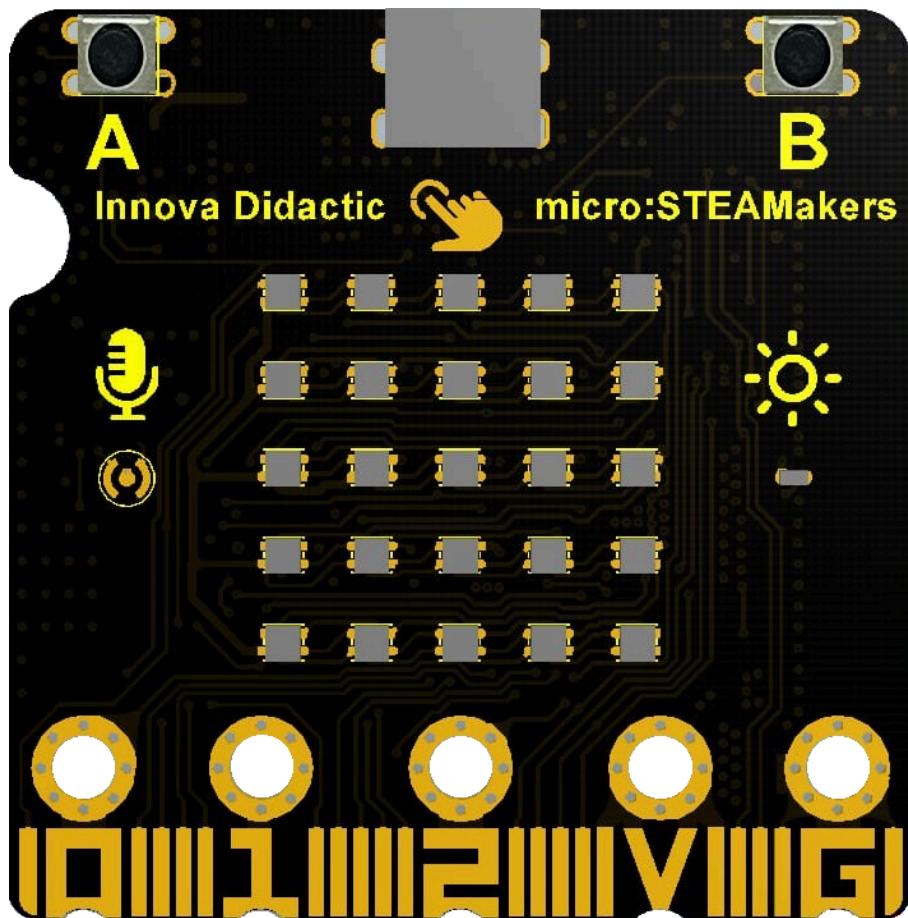


ESP32 micro:STEAMakers



ÍNDICE

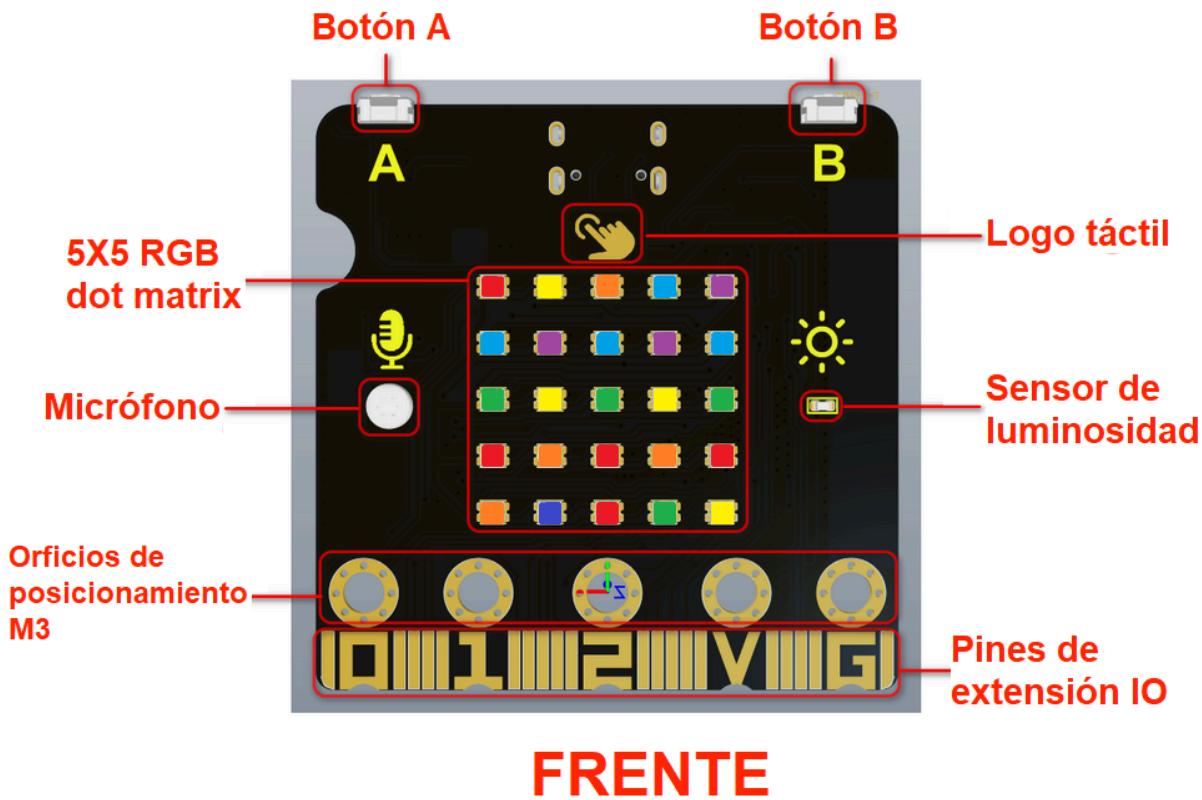
1 Descripción.....	2
2 Especificaciones técnicas.....	5
3 Introducción de arduinoblocks.....	6
4 Actividades con ESP32 micro:STEAMakers.....	6
Actividad 1.1 NeoMatrix 1.....	7
Actividad 1.2. NeoMatrix 2.....	8
Actividad 2.1. Pulsador.....	9
Actividad 2.2. Pulsador + NeoMatrix.....	11
Actividad 3.1. Pines táctiles.....	12
Actividad 4.1. Sensor de temperatura y humedad.....	14
Actividad 6.1. Sensor de sonido (micrófono).....	17
Actividad 7.1. Acelerómetro y giroscopio.....	18
Actividad 8.1. Medidor de consumo de energía (Intensidad).....	22
Actividad 8.2. Medidor de consumo de energía (Potencia).....	23
Actividad 9.1. Zumbador.....	25
Actividad 9.2. Zumbador RTTL.....	26
Actividad 10.1. Puerto de expansión I2C.....	27
Actividad 11.1. Interfaz de expansión de tarjeta SD.....	32
Actividad 11.2. Leer valores de tarjeta SD en arduinoblocks.....	35
Teoría: Comunicación ESP-NOW.....	36
Actividad 12.1. Comunicación ESP-NOW.....	37
Teoría: Sistemas de comunicaciones: Bluetooth y Wifi.....	39
Actividad 13.1. Comunicación Bluetooth.....	42
Actividad 13.2. Comunicación Bluetooth 2.....	46
Actividad 14.1. Comunicación WiFi.....	47
Actividad 14.2. WiFi: Servidor HTTP I.....	52
Actividad 14.3. WiFi: Servidor HTTP II.....	55
Actividad 14.4. WiFi: Servidor HTTP III.....	58
Actividad 14.5. WiFi: Servidor HTTP IV.....	60
Actividad 14.6. Contenido HTML.....	64
Teoría: MQTT.....	69
Actividad 15.1. MQTT: Enviar datos a ThingSpeak.....	71
Actividad 15.2. MQTT: Aplicación móvil de control MQTT I.....	78
Actividad 15.3. MQTT: Aplicación móvil de control MQTT II.....	87

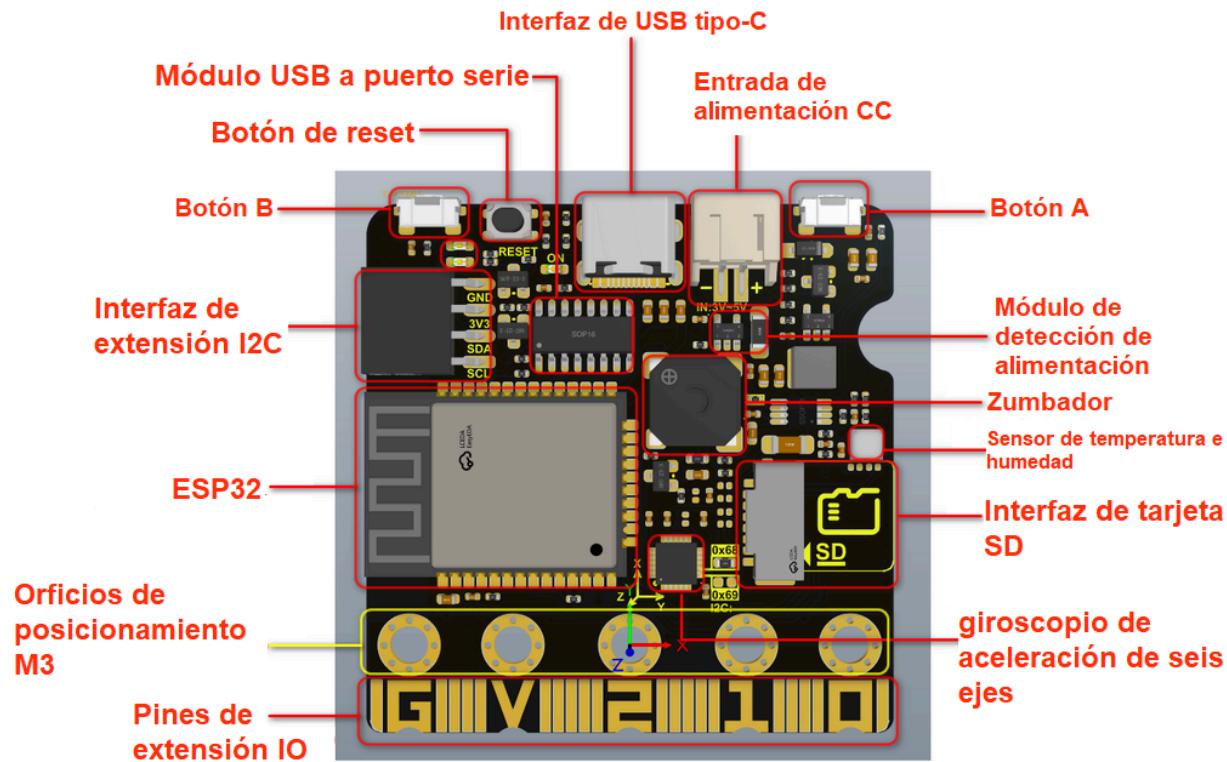
1 Descripción

El presente manual va sobre la placa **ESP32 micro:STEAMakers** y el entorno de programación **arduinoblocks**.

La placa **ESP32 micro:STEAMakers** es una placa con microcontrolador **ESP32**, es muy similar a la ESP32STEAMakers pero en este caso en vez de formato Arduino UNO, tiene el formato de la popular micro:bit permitiendo aprovechar prácticamente todo el hardware disponible de la micro:bit, pero dándole las potentes prestaciones como son la conectividad a internet o la matriz RGB con miles de colores diferentes.

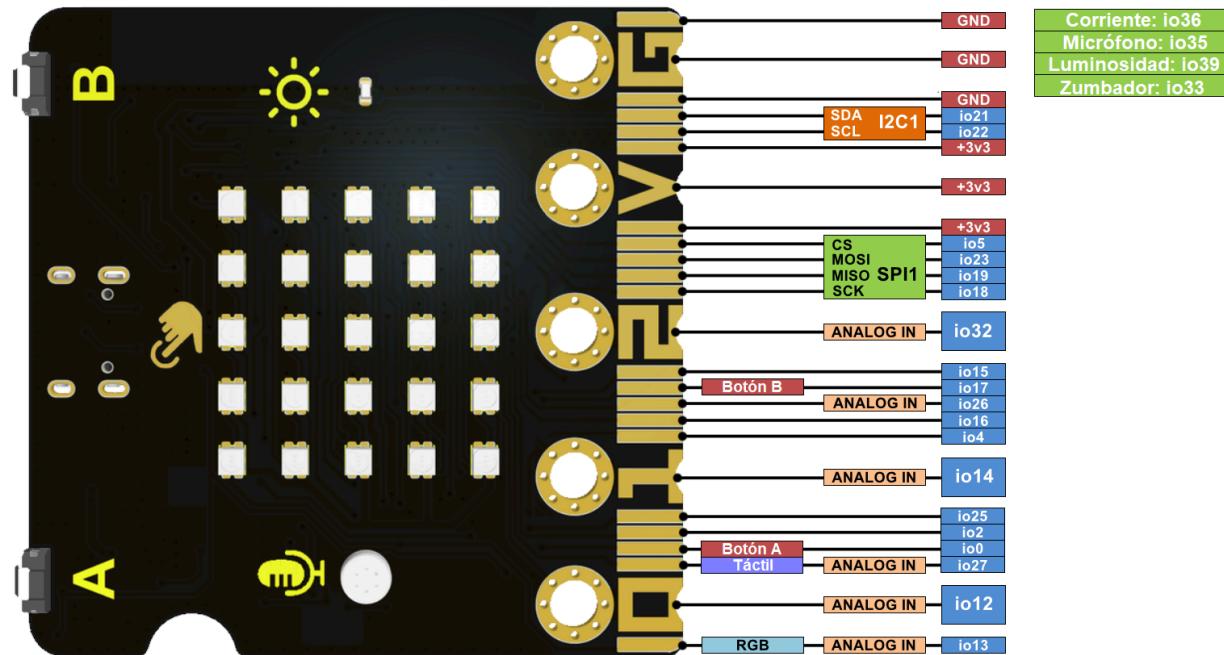
La placa lleva incorporado 2 botones, 1 logo táctil, 5x5 RGB dot matrix, sensor de luminosidad, micrófono, interfaz de tarjeta SD, altavoz, sensor de temperatura y humedad, medidor de energía, acelerómetro de 6 ejes y puerto I2C. Además, sus capacidades Bluetooth y WiFi de bajo consumo son excelentes para la comunicación inalámbrica.



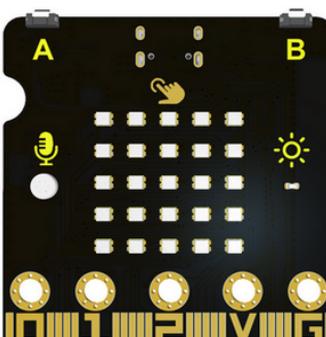
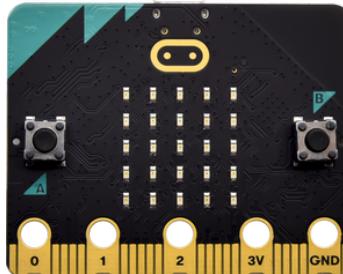


ATRÁS

La placa ESP32 está soldada con pines de expansión con dedos dorados, incluidos todos los pines IO, 19 puertos digitales, 7 puertos analógicos, interfaces I2C, UART y SPI.



Comparación de ESP32 micro:STEAMakers con Micro:bit

		
Nombre	ESP32 micro:STEAMakers	Micro:bit V2
Adecuado para	Primária, secundária	Primária
Procesador	ESP-WROOM-32	nRF52833-QIAA
Dimensiones	52*51mm	52*43mm
Capacidad de memoria	SRAM: 520 KB Flash: 4 MB	SRAM: 16 KB Flash: 256 KB
Lenguajes de programación	arduinoblocks Microblocks	Microblocks Makecode
Puertos E/S	Pines de E/S digitales: 19 Pines de entrada analógica: 6 Pines PWM: 13	Pines de E/S digitales: 19 Pines de entrada analógica: 6
Comunicación	1. Bluetooth 4.3 de bajo consumo 2. WiFi	1. Bluetooth 4.3 de bajo consumo
Funciones	1. 2 botones 2. 7 entradas de pines táctiles 3. acelerómetro de 3 ejes 4. giroscopio de 3 ejes 5. sensor de luminosidad 6. 5X5 RGB dot matrix 7. sensor de temperatura y humedad 8. interfaz de tarjeta SD 9. zumbador pasivo 10. micrófono 11. detección de alimentación 12. Interfaz I2C	1. 2 botones 2. 3 entradas de pines táctiles 3. acelerómetro de 3 ejes 4. magnetómetro de 3 ejes 5. sensor de luminosidad 6. 5X5 LED dot matrix 7. sensor de temperatura

2 Especificaciones técnicas

- Fuente de alimentación: Alimentación USB; CC(PH2.0); Alimentación del puerto de E/S Gold-finger
- Voltaje de funcionamiento: 3.3V
- Corriente de funcionamiento: 100mA / 0,1A
- ESP32 controladora principal:
 - Procesador: ESP32-D0WDQ6 (dual-core)
 - Frecuencia básica: hasta 240 MHZ
 - SRAM: 520 KB
 - Flash: 4 MB
 - Protocolo WiFi: 802.11 b/g/n (802.11n, velocidad hasta 150 Mbps)
 - Rango de frecuencia de funcionamiento: 2412 ~ 2484 MHz
 - Protocolo Bluetooth: cumple con los estándares Bluetooth v4.2BR/EDR y BLE
 - Bluetooth RF: Receptor NZIF con sensibilidad de -97 dBm
 - Audio bluetooth: Audio CVSD y SBC
- Sensores y actuadores integrados
 - 2 botones (A y B botones)
 - MPU6050 giroscopio de aceleración de 6 ejes
 - Velocidad máxima de rotación: 2000°/s
 - Rango de aceleración: ±2g, ±4g, ±8g, ±16g
 - Sensor de luminosidad: fototransistor ALS-PT19-315C
 - Micrófono: 4013-SMD
 - Zumbador: MLT-8530AAC3V
 - 25 RGB: WS2812-2020 RGB-LED
 - Sensor de temperatura y humedad: AHT20
 - Interfaz de expansión de tarjeta SD
 - Medidor de corriente: INA180A1IDBVR
 - Puerto de expansión I2C
- Interfaz de extensión:
 - 19 pines de E/S digitales
 - 2 DAC de 8 bits (io25, io26)
 - 7 pines táctiles (io2, io4, io12, io13, io15, io27, io32)
 - 13 pines PWM (io2, io4, io5, io12, io15, io16, io18, io19, io23, io25, io26, io27, io32)
 - Dispositivo UART de 3 vías (cualquier pin), soporte para control de flujo de hardware y DMA
 - 2 dispositivos I2C (cualquier pin), admiten modo host o esclavo
 - Control remoto IR (cualquier pin) Transceptor IR de 8 canales, admite diferentes estándares de forma de onda.

3 Introducción de arduinoblocks

El siguiente enlace os lleva a wiki de Innova Didàctic y ahí podemos ver la introducción de **arduinoblocks**.

[Introducción de arduinoblocks](#)

Para crear un proyecto con **arduinoblocks** aquí tenemos un enlace:

4 Actividades con ESP32 micro:STEAMakers

En este manual vamos a hacer las siguientes actividades con la placa **ESP32 micro:STEAMakers** y **arduinoblocks**. Todas las actividades son con sensores y actuadores integrados de la placa. Excepto la parte de **I2C** ya que es necesario tener componentes externos para poder conectar el **puerto de expansión I2C**.

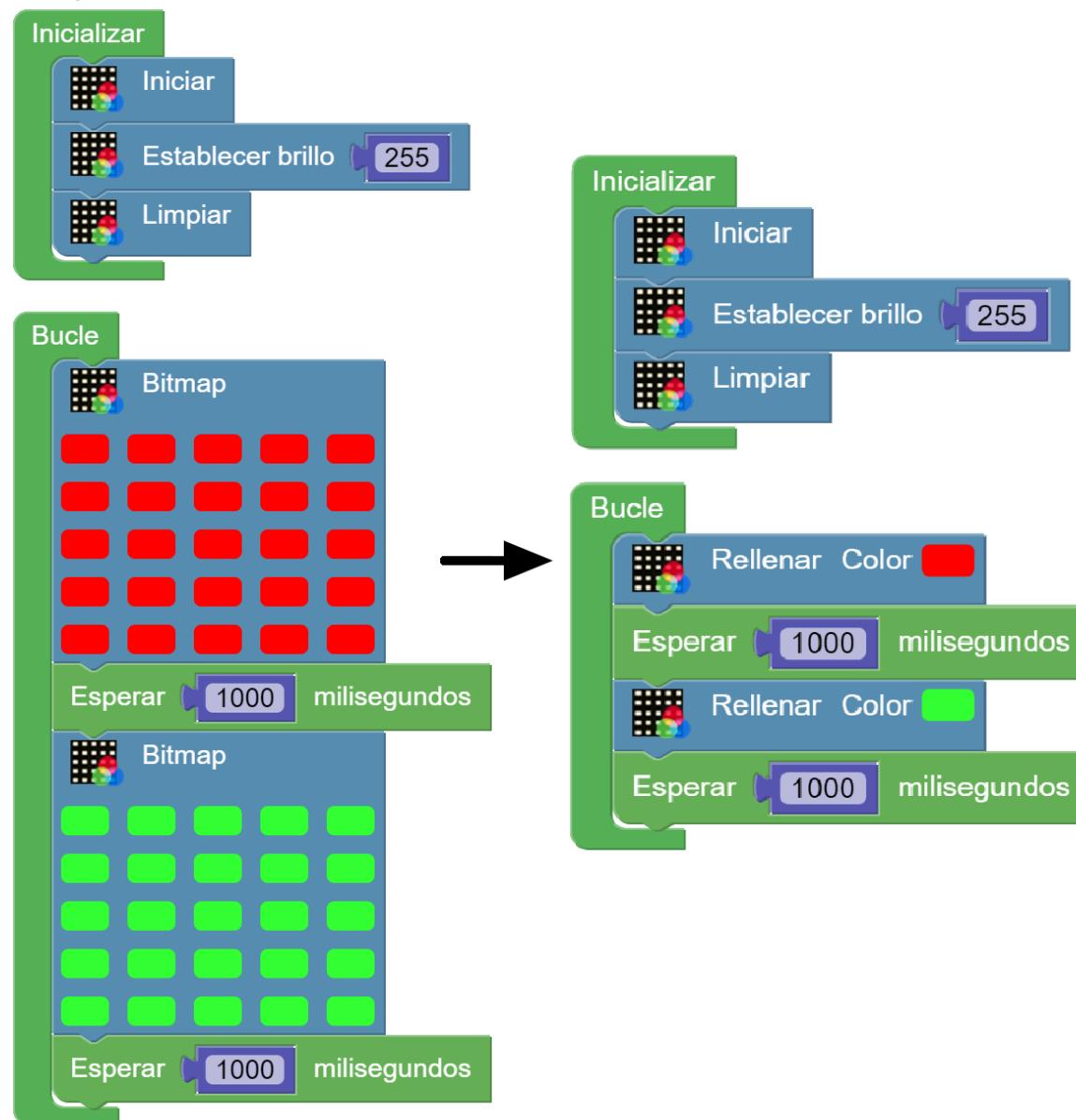
- Actividad 1: NeoMatrix
- Actividad 2: Pulsadores y NeoMatrix
- Actividad 3: Pines táctiles
- Sensores integrados:
 - Actividad 4: Sensor de temperatura y humedad
 - Actividad 5: Sensor de luminosidad
 - Actividad 6: Sensor de sonido(Micrófono)
 - Actividad 7: Acelerómetro y giroscopio
 - Actividad 8: Medidor de consumo de energía
- Actuadores integrados:
 - Actividad 9: Zumbador
- Actividad 10: Puerto de expansión I2C
- Actividad 11: Interfaz de expansión de tarjeta SD
- Actividad 12: Comunicación ESP-NOW
- Actividad 13: Comunicación Bluetooth
- Actividad 14: Comunicación WiFi

Actividad 1.1 NeoMatrix 1

En esta actividad vamos a encender la matriz RGB que tiene 25 neopixels que están al pin IO13. Los neopixels son una tira de luces RGB (Red, Green, Blue) que cuentan con un circuito lógico integrado, lo que hace posible controlar con un solo pin el color de todos los LEDs.

Vamos a encender la matriz entera de un color y después de un tiempo tendrá otro color. Hay dos formas de hacer el mismo programa como vemos en la siguiente imagen.

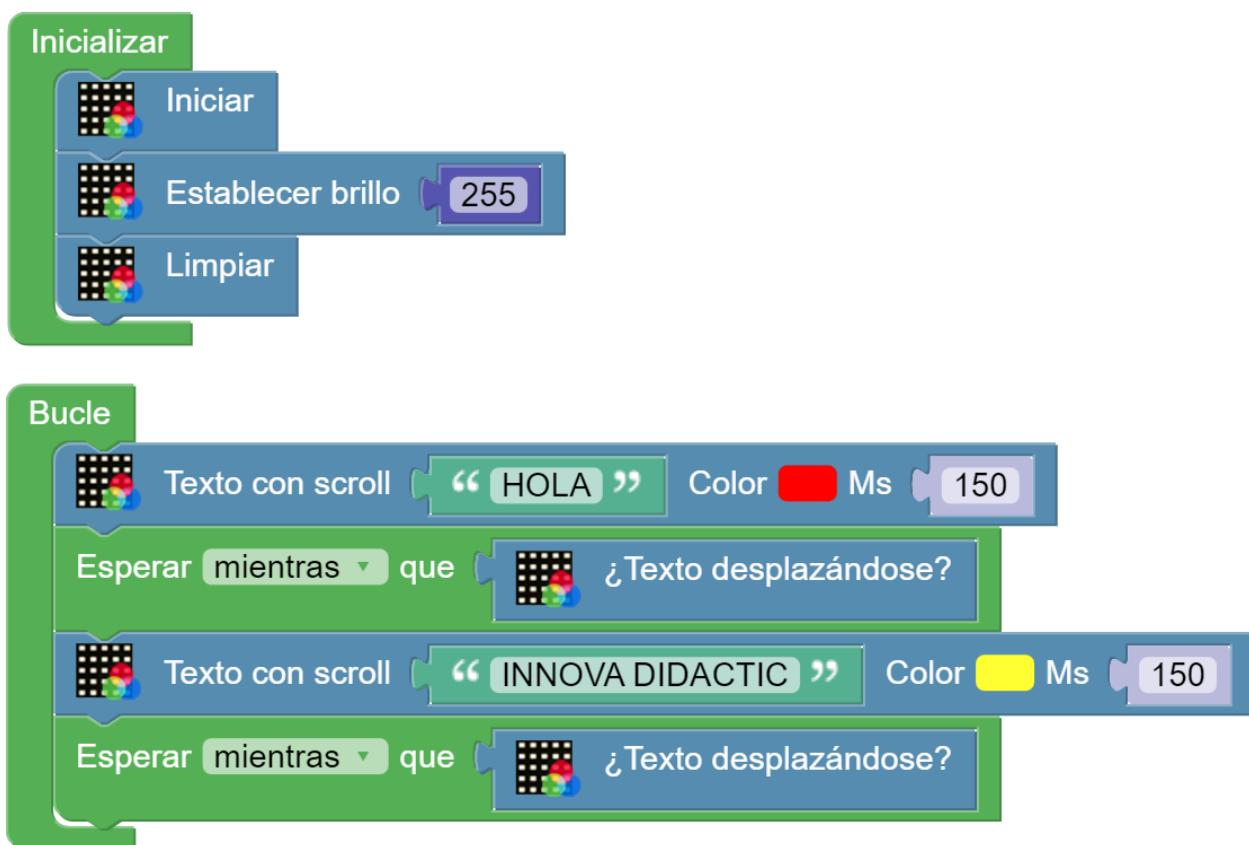
Programa:



Actividad 1.2. NeoMatrix 2

En esta actividad también vamos a encender la matriz de RGB pero ahora en vez de llenar con colores le vamos a poner texto desplazando. En **arduinoblocks** tenemos un bloque de “Texto con scroll” para visualizar texto en la matriz. Hay más funciones para hacer más cosas en la matriz.

Programa:

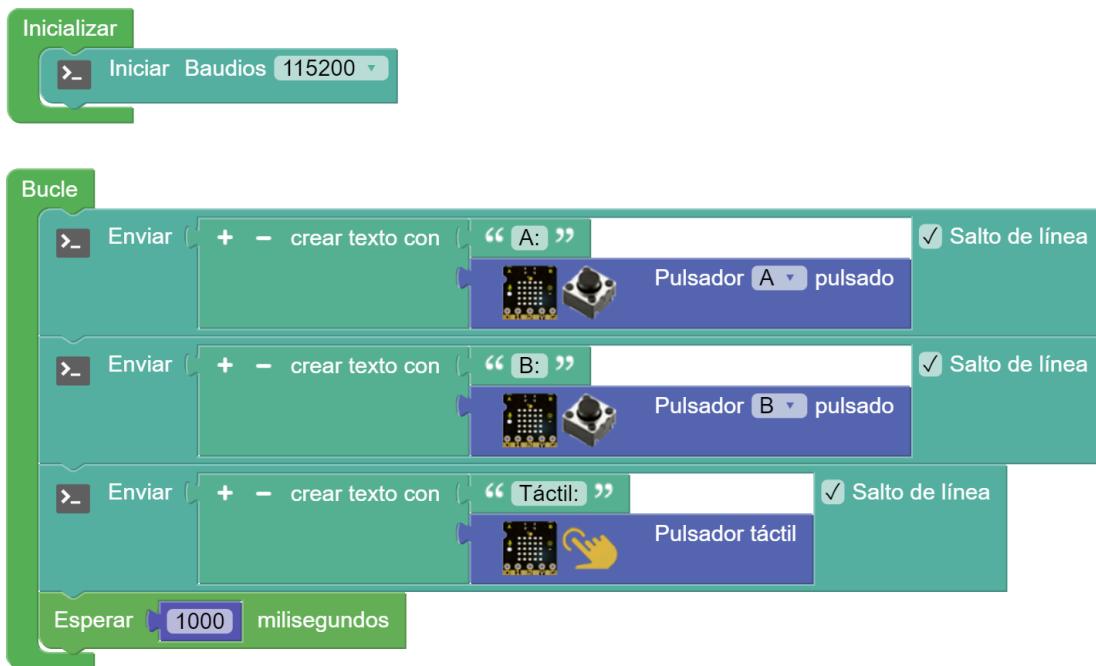


Actividad 2.1. Pulsador

En esta actividad vamos a ver cómo actúan los **pulsadores A, B y el táctil**, el primer paso es saber si al pulsar da 0 o 1 y para leer los valores vamos a usar la **Consola** que es lector de datos por el puerto serie y esta función nos ofrece **arduinoblocks**.

Los bloques de puerto serie los vamos a encontrar en el apartado de **“Comunicaciones”** y allí en **“Puerto Serie”**. El bloque de “crear texto con” está en el apartado **“Texto”**.

Programa:



Para ver los valores tenemos que abrir la **Consola**.



Al pulsar sobre la Consola se abrirá la siguiente ventana dónde tenemos la opción de elegir la velocidad de comunicación en **Baudrate** y al lado tenemos el botón de **“Conectar”** para poder empezar a ver los valores en la pantalla.

Consola serie

×

Baudrate: 115200

Conectar

Desconectar

Limpiar



Enviar

Táctil:0

A:0

B:0

Táctil:0

A:0

B:0

Táctil:1

A:1

B:1

Táctil:0

A:1

B:1

Táctil:0

A:1

B:0

Táctil:0

A:0

B:0

Táctil:0

A:0

B:0

Táctil:0

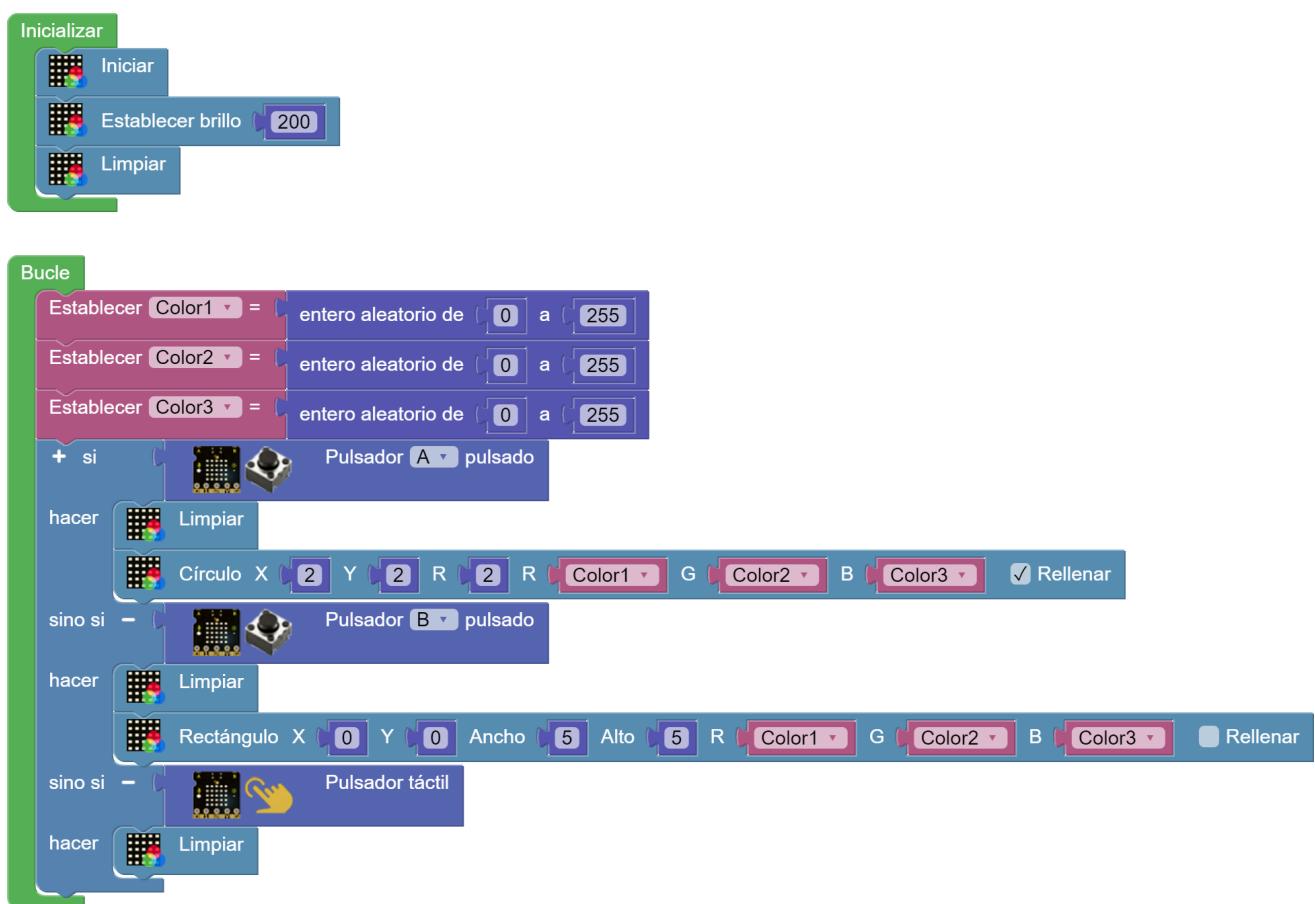
Después de haber visto la ventana podemos llegar a la siguiente conclusión:

Pulsador	Suelto	Pulsado
A	0	1
B	0	1
Táctil	0	1

Actividad 2.2. Pulsador + NeoMatrix

En esta actividad vamos a hacer la combinación de pulsadores más NeoMatrix. Dependiendo de qué pulsador apretamos vamos a ver una cosa o otra en la NeoMatrix. Al apretar el pulsador “A” veremos en la pantalla un círculo de un color aleatorio y al apretar el pulsador “B” veremos rectángulo de algún otro color aleatorio. Y si pulsamos el táctil nos va dejar la NeoMatrix limpia.

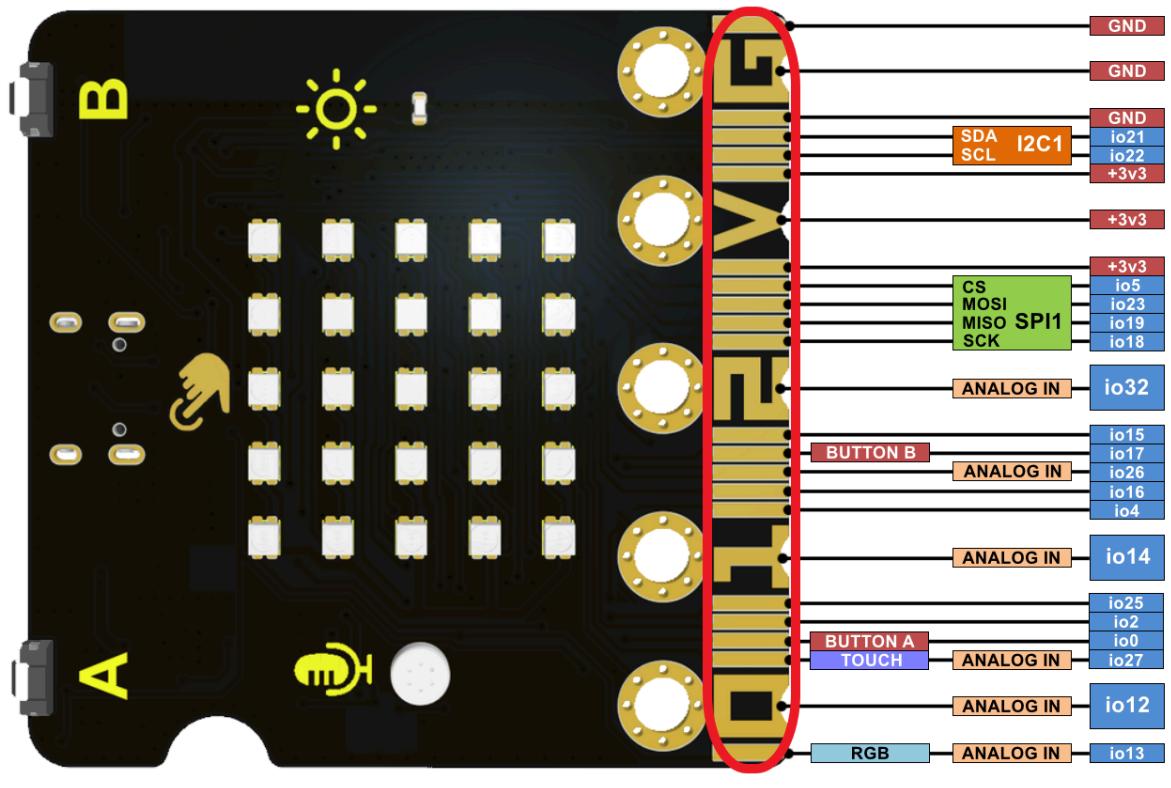
Programa:



Actividad 3.1. Pines táctiles

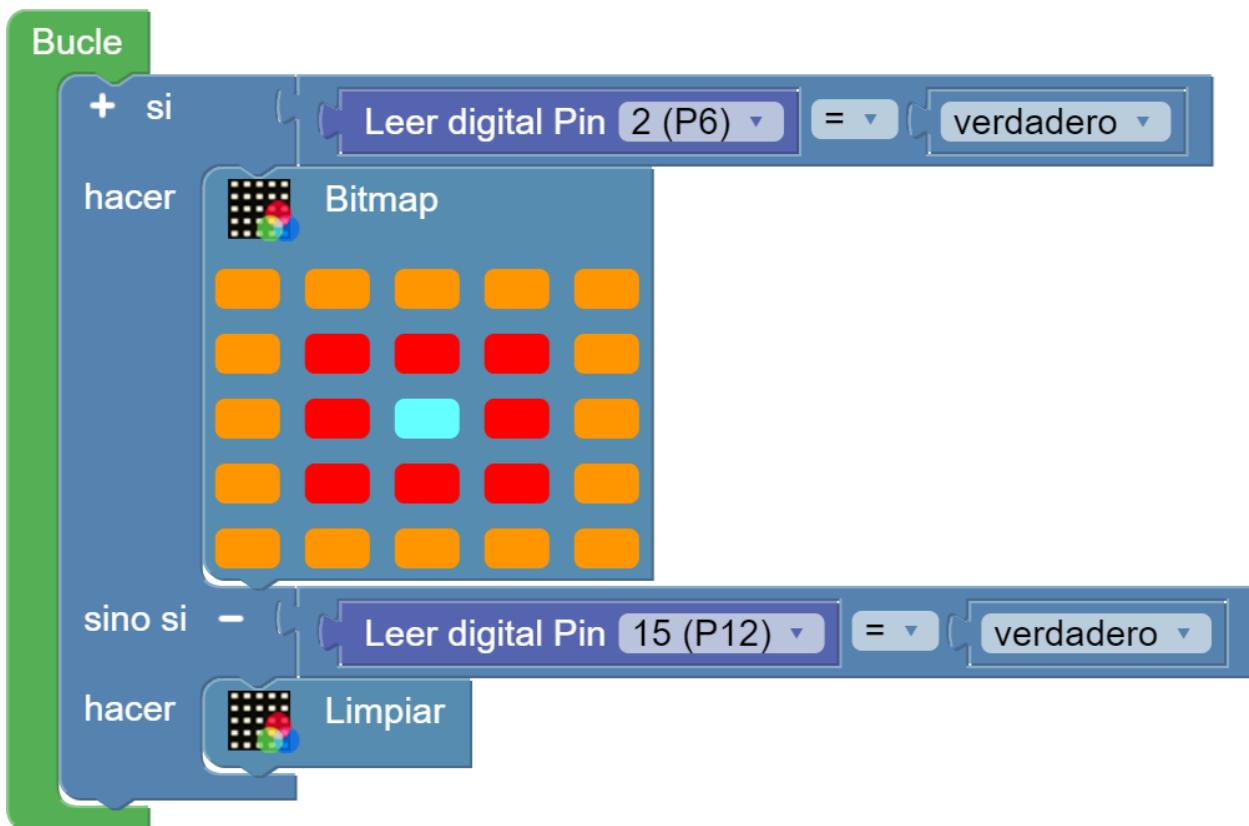
En esta actividad vamos a utilizar algunos de los 7 pines táctiles que tiene la placa ESP32 micro:STEAMakers. Los pines táctiles son los siguientes: io2, io4, io12, io13, io15, io27, io32 y están situados en la parte de abajo que podemos ver en la siguiente imagen.

En este ejercicio vamos a usar dos pines táctiles que son io2 y io15. Cuando pulsamos io2 la NeoMatrix se ilumina de un color y cuando le damos al pin io 15 se limpia la NeoMatrix. El bloque de “Leer digital Pin” está en el apartado de “Entradas/Salida”.



En la siguiente página tenemos el programa.

Programa:

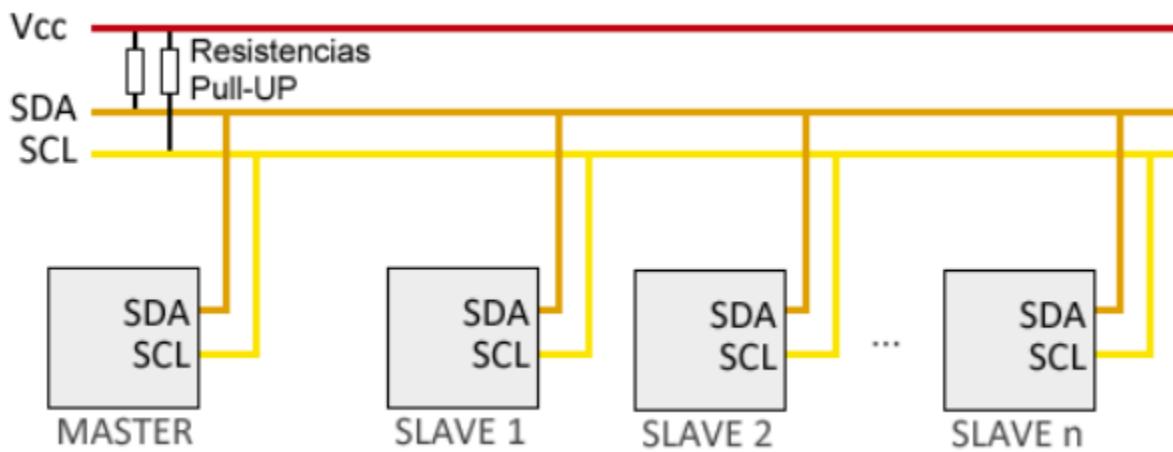


Actividad 4.1. Sensor de temperatura y humedad

En esta actividad utilizaremos el sensor de temperatura y humedad **AHT20** que se comunica con la ESP32 por la comunicación **I2C**.

¿Qué es I2C?

I2C es un protocolo de comunicación que requiere únicamente dos cables para su funcionamiento, uno para la señal de reloj (**SCL**) y otro para el envío de datos (**SDA**), lo cual es una ventaja frente al bus SPI.



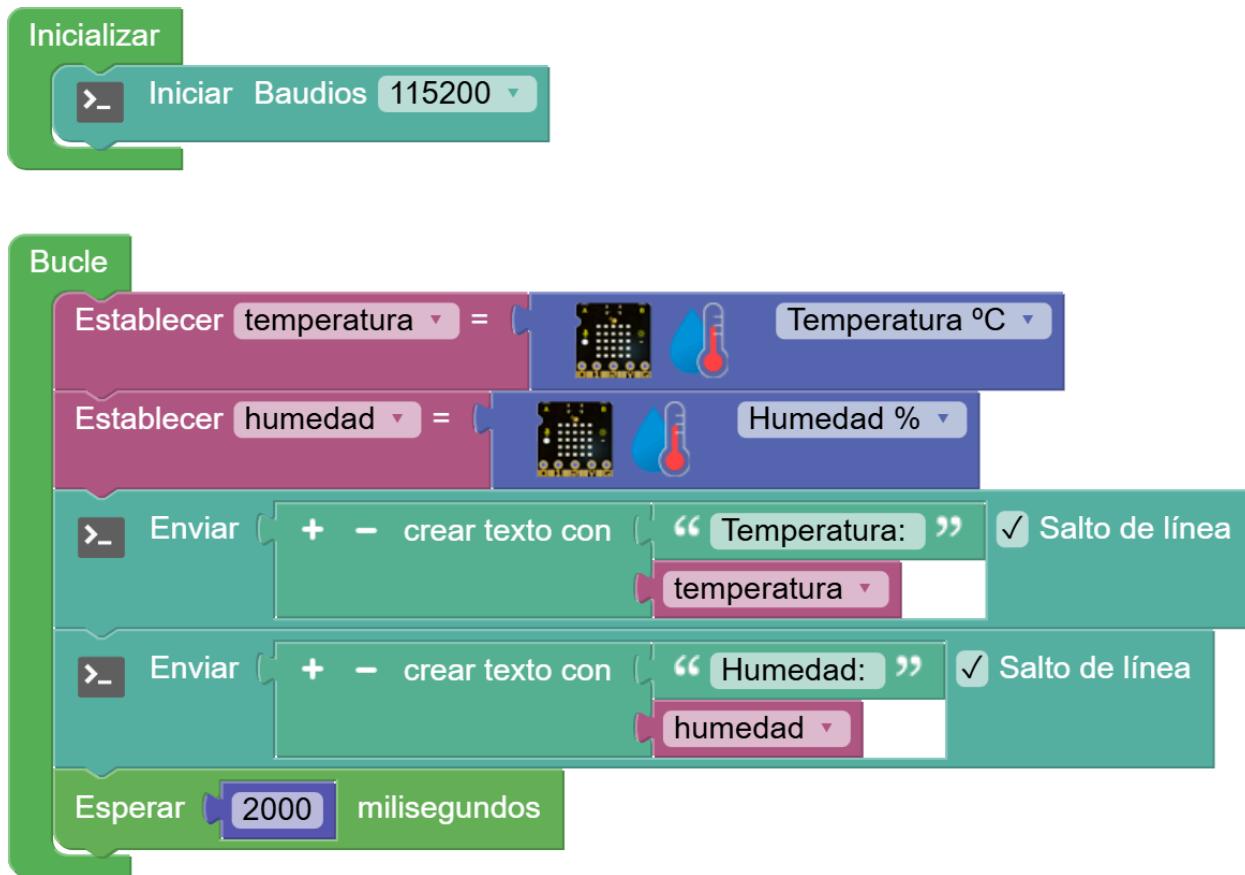
En el bus, cada dispositivo dispone de una dirección, que se emplea para acceder a los dispositivos de forma individual. En general, cada dispositivo conectado al bus debe tener una dirección única. Si tenemos varios dispositivos similares tendremos que cambiar la dirección o, en caso de no ser posible, implementar un bus secundario.

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

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

En esta actividad vamos a leer los valores del sensor de temperatura y humedad que lleva integrado la placa. Es un sensor **AHT20** que es un sensor I2C. Los valores del sensor los vamos a ver en la **Consola**.

Programa:

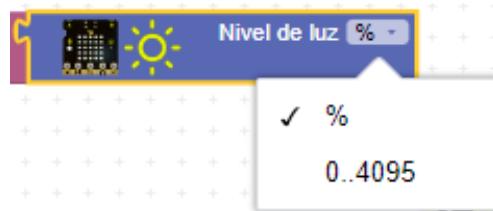


Una vez subido el programa en la placa podemos ver los valores en la **Consola**.

Actividad 5.1. Sensor de luminosidad

En esta actividad vamos a utilizar el sensor de luz ambiental. El sensor de luz ambiental está en el pin **IO39**. Básicamente, lo que vamos a hacer en la actividad es leer los valores del sensor en **arduino blocks** con la Consola.

En **arduinoblocks** podemos leer los valores de sensor en **porcentaje o de 0 a 4095**. En el ejemplo vamos a hacer en porcentaje.



Programa:



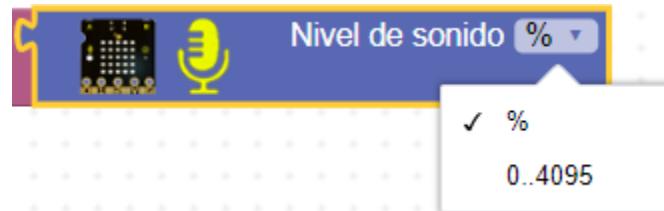
Actividad 6.1. Sensor de sonido (micrófono)

En esta actividad vamos a utilizar el sensor de sonido o micrófono para detectar el ruido. El sensor de sonido está en el pin **IO35**.

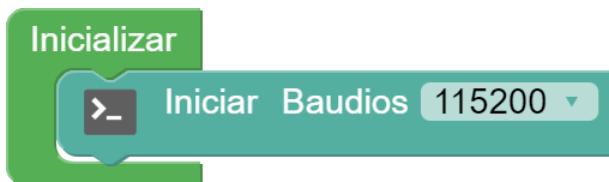
En esta actividad vamos a usar la Consola de **arduinoblocks** para visualizar los valores que nos da el sensor.

En **arduinoblocks** podemos leer los valores de sensor en **porcentaje o de 0 a 4095**.

En el ejemplo vamos a hacer en porcentaje.



Programa:



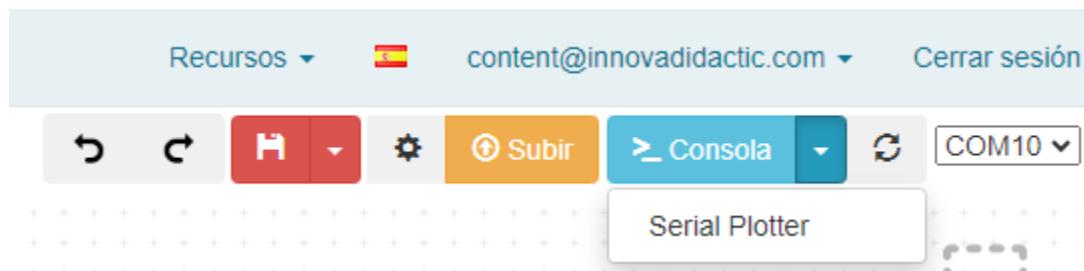
Actividad 7.1. Acelerómetro y giroscopio

En esta actividad vamos a usar el acelerómetro y el giroscopio. Vamos a repartir la actividad en 2 partes. Primero vamos a hacer una actividad con los **parámetros de aceleración** y después vamos a hacer una actividad con los **parámetros de giroscopio**.

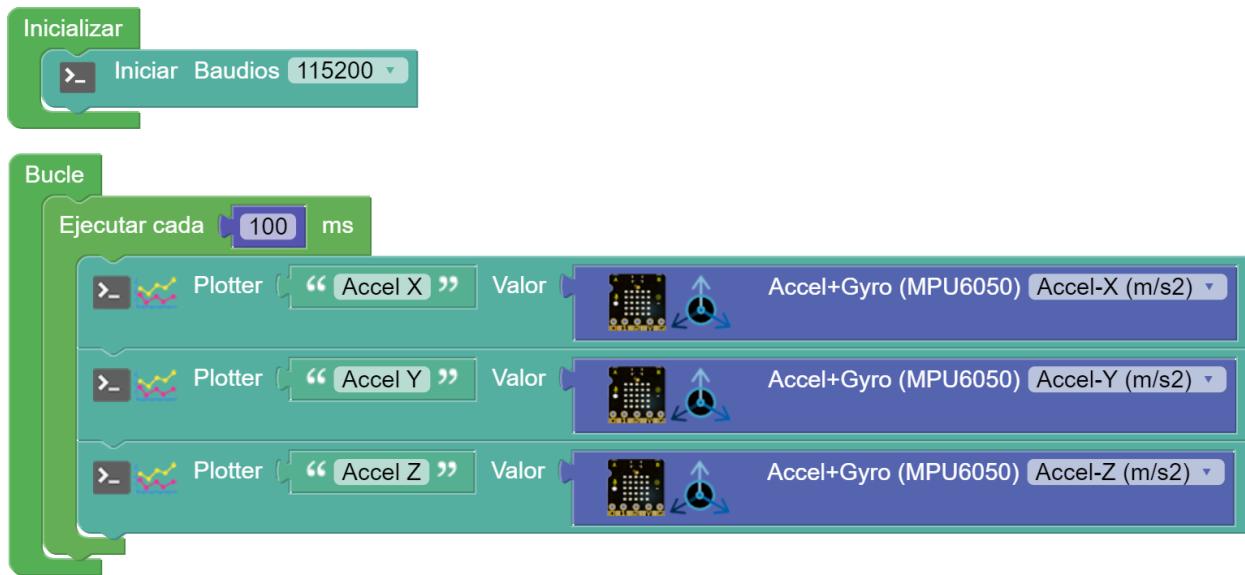
Para ver las aceleraciones de los 3 ejes vamos a usar el Serial Plotter de **arduino blocks**.

Serial Plotter es una función de **arduino blocks** para poder visualizar gráficamente los valores del sensor, variables numéricas, etc. También permite descargar en formato CSV el primero de los sensores de la gráfica.

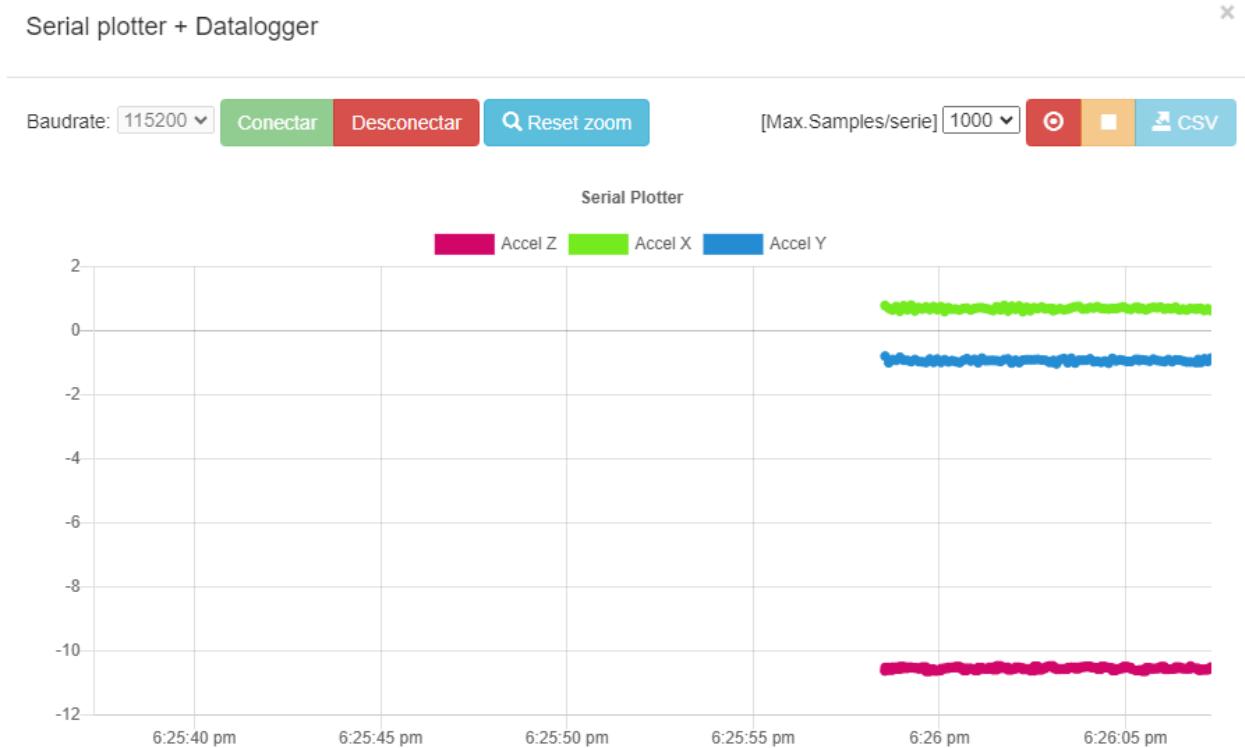
En el apartado de Puerto serie disponemos de un bloque para poder visualizar datos.



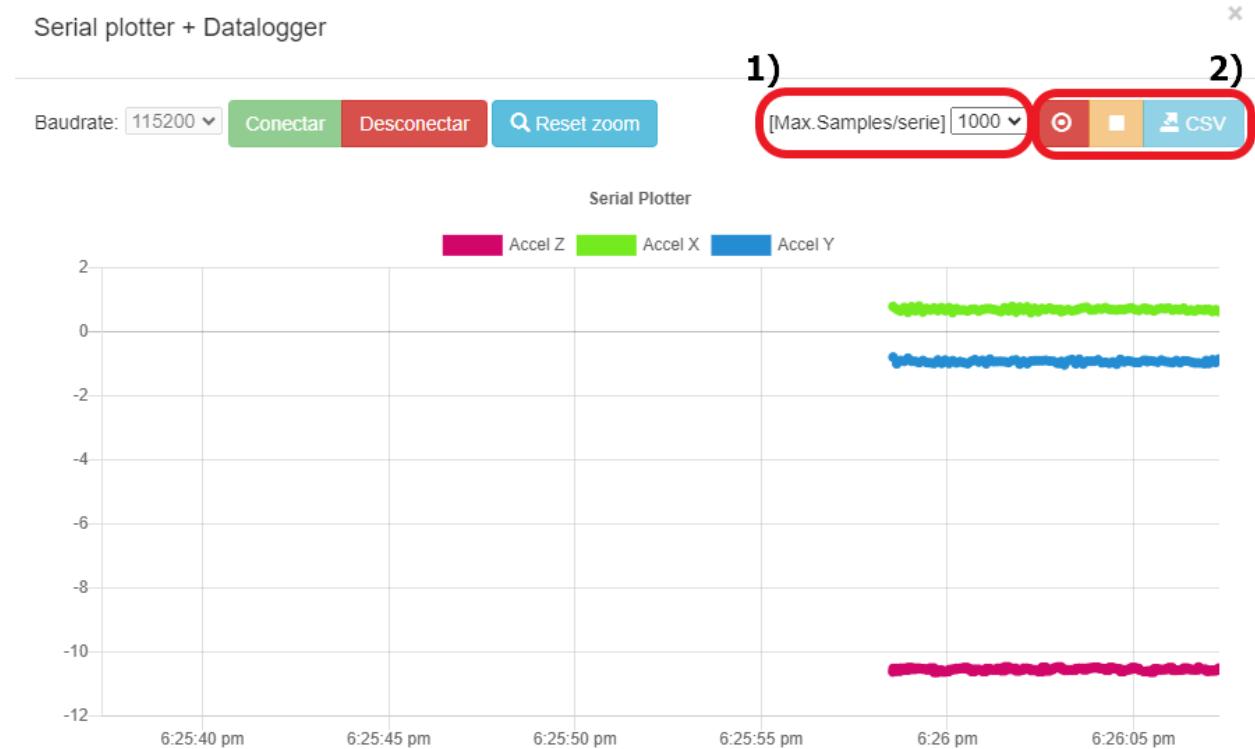
Con este programa vamos a leer los datos del acelerómetro de los 3 ejes: **eje-X**, **eje-Y** y **del eje-Z**.



Una vez el programa subido vamos a abrir Serial Plotter y ver representados los valores en forma de gráfico.



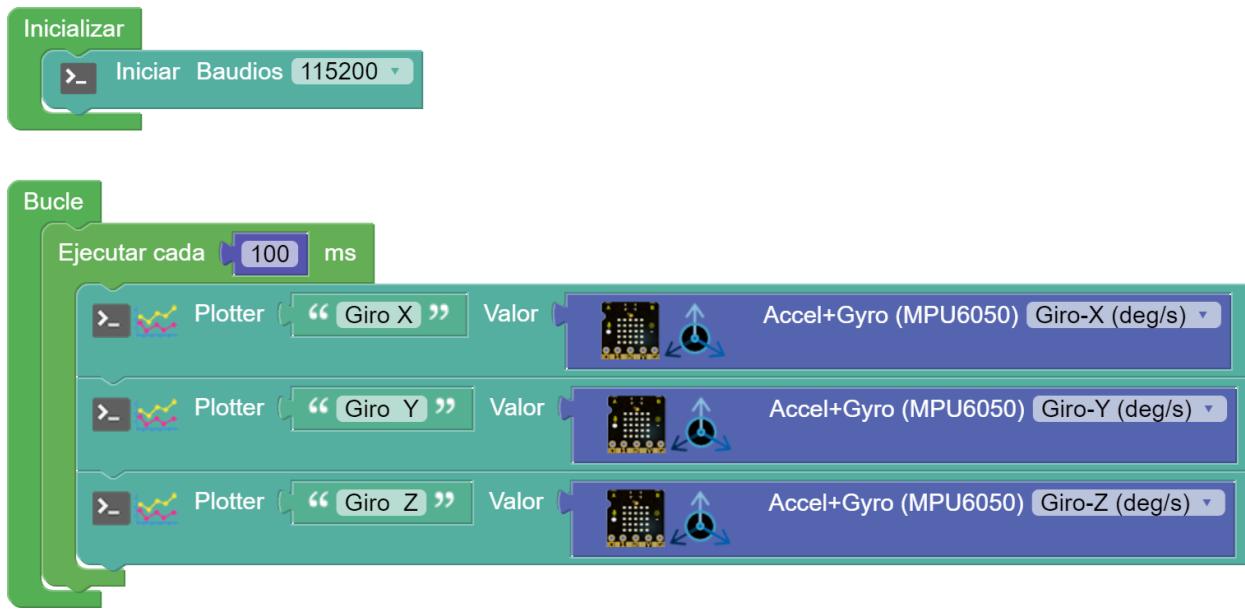
Para descargar los datos en formato de **CSV**:



- 1) En el desplegable le podemos decir el número de muestras que queremos guardar en el archivo **CSV**.
- 2) Con el botón de grabar podemos empezar a grabar los datos hasta el límite o hasta que pulsamos el botón de pausar. Al lado tenemos el botón de CSV que es el que nos descarga las dadas capturadas en formato de **CSV**.

Ahora vamos a hacer el segundo programa para el giroscopio, en **arduinoblocks** nos va a dar valores en deg/s que es velocidad de rotación. En el siguiente programa vamos a ver 3 giros: **Giro-X, Giro-Y y Giro-Z**.

Programa:



Una vez subido el programa vamos a abrir el **Serial Plotter** y ver los valores en **arduinoblocks**.

Actividad 8.1. Medidor de consumo de energía (Intensidad)

La placa **ESP32 micro:STEAMakers** lleva incorporado el medidor de intensidad. Los bloques de energía están en el apartado de Sensores de “**micro:STEAMakers**”. En este ejercicio vamos a ver los valores de la placa **ESP32 micro:STEAMakers** con la Consola de **arduinoblocks**. Si apretamos el **botón A** enciende la NeoMatrix y pasa el consumo en amperios a la **Consola** y si apretamos el **botón B** se apaga la NeoMatrix y vuelve a enviar el consumo en amperios a la **Consola**.

Programa:

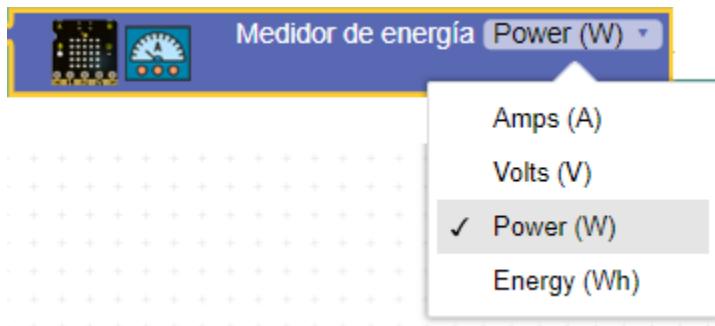


Actividad 8.2. Medidor de consumo de energía (Potencia)

En esta actividad vamos a usar el mismo ejemplo que antes sólo le cambiamos una cosa que en lugar de intensidad en este caso vamos a ver la potencia de la placa cuando está encendida la NeoMatrix y la potencia cuando está apagado.

El bloque de potencia en **arduinoblocks** es el mismo bloque de intensidad que en el desplegable podemos escoger. Tenemos 4 opciones que son :

- **Intensidad : Amps (A)**
- **Voltaje : Volts (V)**
- **Potencia : Power (W)**
- **Energía : Energy (Wh)**



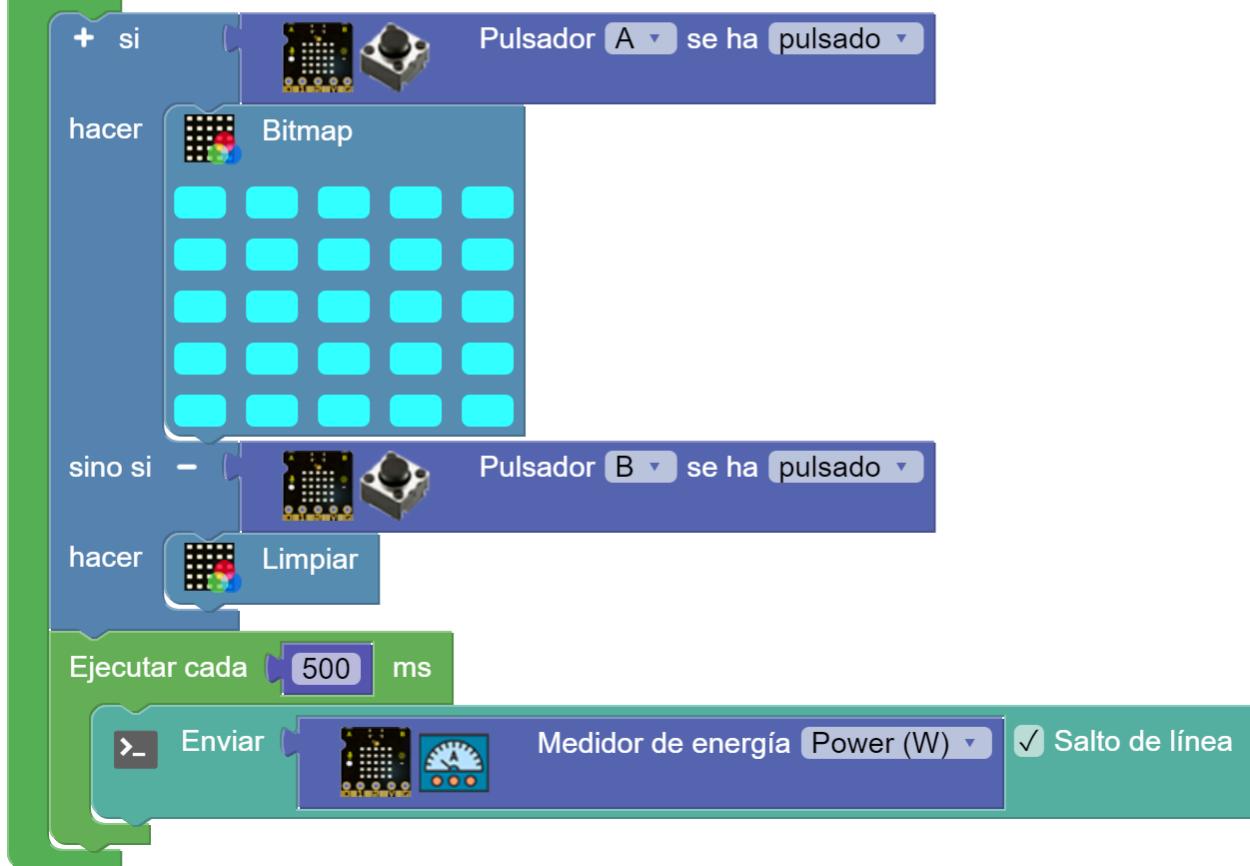
En este ejemplo vamos a usar el bloque de “**ejecutar cada**” que es un bloque que ejecuta los bloques de su interior si el tiempo transcurrido desde su última ejecución es igual o mayor que el tiempo determinado. La diferencia con el **bloque de “espera”** es que este bloque **no bloquea la ejecución del programa**. Este bloque lo encontramos en el apartado de “**Tiempo**”.

El programa de ejemplo está en la siguiente página.

Programa:



Bucle

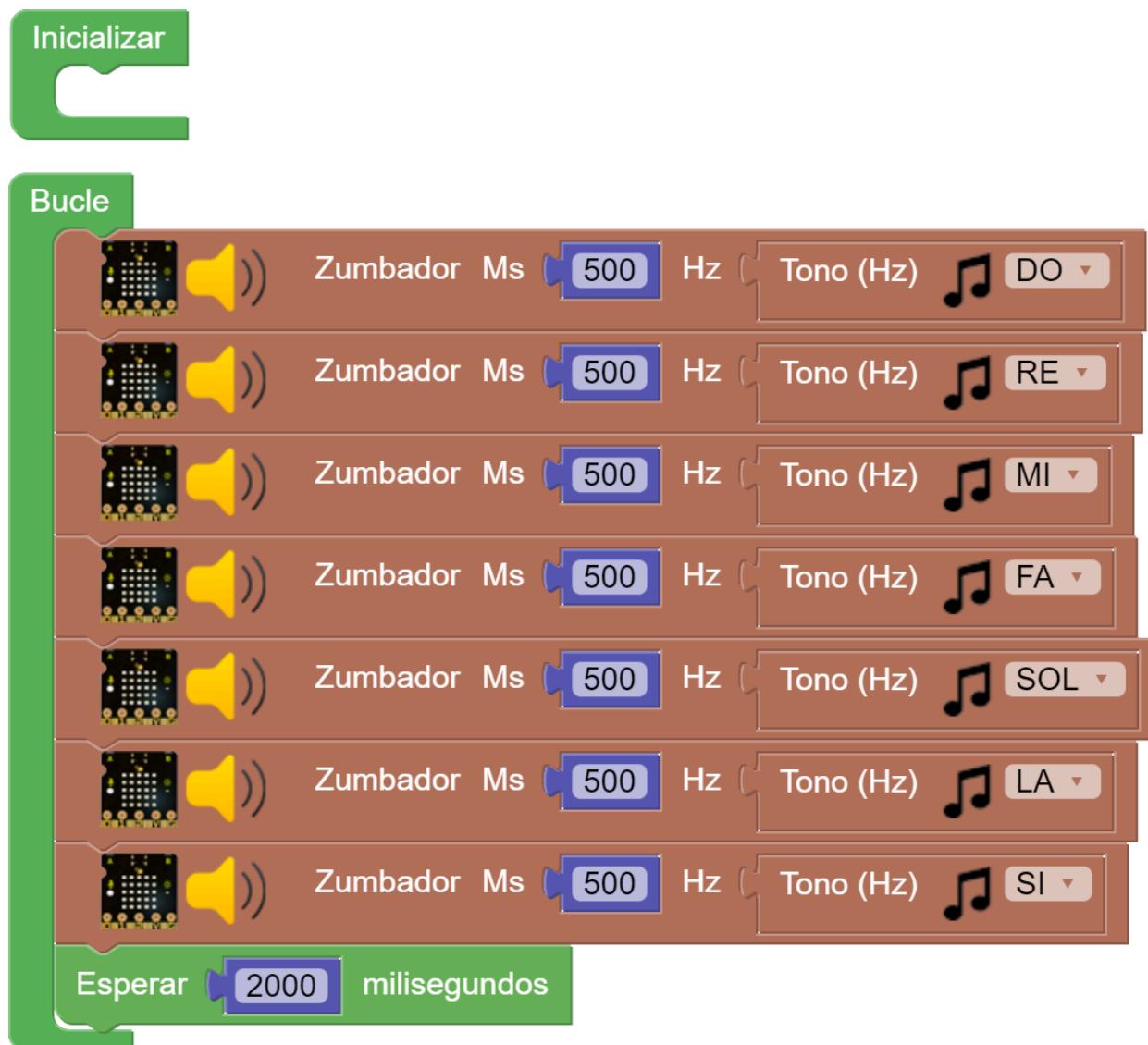


Actividad 9.1. Zumbador

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

El zumbador en la placa **ESP32 micro:STEAMakers** está en el pin **IO33**. El bloque de Zumbador está en el apartado de “**actuadores**” en **micro:STEAMakers**”.

Programa:

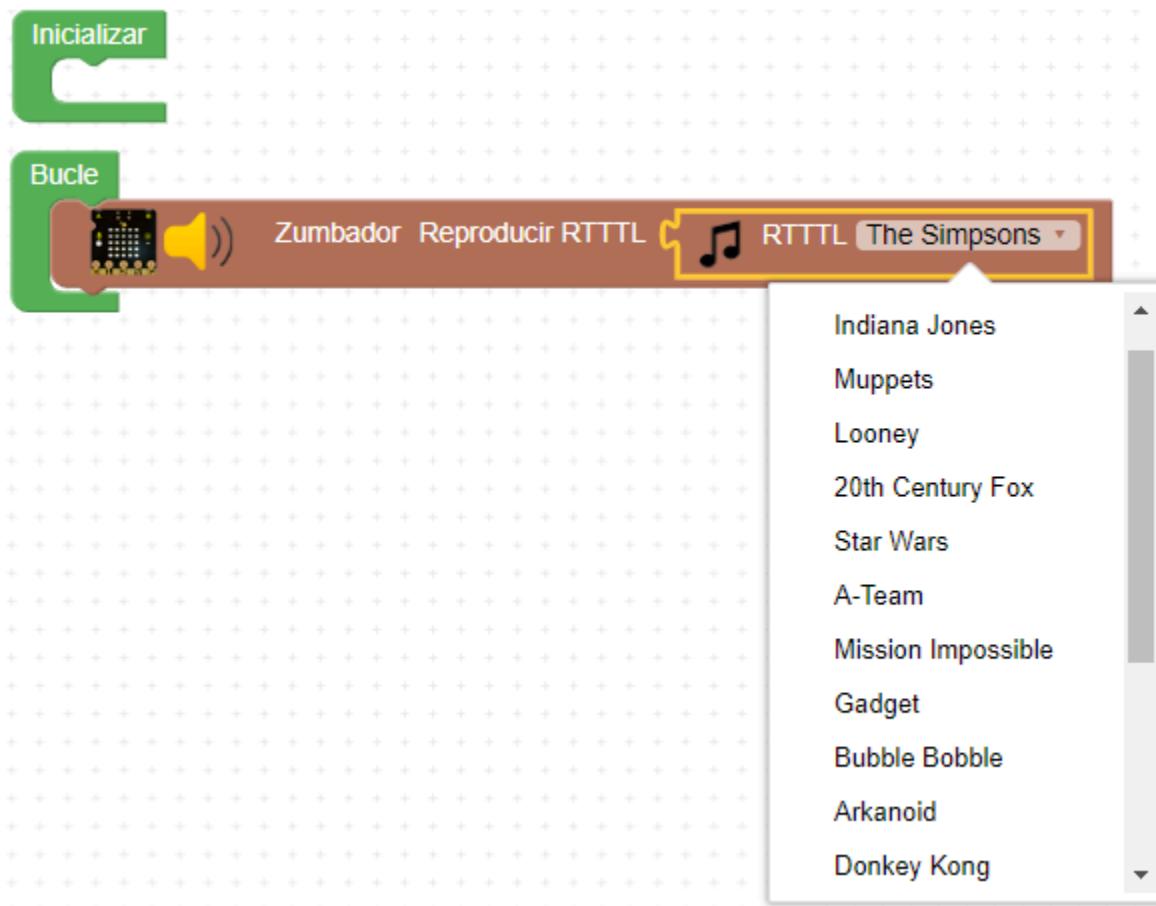


Actividad 9.2. Zumbador RTTL

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

En esta actividad vamos a usar el zumbador de la placa pero ahora vamos usar el bloque “Zumbador RTTTL” para las melodías. El bloque de “Zumbador RTTTL” está en el apartado de “actuadores” en **micro:STEAMakers**.

Programa:



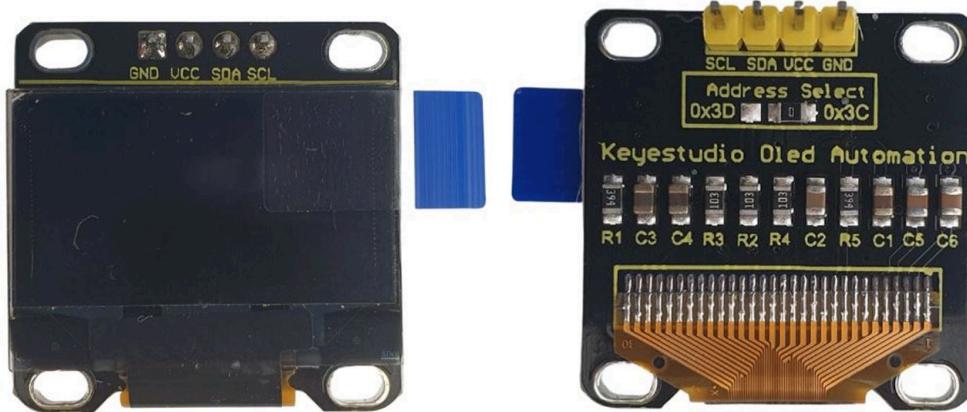
Actividad 10.1. Puerto de expansión I2C

La placa **ESP32 micro:STEAMakers** lleva un puerto de expansión **I2C**. A este bus de comunicación le podemos conectar varios tipos de dispositivos:

- Pantalla OLED
- Acelerómetros
- Giroscopios
- Sensores de temperatura
- Sensor de color
- Sensor de movimiento
- RFID
- Controladores de servomotor
- Otros dispositivos I2C

Vamos a realizar una actividad con la **pantalla OLED**.

La **pantalla OLED** es una pantalla que funciona mediante **bus de comunicaciones I2C**. Como solamente está encendido el píxel elegido es un tipo de pantalla muy eficiente. Es una pantalla muy útil para mostrar información o imágenes.



Conectaremos la pantalla al **conector I2C** de la placa **ESP32 micro:STEAMakers**. Para programar la pantalla OLED tenemos bloques específicos. Estos bloques son los que están en el apartado “**Visualización → Pantalla OLED**”.

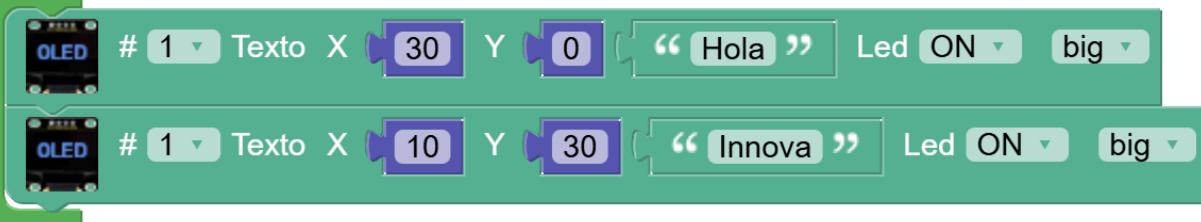
Realizaremos un programa que muestre algún texto por la pantalla OLED. De este texto podremos elegir el tamaño y la posición a dónde colocar.

Programa:

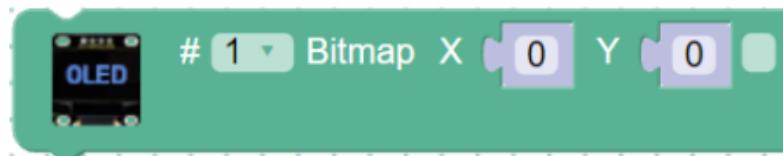
Iniciar



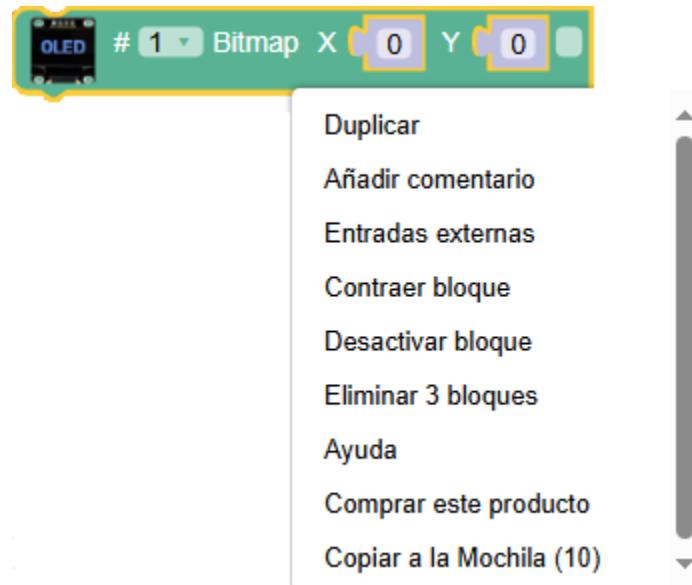
Bucle



También podemos mostrar imágenes por la pantalla OLED. Vamos a ver cómo se realiza este proceso. Primero elegiremos el bloque para introducir un *Bitmap*.



A continuación, pulsamos con el botón derecho del ratón y aparece un desplegable con diferentes opciones.



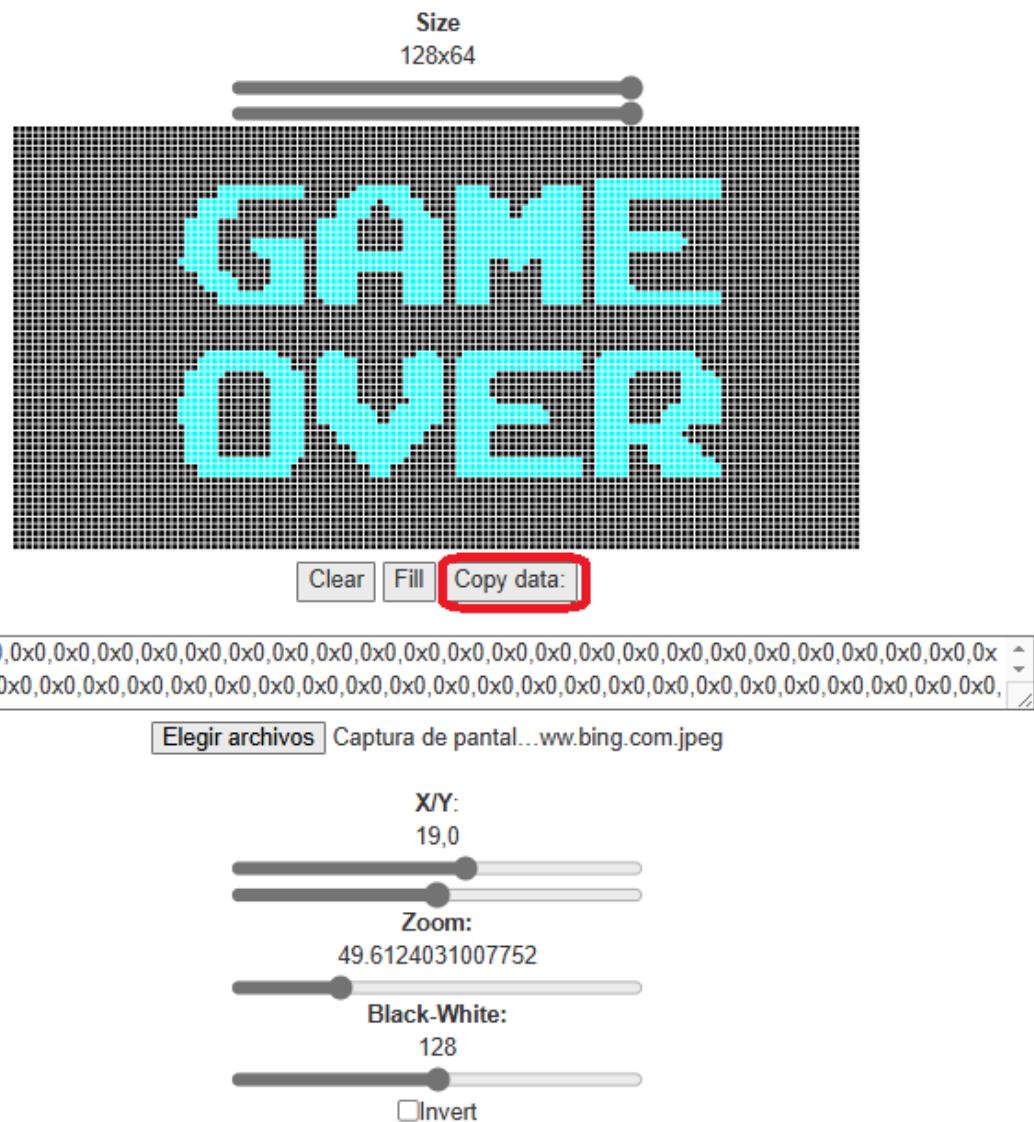
Elegiremos la opción *Ayuda* y nos mostrará una pantalla en el explorador (OLED - Bitmap Data).

OLED - Bitmap Data

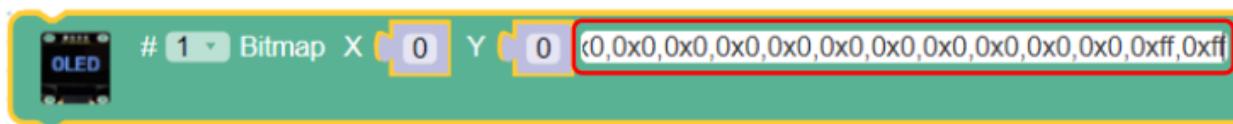


En el botón *Elegir archivos* podemos cargar una imagen. Una vez cargada la imagen podemos invertir, desplazar, etc.

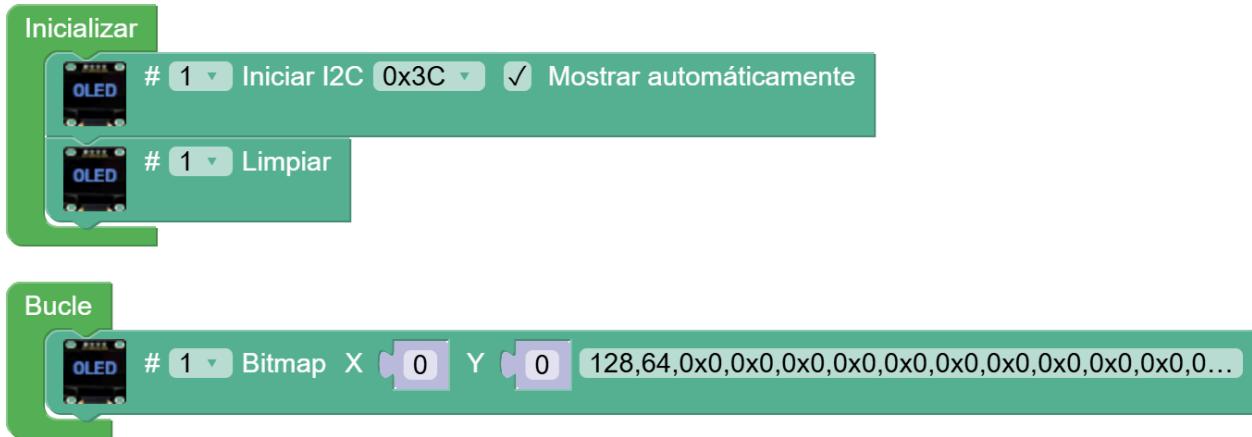
OLED - Bitmap Data



A continuación, copiamos los datos pulsando sobre Copy data. Y los pegamos dentro del bloque.



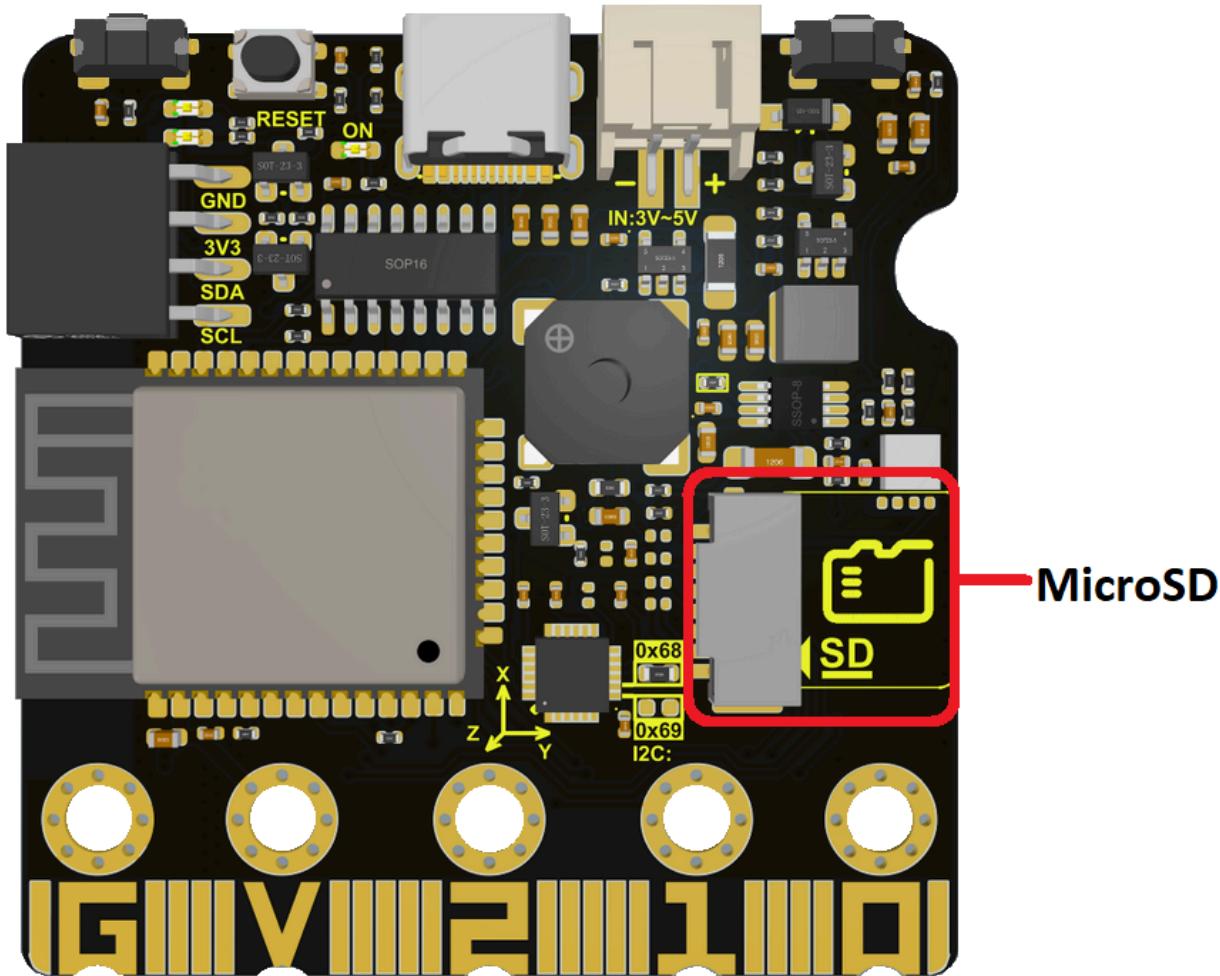
Ahora ya podemos visualizar la imagen en la pantalla OLED con este sencillo programa.



Actividad 11.1. Interfaz de expansión de tarjeta SD

La placa **ESP32 micro:STEAMakers** dispone de un zócalo para poder insertar tarjetas **micro-SD**. Para que funcione tiene que estar formateada en formato **FAT32** permitiendo tamaños de hasta **2TB** y archivos de un tamaño máximo de **2GB**. La tarjeta le da gran flexibilidad a la placa ya que permite poder **escribir y leer datos**. Una de las funcionalidades más útiles es poder almacenar los datos de los sensores a modo de **Datalogger**. También nos permitirá leer datos de la tarjeta (por ejemplo, configuraciones de conexiones Wifi) para poder utilizarlas directamente en un programa.

El zócalo para introducir la tarjeta micro SD se encuentra en la parte trasera de la placa.



Para programar la placa para leer o escribir en la tarjeta SD tenemos una serie de bloques en arduinoblocks. Este apartado está en “**micro:STEAMakers → Tarjeta SD**”.

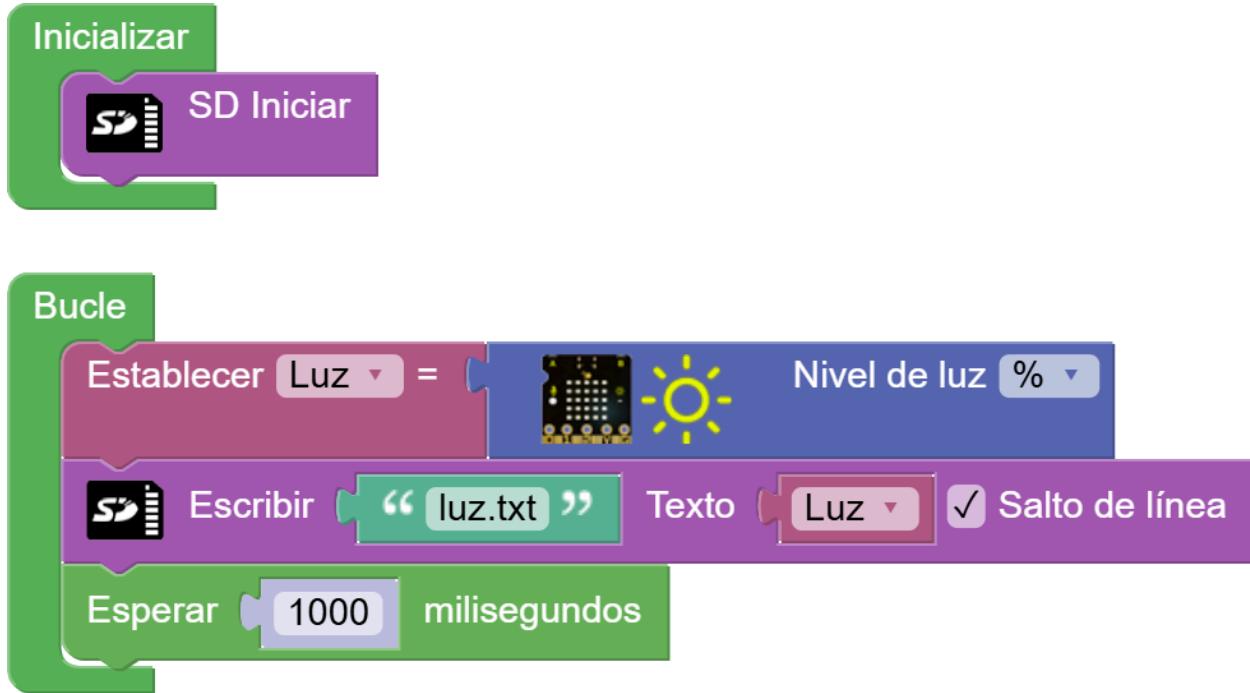
The screenshot shows the arduinoblocks interface. On the left, there is a sidebar with categories: Lógica, Control, Matemáticas, Texto, Variables, Listas, Funciones, micro:STEAMakers (expanded), Sensores, Actuadores, NeoMatrix, Tarjeta SD (selected and highlighted in purple), and ESP. At the top, there are tabs for Bloques, Información, Archivos, and Proyectos. Below the sidebar, several blocks are listed under the 'Tarjeta SD' category:

- SD Iniciar**
- Escribir**: A block with "SD" icon, "log.txt" string, "Texto" block, and a checked "Salto de línea" checkbox.
- Tamaño de archivo**: A block with "SD" icon and "log.txt" string.
- Eliminar archivo**: A block with "SD" icon and "log.txt" string.
- Escribir**: A block with "SD" icon, "log.txt" string, "Byte" block, and value "0".
- Ler byte**: A block with "SD" icon, "log.txt" string, and "Posición" block with value "0".

Ahora vamos a hacer una serie de actividades básicas de la **tarjeta SD** y la **placa micro:STEAMakers**. En estas actividades usaremos los bloques básicos que nos permitirán almacenar y leer datos de la SD.

En la primera actividad vamos a realizar un programa que nos permita almacenar los valores del sensor de luz ambiental en la tarjeta SD.

Programa:

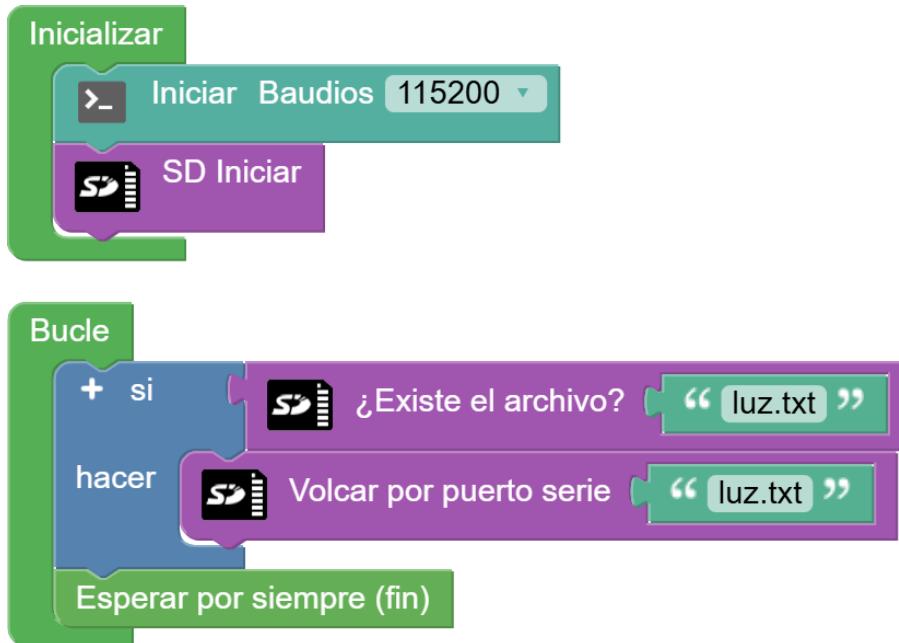


Una vez subido el programa la placa empieza a subir los valores en la tarjeta SD. Estos valores los podemos ver una vez que abramos la tarjeta en nuestro ordenador y en el archivo con el nombre que le hemos puesto en el programa.

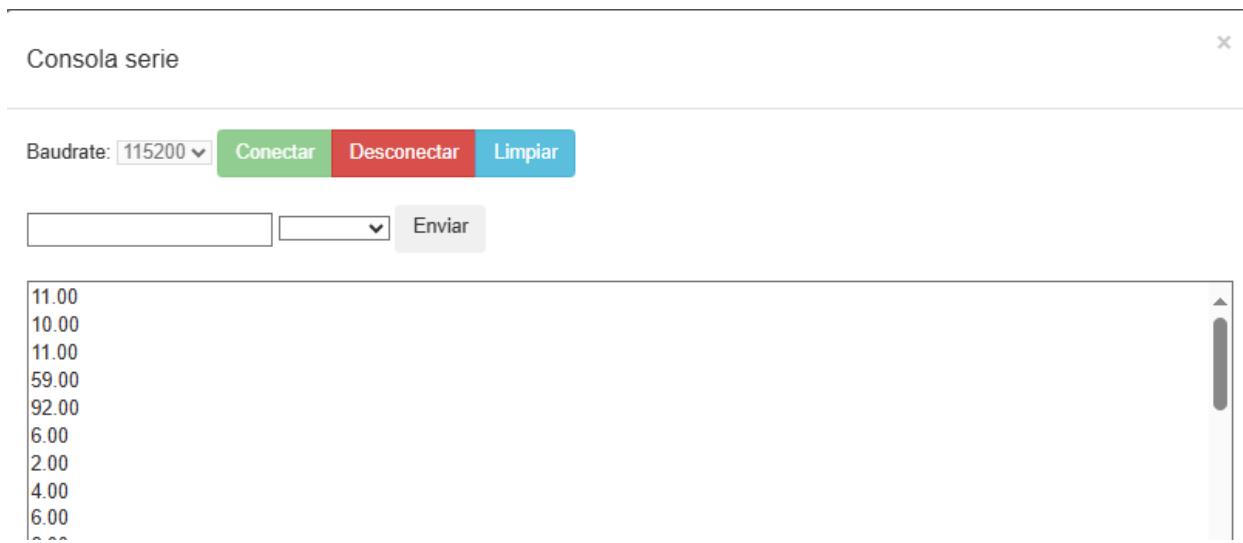
Actividad 11.2. Leer valores de tarjeta SD en arduinoblocks

Ahora vamos a leer los datos que hemos almacenado en la actividad anterior. Vamos a realizar un programa que nos permita volcar todos los datos por el puerto serie.

Programa:



Los datos obtenidos por la consola son los que hay almacenados en el fichero de la tarjeta SD.



Teoría: Comunicación ESP-NOW

ESP-NOW es un protocolo de comunicación inalámbrica desarrollado por **Espressif** para los microcontroladores **ESP32**, diseñado para operar con **baja potencia y baja latencia**. Utiliza la banda de **2.4 GHz** para transmitir datos directamente entre dispositivos ESP32, sin necesidad de una red Wi-Fi, lo que lo hace ideal para aplicaciones en tiempo real.

Características del Protocolo ESP-NOW

- **Baja Latencia:** ESP-NOW ofrece una transmisión de datos casi instantánea, con una latencia extremadamente baja, lo que es vital para aplicaciones en tiempo real.
- **Comunicación en Red:** Permite la interconexión de múltiples dispositivos, formando una red que mejora la cobertura y la robustez de la comunicación.
- **Eficiencia Energética:** Diseñado para ser eficiente en términos de consumo de energía, lo cual es crucial para dispositivos alimentados por baterías.
- **Facilidad de Implementación:** La simplicidad de ESP-NOW permite a los estudiantes concentrarse en el diseño y la funcionalidad de sus robots sin preocuparse por la complejidad de las redes de comunicación.
- **Interactividad en Tiempo Real:** La baja latencia es esencial para la robótica, donde la respuesta rápida a los comandos es fundamental. Esto permite que los robots reaccionen casi instantáneamente a las órdenes del controlador.
- **Proyectos Colaborativos:** Con ESP-NOW, múltiples robots pueden comunicarse entre sí de manera eficiente, posibilitando proyectos colaborativos y la implementación de sistemas robóticos más complejos y coordinados.
- **Portabilidad y Flexibilidad:** Al no depender de una infraestructura Wi-Fi, los proyectos pueden ser implementados en cualquier lugar, aumentando la flexibilidad y el alcance de las actividades educativas.

Para más información sobre ESP-NOW tenemos el siguiente enlace que nos dirige hacia un manual de **arduinoblocks**: [Manual ESP-NOW](#)

Actividad 12.1. Comunicación ESP-NOW

Para hacer las actividades con ESP-NOW necesitamos **2 placas ESP32 micros:steamakers o 1 placa ESP32 micro:STEAMakers y una ESP32 STEAMakers**. Debemos saber la dirección MAC de todas las placas que vamos a usar. Para ver la dirección MAC de la placa haremos el siguiente código.

Programa:



Una vez subido el programa en la placa podemos ver la dirección MAC en la consola serie.

Subimos el mismo programa en la segunda placa y anotamos la dirección MAC de ambas en algún lugar.

Ahora vamos a hacer el programa para ver cómo funciona la **ESP-NOW**. Para saber un poco de qué **bloques** hay en **arduinoblocks** tenemos el siguiente manual donde nos explica sobre todos los bloques de **ESP-NOW**: [Manual ESP-NOW](#)

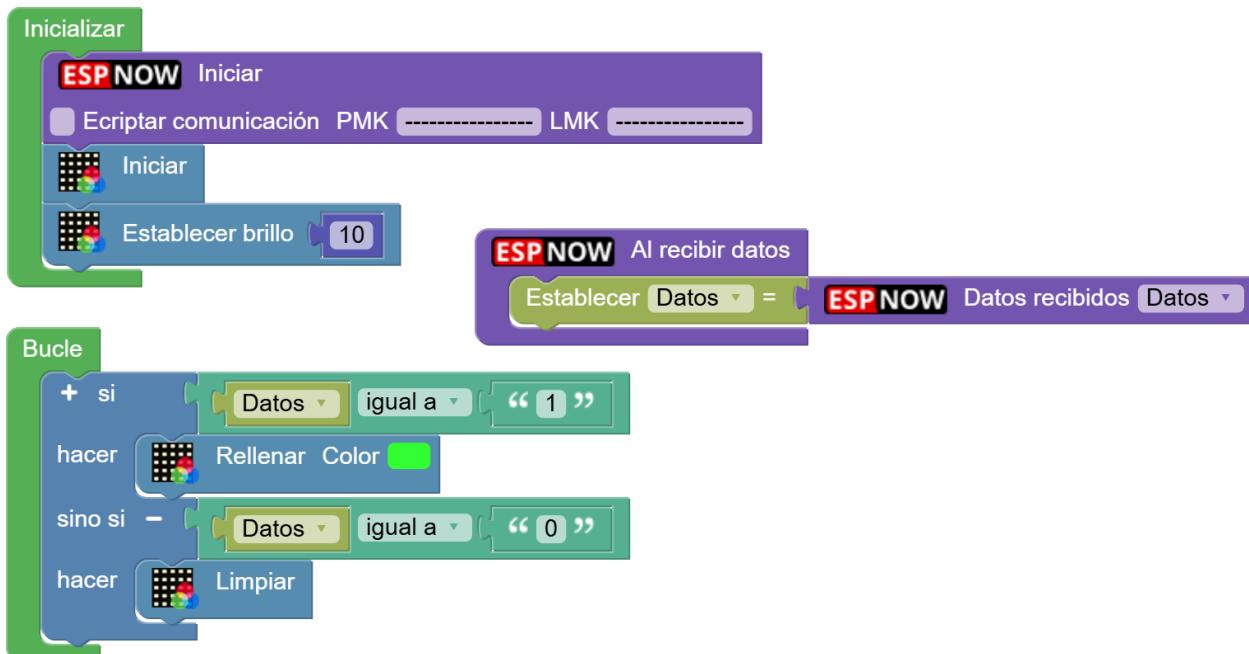
Una placa envía el valor de una variable “NeoMatrix” que cambiará con los botones A y B de la placa y la segunda placa se encenderá o no la NeoMatrix dependiendo de la variable.

Los bloques del ESP-NOW están en el apartado “Comunicaciones → ESP-NOW”.

Placa 1 (emisor): **34:98:7A:BC:A4:3C**



Placa 2 (receptor): **4C:11:AE:D3:99:54**



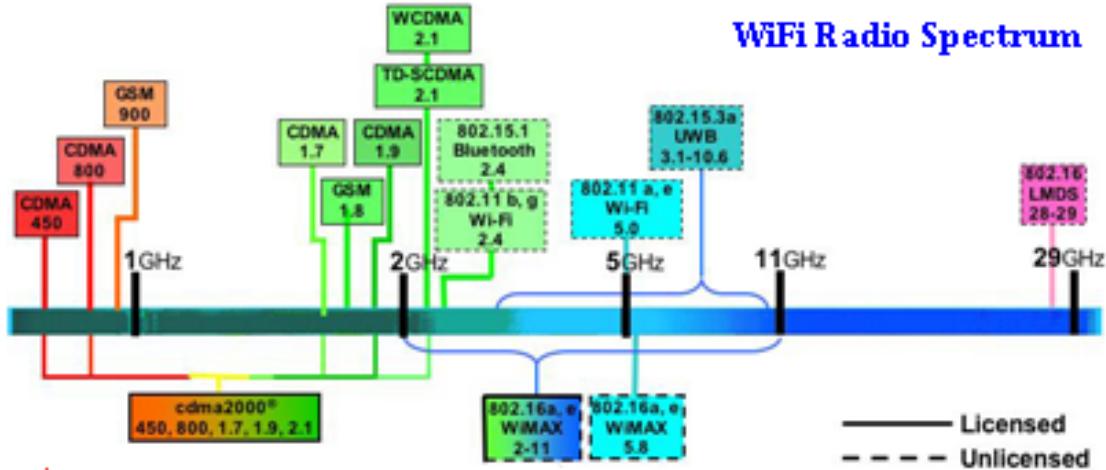
Teoría: Sistemas de comunicaciones: Bluetooth y Wifi

La placa **ESP32 micro:STEAMakers** dispone de conectividad Wifi y Bluetooth integrada en la propia placa, sin necesidad de ningún elemento externo.

Estas dos tecnologías las podemos resumir de la siguiente forma:

- La comunicación **Bluetooth** consiste en: es una tecnología (protocolo) de comunicación que permite la comunicación inalámbrica de corto alcance entre dispositivos digitales. El rango aproximado de **alcance** es de unos **10-20 metros**.
- La comunicación **Wifi** consiste en: es una tecnología (protocolo) de comunicación que permite la comunicación inalámbrica utilizando las ondas de radiofrecuencia. Su **alcance** es de unos **100-150 metros**. Nos permitirá realizar proyectos de IoT.

La comunicación **Wifi** se utiliza más para la conexión de dispositivos a **Internet (IoT)**, mientras que la comunicación **Bluetooth** se utiliza para conectar dispositivos entre sí. La comunicación **Wifi** también permite el intercambio de datos entre dispositivos que están conectados en una misma red.



A continuación, haremos una breve introducción a **IoT (Internet of Things)**. IoT en la interconexión de dispositivos y objetos a través de una red (privada o Internet) donde pueden interaccionar entre ellos. Cualquier elemento que se pueda imaginar podría ser conectado a Internet e interactuar, con o sin la necesidad de la intervención humana. El objetivo, por tanto, es una interacción de máquina a máquina, más conocido como interacción **M2M (Machine to Machine)** o dispositivos **M2M**.

Las aplicaciones de esta tecnología permiten mejorar la vida cotidiana de las personas, así como los entornos industriales donde esté implantado. Algunos ejemplos de aplicaciones son:

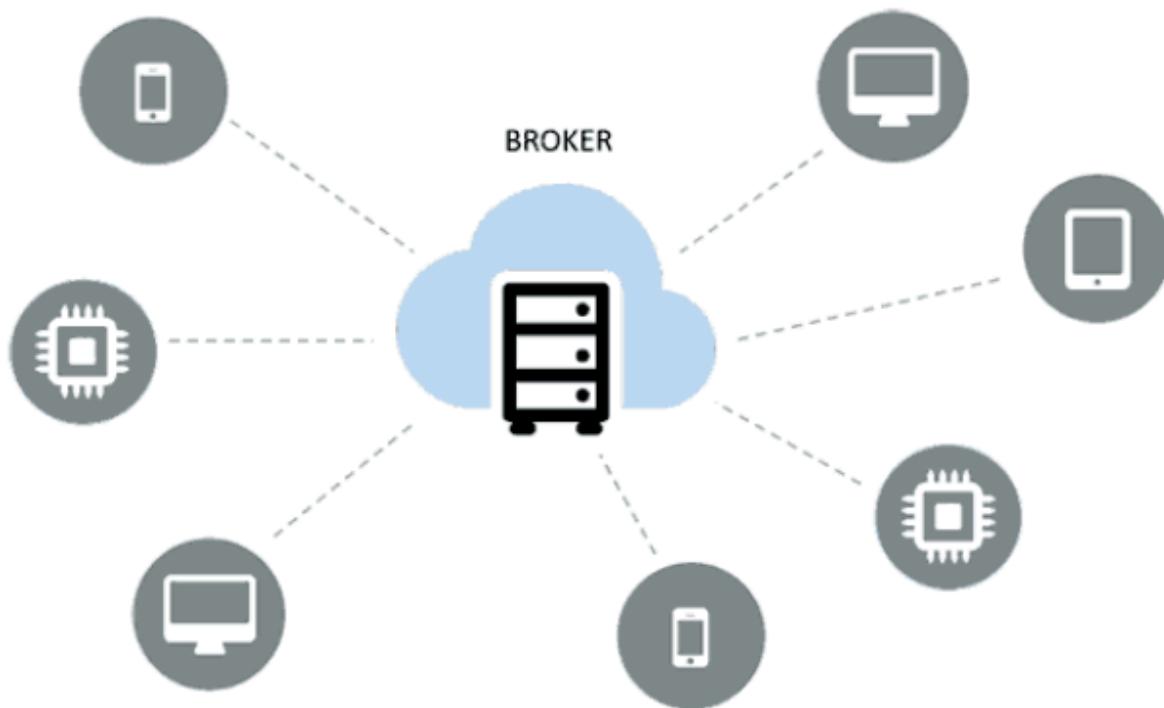
- Sistemas domóticos para hogares
- Conexión de electrodomésticos para su control remoto
- Conexión de objetos cotidianos
- Sistemas de alarma en aplicaciones industriales
- Control de elementos industriales
- Aplicación en Smart Cities y Smart Buildings
- Etc

Para los sistemas **IoT** se utilizan diferentes tecnologías y protocolos de comunicación. Últimamente han aparecido en el mercado gran cantidad de dispositivos con conectividad que son de pequeño tamaño, baratos y con un consumo energético bajo, como el **ESP8266** y el **ESP32**. Para la comunicación entre los diferentes dispositivos **IoT** se utilizan redes **Wifi**. Existen diferentes protocolos (normas definidas para que los dispositivos puedan comunicarse) para realizar esta comunicación **Wifi** entre dispositivos. Algunos de los protocolos **M2M** que existen son:

- **AMQP** (Advanced Message Queuing Protocol): es un protocolo PubSub de Message Queue. Está diseñado para asegurar la confiabilidad e interoperabilidad. Está pensado para aplicaciones corporativas, con mayor rendimiento y redes de baja latencia. No resulta tan adecuado para aplicaciones de IoT con dispositivos de bajos recursos.
- **CoAP** (Constrained Application Protocol): es un protocolo pensado para emplearse en dispositivos de IoT de baja capacidad. Emplea el modelo REST de HTTP con cabeceras reducidas, añadiendo soporte UDP, multicast, y mecanismos de seguridad adicionales.
- **MQTT** (MQ Telemetry Transport): es un protocolo PubSub de Message Service que actúa sobre TCP. Destaca por ser ligero, sencillo de implementar. Resulta apropiado para dispositivos de baja potencia como los que frecuentemente tenemos en IoT. Está optimizado para el routing activo de un gran número de clientes conectados de forma simultánea.
- **STOMP** (Streaming Text Oriented Messaging Protocol): es un protocolo sencillo que emplea HTTP y mensajes de texto para buscar el máximo de interoperabilidad.
- **WAMP** (Web Application Messaging Protocol): es un protocolo abierto que se ejecuta sobre WebSockets, y provee tanto aplicaciones de PubSub como rRPC.
- **WMQ** (Websphere MQ): es un protocolo de Message Queue desarrollado por IBM.

- XMPP (eXtensible Messaging and Presence Protocol) es un protocolo abierto basado en XML diseñado para aplicaciones de mensajería instantánea.

Para comunicar un número de dispositivos que están distribuidos en ubicaciones y redes desconocidas y que se comuniquen de forma fiable es externalizar la comunicación a un servicio de notificaciones centralizado. Podemos disponer de un servidor central que se encarga de recibir los mensajes de todos los dispositivos emisores y distribuirlos a los receptores. De forma genérica llamaremos a este servidor 'Router' o 'Broker'.



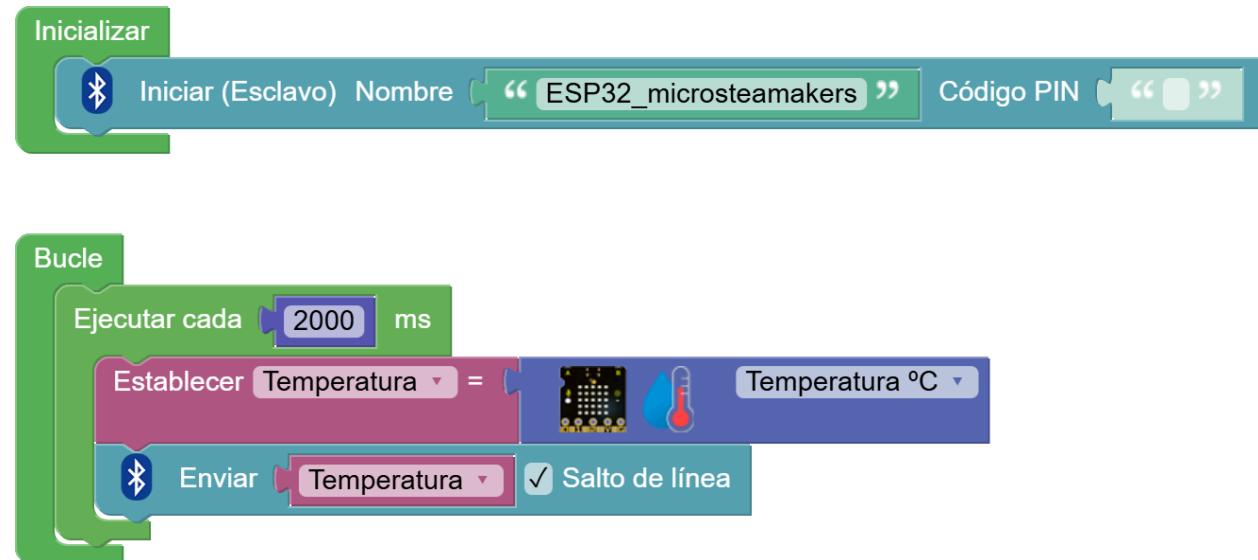
Este servidor, que tiene una dirección fija (o equivalente a un dominio), de forma que es accesible por todos los dispositivos. Así resolvemos el problema de tener que encontrar al otro dispositivo. El servidor mantiene un registro de los dispositivos conectados, recibe los mensajes y los distribuye al resto dispositivos, filtrando los destinatarios según algún criterio. Los dispositivos en ningún momento 'ven' o dependen del resto de dispositivos. Por tanto, esta infraestructura nos proporciona la escalabilidad.

Actividad 13.1. Comunicación Bluetooth

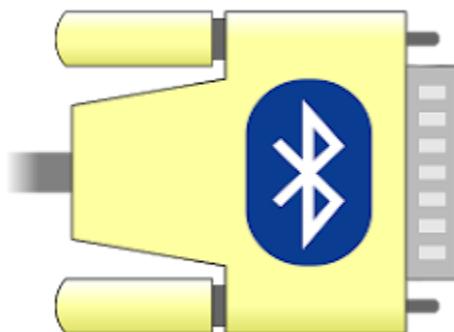
En esta actividad vamos a necesitar la placa **ESP32 micro:STEAMakers** y un móvil con la aplicación que vamos a indicar después para poder ver como funciona Bluetooth. Ahora vamos a realizar un programa con **arduino blocks** en el cual la placa ESP32 micro:STEAMakers va a enviar la temperatura ambiental cada 2 segundos al móvil.

Los bloques para programar bluetooth en **arduino blocks** los tenemos en “Comunicaciones → Bluetooth”

Programa:

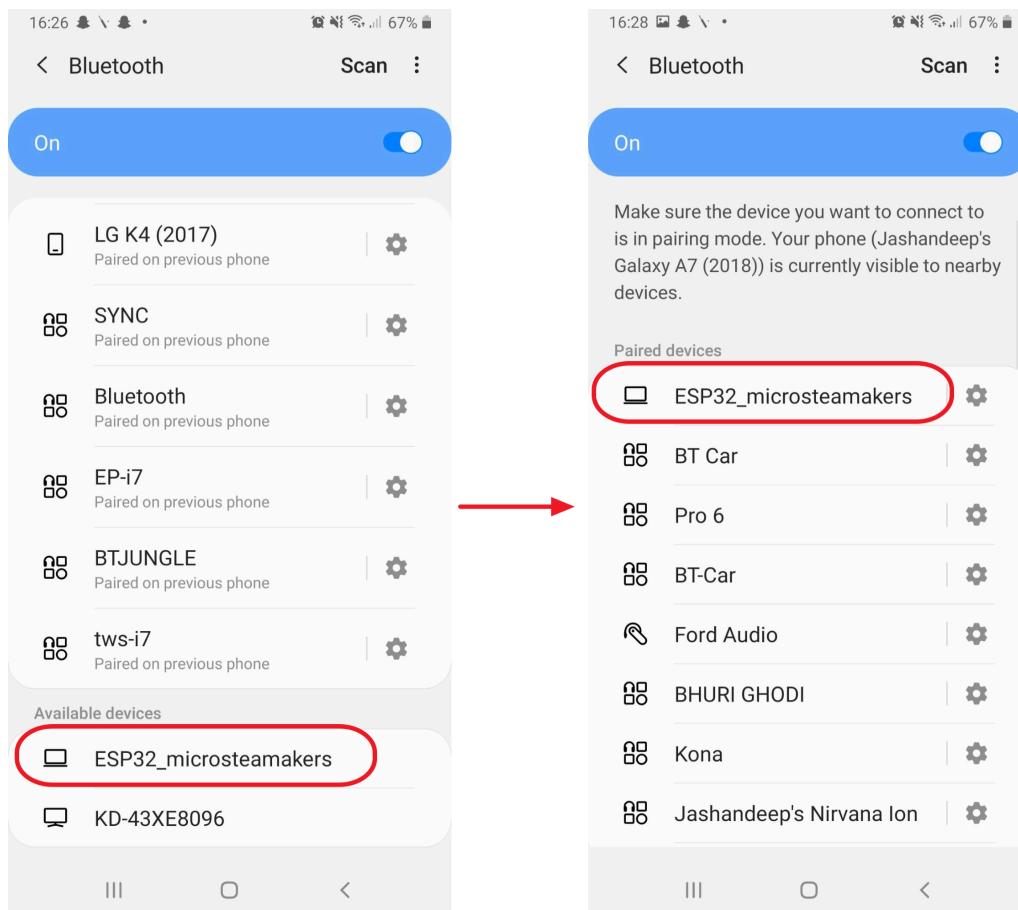


Una vez subido el programa a la placa tenemos que instalar alguna aplicación que pueda leer los datos de bluetooth en el móvil, nosotros recomendamos la siguiente aplicación: **Serial Bluetooth Terminal (Sólo para android)**

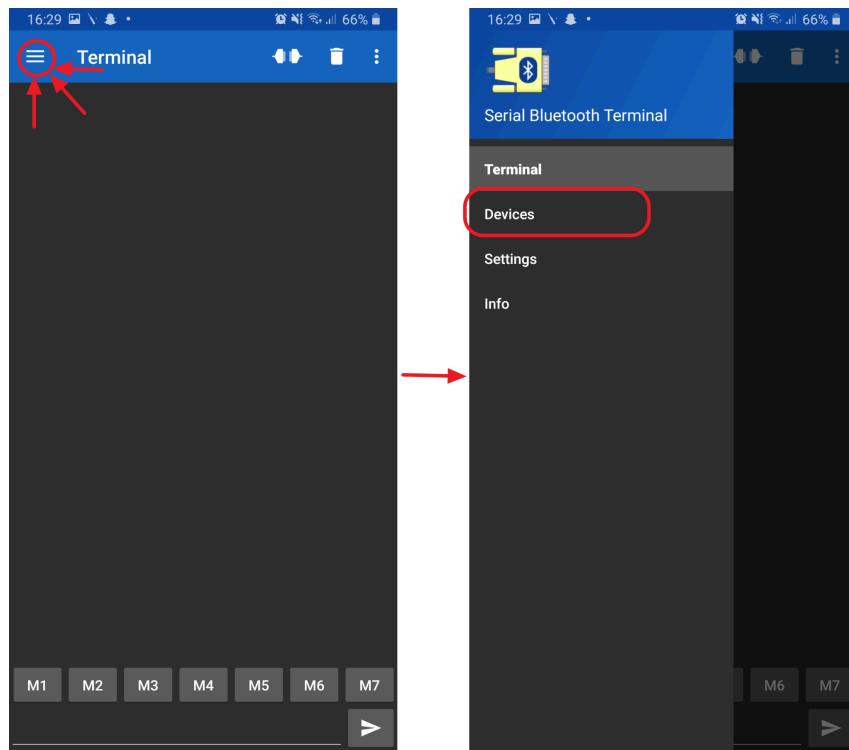


Ahora vamos a seguir los siguientes pasos para poder conectar la placa **ESP32 micro:STEAMakers** con el móvil y ver los datos.

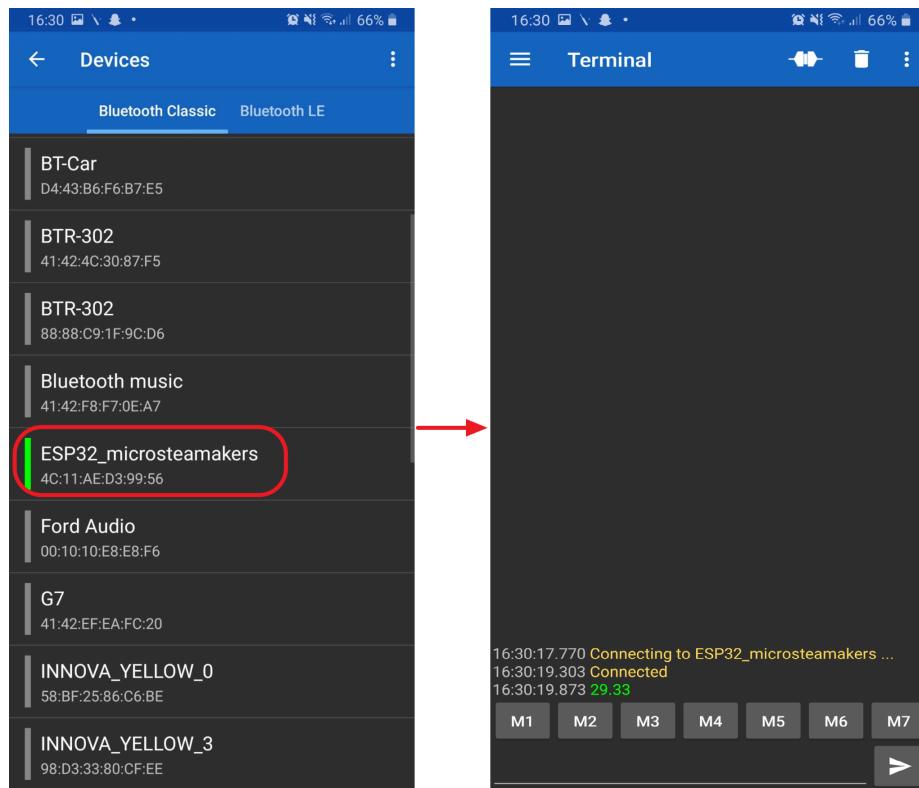
- 1) Activamos el Bluetooth en el móvil y buscamos un nuevo dispositivo. En la lista de nuevos dispositivos aparecerá **ESP32_microsteamakers**, este es el nombre que le hemos asignado.



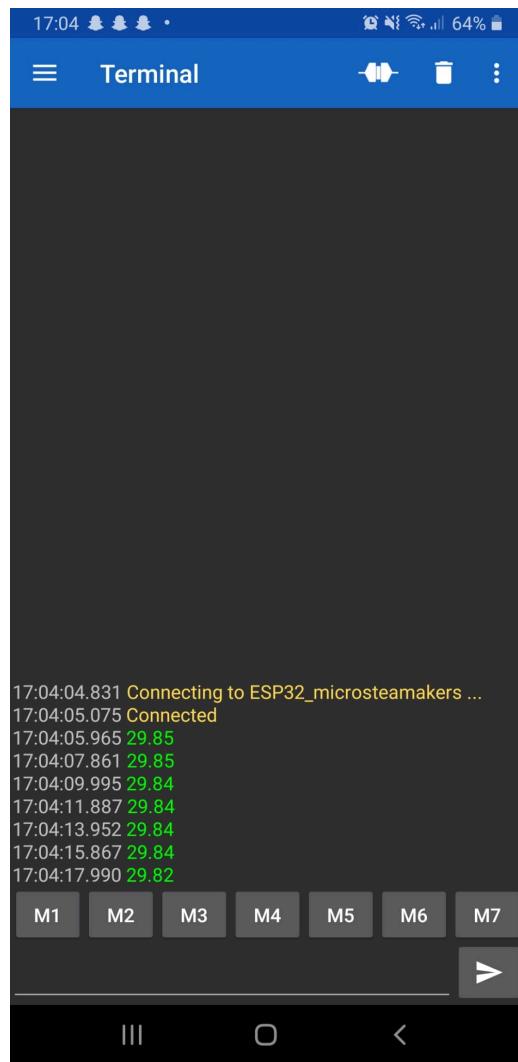
2) Ahora como podemos ver el móvil no está conectado con la placa, ahora vamos a abrir la aplicación que hemos instalado antes para poder ver lo que recibimos desde la placa. Después clicamos en las tres rayas y en "Devices".



3) Seleccionamos el nombre de la placa: **ESP32_microteamakers**



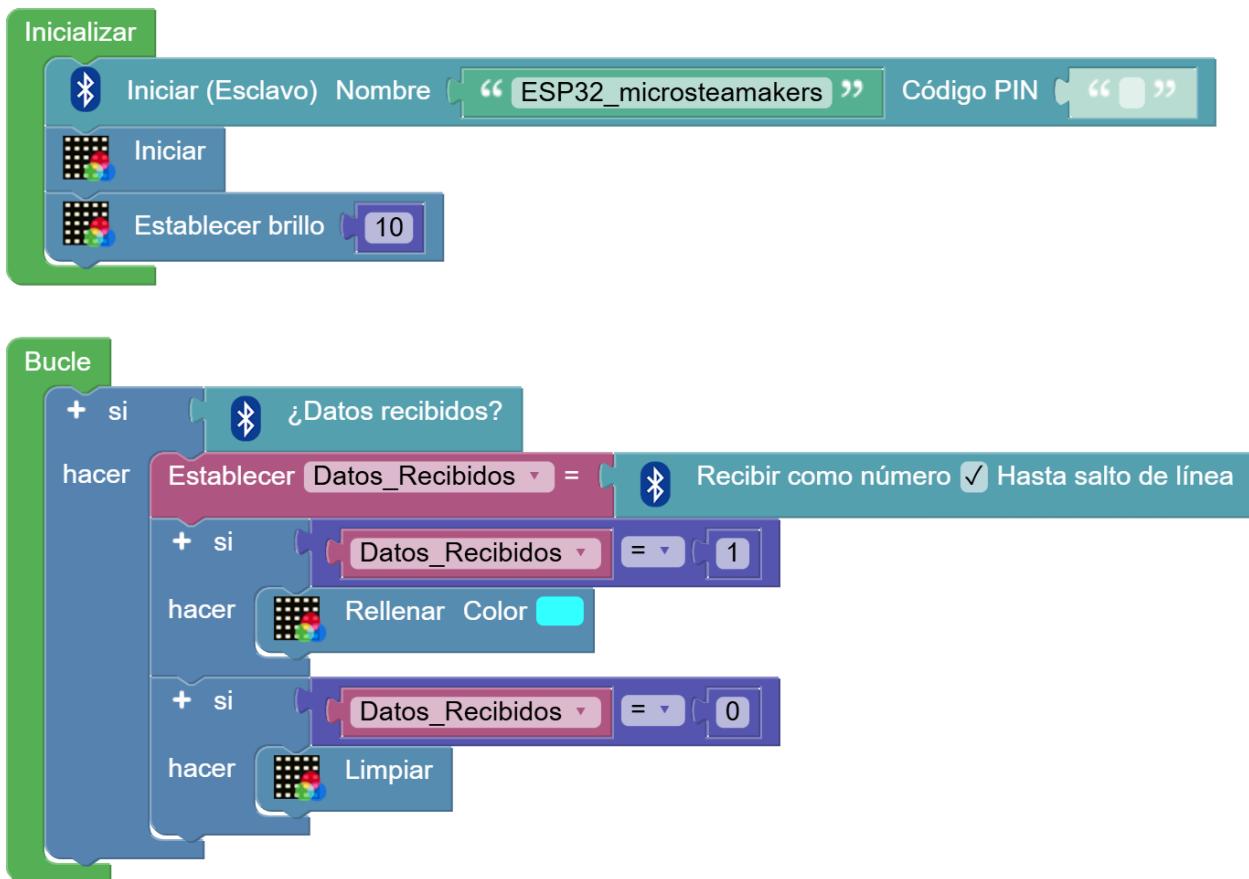
Una vez hecho los pasos ya podemos ver la temperatura que está enviando la placa **ESP32 micro:STEAMakers** al móvil a través del *Bluetooth*.



Actividad 13.2. Comunicación Bluetooth 2

En esta actividad también vamos a usar el **Bluetooth** pero en este caso el móvil va a enviar datos a la placa **ESP32 micro:STEAMakers** y la placa va actuar depende de lo que envía el móvil. Si desde el móvil envíamos “1” la placa encenderá la NeoMatrix y si envíamos “0” la placa apagará la NeoMatrix.

Programa:



Una vez subido el programa abrimos la aplicación en el móvil y veremos el funcionamiento.

Actividad 14.1. Comunicación WiFi

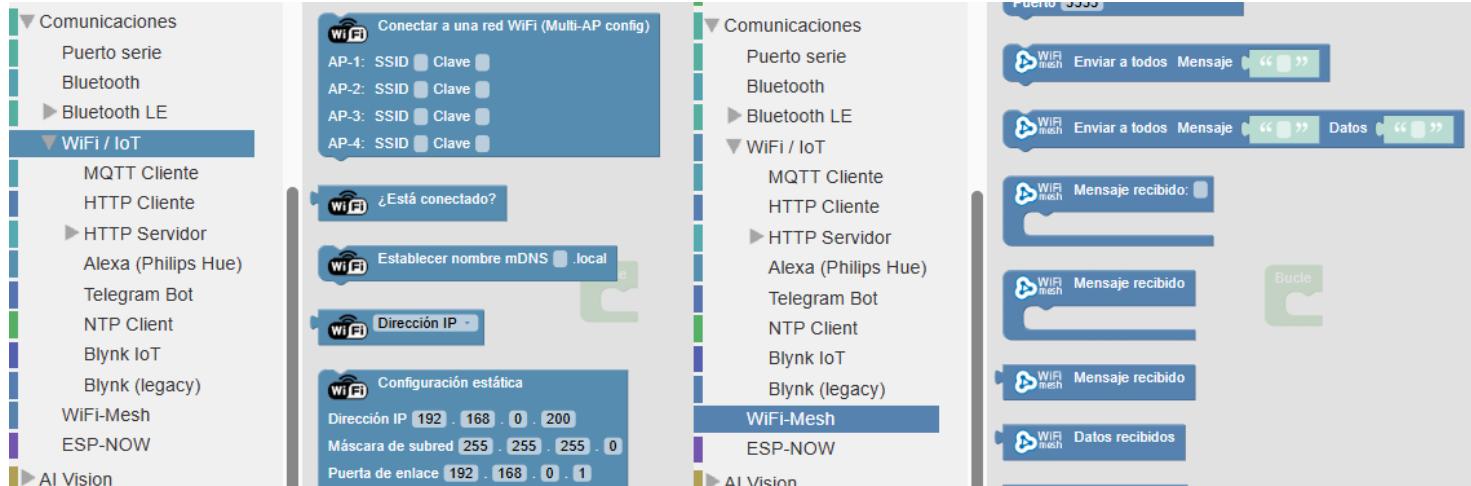
Arduinoblocks dispone de un gran número de bloques dedicados a la comunicación WiFi que nos permiten desarrollar un gran número de aplicaciones.

El propio microcontrolador **ESP32-WROOM-32** implementa el hardware y software compatible con redes **WiFi 802.11 b/g/n 2.4GHz** (soporta WFA/WPA/WPA2/WAPI).

Hay dos categorías de bloques:

- WiFi/IoT
 - MQTT Cliente
 - HTTP Cliente
 - HTTP Servidor
 - Alexa (Philips Hue)
 - Telegram Bot
 - NTP Client
 - Blynk IoT
 - Blynk (legacy)
- WiFi-Mesh

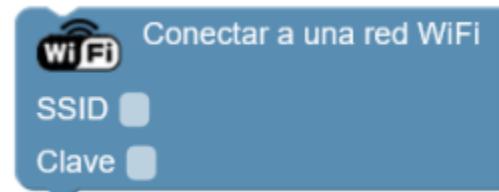
Todos los bloques en **arduinoblocks** están en el apartado “**Comunicaciones**”.



Tenemos diferentes bloques para poder trabajar con redes WiFi:

- Conectarse a una red WiFi como cliente (modo cliente/estación): esta opción es la más habitual. Con este bloque conectamos la placa **ESP32 micros:steamakers** a una red WiFi existente. Para ello necesitamos el nombre de la red WiFi (SSID) y la clave (WEP, WPA, etc.).

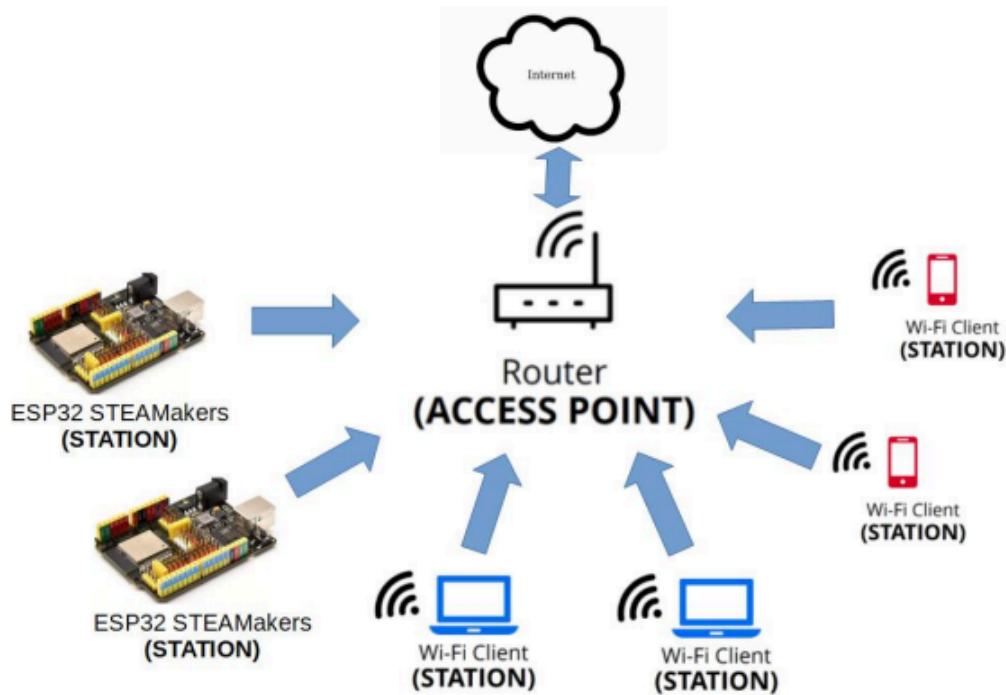
Por defecto, una vez conectados a la red obtendremos la configuración de red (IP, puerta de enlace, etc.) de forma automática (suponiendo que hay un router que lo hace de forma correcta en nuestra red). Si todo es correcto, tendremos acceso a nuestra red WLAN y conexión a internet a través del router.



También tenemos otro bloque de “Conectar a una red WiFi (Multi-AP config)” que básicamente hace lo mismo que el bloque de conectar a una red WiFi pero en este caso podemos guardar más de una configuración de WiFi en la memoria de la placa.

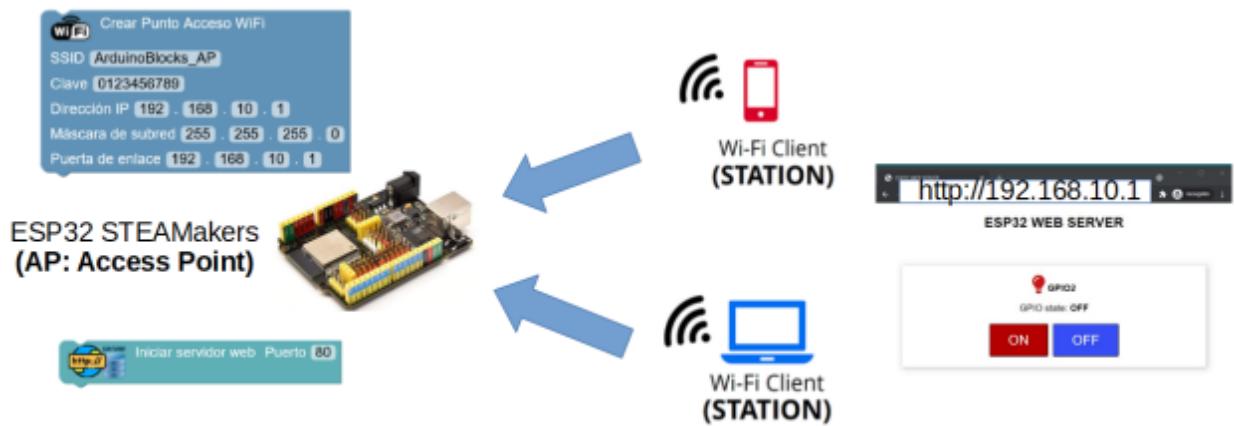


Este es el caso más simple (y el más habitual). Después de este bloque podemos usar cualquiera de los servicios explicados más adelante (MQTT, Blynk, Telegram, etc.).

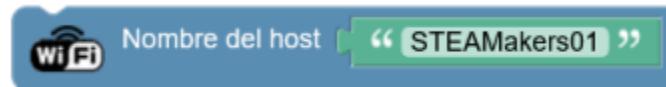


- Crear una red Wifi propia (modo punto de acceso): en algún caso, si no tenemos una red Wifi donde conectarnos, el **ESP32 micro:STEAMakers** es capaz de crear su propia red Wifi. Significa que crea un punto de acceso donde nos podemos conectar con otros dispositivos de red (portátil, Tablet, móvil, otros **ESP32 micro:STEAMakers**, etc.).

Hay que tener en cuenta que al crear un punto de acceso Wifi de esta forma, podemos conectar, por ejemplo, nuestro móvil al nuevo Wifi creado, pero no tendrá conexión a internet con esta conexión, simplemente es una forma de enlazar de forma inalámbrica a dos (o más) dispositivos para intercambiar información.

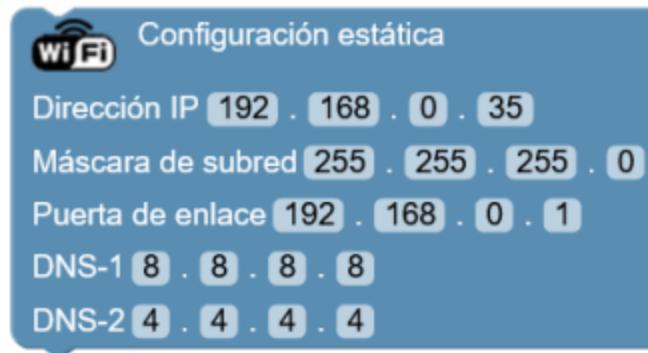


- Cambiar el nombre del dispositivo en la red (Nombre del host - hostname): tenemos un bloque que nos permite cambiar el hostname del dispositivo para la red:



- Dirección IP estática (en vez de usar la configuración automática por defecto): en algunos casos nos puede interesar que nuestro dispositivo tenga una IP fija dentro de la red y no depender del router (servidor DHCP) para que nos asigne cada vez una IP distinta en cada conexión a la red.

Por ejemplo, en casos en los que nuestro dispositivo vaya a ser un servidor web y necesitemos conectarnos a él desde otros dispositivos, o para redes donde el servicio DHCP de IP automática no está activado. Para ello usamos el bloque de configuración de IP estática que permite asignarle al dispositivo una IP y configuración básica de red de forma manual:

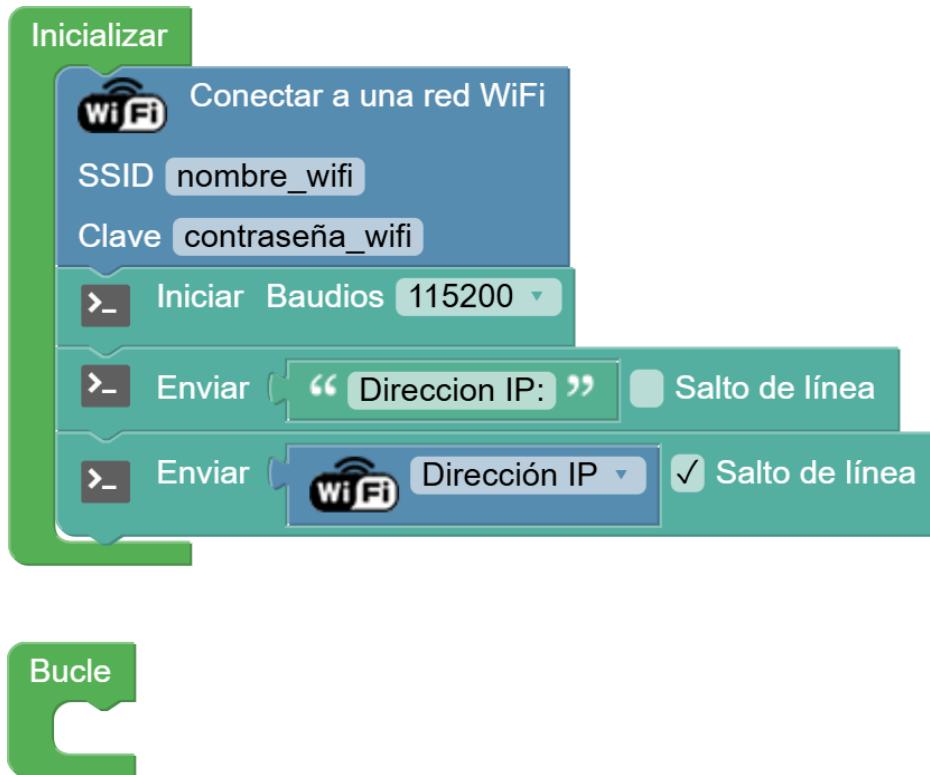


- **Dirección IP:** La dirección que queremos dentro de la red a la que nos vamos a conectar. En un router doméstico uno de los rangos de IP más habituales es 192.168.0.X o 192.168.1.X, siendo la dirección 192.168.0.1 la del router por defecto, y el rango DHCP (las IPs de los dispositivos a los que se les asigna automáticamente) suele empezar en la 192.168.0.100 en adelante (101, 102, ...). Por eso, es recomendable en primer lugar saber si hay otros dispositivos con IP fija en la red y hacer un listado de dispositivos e IPs para evitar duplicados de IPs (error en la red). Y, por tanto, las IPs fijas por seguridad deberían estar fuera del rango del DHCP (menor que la 100). Recomendación en este ejemplo: poner una IP entre la 2 y las 99 para no tener conflicto con la IP del router ni con el rango de IPs asignadas automáticamente a otros dispositivos con el DHCP.
- **Máscara de subred:** Es la máscara que nos separa la parte de la red y los dispositivos dentro de la dirección IP. En el ejemplo he dejado la máscara de 24 bits de red y 8 bits de dispositivos por defecto en la mayoría de redes LAN domésticas.
- **Puerta de enlace:** Es la IP a través la cual podemos acceder a otras redes (internet), en nuestro caso debemos poner la IP del router.
- **DNS:** Son dos direcciones IP de servidores DNS en internet para resolver nombres de dominios. Podríamos usar la dirección IP del router, aunque en el ejemplo he puesto las IPs públicas de los servidores DNS de Google.

Con todo lo visto hasta ahora podemos conectarnos a una red Wifi o crear nuestra propia red Wifi. Pero también podemos conectarnos a Blynk, MQTT, un servidor HTTP (web), etc. A continuación, vamos a trabajar con algunos de estos recursos.

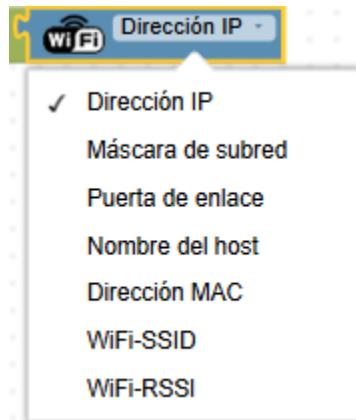
En primer programa vamos a conectar la placa **ESP32 micro:STEAMakers** con la red WiFi y comprobar si se ha realizado correctamente la conexión enviando la dirección IP asignada a la consola.

Programa:



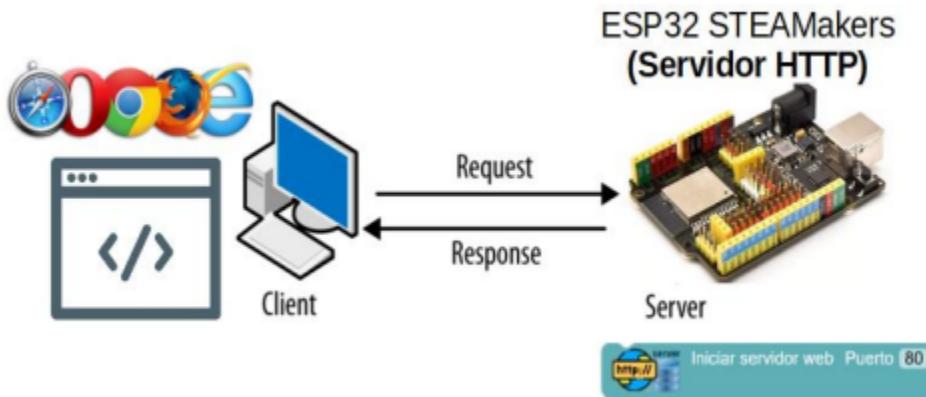
Si podemos observar la dirección IP en la consola significa que se ha establecido la comunicación.

También podemos ver otra serie de datos de nuestra conexión, podemos observar los siguientes datos:

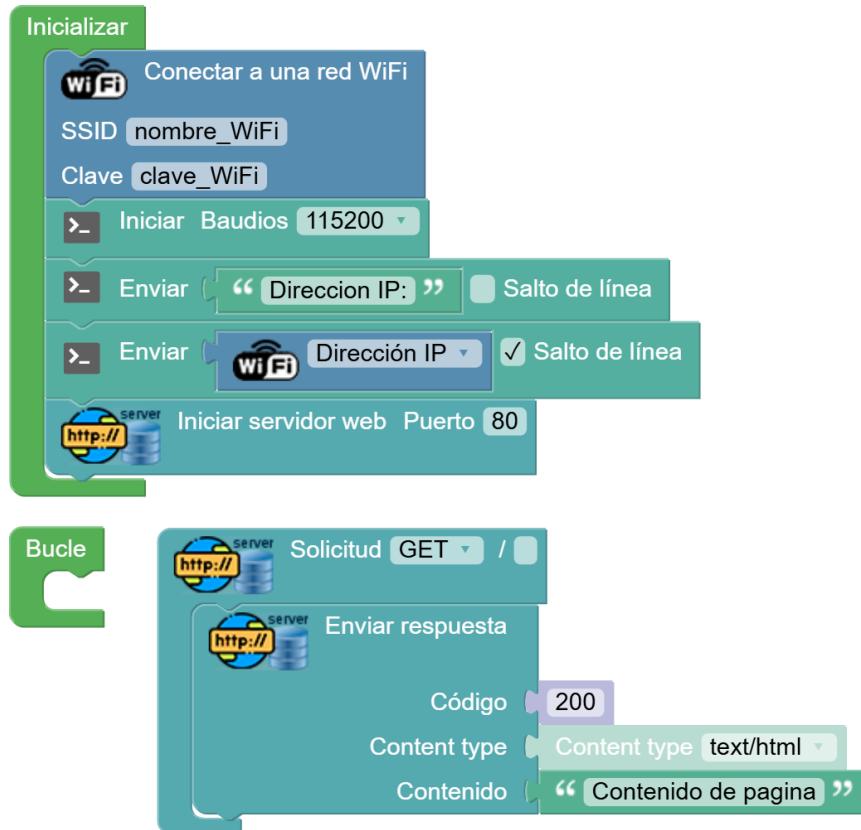


Actividad 14.2. WiFi: Servidor HTTP I

La placa **ESP32 micro:STEAMakers** permite implementar un servidor web de forma sencilla y potente. Un servidor web es un servicio de red que está esperando conexiones. Cuando un dispositivo se conecta le hará una petición en formato HTTP (Hiper Text Transfer Protocol) que normalmente será una página web (documento HTML) o un archivo de otro tipo: CSV, texto, imagen, etc. No siempre el servidor debe retornar datos útiles, el servidor web puede utilizarse como una interfaz para control remoto, permitiendo intercambiar datos y ejecutar acciones remotas.



Vamos a crear un servidor web como interfaz de control remoto.



En la Consola de **arduinoblocks** consultamos la IP asignada, en este caso es 192.168.18.252:

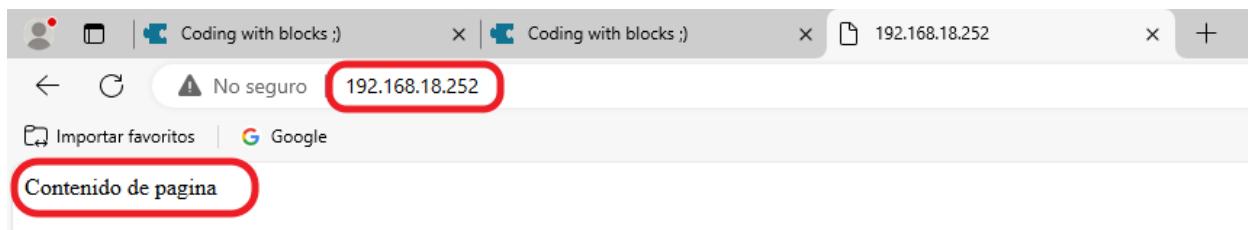
```

ets Jun 8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:10944
load:0x40080400,len:6388
entry 0x400806b4
Direccion IP: 192.168.18.252

```

En el explorador, indicamos la dirección IP y nos mostrará el contenido que lo hemos ajustado en **arduinoblocks**:



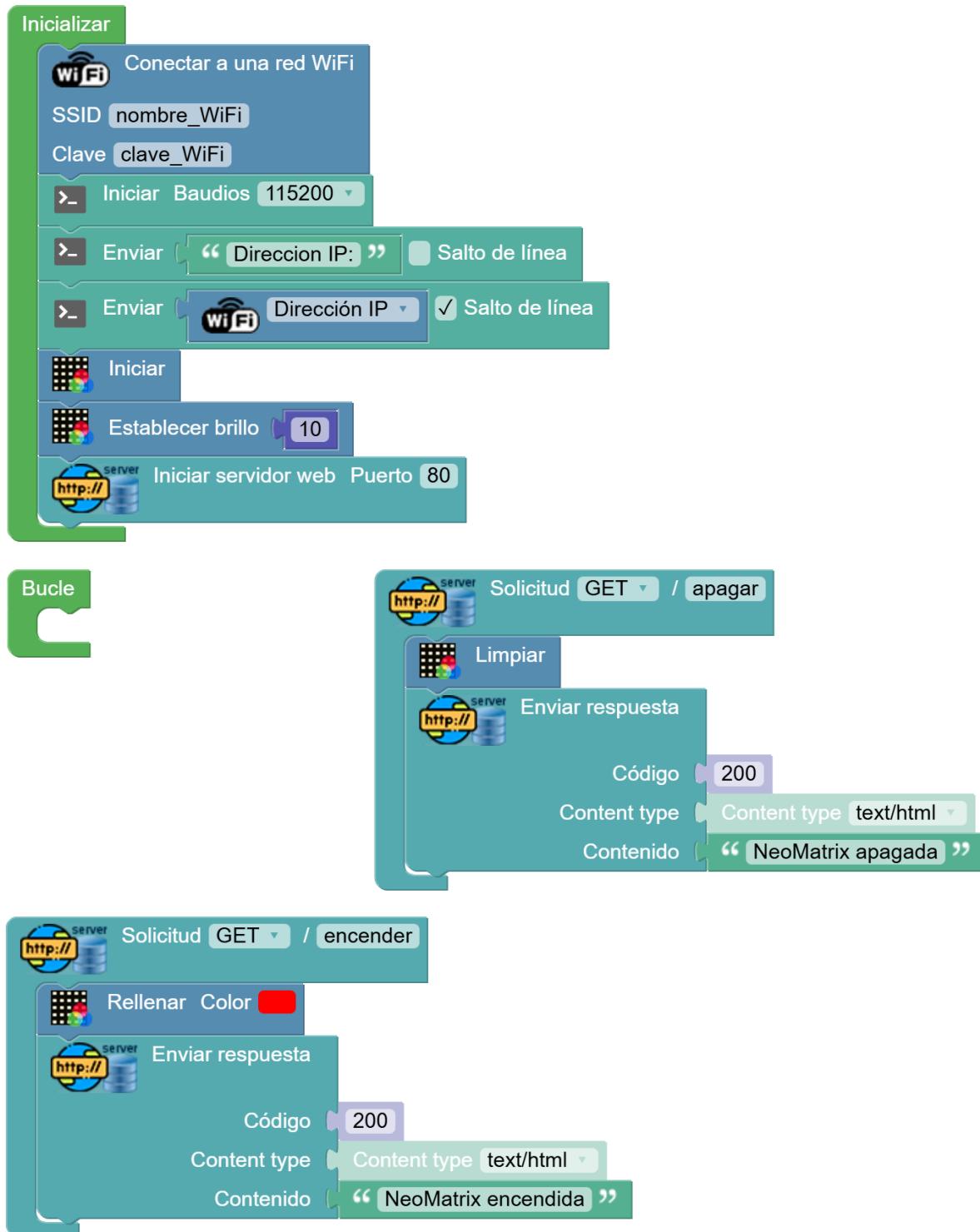
En este ejemplo, la respuesta se compone de 3 partes importantes:



- Código: 200 - Código de respuesta HTTP OK.
- Content-Type: tipo de contenido que se va a enviar, normalmente text/html, pero podría ser text/CSV, text/xml, etc.
- Contenido: el contenido tal cual que se envía, para hacerlo correcto debería ser texto formateado en HTML de forma correcta. Abusando de la capacidad de los navegadores actuales para procesar casi cualquier tipo de contenido “mal formateado” le enviamos un sencillo texto de saludo.

Actividad 14.3. WiFi: Servidor HTTP II

Vamos a realizar un programa para poder encender y apagar el led azul desde el navegador. Solamente cambiaremos el contenido de la Solicitud, pero, además, será crearemos dos solicitudes para poder realizar las dos acciones sobre el led.



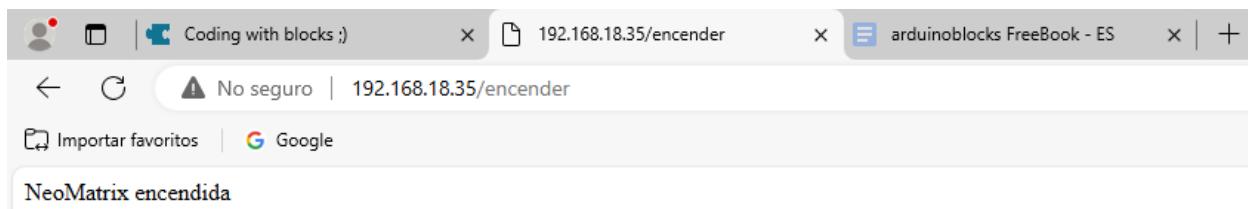
Las peticiones HTTP las haremos normalmente desde un navegador, poniendo la IP de nuestro dispositivo en la barra de direcciones con el protocolo http. Por ejemplo, si nuestro dispositivo tiene la dirección IP 192.168.18.35:

El formato de la URL que podemos solicitar a nuestra **ESP32 micro:STEAMakers** es:

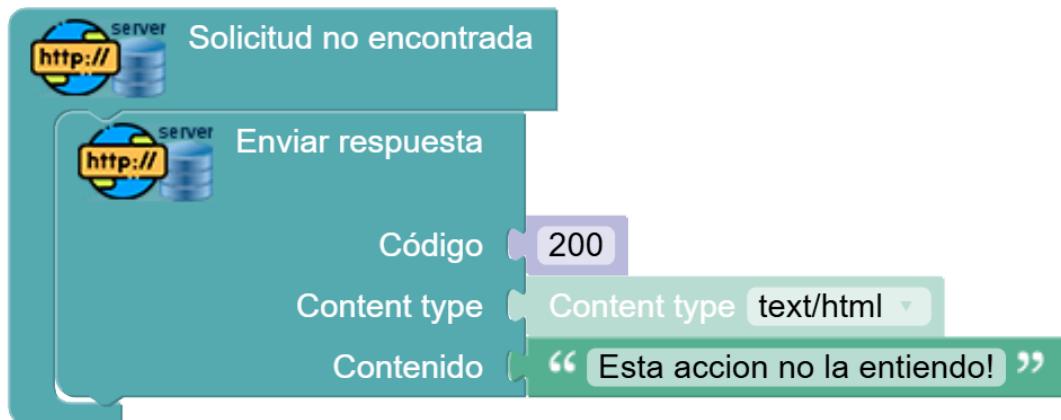
http://direcciónip/acción?parametros

- La **dirección IP** es la que tenga el dispositivo, como hemos visto anteriormente en este ejemplo en mi red doméstica la IP es 192.168.0.155 (variará en cada red, y según el DHCP asigne en cada momento o si la ponemos de forma fija)
- La **acción** (opcional) será la web o comando que queremos pedirle al servidor. En un servidor web real (este es real, me refiero a uno tradicional tipo Apache, etc...) sería el documento del servidor que estamos solicitando (ejemplo: index.html, index.php, guardardatos.jsp, etc.).
- Los **parámetros** (en este caso parámetros tipo GET) son datos adicionales que le damos al servidor normalmente como información extra a la acción solicitada. Los parámetros GET son opcionales, siempre van después del símbolo "?" y se forman con pares "clave=valor" separados entre ellos por "&"

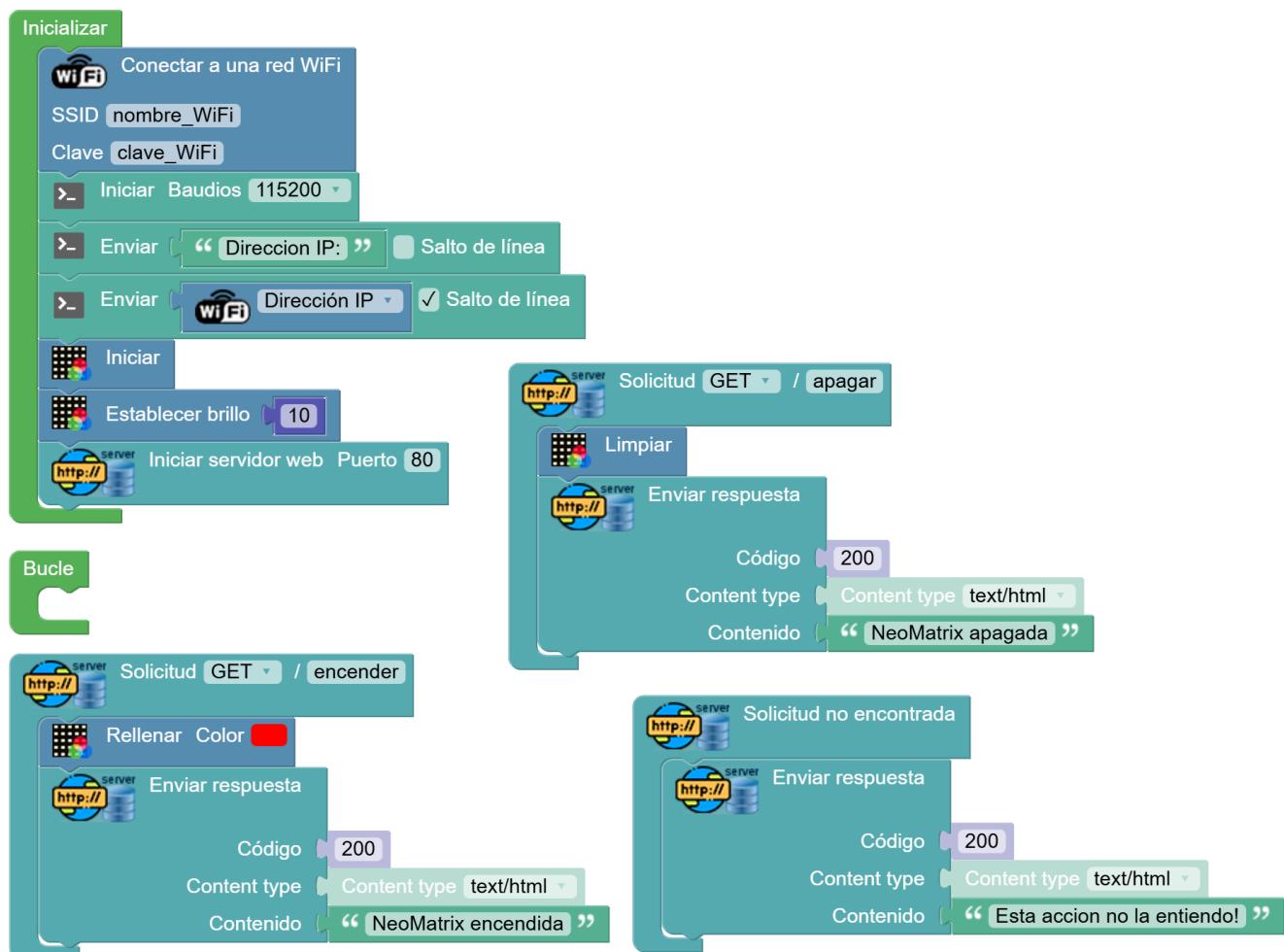
Desde el navegador encenderemos y apagaremos el led azul, utilizando los comandos que hemos creado en la Solicitud.



Por otro lado, deberíamos añadir otra respuesta para cuando se solicite una acción no válida. Por ejemplo, si pido la acción parpadear cuando sólo tengo implementadas la acción de encender y apagar debería indicar un error



Programa:



Actividad 14.4. WiFi: Servidor HTTP III

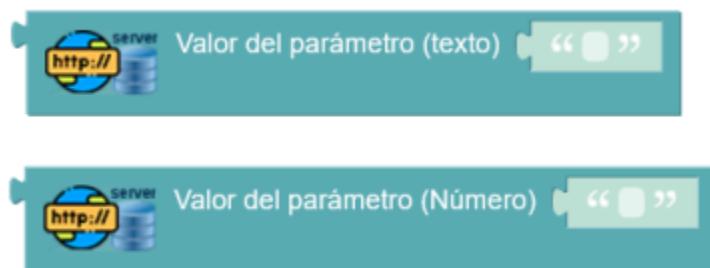
Basándonos en la idea del ejemplo anterior, vamos a hacer uso de los parámetros GET para permitir que nuestro servidor web sea capaz de enseñar el texto que queramos en la NeoMatrix.

Iniciar servidor web (80) y detectaremos que dirección IP tenemos asignada, igual que en la actividad anterior.



Realizaremos la función para encender y apagar NeoMatrix.

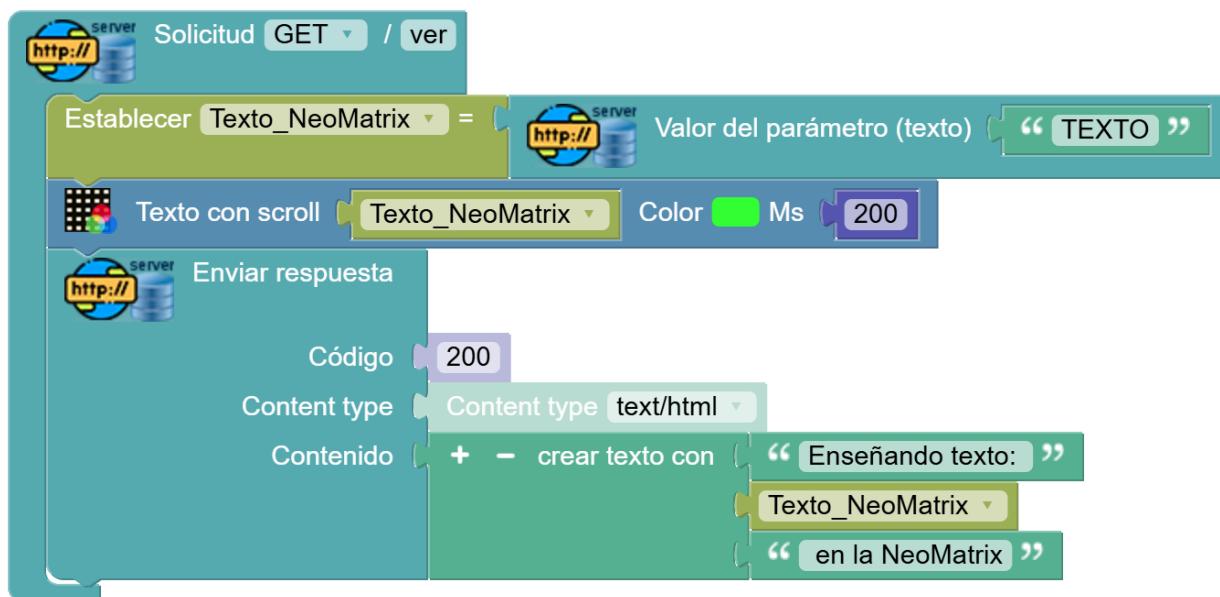
Con los bloques “valor del parámetro” obtenemos parámetros de la URL.



En el ejemplo usaremos la versión del bloque que obtiene el parámetro y lo procesa automáticamente como un valor texto.

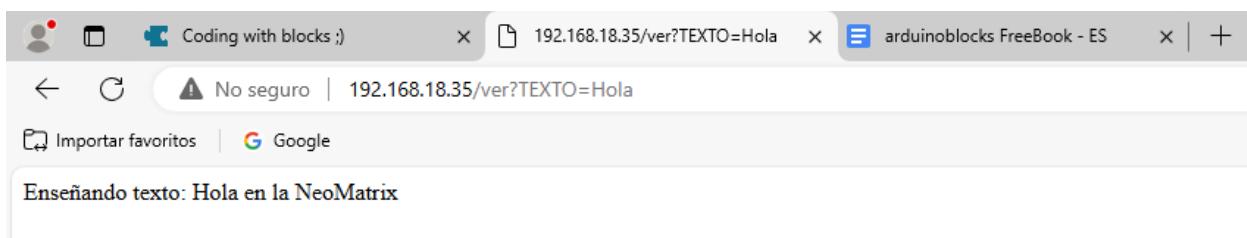


Vamos a realizar la función que nos deja mostrar el texto que queramos en la NeoMatrix de la **ESP32 micro:STEAMakers**.



La petición correcta para mostrar texto sería:

192.168.18.35/ver?TEXTO=Hola



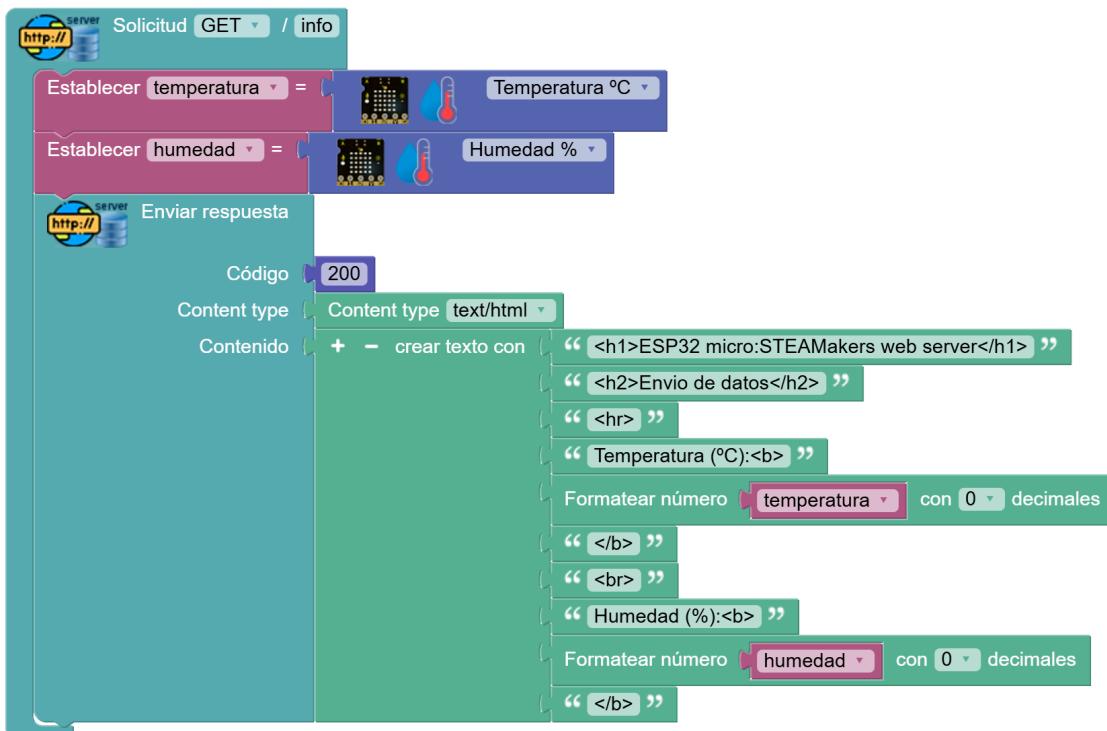
Así podríamos poner cualquier texto en la NeoMatrix.

Actividad 14.5. WiFi: Servidor HTTP IV

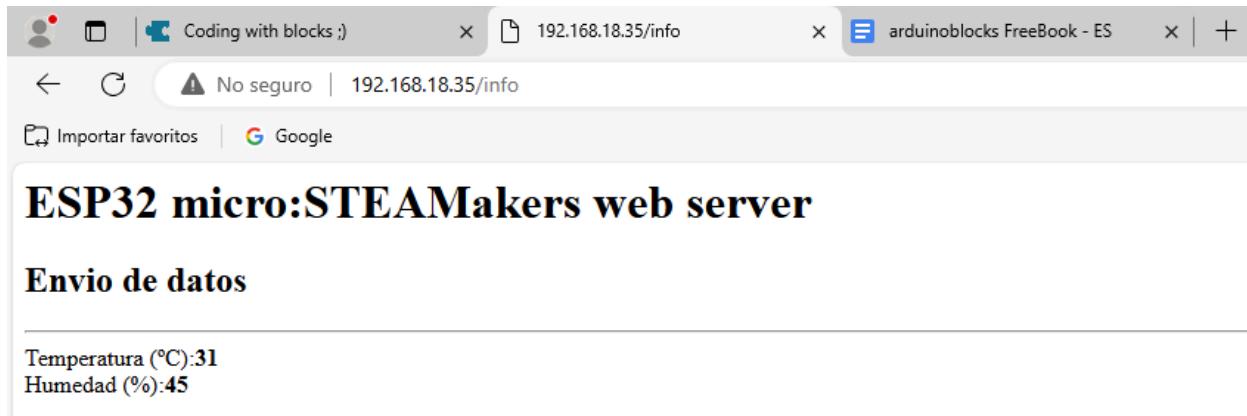
Obtener una web con la información de un sensor de temperatura y humedad:
El bucle de “Inicializar” sería el siguiente:



La acción info nos enviará la información de temperatura y humedad en una página web. Utilizaremos para formatear la web de una forma más vistosa se utilizan códigos HTML básicos como (h2, h3, hr, b), para información sobre códigos HTML básicos podemos consultar en sitios online personalizados (<https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/aprendehtml-tutorial-para-principiantes/>):



El resultado de la web generada por nuestra placa **ESP32 micro:STEAMakers** con el **sensor de temperatura y humedad** es el siguiente (cada vez que actualizamos la página se regenera con los datos actualizados).



Temperatura (°C): 31
Humedad (%): 45

Para dejarlo más vistoso (asumiendo que nuestra red tiene internet) podemos insertar en el código HTML una imagen con la URL completa de internet.

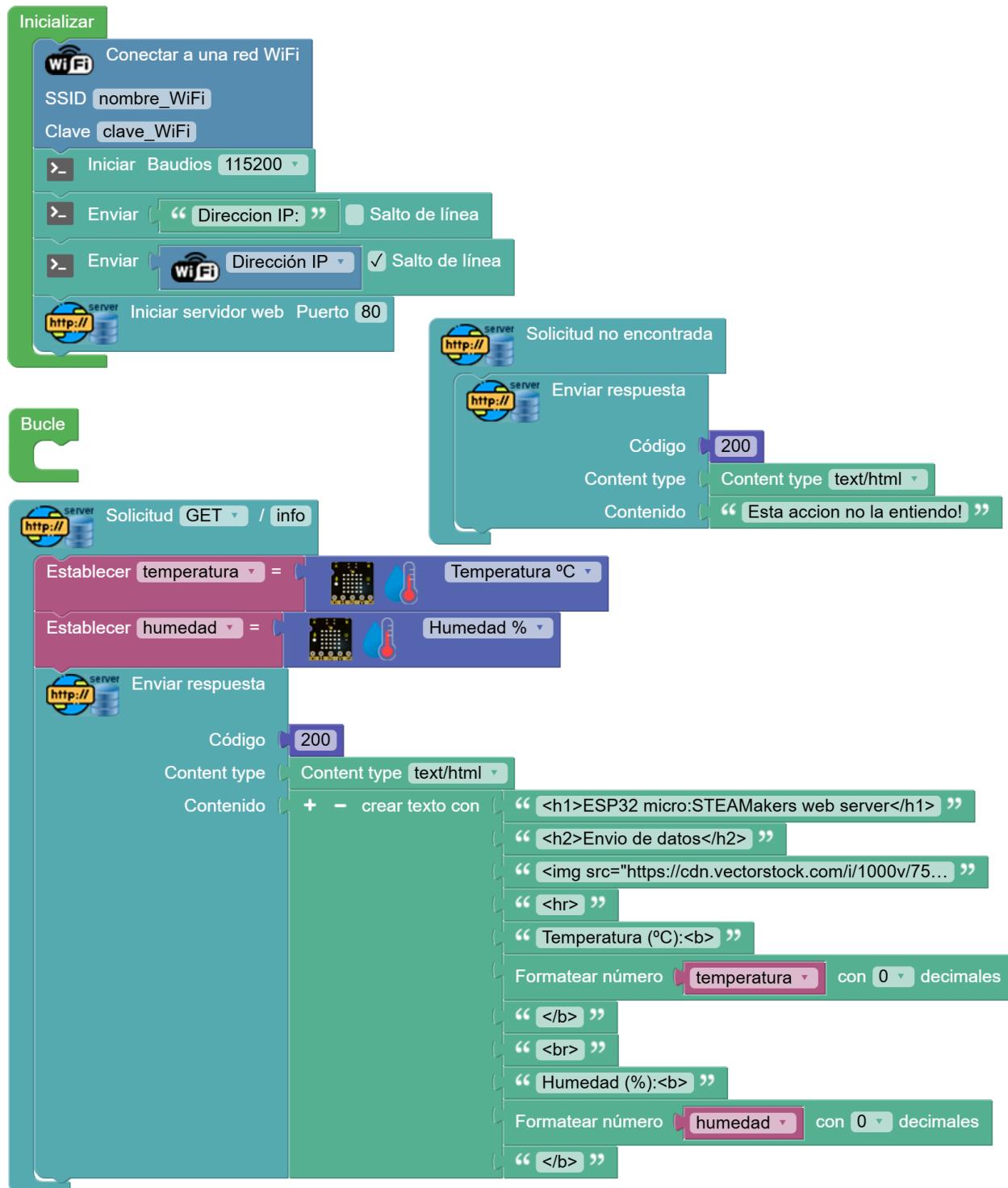
```
<img src='url_de_la_imagen_en_internet' width='ancho' height='alto' />
```

Buscamos un icono de temperatura y humedad y obtenemos su url:

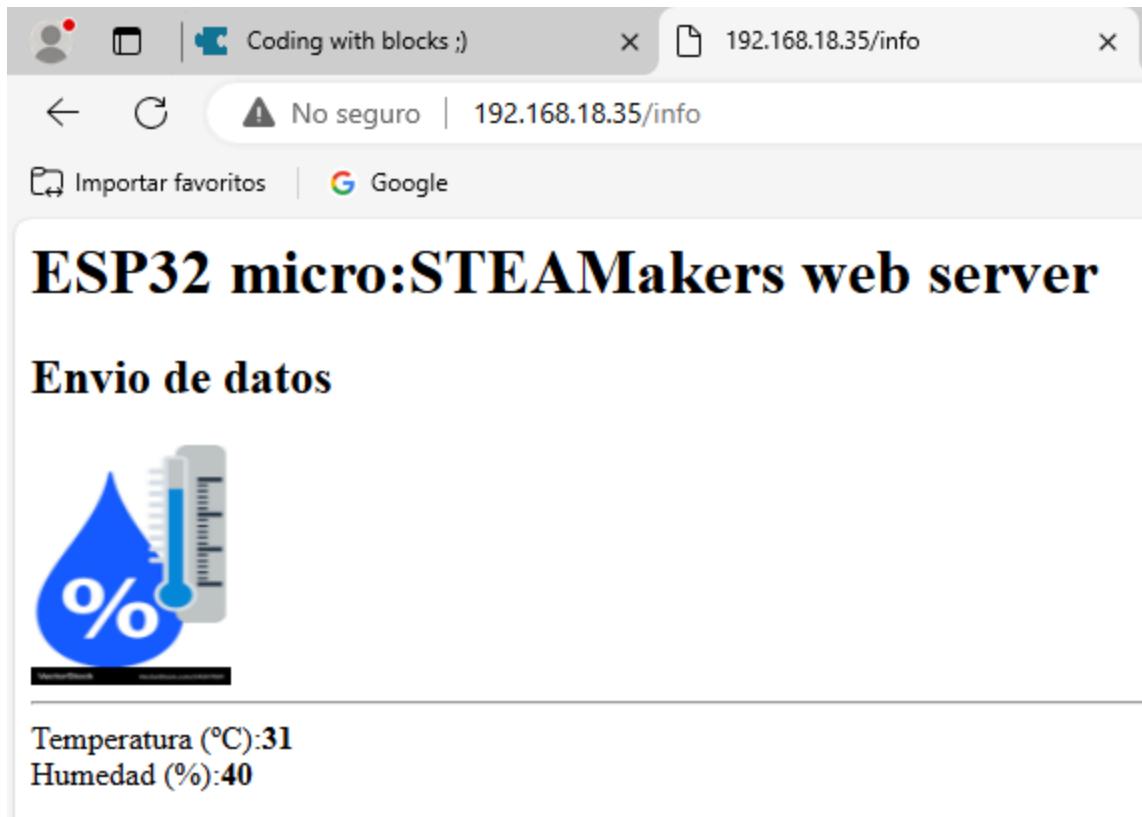
<https://cdn.vectorstock.com/i/1000v/75/91/humidity-icon-vector-24247591.jpg>

Y añadimos el código en el servidor web de la **ESP32 micro:STEAMakers**.

Programa:



Y nos muestra los siguientes datos:



Coding with blocks ;)

192.168.18.35/info

No seguro | 192.168.18.35/info

Importar favoritos | Google

ESP32 micro:STEAMakers web server

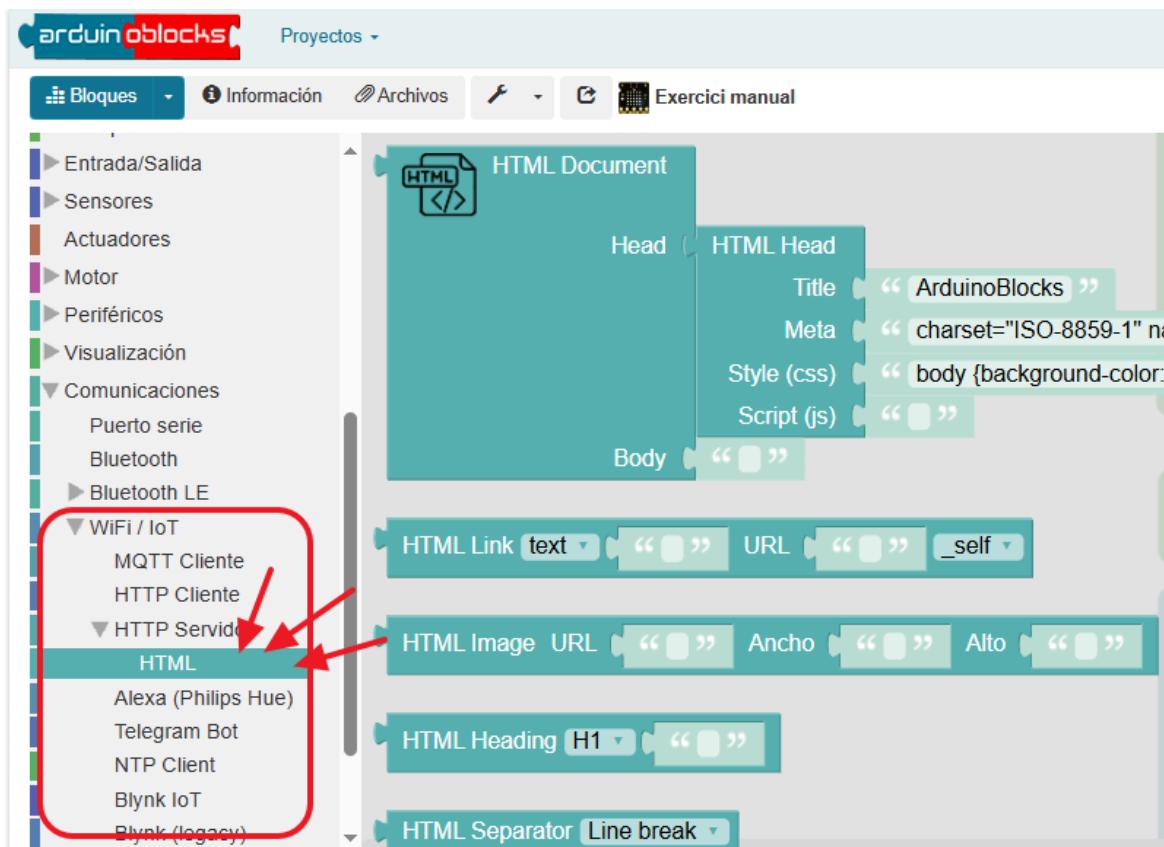
Envio de datos



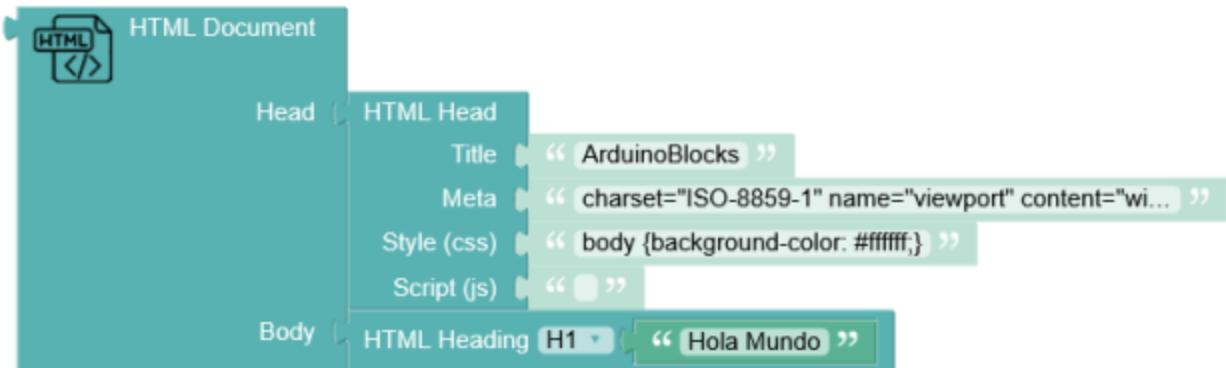
Temperatura (°C):31
Humedad (%):40

Actividad 14.6. Contenido HTML

Podemos generar contenido HTML con **arduino blocks**. Disponemos de una serie de bloques específicos que nos permiten generar texto con formato HTML especialmente pensado para devolver contenido en formato web desde el servidor HTTP de la placa **ESP32 micro:STEAMakers**.



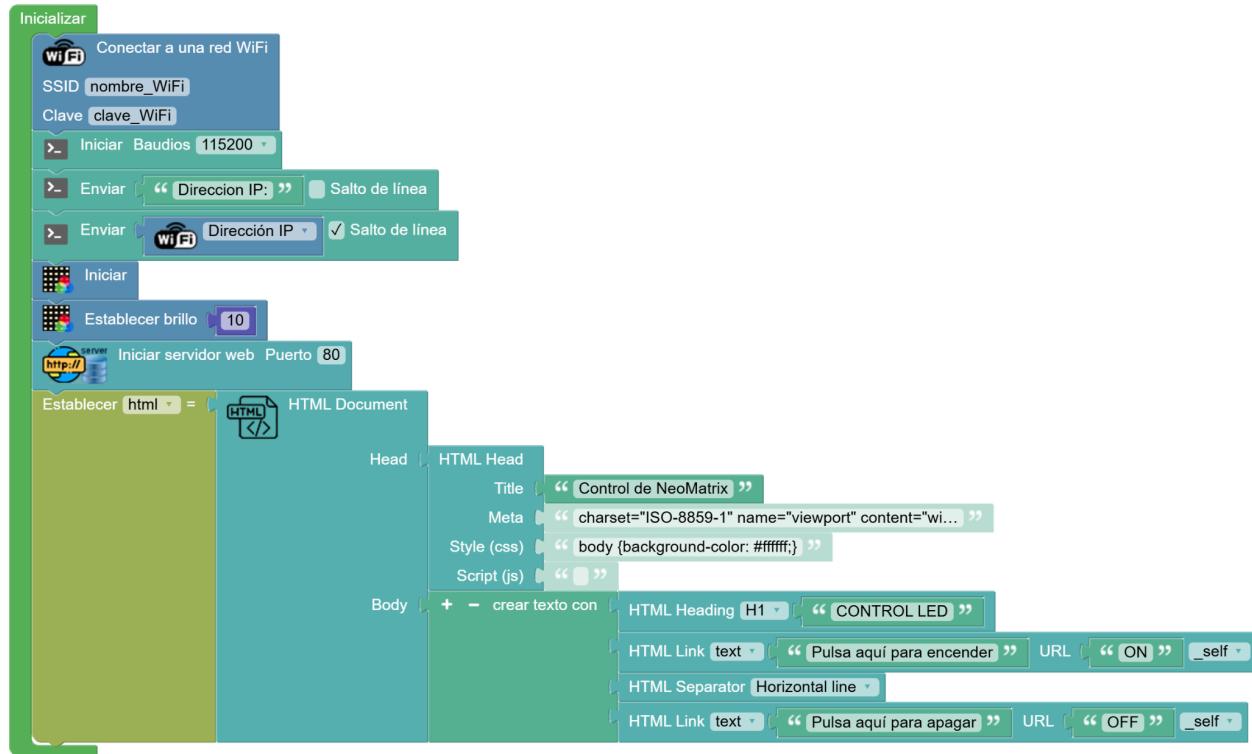
Cada bloque implementa una (o varias) etiquetas básicas del lenguaje de marcas HTML. Un ejemplo de una web básica de “hola mundo”



El contenido es simplemente texto formateado según el estándar HTML. Podemos devolver respuestas en formato HTML en las peticiones del servidor HTTP, por ejemplo, esto generaría un documento HTML en caso de solicitar una web no encontrada, con un mensaje de ERROR en grande y color rojo.

- Conectar a una red Wifi.
- Configuraremos el Puerto serie a 115200 baudios
- Enviaremos la IP por el puerto serie para poder verla por la Consola
- Iniciaremos NeoMatrix
- Establecemos el brillo de NeoMatrix
- Iniciaremos el servidor web.
- Crearemos una variable de tipo texto donde introduciremos el documento HTML (HTML Document).

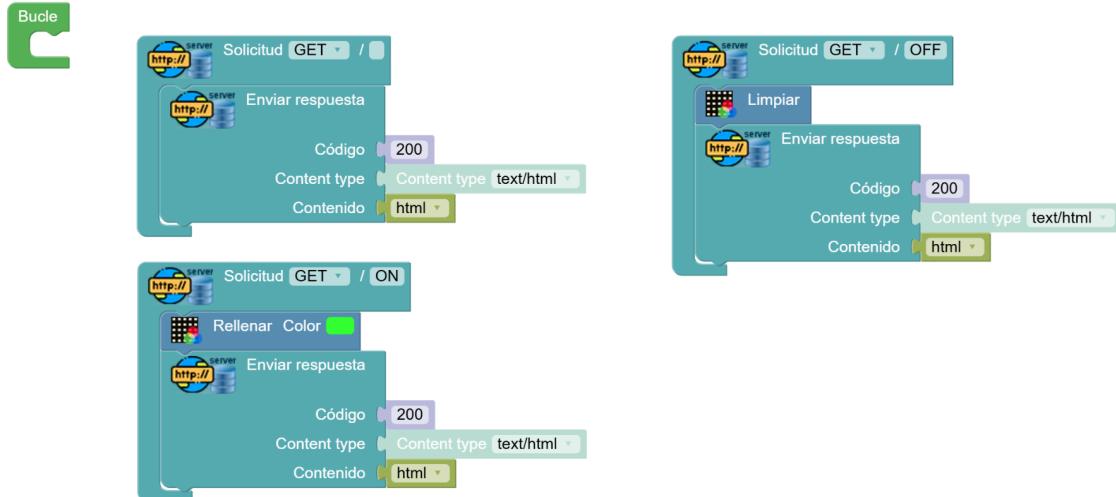
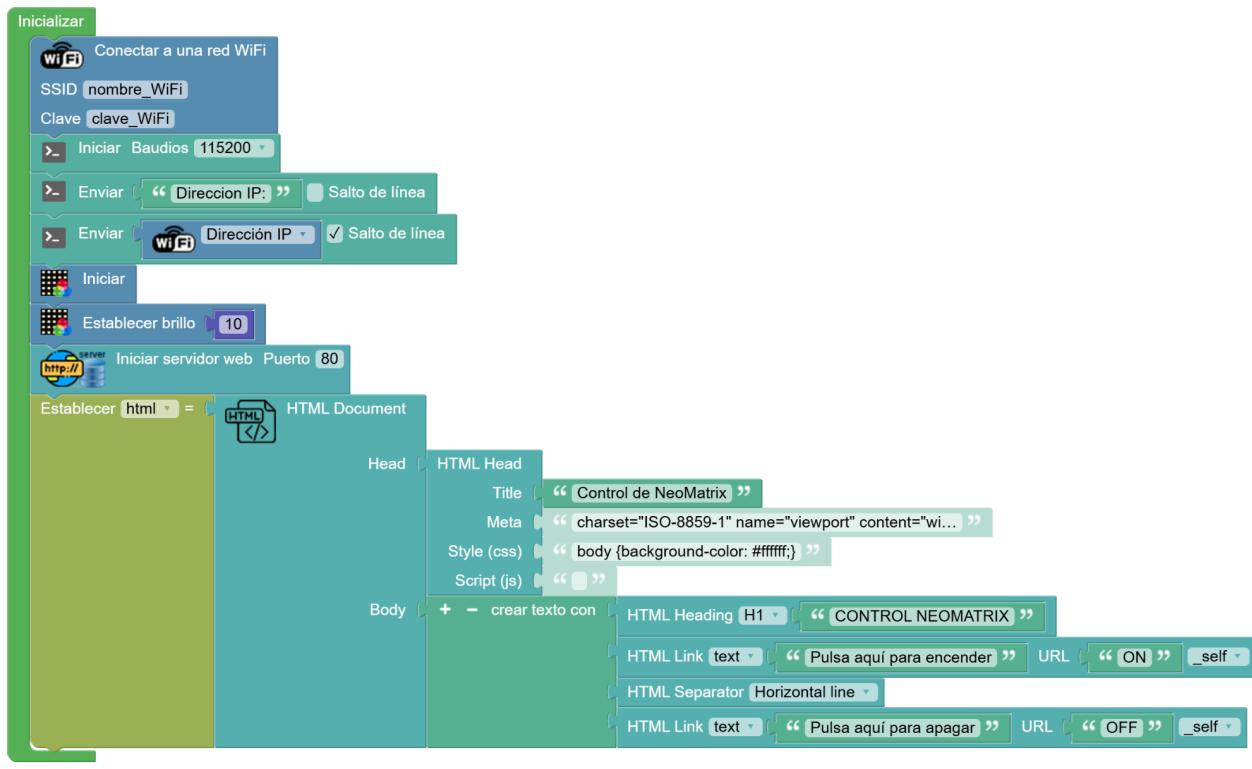
Primero realizaremos la programación de Inicializar con los siguientes elementos:



Después crearemos las tres funciones para encender y apagar la NeoMatrix, y la página de inicio.

En la siguiente página podemos ver el programa resultante.

Programa:



En el navegador introducimos nuestra IP y aparecerá la página web de inicio.



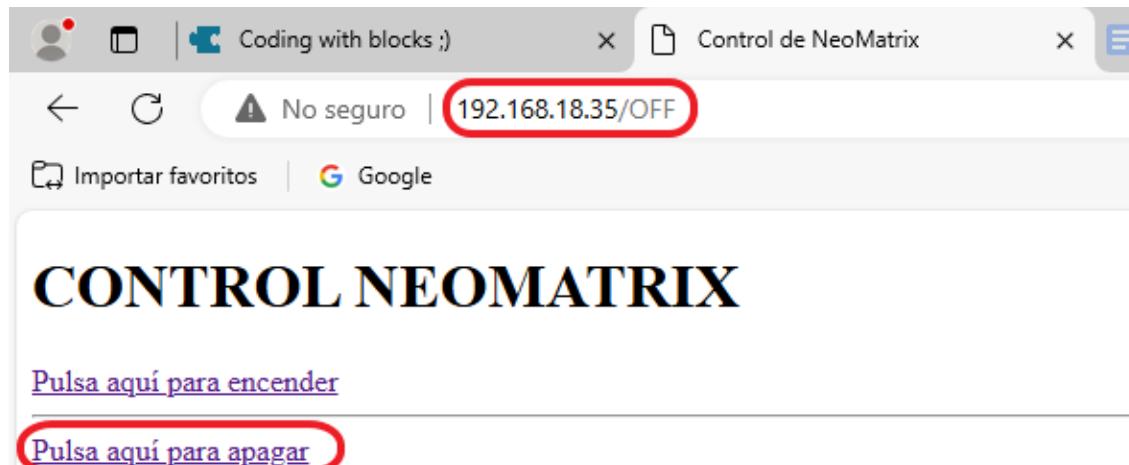
The screenshot shows a web browser window with two tabs: "Coding with blocks ;)" and "Control de NeoMatrix". The address bar shows "No seguro | 192.168.18.35". Below the address bar are links for "Importar favoritos" and "Google". The main content area displays the title "CONTROL NEOMATRIX" in large bold letters. Underneath it are two blue underlined links: "Pulsa aquí para encender" and "Pulsa aquí para apagar".

Cuando pulsemos en Pulsa aquí para encender se mostrará la siguiente imagen y se encenderá la NeoMatrix:



The screenshot shows a web browser window with two tabs: "Coding with blocks ;)" and "Control de NeoMatrix". The address bar shows "No seguro | 192.168.18.35/ON". Below the address bar are links for "Importar favoritos" and "Google". The main content area displays the title "CONTROL NEOMATRIX" in large bold letters. The first blue underlined link, "Pulsa aquí para encender", is highlighted with a red oval. Below it is another blue underlined link: "Pulsa aquí para apagar".

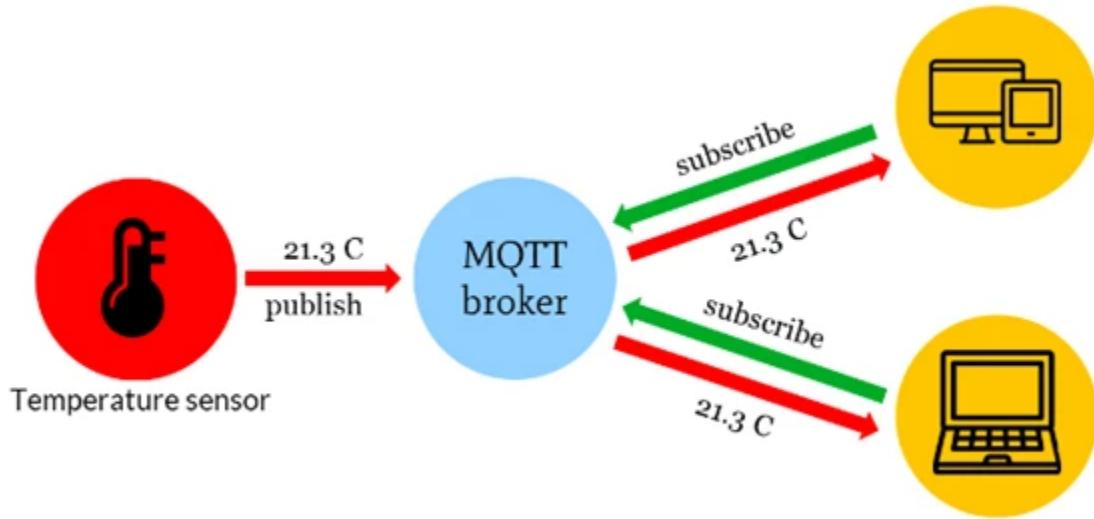
Cuando pulsemos en Pulsa aquí para apagar se mostrará la siguiente imagen y se apagará la NeoMatrix:



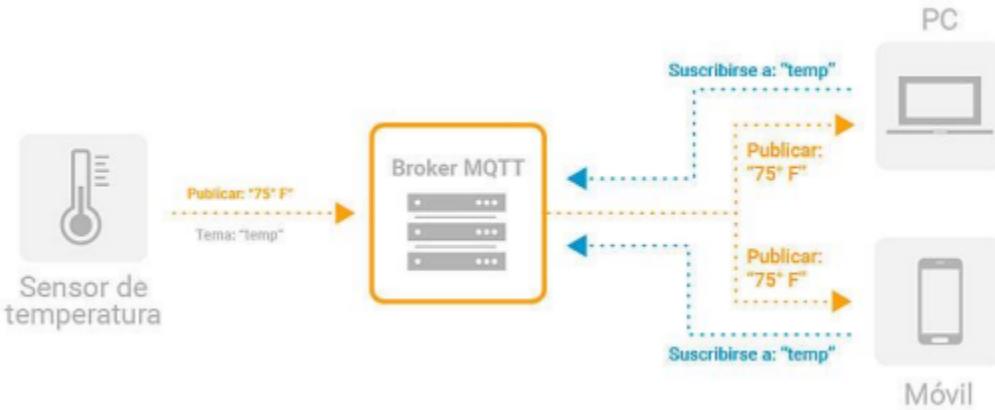
The screenshot shows a web browser window with two tabs: "Coding with blocks ;)" and "Control de NeoMatrix". The address bar shows "No seguro | 192.168.18.35/OFF". Below the address bar are links for "Importar favoritos" and "Google". The main content area displays the title "CONTROL NEOMATRIX" in large bold letters. The second blue underlined link, "Pulsa aquí para apagar", is highlighted with a red oval. Below it is another blue underlined link: "Pulsa aquí para encender".

Teoría: MQTT

MQTT es un protocolo de comunicación entre dispositivos a través de internet. Alguna de las cosas que podemos hacer con MQTT es recibir datos de diferentes sensores y enviarlos a diferentes clientes para poder ver estos datos, hacer aplicación remotas, monitorización, entre otras.



Este sistema, al mantener una conexión constante, hace posible que el servidor ya no espere la petición de los dispositivos ni tenga que preguntar constantemente si hay datos nuevos, ni realizar nuevas conexiones. Solamente quién necesita la información realiza la consulta y envía los mensajes.



Los clientes inician una conexión TCP/IP con el broker, el cual mantiene un registro de los clientes conectados. Esta conexión se mantiene abierta hasta que el cliente la finaliza. Por defecto, MQTT emplea el puerto 1883 y el 8883 cuando funciona sobre TLS.

Para ello el cliente envía un mensaje CONNECT que contiene información necesaria (nombre de usuario, contraseña, client-id...). El broker responde con un mensaje CONNACK, que contiene el resultado de la conexión (aceptada, rechazada, etc.).



Para enviar los mensajes el cliente emplea mensajes PUBLISH, que contienen el topic y el payload.



Para suscribirse o desuscribirse se emplean mensajes SUBSCRIBE y UNSUBSCRIBE, que el servidor responde con SUBACK y UNSUBACK.



Por otro lado, para asegurar que la conexión está activa los clientes mandan periódicamente un mensaje PINGREQ que es respondido por el servidor con un PINGRESP. Finalmente, el cliente se desconecta enviando un mensaje de DISCONNECT.

De igual forma si usamos un servidor/broker MQTT en internet debemos asegurarnos de conectar nuestro dispositivo a internet, mientras que de igual forma podríamos usar la opción de crear un punto de acceso propio siempre que uno de los dispositivos conectados a mi red tuviera el servidor MQTT activo.

Actividad 15.1. MQTT: Enviar datos a ThingSpeak

Vamos a realizar la programación utilizando MQTT aplicado a ThingSpeak. Para poder visualizar los datos enviados desde la placa **ESP32 micro:STEAMakers** utilizaremos el programa ThingSpeak y la aplicación ThingView (opcional).

Deberemos realizar los siguientes programas:

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

Para configurar los tres elementos hemos de seguir los pasos descritos a continuación, aunque solamente con **arduinoblocks** y ThingSpeak ya podemos hacer nuestra primera práctica de IoT.

Primero de todo tenemos que crear una cuenta en ThingSpeak:
<https://thingspeak.com/>

Create MathWorks Account

Email Address

 !

Missing required information

To access your organization's MATLAB license, use your school or work email.

Location

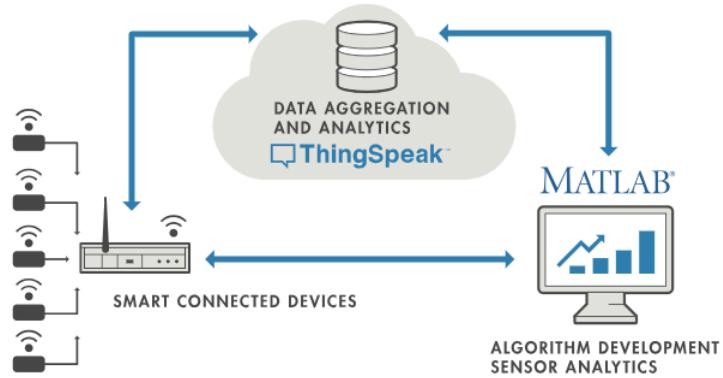
United States

First Name

Last Name

Continue

Cancel



Una vez creada la cuenta tenemos que configurar el canal en ThingSpeak.

Configuración canal ThingSpeak

Primero de todo debemos registrarnos en la plataforma de ThingSpeak.

Para crear el canal debemos seguir los siguientes pasos:

- 1) Debes ir a donde dice “**Channels**” y desde allí pinchar sobre “**New Channel**”

My Channels

New Channel

Search by tag

- 2) Una vez adentro pones los datos necesarios como qué nombre quieres poner, los campos que creas necesarios hacer,etc.

New Channel

Name	<input type="text"/>
Description	<input type="text"/>
Field 1	Field Label 1 <input checked="" type="checkbox"/>
Field 2	<input type="text"/> <input type="checkbox"/>

- 3) Una vez configurado el canal vamos al apartado de arriba que dice “**Devices**” y de allí seleccionamos el apartado de “**MQTT**”.

ThingSpeak™

Channels ▾ Apps ▾ Devices ▾ Support ▾

ESP32STEAMakers con

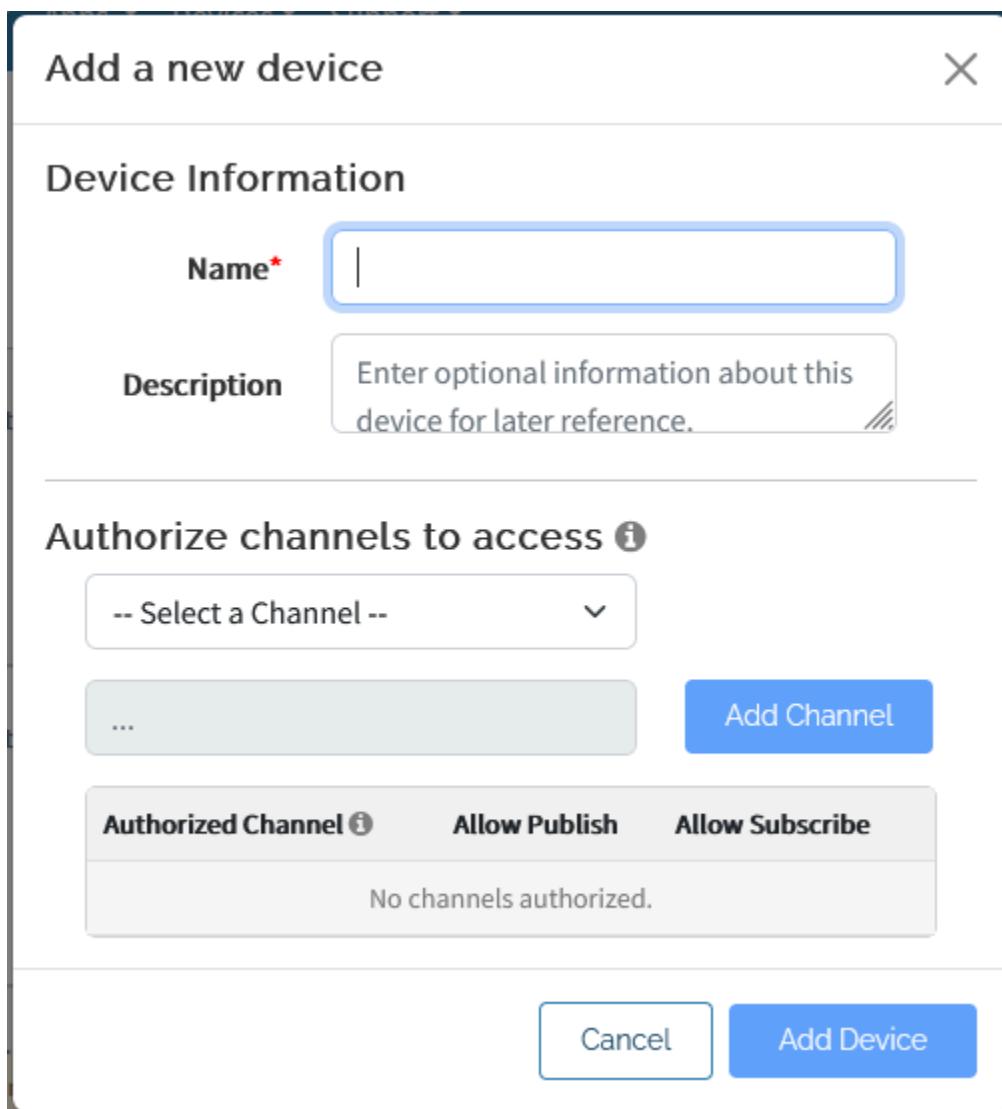
MQTT

- 4) Le damos clic al botón que dice “**Add a new device**” para añadir nuestro dispositivo en ThingSpeak

MQTT Devices

Add a new device ▾

- 1) Nos muestra una ventana que nos pide información de este dispositivo que es: el nombre que le queremos poner que puede ser cualquier inventado, una descripción queréis poner. La parte más importante es la de abajo a donde dice “Authorize channels to access”. Si hacemos clic en el desplegable se nos mostrará una lista de los canales recientes que hemos creado y desde ahí seleccionamos el canal donde queremos escribir con nuestra placa **ESP32 micro:STEAMakers**. Después damos en “Add channel” y podemos seleccionar qué permisos dar a este dispositivo respecto al canal, y finalmente apretamos “Add Device”.



- 2) Nos muestra otra ventana a donde tenemos toda la información sobre el dispositivo que acabamos de añadir. Esta información es muy importante para nosotros a la hora de programar en **arduinoblocks**. En el apartado de “**MQTT Credentials**” tenemos el cliente ID, nombre de usuario y la contraseña del dispositivo para escribir en el canal que hemos escogido antes. **Muy Importante** ThingSpeak no almacena una copia de la contraseña, por lo tanto debemos guardar escritos en algún lugar o descargar el archivo que nos proporciona ThingSpeak.

New Device Added

Device Information

ThingSpeak has added a new MQTT device and authorized it to access the channels you selected.

Device Name: placa

MQTT Credentials

Use these MQTT credentials to publish and subscribe to ThingSpeak channels. [Learn More](#)

Client ID





Username





Password

.....



ThingSpeak does not store a copy of your device's MQTT password.
Download or copy it to keep it safe.

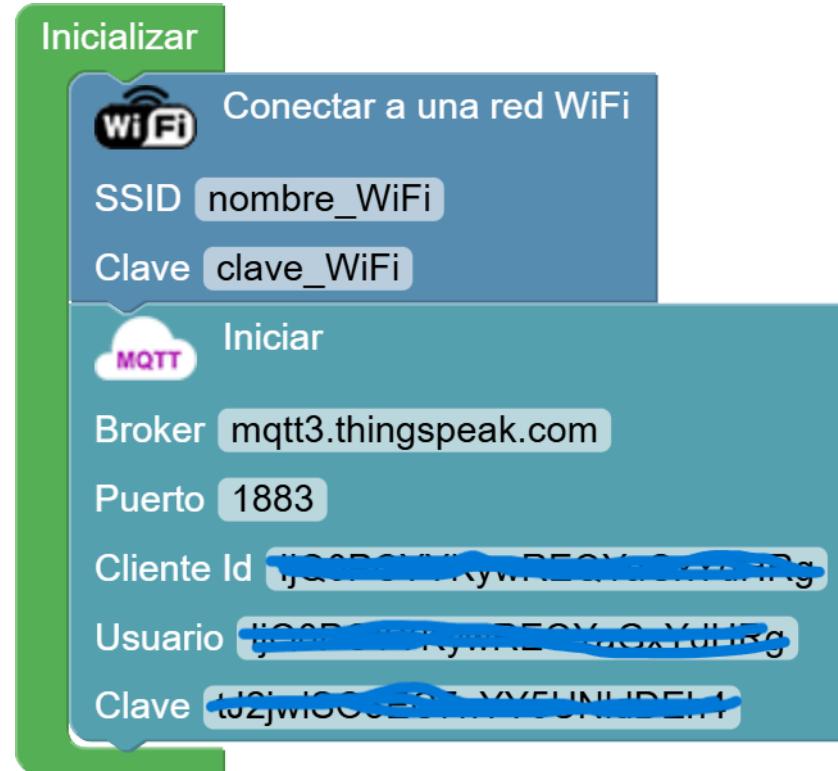
[Download Credentials](#) ▾

[Done](#)

Conectar a WIFI y servidor ThingSpeak

Para trabajar con ThingSpeak hay distintos bloques en **arduinoblocks**, pero antes de nada debemos conectar a WIFI y conectar con el servidor de MQTT de Thingspeak. Para esto debemos seguir los siguientes pasos:

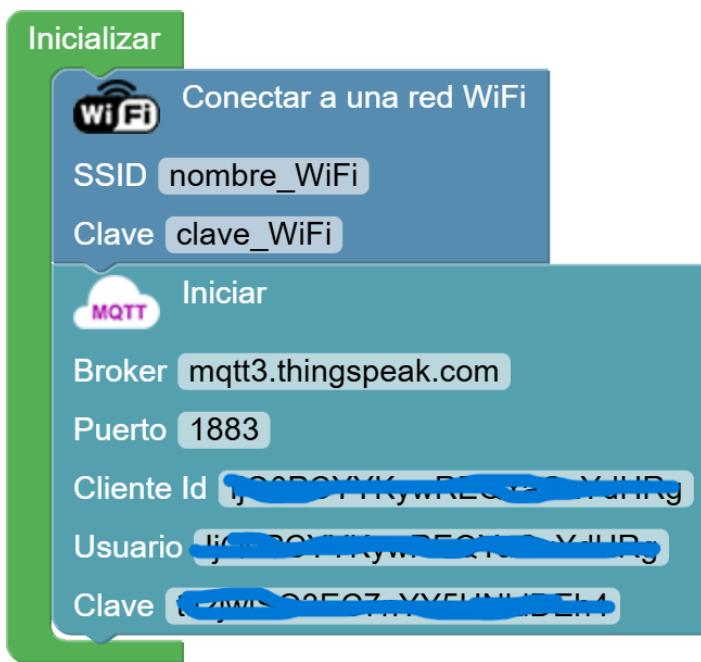
- Los bloques que usaremos en esta parte deben ir en el apartado de “**Inicializar**”
- En el apartado de “**Comunicaciones**” en “**WIFI / IOT**” hay el bloque de “**Conectar a una red WiFi**”
- Después en el apartado de “**MQTT Cliente**” encontramos el bloque de “**Iniciar**”
- En el Broker del bloque de “**Iniciar**” ponemos el servidor de ThingSpeak que es “**mqtt3.thingspeak.com**” el puerto es “**1883**” y en los siguientes apartados debemos poner las credenciales que antes hemos generado en ThingSpeak.



Programa para subir datos en ThingSpeak

Para subir datos en ThingSpeak necesitamos el bloque de “**Publicar**” en **arduino blocks** que es el bloque general para publicar datos en protocolo MQTT.

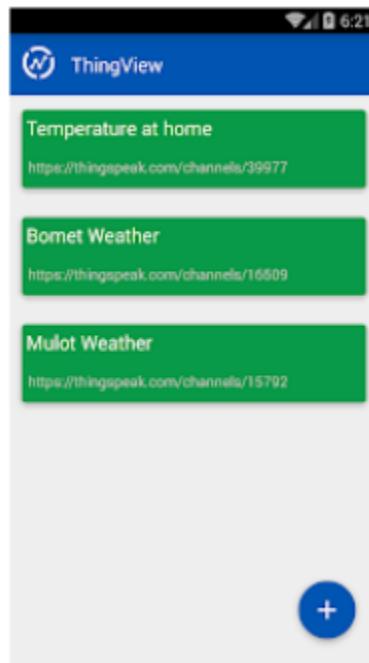
En el apartado de “Tema” tenemos que poner otro bloque “**ThingSpeak publish**” que es más específico para ThingSpeak En este bloque en “Channel ID” tenemos que poner ID del canal a donde queremos publicar nuestros datos y en “Field” ponemos en cuál de los 8 campos publicamos. En el apartado de “Valor” ponemos que es lo que queremos enviar en ThingSpeak. Y depende del tipo de cuenta de ThingSpeak pones la espera de 1 segundo o de 16 segundos.



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



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



Y podemos añadir nuestro canal en la aplicación y visualizar en el móvil.

Actividad 15.2. MQTT: Aplicación móvil de control MQTT I

En esta actividad vamos a controlar la placa **ESP32 micro:STEAMakers** con el móvil con una aplicación que va a conectar gracias al protocolo **MQTT**.

Para realizar cualquier actividad debemos conectar a un servidor MQTT que gestionará todos los mensajes que recibe desde diferentes sensores o dispositivos y envía estos mensajes al dispositivo que toca. Estos servidores pueden ser creados por nosotros o bien podemos usar diferentes servidores públicos que existen en internet.

Aquí están algunos de los servidores:

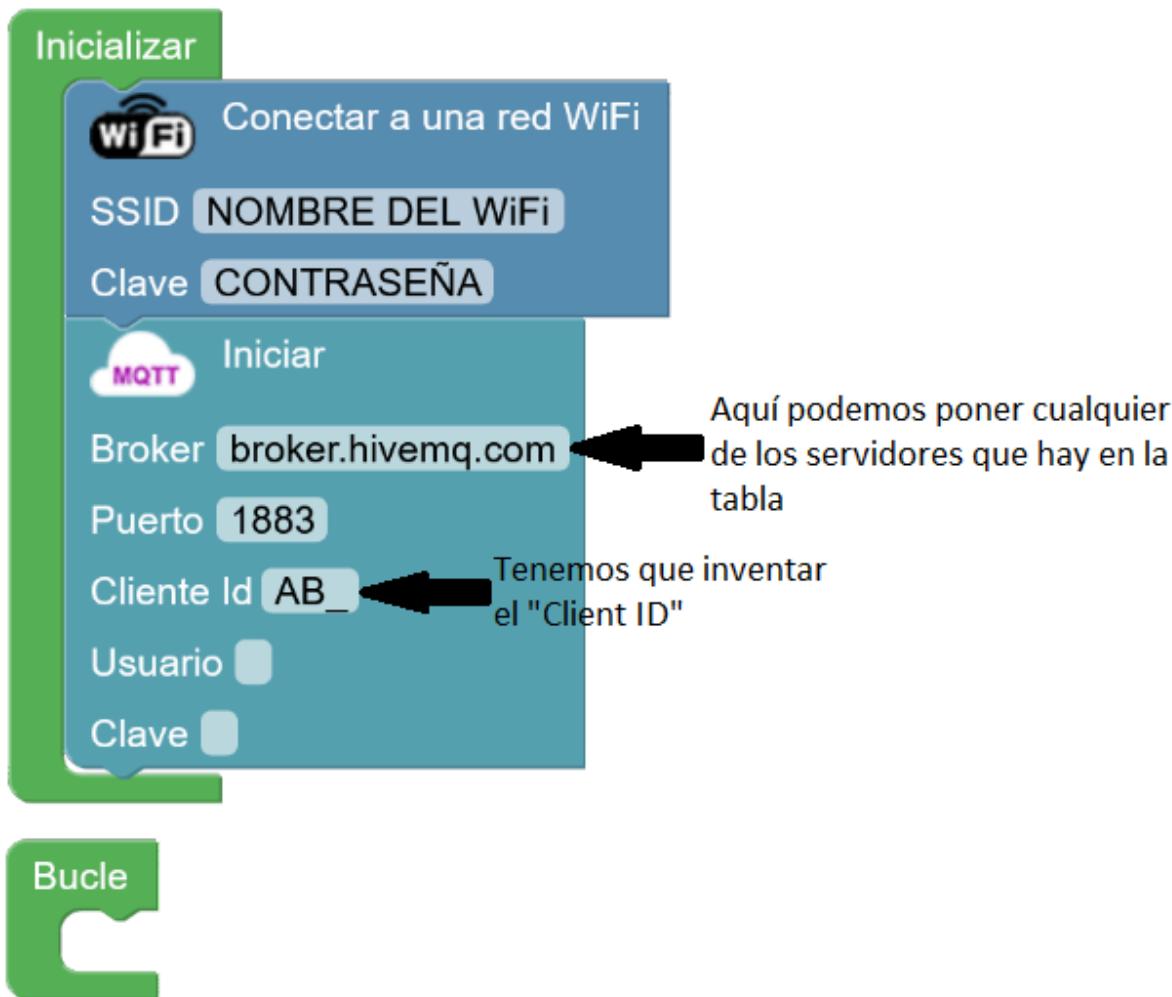
mqtt.eclipseprojects.io
test.mosquitto.org
broker.hivemq.com
http://broker.emqx.io/

Todos los servidores que hay en la tabla de encima utilizan el mismo número de puerto que es **1883**.

Conectar a WIFI y servidor MQTT

Hay distintos bloques en **arduinoblocks** para trabajar con MQTT, pero antes de nada tenemos que conectar con WiFi y conectar con el servidor MQTT. Para hacer esto tenemos que seguir los siguientes pasos:

- Los bloques que usaremos en esta parte deben ir en el apartado de “**Inicializar**”
- En el apartado de “**Comunicaciones**” en “**WIFI / IOT**” hay el bloque de “**Conectar a una red WiFi**”
- Después en el apartado de “**MQTT Cliente**” encontramos el bloque de “**Iniciar**”



Programa Ejemplo 1 ([arduino blocks](#))

Para hacer programas con MQTT hay dos bloques importantes:

- **Publicar:** Este bloque sirve para publicar valores desde la **ESP32 micro:STEAMakers** en un canal de MQTT.



- **Suscribir:** Este bloque sirve para suscribirse a un tema. El **arduino blocks** lee el valor de este tema y lo guarda en una variable que puede ser numérica o texto.
 - **Para una variable numérica**

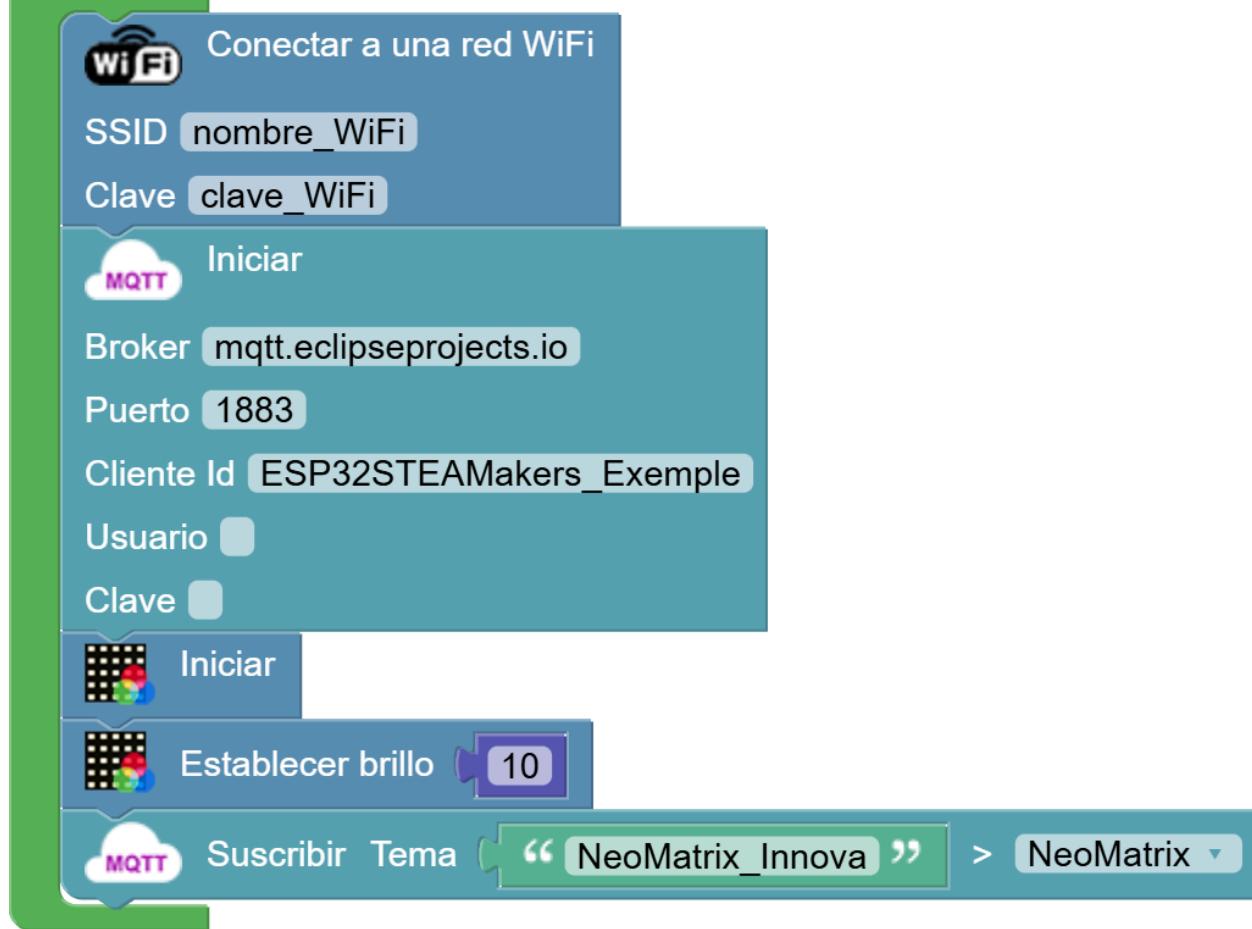


- **Para una variable de texto**

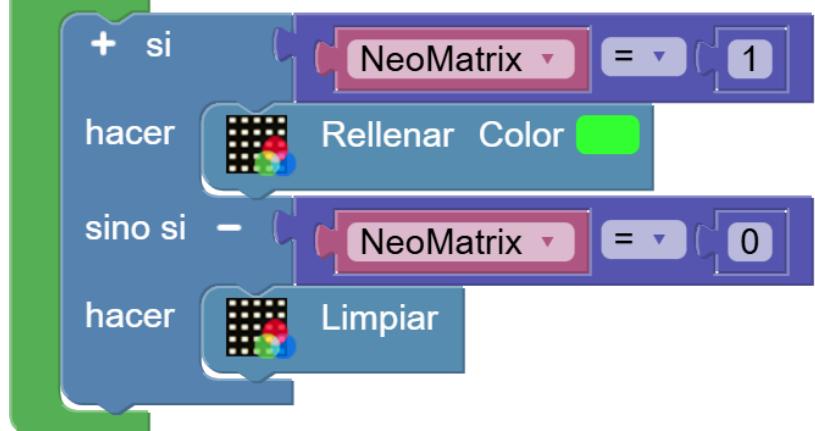


En el primer ejemplo enviamos una señal numérica desde el móvil hacia la placa **ESP32 micro:STEAMakers** y dependiendo de esta señal la enciende o apaga la NeoMatrix. El nombre del canal debe ser inventado.

Iniciar



Bucle



Aplicaciones MQTT para el móvil

Una vez que tenemos la placa **ESP32 micro:STEAMakers** programada con el programa anterior ya podemos preparar la aplicación para enviar el orden en el canal “NeoMatrix_Innova” en el nuestro caso.

Para enviar o recibir mensajes MQTT existen varias aplicaciones MQTT que te permiten crear paneles de control o monitoreo, como por ejemplo:

IOT MQTT Panel (ANDROID i IOS)

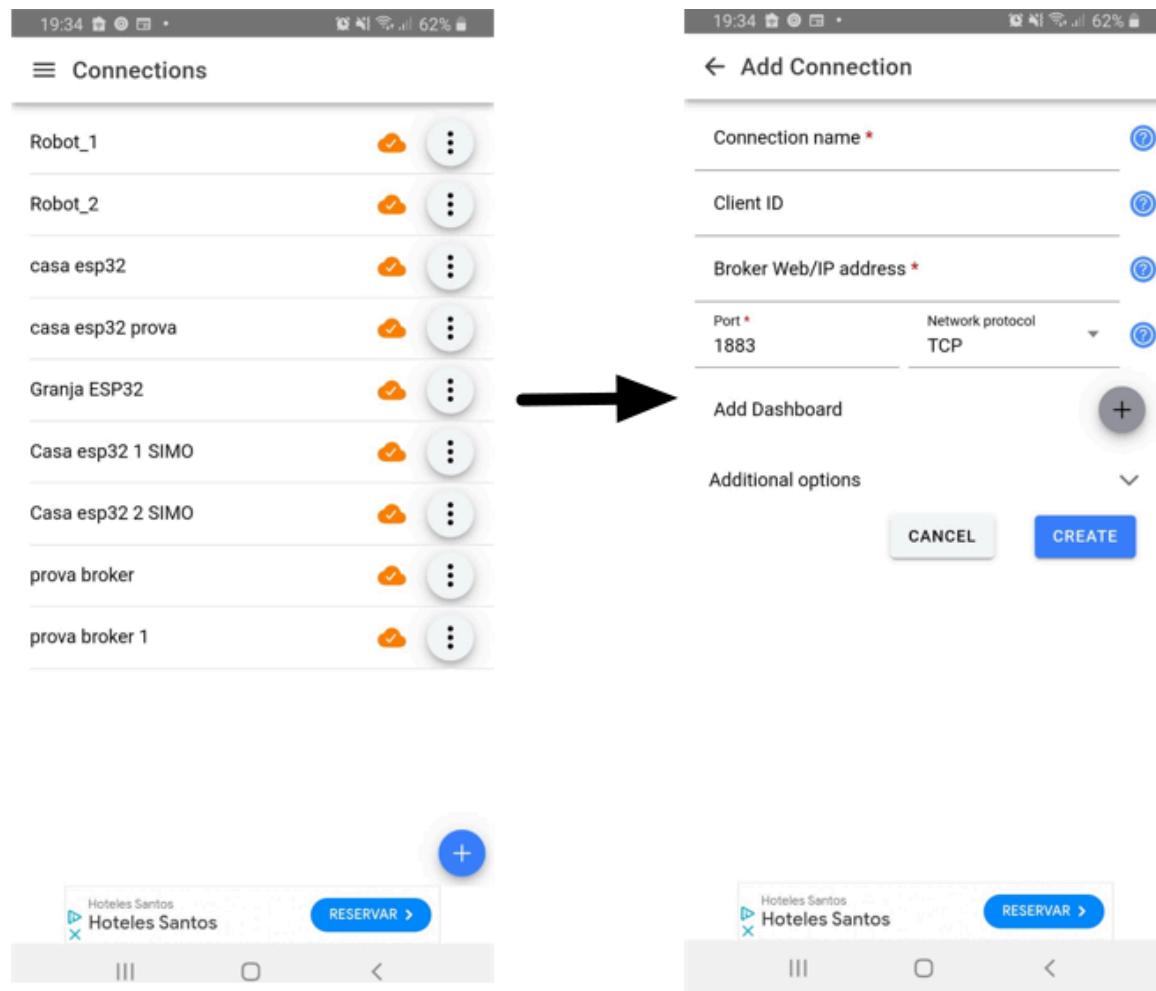


MQTT DASH (ANDROID)



Ejemplo 1 parte de la aplicación

En este ejemplo veremos cómo usar la aplicación IOT MQTT Panel. Primero de todo tenemos que añadir o crear una conexión en la aplicación. Para crear conexión tenemos que clicar el signo “+” que está debajo.

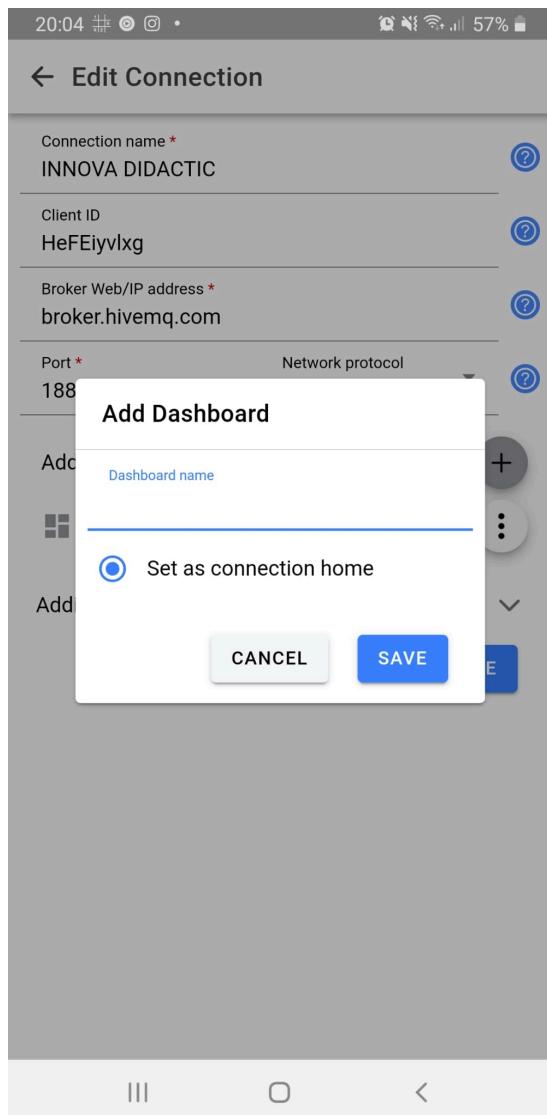


En la conexión tenemos que poner los siguientes datos:

- **Connection name:** Es para poner nombre a la conexión que puede ser cualquier inventado. Es el título de la conexión, no influye en nada con la comunicación.
- **Client ID:** Como en la **ESP32 micro:STEAMakers** en la aplicación debemos poner un “Client ID” que también debe ser inventado. ¡¡MUY IMPORTANTE!! NO tiene que ser la misma que hemos puesto en la **ESP32 micro:STEAMakers**. Ya que un servidor no puede tener 2 “Client IDs” iguales. Si dejamos este espacio en blanco la aplicación nos asigna un “CLIENT ID” automáticamente.

- **Broker Web/IP address:** Aquí ponemos el mismo servidor que le hemos puesto a la placa **ESP32 micro:STEAMakers**.
- **PORT:** Tenemos que poner el puerto en el que trabajamos en el servidor, que en nuestro caso es **1883**.
- **Network protocol:** Normalmente MQTT trabaja en el TCP.

Una vez que tengamos llenos los espacios necesarios haciendo clic en el botón “+” al lado de “**Add dashboard**” podemos añadir los “Dashboards” o escritorios que necesitamos que puede ser uno o más.



- Le ponemos un nombre a nuestro escritorio que puede ser Casa, Oficina, ESP32...
- La opción de “**Set as connection home**”: Es para cuando abrimos la conexión siempre nos abra en este escritorio.

Una vez hechos los pasos anteriores le damos al botón de guardar y ya tenemos la nuestra conexión creada. Si hemos hecho bien todos los pasos la nuestra conexión ya se habrá conectado con el servidor, para ver que se ha conectado podemos ver con el núvol que tenemos arriba de la pantalla. Este núvol puede estar en tres estados:

- **De color naranja:** esto quiere decir que está conectado con el servidor.



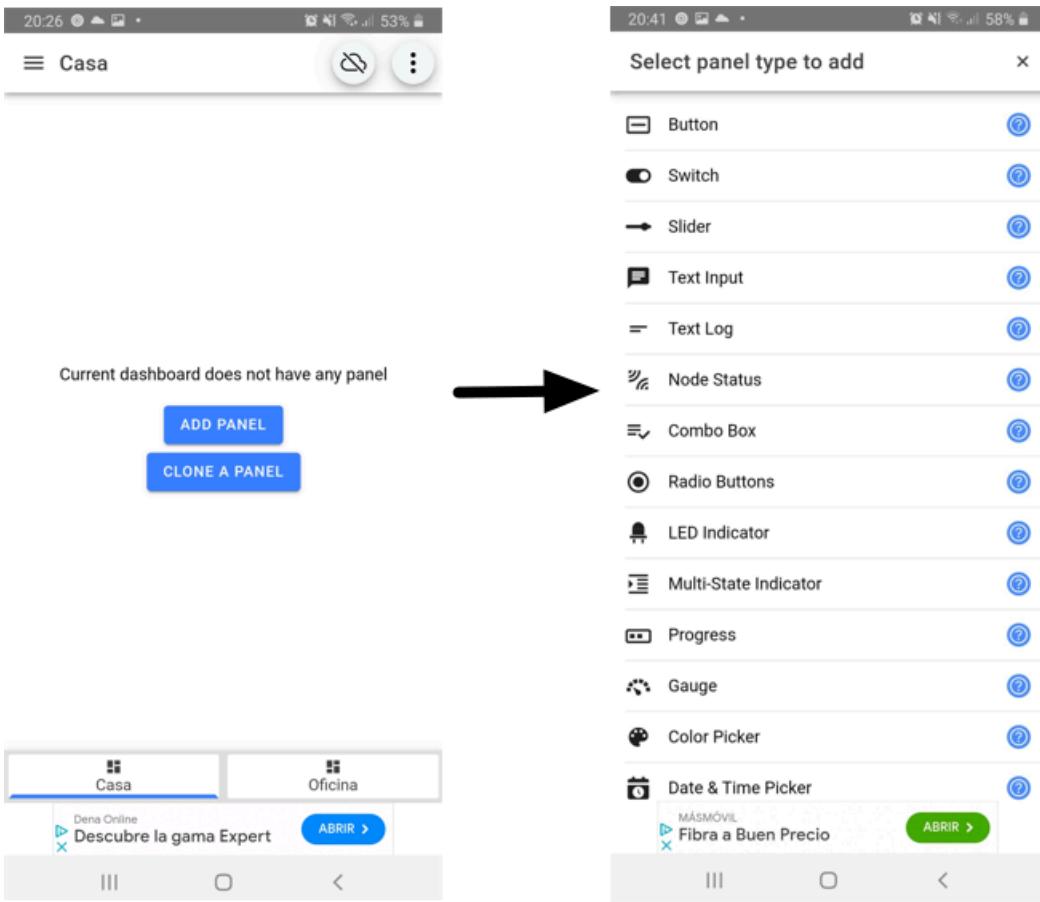
- **De color blanco:** quiere decir que no estamos conectados con el servidor



- **Intermitente (blanco y naranja):** quiere decir que el “Client ID” de la nuestra conexión está repetida en el servidor. Tenemos que cambiar el “Client ID”.

Ahora podemos empezar a crear los botones, gráficos, mensajes, etc.

Para crear un botón (Interruptor) hacemos clic en el signo de “+” abajo o si es el primer botón en el centro en “**ADD PANEL**” y escogemos la opción de “**Switch**”.



18:33 84% •

← Edit panel

Panel name *
NeoMatrix

Disable dashboard prefix topic

Topic *
NeoMatrix_Innova

Subscribe Topic

Payload on *
1

Payload off *
0

Use icon switch

On icon Icon color #ffffd00

Off icon Icon color #000000

Icon size Large ▾

Enable notification

Payload is JSON Data

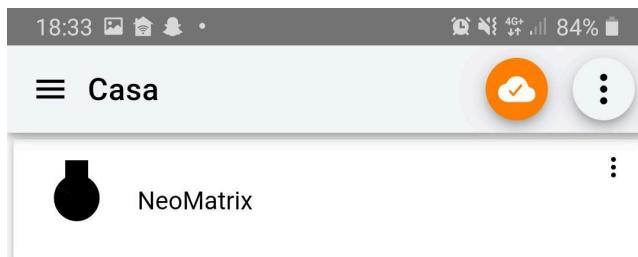
Show received timestamp

Hisense Orgullo del fútbol MÁS INFORMACIÓN >

III O <

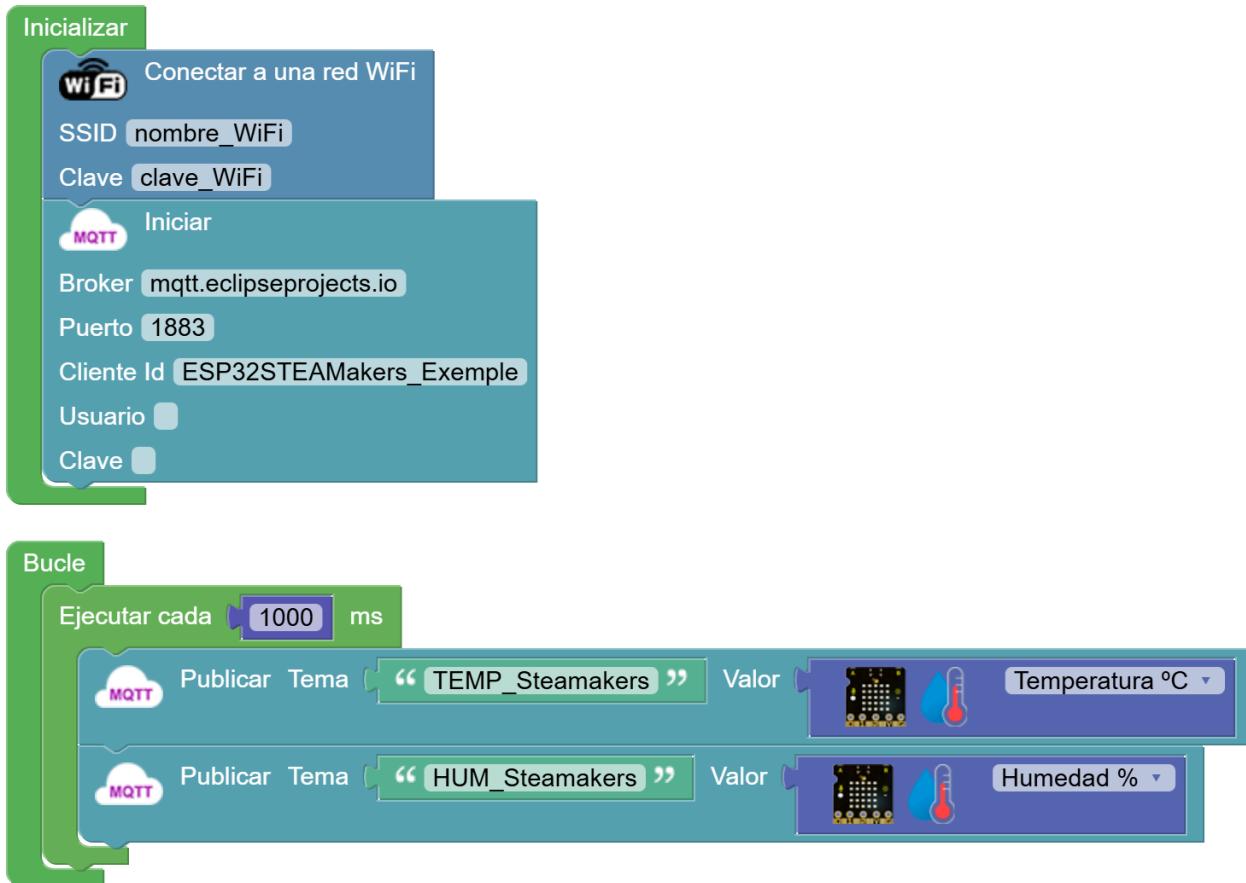
- **Panel name:** Es el nombre del panel que puede ser cualquier.
- **Topic:** El “Topic” es el que sirve para publicar o suscribir el nuestro mensaje. El “Topic” debe ser el mismo que tenemos programado en la placa **ESP32 micro:STEAMakers**.
- **Subscribe Topic:** Sirve para si tenemos más de un dispositivo enviando órdenes en el mismo “Topic”, gracias a esta función podemos saber el estado del dispositivo.
- **Payload on:** Es la señal que envía en el “Topic” cuando el botón está activado.
- **Payload off:** Es la señal que envía en el “Topic” cuando el botón está desactivado.
- En el campo de “Payload on o off” pueden ser números o letras (textos).

Después de llenar los campos necesarios ya podemos decir “Create” el botón y ya tenemos el botón creado como se ve en la imagen siguiente. Ya podemos encender y apagar el NeoMatrix de la **ESP32 micro:STEAMakers** a través del móvil.



Actividad 15.3. MQTT: Aplicación móvil de control MQTT II

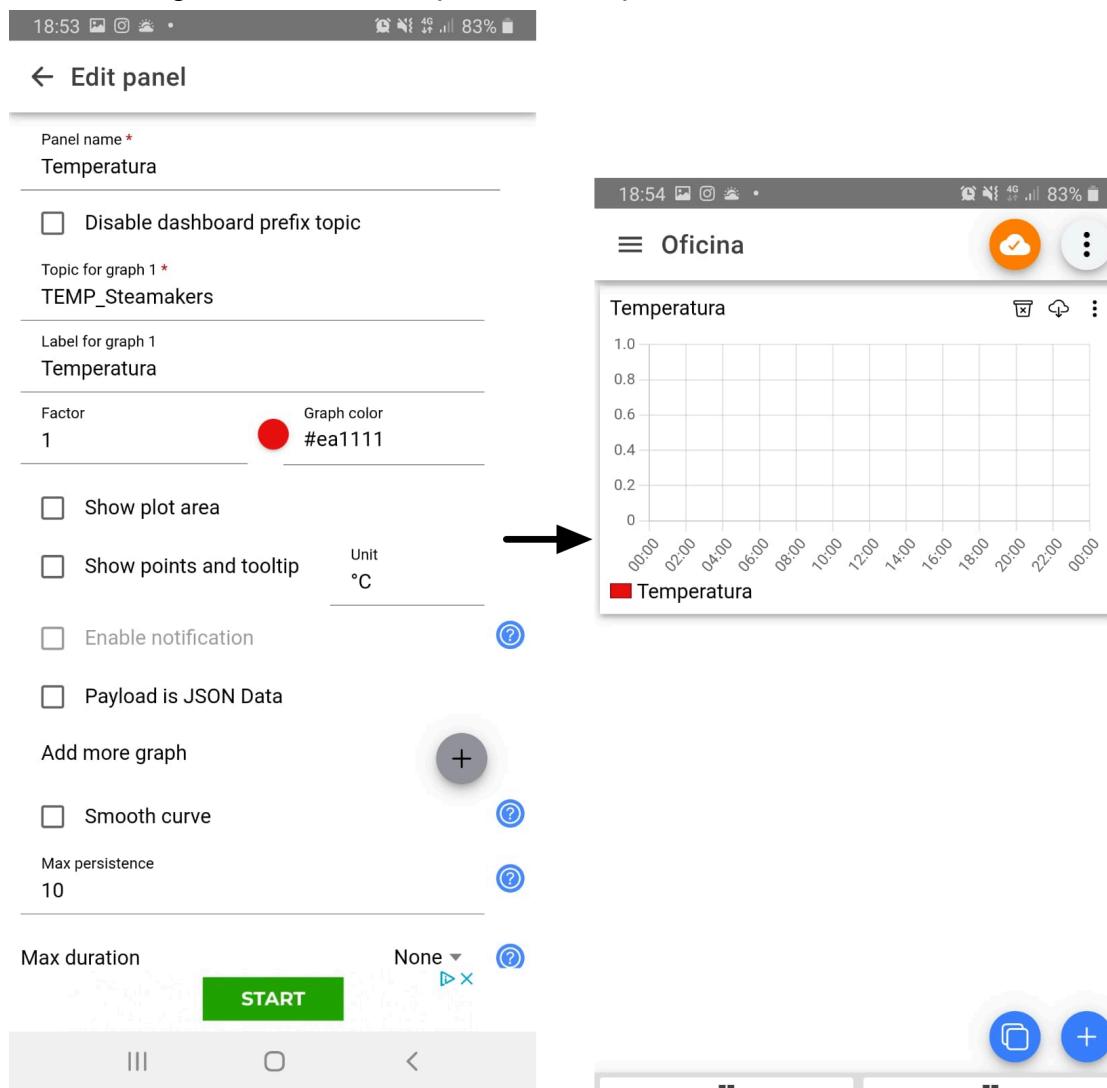
En este programa usaremos el bloque de “Publicar”. Conectamos el sensor de temperatura y humedad en la placa **ESP32 micro:STEAMakers** y los valores de temperatura y humedad que dará el sensor los pasamos al canal de MQTT.



Aplicación

Con los valores de los sensores del ejercicio anterior hacemos un gráfico en la aplicación. Podemos hacer un gráfico con las dos variables o hacer un gráfico por cada variable. Para crear el panel de gráfico tenemos que seguir los siguientes pasos:

- Pulsamos el botón de “+” para añadir un nuevo panel y escogemos “Line chart” o cualquier tipo de gráfico que nos gusta.
- Rellenamos los campos necesarios como el ejercicio anterior (Ejemplo 1 de aplicación) con “Panel name, Topic...”.
- Una vez llenados todos los campos hacemos clic en el botón de “Create” y ya tenemos el gráfico creado. Ya podemos empezar a visualizar los valores.



De la misma forma tenemos que crear el segundo gráfico que es de humedad y ya podemos ver los valores del sensor en el móvil.