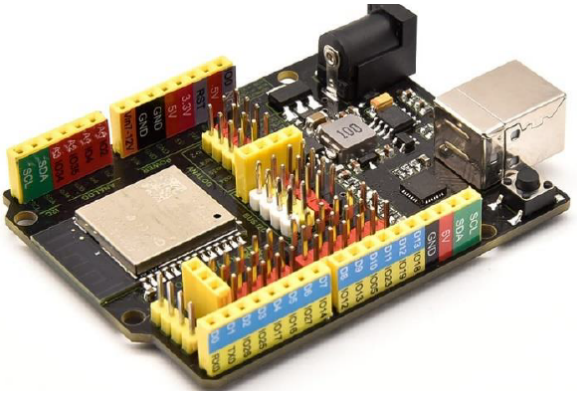
ESP32 STEAMakers



nuevos bloques

WiFi · IoT · WifiMesh

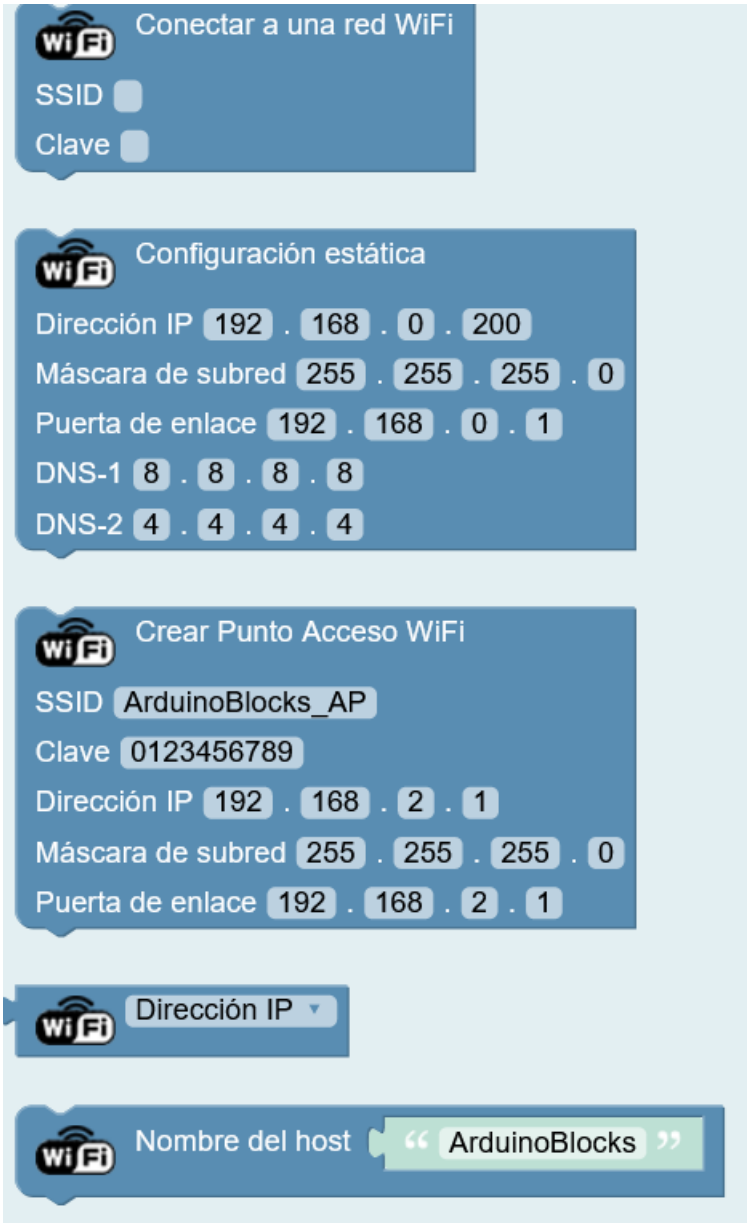
modo wifi cliente / modo wifi punto de acceso / configuración IP
cliente http / servidor http / blynk / mqtt
malla wifi entre dispositivos

Juan José López Almendros

Conectividad WiFi

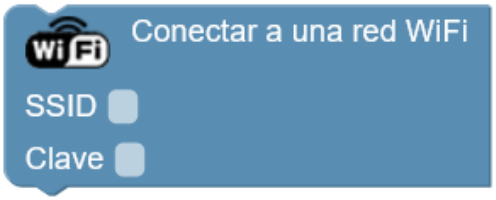
ESP32 STEAMakers incorpora conectividad WiFi sin necesidad de añadir ningún periférico o shield. El propio microcontrolador implementa el hardware y software compatible con redes Wifi 802.11 b/g/n 2.4GHz (soporta WEP/WPA/WPA2/WAPI).

Una vez iniciado en ArduinoBlocks un tipo de proyecto "*ESP32 STEAMakers*" accedemos en la barra de herramientas a los bloques de conectividad.



Conectarse a una red WiFi como cliente (modo cliente/estación)

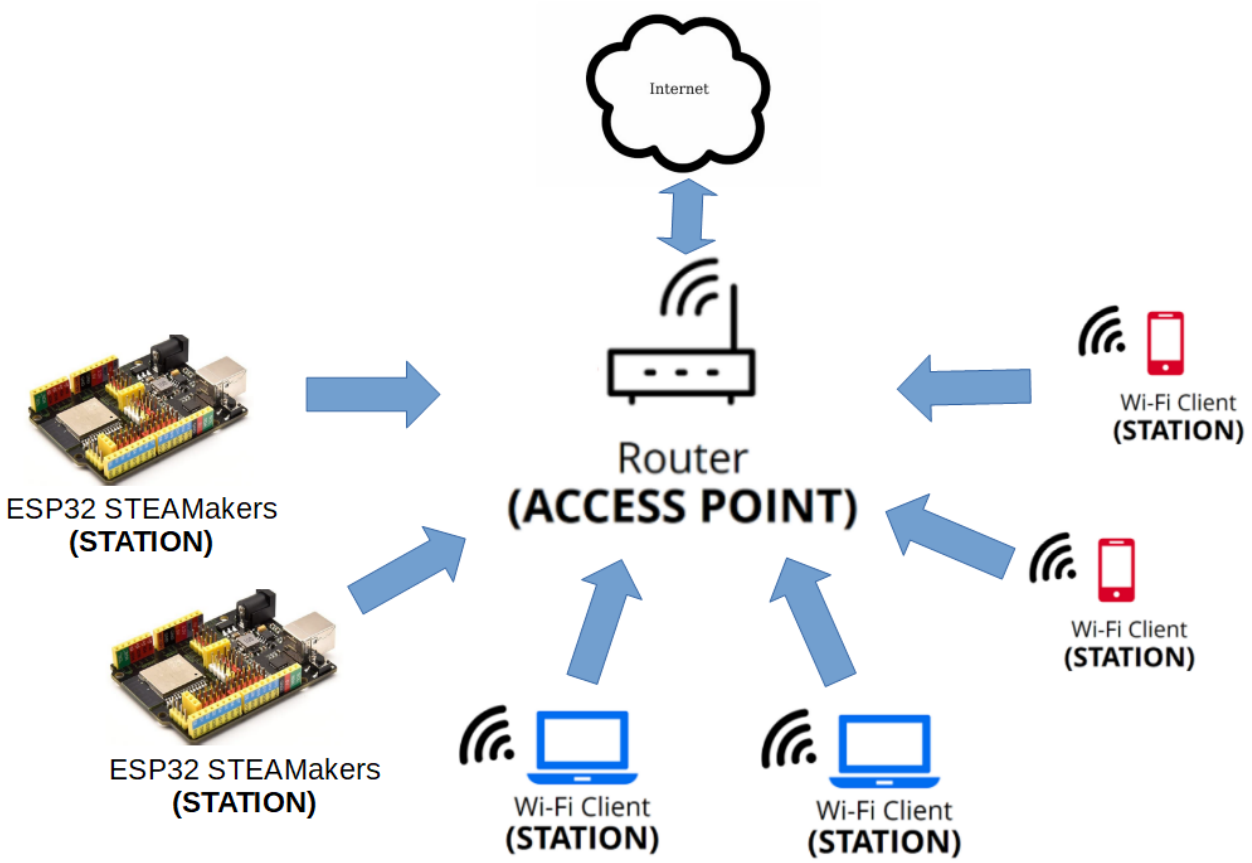
Esta opción es la más habitual. Con este bloque conectamos la placa ESP32 STEAMakers a una red WiFi existente. Para ello necesitamos el nombre de la red WiFi (SSID) y la clave (WEP,WPA,...)



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 la red a la que nos conectamos). Si todo es correcto, tendremos acceso a nuestra red WLAN y conexión a internet a través del router.

Este es el caso más simple (y el más habitual) después de este bloque ya podremos usar cualquiera de los servicios explicados más adelante (Blynk, MQTT, …)

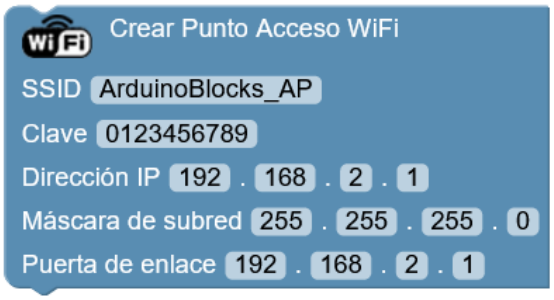
Ejemplo de una red WiFi donde varias ESP32 STEAMakers estarían en red con otros dispositivos como ordenadores, tablets, móviles, etc. A la vez que el router daría acceso a internet a todos los dispositivos.



Crear una red WiFi propia (modo punto de acceso)

En algún caso si no tenemos una red WiFi donde conectarnos, el ESP32 STEAMakers es capaz de crear su propia red WiFi, esto es que crea un punto de acceso donde nos podemos conectar con otros dispositivos de red (portátil, tablet, móvil, otros ESP32 STEAMakers, …)

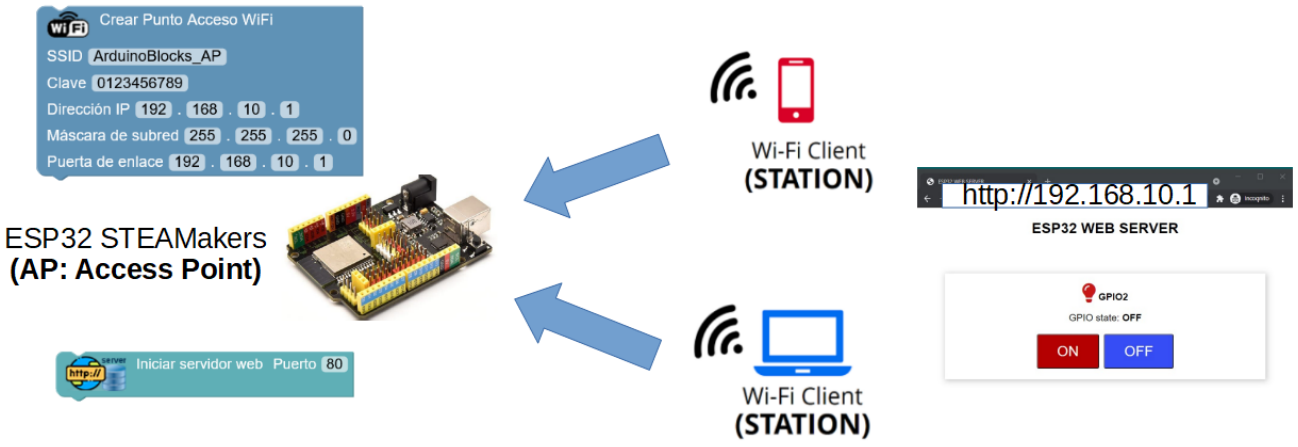
Para crear una red WiFi usaremos el siguiente bloque:



SSID: nombre de la nueva red que vamos a crear
Clave: clave para conectarse a la nueva red
Dirección IP: dirección de este nuevo punto de acceso (del propio dispositivo)
Máscada de red: por defecto 255.255.255.0
Puerta de enlace: si tenemos conexión con otro dispositivo en la red que nos de acceso a internet o a otra red podemos poner la IP como puerta de enlace, si no dejamos la misma del dispositivo.

Hay que tener en cuenta que al crear una 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 los dos (o más) dispositivos para intercambiar información.

Ejemplo: ESP32 STEAMakers en modo punto de acceso a la que se conectan dispositivos para acceder a través de un navegador a un servidor web en la propia STEAMakers:



Otros ejemplos:

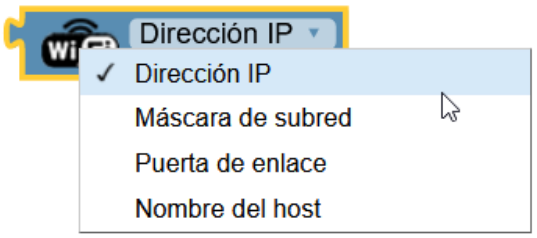
Queremos montar una estación meteorológica en un lugar apartado donde no hay conectividad WiFi, podemos conectar todos los sensores de la estación a la placa ESP32 STEAMakers y programarla como punto de acceso WiFi, de forma que cuando estemos cerca con nuestro móvil o tablet nos conectemos a esa red y podemos acceder a un servidor web del propio dispositivo donde nos dé la información de los sensores.

Un coche con la placa ESP32 STEAMakers al que nos conectemos a su red WiFi propia, y una vez conectados con una aplicación desde el móvil podamos controlar remotamente el coche.

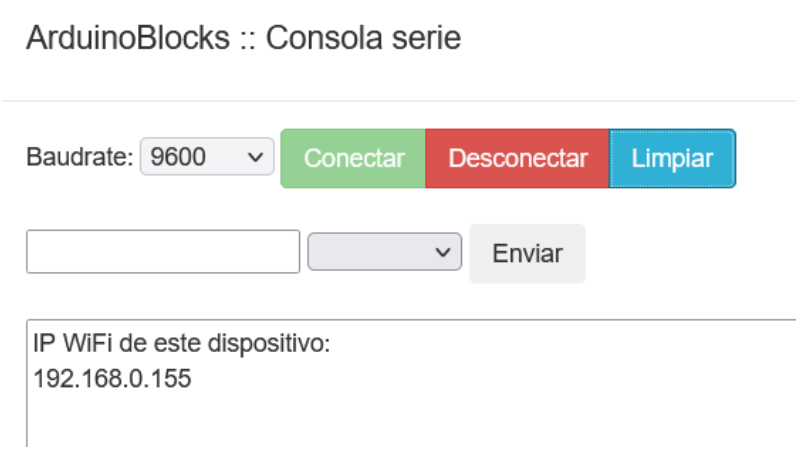
Obtener datos de mi conexión

Especialmente cuando nos conectamos a una red WiFi como cliente y el router o punto de acceso nos asigna automáticamente la configuración de red (IP) , luego necesitaremos obtener la de la dirección IP que se nos ha asignado y algunas cosas más.

Con este bloque obtenemos la dirección IP (y otros datos) de la ESP32 STEAMakers en nuestra conexión actual:



Un ejemplo muy habitual, es una vez conectados a la red WiFi mostrar en por la consola o en una pantalla LCD la IP del dispositivo para poder luego conectarnos a él desde otros dispositivos (portátil, tablet, móvil,... en la misma red)



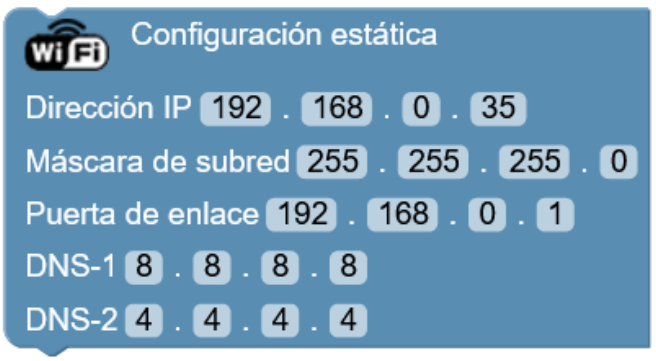
Cambiar el nombre del dispositivo en la red (hostname)

Si queremos cambiar el nombre del dispositivo en la red, tenemos el 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, 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 automático) 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.

En el ejemplo he elegido la 192.168.0.35

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.

Ya me he conectado a una red, o se pueden conectar a la mía propia… ¿ahora qué?

- **Blynk**

  Para utilizar el servicio Blynk tenemos dos opciones: o usar el servidor Blynk de internet (con limitaciones si no pagamos) o montar un servidor propio de Blynk.

  Los bloques Blynk están documentados en el libro ArduinoBlocks free edition.

  https://docs.google.com/document/u/1/d/e/2PACX-1vQSrOKHpbLQHVbGFdAvp7DcndoftoHDI20nvwGMaxu_7bGc1bUCmi4U6DZrJWRSudc2iXBg43QMuzCT/pub

  En cualquier caso si utilizamos el servicio online limitado de Blynk debemos asegurarnos que la conexión a la red WiFi de nuestro dispositivo ESP32 STEAMakers nos proporciona conexión a internet (por tanto se desaconseja la opción de crear una punto de acceso)

  Si utilizamos la opción de punto de acceso para que otros dispositivos se conecten a nuestro dispositivo, el servidor Blynk debe estar instalado en unos de los dispositivos que se conectan a mi red.

- **MQTT**

  Los bloques MQTT están documentados en el libro ArduinoBlocks free edition también.

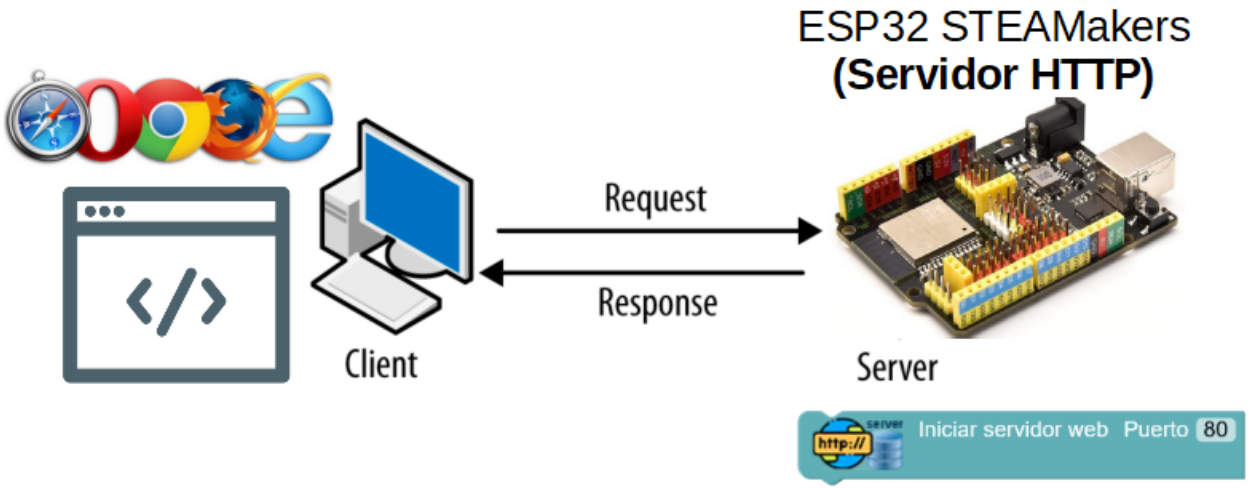
  https://docs.google.com/document/u/1/d/e/2PACX-1vQSrOKHpbLQHVbGFdAvp7DcndoftoHDI20nvwGMaxu_7bGc1bUCmi4U6DZrJWRSudc2iXBg43QMuzCT/pub

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 una punto de acceso propio siempre que uno de los dispositivos conectados a mi red tuviera el servidor MQTT activo.

- **Servidor HTTP (Web)**

La placa ESP32 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, imágen, …

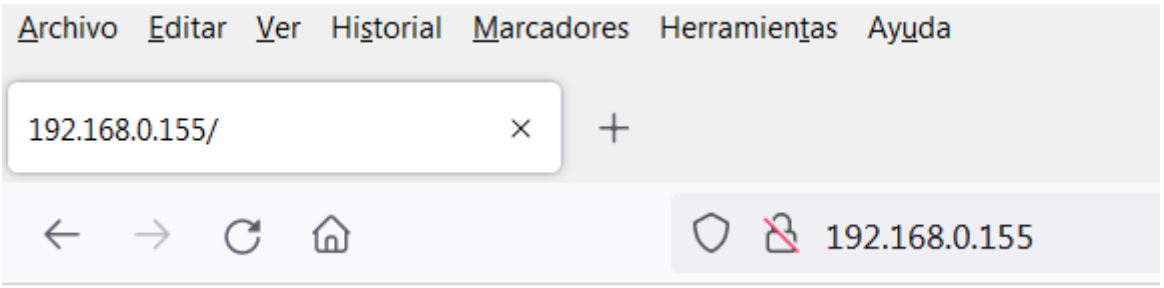
No siempre el servidor debe retornar datos útiles, el servidor web puede utilizarse como una interfaz para control remoto, permitiendo intercambiar datos y ejecutar acciones remotas.



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 192.168.0.155:

http://192.168.0.155



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

http://direccionip/accion?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)

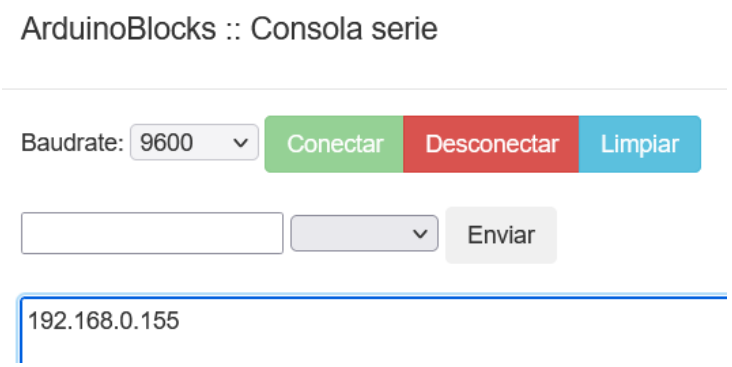
La acción (opcional) será la web o comando que queremos pedirle al servidor. En un servidor web real (este es real, me refiero a uno tradicional tipo Apache, etc…) sería el documento del servidor que estamos solicitando (ejemplo: index.html, index.php , guardardatos.jsp, ….)

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 "*&*"

1.-Ejemplo WebServer-1: Servidor web con una acción por defecto "/" para cuando no defino ninguna, y dos acciones "*encender*" y otra "*apagar*" que me permitirán controlar el encendido de un led a través de peticiones HTTP desde un navegador

1.1) Conectar a la red WiFi e iniciar el servidor web (puerto 80 por defecto)

1.2) Por la consola obtenemos la IP que tiene nuestro dispositivo



1.3) Añadimos la acción “/ “ para cuando hagamos una petición sin indicar acción:

Si no añadimos está acción, al no indicar ninguna acción devolverá un error de página no encontrada.

Pero para hacerlo bien, esta acción debería devolver algo al navegador, por ejemplo una página informativa indicando las acciones disponibles:

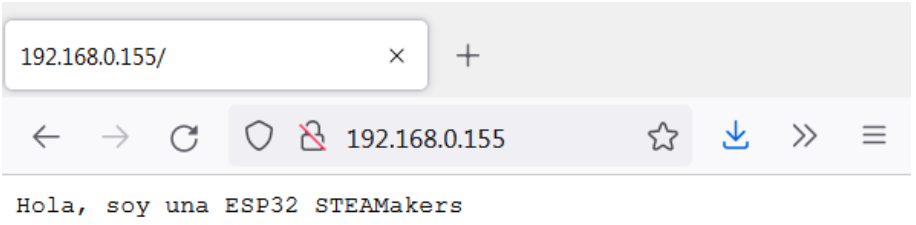
En este ejemplo la respuesta se compone de 3 partes importantes:

Código: 200 -> Código de respuesta HTTP OK

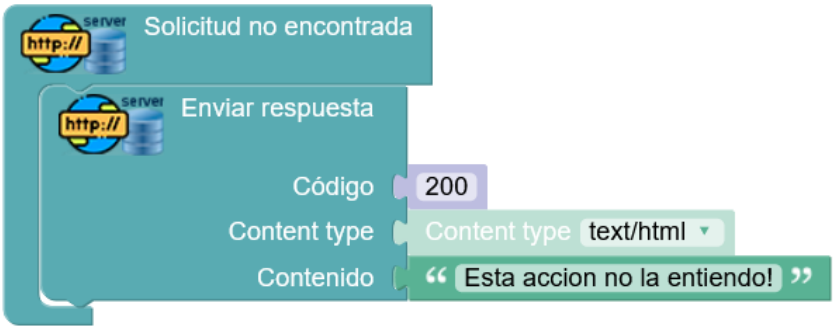
Content-Type: tipo de contenido que se va a enviar, normalmente text/html, pero podría ser text/csv, text/xml, …

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.

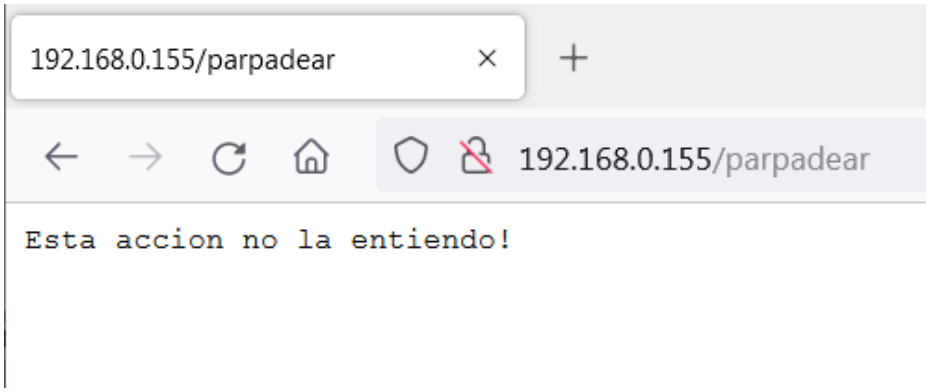
Si hacemos la solicitud desde un navegador, por ejemplo desde un móvil o un ordenador conectado a la misma red, obtenemos este resultado:



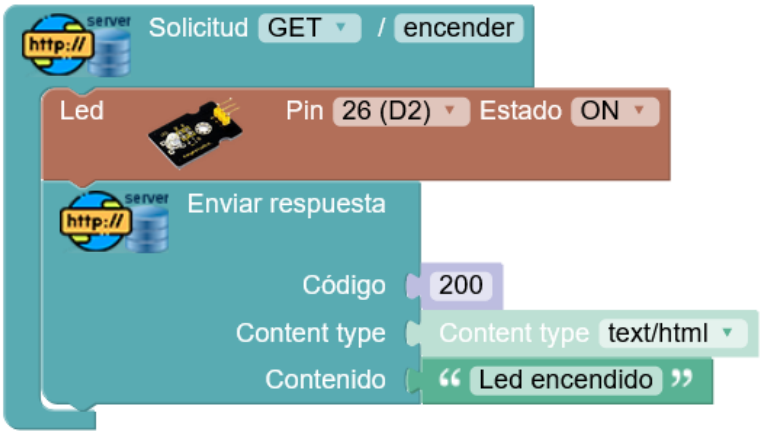
1.4) Por otro lado deberíamos añadir otra respuesta para cuando se solicite una acción no válida. Por ejemplo si pido la acción "*parpadear*" cuando sólo tengo implementadas la de "*encender*" y "*apagar*" deberíamos indicar un error.



El resultado con el ejemplo de http://192.168.0.155/parpadear es:

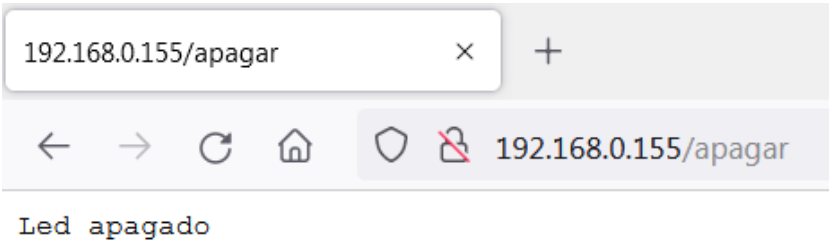


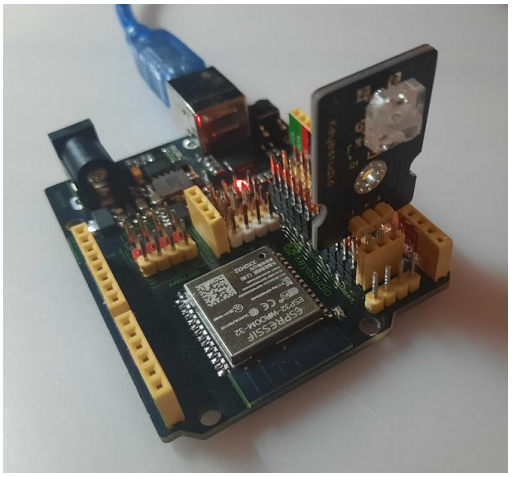
1.5) Definir las acciones correspondientes a "*encender*" y "*apagar*" el led



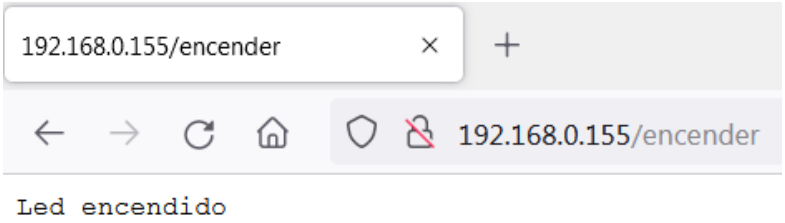
El resultado sería:

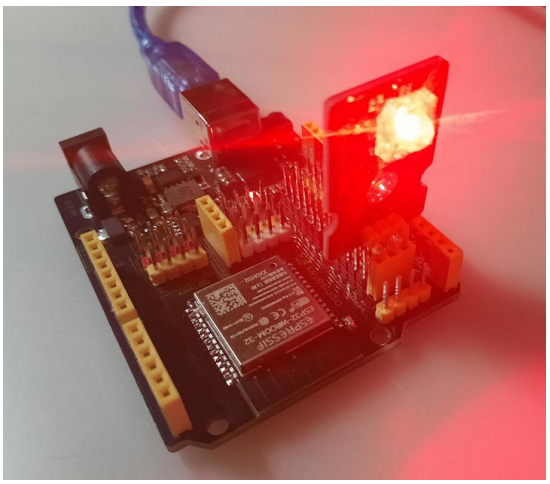
http://192.168.0.155/apagar





http://192.168.0.155/encender

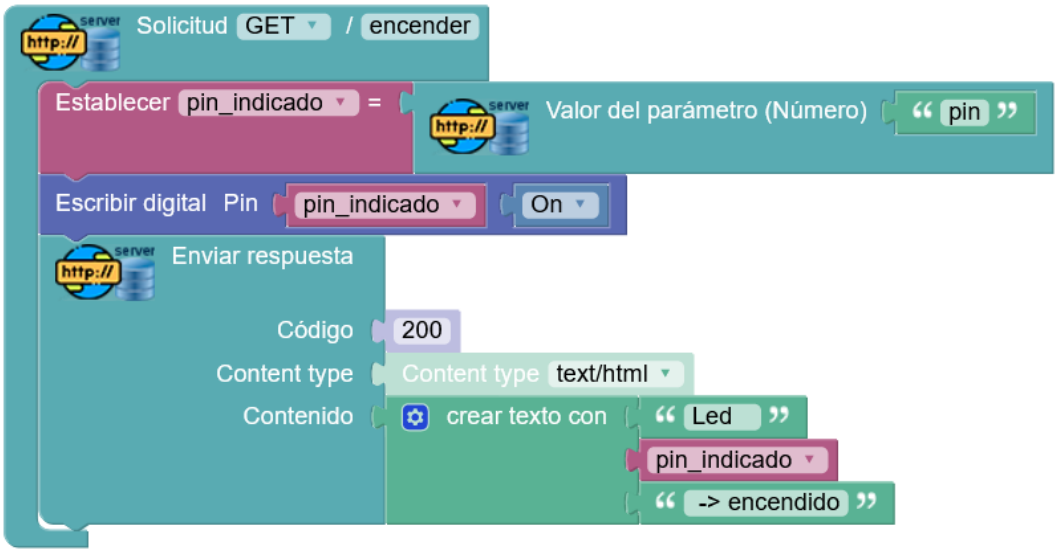




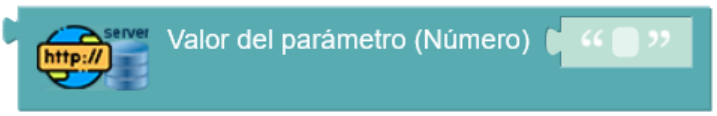
2.-Ejemplo WebServer-2: Basándonos en la idea del ejemplo anterior vamos a hacer uso de los parámetros GET para permitir que nuestro servidor web sea capaz de encender o apagar cualquier led conectado a cualquiera de los pines disponibles.

2.1) Conectar, iniciar servidor web (80), y acciones por defecto tal como en el ejemplo anterior

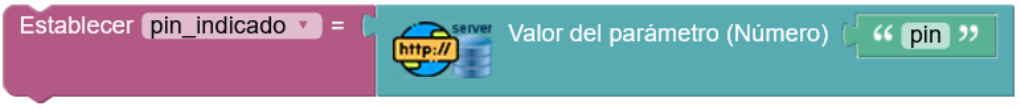
2.2) Acción encender mejorada, con un parámetro para indicar el pin



Con los bloques "*valor del parámetro*" obtenemos parámetros de la URL.

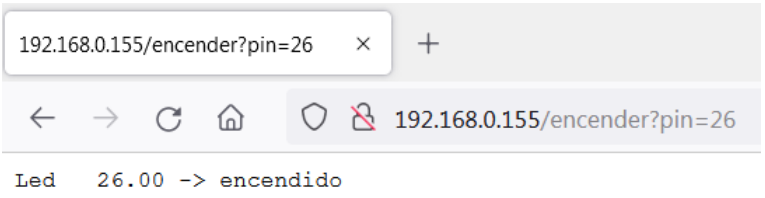


En el ejemplo usamos la versión del bloque que obtiene el parámetro y lo procesa automáticamente como un valor numérico.



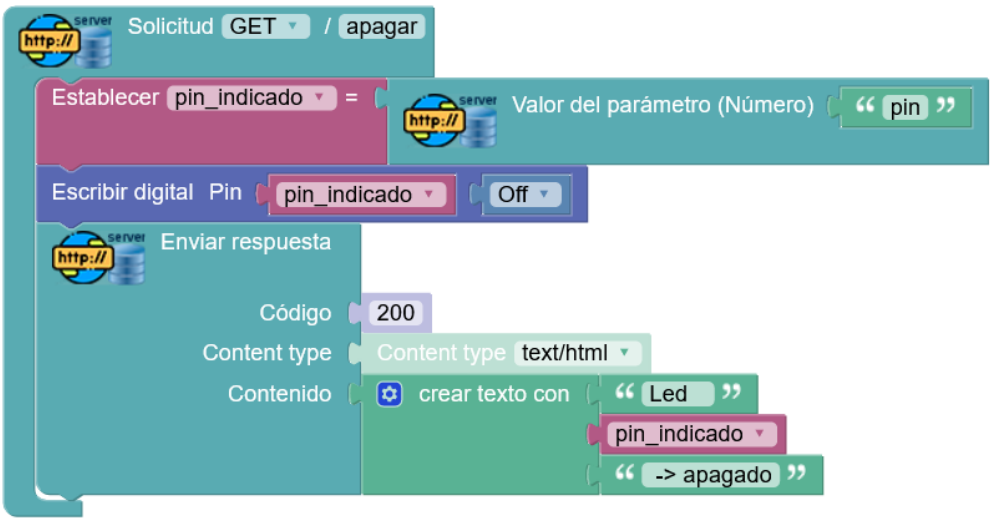
La petición correcta para encender el led anterior , conectado al pin 26 (GPIO 26) sería:

http://192.168.0.155/encender?pin=26



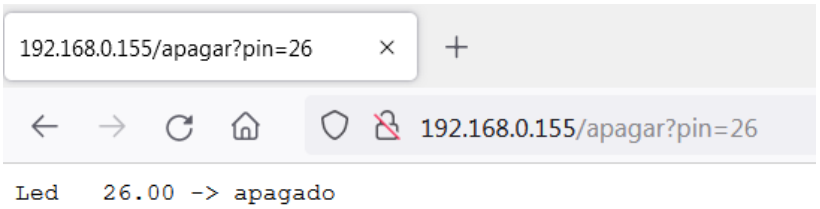
Y así podríamos indicar cualquiera de los pines GPIO disponibles para activarlo a "*ON*"

2.2) Acción apagar mejorada, viendo la anterior creo que sobran las palabras…



La petición correcta para encender el led anterior, conectado al pin 26 (GPIO 26) sería:

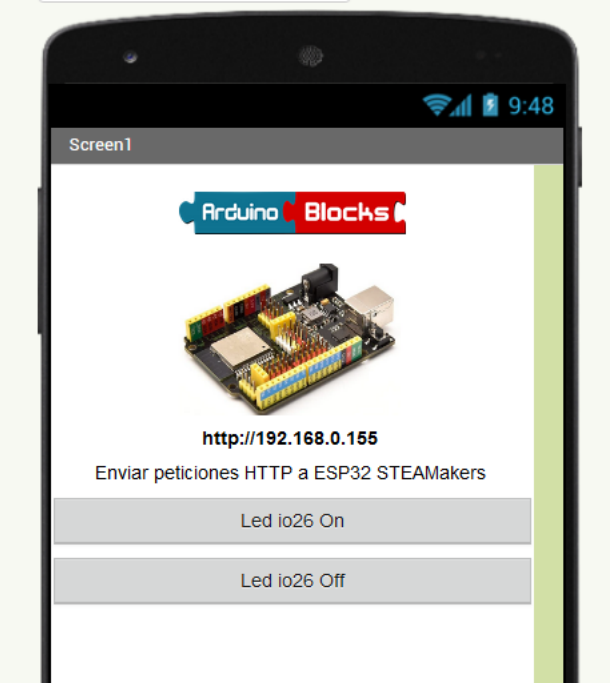
http://192.168.0.155/apagar?pin=26

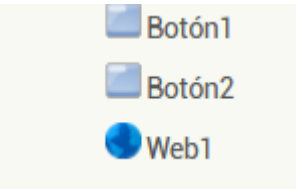


Cliente AppInventor:

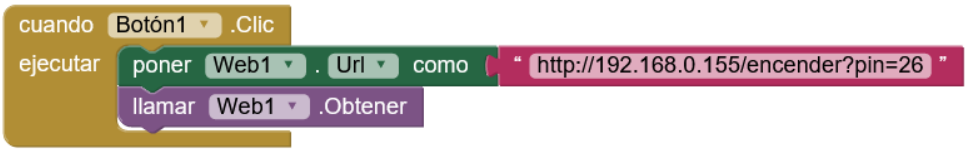
Las peticiones HTTP con parámetros GET son muy fáciles de probar y usar con el navegador, pero no siempre usamos un navegador , si no que implementamos esta opción para comunicar y controlar desde otros sistemas más personalizados, por ejemplo podríamos hacer una aplicación móvil con AppInventor que fácilmente envía peticiones HTPP (de forma transparente) desde la aplicación.

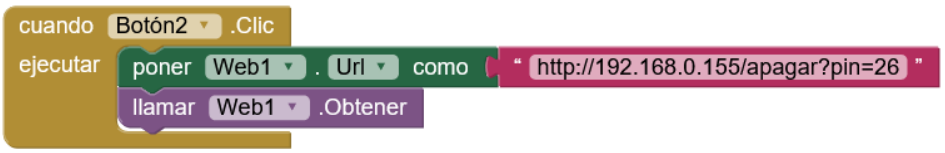
Sencilla aplicación en appinventor para probar el ejemplo anterior. Dos botones y un componente "*web*" es lo único que necesitamos:





En los bloques de código de appinventor definimos las peticiones HTTP al servidor de nuestra ESP32 STEAMakers con los parámetros para encender o apagar el pin io26



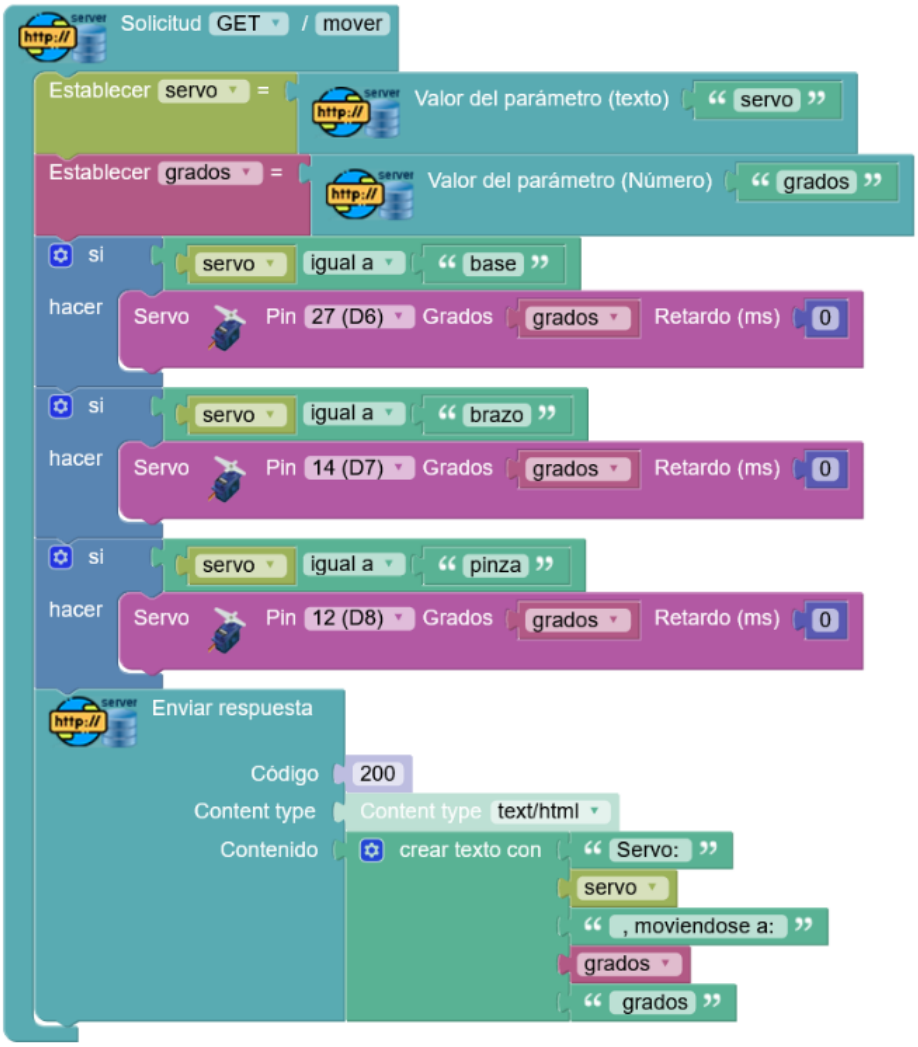


3.-Ejemplo WebServer-3: Con lo aprendido hasta ahora realizar el control de varios servos via peticiones HTTP desde un navegador.

3.1) Conectar, iniciar servidor web (80), y acciones por defecto tal como en los ejemplos anteriores

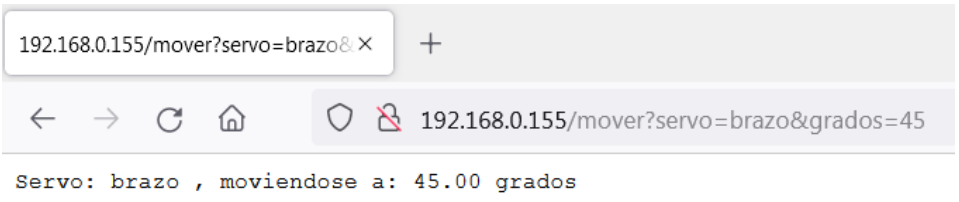
3.2) Definir acción "*mover*" con la que usaremos 2 parámetros , "*servo*" para indicar el servo (a cada uno le hemos puesto un nombre) y "*grados*" para indicar la posición a donde mover ese servo indicado.

Nombres de los servos: "*base*" , "*brazo*" y "*pinza*"



Ejemplo:

http://192.168.0.155/mover?servo=brazo&grados=45



Otros ejemplos válidos:

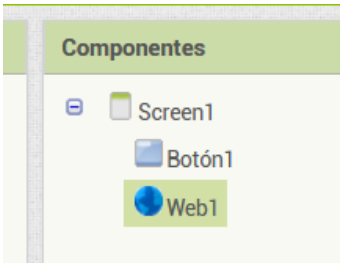
http://192.168.0.155/mover?servo=base&grados=0

http://192.168.0.155/mover?servo=pinza&grados=180

Cliente HTTP AppInventor:

Ejemplo de enviar peticiones desde AppInventor para el ejemplo anterior:

Sólo necesitamos un botón y un componente web:

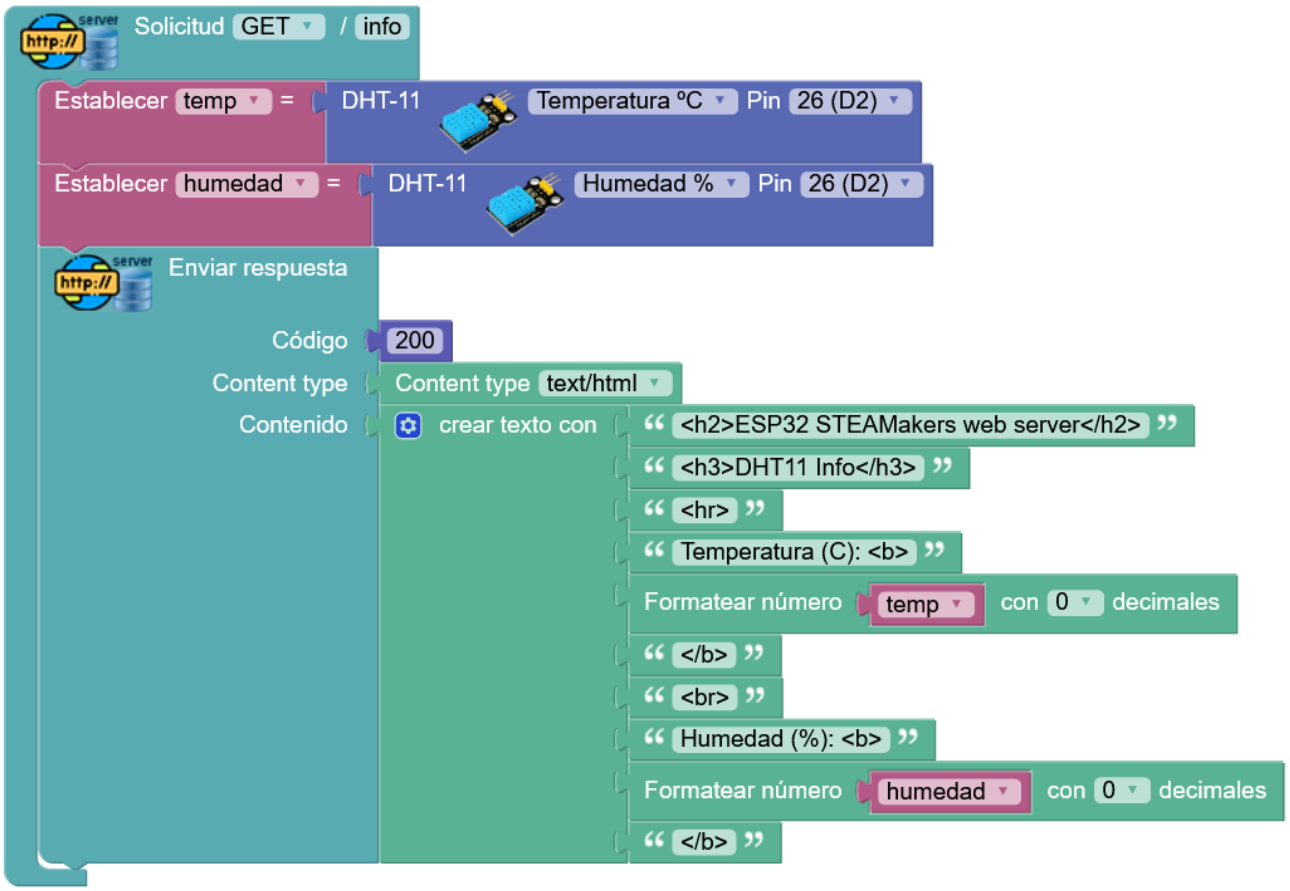


El botón al hacer clic hará una petición HTTP con la url como en los ejemplos anteriores (de forma transparente para la aplicación)

4.-Ejemplo WebServer-4: Obtener una web con la información de un sensor DHT11 que mide temperatura y humedad:

4.1) Conectar, iniciar servidor web (80), y acciones por defecto tal como en los ejemplos anteriores

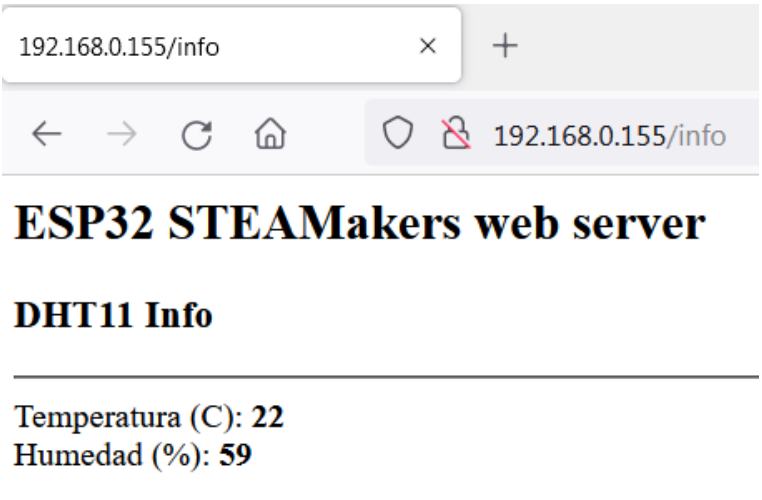
4.2.) La acción "*info*" nos dará la información de temperatura y humedad en una página web

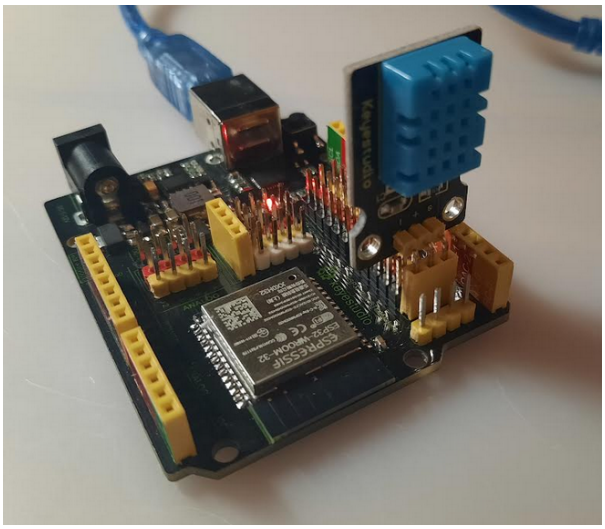


En el ejemplo para formatear la web de una forma más vistosa se utilizan códigos HTML básicos como (h2, h3, hr, b), para información sobre códigos HTML básicos podemos consultar en sitios online personalizados.

https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/aprende-html-tutorial-para-principiantes/

El resultado de la web generado por nuestra placa STEAMakers con el sensor DHT11 es el siguiente (cada vez que actualizamos la página se regenera con los datos actualizados)



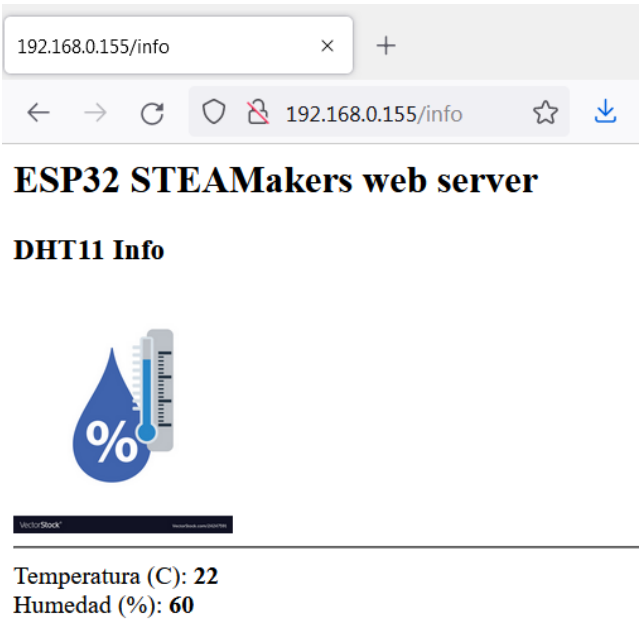


Y ya para dejarlo más vistoso... (asumiendo que nuestra red tiene internet) podemos insertar en el código HTML una imagen con la URL completa de internet.

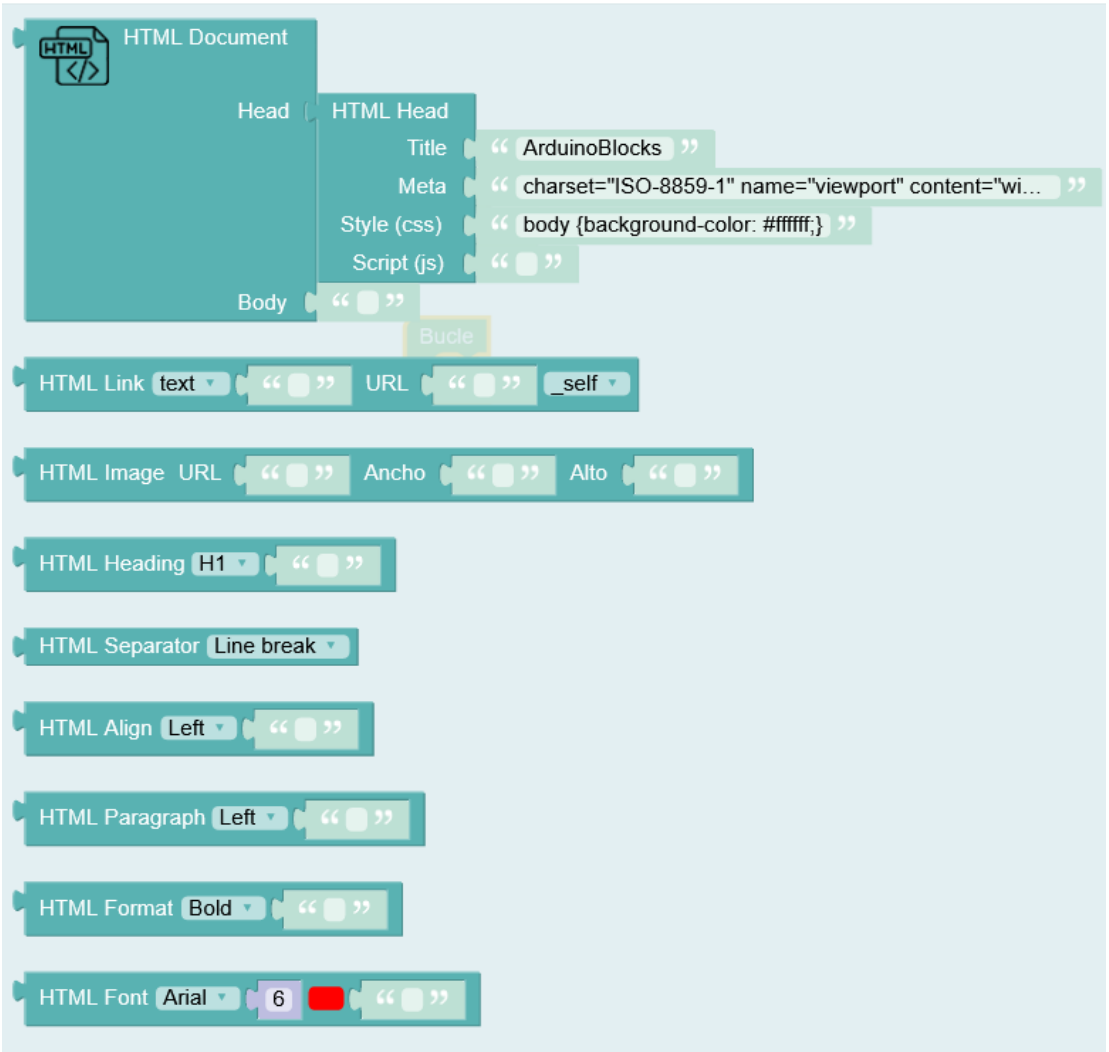
Buscamos un icono de temperatura y humedad y obtenemos su url:

https://cdn2.vectorstock.com/i/1000x1000/75/91/humidity-icon-vector-24247591.jpg

Y añadimos el código en el servidor web de la STEAMakers

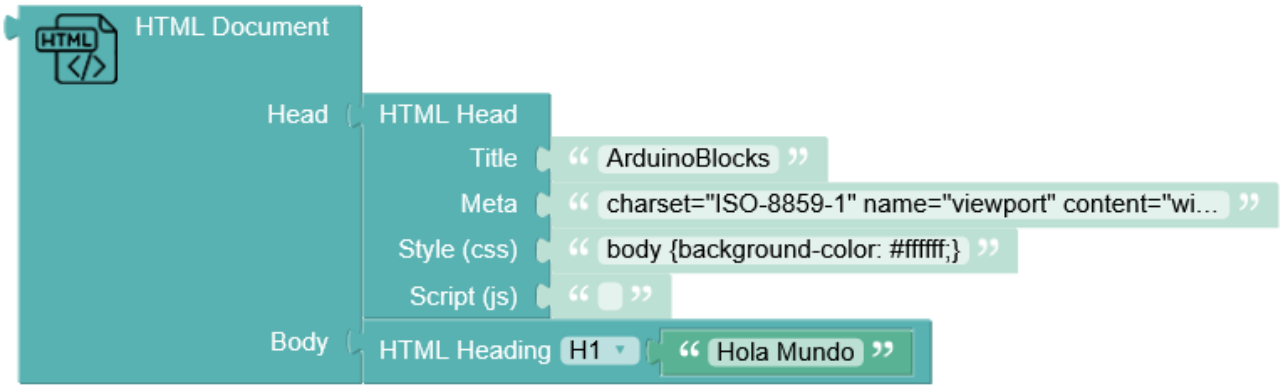


Generación de contenido HTML: En ArduinoBlocks tenemos bloques que nos permiten generar texto con formato HTML especialmente pensado para devolver contenido en formato web desde el servidor HTTP de la placa ESP32 STEAMakers.



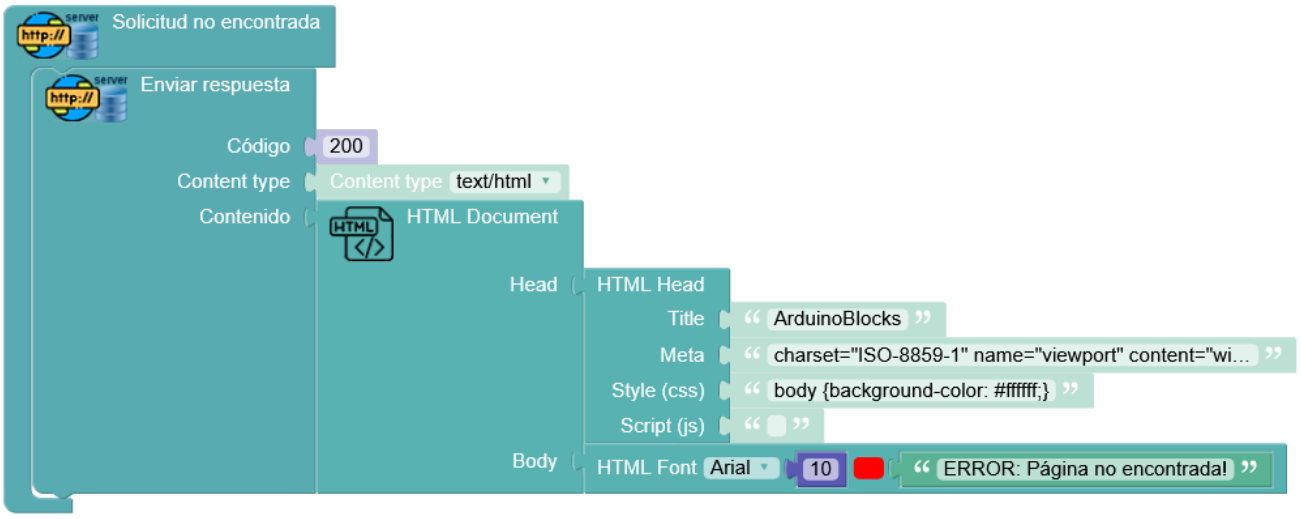
Cada bloque implementa una (o varias) etiquetas básicas del lenguaje de marcas HTML.

Ejemplo de una web básica de “hola mundo”

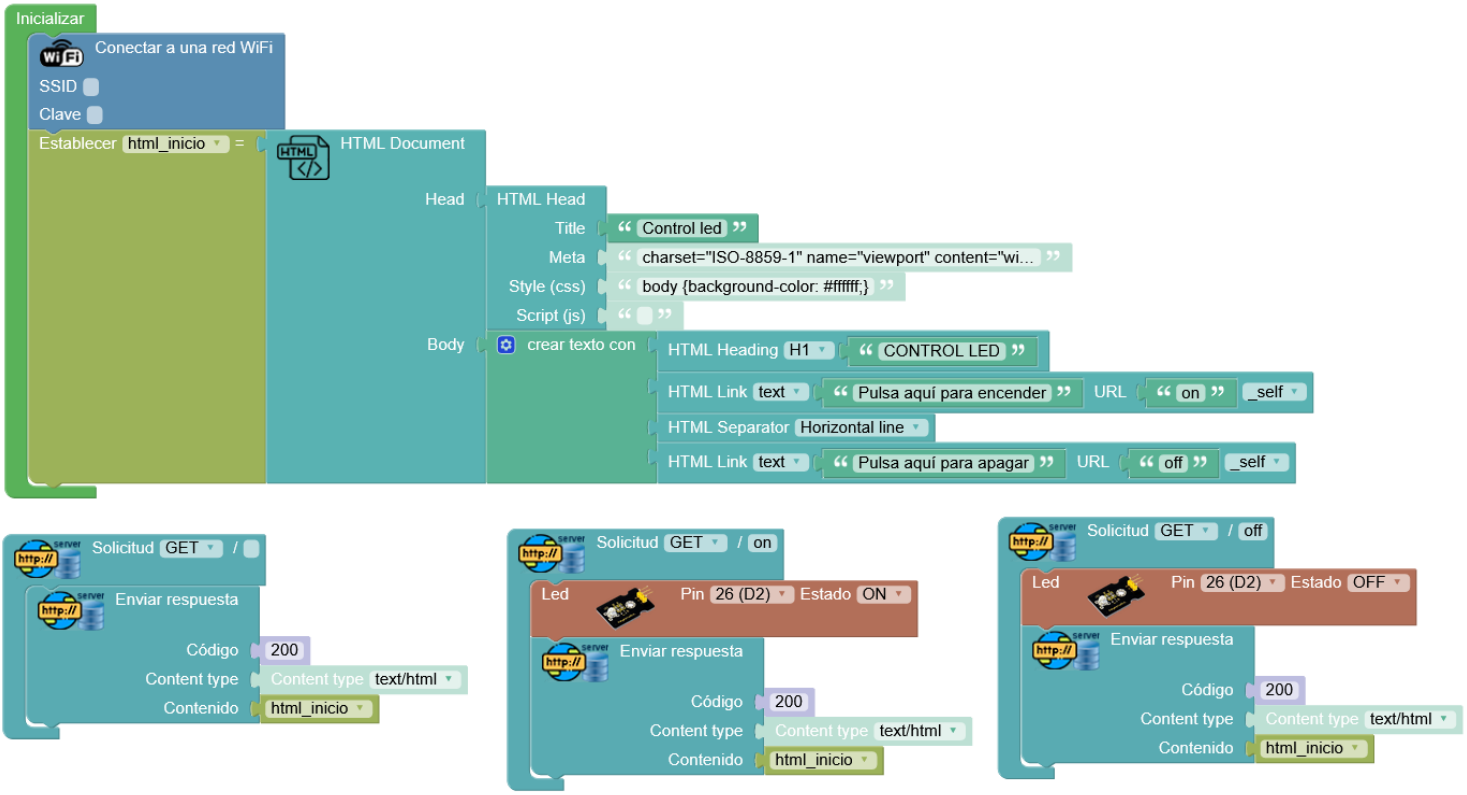


El contenido es simplemente texto formateado según el estándar HTML.

Podemos devolver respuestas en formato HTML en las peticiones del servidor HTTP, por ejemplo ésto generaría un documento HTML en caso de solicitar una web no encontrada, con un mensaje de ERROR en grande y color rojo.



En muchos casos, se devolverá la misma web para varias solicitudes. En ese caso podemos almacenar el contenido HTML en una variable de texto y devolverla en las distintas solicitudes del servidor HTTP:



- Cliente HTTP (Web)

Desde un cliente HTTP podemos hacer básicamente lo que hace el navegador web cuando hacemos una petición web.

En un dispositivo como la ESP32 STEAMakers la funcionalidad de cliente HTTP puede tener principalmente los siguientes usos:

-Hacer peticiones a un servicio web para actuar sobre el servidor ejecutando acciones remotas:

Siguiendo el ejemplo del servidor web anterior para encender o apagar un led, desde otro dispositivo ESP32 STEMakers podríamos actuar conectando con el cliente HTTP:

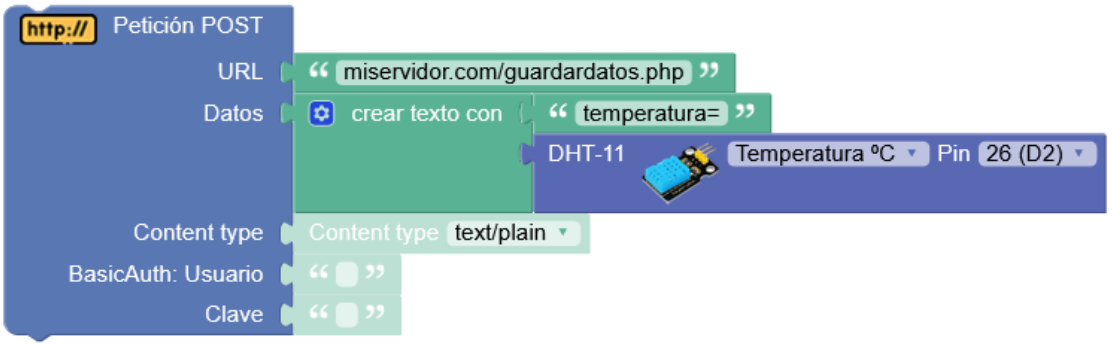
Si nos interesa la respuesta del servidor, podemos usar el bloque para obtener la cadena de texto con la respuesta (normalmente HTML, JSON, CSV, ….)

-Obtener información de servicios web remotos (APIs)

Ejemplo: obtener la información meteorológica en forma de texto para unas coordenadas en concreto

http://www.7timer.info/bin/astro.php?lon=-0.6&lat=38.6&ac=0&unit=metric&output=json&tzshift=0

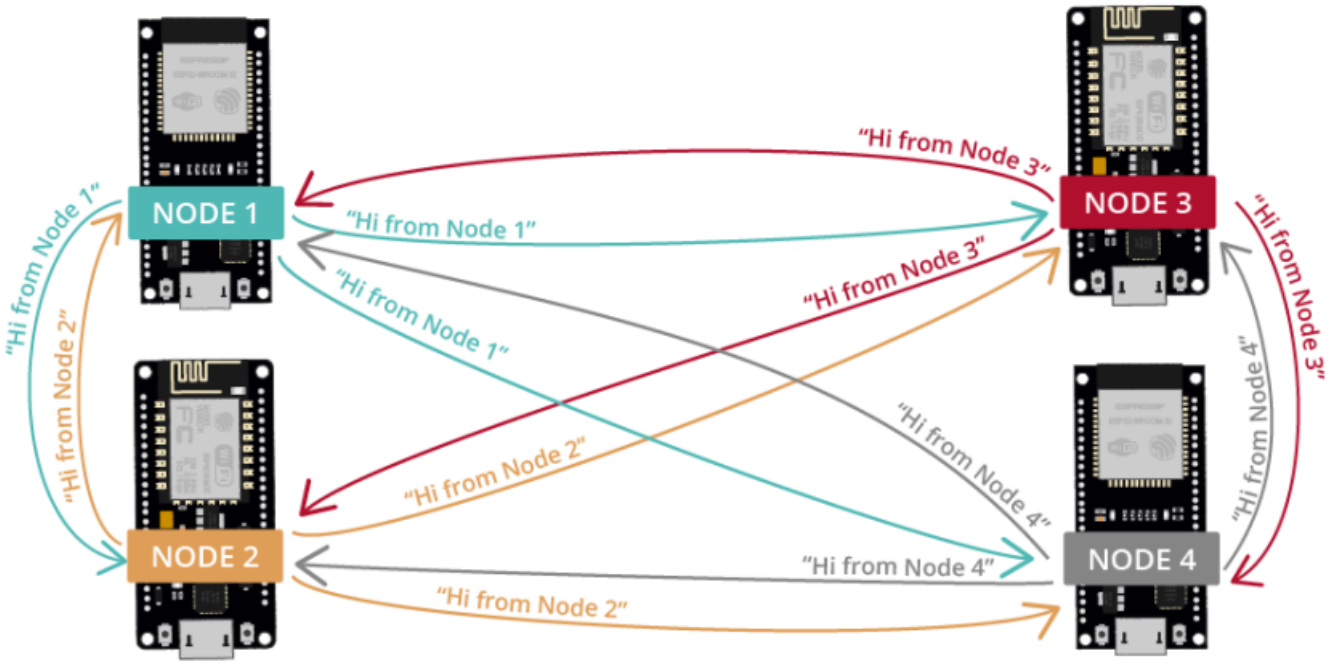
-Data logging remoto, enviando los datos como parámetros a páginas web que los recogen y almacenan en una base datos (aplicaciones php, …)



-Ciertos servicios como IFTTT , Thingspeak, … nos permiten enviar o ejecutar acciones a través de peticiones HTTP para lo que usaremos el cliente HTTP.

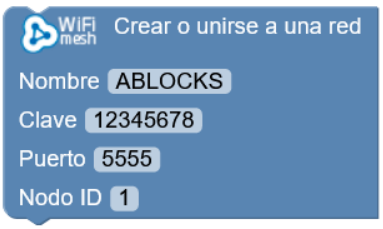
WifiMesh

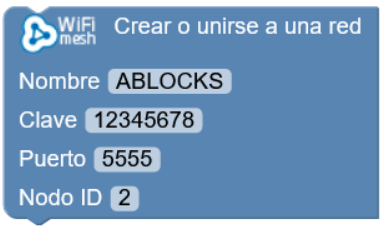
WifiMesh implementa una malla WiFi de dispositivos descentralizada, de forma que todas las placas ESP32 STEAMakers estarán interconectadas de forma sencilla y sin necesidad de una red WiFi previa o punto de acceso (la crean ellas mismas automáticamente). Cada dispositivo puede enviar mensajes al resto y recibir de cualquiera (¡hablan todas con todas!)

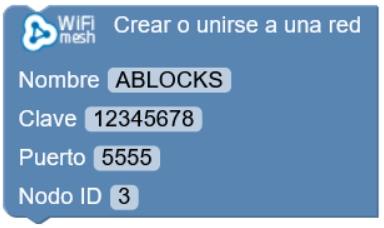


En ArduinoBlocks el proceso de intercambio de mensajes entre placas ESP32 STEAMakers es muy fácil de utilizar. No necesitamos una red previa donde conectarnos, los propios dispositivos crean un red propia entre ellos para intercambiar información.

Todos los dispositivos (nodos) que vayan a formar parte de la red Mesh (malla) deben compartir la misma configuración de nombre, clave de la red y puerto. Y cada uno un número de nodo diferente.

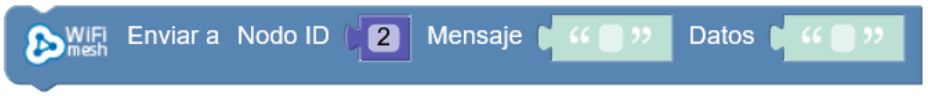




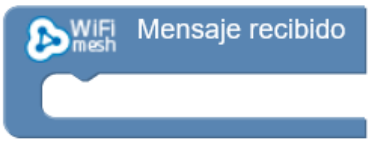


Enviar una mensaje a todos los nodos de la red: envía un mensaje y datos (opcionales) a todos los nodos conectados a nuestra red “*mesh*”

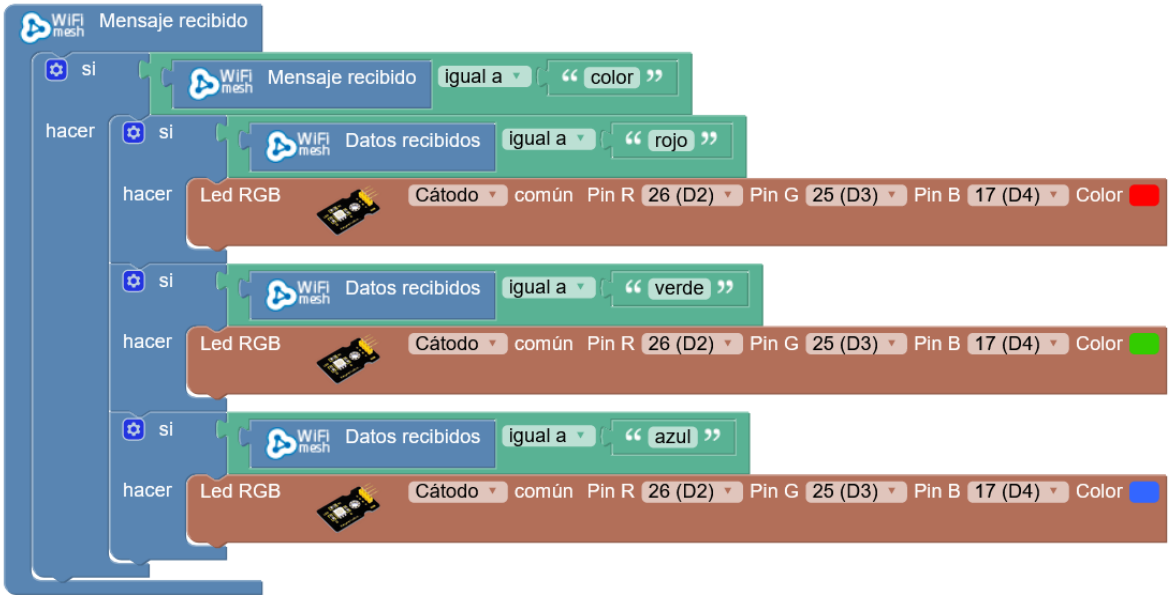
Si no queremos enviarlo a todos, podemos indicar un número de nodo en concreto:



Recibi mensajes de otros nodos (evento de recepción), este bloque recibe todos los mensajes y ya dentro podré determinar el mensaje y los datos recibidos:



Y si añadimos datos extras al mensaje, podemos recibirlos de igual forma:



Para facilitar, podemos usar el bloque de recibir mensajes en concreto:

O siguiendo con el ejemplo anterior: