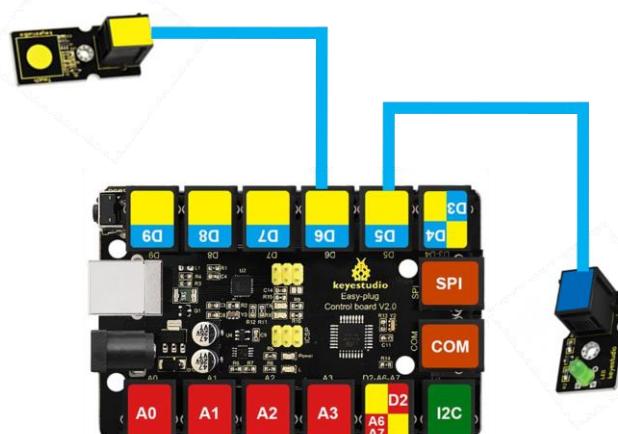
A04. – Sensor pulsador capacitivo

En la siguiente actividad vamos a utilizar el pulsador capacitivo. Previamente debemos recordar la diferencia entre un pulsador y un interruptor. Un interruptor es un dispositivo que abre o cierra el paso de la corriente eléctrica, por ejemplo, los interruptores de la luz de nuestras casas; cada vez que los pulsamos cambian de estado y permanecen en él hasta ser pulsados de nuevo. Sin embargo, un pulsador sólo se activa mientras dure la pulsación volviendo a su estado inicial en el momento en el que se deje de tocarlo. ¿Os imagináis que el timbre de vuestra casa fuera un interruptor y no un pulsador?

Vamos a realizar el siguiente montaje, conectamos el pulsador capacitivo en el pin D6 y un LED en el Pin D5 tal y como muestra la imagen.

Ahora vamos a realizar un programa en el cual al pulsar sobre el pulsador se encienda el LED y se apague cuando no lo pulsemos.

En el apartado “**Sensores**” encontramos el bloque correspondiente al pulsador táctil.



The screenshot shows the Arduino Blocks software interface with the 'Sensores' category selected in the sidebar. The main area displays several sensor blocks:

- Potenciómetro A0 %
- Potenciómetro A6 %
- Pulsador D5 se ha pulsado Lógica invertida
- Pulsador táctil D5
- Dector de obstáculos (IR) Pin D5
- Detector de movimiento (PIR) D5



Para realizar este programa necesitamos conocer una de las funciones más utilizadas en programación, la función de “**Condición**”, que se encuentra en las funciones del menú “**Lógica**”.



Condiciones “Si...hacer”.



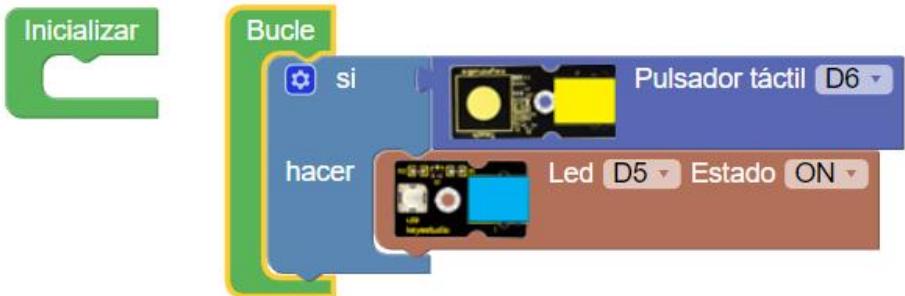
Se trata de la famosa función “**Si**” (“**If**” en inglés) que es uno de los pilares de la programación, ya que permite evaluar estados y tomar decisiones en consecuencia.

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



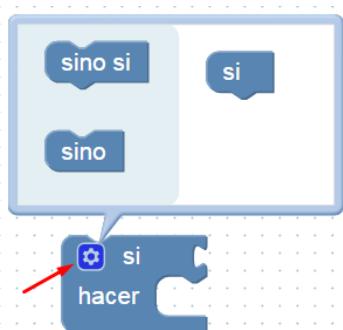
En el apartado de condiciones se pueden introducir multitud de factores: estado de sensores (analógicos o digitales), comparaciones, igualdades, operaciones matemáticas, etc.

Usando el bloque lógico “condicional” de “**Si....Hacer...**” el programa quedaría como la imagen, pero ¿qué va a ocurrir? Pruébalo.



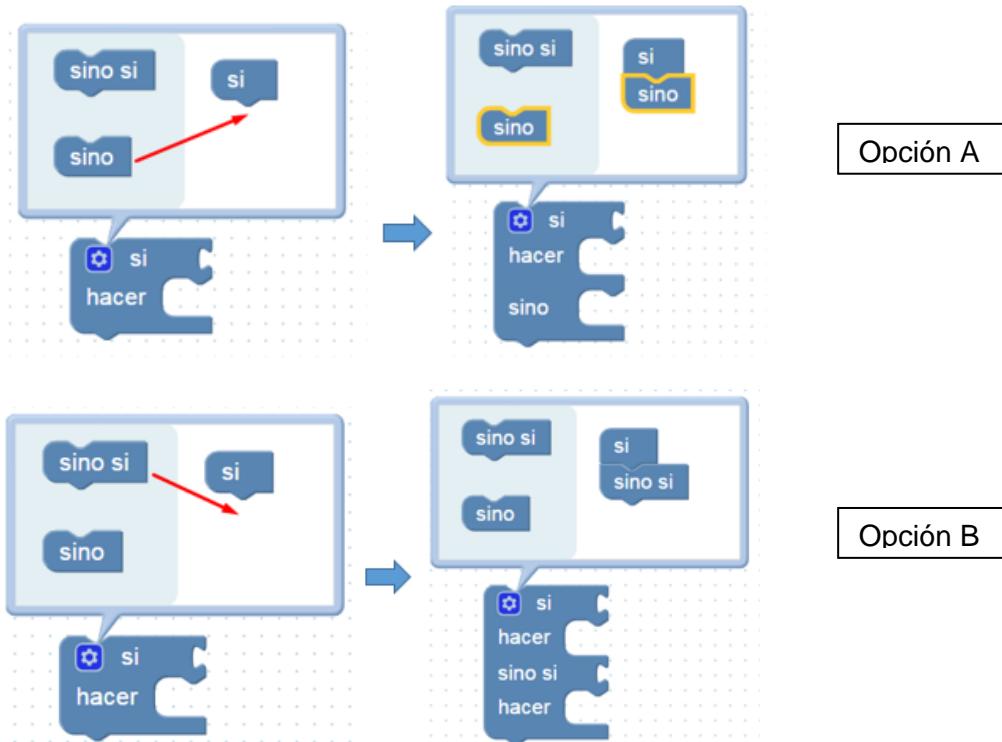
No se apaga el LED nunca, ¿no?, es lógico, en ningún momento del programa decimos que el LED tenga que estar en la posición OFF. Necesitamos introducir un “**Sino**”.

Si dejáramos el programa así, tendríamos un problema, ya que la primera vez que se cumpliera la condición, el LED se encendería, pero nunca se apagaría. Debemos poner una nueva condición, un “**Sino**”, para en ella cambiar el estado del LED a OFF. Para ello debemos ampliar el condicional de la siguiente manera;

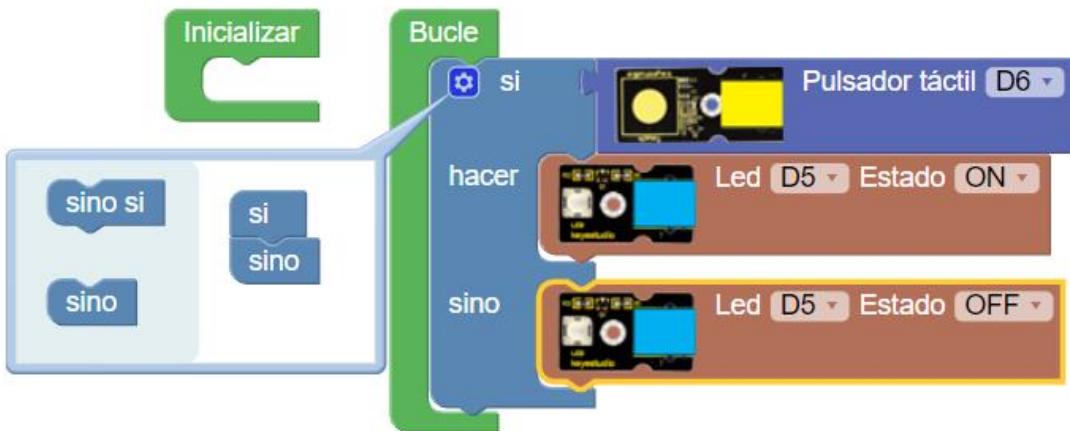


Haciendo clic sobre el símbolo del engranaje señalado con la flecha roja en la imagen, nos aparece un cuadro con funciones con las que podemos ampliar el condicional “**Si**”.

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



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



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

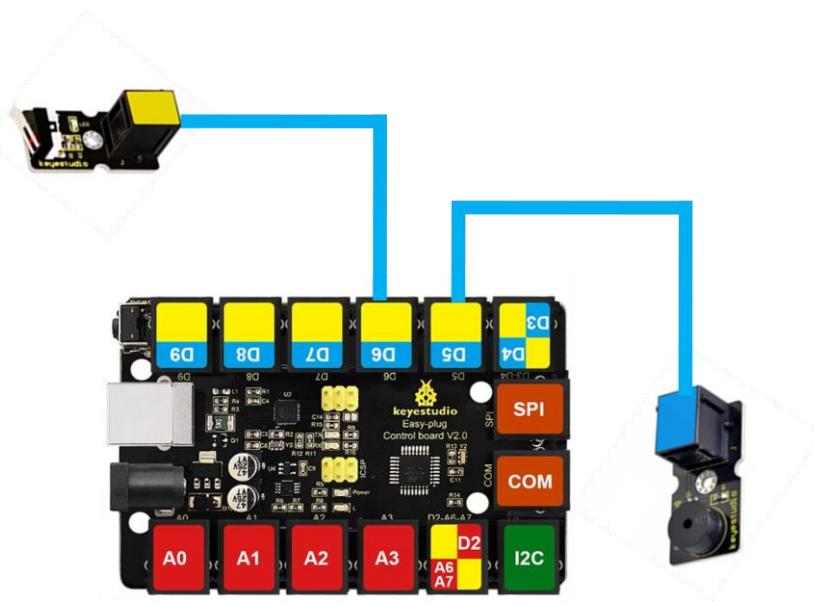
A05. – Sensor de colisión

Un sensor muy utilizado en la industria es el sensor de colisión o sensor de choque o también llamado final de carrera.

Básicamente su funcionamiento es como un pulsador. Sus contactos pueden estar N.A. (normalmente abiertos) o N.C. (normalmente cerrados). Este sensor dará señal cuando detecte presión sobre su palanca.



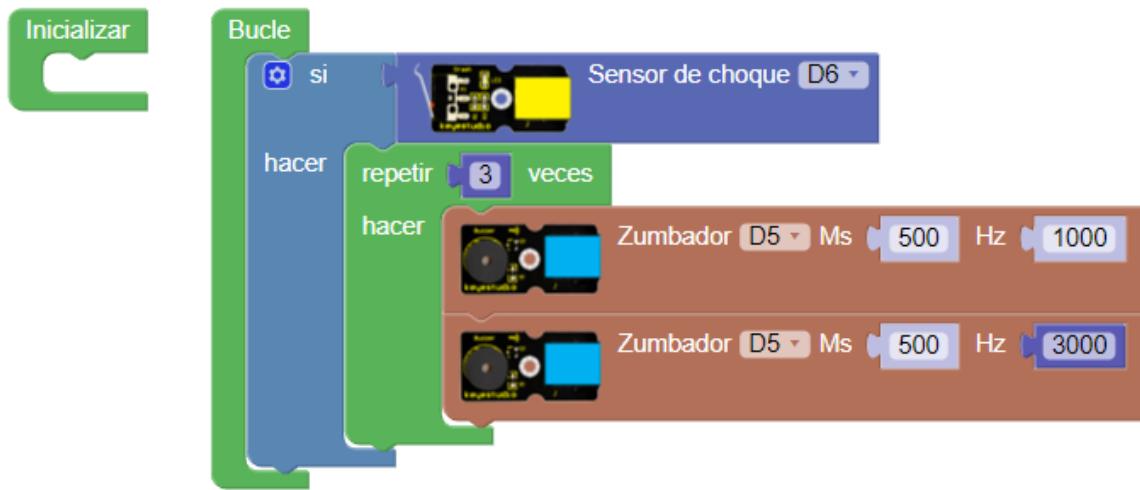
Vamos a realizar una sencilla actividad en la cual imaginemos que tenemos un sensor de colisión al final de una cinta transportadora y necesitamos que cuando un paquete llegue a esa posición suene un timbre para avisar. Necesitamos el sensor de colisión, un zumbador y la Placa Easy Plug. Realizamos el siguiente montaje conectando el sensor de colisión al Pin D6 y el buzzer al Pin D5.



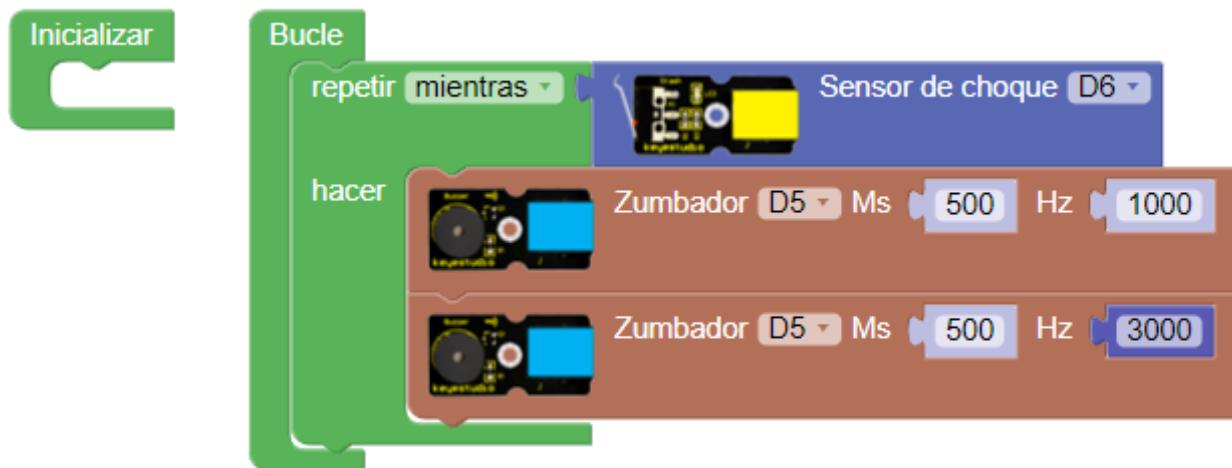
Para hacer esta práctica utilizaremos una nueva estructura de control que es la de **“Repetir... veces, hacer...”**. Con ella lo que conseguiremos es repetir el número de veces que consideremos la acción que queremos que se ejecute. En nuestro caso queremos repetir tres veces un sonido a modo de alarma.



El programa quedaría de la siguiente manera:



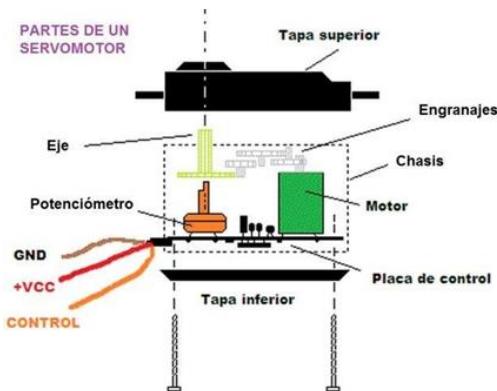
Tenemos otra opción y es hacer que mientras el sensor de choque esté accionado el pitido se repita indefinidamente. Para ello usaremos el bloque de control “**repetir mientras... hacer...**”:



A06. – Control de un Servomotor.

En esta actividad vamos a necesitar un servomotor (Ref. [KS0194](#) o [KS0209](#)) y el adaptador Easy Plug Servo KS9006.

Un elemento muy utilizado en robótica y el mundo de Arduino es un servomotor. Es un motor especial que puede posicionar su eje en un ángulo determinado y lo puede mantener en esta posición. Para funcionar sólo necesita alimentación GND, VCC (5 voltios) y una señal de control.



Los servomotores estándar sólo pueden girar 180°, aunque en el mercado podemos encontrar de 270° y de 360° (giro continuo).

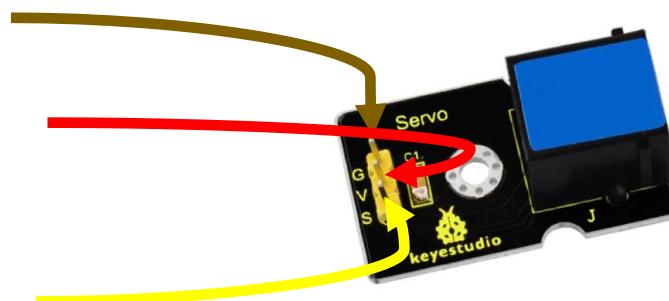
The screenshot shows the ArduinoBlocks software interface. On the left, a sidebar lists categories: Lógica, Control, Matemáticas, Texto, Variables, Listas, Funciones, Entrada/Salida, Tiempo, Puerto serie, Bluetooth, Sensores, and Actuadores. The 'Actuadores' category is selected and highlighted in orange. On the right, several blocks are displayed:

- Zumbador D5 Ms 500 Hz 1000
- Tono (Hz) DO
- Zumbador D5 Reproducir RTTTL "song: "
- RTTTL The Simpsons
- Girar izquierda
- Servo D5 Grados Ángulo 0° Retardo (ms) 100

En el menú “**Actuadores**” encontramos un bloque que se llama “**Servo**”. Con él tenemos la posibilidad de controlar el servomotor, indicando los grados de rotación (Ángulo) que queremos y el tiempo de retardo (Tiempo que tarda en ir de una posición a otra).



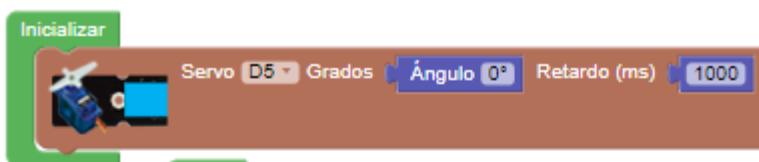
Para proceder a la conexión del servomotor a la placa Easy Plug debemos utilizar el adaptador fijándonos bien en como conectamos los tres cables. Si nos fijamos en el color de los cables, el marrón es el GND (G), el rojo el 5V (V) y el amarillo el control o señal (S).



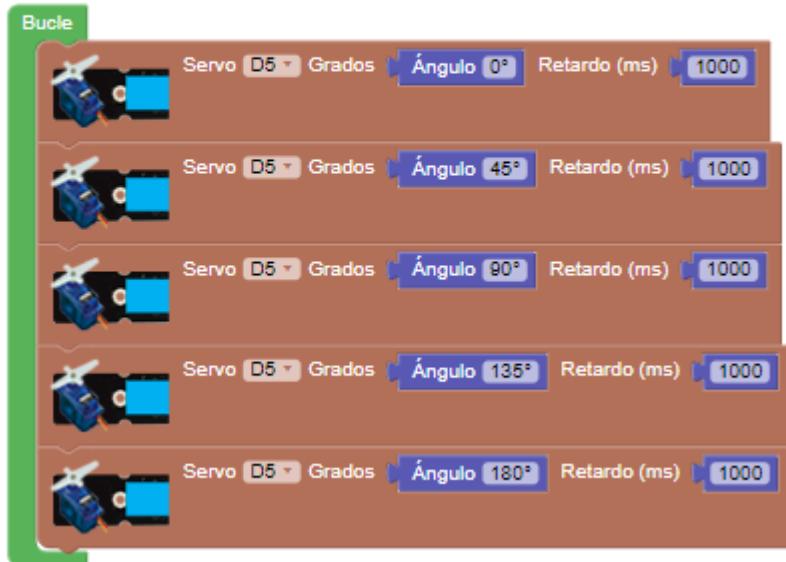
Una buena idea es programar, por primera vez el Ángulo a 0° para descubrir el punto de origen y a partir de aquí montar alguno de los accesorios que vienen con el servo para poder visualizar la rotación del eje. Vamos a conectar nuestro servo en el Pin D5 y cargar este programa.



Ahora ya podemos practicar con distintos grados, y observar donde apunta la “flecha” del eje.



Accesorios servo



Un servomotor nos puedes servir para accionar todo lo que se nos ocurra; para mover un brazo de un robot humanoide, palancas, flechas indicadoras, ... las posibilidades son muchas.

Prueba a cambiar de ángulo y de tiempos de retardos. Pero recuerda; **tienes que dar tiempo suficiente para que el servo realice el movimiento por completo antes de cambiar a otro ángulo.**



A07. – Sensor infrarrojo para evitar obstáculos.

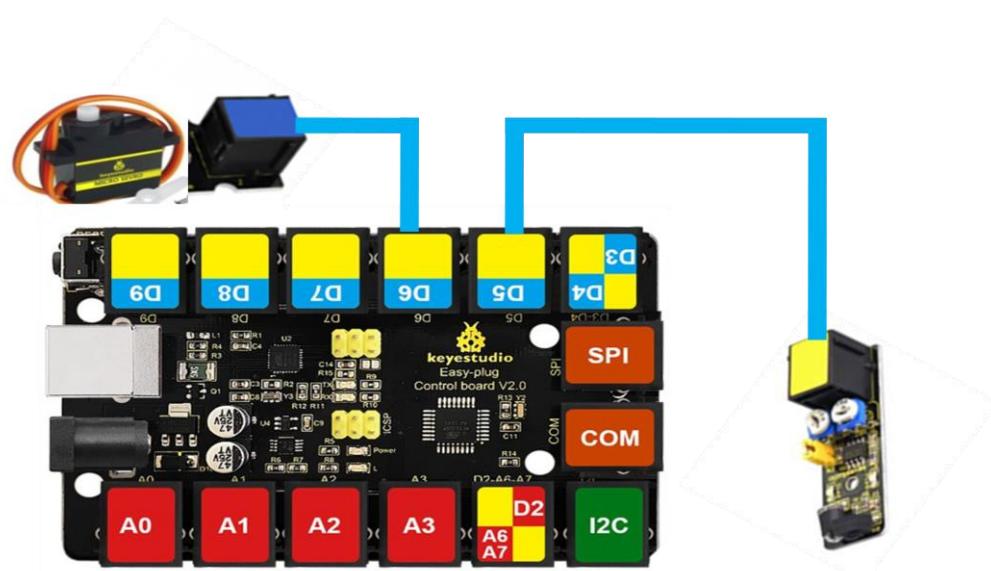
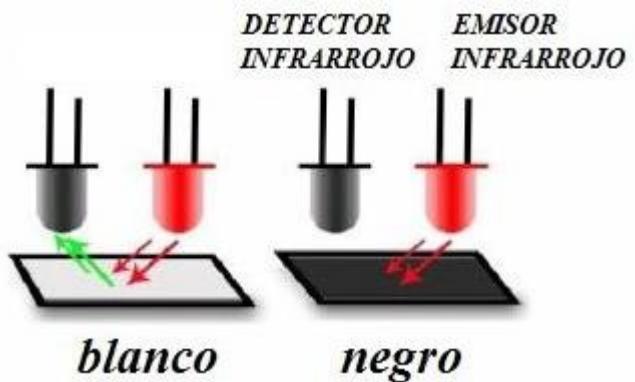
El principio de funcionamiento de un sensor de infrarrojos es utilizar la diferente reflectividad de la luz infrarroja para el color y convertir la intensidad de la señal reflejada en una señal de corriente.

El sensor de infrarrojos tiene un emisor de infrarrojos y un fototransistor que recibe la luz reflejada. En el caso de que el color sea negro, este absorbe todo el espectro de luz por lo que no refleja nada y el fototransistor no recibirá nada. En caso del color blanco ocurre lo contrario, sí refleja la luz por lo que el fototransistor la recibirá. Durante el proceso de detección, el negro está activo en el nivel ALTO, pero el blanco está activo en el nivel BAJO. La distancia de detección oscila entre 0,5 a 3 cm. El sensor dispone de dos resistencias variables con las que podemos ajustar su sensibilidad.



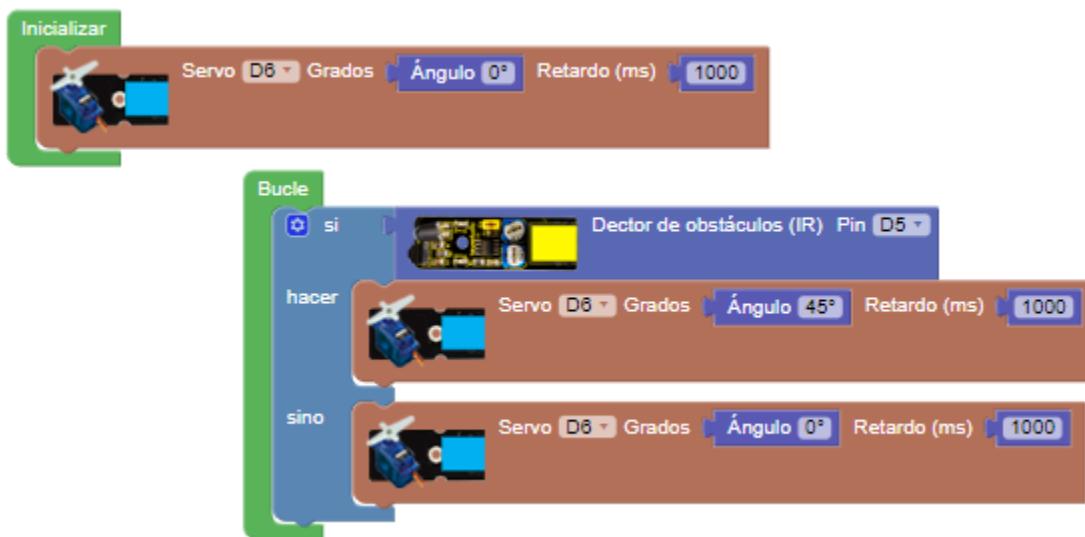
En el sensor existe un pequeño led rojo que se enciende cuando el sensor detecta color blanco y se apaga con el color negro.

Visto de otra manera el color negro es lo mismo que no detectar nada delante (el fototransistor no recibe luz) y gracias a eso vamos a realizar la siguiente actividad. Imaginemos la misma cinta transportadora de la actividad del sensor de choque, pero en esta ocasión queremos que cuando un objeto que va por ella sea detectado por el sensor, un brazo, accionado por un servo, lo mueva y lo desvíe a otra cinta. Vamos con el esquema de conexionado; conectamos el sensor infrarrojo en el Pin D5 y el servomotor en el Pin D6.



(A estas alturas ya habréis comprendido el porqué de los colores en los pines: el Amarillo son para entradas de sensores digitales y el Azul para salidas de actuadores. El resto de los colores los iremos viendo.)

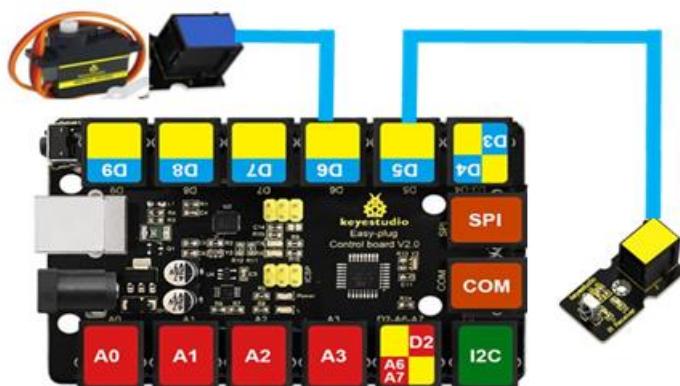
Vamos con el programa:



Es una buena recomendación inicializar los programas colocando el servo a su posición inicial en el programa, en este caso a un ángulo de 0° . Cuando el sensor IR detecte un objeto el servomotor girará 45° desplazando el objeto detectado a otra cinta o lugar.

A08. – Receptor infrarrojo IR.

Un receptor infrarrojo IR es un sensor que recibe la señal de por ejemplo un mando a distancia, que es un dispositivo de control que emplea un LED infrarrojo para enviar una señal al receptor. Usaremos un receptor infrarrojo IR para recibir esta señal y pueda ser detectada para controlar un autómata o procesador como Arduino. Por ejemplo, hubiéramos podido usar un mando para controlar los ángulos del servo del ejercicio anterior.

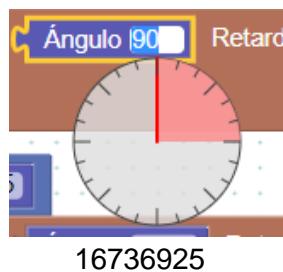
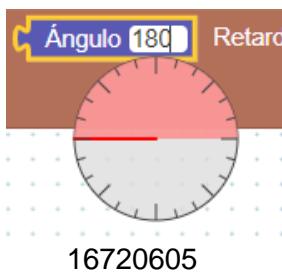


Primero haremos un pequeño programa para leer los valores de las teclas de las flechas de un mando universal o en este caso, uno de Keyestudio:



¡Importante añadir un tiempo después de usar la función “**Enviar**” para no saturar y bloquear la conexión!

A través de la consola podemos ver los valores que nos está detectando el receptor IR al pulsar las flecha en el siguiente orden: arriba, abajo, derecha e izquierda. Como el servo es de 180°, usaremos la flecha derecha para 0°, flecha arriba para 90° y flecha izquierda para los 180° (descartando flecha hacia abajo...):



Y ahora, conociendo los valores de las teclas, crearemos un programa donde la flecha de arriba serán 90° , derecha serán 0° e izquierda los 180° .



90°



180°

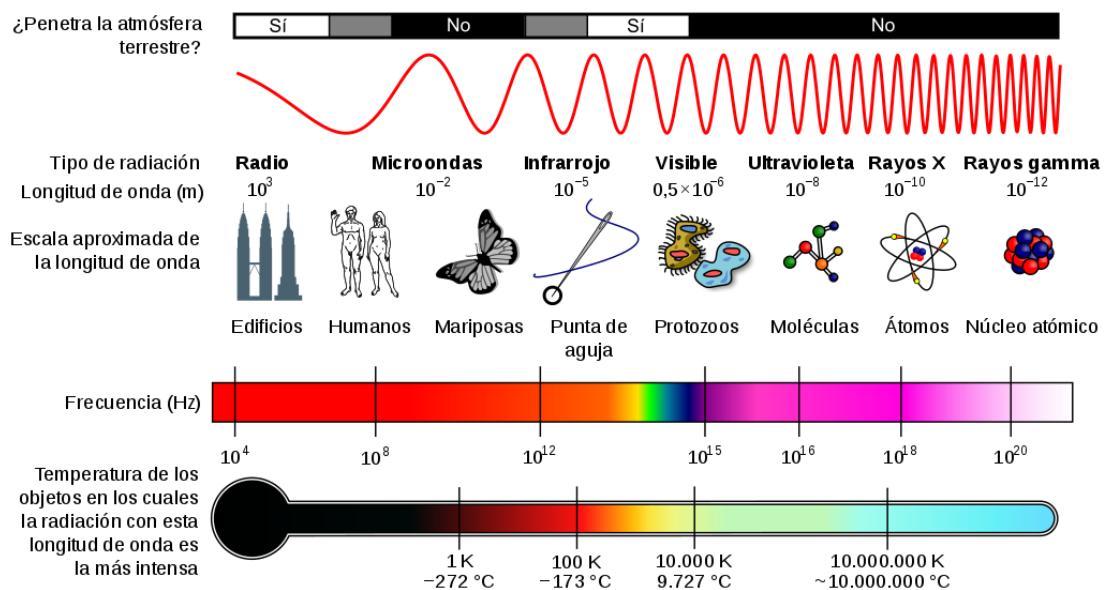
0°



A09. – Sensor de llama y robot bombero.

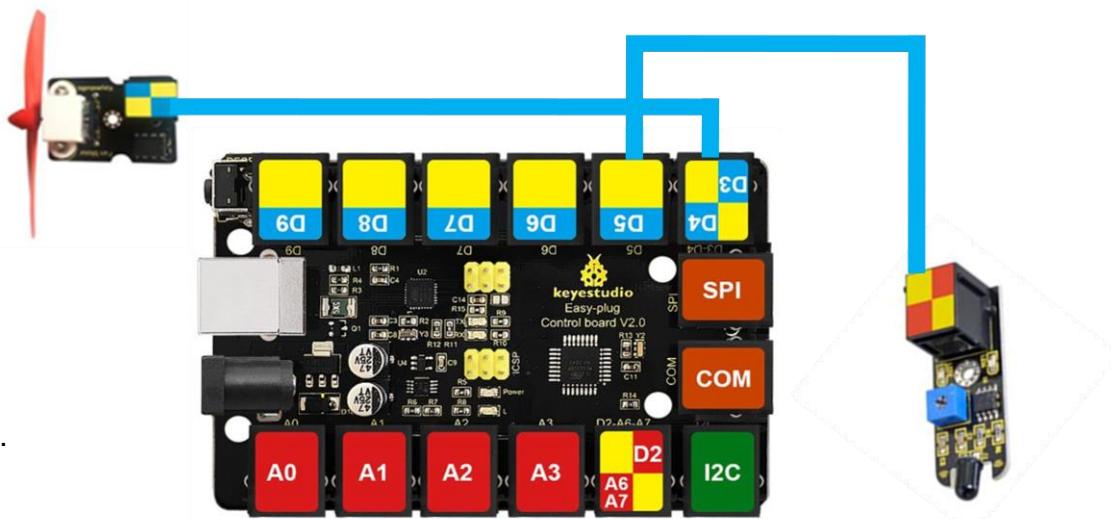
Ahora una actividad muy divertida, vamos a hacer un robot bombero, cuando el sensor detecte fuego un ventilador se pondrá en funcionamiento y lo apagará. Necesitaremos el sensor de llama y el módulo de motor con hélice.

El sensor de llama detecta longitudes de onda de 760 nm a 1100 nm de luz. Este rango de longitudes de onda son las que emiten en general cualquier cuerpo en llamas.

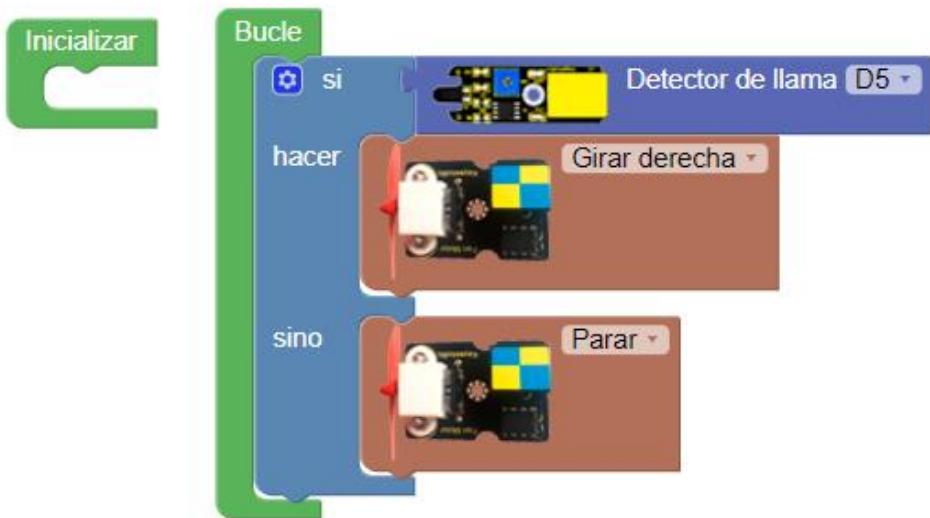


https://es.wikipedia.org/wiki/Radiaci%C3%B3n_electromagn%C3%A9tica

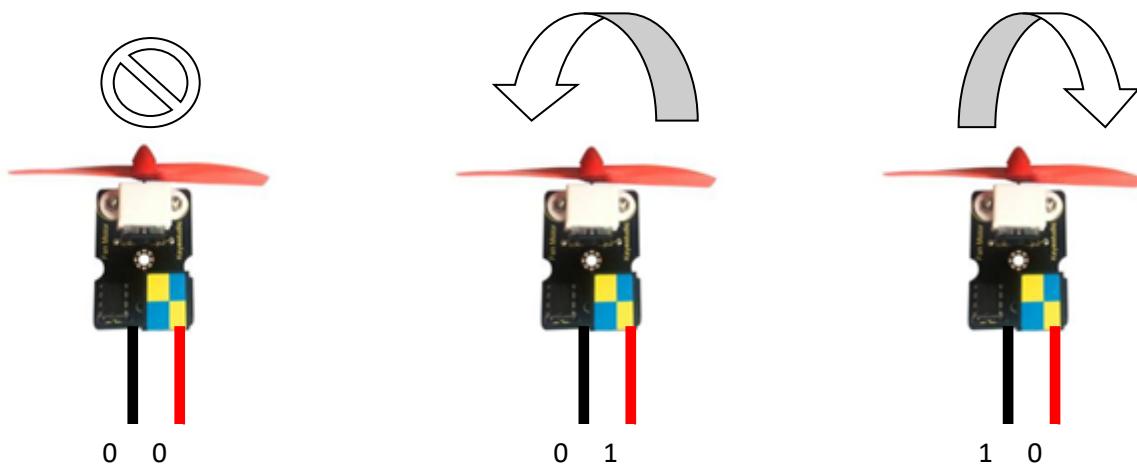
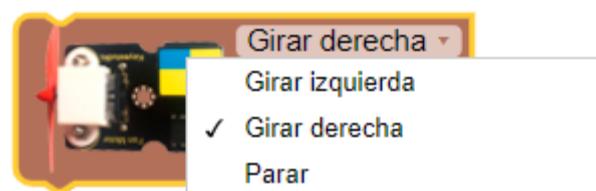
Realiza el siguiente montaje, conectando el sensor de llama al Pin D5 y el motor con hélice al Pin D3/D4. *El módulo del motor-hélice sólo se puede conectar a este Pin doble para que funcione correctamente.*



El programa que vamos a ejecutar es el siguiente:



Fíjate como el bloque del motor-hélice no tiene opción de seleccionar ningún Pin en su programación. Eso es debido a que sólo pudo ser conectado al Pin doble D3/4 y tiene una explicación. El motor puede tener tres posiciones girando hacia la derecha, girando hacia la izquierda o parado, y eso solamente se puede conseguir conectándolo a dos pines (D3 y D4), para ello el módulo lleva un **"driver"** incorporado para gestionar los giros.



A10. – Potenciómetro deslizante.

En esta ocasión vamos a utilizar el potenciómetro deslizante de nuestro kit y para ver sus valores utilizaremos el puerto serie.

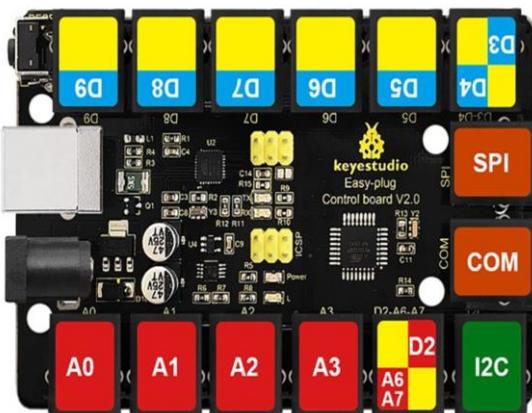
¿Qué es un potenciómetro? Los potenciómetros también se llaman resistencias variables ya que son un tipo de resistencias especiales que tienen la capacidad de variar su valor cambiando de forma mecánica su posición. Son muy utilizados para el control de volumen o brillo de muchos aparatos electrónicos.



En primer lugar, fijaros en el color del pin del sensor. Es de color rojo, eso nos indica que se trata de un sensor **analógico** y como tal, lo deberemos conectar a las entradas analógicas de la placa Easy Plug. [ArduinoBlocks](#) sólo nos permite conectar este sensor en el Pin A6/7.

La diferencia entre un sensor **analógico** y uno **digital** es que mientras este último, el digital, sólo permite dos tipos de entradas, 0 o 1, alto o bajo, high o low, on u off, ... un sensor analógico puede tener infinidad de valores. En Arduino las entradas analógicas pueden tener 2^{10} valores, es decir, valores comprendidos entre 0 y 1023.

Realiza la conexión del sensor tal y como está en la imagen, conectando el sensor en la entrada analógica A6/7.



Para hacer una lectura de los valores del sensor es necesario utilizar la consola serie que nos ofrece [ArduinoBlocks](#), vamos a ver como se hace.



En primer lugar, generamos una variable a la que llamaremos "ValorPot". Vamos al menú "**Variables**" y seleccionamos

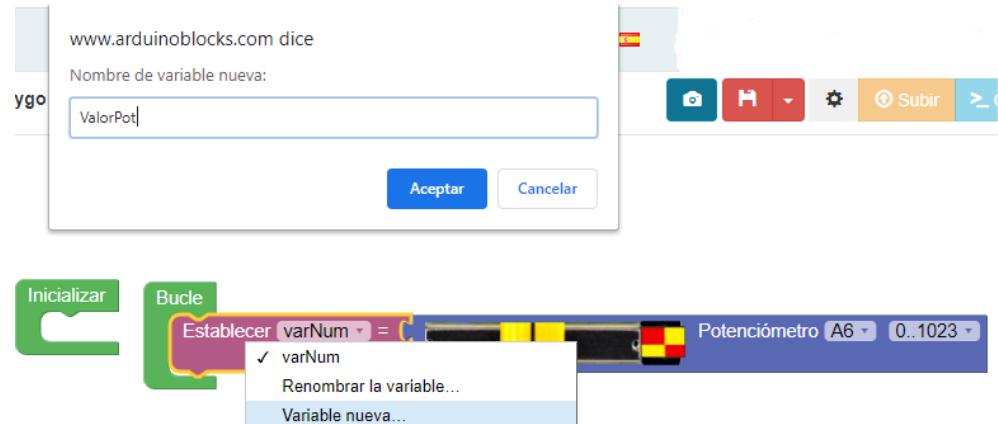
"Establecer..... =..."

The screenshot shows the ArduinoBlocks interface. On the left, there is a palette titled "Variables" which includes categories like Lógica, Control, Matemáticas, Texto, Variables, Listas, Funciones, Entrada/Salida, Tiempo, Puerto serie, Bluetooth, and Sensores. In the main workspace, there is a vertical stack of blocks:

- Establecer varNum = 0
- varNum
- Establecer varTexto = " "
- varTexto
- Establecer varBool = On
- varBool

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

Para cambiar el nombre de la variable clicaremos sobre el menú desplegable del bloque de la variable y elegiremos “**Variable nueva...**” nos aparecerá una ventana en la que escribiremos el nuevo nombre y daremos aceptar.



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

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

The screenshot shows the ArduinoBlocks interface with the 'Puerto serie' library selected in the left sidebar. The workspace contains the following sequence of blocks:

- An 'Iniciar Baudios 9600' block.
- A 'Fijar timeout 1000' block.
- A 'Enviar " " Salto de línea' block.
- A 'Enviar byte 0' block.
- A '¿Datos recibidos?' block.
- A 'Recibir texto Hasta salto de línea' block.
- A 'Recibir byte' block.
- A 'Recibir como número Hasta salto de línea' block.

To the right, there is a note: "Fíjate como queda el programa:" followed by two hand icons pointing towards the workspace. Below this, the final program structure is shown:

```

    Iniciar
      -> Iniciar Baudios 9600
    Bucle
      -> Establecer ValorPot = Potencímetro A6 0..1023
      -> Enviar ValorPot Salto de línea
      -> Esperar 1000 milisegundos
  
```

Dentro del bloque “Enviar...” debemos colocar el valor de la variable creada “ValorPot”.



Es importante que después de “Enviar” pongamos un tiempo para no enviar demasiadas veces el valor pudiendo llegar a bloquear la comunicación.

Sube el programa y después clica sobre el botón de “Consola”.



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

ArduinoBlocks :: Consola serie

Baudrate: 9600

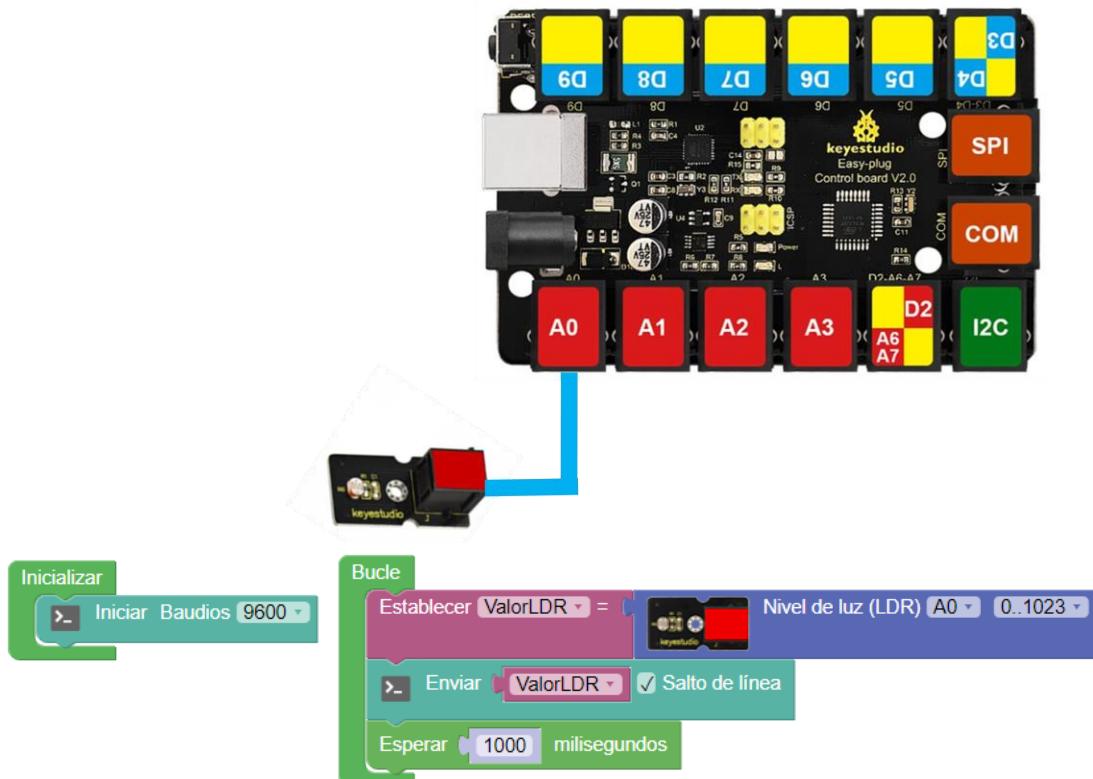
281.00
343.00
383.00
418.00
428.00
428.00
428.00
428.00

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

A11. – Lectura del valor de la fotocélula (LDR).

Ahora que ya sabemos usar el “**Puerto Serie**” para leer los valores de los sensores, vamos a utilizarlo para ver el valor de una fotocélula (LDR). ¿Qué es una LDR? Una LDR es un resistor que varía su valor de resistencia eléctrica dependiendo de la cantidad de luz que incide sobre él.

Conectemos nuestra LDR al Pin A0 como muestra la imagen y realiza el siguiente programa:



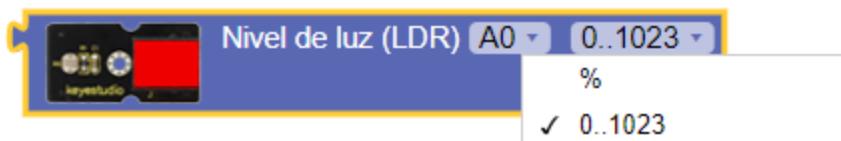
Conecta la consola serie para ver sus valores. Cambia la luz que incide sobre la LDR, por ejemplo, poniendo la mano encima o iluminando con la linterna del móvil, y observa como varían.

ArduinoBlocks :: Consola serie

Baudrate: 9600 ▾ Conectar Desconectar Limpiar

Enviar

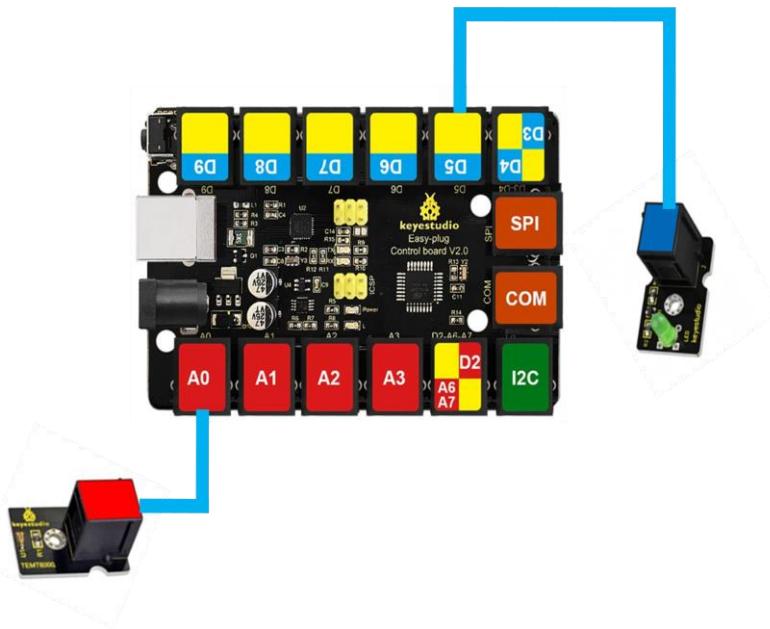
37.00
37.00
37.00
37.00



A12. – Encendido de una farola. Sensor luz ambiental.

¿Te has preguntado alguna vez quién enciende las farolas de toda tu ciudad cuando se hace de noche? Pues seguramente nadie, seguramente se enciendan solas gracias a sistemas de control programados como el que vamos a hacer a continuación.

Para ello necesitamos un LED y el sensor de luz ambiental TEMT6000. Conéctalos a la placa Easy Plug de la siguiente manera, el sensor de luz en el Pin A0 y el LED en el Pin D5.



Al igual que en la actividad anterior, lo primero necesitamos leer los valores que registra nuestro sensor para después usarlos como umbral en el siguiente programa.

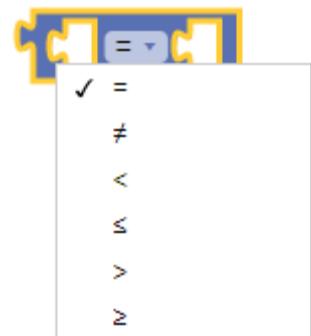


En esta ocasión la lectura de los valores la vamos a realizar en %. Después abrimos la consola serie de **ArduinoBlocks**, conectamos y hacemos una lectura de diferentes valores, una de la luz ambiental y otra tapando el sensor con la mano. Tomaremos esta última lectura, que seguramente sea por debajo del 20% y la usaremos para realizar el siguiente programa.

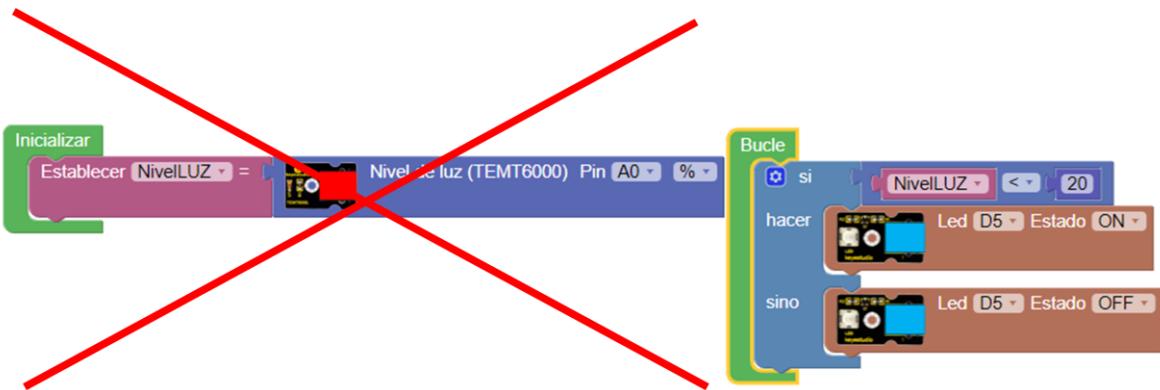


Acabamos de introducir una condición en la cual “**si algo es menor que**” realiza una acción. Este nuevo bloque de igual, diferente, menor, mayor, menor o igual, mayor o igual se encuentra dentro del menú “**Lógica**”.

Para dar el valor de 20 necesitamos el bloque numérico que se encuentra dentro del menú matemáticas.



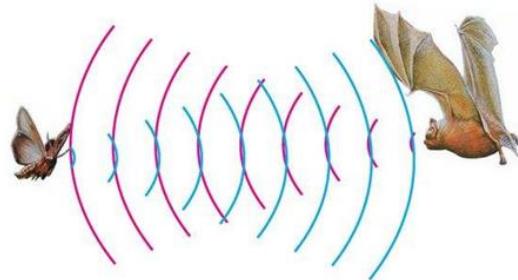
Recuerda: No debes poner el establecimiento de la variable en “**Inicializar**”.



A13. – Sensor de Ultrasonidos I

El kit **Keyestudio Easy Plug Secundaria** dispone de un sensor de ultrasonidos. Al igual que los murciélagos tienen muy poca vista y se guían por ultrasonidos, tú puedes hacer diferentes proyectos utilizando este sensor.

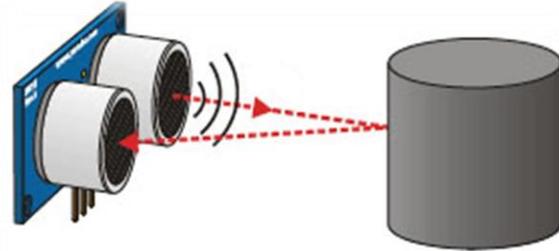
El modo de funcionamiento de este sensor es muy sencillo, uno de los dos “ojos” emite ondas ultrasónicas (no perceptibles para el oído humano) después de la señal de disparo. Cuando estas ondas ultrasónicas encuentran un objeto chocan y se reflejan como eco. Al volver son recibidas por el otro “ojo”, por lo que se puede determinar la distancia del objeto a partir de la diferencia de tiempo entre la señal de disparo y la señal de eco. Conociendo la velocidad del sonido, que es de 340 m/s, y sabiendo el tiempo transcurrido entre la emisión del sonido y su recepción, calcularemos muy fácilmente la distancia del objeto.



Como todo en la vida, este sensor tiene sus limitaciones, no podremos localizar objetos a distancias mayores de 3 metros ni a distancias menores de 2,5 cm. Aun así, el rango de valores es más que suficiente para nuestros propósitos.

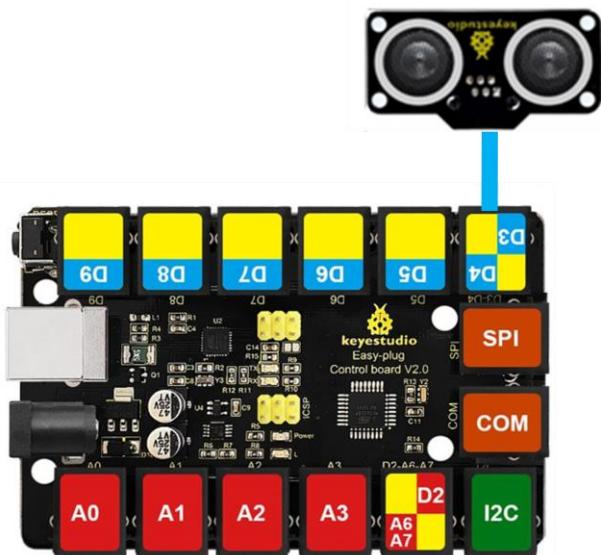
Nuestro sensor de ultrasonidos **sólo** lo podemos conectar al Pin doble **D3/D4**. Esto se debe a que para el funcionamiento del sensor necesita dos Pines, uno para emitir el ultrasonido (el **Pin Trig**) y otro para recibirlo (el **Pin Echo**).

Por ese motivo en **ArduinoBlocks** no da opción de seleccionar el Pin en el que se quiere conectar.

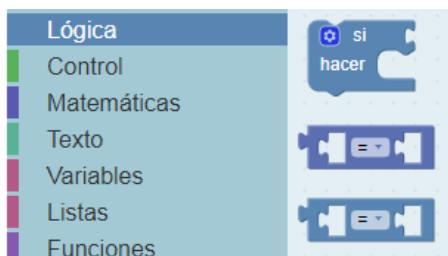


$$\text{Tiempo} = 2 * (\text{Distancia} / \text{Velocidad})$$

$$\text{Distancia} = \text{Tiempo} \cdot \text{Velocidad} / 2$$



En la siguiente actividad vamos a utilizar el sensor de ultrasonidos para que cuando detecte un objeto a una distancia menor de 25 cm se encienda un LED. (Conectaremos el LED en el Pin D5).

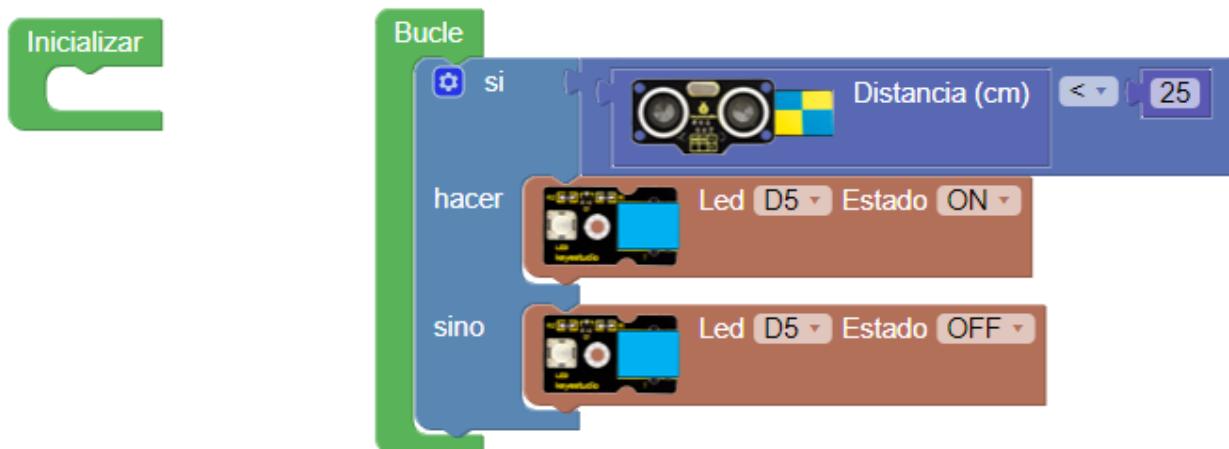


Dentro del menú de “**Matemáticas**” necesitaremos usar el bloque “**Número**” para dar el valor de los 25 cm.

Utilizaremos nuevamente el bloque condicional para realizar esta actividad. En esta ocasión usaremos el “**Menor que**”.



Bien, con todo esto el programa quedaría de la siguiente manera:



A14. – Sensor de Ultrasonidos II

En esta nueva actividad vamos a utilizar nuevamente el “**Puerto Serie**” para poder visualizar por la pantalla del ordenador los datos que nos envía el sensor de ultrasonidos y vamos a usar un nuevo bloque del menú “**Tiempo**” que es la función “**Ejecutar cada**”.

Vamos a aprovechar la función “**Ejecutar cada**” junto con la función “**Enviar**” para ver los datos del sensor de ultrasonidos en la pantalla del ordenador.

Pero antes de hacer el ejercicio vamos a ver las distintas funciones poco a poco.

- Gestión de tiempos con el bloque “**Ejecutar cada**”.



Encontramos el bloque en el menú “**Tiempo**”. Este bloque ejecuta una vez cada X tiempo las órdenes que estén dentro de él. La gran diferencia con bloque “**Esperar**” es que **NO** detiene el programa en el estado anterior a él, como si hace el bloque “**Esperar**”.



Dentro de menú “**Puerto serie**” tenemos el bloque “**Enviar**”, que es la función que utilizaremos para imprimir datos enviados desde la placa de control al ordenador.



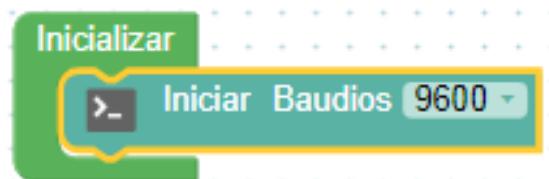
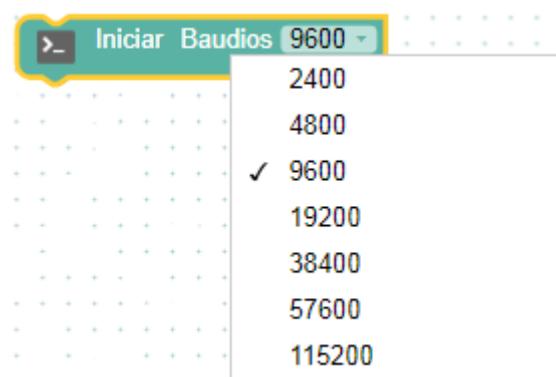
Podemos enviar cualquier orden o dato que queramos solo con introducirlo en el espacio en blanco entre comillas del bloque.



Lo primero que debemos hacer es inicializar la comunicación entre la placa Arduino y el ordenador. Utilizaremos el siguiente bloque:

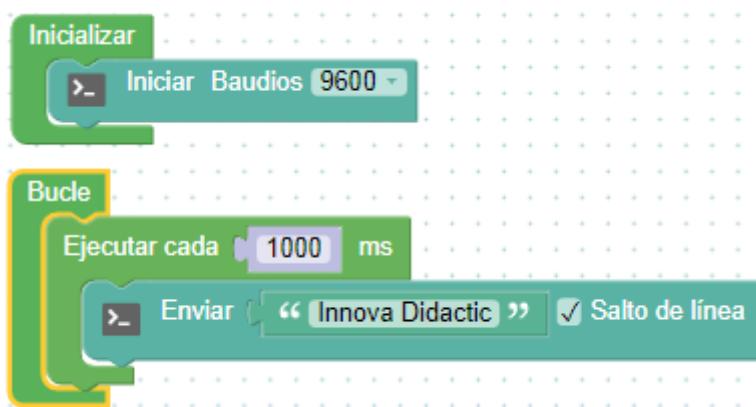
Por defecto lo dejaremos a la velocidad de 9.600 Baudios.*

*El baudio (en inglés baud) es una unidad de medida utilizada en telecomunicaciones, que representa el número de símbolos por segundo en un medio de transmisión digital.¹ Cada símbolo puede comprender 1 o más bits, dependiendo del esquema de modulación.



Este bloque lo colocaremos dentro de la función “**Iniciarizar**”.

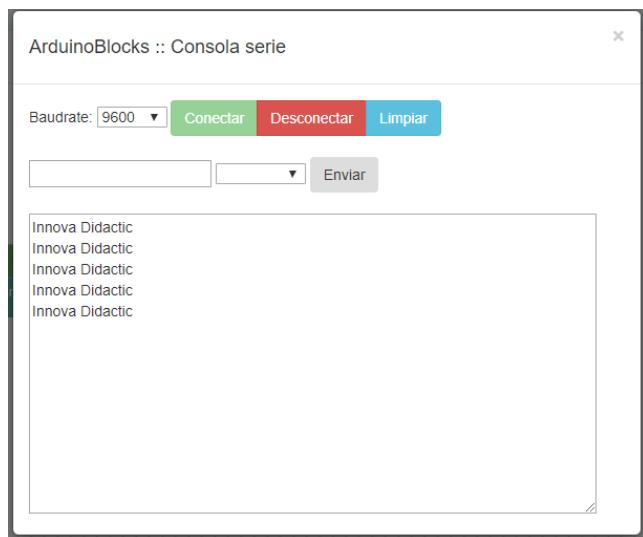
Bien, con todo lo visto por el momento, realiza el siguiente programa de ejemplo:



Con este programa lo que hacemos es enviar a la pantalla del ordenador la frase “**Innova Didactic**” una vez por segundo (1000 ms).

Tras “**Subir**” y cargar el programa, pulsa el botón “**Consola**”, se abrirá una nueva ventana en la que clicaremos sobre el botón “**Conectar**”, veremos en la pantalla del ordenador los valores enviados.





Hay que destacar que, si no usamos el bloque "**Ejecutar cada**", enviamos al ordenador un dato cientos de veces por segundo, lo que satura la comunicación y bloquea el sistema. De ahí la importancia de este bloque.

También podríamos usar el bloque "**Esperar**" para enviar datos cada segundo, pero entonces, en los tiempos de espera, la placa no podría escuchar otras órdenes ni realizar otras acciones. Tranquilos, veremos todo esto más adelante con calma... ¡y con más ejemplos!

Ahora vamos a aplicar estos dos conceptos para leer los valores del sensor de ultrasonidos.

Para hacerlo vamos a utilizar el bloque del sensor de ultrasonido como dato para enviar a la pantalla del ordenador.



El programa quedará de la siguiente manera:



ArduinoBlocks :: Consola serie



```
14.66
15.03
12.83
12.86
15.76
18.78
19.05
18.17
14.88
18.38
20.53
7.86
```

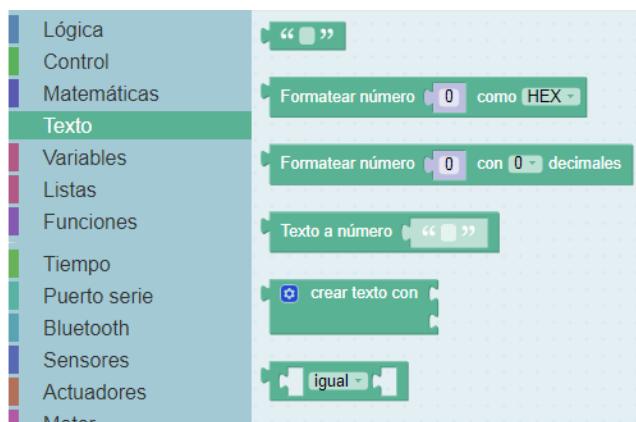
Introducimos el siguiente programa y abrimos la consola:

Vemos como al colocar la mano delante del sensor e ir cambiando su distancia, el valor que aparece en la pantalla va variando. Los valores que aparecen son directamente en cm.

Fíjate en la velocidad del **Baudrate**, por defecto es 9.600, es importante que sea la misma que tenemos en el programa al “**Iniciar**”.

Para que quede un poco más bonito y aparezca “cm” al lado del número que indica la distancia debemos ir al menú “**Texto**” y seleccionar el bloque “**Crear texto con**”.

Y el bloque para escribir “**Texto**”:



Con estos dos nuevos bloques modificamos el programa anterior quedándonos de la siguiente manera:



El resultado es el siguiente:

ArduinoBlocks :: Consola serie

Baudrate: 9600 ▾ Conectar Desconectar Limpiar

▾ Enviar

```
13.19 cm
13.19 cm
13.21 cm
13.21 cm
13.21 cm
13.21 cm
14.22 cm
31.05 cm
16.26 cm
21.21 cm
```

A15. – Control de temperatura de una habitación.

En la siguiente actividad vamos a controlar la temperatura de una habitación utilizando el sensor de temperatura DS18B20 y vamos a hacer que cuando esa temperatura sobrepase cierto valor, el motor con hélice se ponga en funcionamiento para refrescar el ambiente.

El sensor de temperatura DS18B20 tiene un rango de temperatura de -55 a +125 °C y una resolución de 0.5 °C

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

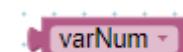
Las variables son elementos muy comunes en programación. Básicamente, crear una variable es darle un nombre a un dato o a una lectura. Por ejemplo, las mediciones de valores de temperatura las podemos guardar en una variable que se llame “Temperatura” o las del sensor de ultrasonidos en una llamada “Distancia”,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

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

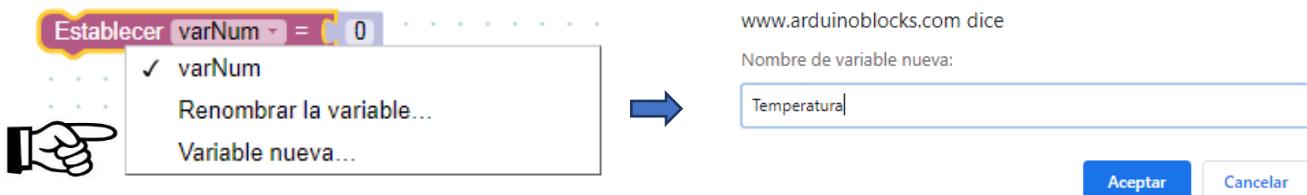
1. El bloque en el que le damos valor a la variable:



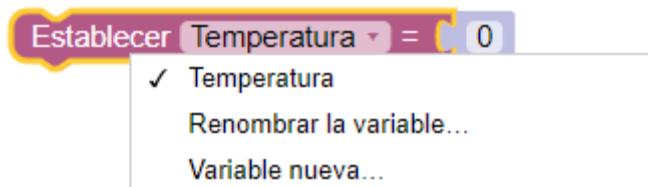
2. Y el bloque de la propia variable creada, para poder insertarla y combinarla con otros bloques:



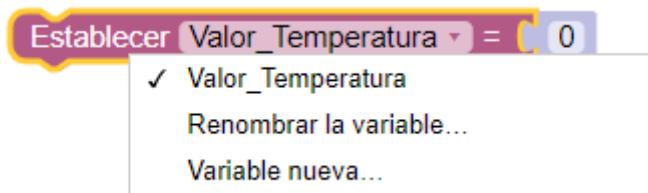
También podemos personalizar el nombre de la variable, de la siguiente forma:



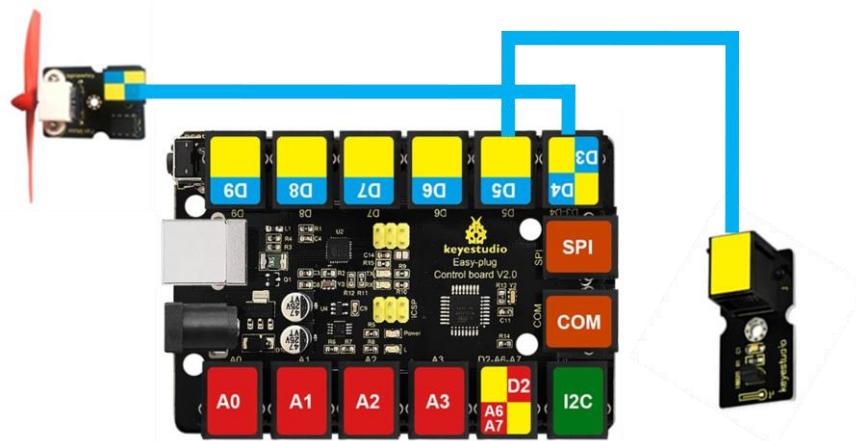
Una vez creada la nueva variable, podemos seleccionarla haciendo clic sobre el desplegable:



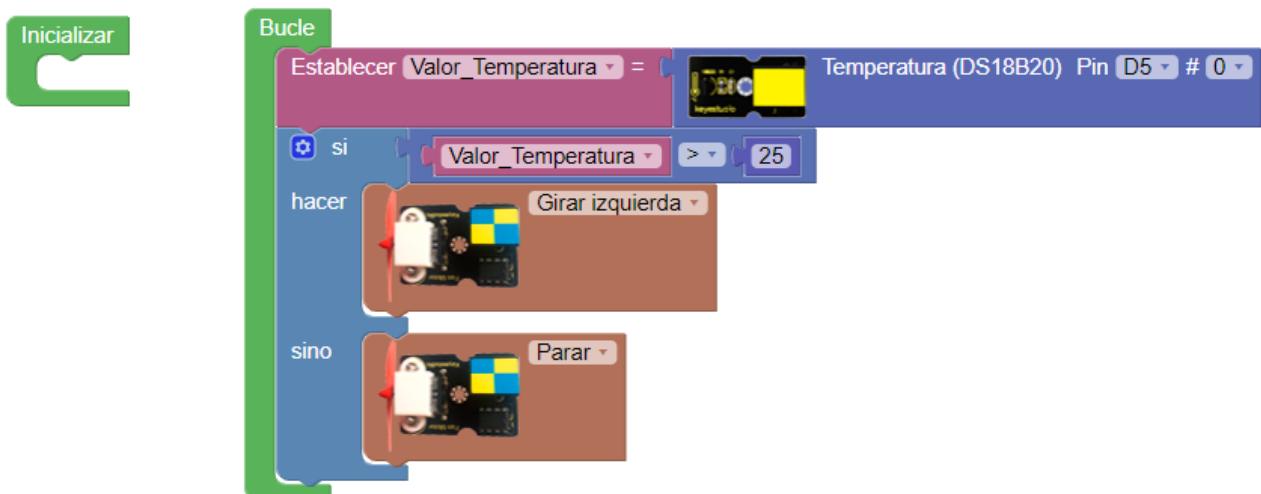
Ten en cuenta que las variables solo pueden estar formadas por una palabra. Si quieres incluir varias palabras, puedes usar el truco de separarlas con una barra baja “_”, como en el ejemplo, “**Valor_Temperatura**”.



Conecta el sensor de temperatura en el Pin D5 y el motor-hélice en el Pin doble D3/D4 tal y como muestra la imagen:



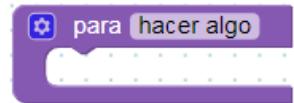
Ahora vamos a realizar un programa con el cual al pasar la temperatura de 25 °C el ventilador se pondrá en funcionamiento y parará cuando sea menor.



A16. – Sensor de presencia y Funciones.

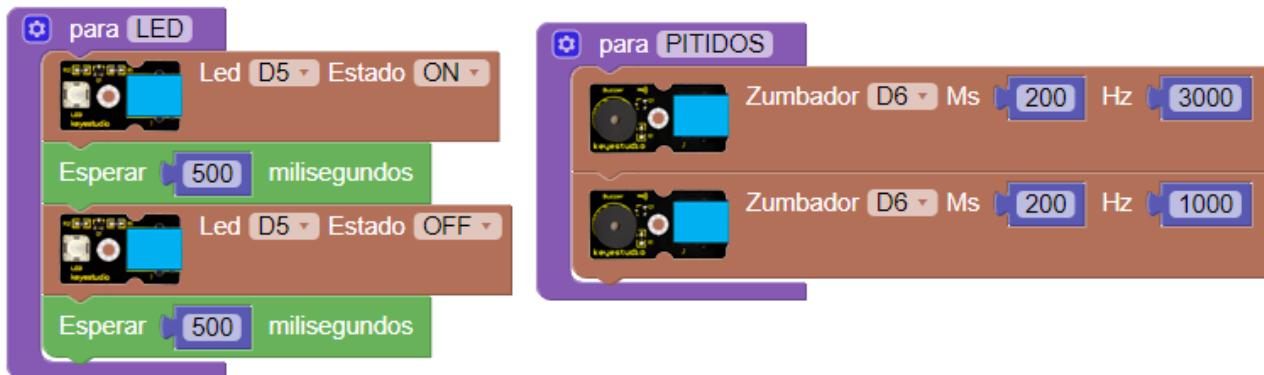
Una función es simplemente un conjunto de instrucciones a las que damos un nombre, para no tener que repetirlas en diferentes partes del programa y para clarificar y ordenar el código. Al crear una función, se genera automáticamente un bloque de dicha función, que ya puede ser insertado en cualquier parte del programa.

Para ello, tenemos que usar el bloque “**para**” que encontramos en el apartado “**Funciones**” de ArduinoBlocks.

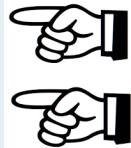
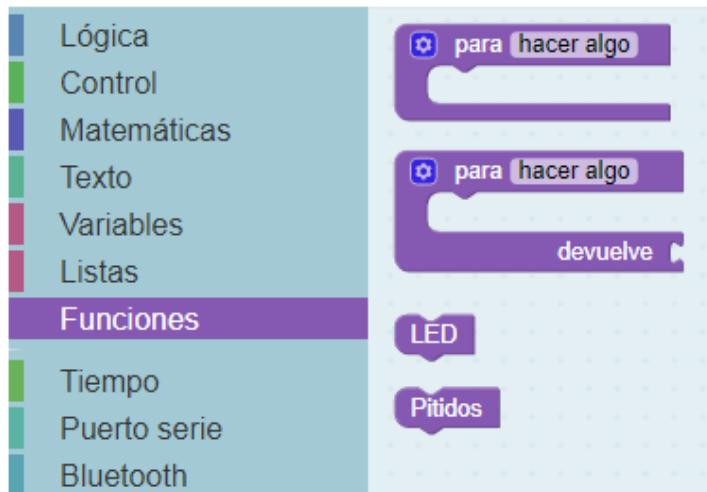


Primero tenemos que darle un nombre a la función, e incluir dentro de la misma, el conjunto de instrucciones que queremos que realice nuestra placa de control cada vez que la usemos.

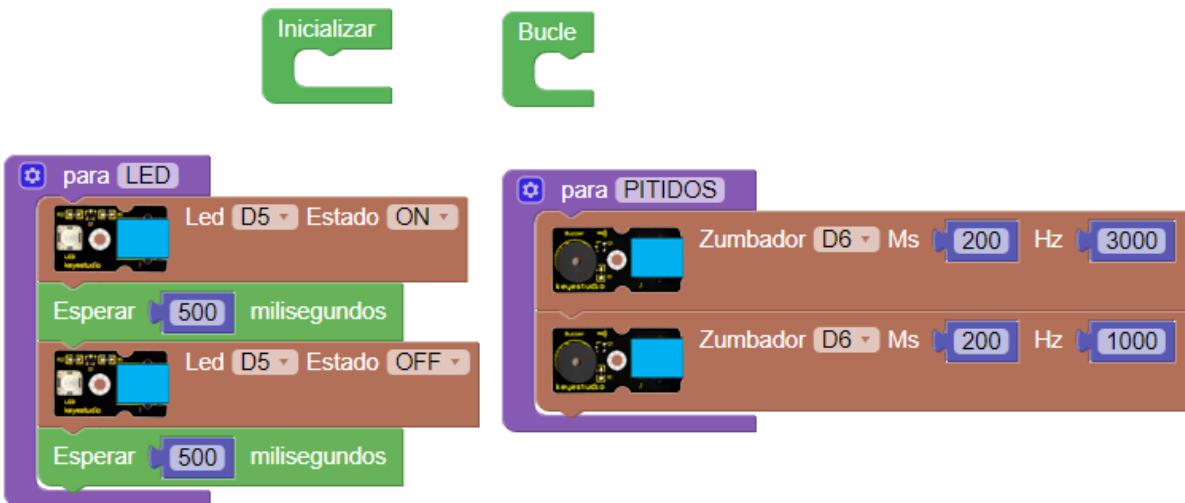
Por ejemplo, podemos definir dos funciones; la primera la llamaremos “LED” y en ella encenderemos y apagaremos el LED del Pin D5. La segunda función la llamaremos “PITIDOS” y su finalidad será activar el zumbador en el Pin D6 con dos pitidos distintos. Quedarían así:



Si ahora vamos a la sección “**Funciones**” del panel izquierdo de ArduinoBlocks, encontraremos ya las dos nuevas funciones creadas, disponibles para ser seleccionadas e insertadas en cualquier parte del código.



Es importante destacar que las funciones no se definen dentro del “**Bucle**”. Se crean fuera y luego se insertan dentro en los momentos en los que queramos que se ejecuten.



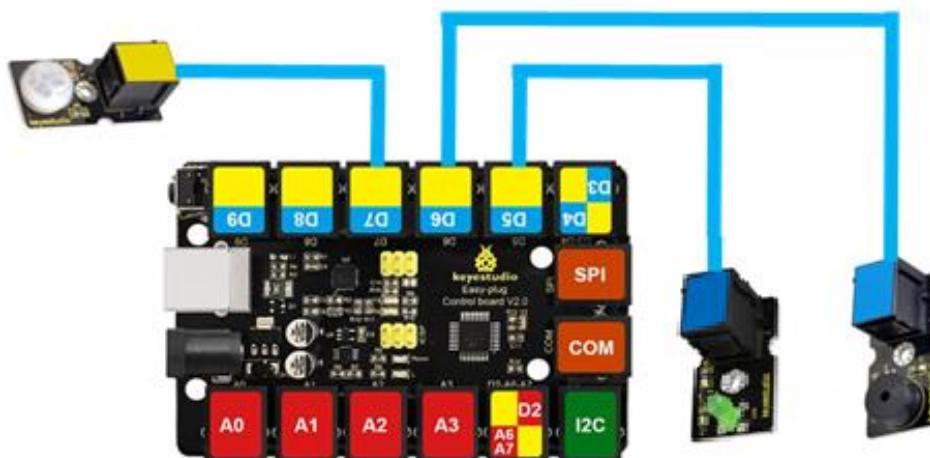
Cuando creamos un programa pequeño (de pocos bloques), no suele merecer la pena definir funciones, ya que no ahorra mucho tiempo. Pero cuando realizamos programas más extensos o con partes iguales que se repiten, es una buena práctica hacer uso de ellas, tanto por comodidad, como por claridad del programa realizado.

Con las dos funciones creadas vamos a hacer una alarma luminosa y acústica de detección de intrusos en una vivienda, necesitaremos también el sensor PIR. Un sensor PIR es un sensor de movimiento infrarrojo piroeléctrico que puede detectar señales infrarrojas de una persona o animal en movimiento.

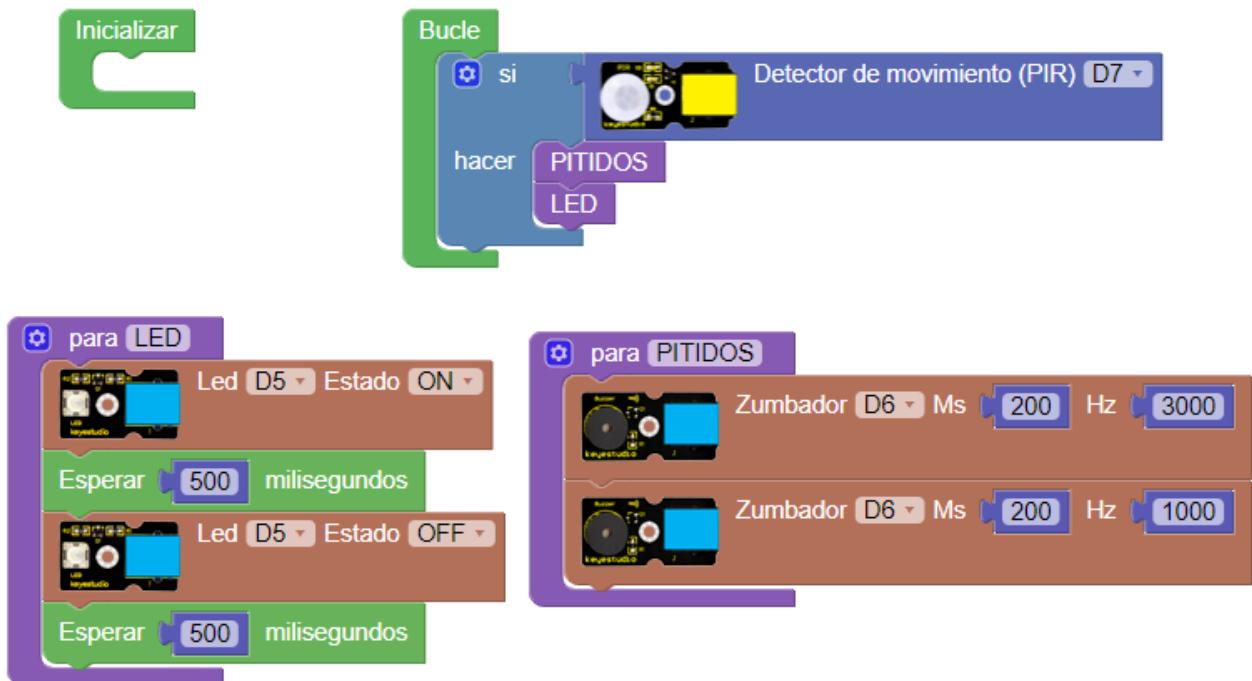


Realiza las conexiones de la placa de la siguiente manera:

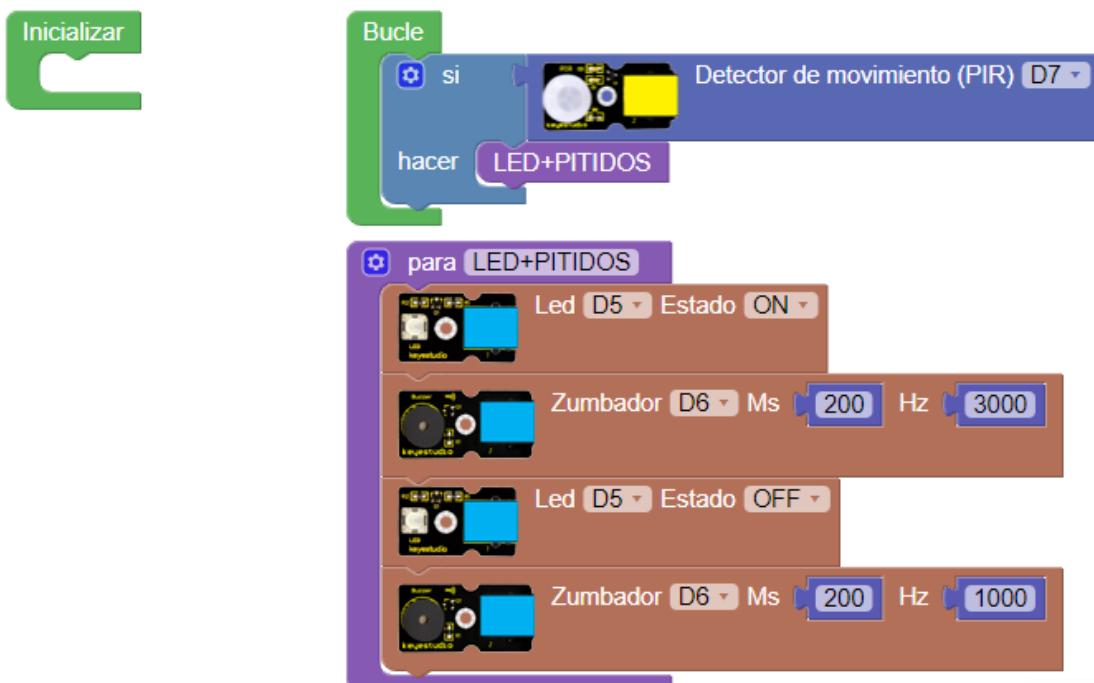
El LED en el Pin D5, el buzzer en el Pin D6 y el sensor PIR en el Pin D7.



El programa completo sería el siguiente:



Este ejemplo lo hemos hecho para entender cómo se hacen las funciones. Prueba a variar el programa y déjalo como el siguiente:



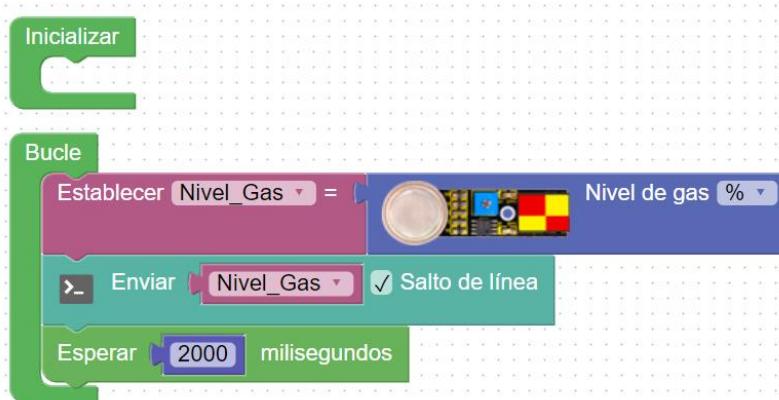
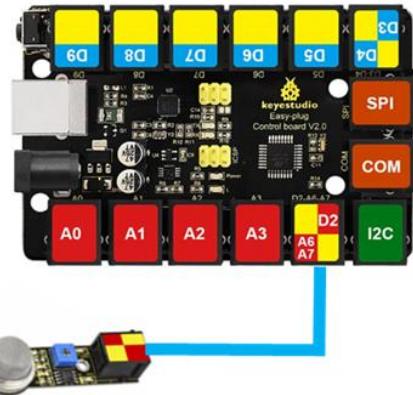
A17. – Sensor de gas y mediciones por Consola

Vamos a practicar la lectura de sensores analógicos, en esta ocasión usaremos un sensor que detecta distintos tipos de gases y enviaremos su lectura a la consola para ver como varían sus valores como reacción a distintos estímulos.

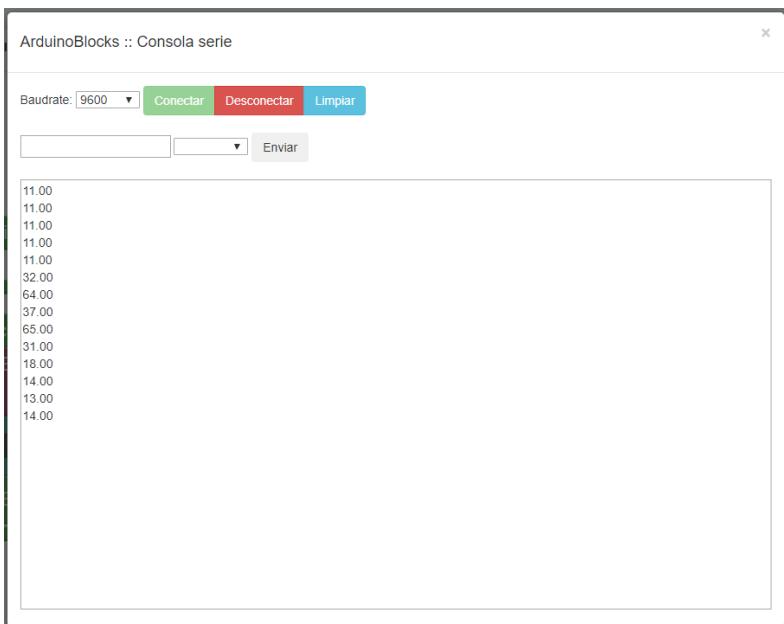
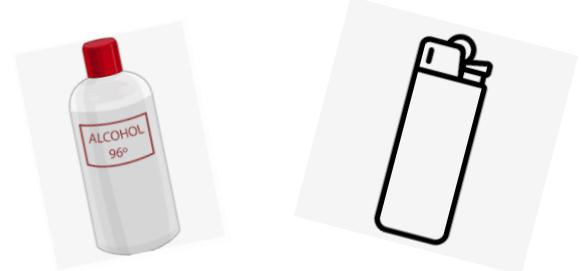
Conectaremos el sensor de gas al único conector disponible para su uso según la etiqueta de colores Amarilla-Roja.

Y crearemos un programa donde estableceremos una variable “Nivel_Gas” y le asignaremos el sensor.

Esta variable la enviaremos por consola para poder leer su valor.



Cuando lo tengas preparado, prueba a acercar el sensor algún estímulo que contenga “gas”, por ejemplo, acercándolo a la salida de una botella de Alcohol etílico 96º o la salida de un encendedor, dejando salir gas sin llegar a generar llama.



¿Ves cómo el valor aumenta al acercar y reduce al apartar?

En el siguiente ejercicio seguiremos usando sensores analógicos y lo vincularemos a un ejercicio muy importante en nuestros días, la calidad del aire, indicaremos su estado de una forma muy gráfica.

A18. – Calidad del aire y Matriz de Led I

Para realizar la siguiente práctica es necesario utilizar el sensor de calidad de aire y la matriz de Easy Plug LED 8x8 I2C.

La matriz de LED se conecta a través de un bus llamado I2C, es el único de color verde de la placa Easy Plug.

I2C es un protocolo para poder conectar varios dispositivos de forma muy fácil con un par de cables para los datos y otros dos para alimentación y tierra o GND.



En el kit hay un Hub que permite conectar más de un sensor o actuador a la vez al bus I2C. Por ejemplo, podríamos conectar la matriz de LED y la pantalla LCD en el Pin I2C utilizando el Hub de expansión o podríamos conectar varias matrices a la vez.

La matriz de LED 8x8 dispone de un total de 64 leds con los cuales podemos hacer infinidad de símbolos, caras, iconos, letras, números, ... hay cantidad de opciones ya prediseñadas y [ArduinoBlocks](#) da la opción de crearlas por ti mismo.

Como no podía ser de otra manera [ArduinoBlocks](#) tiene un menú específico para programar la matriz de LED 8x8 en modo **Easy Plug**.

The screenshot shows the ArduinoBlocks IDE interface. On the left, there is a vertical palette of blocks categorized by color: Lógica (blue), Control (green), Matemáticas (purple), Texto (teal), Variables (pink), Listas (dark purple), Funciones (light purple), Tiempo (light teal), Puerto serie (medium teal), Bluetooth (dark teal), Sensores (dark blue), Actuadores (brown), Motor (pink), Pantalla LCD (light green), Pantalla OLED (light blue), and **LedMatrix 8x8** (orange). A hand icon points towards the 'LedMatrix 8x8' block. To the right of the palette, several examples of I2C commands are shown as blocks:

- # 1 Iniciar I2C 0x70 v1
- # 1 Rotación 0°
- # 1 Limpiar
- # 1 Bitmap
- # 1 Bitmap Face normal
- # 1 Píxel X 0 Y 0 Led ON
- # 1 Línea X1 0 Y1 0 X2 7 Y2 7 Led ON

Buttons for 'Iniciarizar' and 'Bucle' are also visible on the right side of the workspace.

Lo primero que debemos hacer es “**Inicializar**” la LedMatrix 8x8. Podemos poner hasta 4 distintas utilizando el Hub de expansión por ello debemos indicar de cual se trata (1, 2, 3 o 4). Lo segundo que debemos hacer es dar la dirección que por defecto será 0x70 y por último hay dos modelos v1 y v2 (utiliza al v2, si ves que los iconos no están centrados cambia al v1).



Una vez inicializada la matriz de LED vamos con el “**Bucle**”. En esta ocasión haremos que salgan dos caras, una FELIZ y otra TRISTE con una espera de 1 seg. entre ellas.

Fíjate en la cantidad de iconos de los que dispones:

- Face normal
- Face happy
- Face sad
- Face angry
- Eye normal
- Eye medium
- Eye small
- Eye closed
- Eye look left
- Eye look right
- Eye look up
- Eye look down
- Eye angry left
- Eye angry right
- Eye sad
- Eye surprise
- Icon heart
- Icon user
- Icon clock
- Icon arrow up
- Icon arrow down
- Icon arrow left



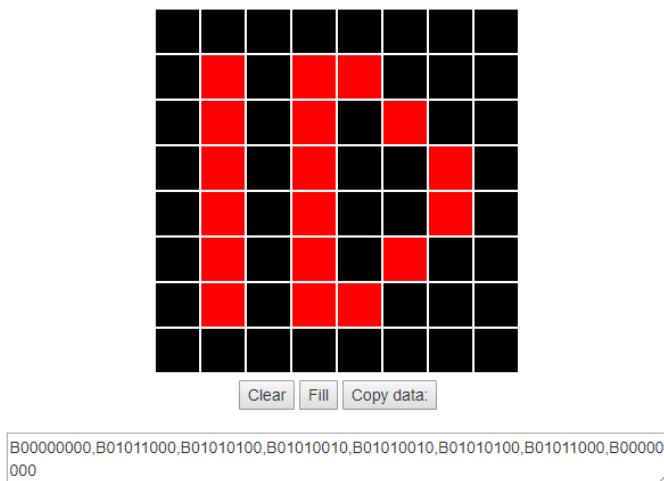
Prueba a cambiar y combinar unos cuantos de ellos cambiando también los tiempos de esperas.

Otro bloque muy curioso y con el que podrás crear tus propios iconos es el bloque de “**Bitmap**”. Vamos a ver cómo se utiliza.



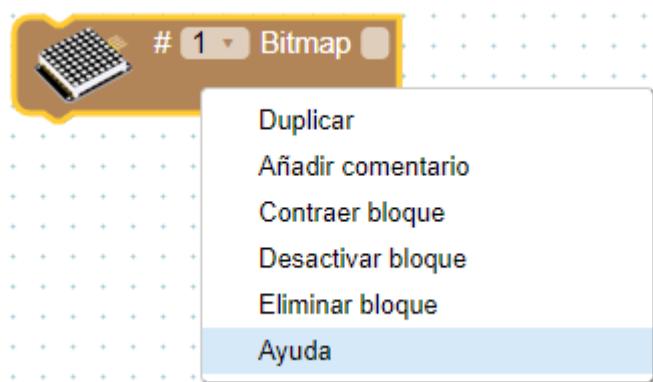
Seleccionando “**Ayuda**” con el botón derecho, se nos abre otra ventana/pestaña del navegador con la opción de dibujar lo que deseamos:

LedMatrix - Bitmap Data



Este es el resultado final.

También puedes hacer gif animados como por ejemplo el típico hombre andando del semáforo en verde de peatones. ¿Te animas?



Al hacer click sobre cada cuadradito negro se selecciona y pasa a color rojo. Una vez creado el dibujo que queramos, clicamos en “**Copy data:**” y volviendo a la ventana de programación sobre el bloque “**Bitmap**”, en el cuadro en blanco, le damos a “**Pegar**”.

Emoji	Win + Punto
Deshacer	Ctrl + Z
Rehacer	Ctrl + Mayús + Z
Cortar	Ctrl + X
Copiar	Ctrl + C
Pegar	Ctrl + V
Pegar como texto sin formato	Ctrl + Mayús + V
Seleccionar todo	Ctrl + A



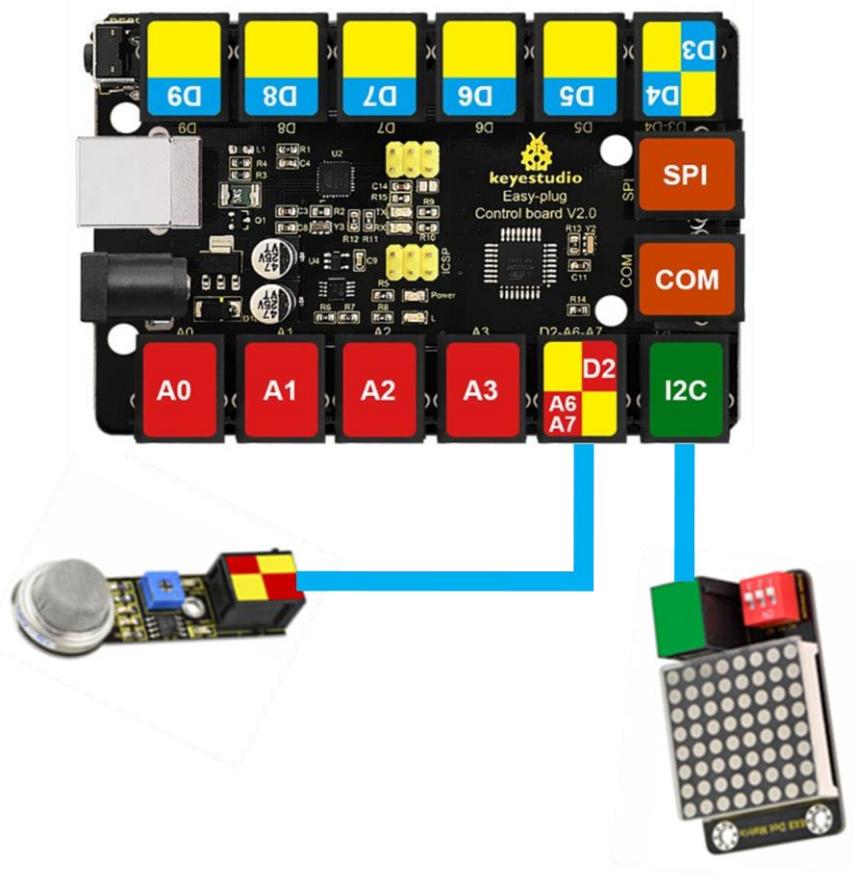
A18. – Calidad del aire y Matriz de Led II

Ahora que ya sabemos cómo funciona la matriz de LED vamos a hacer un dispositivo que nos mida la calidad del aire y que según sea, aparezca en la matriz de LED una cara sonriente (si la calidad es buena), cara normal (para valores medios) y una cara triste (si los valores son malos).

El sensor de calidad de aire que vamos a usar, el MQ-135, adopta el SnO₂ como material sensible a los gases ya que el SnO₂ tiene **baja conductividad eléctrica** en el aire limpio. Su funcionamiento es muy sencillo; según aumente la cantidad de contaminantes en el aire también aumentará su conductividad eléctrica y viceversa.

MQ-135 tiene una alta sensibilidad al amoníaco, sulfuro, vapor de benceno, humo y otros gases nocivos.

Realizamos la siguiente conexión de nuestro sensor y actuador; la matriz de LEDs 8x8 la conectaremos al Bus I2C mientras que por las especiales características del sensor de calidad del aire le conectaremos en el Pin múltiple A6/7 D2.

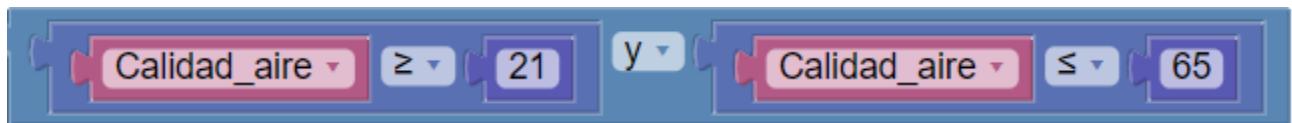


Ahora vamos con el programa:

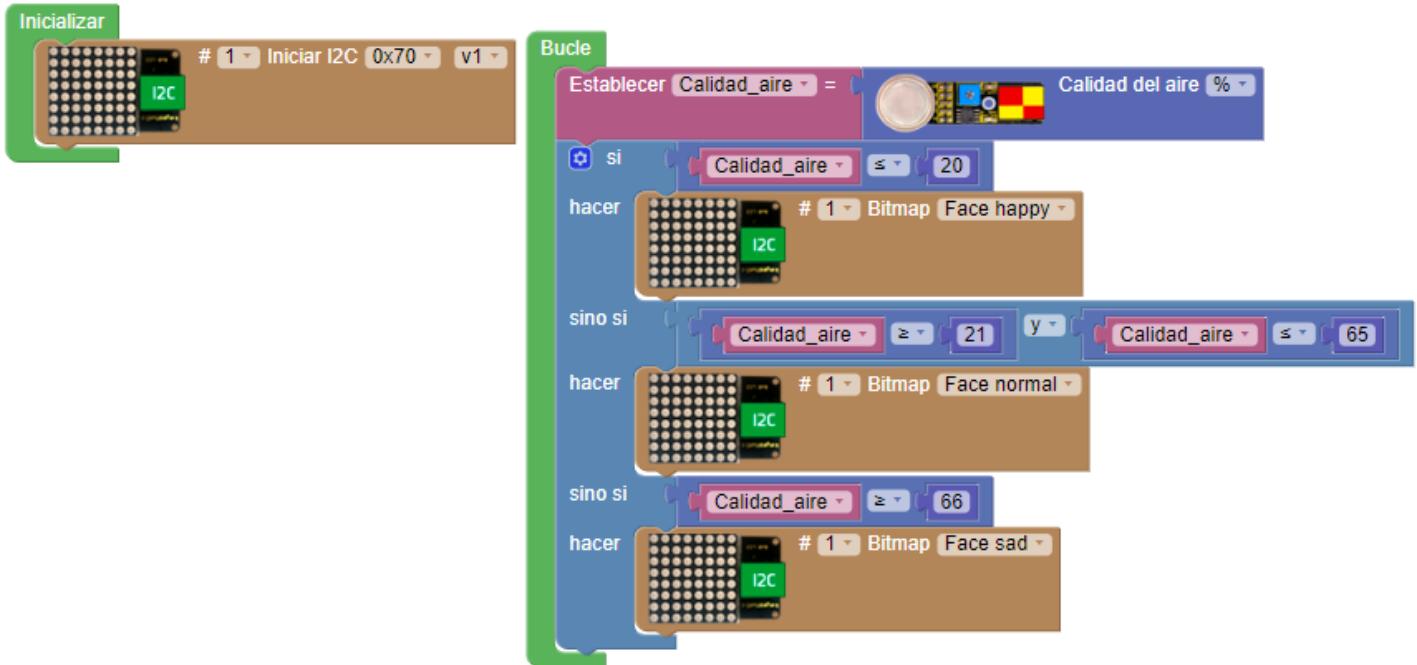
En el programa necesitamos tres condiciones diferentes. Para poder hacerlas iremos al menú “**Lógica**” e iremos usando las condicionales lógicas de igualdad, o mayor o menor que, o las condiciones Y.



Con ellos podrás crear un bloque como el siguiente:



El programa resultante es el siguiente;



Ajusta los valores comparándolos con una estación de calidad de aire.

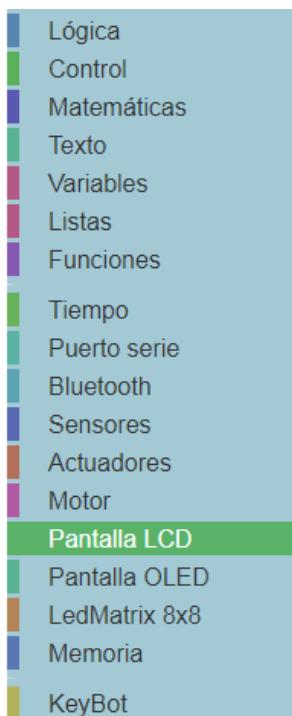


A19. – Pantalla LCD y Sensor DHT11

En esta actividad vamos a leer los valores de temperatura y humedad del sensor DHT11 en la pantalla LCD 1602.

Esta pantalla LCD al igual que la Matriz de led 8x8 se comunica por I2C.

En la pantalla podemos enviar datos como los valores de los distintos sensores, temperatura, luminosidad, distancias, ... o simplemente textos que escribamos.

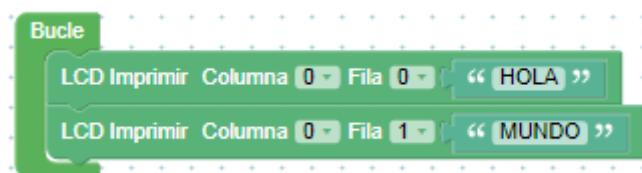


Inicialmente vamos a ver cómo funciona la pantalla LCD. En el bloque “**Pantalla LCD**” encontramos la función “**LCD Iniciar**”, sirve para indicar que vamos a usar la pantalla, que hay que poner al bloque “**Inicializar**”. Usaremos **2x16** si es una pantalla de 2 filas x 16 columnas y **4x20** si es de 4 filas y 20 columnas. El segundo parámetro establece el ADDR que lo dejaremos por defecto en **0x27***.



Seguidamente dentro del “**Bucle**” incluiremos la función “**LCD Imprimir**”. Se puede “imprimir” los datos en la primera fila=0 o en la segunda fila=1 y empezar a escribir en la columna deseada desde la primera columna=0 hasta la última columna=15 o 19 (según modelo).

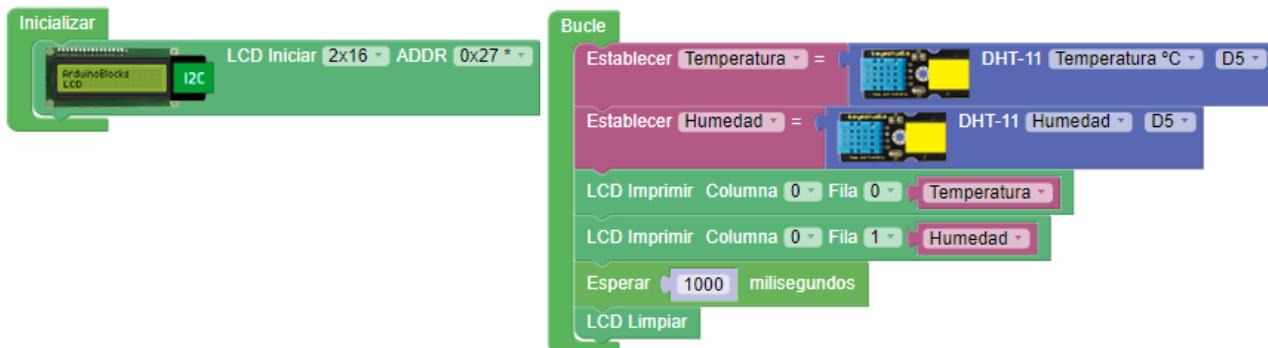
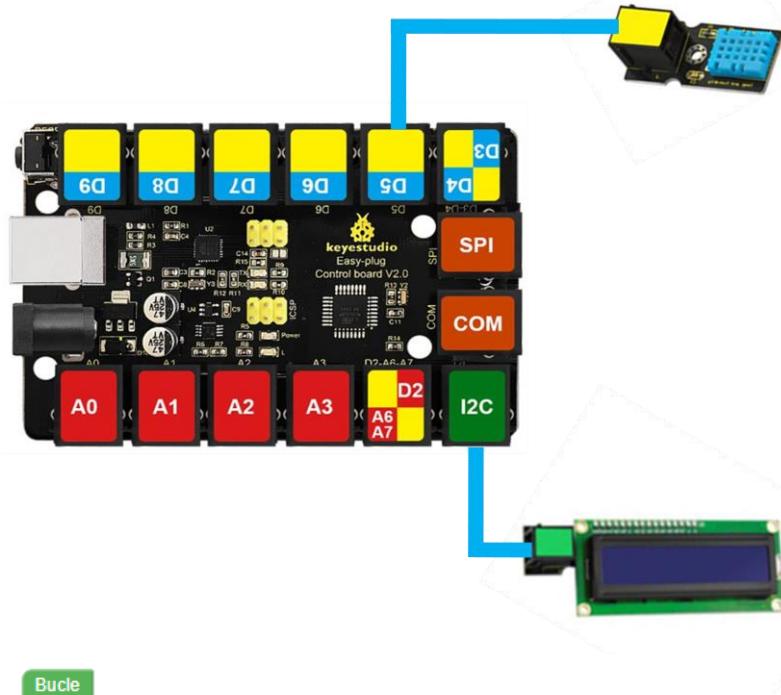
Vamos con nuestro particular “Hola Mundo”. Realiza este programa:



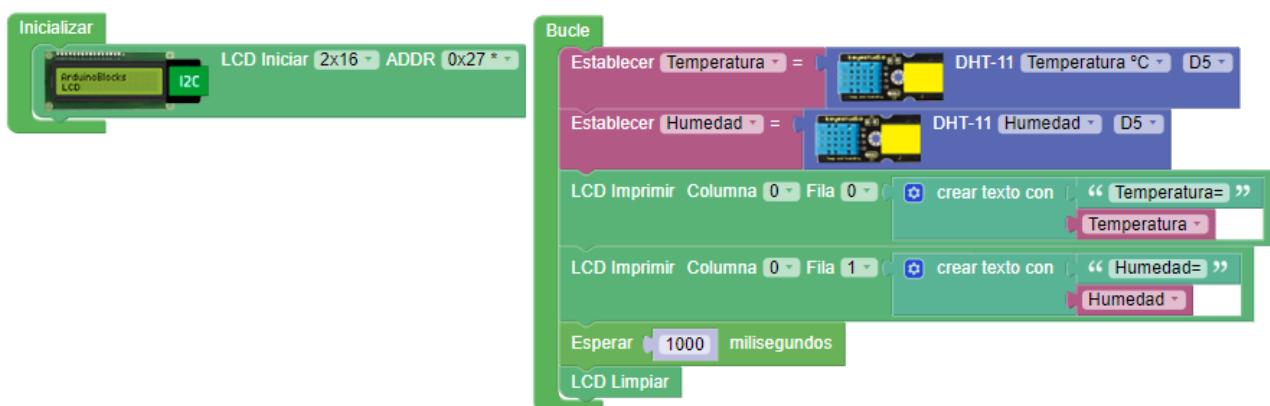
Ahora cambia los valores de “Columna” a 5 y 4 en el primer y segundo bloque de imprimir respectivamente. ¿Entiendes los conceptos de Columnas y Filas?

Conecta el sensor y la pantalla LCD como muestra la imagen:

Ahora vamos a ver los valores que nos lee el sensor DHT11 por la pantalla LCD. Creamos dos variables que las llamaremos “Temperatura” y “Humedad” a las cuales daremos los valores del sensor DHT11 y después “imprimiremos” esos valores. Dejaremos un tiempo de espera para que dé tiempo a ver el valor y por último “limpiaremos” la pantalla para que quede vacía y no quede nada debajo. Puedes hacer la prueba de no poner ese último bloque y comprobar lo que pasa.



Mejorando un poco el programa anterior nos podría quedar como sigue:



A20. – Módulo Bluetooth

En esta actividad aprenderemos a controlar el encendido y apagado de un LED con un móvil Android a través del módulo Bluetooth. El Bluetooth es un método inalámbrico de transmisión de datos. La tecnología Bluetooth es una tecnología estándar inalámbrica que permite el intercambio de datos de corto alcance entre dispositivos fijos, dispositivos móviles y redes de edificios de área personal (ondas de radio UHF en la banda ISM de 2.4 a 2.485 GHz).

Hay dos tipos de módulos Bluetooth de uso común en el mercado, los modelos HC-05 y HC-06. La diferencia entre ellos es que el HC-05 es maestro-esclavo, quiere decir que además de recibir conexiones desde un PC, Tablet o móvil Android, también es capaz de generar conexiones hacia otros dispositivos bluetooth. El HC-06 solo puede funcionar en modo esclavo, que solo puede aceptar el comando superior. El Bluetooth de nuestro KIT es HC-06.

Un módulo de comunicaciones Bluetooth estándar tiene 4 conexiones:

- Vcc: Alimentación 5V (+).
- Gnd: Alimentación 0V (-).
- Tx: Transmisión de datos.
- Rx: Recepción de datos.



El de nuestro Kit al tener las conexiones RJ11 todo esto está simplificado.

¡¡¡IMPORTANTE!!! Para cargar los programas siempre es necesario desconectar el módulo Bluetooth, si lo dejamos conectado dará problemas al cargar.

El Bluetooth 2.0 actualmente, sólo es compatible con los dispositivos Android. No es compatible con dispositivos Apple.

Para realizar el control del LED por Bluetooth hacen faltar dos cosas; la placa Easy Plug con el módulo Bluetooth y un móvil con una aplicación que permita el control. Para ello te puedes crear tu propia aplicación para Android utilizando APPInventor (<https://appinventor.mit.edu/>). Una plataforma online en la que puedes programar tus propias aplicaciones de modo muy similar a Scratch.

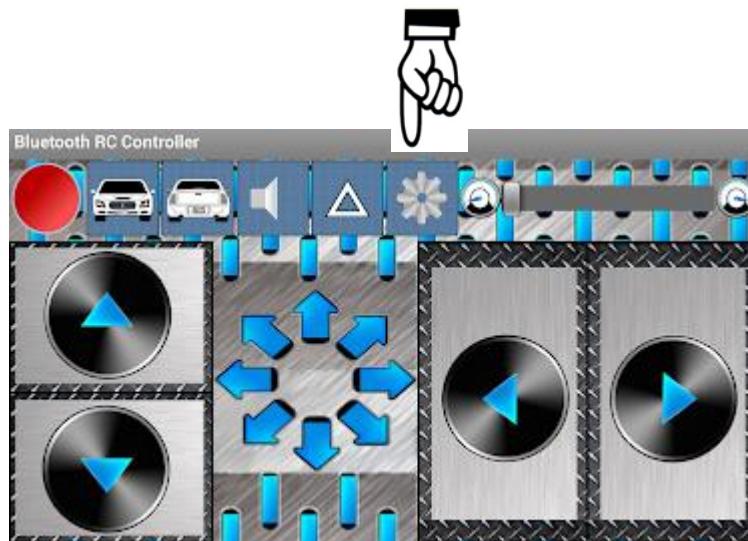
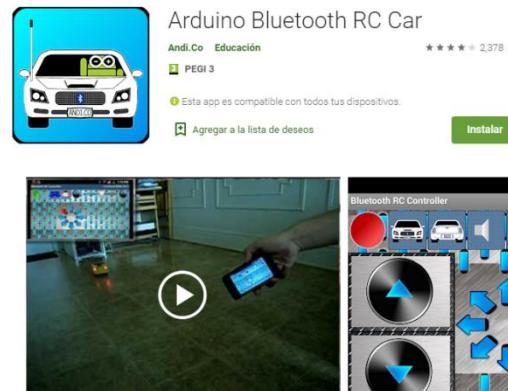


Pero si tienes muchas ganas de probarlo puedes empezar utilizando esta aplicación del PlayStore

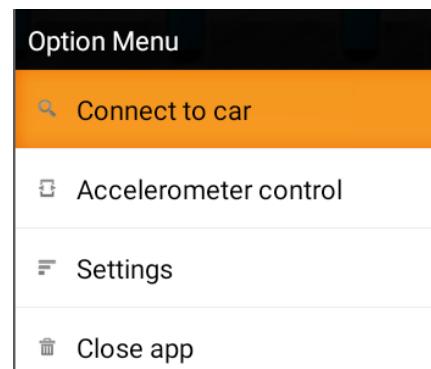
https://play.google.com/store/apps/details?id=braulio.calle.bluetoothRCcontroller&hl=es_419



Instala la aplicación en tu móvil Android.



Esta es la interfaz de la aplicación. Al pulsar sobre el botón de menú, aparecen las siguientes opciones:



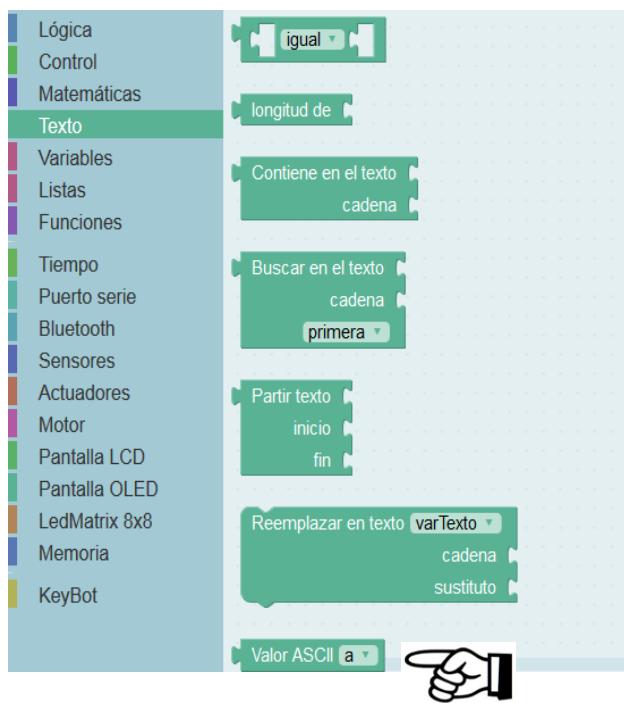
Bluetooth RC Controller	
Commands/characters sent to the car:	
Forward ->	F
Back ->	B
Left ->	L
Right ->	R
Forward Left ->	G
Forward Right ->	I
BackLeft ->	H
Back Right ->	J
Stop ->	S
Front Lights On ->	W (upper case)
Front Lights Off ->	w (lower case)
Back Lights On ->	U (upper case)
Back Lights Off ->	u (lower case)
Horn On ->	V (upper case)
Horn Off ->	v (lower case)
Extra On ->	X (upper case)
Extra Off ->	x (lower case)
Speed 0 ->	0 (zero)
Speed 10 ->	1
Speed 20 ->	2

Si clicamos sobre “**Settings**” aparece una ventana con diferentes opciones. Lo que nos interesa en este momento son los comandos/caracteres que enviamos a la placa de control desde la APP. Por ejemplo, el botón FORWARD envía una “F”, el BACK envía una “B”, el botón HORN ON una “V” y el HORN OFF una “v”.

Estamos trabajando con lecturas en bytes y esto nos obliga a leer los caracteres en código ASCII. Este código es una forma de codificar en números, los caracteres y símbolos del lenguaje. Pero tranquilos porque **ArduinoBlocks** dispone de un bloque para hacer la traducción inmediata, por lo que no tendremos que hacer ningún paso extra.

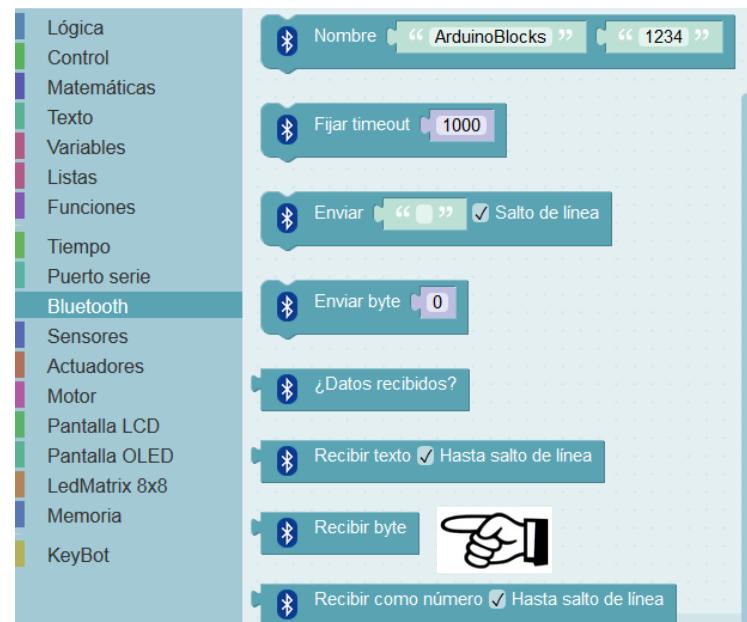
Si, por ejemplo, la App indica que cuando pulsas la flecha de movimiento hacia delante, envía por bluetooth una “F”, nosotros deberemos

seleccionar una “F” en la programación en ArduinoBlocks, mediante el siguiente bloque:

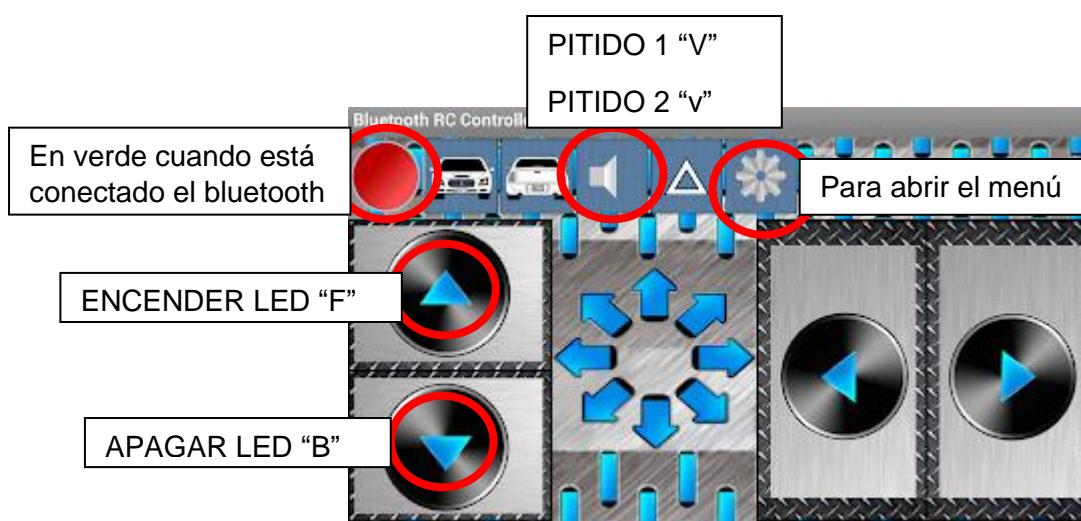


El bloque para traducir los datos leídos en bytes es “**Valor ASCII**”, se encuentra al final del menú “**Texto**”.

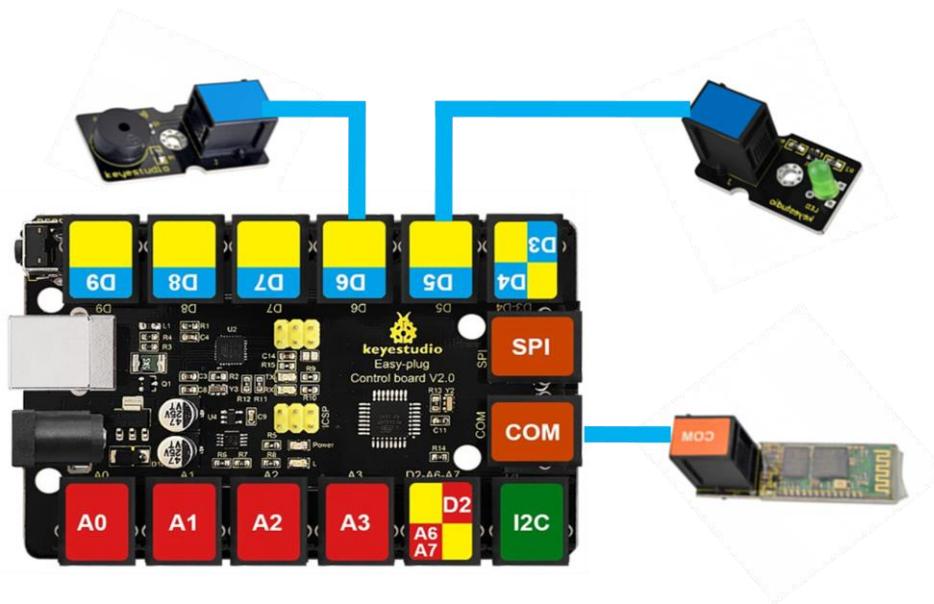
Para leer los datos procedentes del bluetooth vamos a necesitar el bloque “**Recibir byte**”, ubicado en el apartado “**Bluetooth**”.



Visto esta introducción vamos con el programa. Vamos a encender y apagar el LED del Pin D5 y emitir pitidos con el zumbador conectado al PIN D6 utilizando el móvil.

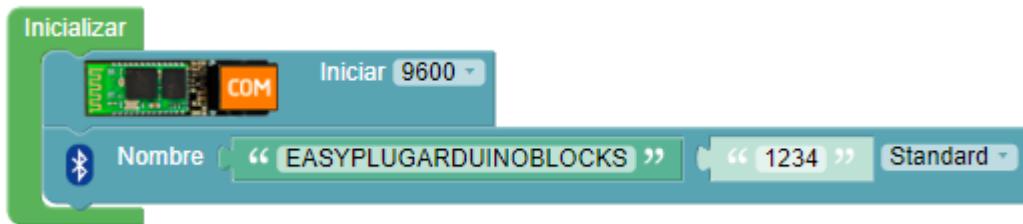


Vamos a realizar las conexiones en nuestra placa como refleja la imagen adjunta, fíjate como el módulo bluetooth sólo se puede conectar en el PIN COM.

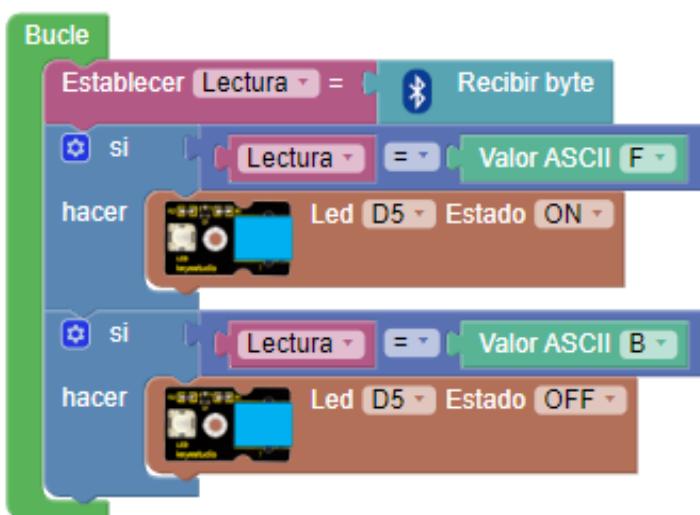


En [ArduinoBlocks](#) disponemos de un menú específico para el bluetooth con todos los bloques necesarios para programarlo.

Comenzaremos “**Inicializando**” el bluetooth. Y fijaremos una velocidad de comunicación de 9600 Bauds.



A continuación, realizaremos el siguiente programa en el cual al recibir la orden “F” enviada desde el móvil se encienda el LED y al recibir la orden “B” se apague.



Carga el programa en la placa, pero recuerda “**EL MÓDULO BLUETOOTH DEBERÁ ESTAR DESCONECTADO**”. Una vez cargado el programa en la placa es el momento de emparejar el bluetooth con el móvil.

Primero conecta el cable del Bluetooth.

Después abrir el Bluetooth en el móvil, buscar un dispositivo Bluetooth. Si encuentra un dispositivo Bluetooth llamado HC-06, emparejalo e ingresa la contraseña 1234, finalmente deberías de ver el dispositivo emparejado a tu móvil.

Nota: Cuando el Bluetooth está desconectado hay un pequeño led rojo parpadeando, cuando está emparejado el led rojo se queda fijo.

Al abrir la APP pulsa sobre el botón de Menú y clica sobre “**Connecto car**”. Buscar el dispositivo HC-06 o “**EASYPLUGARDUINOBLOCKS**” y conectar. En ese momento el botón rojo de la APP pasará a color verde y el led rojo del Bluetooth dejará de parpadear y se quedará fijo.

Ya puedes empezar a encender y apagar el LED con la aplicación. ¿Damos un paso más?

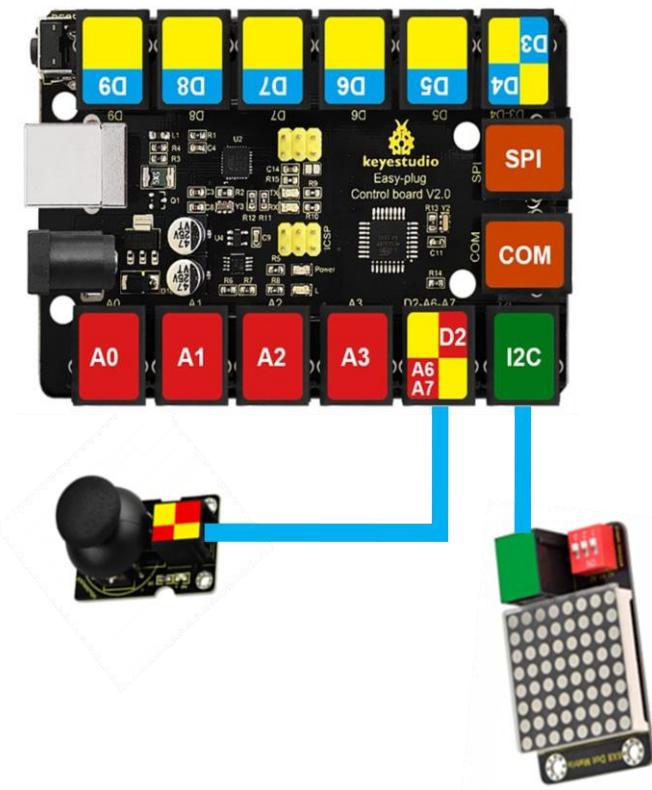
Ampliemos el programa haciendo que suene el zumbador, recuerda que vas a usar estos dos caracteres “V” y “v”. El programa quedará de la siguiente manera:



A21. – La matriz LED 8x8 y el módulo Joystick.

El **Módulo Joystick** está compuesto por dos potenciómetros, uno para el eje X y otro para el eje Y. Estos corresponderían a entradas analógicas. También dispone de un botón (al presionar hacia abajo) que sería una entrada digital.

En esta práctica vamos a controlar el movimiento de un ojo simulado en la matriz de LEDs con el módulo Joystick.



En este programa sólo está el movimiento del ojo en el eje X, hacia la derecha y hacia la izquierda. Complétalo haciendo los movimientos del ojo hacia arriba y hacia abajo con el eje Y.



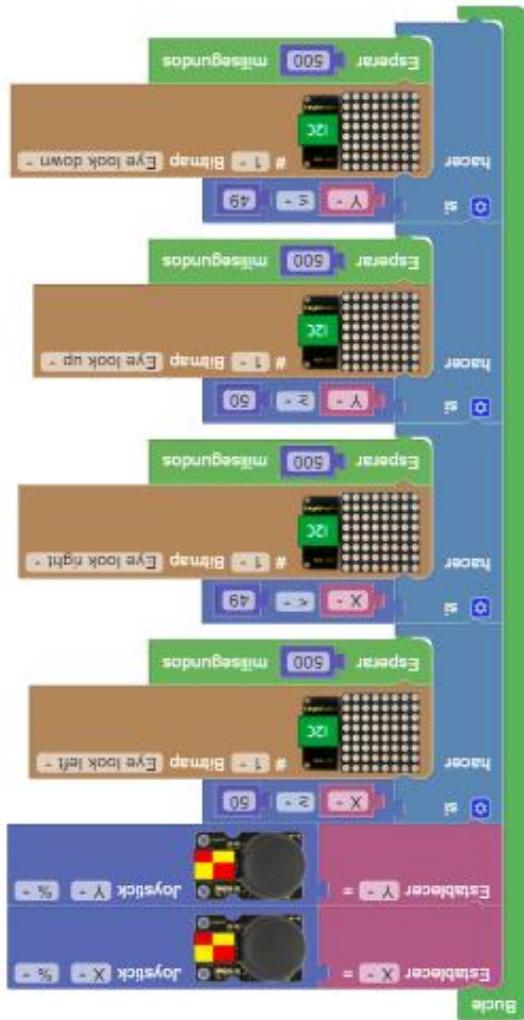
Conectamos los componentes de la manera que indica la imagen.

Fíjate como el joystick sólo se puede conectar al Pin múltiple A6/7 D2 ya que las dos entradas analógicas se corresponderán con el potenciómetro X e Y y el digital D2 con el botón.

En el programa hay que crear dos variables “X” e “Y” y establecer cada una de ellas con el valor del joystick en X y en Y.



Solución:



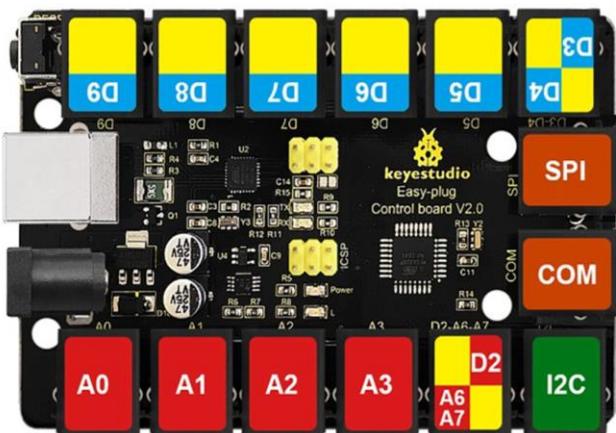
A22. – Pantalla OLED

OLED es la abreviatura de **diodo emisor de luz orgánico**. A nivel microscópico, una pantalla OLED es una matriz de LED orgánicos que se iluminan cuando emiten energía. La tecnología antigua de 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 debajo. Esto consume mucha energía porque en el momento en que la pantalla está encendida, se debe proporcionar suficiente luz para todos los píxeles. La nueva tecnología OLED sólo usa electricidad por píxel. Debido a que cada píxel crea su propia luz, sólo los píxeles que están encendidos usan electricidad. Esto hace que la tecnología OLED sea muy eficiente. Además, la forma en que se construyen estos tipos de OLED permite que sean muy delgados en comparación con la pantalla LCD.



La pantalla OLED de nuestro Kit tiene un tamaño de 128×64 píxeles y se comunica mediante Bus I2C.

Conectemos la pantalla OLED a nuestro Bus I2C de la placa de control.



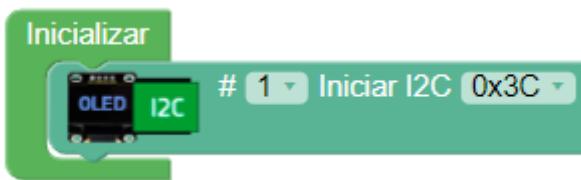
ArduinoBlocks dispone de un menú completo exclusivo para las pantallas OLED.

Vamos viendo sus utilidades.

The screenshot shows the ArduinoBlocks software interface. On the left, there is a sidebar with a tree view of block categories. The 'Pantalla OLED' category is highlighted. On the right, there is a workspace with several examples of how to use the OLED block. Each example consists of a green 'OLED I2C' block followed by specific parameters:

- # 1 Iniciar I2C 0x3C
- # 1 Rotación 0°
- # 1 Limpiar
- # 1 Texto X 0 Y 0 “ ” Led ON small
- # 1 Bitmap X 0 Y 0
- # 1 Pixel X 0 Y 0 Led ON
- # 1 Línea X1 0 Y1 0 X2 64 Y2 32 Led ON
- # 1 Rectángulo X1 0 Y1 0 W 64 H 32 Led ON Rellenar

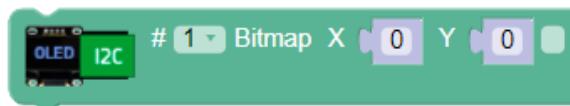
Comenzaremos inicializando la pantalla OLED:



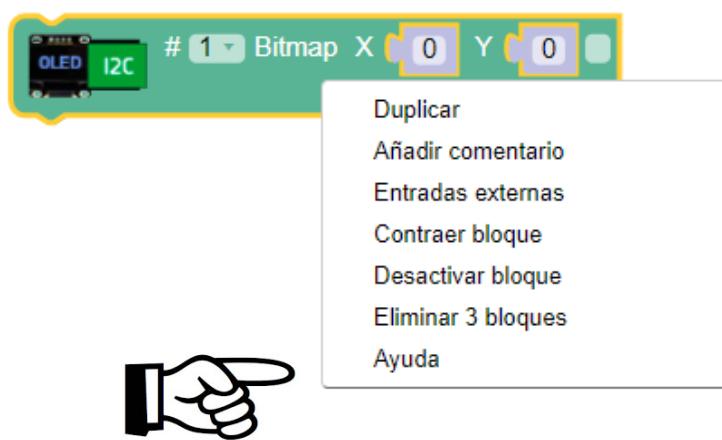
En esta actividad vamos a centrarnos en los relacionados con gráficos (bitmaps) y cómo hacer animaciones sencillas.

Para mostrar gráficos debemos generar un mapa de bits, que no es más que indicar los píxeles que están encendidos y los que están apagados. Este proceso puede ser complejo si lo queremos hacer a mano, pero en [ArduinoBlocks](#) tenemos una herramienta que nos permite generar fácilmente el código de los bitmaps de una manera, como no, visual...

El bloque para mostrar un bitmap es el siguiente:

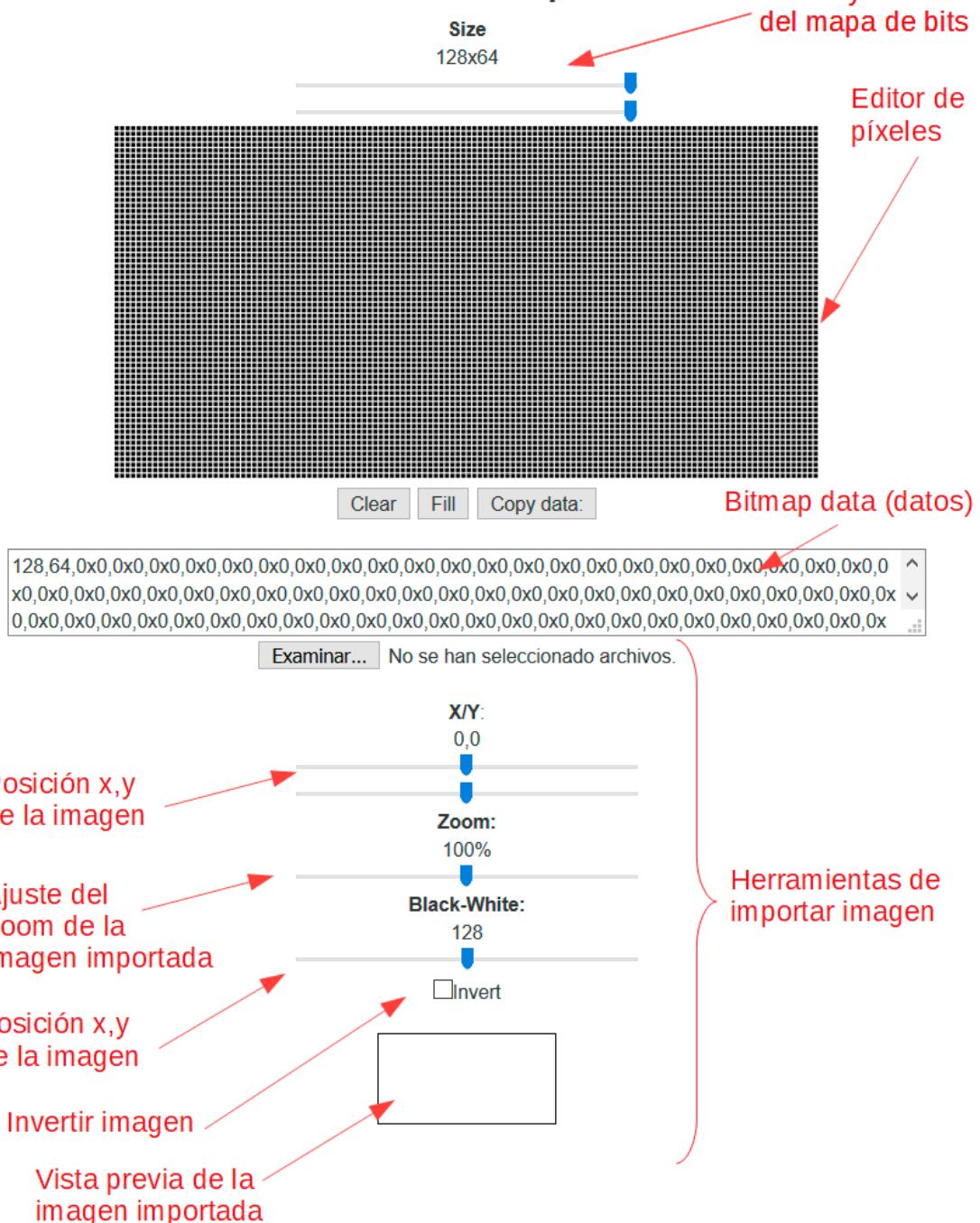


Con el botón derecho sobre este bloque, y pinchando en la opción “**Ayuda**” se abrirá una nueva pestaña con el editor de mapas de bits para pantalla OLED:



Este es el editor OLED:

OLED - Bitmap Data



Podemos crear un mapa de bits manualmente pinchando encima del editor de píxeles (píxel a píxel), pero el proceso puede ser bastante complicado y entretenido... Si tenemos arte dibujando seguramente utilicemos otros editores de gráficos más avanzados (Paint, Gimp, Photoshop, ...) o simplemente podemos descargar una imagen e importarla para convertirla en un mapa de bits binario (blanco y negro) adaptado a nuestra pantalla OLED. Por ejemplo, vamos a descargar el logotipo de Arduino y adaptarlo para mostrarlo en la pantalla OLED.



<https://images.app.goo.gl/nDUcqC4GjzhorfvF7>

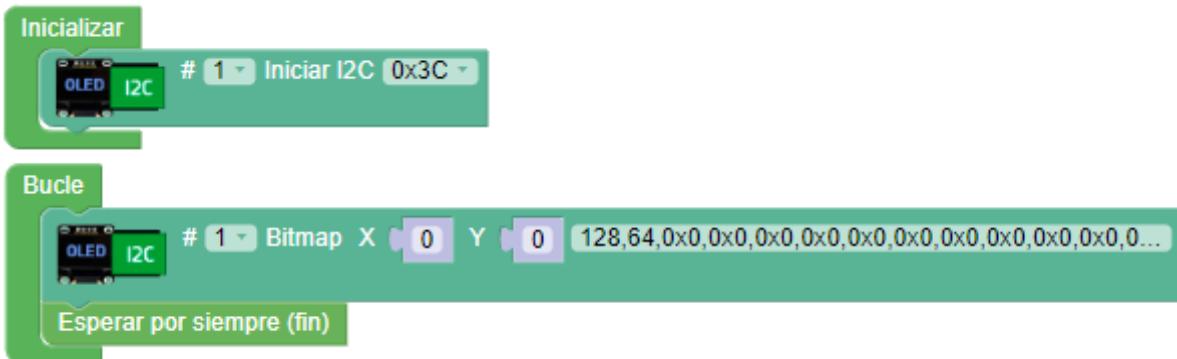
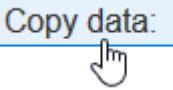
Una vez importado en el editor podemos ajustar la posición x/y, el ancho y alto del bitmap, el zoom y sobre todo debemos ajustar el umbral de conversión a B/N para obtener la mejor calidad posible en la conversión:

OLED - Bitmap Data

Examinar... logo.jpg



Ahora, simplemente debemos hacer un “**Copy data:**” y pegarlo en el bloque de bitmap:

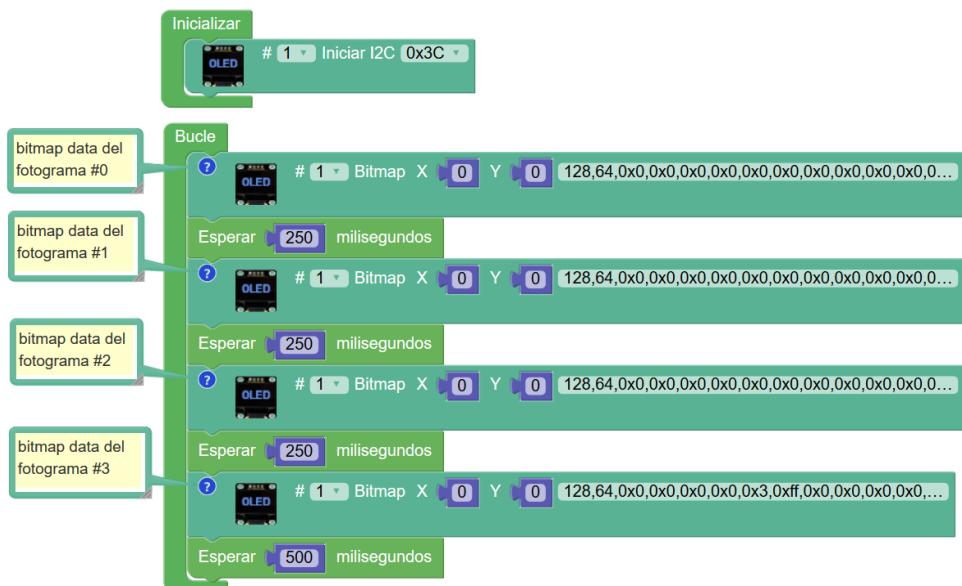


Y el resultado es:



Prueba con diferentes imágenes o incluso prueba a realizar un gif animado, verás que divertido.

Este programa es una posible solución para realizar un gif animado con 4 fotogramas.



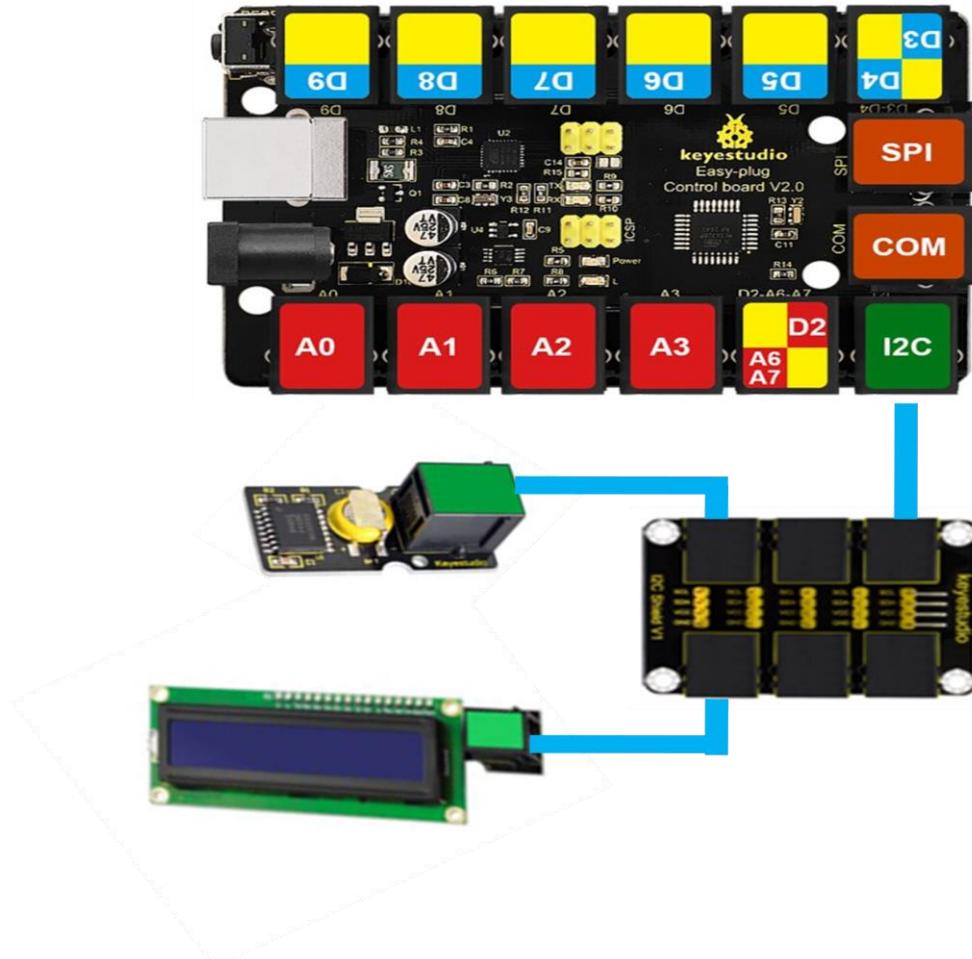
A23. – Reloj DS3231 + Pantalla LCD

En multitud de ocasiones es muy útil disponer de un reloj que nos cuente la hora y la fecha en la que se recogen los datos, como por ejemplo en una estación meteorológica en la cual nos indique el día y la hora en la que se registra cada dato.

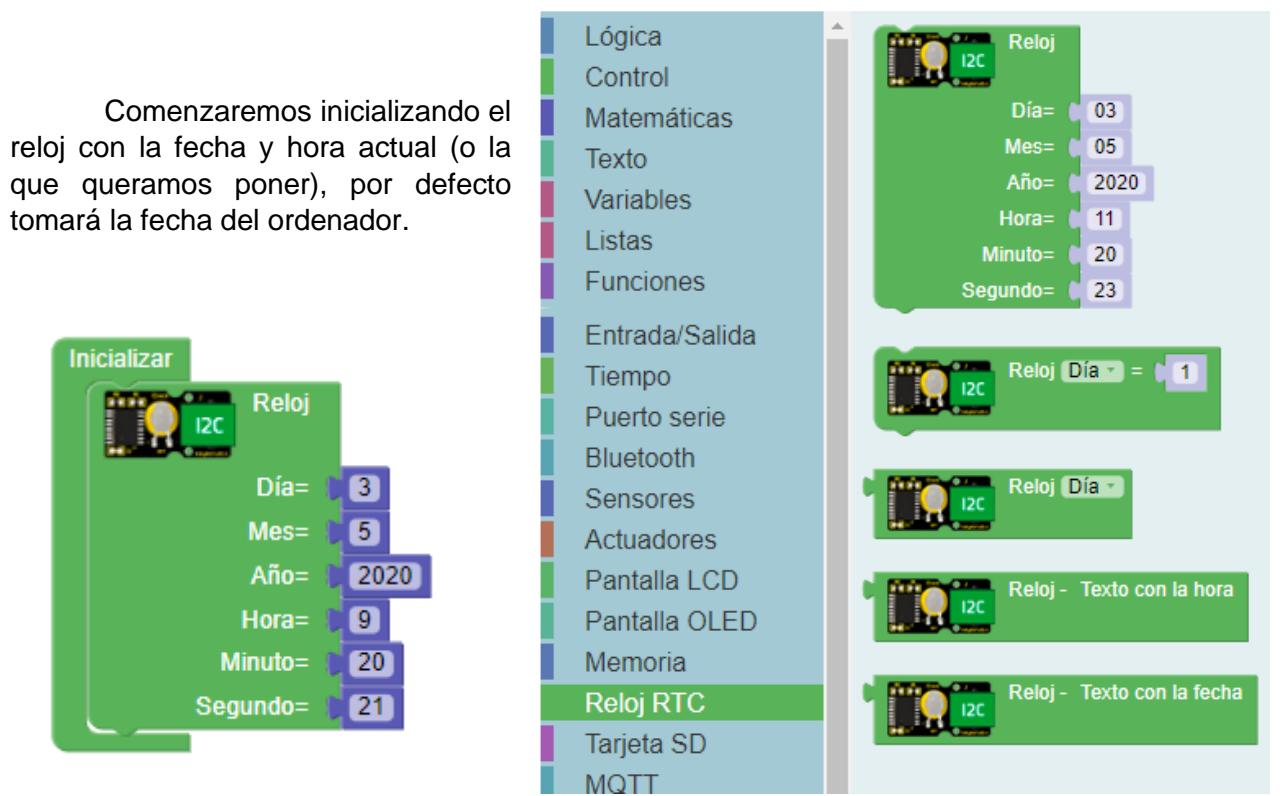
En la siguiente actividad vamos a hacer una lectura en la pantalla LCD de la fecha y la hora.

Este módulo reloj DS3231 va por I2C al igual que la pantalla LCD, por ese motivo se hace necesario el uso del Hub.

Conectamos el módulo reloj y la pantalla LCD como muestra la imagen:



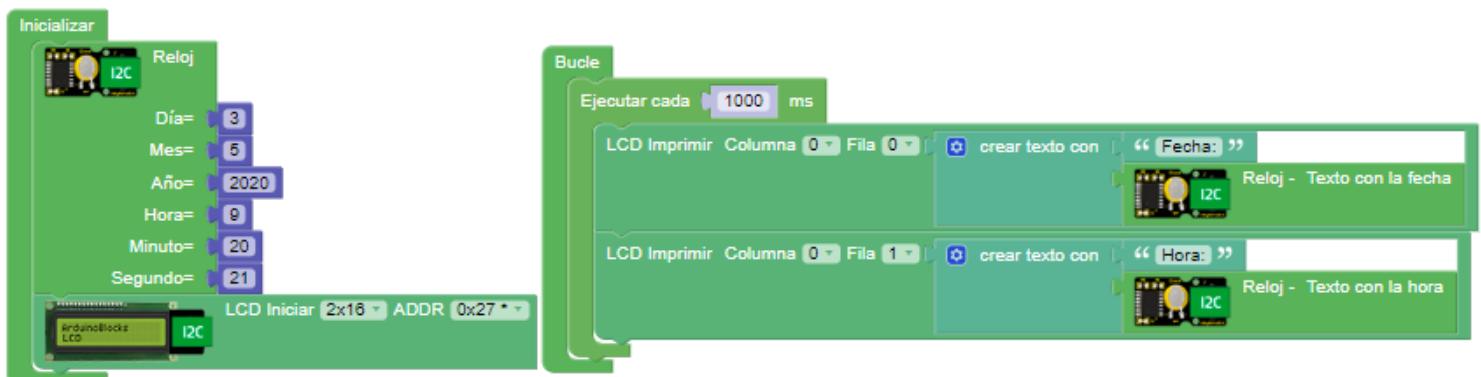
Y como no, ArduinoBlocks tiene un menú específico con bloques para programar el Reloj RTC.



Como también vamos a usar la pantalla LCD debemos inicializarla también con este bloque:



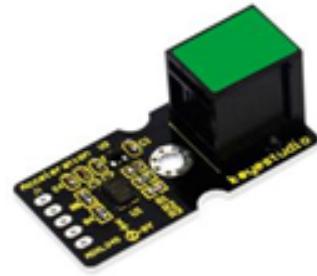
Finalmente, el programa quedaría tal y como muestra la imagen:



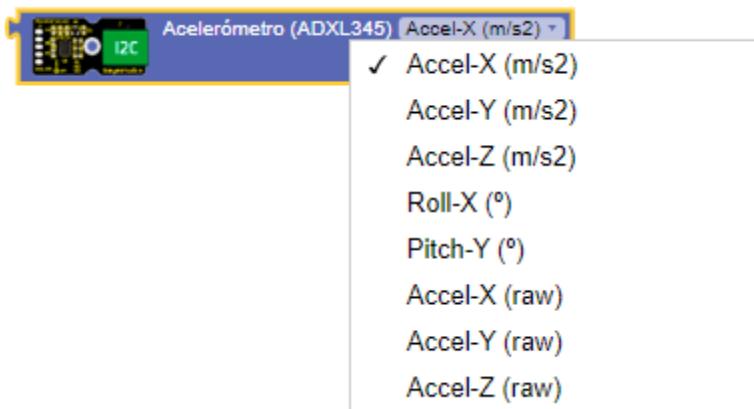
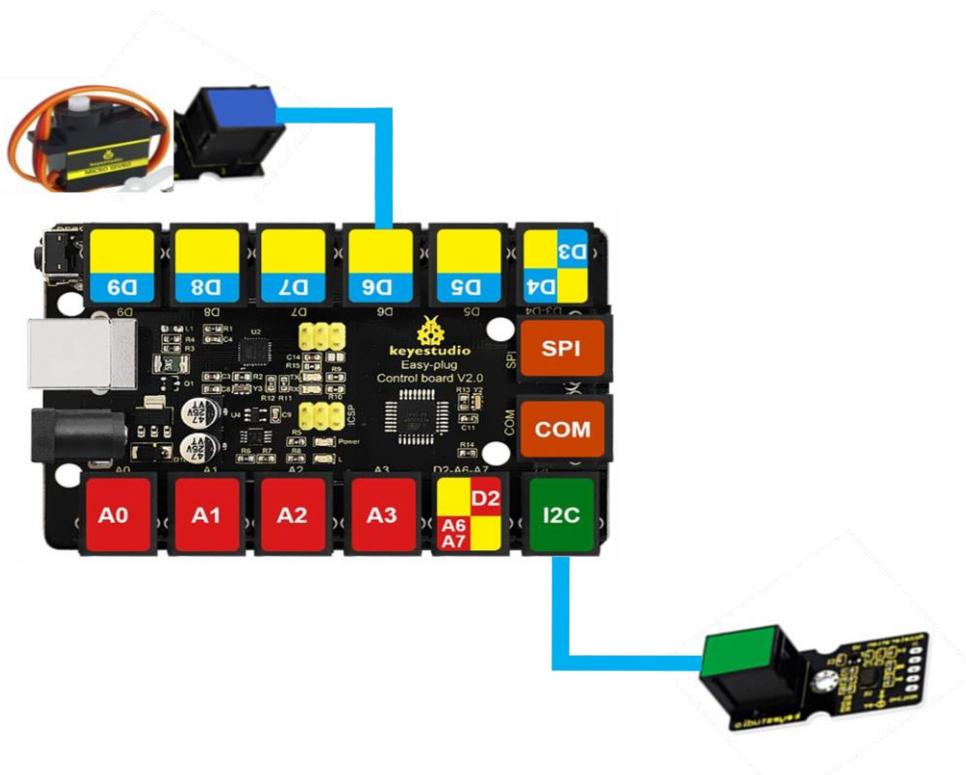
A24. – Control del servo con Acelerómetro

En esta actividad vamos a realizar un sencillo ejemplo de control de un servomotor con un acelerómetro.

El módulo ADXL345 es un acelerómetro MEMS de 3 ejes con bajo consumo de energía y diseño compacto. Detecta la aceleración en los ejes X, Y y Z. Es de alta resolución de 13 bits, medición de hasta ± 16 g (fuerza de gravedad). Los datos de salida digital están formateados como un complemento de dos dígitos de 16 bits y se puede acceder a través de una interfaz digital SPI o I2C.



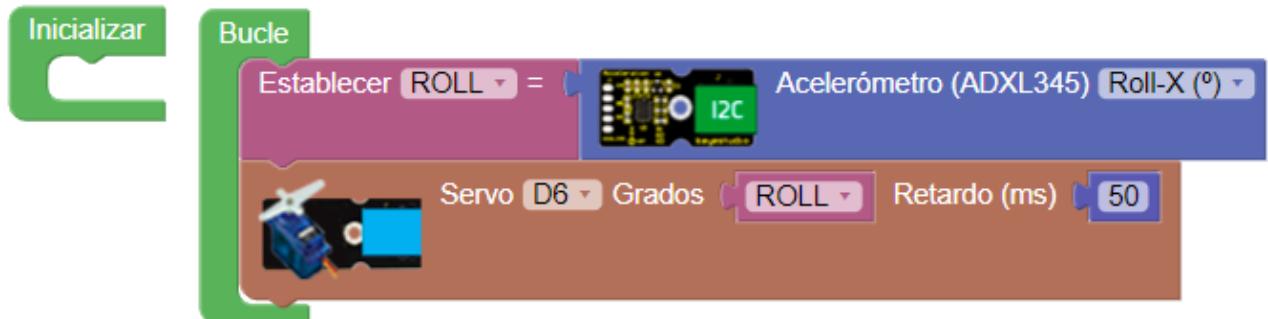
Como nos indica su color verde, este sensor deberá ir conectado al Pin I2C de la placa Easy Plug. El servomotor lo conectaremos al Pin D6.



ArduinoBlocks simplifica al máximo la programación del sensor acelerómetro de tal manera que únicamente tiene un solo bloque para programarlo. Para nuestro ejemplo nos interesa seleccionar Roll-X(°).

Generamos una variable y la llamaremos ROLL, para después sustituirla en el valor de los ángulos en el bloque del servo.

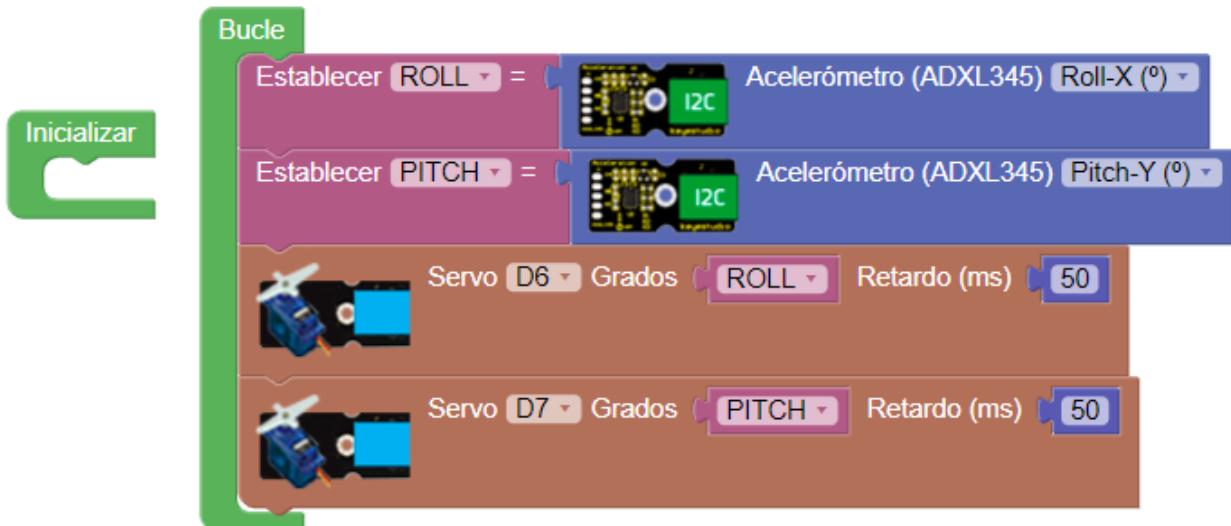
El programa final quedaría de la siguiente manera. Cárgalo en tu placa y comprueba su funcionamiento.



Imagina un proyecto en el que tengas que controlar dos servos uno para realizar un movimiento sobre el eje X (en horizontal) y el otro para realizar un movimiento sobre el eje Y (en vertical). Utilizando un soporte similar al de la imagen adjunta.



Podrías realizar la programación añadiendo dos bloques más:



A25. – LED RGB

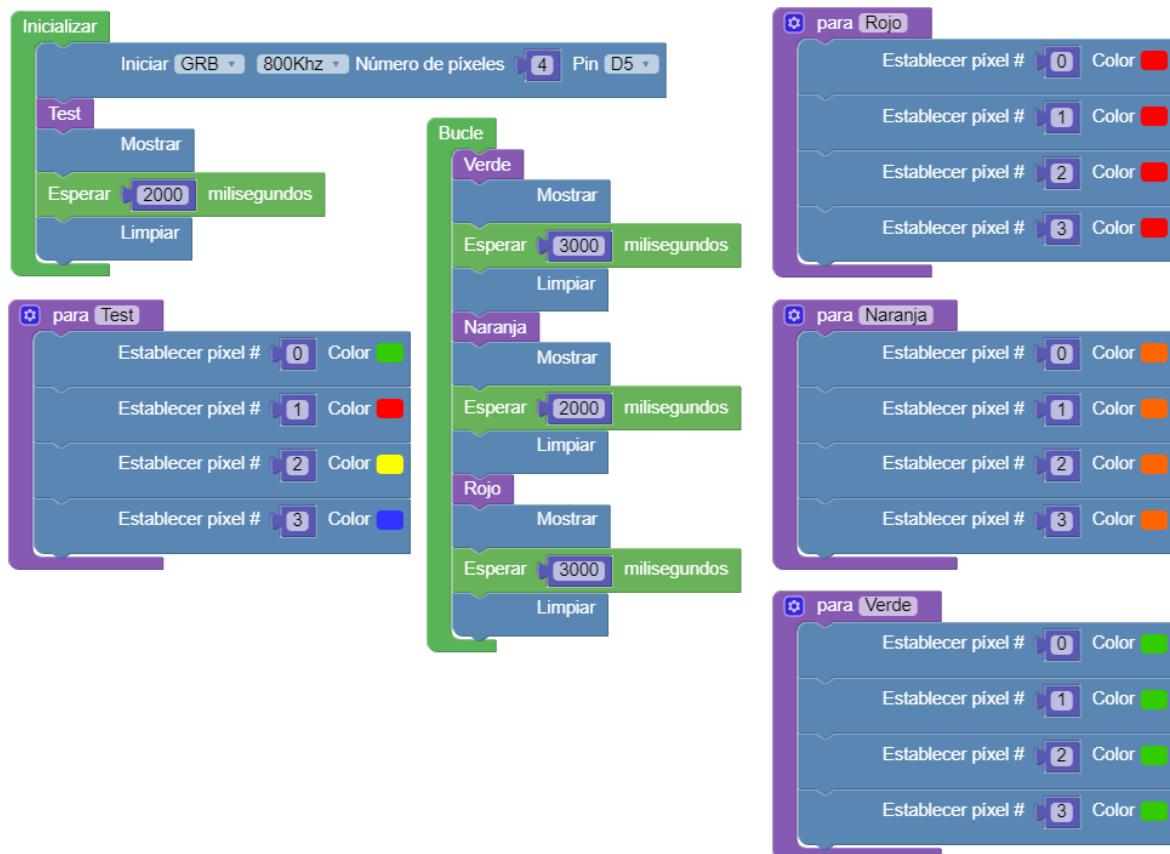
Este módulo LED RGB, en realidad es un módulo con 4 leds Neopixel. Los leds Neopixel son muy interesantes porque se puede encender el led que quieras con el color que te apetezca, cada uno de los leds tiene una “dirección única” y es RGB, todo gestionado por un solo cable de datos y un controlador integrado en el mismo elemento. Un formato muy habitual son las tiras leds, también hay anillos, círculos, matrices, pero en este caso disponemos de una formación de leds de 2x2, pero el funcionamiento es el mismo.



Vamos a preparar un ejercicio para simular un semáforo, adjudicando a cada uno de los leds el color que queramos. Y para ver que se pueden encender led por led, en el bloque “Inicializar” vamos a hacer un “Test” asignando un color para cada led.

Aprovechamos también, para utilizar el tema de la “Funciones”.

El programa quedaría de la siguiente forma utilizando en PIN D5:



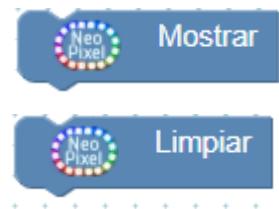
Por un lado, tenemos el bloque para “Inicializar” el módulo (tira led, matriz...) indicando el PIN donde lo conectamos al Arduino y el número de LEDs disponibles en el elemento.



Por otro lado, tenemos distintas funciones para asignar el led (empezando por el 0) y el color que deseamos poner en cada uno de los leds. En el ejercicio de ejemplo se ha utilizado la función “**Establecer pixel #**”, donde indicamos que led encender y que color asignar.



Pero con asignar el led y el color no es suficiente, hay que utilizar el bloque “Mostrar” para dar la orden y que se ejecute en el elemento, y también hemos usado el bloque “Limpiar” para dejar todos los leds apagados entre transiciones.



Se han vuelto a utilizar otra vez las “**Funciones**” para practicar con ellas y para hacer que el bloque “**Bucle**” quede un poco más simplificado y fácil de entender.

En este caso tres funciones para los tres colores de un semáforo:

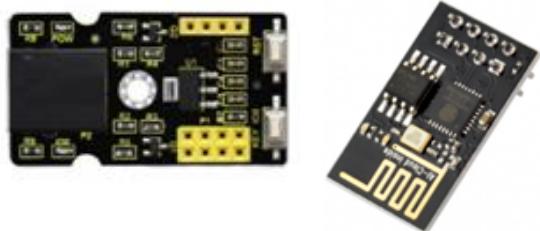


La función “Test” no es necesaria, pero se ha realizado como ejemplo de que cada uno de los leds se puede programar como deseamos, en este caso se ejecuta una sola vez al arrancar el programa, cada uno de los leds de un color distinto y así aprovechamos para ver que todos los leds funcionan y podemos ver todas sus posibilidades.



A26. – Comunicaciones por wifi, enviando información.

El Kit Easy Plug presenta un módulo ESP-01 basado en el microcontrolador ESP8266 que permite realizar comunicaciones vía wifi. Para facilitar la conexión de este módulo a la placa se dispone de un zócalo para facilitar su montaje con los conectores RJ11.



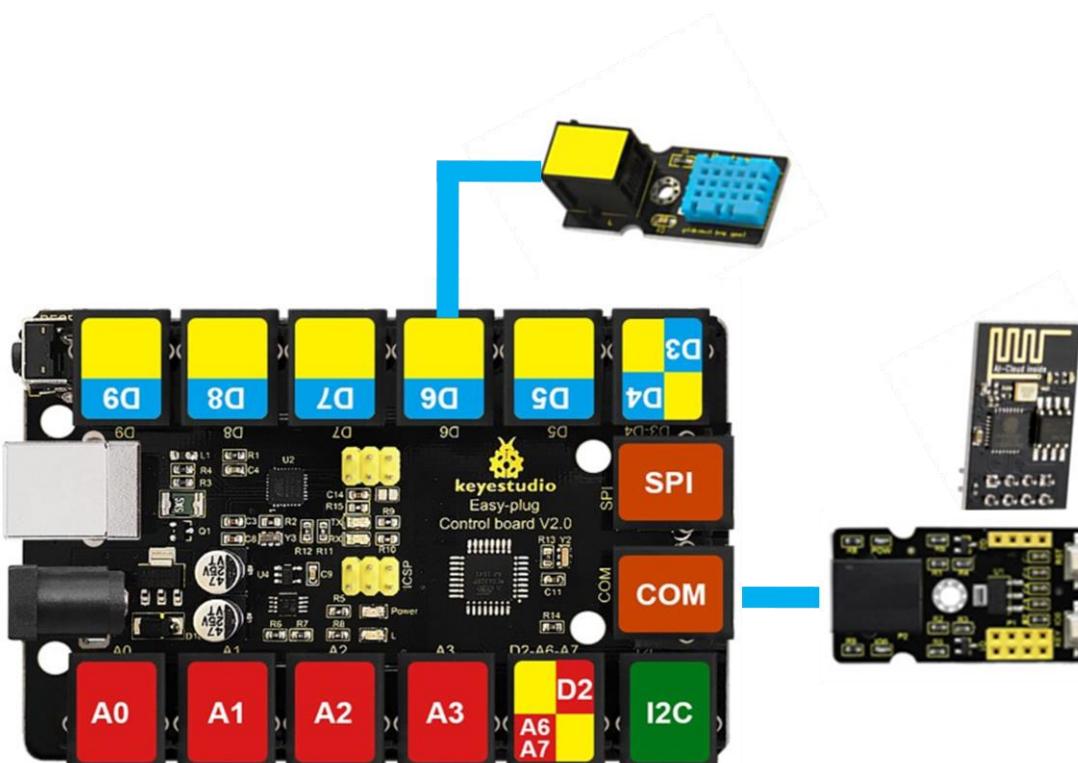
En esta actividad vamos a enviar por wifi los valores de temperatura y humedad registrados por el sensor DHT-11 y poder verlos remotamente en nuestro propio móvil.

Utilizando los bloques MQTT vamos a realizar un programa de prueba para enviar la temperatura y humedad medida con un sensor DHT-11 a través del servidor gratuito iot.eclipse.org y visualizaremos los datos en un móvil o tablet con un cliente MQTT

**Al utilizar un servidor gratuito sin seguridad (no tenemos ni usuario ni clave) cualquier usuario que se suscriba a los mismos “topics” podrá visualizar nuestros datos, pero para pruebas es una buena primera opción.*

Vamos a realizar las conexiones en nuestra placa tal y como muestra la imagen.

RECUERDA: el ESP-01 va conectado en el Pin COM por lo que deberá de desconectarse siempre que se cargue un programa nuevo, al igual que ocurría con el módulo bluetooth.

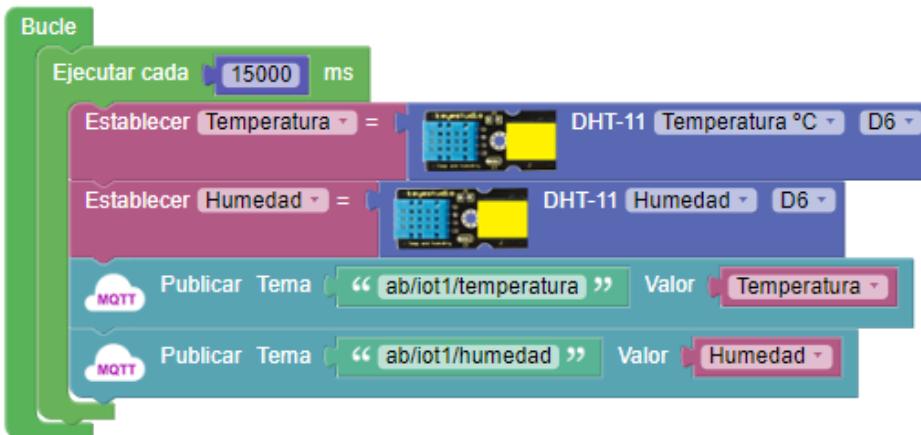


Ahora comencemos con la programación; para ello definamos dos “Topics”:

Para la temperatura: **ab/iot1/temperatura**

Para la humedad: **ab/iot1/humedad**

Inicializamos el programa introduciendo nuestros datos de nuestra red wifi a la que estamos conectados y la clave.



(Se debe especificar el SSID y clave de la red WiFi correctamente)

Importante: La comunicación WiFi necesita procesar datos continuamente, por lo que debemos evitar bloques tipo “esperar” o condiciones de bloqueo, por lo que utilizaremos el bloque **“Ejecutar cada”**.

Una vez subido, nuestro Arduino debería conectarse a la red WiFi y publicar los datos al servidor MQTT cada 15 seg.

Ahora nos falta como realizar la configuración de un cliente MQTT en Android.

Utilizaremos la aplicación MQTT Dash, donde configuraremos una conexión con el servidor iot.eclipse.org y añadiremos una suscripción a los mismos topics para visualizar los datos publicados.

https://play.google.com/store/apps/details?id=net.routix.mqtdash&hl=es_419

- 1º.- Definir la conexión al servidor MQTT
 - 2º.- Definir los componentes y asociarlos a un “topic” del que recibirán la información.
- En este caso creamos componentes del tipo “text”.
- 3º.- Una vez definido todos los componentes (temperatura y humedad) y sus topics, ya podemos visualizar los datos en el panel principal.

1

2

3

En este ejercicio hemos practicado el envío y recopilación de datos hacia un Servidor Online o Servidor Cloud, donde podemos almacenar los datos para poder trabajar con ellos, analizarlos al estilo “Big Data”, “Data Analytics”, ... y todo tipo de tecnologías relacionadas con el análisis y procesamiento de datos.

Pero ahora haremos otro ejercicio donde en vez de recopilar datos, daremos orden de encender y apagar un led, un ejercicio más enfocado hacia un uso tipo “Domótica” para, por ejemplo, crear una Smart Home o Automatización Industrial entre muchas opciones posibles.

Así de esta forma, vemos que el IoT, Internet of Things, es cada vez más importante tanto para el ámbito de consumo como empresarial, para relacionarnos con nuestro entorno físico, englobando una gran cantidad de tecnologías y pudiéndolo aplicar en muchos sectores con unas posibilidades muy amplias.

A27. – Comunicaciones por wifi, activando un elemento.

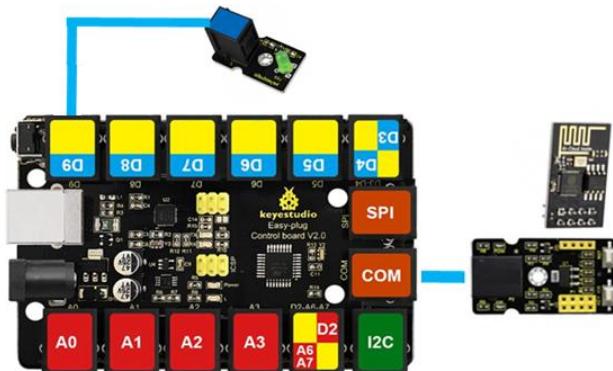
Como comentábamos, vamos a hacer un ejercicio utilizando otra vez los bloques MQTT y el módulo ESP8266 para activar y desactivar un led a través de internet.

RECUERDA: el *ESP8266 va conectado en el Pin COM por lo que deberá de desconectarse siempre que se cargue un programa nuevo, al igual que ocurría con el módulo bluetooth.*

El programa sería el siguiente, hay que rellenar el nombre de la red en “WiFi red” e introducir la contraseña de esta en “clave”.

The screenshot shows two main sections of the ArduinoBlocks IDE:

- Initializar:** This section contains the MQTT WiFi configuration block. It specifies the WiFi module as "keyestudio", the Rx pin as "COM-TX", the Tx pin as "COM-RX", baud rate as "115200", WiFi network name as "NOMBRE_WIFI", password as "CLAVE_WIFI", broker as "mqtt.eclipse.org", port as "1883", client ID as "ID", user as "", and password as "".
- Bucle:** This section contains a "Bucle" (Loop) block. Inside the loop, there are two "si" (If) blocks. The first "si" block checks if "Estado_Led" is equal to 1, and if true, it executes the action "hacer" (do) which is "Led D9 Estado ON". The second "si" block checks if "Estado_Led" is equal to 0, and if true, it executes the action "hacer" (do) which is "Led D9 Estado OFF".

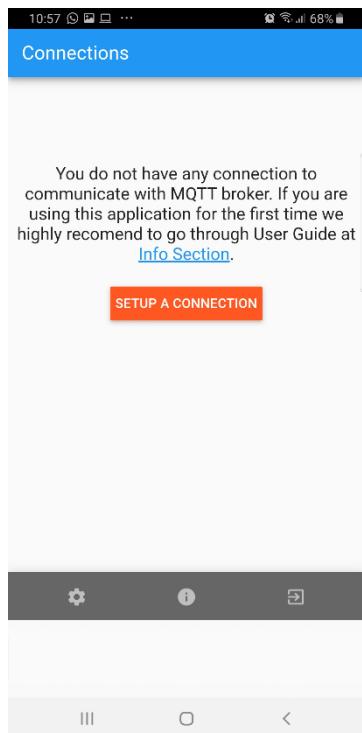


Esta vez en el programa vamos a “Suscribir” un Tema que se llame “INNOVALED”, este tema va a ser el estado en que deba encontrarse el led conectado a D9, por lo tanto, lo asociaremos a una variable que se llame Estado_Led.

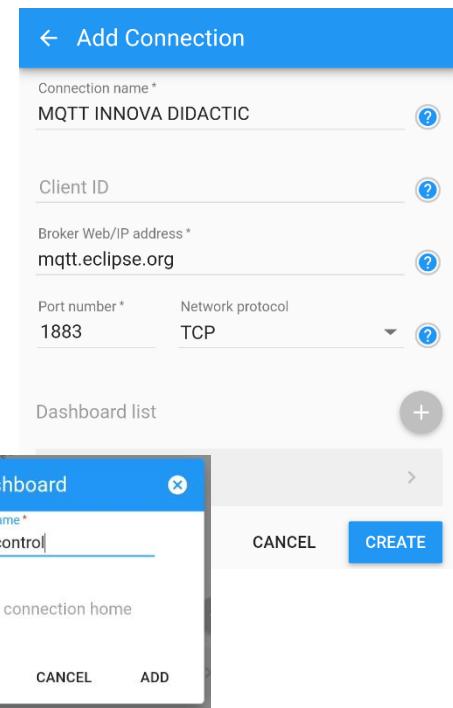
Ahora en el “Bucle” vamos a programar las condiciones correspondientes al estado 1 y estado 0, como encendido y apagado respectivamente.

Este estado va a cambiar a través de una aplicación del móvil donde configuraremos un “Interruptor” asociado el mismo Tema “INNOVALED”, vamos a ver en las siguientes fotos, donde pondremos los nombres y los estados.

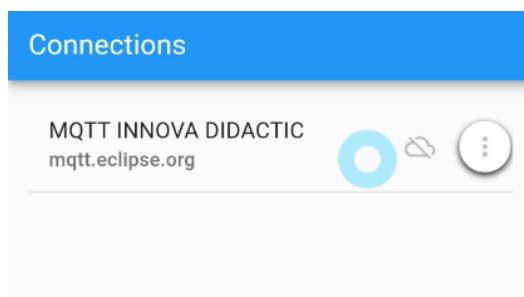
Vamos a utilizar el programa IoTMQTTPPanel que se puede encontrar en Google Play Store entre muchas disponibles.



Al iniciar el programa por primera vez nos dice que no tenemos ninguna conexión. Le damos a "Setup a Connection" y rellenamos los campos necesarios, un nombre descriptivo para la conexión, los datos del servidor mqtt.eclipse.org, puerto 1883 y protocolo TCP, además pulsamos sobre el símbolo "+" para crear un Panel o Dashboard, poniendo también un nombre descriptivo y para finalizar le damos a "Create".

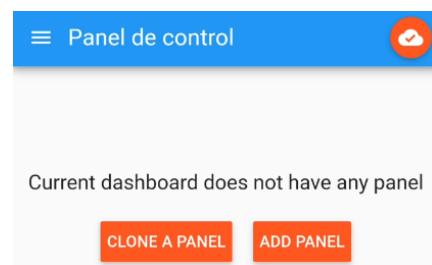


Un Dashboard es un tablero o panel donde podremos poner distintos elementos de control, como por ejemplo en nuestro caso, un interruptor para controlar un LED.



Cuando ya tenemos una conexión creada, pulsamos encima de esta y nos dice que nuestro panel está vacío.

Clicamos sobre "Add Panel" y nos sale una lista muy grande de elementos que podemos añadir, para nuestro ejercicio vamos a seleccionar un "Switch".



[← Add a Switch panel](#)

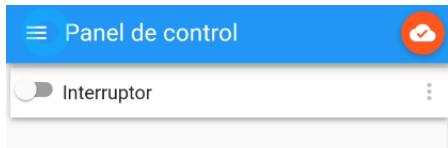
Panel name *	Interruptor
Topic *	INNOVALED
Subscribe Topic	?
Payload on *	1
Payload off *	0
<input type="checkbox"/> Use icon switch	

Para este elemento, lo más importante, son los campos “Topic” y “Payload”.

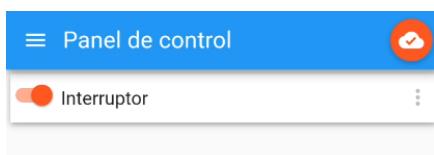
Topic es el nombre del tema MQTT que hemos puesto en el programa de [ArduinoBlocks](#), debe tener el mismo nombre “INNOVALED”.

Playload es el valor que tendrá según la posición del interruptor, para “On” pondremos el valor 1 y para “Off” pondremos 0, bajamos hacia abajo del todo para pulsar “Create”.

Recuerda que estos valores son los que sirven para controlar el estado del LED con la función “**Si... Hacer...**”.



Una vez cargado el programa a la placa, y configurado la aplicación de móvil, si todo se ha conectado correctamente a internet, prueba a cambiar el estado del “Interrupor”, verás que como el LED se enciende y se apaga.



¡¡¡No es magia, es Internet of Things!!!

¿Te atreves a controlar las luces de tu casa?

IoT sirve para esto y mucho mas.

¡¡¡Hasta aquí el tutorial para el KIT KEYESTUDIO EASY PLUG SECUNDARIA!!!

Lo que hemos visto es solo una pincelada del mundo Arduino. Hay infinidad de proyectos para crear con la gran cantidad de sensores y actuadores que existen. Consulta los Kits para primaria, secundaria y formación profesional disponibles en ¡¡¡INNOVA DIDACTIC!!!



