Una introducción al Internet de las Cosas y los Sistemas Embebidos utilizando programación basada en bloques con BIPES y ESP8266 / ESP32

Traducción al español de la obra original:
“An Introduction to the Internet of Things and Embedded Systems using block-based programming with BIPES and ESP8266 / ESP32”
Autores: Rafael Vidal Aroca, Wesley Flavio Gueta,
Jorge André Gastmaier Marques, Tatiana F. P. A. Taveira Pazelli

1ª Edición (diciembre de 2021)

Traducción al español realizada por:

Henry B. Guerrero
y
Danna Celena Montenegro Orjuela

Basado en el proyecto académico de código abierto BIPES (bipes.net.br), con apoyo de la Universidade Federal de São Carlos (UFSCar) y CNPq/MCTI (Brasil).

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Una introducción al internet de las cosas y los sistemas embebidos utilizando programación basada en bloques con BIPES y ESP8266 / ESP32 [livro eletrônico] / Rafael Vidal Aroca ... [et al.] ; [tradução Henry B. Guerrero, Danna Celena Montenegro Orjuela]. -- 1. ed. -- São Carlos, SP : Rafael Aroca, 2025.
PDF

Título original: An introduction to the Internet of things and embedded systems using block-based programming with BIPES and ESP8266 / ESP32
Outros autores: Jorge André Gastmaier Marques, Wesley Flavio Gueta, Tatiana de Figueiredo
Bibliografia
ISBN 978-65-01-79539-3

1. Base de dados 2. Ciência da computação 3. Internet das coisas 4. Internet (Rede de computador) 5. Programação (Computadores) - Estudo e ensino 6. Python (Linguagem de programação para computadores) I. Aroca, Rafael Vidal. II. Marques, Jorge André Gastmaier. III. Gueta, Wesley Flavio. IV. Figueiredo, Tatiana de. V. Guerrero, Henry B. VI. Orjuela, Danna Celena Montenegro.

25-315587.0 CDD-004.678

Índices para catálogo sistemático:

1. Internet das coisas : Ciência da computação 004.678

Maria Alice Ferreira - Bibliotecária - CRB-8/7964

Contenido

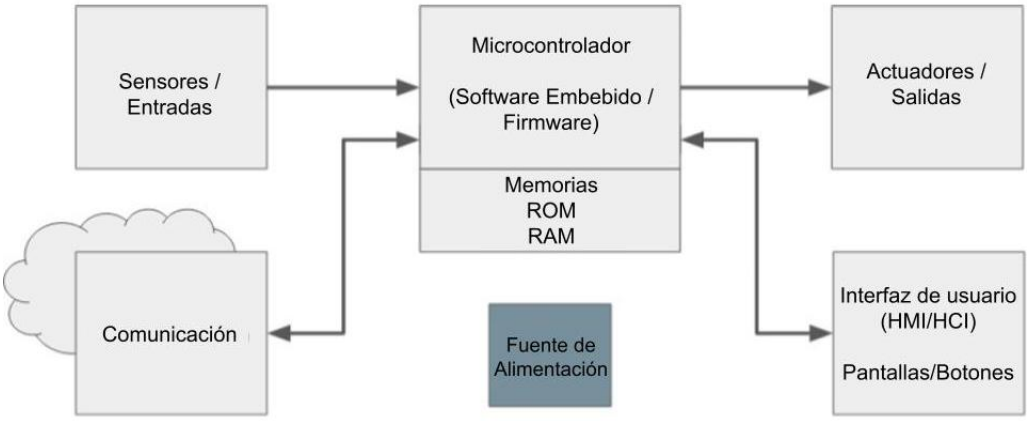
Internet de las cosas usando BIPES

Introducción rápida

BIPES: Plataforma Integrada basada en Bloques para Sistemas Embebidos (http://bipes.net.br) es una plataforma gratuita y de código abierto creada para la programación de sistemas embebidos y aplicaciones de Internet de las Cosas [1].

Los sistemas embebidos están presentes en todos los aspectos de la vida cotidiana: en el comercio, la industria, los automóviles, hornos, televisores, fotocopiadoras, puertas de garaje, alarmas, refrigeradores y otros dispositivos. Un sistema embebido está compuesto de hardware y software que realizan una tarea específica y está “embebido” en un producto. De ahí los nombres: sistema embebido, hardware embebido y software embebido. Existen varios dispositivos para implementar sistemas embebidos, como microprocesadores y microcontroladores. Los microcontroladores, en particular, son pequeños circuitos integrados (chips), usualmente de bajo costo, que requieren pocos componentes externos y pueden implementar una lógica de operación basada en entradas y otra información para controlar salidas y realizar otras acciones. Son una especie de computadora pequeña y completa diseñada para aplicaciones específicas. El software embebido, en general, es lo que determina la lógica de funcionamiento de un sistema embebido.

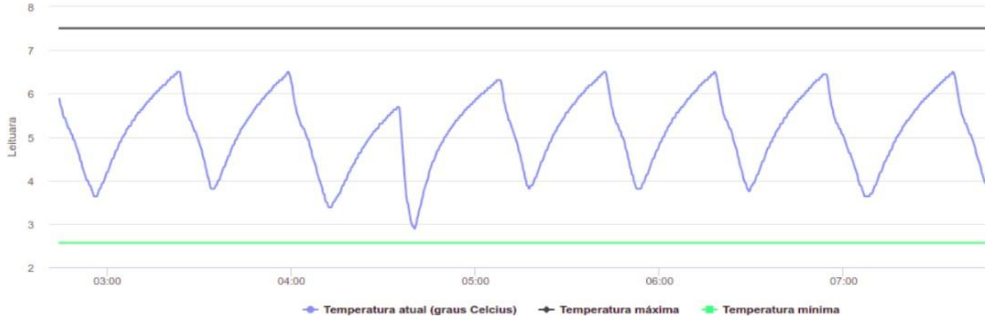
La figura a continuación muestra una visión general de un sistema embebido compuesto por un microcontrolador y varios otros dispositivos. Sin embargo, no todos estos componentes son necesarios para todas las aplicaciones. Por lo tanto, cada aplicación determinará la necesidad de sensores, actuadores y otros elementos en un sistema embebido.



- **Microcontrolador**: Es un componente electrónico, generalmente compuesto por puertos de entrada y salida para señales eléctricas, por memorias y un procesador capaz de ejecutar la lógica definida por el software o firmware embebidos.

- **Sensores**: Los sensores son dispositivos que detectan/miden el estado/nivel de una magnitud física específica del entorno, como la temperatura, la humedad, la iluminación, la vibración, el campo magnético, entre otros;
- **Actuadores**: Los actuadores son dispositivos que pueden cambiar el estado del entorno, como motores, calentadores, electroimanes y otros dispositivos que, mediante comandos de microcontrolador, pueden actuar sobre el entorno. La mayoría de los microcontroladores no pueden accionar un actuador directamente, por lo que deben utilizarse circuitos de potencia (drivers), que pueden incluir transistores o puentes H, para que los microcontroladores puedan accionar los actuadores;
- **Comunicación**: Algunos sistemas embebidos pueden incluir mecanismos de comunicación, lo que les permite enviar datos a sistemas de monitoreo remotos o recibir comandos a distancia. Este tipo de conectividad puede ser alámbrica o inalámbrica, y puede o no contar con una conexión a Internet;
- **Fuente de alimentación**: Un sistema embebido requiere una fuente de energía para funcionar. La energía puede obtenerse de la red eléctrica, baterías, paneles solares, entre otras fuentes de energía eléctrica. Una tendencia en algunos sistemas embebidos y sistemas del Internet de las Cosas son los sistemas que recolectan energía del entorno, por ejemplo, de ondas de radio;
- **Interfaz de usuario**: Varios sistemas embebidos necesitan interactuar con el usuario, ya sea a través de pantallas, botones, teclados, mensajes de voz, comandos de voz o incluso mediante interfaces de control y visualización basadas en teléfonos inteligentes.

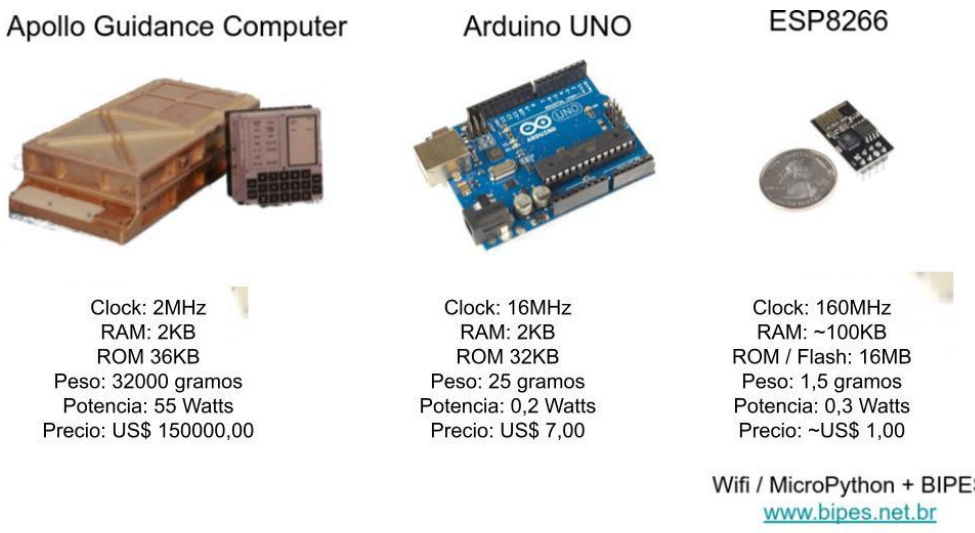
Considerando estas definiciones, la figura a continuación muestra un ejemplo del resultado de la operación de un sistema embebido compuesto por un microcontrolador, un sensor de temperatura y un compresor (actuador) que controla la temperatura interna de un refrigerador. El sensor de temperatura y el microcontrolador cuestan menos que una hamburguesa, y es posible implementar distintas lógicas y funcionalidades. En esta figura, se puede observar que cada vez que la temperatura supera los 6 grados Celsius, ocurre algo y la temperatura comienza a descender. El sistema embebido detecta que la temperatura ha excedido los 6 grados Celsius y activa el compresor del refrigerador, el cual enfría el ambiente. En consecuencia, el sensor de temperatura también detecta este cambio en el entorno. Al alcanzar la temperatura de 4 grados, el actuador se apaga. Así, en esta aplicación, el ciclo se repite en una acción de control conocida como “on-off”, siguiendo la lógica implementada en el software embebido.



Este sistema también podría estar conectado a Internet, permitiendo que la configuración de temperatura deseada sea monitoreada y ajustada de forma remota mediante un teléfono inteligente. Además, el sistema podría emitir alertas y alarmas en caso de fallo del compresor o cuando se alcance un umbral de temperatura inapropiado. Esta asociación de un sistema embebido, con entradas del usuario y monitoreo a través de Internet, es un ejemplo de un sistema de ***Internet de las Cosas*** (IoT). Así, miles de dispositivos están conectados a Internet, como bombillas, medidores inteligentes, refrigeradores, aires acondicionados, asistentes domésticos y otros.

Aun así, es impresionante analizar los avances de los últimos años en el contexto de los sistemas embebidos. Este libro presenta actividades prácticas utilizando el módulo ESP8266 o el ESP32, ambos de Espressif Systems. Estos módulos, clasificados como *Sistemas en Chip* (SoCs), son computadoras completas que incluyen procesador, memoria RAM, memoria FLASH para almacenar programas, conexión inalámbrica WiFi, entradas para sensores, salidas para actuadores y otras diversas características. ¡Y cuestan menos que una hamburguesa! Si lo piensas, son mucho más potentes que la computadora que llevó al Módulo Lunar Apolo a la superficie de la Luna. Consulta la siguiente figura para una comparación.

Aunque la figura no muestra el módulo ESP32 en la comparación, este dispositivo es más potente que el ESP8266 y cuenta con varias características adicionales, incluyendo un procesador de doble núcleo, Bluetooth, mayor memoria y capacidad de procesamiento, entre otras funciones. Es importante señalar que todas las explicaciones y actividades presentadas en este libro están ilustradas con ejemplos para el ESP8266, los cuales funcionan igualmente en el ESP32, aunque ocasionalmente presentan algunas diferencias, como el cambio en los números de pines debido a que ambos módulos tienen configuraciones distintas de entradas y salidas de pines (terminales de conexión).



BIPES (http://bipes.net.br), se utiliza para todas las actividades propuestas en este libro. Versátil, abierta y gratuita, la plataforma BIPES permite programar utilizando varias placas microcontroladoras como ESP32, ESP8266, Arduino, mBed, entre otras. Además, **BIPES** puede usarse directamente en navegadores web para múltiples dispositivos y no

requiere la instalación de ningún software, complemento ni la configuración de algún parámetro en particular. Además de facilitar el desarrollo mediante la programación por bloques, **BIPES** también ofrece una plataforma de Internet de las Cosas, que permite la creación de paneles de visualización y control remoto (**dashboards / paneles de control**) para aplicaciones de IoT [2].

Aunque la programación basada en bloques puede parecer enfocada en aplicaciones educativas debido a su facilidad, ha sido adoptada y utilizada cada vez más en entornos comerciales e industriales. Ya se han notado varias ventajas: reducción del tiempo de desarrollo, disminución de fallos lógicos, mayor facilidad de comprensión y un mantenimiento más rápido y sencillo. Una publicación del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) difundió un artículo en el que se habla sobre el nacimiento de la era de la programación sin código [3].

Como se mencionó, esta es una introducción rápida al tema. Hay varias publicaciones de calidad disponibles para que puedas profundizar en este asunto. El objetivo aquí es guiarte a través de actividades rápidas y sencillas para implementar sistemas IoT de manera práctica.

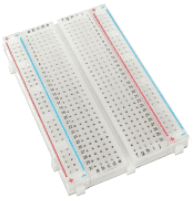
Referencias:

[1] AGDS Junior, LMG Gonçalves, GA de Paula Caurin, GTB Tamanaka, AC Hernandes, RV Aroca. BIPES: Block Based Integrated Platform for Embedded Systems. IEEE Access, v. 8, p. 197955-197968, 2020.
Disponible en: <https://ieeexplore.ieee.org/document/9246562>.

[2] CA Silva. Desenvolvimento e validação de módulo de comunicação MQTT para plataforma BIPES para aplicações de Internet das Coisas. 2020. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) - Universidade Federal de São Carlos, 2020.
Disponible en: <https://repositorio.ufscar.br/handle/ufscar/13656>.

[3] RD Caballar. Programming without code: The rise of no-code software development. IEEE Spectrum. Tech Talks, Nov. 2020, Disponible en:
<https://spectrum.ieee.org/programming-without-code-no-code-software-development>

Lista de materiales

Artículo		Descripción
1		Cualquier placa con un módulo ESP8266 o ESP32. Específicamente, todos los ejemplos de este libro utilizaron el módulo Wemos D1 Mini, basado en el SoC ESP8266.
2		Sensor de Humedad y Temperatura DHT11
3		Sensor de resistencia dependiente de la luz (LDR)
4		Placa de prototipos - protoboard
5		Conector eléctrico de CA
6		Placa con relé
7		Receptáculo de enchufe eléctrico de CA

8		Adaptador de enchufe de CA para bombilla/lámpara
9		Terminales
10		*Cables jumper*
11		*Batería externa (Power bank)*
12		Bombilla 127 Voltios CA
13		Potenciómetro
14		Cable paralelo (blanco) para conexión a la red eléctrica

15		Interruptor (botón) / Botón pulsador
16		Micro servo motor / RC Hobby Servo
17		Resistencias de diferentes valores: 1K, 570 ohmios y 220 ohmios
18		*Buzzer (emisor de sonido piezoeléctrico)*
19		Diodo emisor de luz (LED)
20		Diodo emisor de luz (LED) RGB de 3 colores Opcional: solo para la actividad “Cambiando el color del gallo del clima”
21		Shield de matriz LED con controlador TM1640 Opcional: solo para la segunda parte de la actividad “Cambio de color del gallo del clima”

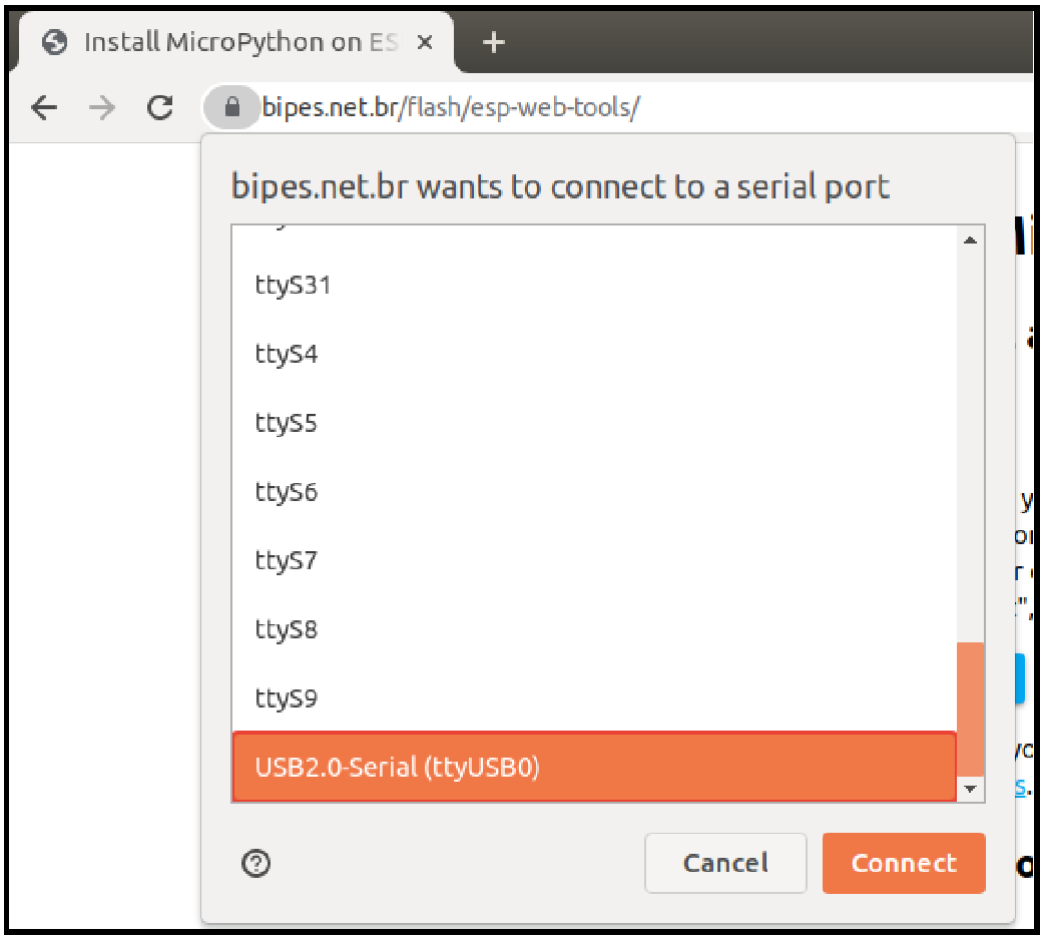
Preparación de la placa ESP8266 o ESP32

Varias placas pueden realizar las actividades propuestas en este texto. Sin embargo, en esta ocasión nos centraremos en el uso de la placa con el módulo ESP8266. Además, debe instalarse MicroPython (https://micropython.org/) en esta placa, lo que permite programar, depurar y gestionar aplicaciones de manera práctica y dinámica, con una implementación minimalista del lenguaje de programación Python. Utilizando los recursos del proyecto BIPES, instalar MicroPython en el ESP8266 es bastante sencillo. Los siguientes pasos funcionan en los navegadores Google Chrome y Microsoft Edge en entornos MAC, Linux y Windows.

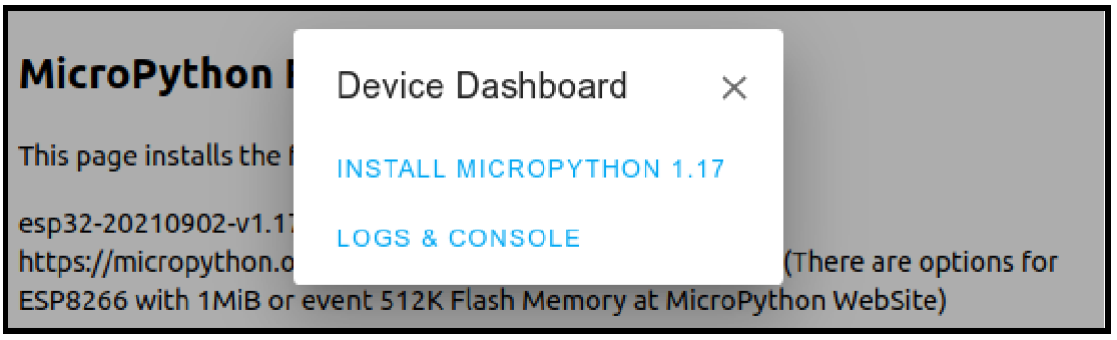
Para instalar MicroPython en el ESP8266:

1. Ir a: https://bipes.net.br/flash/esp-web-tools/.
2. Conecta la placa ESP8266 al puerto USB de la computadora.
3. Clic en **CONNECT**.

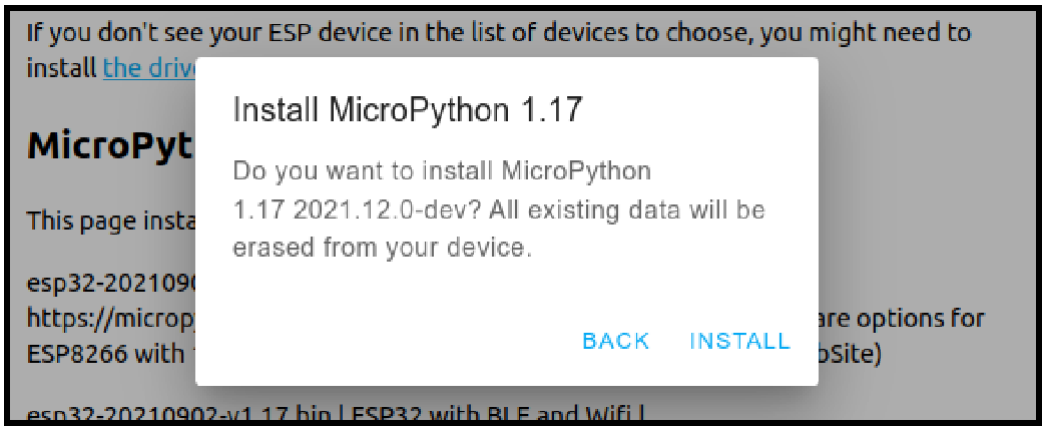
4. Seleccione el puerto serie (COM) utilizado por su dispositivo y haga clic en **Connect**:



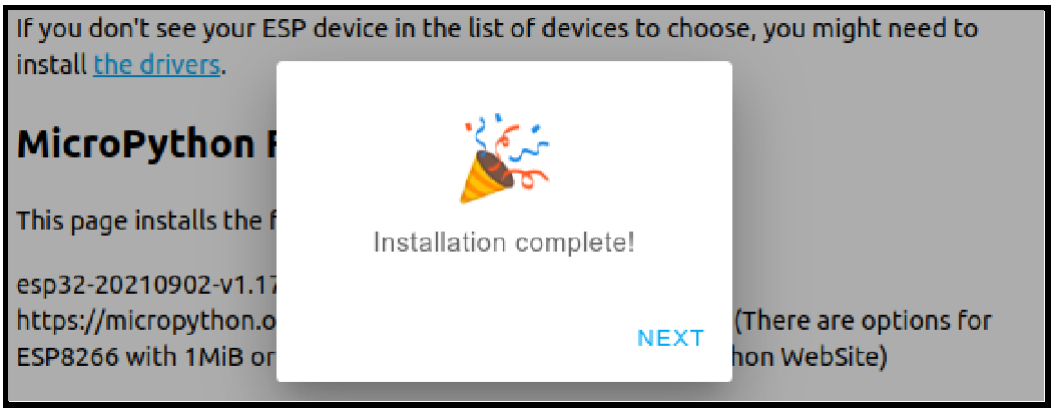
5. Ahora haga clic en **INSTALL** MICROPYTHON:



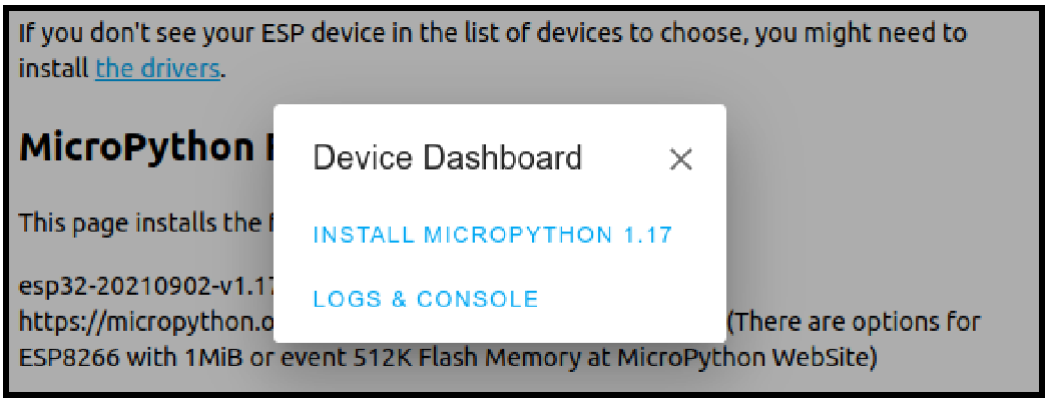
6. Entonces haga clic en **INSTALL**:



7. Después de la instalación, verás el mensaje:



8. Clic en **NEXT** y luego en **LOGS & CONSOLE**.



Observe el comportamiento de la placa en la consola. Por defecto, MicroPython crea una red “MicroPython” para la conexión remota y la programación mediante WiFi. Sin embargo, algunas placas genéricas ESP8266 tienen circuitos de alimentación que suministran una corriente eléctrica insuficiente para satisfacer la demanda de energía cuando

la placa ESP8266 inicia el servicio de red WiFi Access Point. Como resultado, la placa puede comenzar a reiniciarse de manera continua (como se muestra en la imagen a continuación).

Dependiendo de tu placa ESP8266, este problema no ocurrirá. De todas formas, si sucede, la solución es desactivar el modo *"Access Point"*. Para hacerlo, copia la siguiente línea y pégala en la terminal/consola inmediatamente después de que la placa ESP8266 reinicie. Deberás estar atento al momento del reinicio y escribir el comando justo después del reinicio. De lo contrario, la placa seguirá reiniciándose continuamente.

Comando que debe copiarse y pegarse inmediatamente después de que la placa se reinicie:

```
import network; network.WLAN(network.AP_IF).active(False)
```

Con MicroPython instalado y estable, puedes proceder a los siguientes pasos.

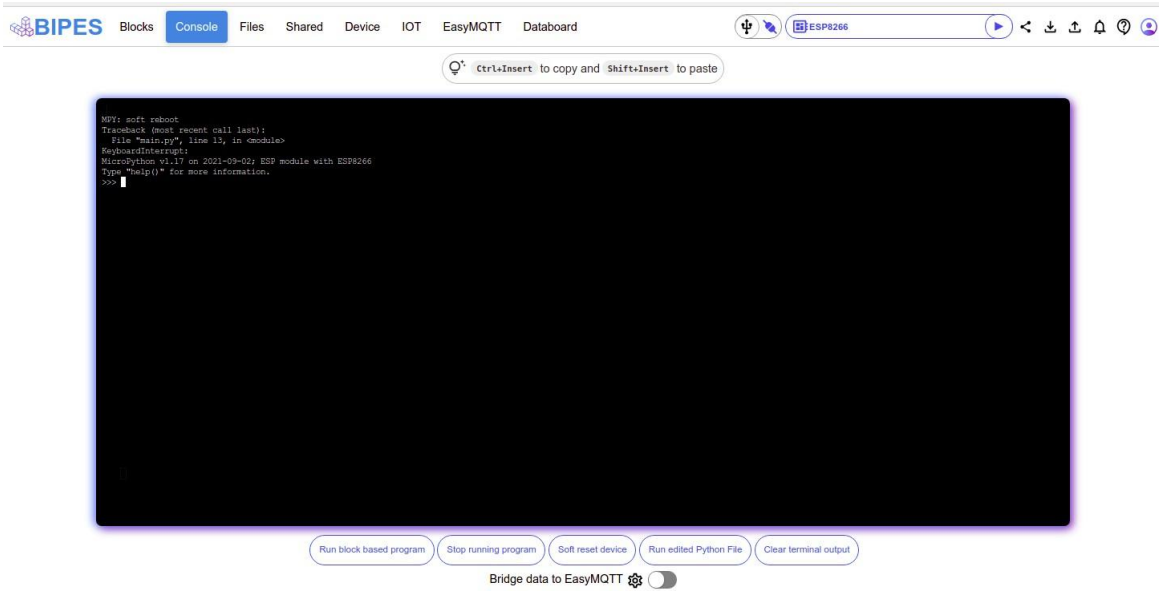
Nota: Todas las actividades y comandos mencionados en este texto funcionarán en ESP8266 y ESP32.

Parpadeando el LED integrado

El diodo emisor de luz, o LED, es una luz indicadora que se utiliza en muchas situaciones para señalar el estado de un sistema embebido. Por ejemplo, en un módem, puede mostrar si el Internet está conectado. Por ello, “parpadear el LED” es una de las primeras actividades que se realizan para probar o aprender sobre una nueva placa o dispositivo utilizado para construir un sistema embebido. En esta sección se realizará una actividad paso a paso para hacer parpadear el LED integrado en la placa ESP8266. En otras palabras, no es necesario conectar ningún LED externo. Simplemente use el que ya viene incorporado en la placa.

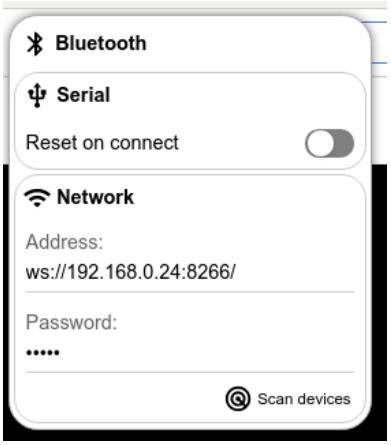
1. Conecta el módulo ESP8266 al puerto USB de la computadora.
2. Accede al entorno de desarrollo **BIPES**: https://bipes.net.br/ide/.
3. Elige, en la esquina superior derecha, la opción **ESP8266**:

4. Accede a la pestaña **Console**:

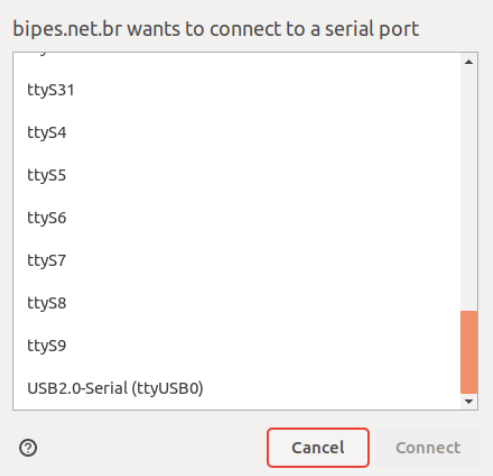


5. Haz clic en el icono de conectar: 

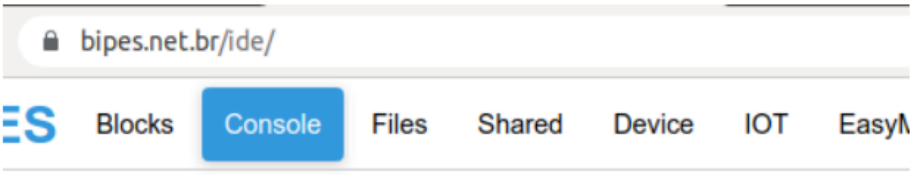
6. Haga clic en **Serial**:

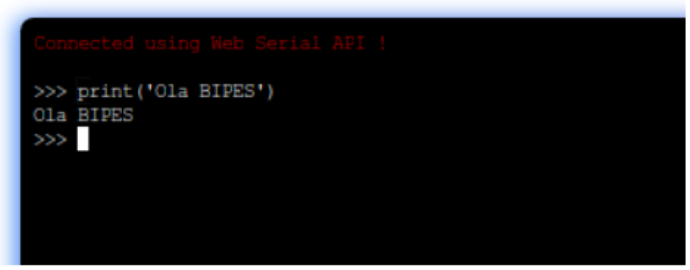


7. Elige el puerto asociado con la placa y haz clic en **Connect**:

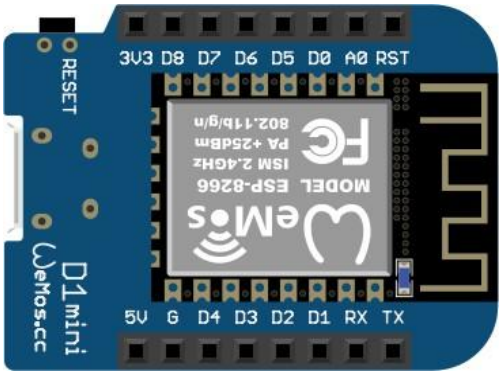


8. Comprueba si la conexión fue exitosa enviando comandos a la placa. Por ejemplo, intenta escribir *help()* o *print("Hello BIPES")*. Si la placa no responde, también puedes intentar reiniciarla presionando el botón de **RESET** o retirándola del puerto USB y volviéndola a conectar (regresa al paso 5).

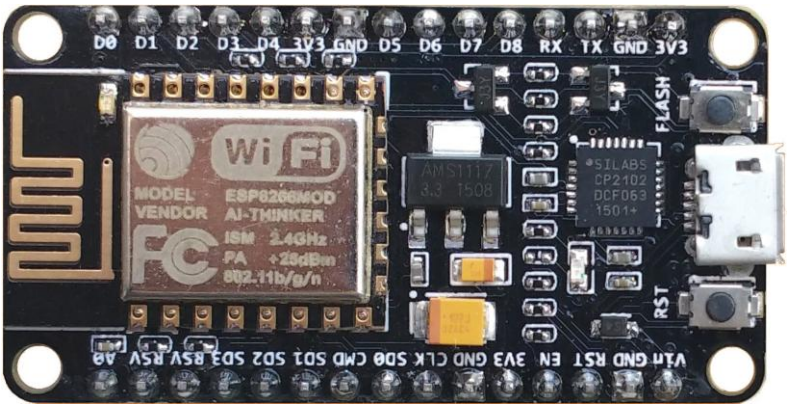




Después de conectar, verifica que cada placa tenga un **mapa de pines**. En particular, la luz indicadora LED interna de la placa, que queremos utilizar, está conectada al pin D4 / GPIO2 en la mayoría de las placas ESP8266, pero revisa el mapa de pines de tu placa para estar seguro. La figura a continuación muestra un ejemplo de un mapa de pines de la placa WeMos D1 mini con el módulo ESP8266:

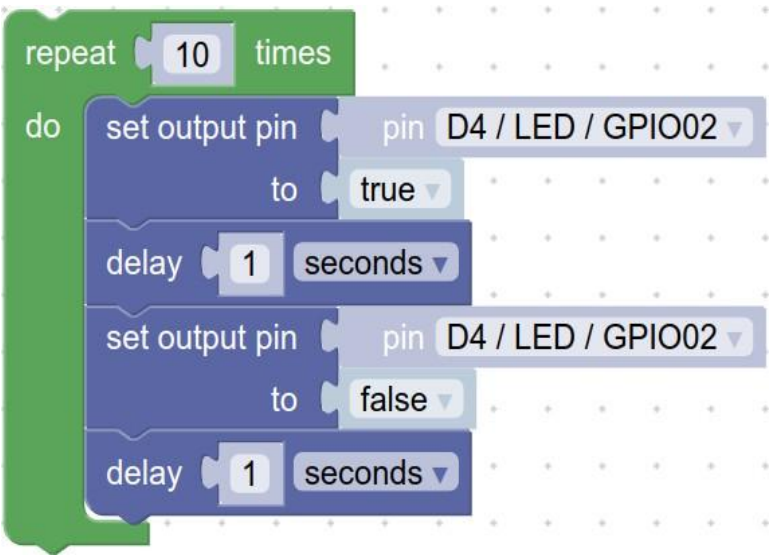


En la imagen de arriba, observa que cada terminal para conectar componentes externos tiene una etiqueta, como D8, D7, A0, D2, etc. El pin 5V proporciona 5 voltios para los circuitos externos, y el 3V3 proporciona 3,3 voltios. El pin G es la tierra (Ground) del circuito. Ten en cuenta que existen docenas de modelos/opciones de placas con el módulo ESP8266, por lo que necesitarás identificar los pines/LEDs y conexiones de tu placa. La figura de abajo muestra otra opción de placa con el módulo ESP8266, comúnmente llamada NodeMCU o “DevKit” por los proveedores chinos. Esta placa, en particular, tiene dos LEDs: uno en el GPIO2 y otro en el GPIO16.



9. Prepare el siguiente programa utilizando **BIPES**, como se muestra a continuación.

Tip: Arrastra y suelta los bloques disponibles en bucles **(Loops)**; Temporizador **Timing** y Maquina **Machine** → In/Out Pins; recuerda siempre incluir un retardo **(delay)** para que puedas ver el cambio de estado del LED entre cada acción. Sin delay, el LED parpadeará tan rápido que será imposible verlo parpadear con nuestros ojos.



Nota: Debido a las características del circuito interno de la placa WeMos D1 mini, el LED interno de esta placa se enciende con un nivel lógico bajo (0 / falso) y se apaga con un nivel lógico alto (1 / verdadero). Esta característica puede variar de una placa a otra.

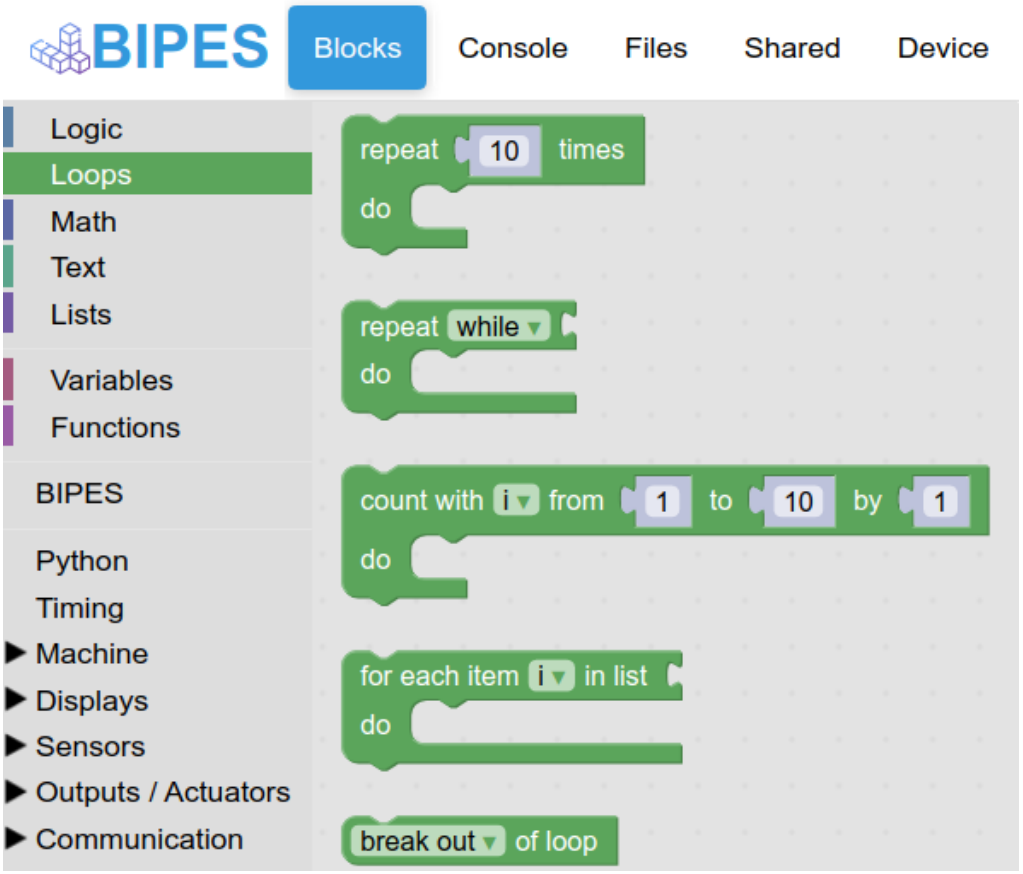
10. Después de preparar el programa, haz clic en el icono para ejecutarlo (**Símbolo play**):

Puedes verificar el programa que se está transfiriendo a la placa a través de la **consola** si lo deseas. El programa hará que el LED parpadee diez veces y luego finalizará.

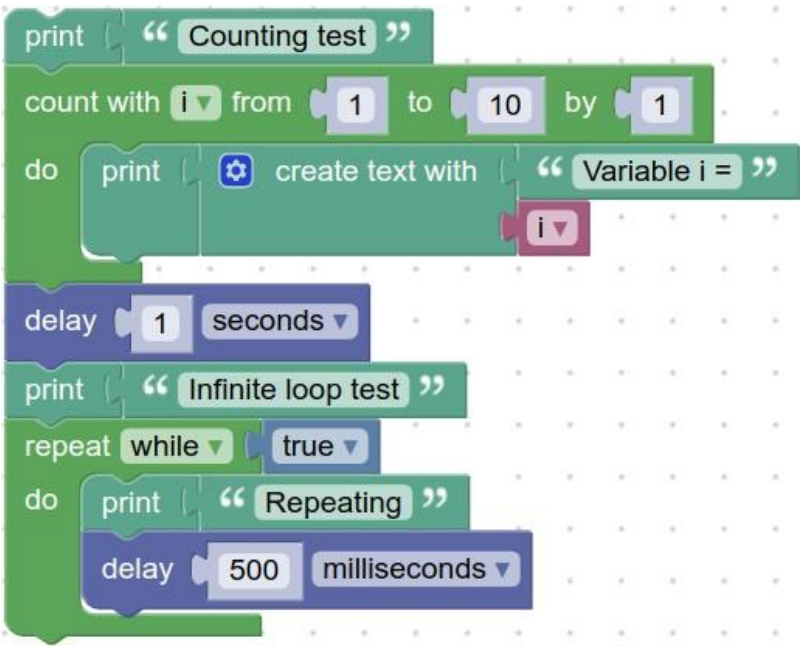
Actividad:
Cambia el programa para que el LED parpadee con diferentes intervalos de tiempo y un número distinto de veces.

	Tip: Utiliza el icono de usuario en la esquina superior derecha de la pantalla para elegir el idioma de la interfaz de BIPES (inglés u otro).

Hemos utilizado el bloque "**repeat 10 times / repetir 10 veces**", disponible en la sección de la caja de herramientas a la izquierda: **Loops / Bucles**. Sin embargo, la misma sección de herramientas también ofrece otras opciones de estructuras de repetición, como "**count from 1 to 10 / contar de 1 a 10**" y "**repeat while true / repetir mientras sea verdadero**" (repetir infinitamente). De hecho, puedes hacer clic en los números y cambiar los valores como desees.

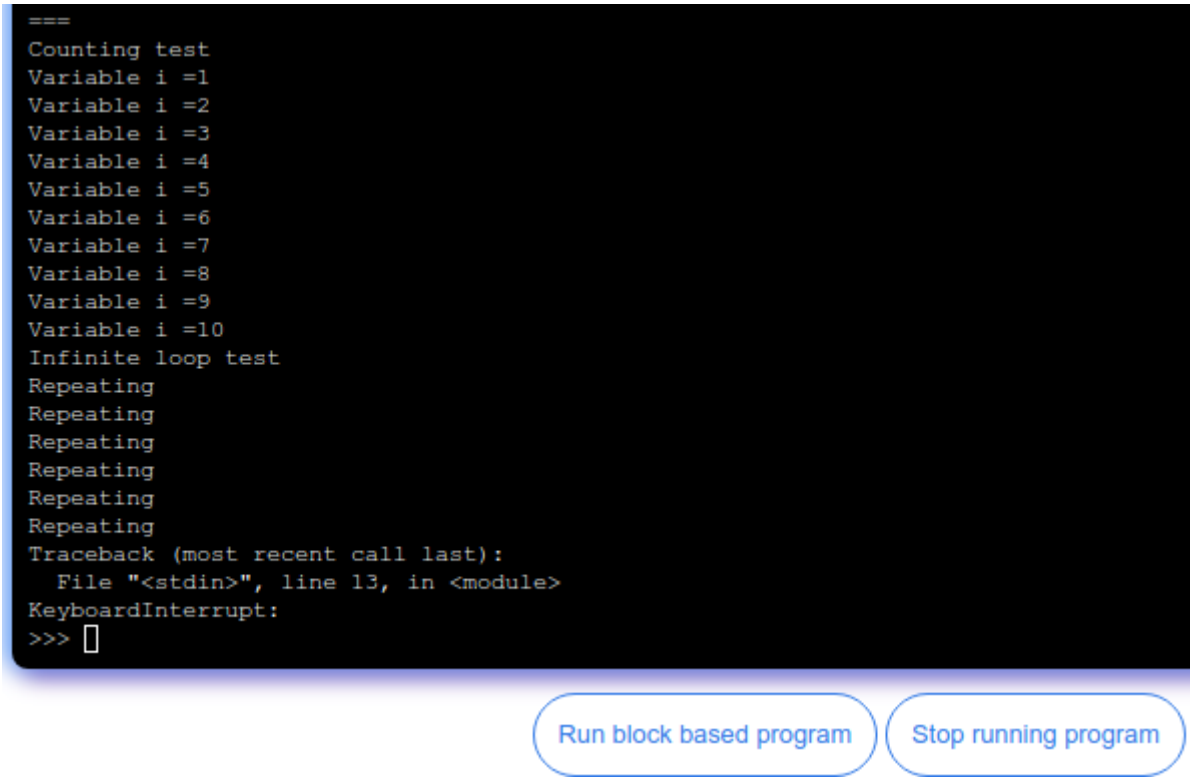


En el caso de contar del 1 al 10, cada iteración del bucle actualiza la variable “i” (un espacio de almacenamiento para datos del programa), la cual puede usarse dentro de la estructura de repetición. En el siguiente ejemplo, el programa imprime en la **Consola** del 1 al 10 con un texto inicial “**i =** ” para mostrar el nombre de la variable. A continuación, se muestran ejemplos de conteo y repetición infinita:

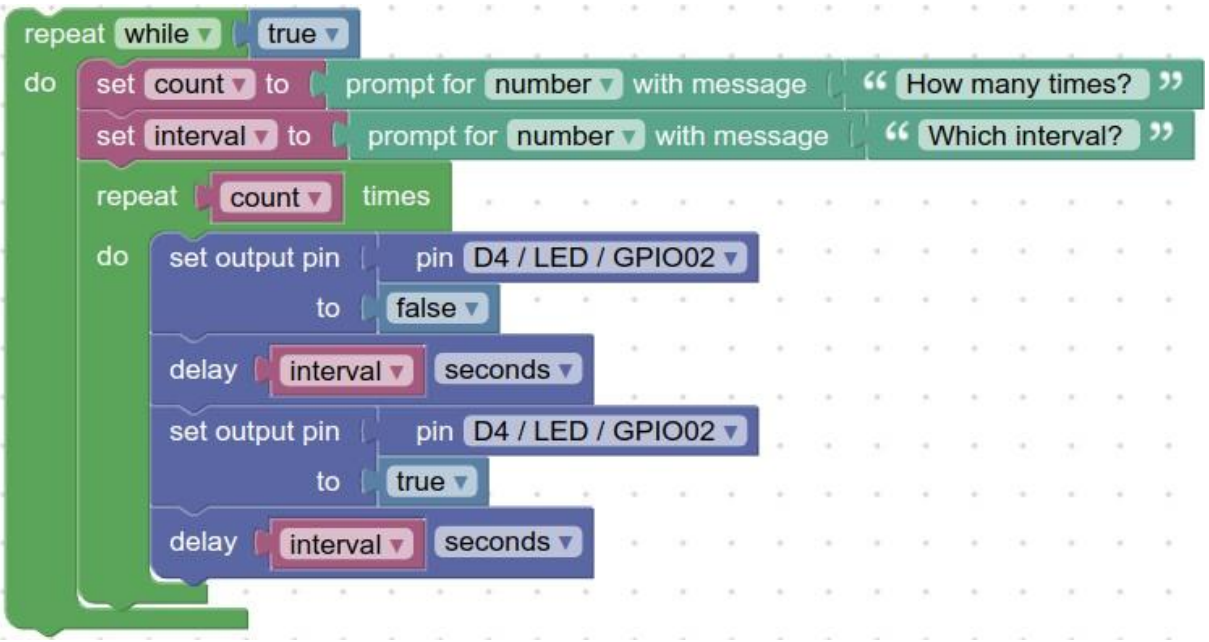


La estructura de repetición infinita “**repeat while true / repetir mientras sea verdadero**” es una de las estructuras de repetición más utilizadas en sistemas embebidos, ya que muchos de estos sistemas realizan una secuencia de operaciones de manera continua y repetitiva mientras el sistema está encendido. Por ejemplo: leer datos de un sensor, verificar la temperatura de un ambiente y encender o apagar el compresor de un refrigerador, entre otros. Sin embargo, es importante incluir un método para salir del bucle cuando se cumpla una condición. Esto se puede hacer con el bloque “**Loops >> break out of loop / bucle >> salir del bucle**”, que permite “salir” del bucle o entrar en otra rutina sin necesidad de presionar directamente el botón de reinicio o ejecutar CTRL+C desde la consola. De cualquier manera, el comando CTRL+C en la consola o el botón “**Stop Running Program / Detener el programa**” detendrán un bucle infinito.

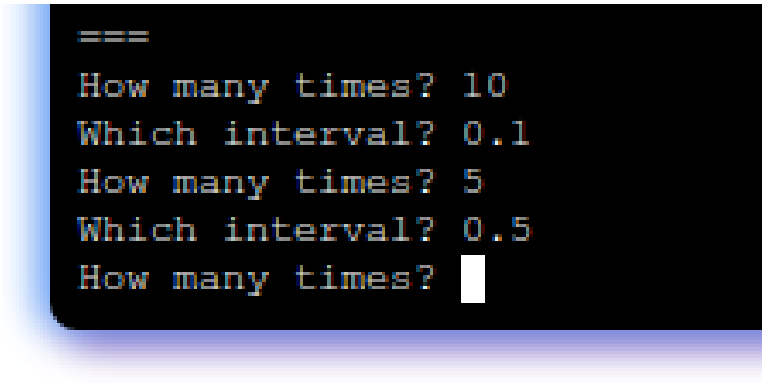
El resultado se puede ver en la **Consola** (haz clic en “**Stop / Detener**” para salir del bucle infinito).



Otra función útil asociada con la **Consola** es solicitar datos al usuario a través de la **Consola** (pedir información al usuario). El bloque “**prompt for number**” puede utilizarse para este propósito y se encuentra en el área de texto de la Barra de herramientas. Así, el programa a continuación presenta una versión interactiva que hace parpadear el LED. En esta versión, el programa pregunta cuántas veces quiere el usuario que parpadee el LED y el intervalo de tiempo con el que el LED alternará su estado.

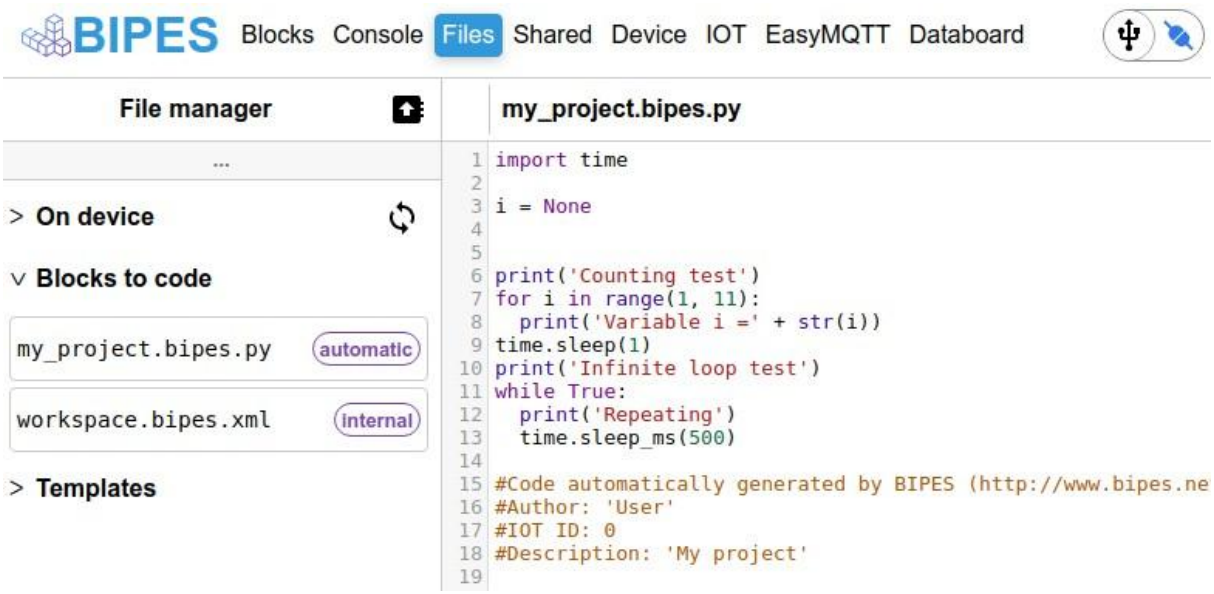


La siguiente figura muestra un ejemplo de cómo usar dicho programa, donde el usuario escribió 10 como número de veces y 0.1 como intervalo de tiempo. Después de que el programa ejecuta la secuencia de parpadeos del LED 10 veces, el programa vuelve a pedir las opciones al usuario. La segunda vez, el usuario ingresó 5 y 0.5.



Para los curiosos:

Verifica que este programa basado en bloques haya generado el código fuente en lenguaje Python (disponible en la pestaña Files de BIPES). En esa pestaña, puedes ver el programa generado automáticamente en la pestaña **Files >> Blocks to code**:



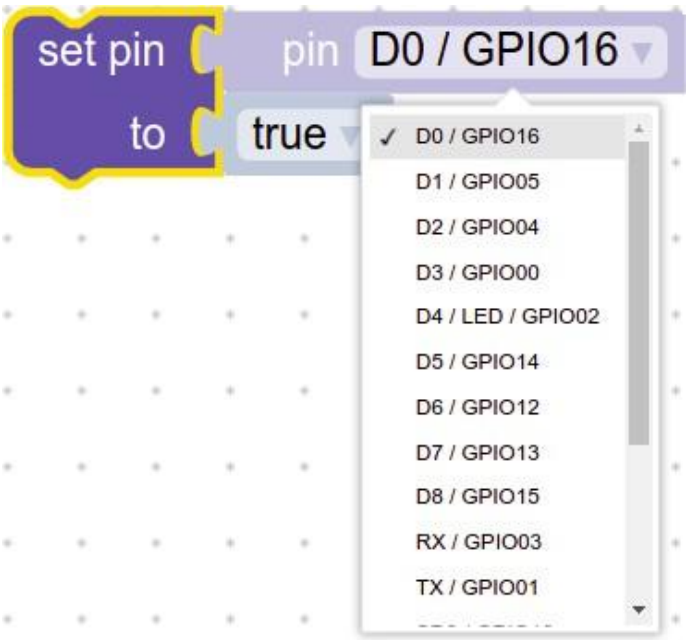
Avanzado:

Si lo deseas, puedes editar el archivo en Python, guardarlo (botón "**Guardar una copia / Save a copy**") y ejecutar el programa editado en forma de texto. Como se mencionó, la placa ESP8266 utiliza MicroPython, una versión de Python optimizada para sistemas embebidos y microcontroladores. Además, puedes guardar este archivo como "**main.py**" desde la pestaña Archivos de BIPES (botón **Guardar una copia / Save a copy**).

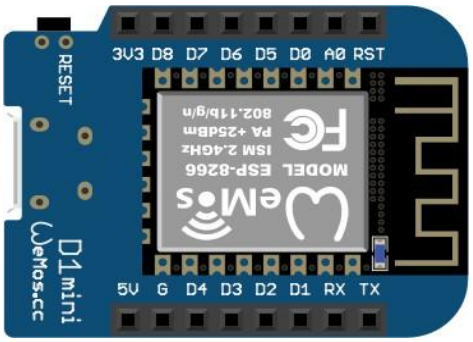
El programa "**main.py**" se ejecutará automáticamente cada vez que se encienda la placa, de forma autónoma e independiente. En otras palabras, tu solución quedará guardada e integrada en la placa y no dependerá de BIPES ni de la computadora para funcionar. Si quieres probarlo, puedes conectar la placa a una batería externa (power bank) o a un adaptador de corriente enchufado a la red eléctrica y verificar que el programa/sistema funciona de manera autónoma.

Entrada digital y comprobación de una condición

Los microcontroladores, en general, tienen dos tipos principales de entradas: digitales y analógicas. Las entradas suelen estar asociadas a los pines de ***Propósito General de Entrada/Salida*** **/ General Purpose Input Output** (GPIO), que se indican en la placa como GPIOX, donde X es el número del pin con códigos como D0 o D1. Por lo tanto, al seleccionar el modelo de placa en BIPES, los pines de esta placa se mostrarán automáticamente para su elección, facilitando la programación, como se muestra en la figura a continuación:



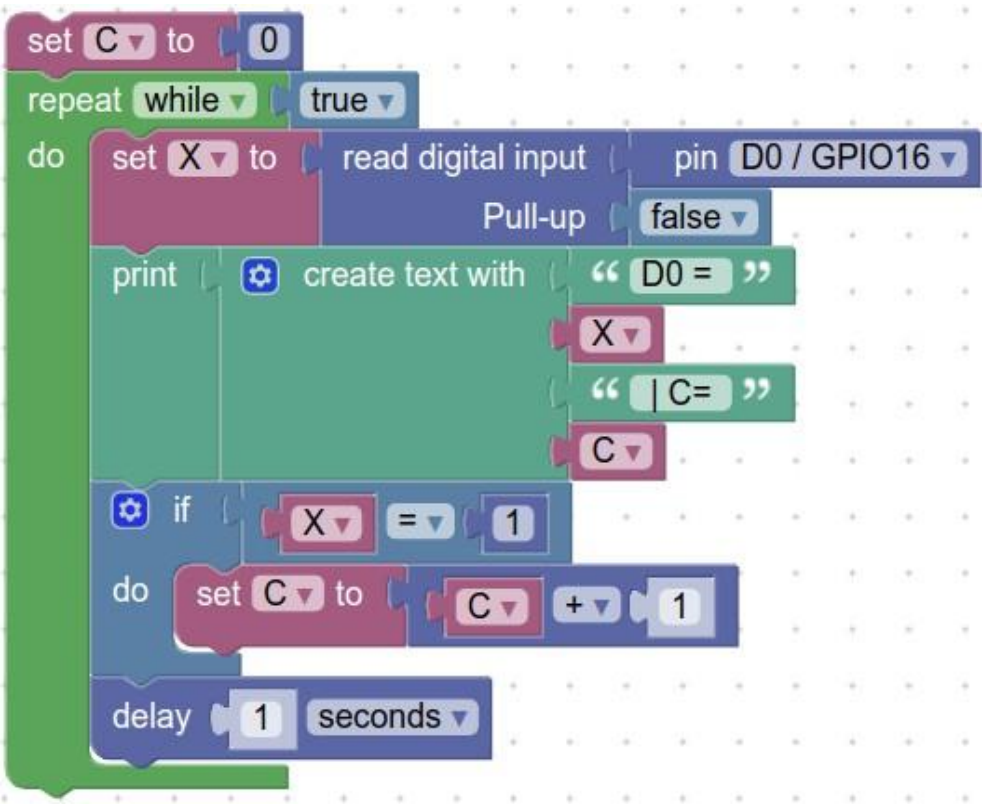
Tenga en cuenta que esta lista de pines estará asociada con los pines físicos de la placa:



En el caso del modo digital, estos pines pueden utilizarse para controlar dispositivos de salida y pueden asumir dos estados: [on / verdadero / **1**] o [off / falso / **0**]. Así, diferentes dispositivos pueden conectarse y controlarse mediante estos pines. Los mismos pines también pueden utilizarse en modo de entrada, para “leer” señales de entrada digital: [on / verdadero / **1**] o [off / falso / **0**].

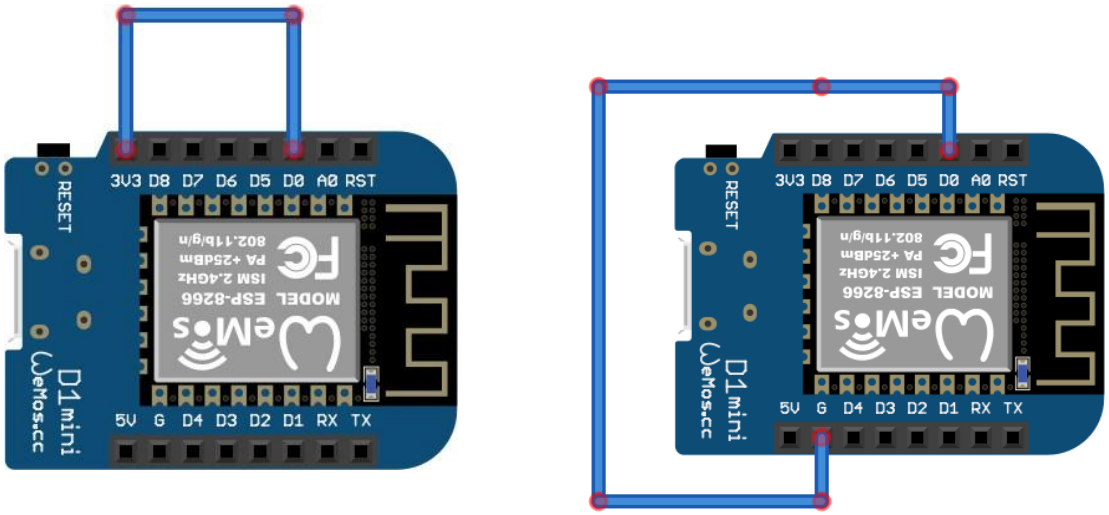
Dado que el módulo ESP8266 funciona con 3.3 voltios, una diferencia de potencial de 3.3V entre el pin referido y la referencia (G) representa un valor lógico 1, mientras que una diferencia de potencial cero (0V) representa un valor lógico 0.

El siguiente ejemplo muestra un programa que monitorea la entrada digital y verifica una condición (**IF**) para que una variable (espacio de almacenamiento de datos del programa) incremente su valor en una unidad cada segundo SI el estado físico de este pin es 1 (activo). El programa comienza estableciendo el valor de la variable C, un contador temporal, en 0 [**set C to 0 / inicializar C en 0**]. Luego, el programa continúa con un **infinite loop / bucle infinito [repeat while true]**. Dentro de esta estructura de repetición, la variable X recibe la lectura del pin digital D0 [**set X to read digital pin D0 / establecer X para leer el pin digital D0**], de modo que la variable X recibirá el valor 0 si el pin D0 tiene 0 Voltios o 1 si el pin D0 tiene 3.3 Voltios. A continuación, el programa imprime el valor de la variable X y de la variable C en la **Consola**, en el formato “D0 = n | C = n”, mediante el comando **print**. Después, la instrucción **IF** verifica si la variable X, que contiene la lectura del pin digital, es igual a 1, y de ser así, incrementa el valor de C en 1. Finalmente, el programa espera 1 segundo y reinicia la secuencia.

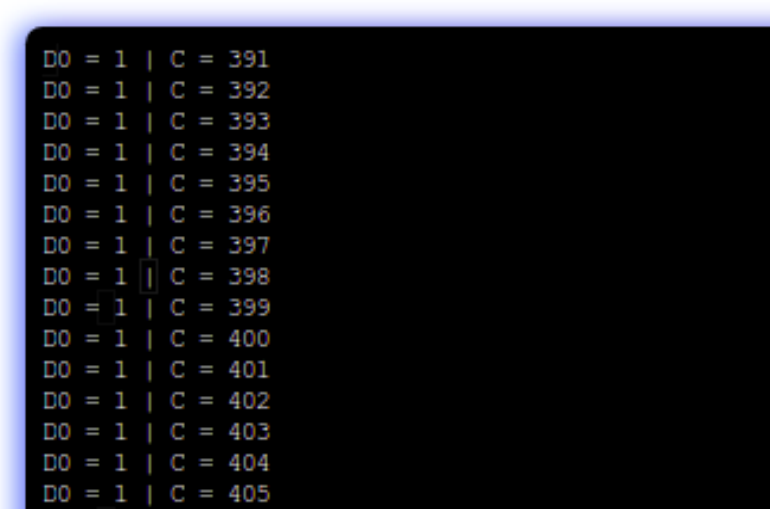


	Verifica que el bloque “**create text with / crear texto con**” tenga un pequeño engranaje azul. Es posible configurar la cantidad de entradas de este bloque haciendo clic en este ícono. En las siguientes prácticas, utiliza este botón azul para configurar las opciones del bloque cuando sea necesario.

La siguiente figura muestra una manera sencilla de probar el programa que acabamos de crear. Primero, inicia la ejecución del programa y utiliza un *cable jumper* para cambiar el valor de entrada del pin configurado en el programa (D0), conectándolo al pin 3V3 (nivel lógico 1) y luego al pin G (tierra, nivel lógico 0). Observa los cambios en la pestaña de **Consola** y cómo el contador aumenta cada segundo cuando el nivel lógico recibido es 1.



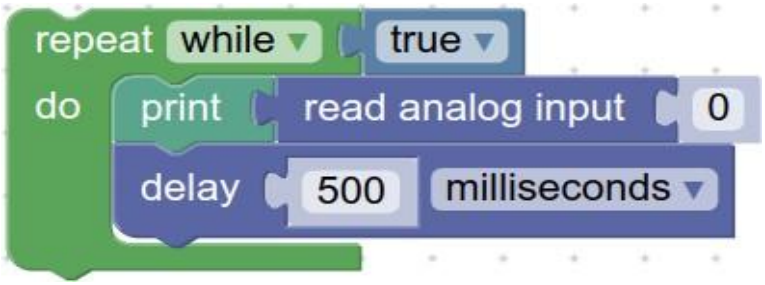
Ejemplo de resultado que se puede ver en la **Consola**:



Basándose en este ejemplo, se pueden crear varias aplicaciones, como contar la cantidad de vehículos o personas que han pasado por un portal, verificar si una puerta está abierta o cerrada, comprobar la presencia de personas mediante un sensor de presencia/alarma, revisar el nivel de un tanque de agua (si está lleno o no) con un sensor digital tipo boya, entre otras posibilidades.

Entrada analógica y sensor de luz LDR

Otra posibilidad de leer entradas externas es a través de los pines de entrada analógica. Mientras que los pines digitales solo detectan señales 0 o 1, los pines analógicos son sensibles a valores de medición variables dentro de un rango predefinido. Por ejemplo, pueden medir voltaje, temperatura u otros valores que varían entre distintas posibilidades. El ESP8266 tiene solo un puerto de entrada analógica (pin A0), que puede medir valores de entrada entre 0 y 3.3 voltios, mapeados internamente entre 0 y 1023, siendo 0 equivalente a 0 voltios y 1023 equivalente a 3.3 voltios[1]. El ESP32, por su parte, cuenta con varios pines de entrada analógica, que pueden usarse simultáneamente. Para utilizar la entrada analógica del módulo ESP8266, puedes emplear el siguiente programa con el bloque “Read ADC Input” o “leer entrada analógica”, que lee el pin analógico y está disponible en la sección Machine → In/Out Pins de la caja de herramientas de BIPES.

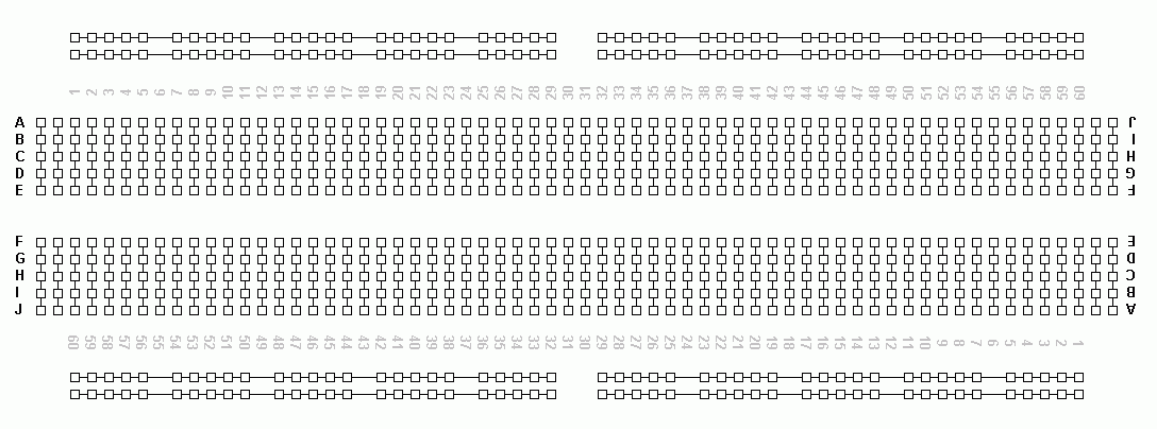


Puedes ver valores que van de 0 a 1023 en la pestaña Console mientras ejecutas el programa anterior. Durante la prueba, utiliza un cable de puente para conectar el pin A0 a G y observa la lectura de 0. Luego, conecta el pin A0 al pin 3V3 y nota que se imprime un valor cercano a 1023 en la Console. Atención: Los pines de entrada del módulo ESP8266 soportan un máximo de 3.3 voltios. No apliques valores de voltaje eléctrico superiores, ya que corres el riesgo de dañar o quemar el módulo.

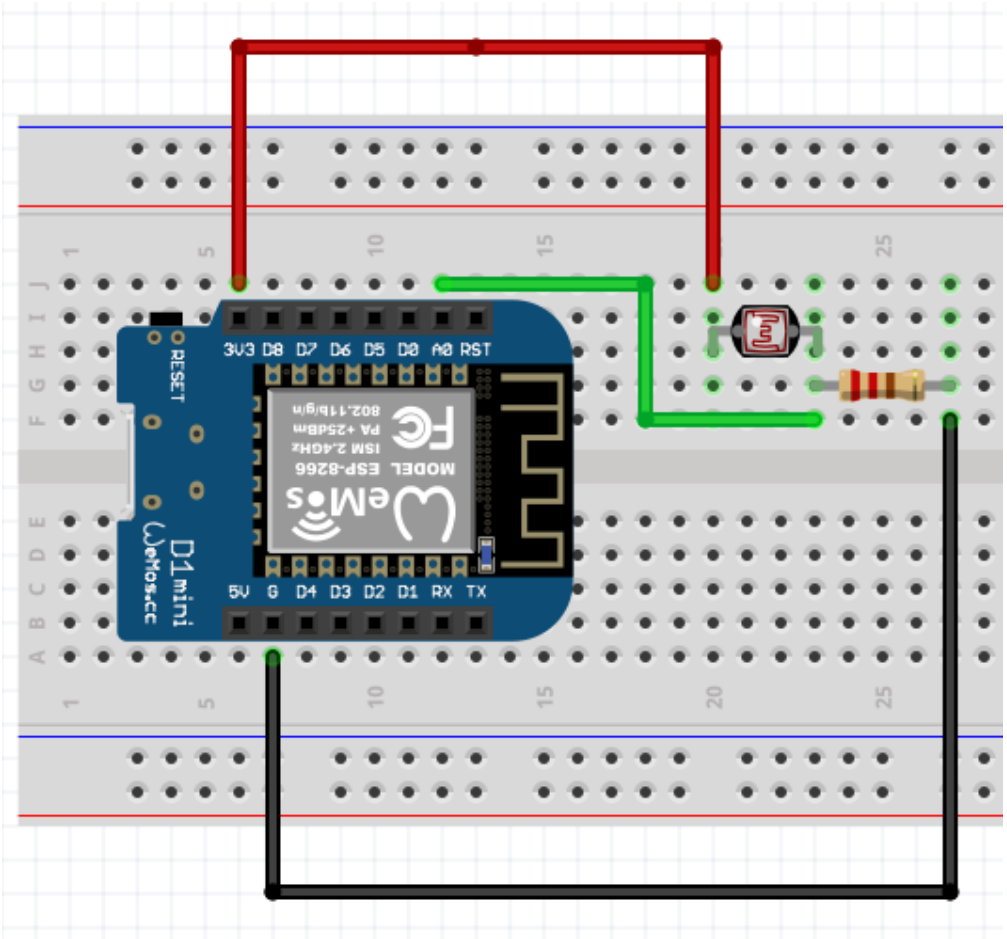
Actividad:
Teniendo en cuenta que la lectura 1023 corresponde a 3.3 Voltios, modifica el programa anterior para que muestre, en la pestaña de Consola, el valor en Voltios aplicado al pin A0 del módulo.

Ahora utilizaremos un sensor LDR (Resistor Dependiente de la Luz), que varía su resistencia eléctrica según la luz ambiental y puede ser monitoreado a través de una entrada analógica del módulo ESP8266. Primero, como se muestra en la siguiente figura, ensamble el circuito utilizando una placa de pruebas (protoboard o breadboard). La figura a continuación ilustra cómo la protoboard interconecta las partes: columnas en su centro y filas en sus bordes. Así, todos los componentes conectados en la misma columna tendrán automáticamente sus terminales conectados eléctricamente.

[1] El bloque de lectura de entrada analógica (**read analog input**) convierte el valor de la señal de voltaje analógico en la entrada a un valor digital de 10 bits.



Montaje del circuito con el módulo ESP8266, sensor de luz LDR y una resistencia de 220 ohmios (rojo, rojo, marrón):

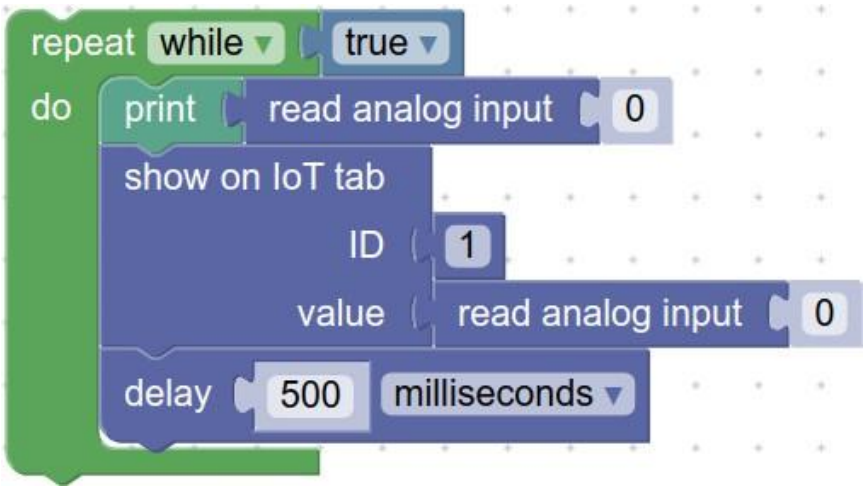


La configuración anterior se basa en una disposición electrónica llamada divisor de voltaje, de modo que la selección adecuada de 2 resistencias conectadas en serie puede establecer el valor del voltaje de salida en el terminal común entre las dos resistencias. En el caso de este circuito, el sensor LDR es una resistencia variable y, a medida que cambia la iluminación, la resistencia del LDR también varía, modificando el voltaje de salida, el cual es medido por

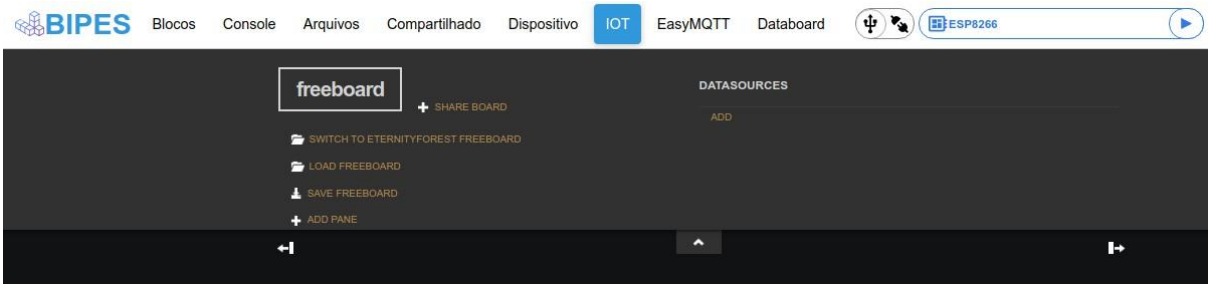
el pin A0 (cable verde). El cable rojo (V) está conectado a los 3V3 de la placa ESP8266, suministrando corriente al circuito. El cable de salida (verde) está conectado a la entrada analógica de la placa (AIN / ADC0), y el cable negro está conectado a la tierra de la placa (G o GND), cerrando el circuito. Ahora podemos usar el programa ya creado para analizar la variación de la iluminación detectada por el LDR.

El programa anterior “leerá” la entrada analógica y la mostrará en la pestaña Consola de BIPES. Intenta cubrir el sensor LDR con tu mano y observa cómo cambian los valores en la consola. También prueba apuntar una linterna al LDR y verifica las lecturas.

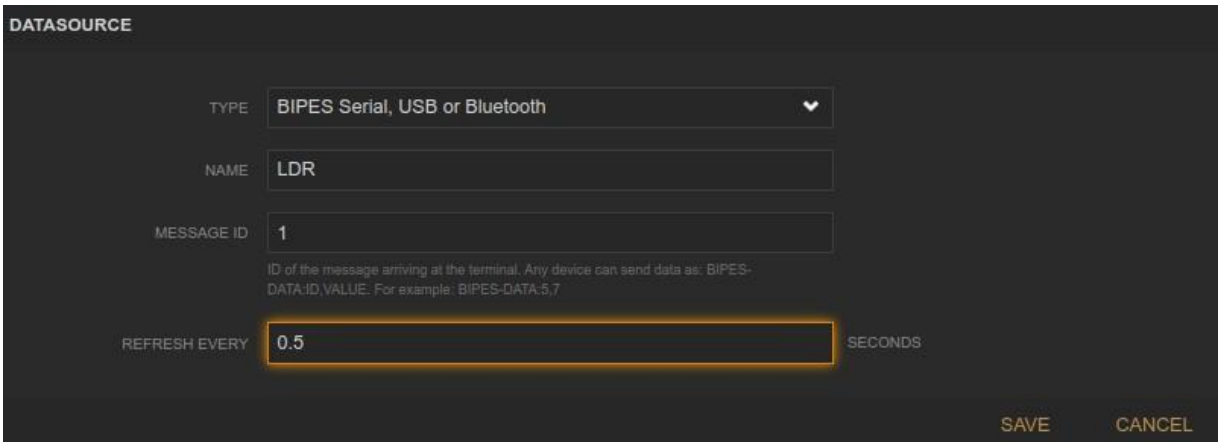
A continuación, podemos avanzar con esta configuración y visualizar las lecturas gráficamente. Para hacerlo, agrega el bloque “**show on IoT tab**”:



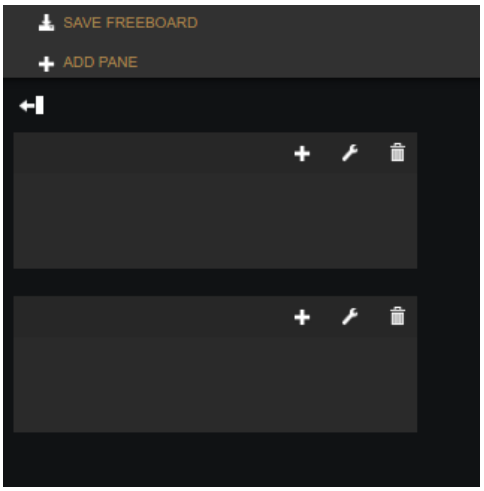
Ahora utiliza la pestaña BIPES IOT:



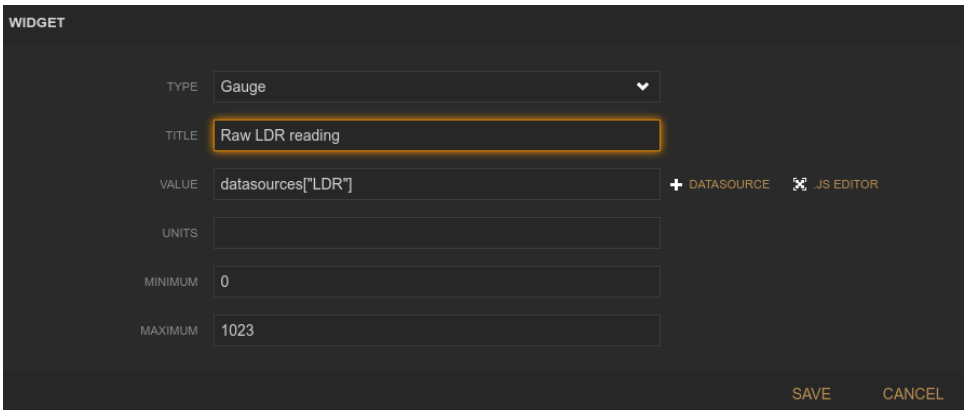
En **DATASOURCES**, haz clic en **ADD** y selecciona “**BIPES Serial, USB or Bluetooth**” y escribe **LDR** como el nombre, **1** como el ID del mensaje y **0.5** como la tasa de actualización:



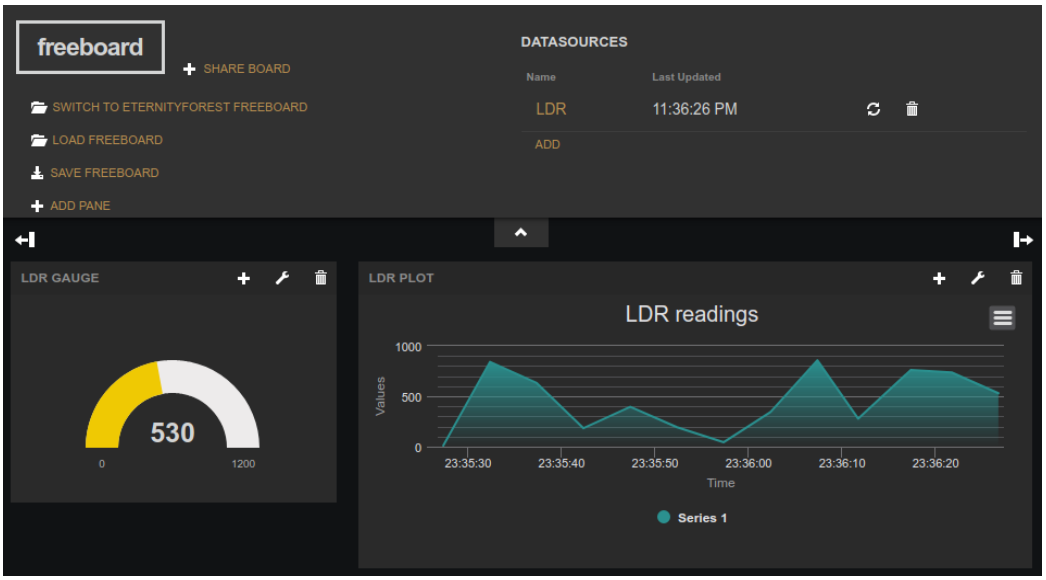
Guarda y añade uno o más "**panes**" (paneles), con el botón **ADD PANE**:



Arrastra y suelta los “paneles” para ajustar sus posiciones según tu preferencia. Luego, en cada “panel”, añade componentes de visualización (haciendo clic en el icono +), como prefieras, e ingresa “datasources[“LDR”]” en el campo **VALUE** de cada widget que hayas añadido.



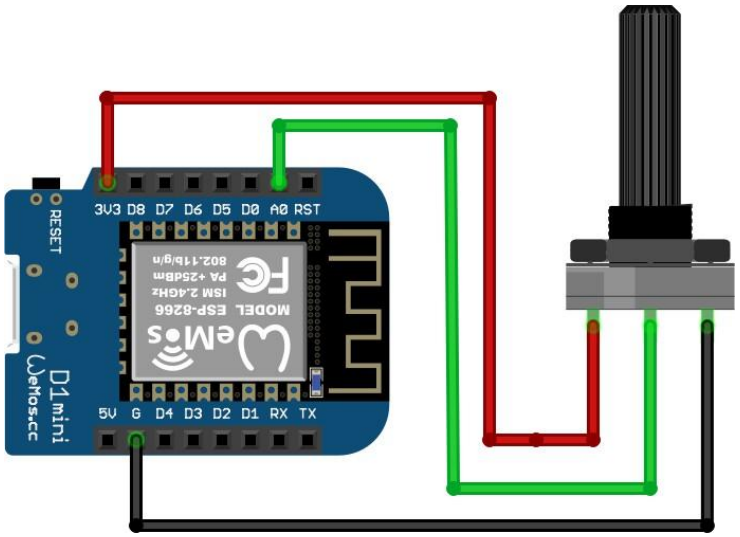
En el ejemplo a continuación, seleccionamos un widget de tipo **Gauge** y uno de tipo **Time series** (Highcharts):



Ejecuta el programa y revisa la visualización. Al variar la iluminación a la que está expuesto el sensor LDR, las lecturas cambiarán y se mostrarán en tiempo real en los widgets que has añadido al panel de control.

Actividad:
Cambia la LDR por un potenciómetro y verifica cómo cambian los valores al girar el eje del potenciómetro. Investiga o analiza qué es y cómo funciona el potenciómetro.

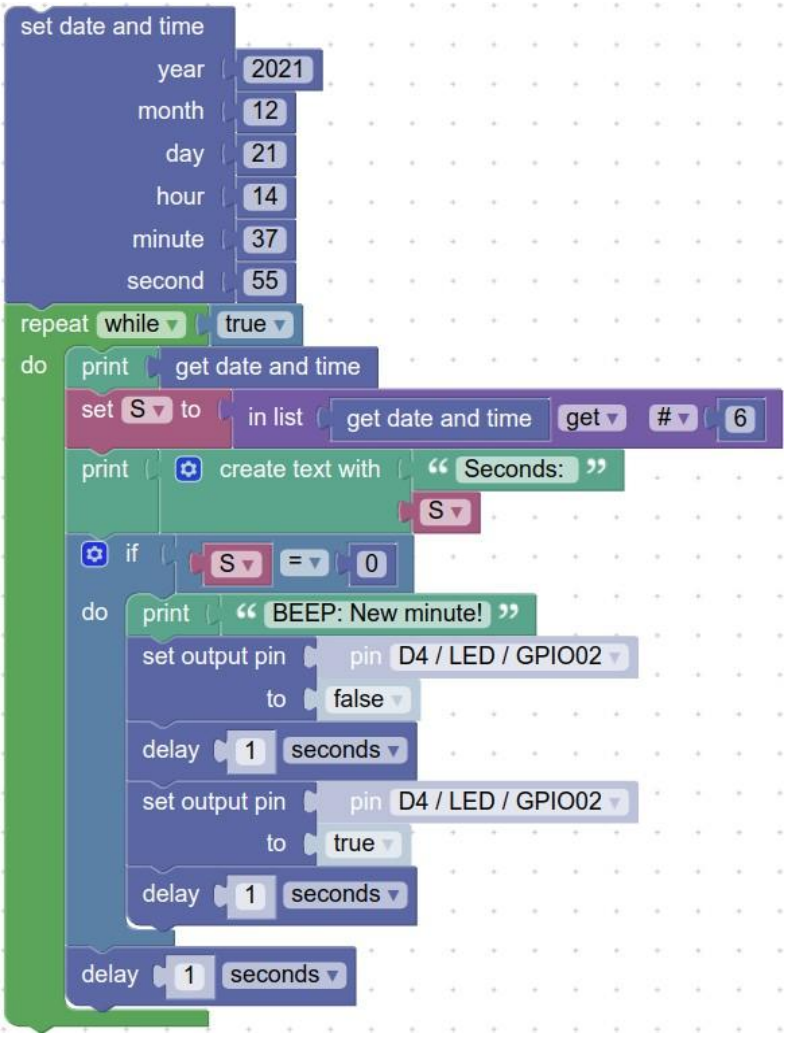
Sugerencia para el montaje del circuito:



Fecha y hora (RTC)

El MicroPython instalado en la placa ESP8266 o ESP32 permite acceder a una función de Reloj en Tiempo Real (RTC). El RTC permite al sistema establecer una fecha y una hora y mantener un registro actualizado del tiempo, independientemente de los programas ejecutados en la placa. El RTC interno del sistema se puede ajustar de varias maneras: manualmente, desde un RTC externo con batería, que proporciona la fecha y hora correctas incluso si el sistema está apagado, o a través de Internet, utilizando el bloque ***Network Time Protocol*** **(NTP)**.

El siguiente ejemplo muestra un programa que inicia el reloj con la fecha y hora 21/12/2021 a las 14:37:55. Después, el programa trata la fecha y la hora como una estructura de lista, separada por comas: **(2021, 12, 21, 1, 14, 37, 55, 0)**[2]. Luego, es posible usar el bloque "en lista" para tomar el sexto elemento (comenzando desde 0), que corresponde a los segundos, y almacenar su valor en la variable S. Después, se verifica SI la variable S es igual a 0 y, cuando esto ocurre, se enciende el LED durante un segundo.

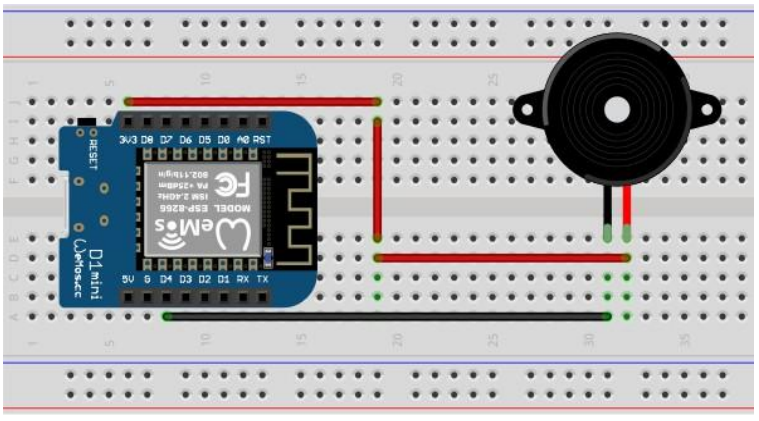


[2]Año, mes, día, día de la semana, hora, minuto, segundo, milisegundo)

El resultado puede ser verificado en la consola:

```
===
(2021, 12, 21, 1, 14, 37, 55, 222)
Seconds: 55
(2021, 12, 21, 1, 14, 37 , 56, 226)
Seconds: 56
(2021, 12, 21, 1, 14, 37, 57, 231)
Seconds: 57
(2021, 12, 21, 1, 14, 37, 58, 236)
Seconds: 58
(2021 , 12, 21, 1, 14, 37, 59, 241)
Seconds: 59
(2021, 12, 21, 1, 14, 38, 0, 245)
Seconds: 0
BEEP: New minute!
(2021, 12, 21, 1, 14, 38, 3, 233)
Seconds: 3
(2021, 12, 21, 1, 14, 38, 4, 232)
Seconds: 4
(2021, 12, 21, 1, 14 , 38, 5, 227)
Seconds: 5
```

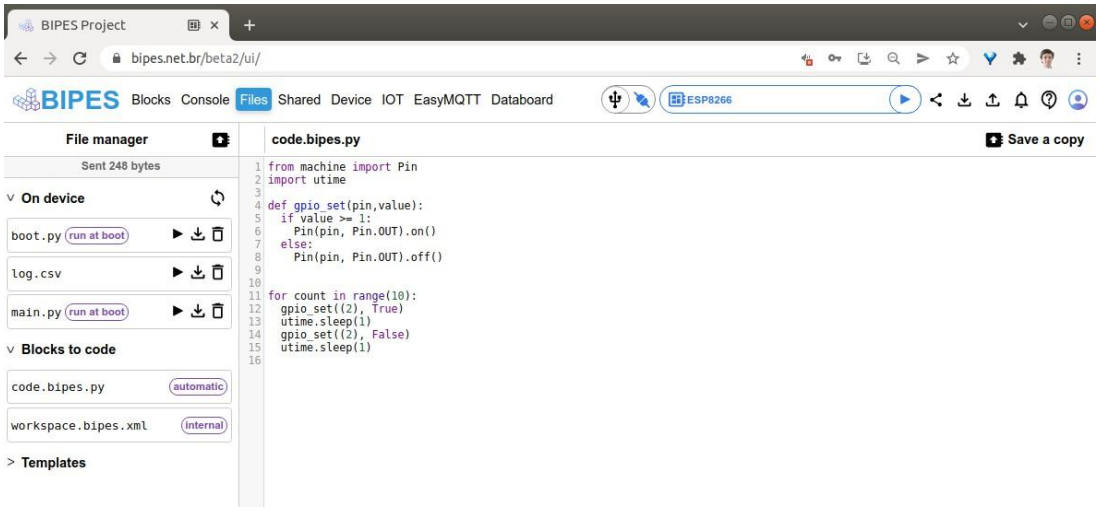
Un zumbador puede conectarse entre el pin 3V3 y el pin digital D4, permitiendo que el sistema emita un pitido de 1 segundo cada minuto. Por ejemplo, la figura de abajo muestra la posibilidad de conectar un zumbador utilizando los pines 3V3 conectados al + del zumbador y el D4 al otro terminal del zumbador. El circuito del zumbador se muestra en la siguiente figura.



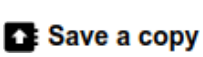
Tenga en cuenta que la fecha y la hora no se pierden incluso con el retraso de 3 segundos (2 segundos dentro del IF y uno en el bucle de repetición). La razón es que el RTC mantiene un registro actualizado de la fecha y la hora, independientemente del funcionamiento del programa. La fecha y hora actual se obtendrán siempre que se utilice el bloque “obtener fecha y hora”. Más adelante, puede usar un RTC externo que, respaldado por una batería, mantendrá la fecha y la hora siempre actualizadas, o bien obtener la fecha y hora de Internet mediante el bloque NTP.

Archivos en placa

Si ya has utilizado Arduino, debes recordar que en Arduino no existe el concepto de archivos almacenados en la placa. En el caso de BIPES con MicroPython, cada placa puede almacenar varios archivos, que pueden ser programas, datos, configuraciones o cualquier otra información que desees guardar en archivos. Por lo tanto, la gestión de archivos en BIPES es bastante sencilla y se puede realizar a través de la pestaña **FILES**. Todo este entorno de gestión de archivos puede utilizarse mediante un cable USB o por WiFi si la placa está conectada a una red WiFi. La siguiente figura muestra un ejemplo de la pestaña de archivos de **BIPES**:



La siguiente tabla explica cada botón disponible en la pestaña **FILES**:

	**Upload script to the device:** Envía un archivo desde tu computadora a la placa
	**Salve a copy:** Guarda una copia del archivo abierto con un nombre personalizado en la placa
	**Refresh device file list:** Actualiza la lista de archivos
	**Run File:** Ejecuta un programa de Python almacenado en ese archivo. También es posible usar atajos de teclado: Ctrl+Shift+R para iniciar un programa y Ctrl+Shift+S para detener un programa. Los atajos de teclado se pueden usar desde cualquier pestaña de BIPES.
	**Download File:** Descarga el archivo de la placa a tu computadora

	**Delete File:** Eliminar archivo

code.bipes.py	**Filename:** Nombre del último archivo abierto en el editor de texto BIPES o que será guardado mediante el comando Guardar una copia. Puedes hacer clic en esta área y editar el nombre del archivo. De este modo, es posible guardar copias o guardar modificaciones.

En el área de **Blocks to Code**, puedes ver el programa generado automáticamente a partir de los bloques de BIPES, el cual puede ser editado y tener copias guardadas. Además de almacenar programas y configuraciones, también es posible usar archivos para guardar los datos recolectados sin necesidad de una conexión a Internet. Por ejemplo, el siguiente programa lee la entrada analógica y la escribe en el archivo **log.csv** junto con la fecha y hora de la lectura, realizando la recolección de datos cada 60 segundos. Un programa como este podría ser utilizado en algunos sistemas de recolección de datos ambientales.

Los datos recopilados se almacenan en la memoria FLASH de la placa y se conservan incluso si ocurre un corte de energía. Por cierto, MicroPython también se guarda en la memoria FLASH, por lo que gestiona esta memoria para acomodar los archivos y programas del sistema y del usuario.

Cuando la placa está conectada a una computadora, es posible descargar el archivo log.csv, que tendrá el siguiente formato:

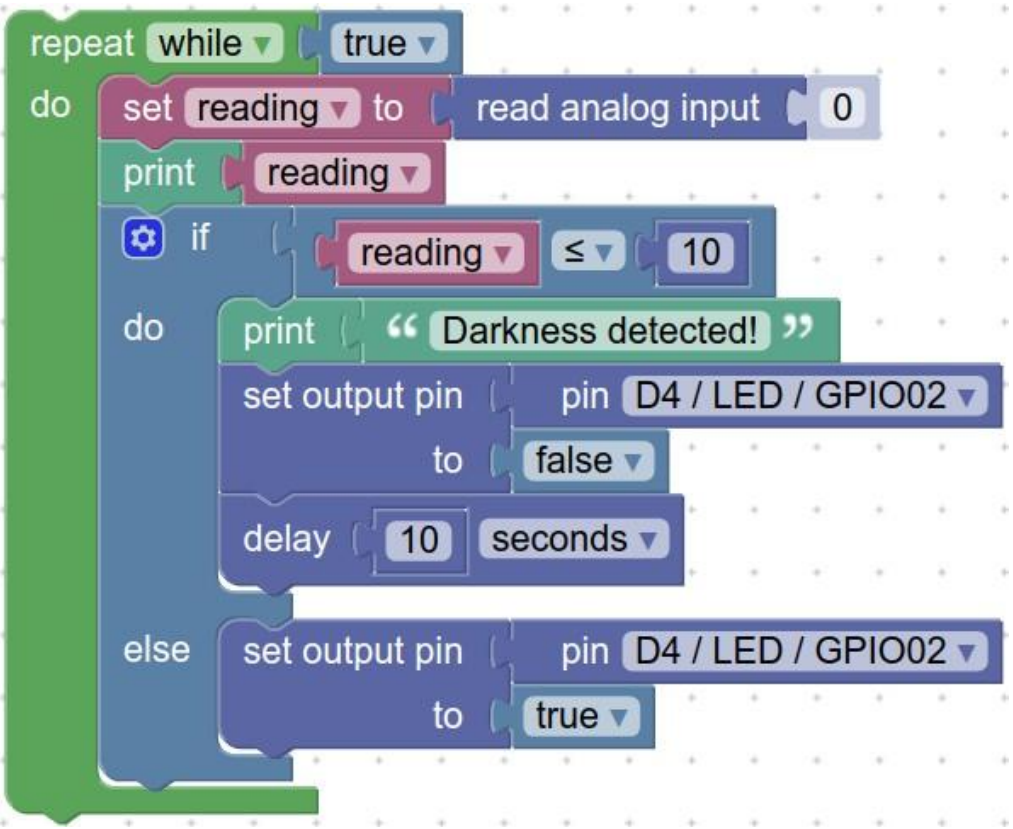
```
(2021, 12, 15, 2, 22, 42, 50, 3):8
(2021, 12, 15, 2, 22, 43, 0, 60):8
(2021, 12, 15, 2, 22, 43, 10, 119):8
(2021, 12, 15, 2, 22, 43, 20, 173) :8
(2021, 12, 15, 2, 22, 43, 30, 228):8
```

Como puedes ver, la fecha y la hora se escriben en cada línea del archivo, y luego la lectura de la entrada analógica también se escribe después de los dos puntos ":", tal como se indica en el programa por el bloque set L para crear el texto con M : leer entrada analógica 0 (**set L to create text with M : read analog input 0**). Si es necesario, MicroPython también permite realizar estas operaciones en una tarjeta de memoria SD, lo que permite un mayor espacio de almacenamiento para la recopilación de datos. Las tarjetas SD pueden añadirse fácilmente a los proyectos usando placas con ranuras integradas para tarjetas SD o shields para tarjetas SD.

Finalmente, después de leer la última sección de este libro (enviar datos a Internet), podrías crear programas que recojan datos en archivos y los envíen a Internet cuando haya una conexión disponible, evitando cualquier pérdida de datos, ya que el archivo de datos puede almacenarse en la memoria FLASH o en una tarjeta SD.

Comprobando una condición (Parte 2)

Ya hemos utilizado la estructura condicional **IF** para tomar acciones basadas en una entrada digital. Ahora, vamos a hablar de otra función de un sistema embebido, que consiste en tomar una decisión basada en la lectura de un sensor analógico; por ejemplo, activar un dispositivo cuando un sensor detecta un nivel predeterminado. En el caso del circuito con la LDR, un ejemplo sería activar el LED durante 10 segundos cuando el sensor detecta que se ha oscurecido. Para este ejemplo, definimos “oscuro” cuando la LDR proporciona un valor de medición menor que 10 (dentro del rango de 0 a 1023). Por ejemplo:



Prueba el programa e intenta ajustar diferentes niveles para la detección de oscuridad.

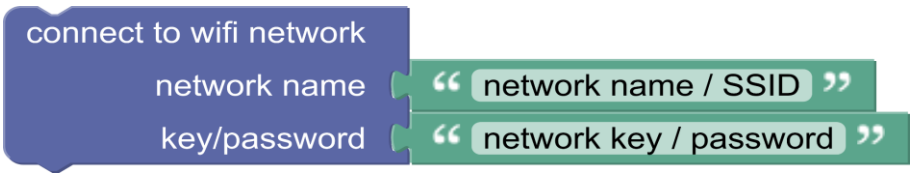
Actividad:
Haz que el LED parpadee mientras la iluminación esté por debajo del nivel especificado y deje de parpadear cuando esté por encima de este nivel.

Además, guarda el programa como main.py usando la pestaña **FILES** y prueba el sistema desconectado de la computadora, por ejemplo, conectado a un Power Bank o a un adaptador de corriente USB enchufado a la red eléctrica. Ten en cuenta que el programa se ejecuta de forma independiente en el ESP8266 y no en tu computadora.

Redes: listar redes Wi-Fi

En las siguientes actividades, ¡conectaremos nuestra placa a Internet! Pero antes de eso, es posible listar las redes disponibles. Inténtalo tú mismo. Las opciones de red se encuentran al final del cuadro de herramientas, bajo la opción "**Network and Internet**". Por ejemplo, el siguiente programa muestra las redes wifi disponibles:

Redes: conectarse a Internet

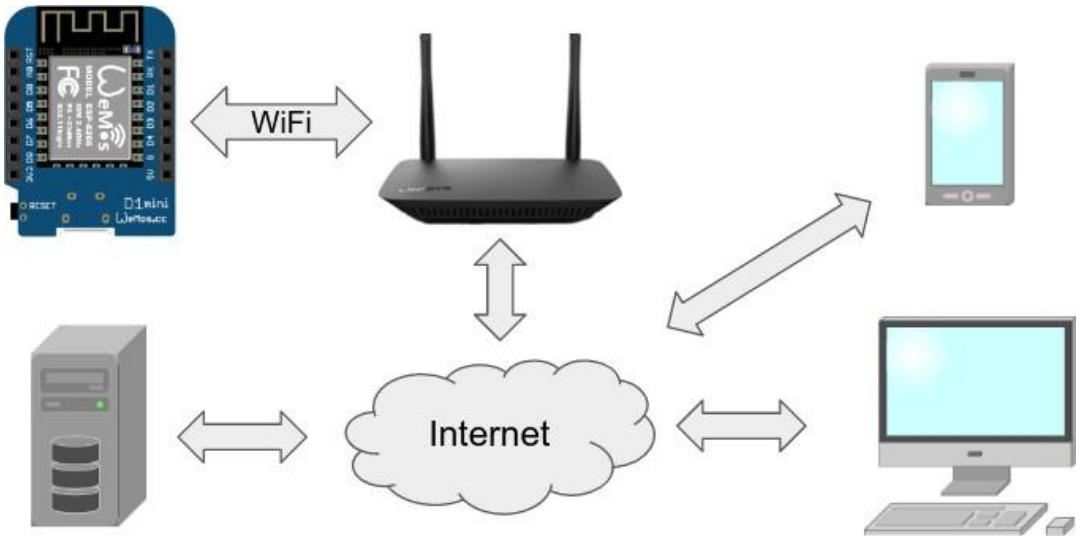


Utiliza el bloque de arriba para ingresar el nombre de la red y la contraseña y conectar a Internet. A continuación, intenta ejecutar el programa para que la placa se conecte a Internet. Este bloque obtendrá automáticamente una dirección IP de la red usando el Protocolo de Configuración Dinámica de Host (DHCP). En algún momento, podrías necesitar saber la IP obtenida: dicha información se puede obtener usando el bloque **Wifi current IP**. Después de listar las redes y conectarse a Internet, ¡ahora podemos enviar datos a la nube!

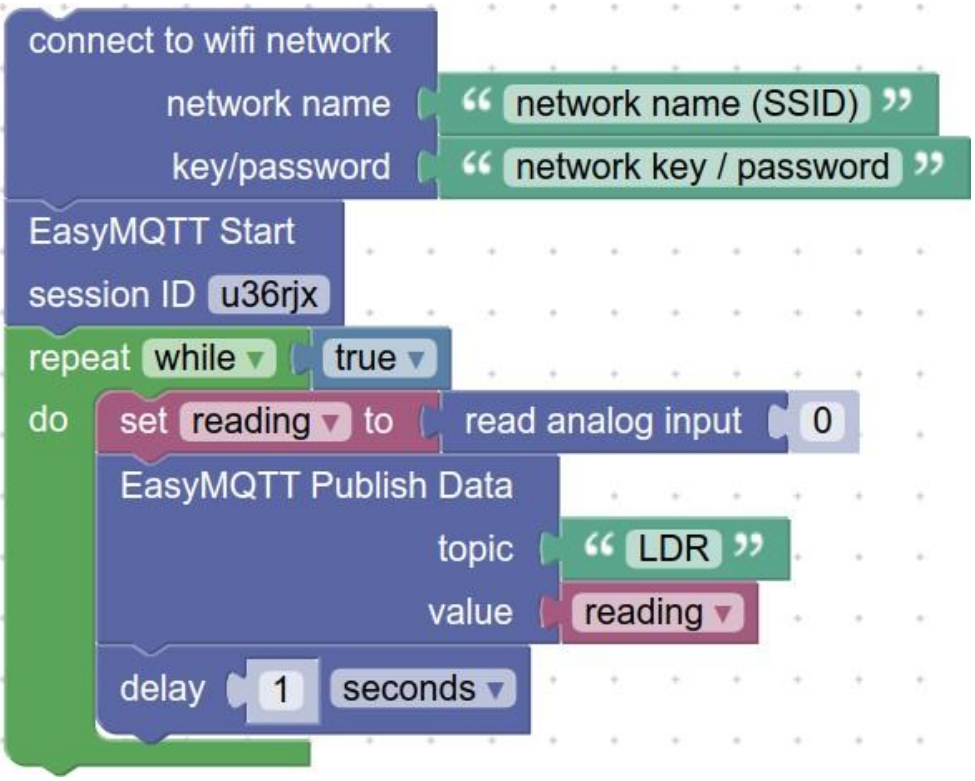
Redes: Envío de datos a Internet/nube

La figura a continuación ilustra un escenario típico de una aplicación que envía datos a una computadora remota, la cual cuenta con el software adecuado para recibir, almacenar y poner a disposición dichos datos recibidos. Esta computadora se denomina servidor y está representada en la parte inferior izquierda de la figura y, gracias a Internet, puede encontrarse en cualquier parte del mundo. Dicho servidor puede ser una máquina física o virtual, y también puede estar activo en servicios de "nube" como Google Cloud Computing (GCC) o Amazon Web Services (AWS). El servidor del proyecto BIPES, por ejemplo, está alojado en la plataforma en la nube de Google. La figura también muestra que varios dispositivos, como computadoras, tabletas y teléfonos inteligentes, pueden acceder a los datos almacenados en este servidor gracias a la conectividad a Internet. Además, la figura ilustra el sistema embebido, basado en la tarjeta ESP8266, que se conecta a Internet a través de un punto de acceso inalámbrico (Access Point).

De esa manera, un módulo ESP8266 puede recopilar información de sensores, enviar los datos a un servidor, y este servidor puede almacenar la información recibida y compartirla con otros dispositivos a través de una página web, por ejemplo.



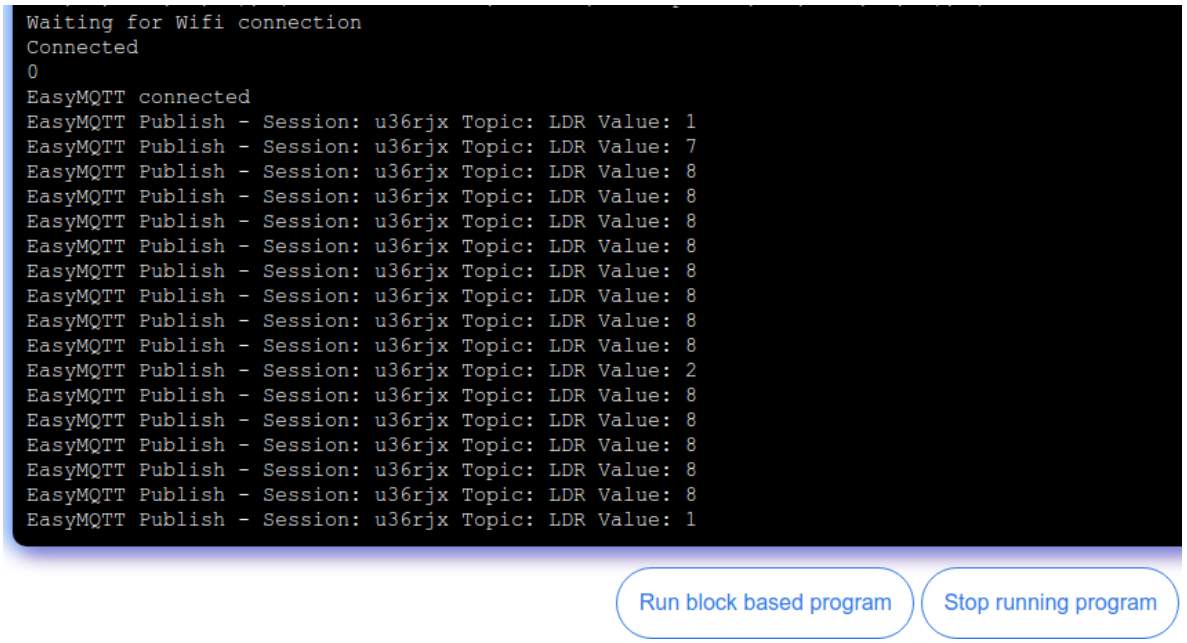
Con la placa conectada a Internet, utiliza los bloques en la caja de herramientas bajo **Network and Internet >> EasyMQTT** para crear un programa que envíe datos a la nube. Define el "Topic" (Tópico) y la fuente de datos o valor (Data). El identificador de sesión (Session ID) se crea automáticamente. Sin embargo, puedes definir manualmente el Session ID si es necesario. Por ejemplo, el siguiente programa envía lecturas analógicas del LDR a la nube usando MQTT:



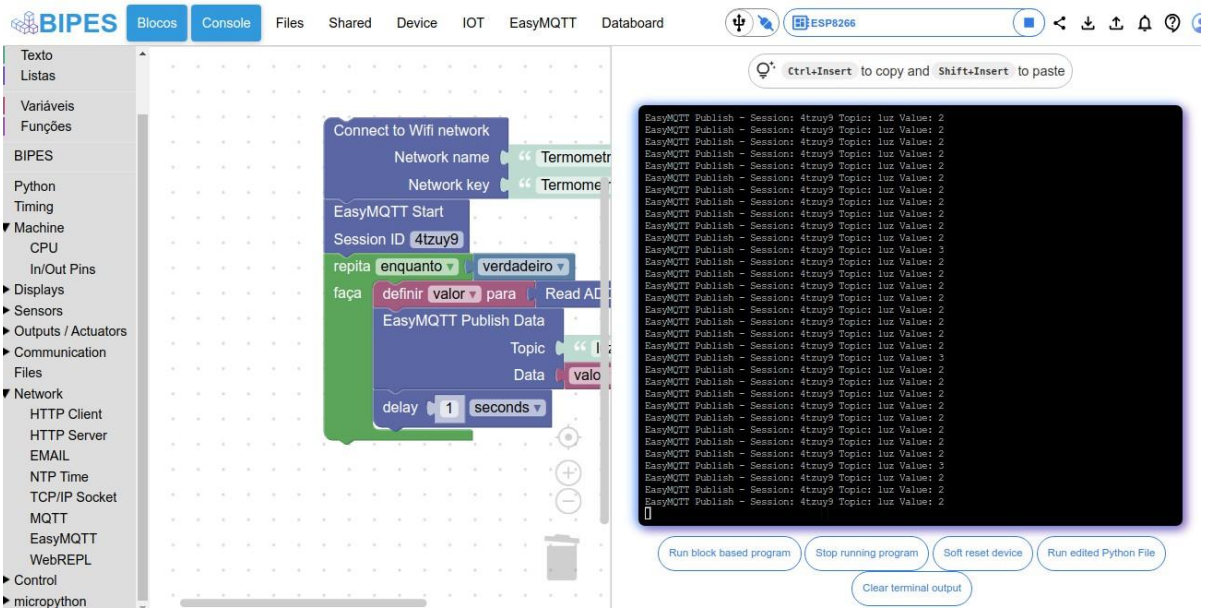
Message Queuing Telemetry Transport (MQTT) es un protocolo estándar de mensajería para el Internet de las Cosas (IoT). De esta manera, BIPES ofrece la función EasyMQTT, que

proporciona una abstracción fácil de usar para servicios MQTT, permitiendo una creación de prototipos de IoT rápida y flexible. Además, BIPES también te permite crear programas que se comunican con cualquier otro dispositivo o servicio MQTT, como ThingSpeak, ThingsBoard o incluso la plataforma IoT de SAP.

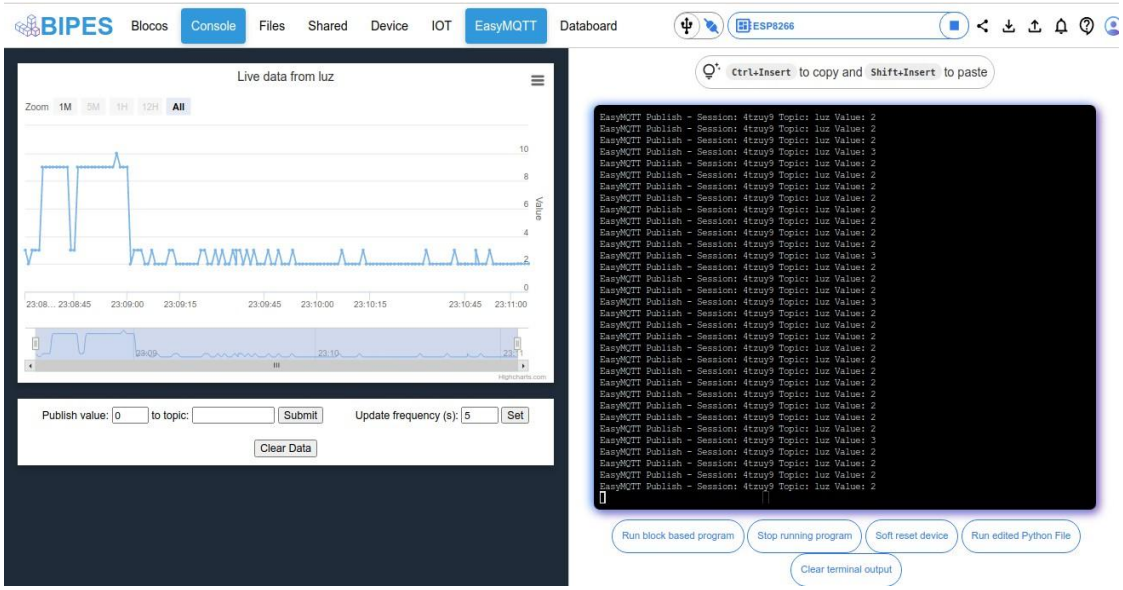
Al ejecutar el programa, puedes ver el resultado en la pestaña Console:



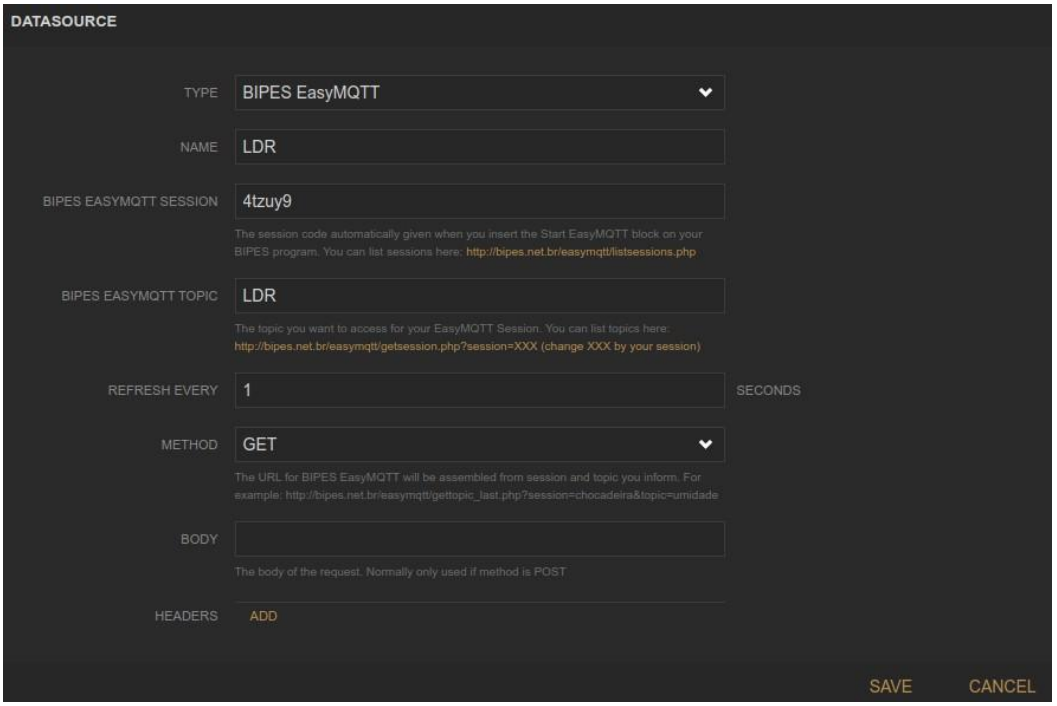
Recuerda que BIPES tiene una función muy útil: trabajar/ver dos pestañas simultáneamente haciendo clic derecho con el ratón en una pestaña y clic izquierdo con el ratón en la otra pestaña deseada. Así:



Así, puedes hacer clic con el botón izquierdo del ratón en la consola y con el botón derecho en EasyMQTT para ver los resultados enviados a la nube mientras supervisas la ejecución del programa.

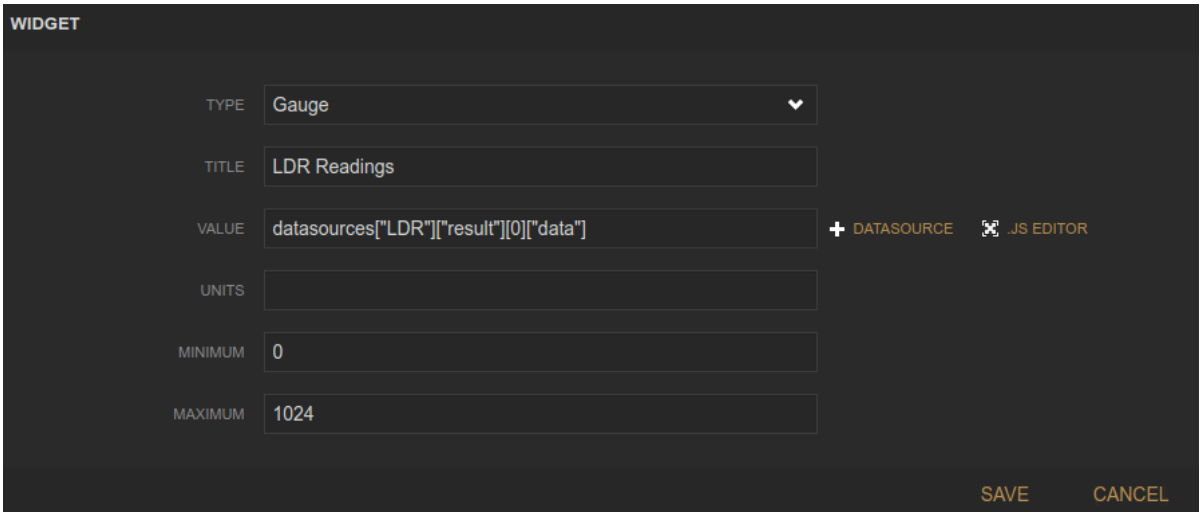


Compartir: Es posible hacer clic en el botón para compartir (<) y se generará un enlace de Internet para compartir tu programa públicamente. Puedes enviar este enlace a cualquier persona, y cuando accedan a él, podrán ver los datos en tiempo real enviados por tu placa haciendo clic en la pestaña **EasyMQTT**. Finalmente, también es posible crear un panel personalizado (dashboard) para visualizar estos datos, utilizando nuevamente la pestaña IoT (ya lo hemos hecho para el cable USB, pero ahora lo haremos usando Internet).



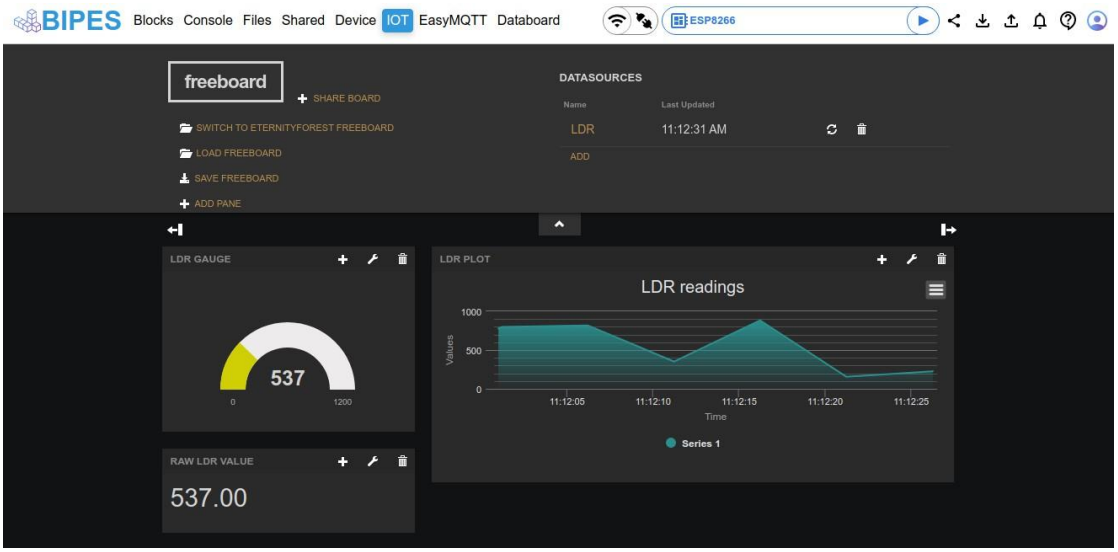
Esta vez, vamos a usar ***datasource*** EasyMQTT:

En *datasource*, informa la sesión de EasyMQTT y el tópico definidos en tu programa y haz clic en GUARDAR. A continuación, vamos a añadir los paneles y los widgets gráficos. Primero, agrega los paneles y luego añade widgets a tus paneles. Para cada widget, ingresa el valor del origen de datos. Puedes hacer clic en el botón “+ DATASOURCE” al lado del campo de valor para ayudarte con el autocompletado:



En el ejemplo, la fuente de datos utilizada es: **datasources["LDR"]["result"][0]["data"]**

Puede agregar otros componentes y personalizar su panel. Por ejemplo:



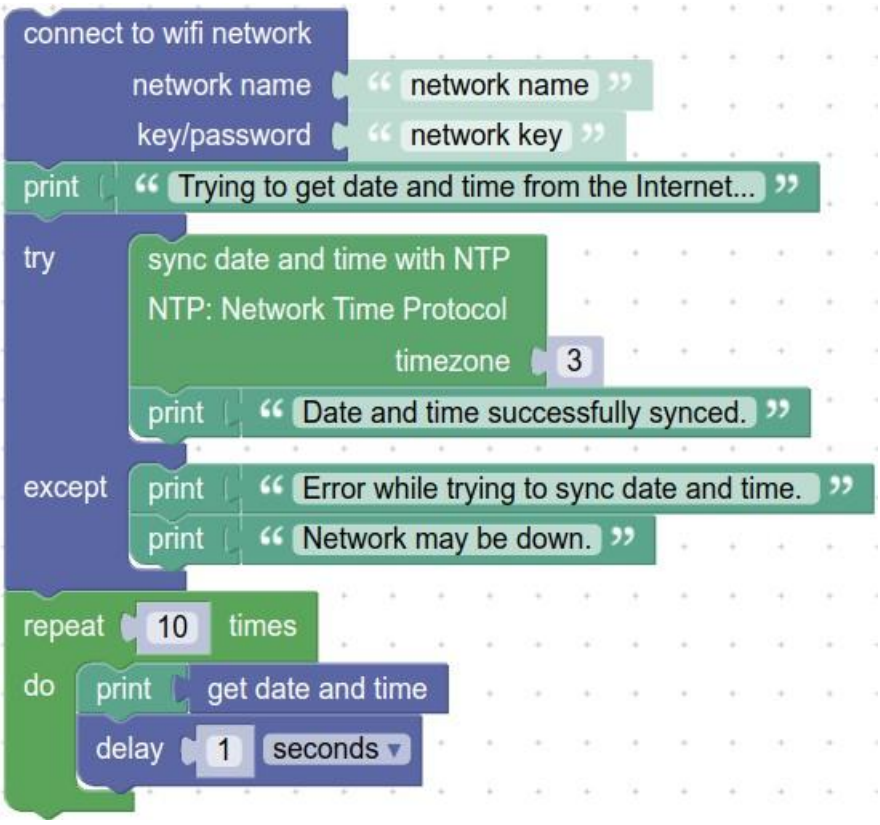
Una vez más, ten en cuenta que puedes compartir este programa con otra persona a través del enlace, usando el botón ( ). Esa persona tendrá acceso al panel de control y a los datos enviados por tu dispositivo desde cualquier parte del mundo. La opción de compartir permitirá compartir el programa y el acceso a las pestañas de IoT y **EasyMQTT**.

Manejo de errores

Varias situaciones inesperadas pueden ocurrir en un sistema embebido, y en muchos casos, no habrá un usuario para reiniciar el sistema. Además, algunos sistemas embebidos pueden encontrarse en lugares inaccesibles.

Por lo tanto, es importante contar con métodos para detectar y manejar errores. El bloque **try-except**, disponible en la caja de herramientas de **Python**, permite que el sistema intente ejecutar una operación y, si el proceso falla, ejecute un código alternativo. Esta verificación es útil para diversos sistemas. Para las aplicaciones conectadas a la red, también resulta útil porque, por varias razones, la conexión puede perderse inesperadamente. Además, en algunos casos, la falla de la conexión de red puede interrumpir la ejecución de todo el programa hasta que un usuario u operador lo reinicie.

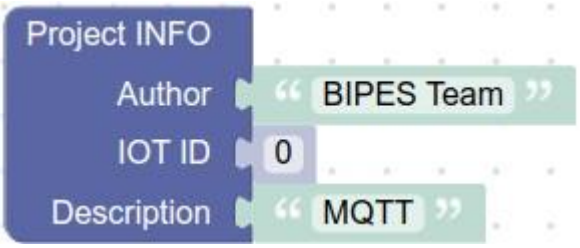
El siguiente programa ilustra un ejemplo de sincronización de fecha y hora desde Internet utilizando el bloque NTP (disponible en la caja de herramientas de **Network and internet**). Incluso si se produce una falla de comunicación con el servidor NTP, el programa continúa.



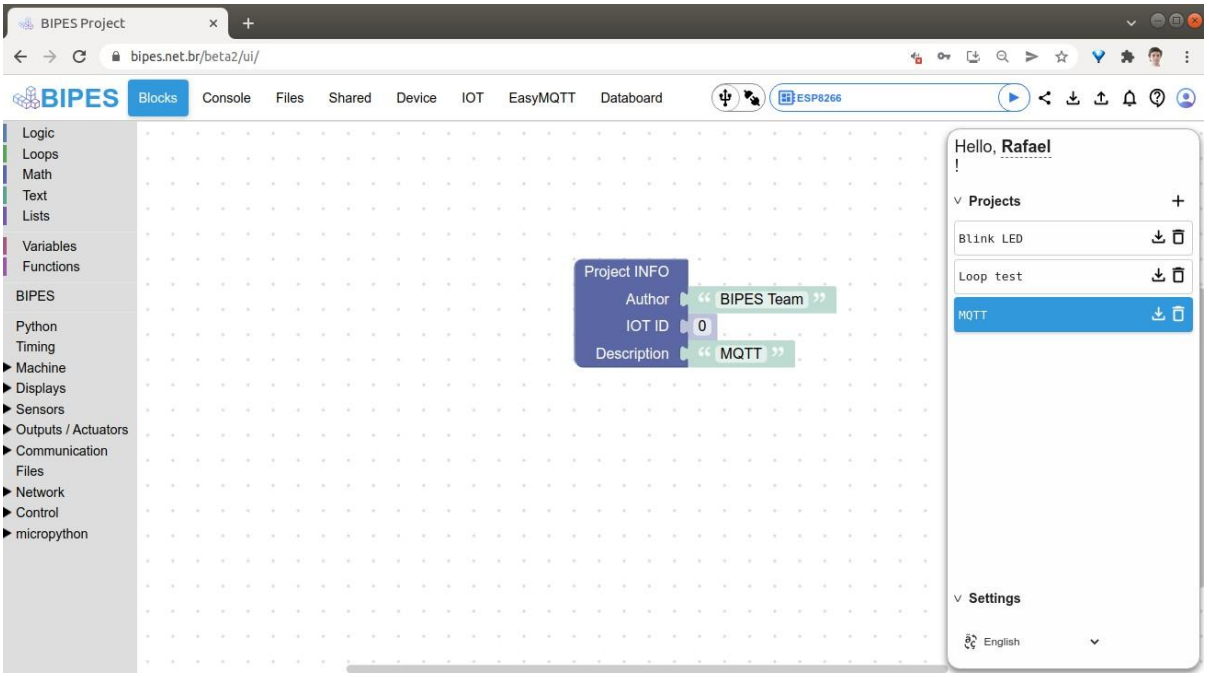
Sin la estructura **try**, si el programa falla al conectarse al servidor NTP, nunca se ejecutarán los bloques dentro de la estructura **repeat**. Usando **try**, incluso si ocurre un error, el flujo del programa continúa y se ejecuta repeat.

BIPES: Múltiples proyectos

Con la facilidad que ofrece BIPES, no podrás resistirte a probar varios programas y experimentos. Para facilitar este proceso, ten en cuenta que cada vez que creas un nuevo programa en BIPES, el siguiente bloque se pone automáticamente a disposición en el escritorio:



Este bloque permite definir el nombre y el autor del programa. También observa que, en la esquina superior derecha de la pantalla de BIPES, donde se puede utilizar un icono de perfil: . Este ícono accederá a la siguiente área de BIPES en la parte derecha de la pantalla:

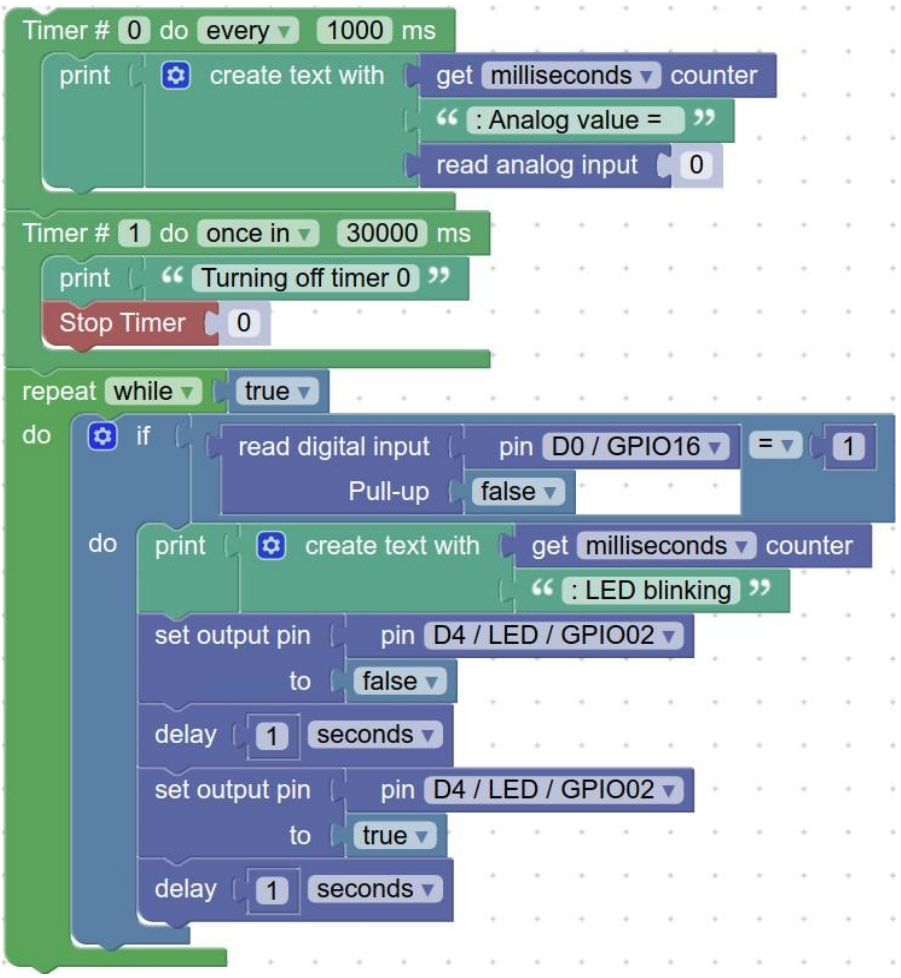


En esta área, es posible crear nuevos programas, explorar entre los programas ya realizados, cambiar entre programas, eliminar o descargar el archivo XML correspondiente a ese programa en la computadora local. La misma área también permite cambiar el idioma de la interfaz de BIPES. Ten en cuenta que estos programas se almacenan localmente en el área de almacenamiento de tu navegador web.

Actividades paralelas: temporizadores (timers)

Otra característica que se utiliza frecuentemente en los sistemas embebidos son los *timers*. Los *timers* son mecanismos de temporización respaldados por hardware que generan eventos automáticamente en momentos programados. Los *timers* de hardware generan interrupciones en los programas en ejecución, desviando el flujo de ejecución del programa hacia una rutina asociada con un *timer*. Cuando la rutina del *timer* se completa, el sistema retorna el flujo de ejecución exactamente al punto donde se desvió del programa principal. Como estos cambios ocurren rápidamente, el efecto percibido es que el módulo está realizando múltiples tareas de manera simultánea.

El siguiente programa muestra un ejemplo donde el ***Timer 0*** está configurado para imprimir la lectura de la entrada analógica en la **Consola** cada 1000 ms. Cada línea impresa muestra el conteo de tiempo, en milisegundos, desde que se encendió el ESP8266 con el bloque **[get milliseconds counter]**, seguido de la lectura analógica. El ***Timer 1*** está configurado para apagar el **Timer 0** después de 30 segundos. El programa también incluye un bucle de repetición que hace parpadear el LED en el pin D4 mientras el pin digital D0 tiene una señal lógica alta (1), mostrando el instante de tiempo en que ocurre este evento.



La siguiente lista muestra la salida de la Consola después de que este programa haya estado en ejecución durante unos segundos. Los textos entre los símbolos “<< >>” indican nuestras notas, las cuales no fueron impresas en la Consola por el programa en ejecución.

```
===
<< PROGRAM STARTED WITH PIN D0 IN LOW (0) LOGIC LEVEL (D0 ⇦ GND)>>
3816: Analog value = 1
4816: Analog value = 1
5816: Analog value = 1
6816: Analog value = 1
7816: Analog value = 1
8816: Analog value = 1
9816: Analog value = 1
10816: Analog value = 1
11816: Analog value = 1
12816: Analog value = 7
13816: Analog value = 7
14816: Analog value = 8
<< PIN D0 RECEIVES HIGH LOGIC LEVEL (1) (D0 ⇦ 3V3)>>
15445: LED blinking
15816: Analog value = 7
16816: Analog value = 7
17453: LED blinking
17816: Analog value = 8
18816: Analog value = 8
19461: LED blinking
19816: Analog value = 8
20816: Analog value = 8
<< SOME SECONDS OF DATA SHOWN ON THE CONSOLE WERE OMITTED >>
31510: LED blinking
31816: Analog value = 1
32816: Analog value = 8
Turning off timer 0
<< AFTER 30 SECONDS, TIMER 0 IS TURNED OFF AND THE LOOP CONTINUES >>
33518: LED blinking
35526: LED blinking
37534: LED blinking
39543: LED blinking
41551: LED blinking
43559: LED blinking
```

Los números que aparecen antes de “:” indican el tiempo, en milisegundos, de cada evento. Ten en cuenta que los eventos del bucle principal y de los dos temporizadores (timers) ocurren de manera independiente.

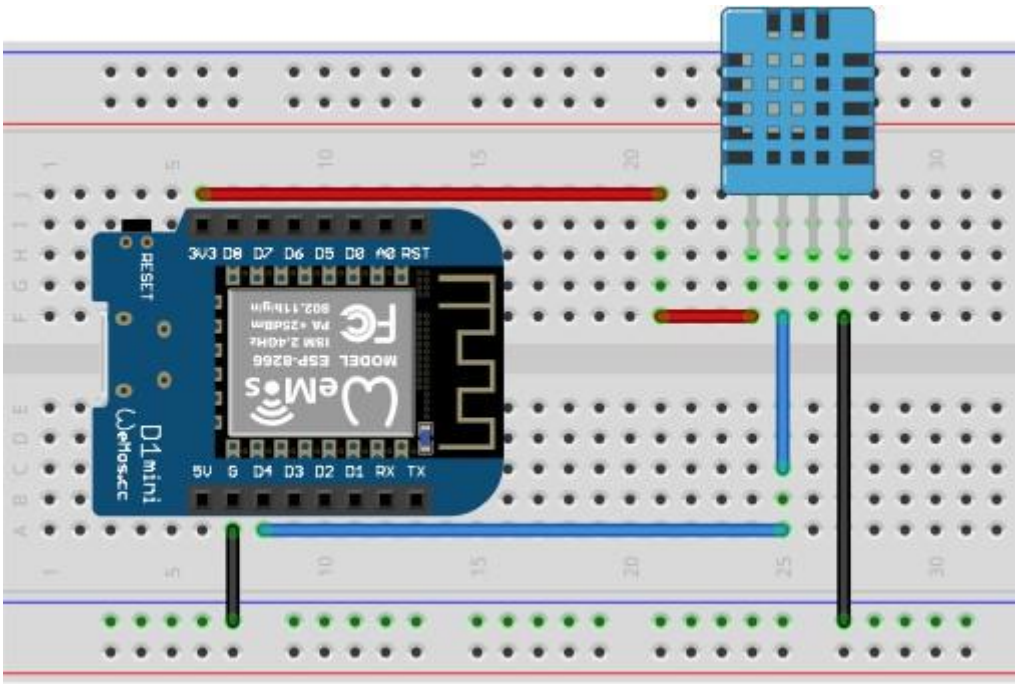
Sensor de temperatura y humedad

¡También podemos incluir un sensor de temperatura y humedad en el sistema y enviar las mediciones a la nube! Utiliza cables jumper directamente, o usa la protoboard para conectar el sensor de humedad y temperatura DHT11 al módulo ESP8266. Ten en cuenta que el sensor debe conectarse al pin 3V3 para recibir energía, a tierra (GND), y la salida (segundo pin) al pin de entrada D1 o D4 (o al que consideres adecuado) en la placa.

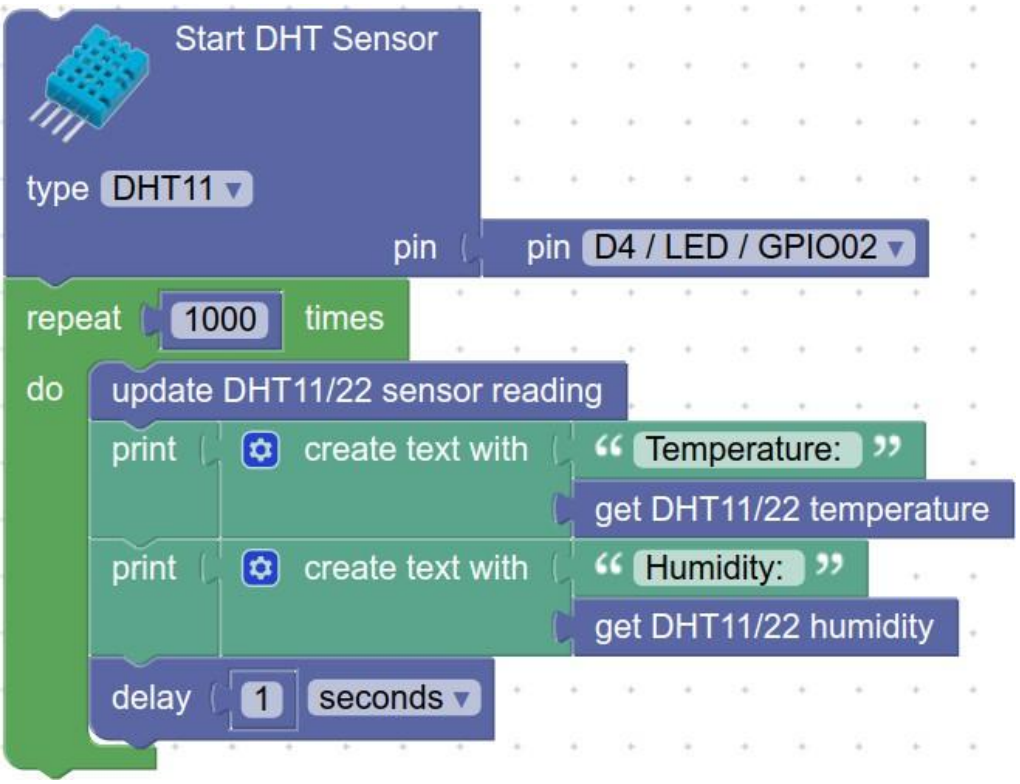
Sugerencia de conexión:

Pin del DHT11	**Pin del ESP8266**
GND (Pin 4)	**G**
VCC (Pin 1)	**3V3**
DATA (Pin 2)	**2 (D4)**

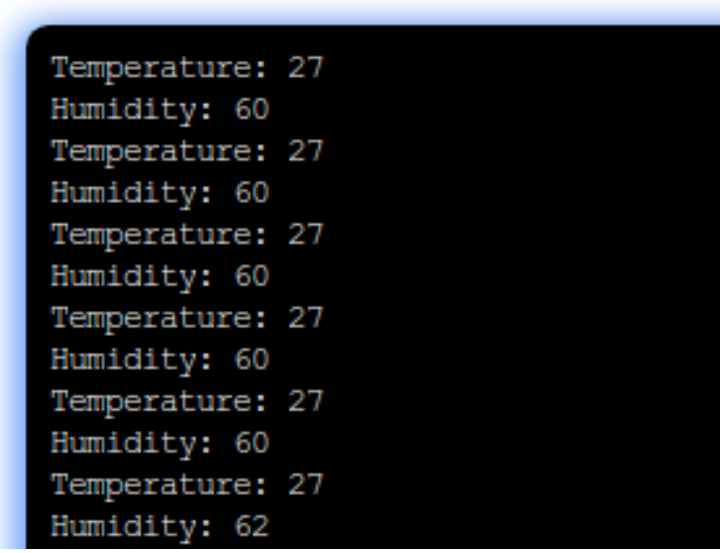
Sugerencia de ensamblaje de circuito para la placa WeMos D1 mini con el módulo ESP8266 conectado al DHT11:



Programa:

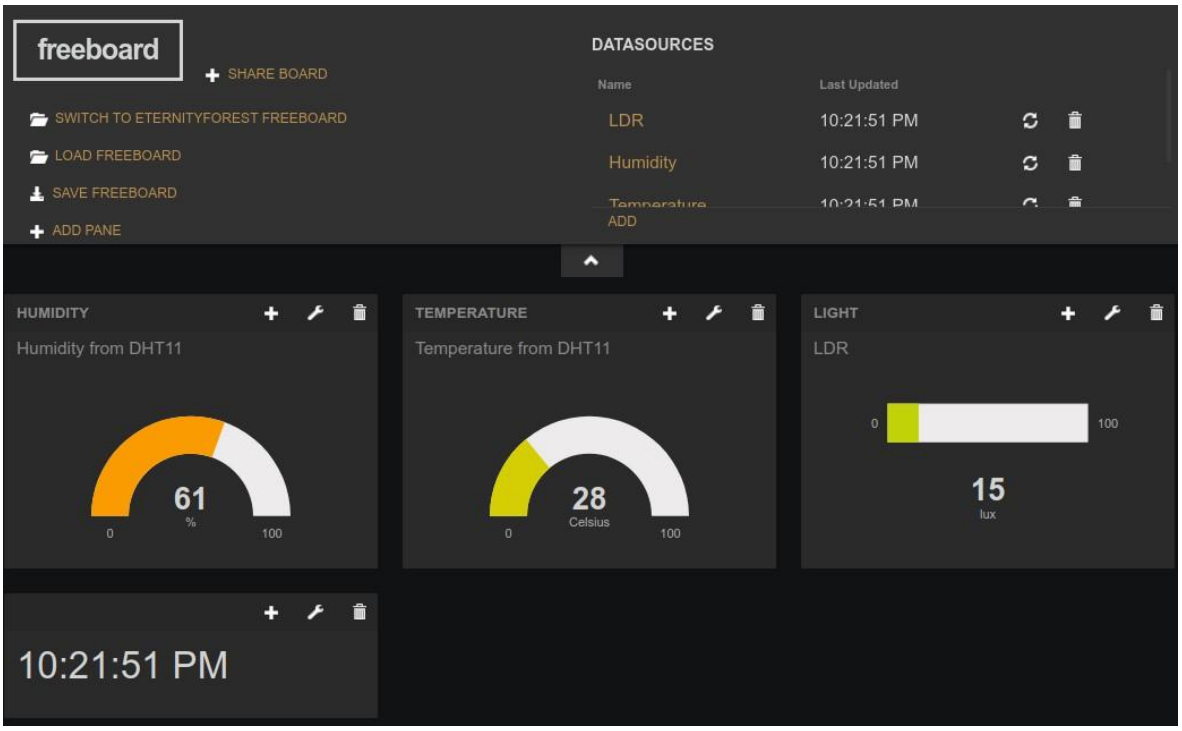


Ejecuta el programa y visualiza el resultado en la pestaña Consola. Resultado del ejemplo:



Actividad:
Utiliza EasyMQTT para incluir una función adicional que envíe la humedad y la temperatura a la nube, junto con las lecturas del LDR. En la pestaña de IoT, configura un panel IoT (panel de control) y monitorea las lecturas enviadas por la placa usando otro dispositivo/computadora.

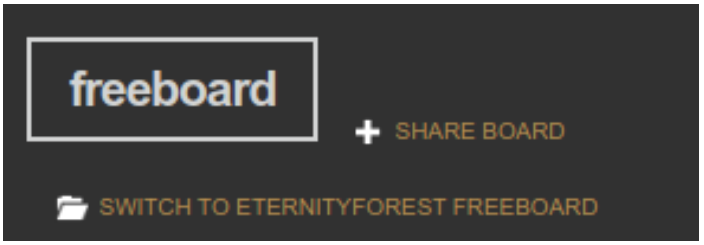
Ejemplo de resultado de la actividad:



Comparte el panel de control con teléfonos inteligentes y otros dispositivos

Después de que la aplicación esté lista, es posible compartir el enlace únicamente para el panel de visualización (freeboard) con cualquier dispositivo, incluyendo la optimización para teléfonos móviles (tecnología web adaptable).

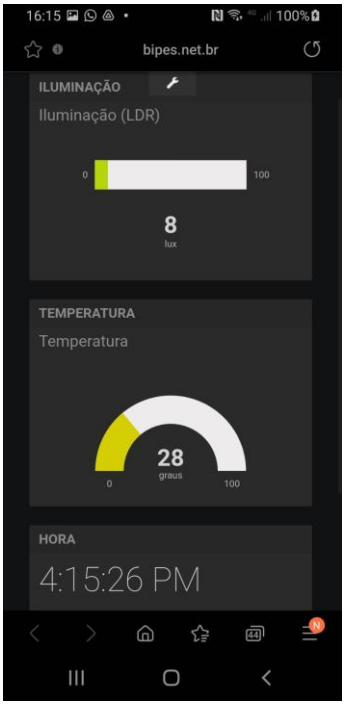
Haz clic en el botón de compartir de BIPES, y se generará un enlace, similar a: http://bipes.net.br/beta2/ui/#4c6bfm. Reemplaza el término ***beta2/ui/*** por ***freeboard***, de modo que el nuevo enlace será: http://bipes.net.br/freeboard#4c6bfm. Este último enlace solo compartirá el panel (freeboard). Otra posibilidad es usar la opción **SHARE BOARD**, junto al icono de freeboard:



Cuando haces clic en el botón "**+ SHARE BOARD**", se genera un enlace junto con su código QR correspondiente, y ambos se mostrarán en una nueva ventana, como se indica a continuación. Ten en cuenta que puede que tengas que autorizar las ventanas emergentes en tu navegador web.

Ahora puedes compartir este enlace o código QR con cualquier usuario que verá los datos e incluso podrá enviar comandos a tu sistema (si incluyes una aplicación interactiva con botones (interruptores)). Además, al acceder al código QR o al enlace desde un celular, el usuario también tiene acceso a un panel de control responsivo, similar al de la derecha, ¡gracias a freeboard!

Para obtener más información sobre freeboard, consulta el repositorio de freeboard en GitHub: https://github.com/Freeboard/freeboard

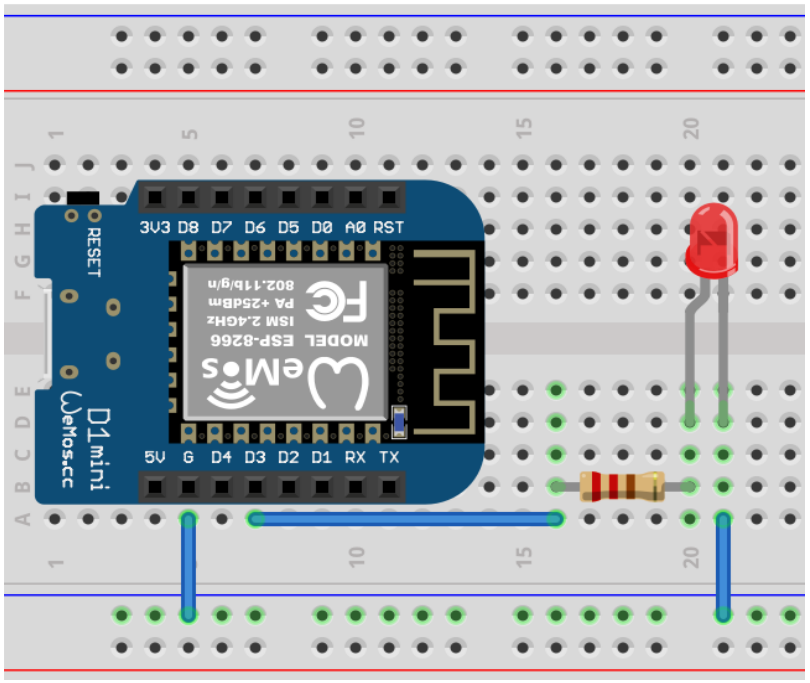


PWM: Control de brillo LED

Ya hemos hablado sobre cómo controlar una salida digital y verificar el estado de los pines de entrada digital y entrada analógica. Otra posibilidad es utilizar la salida digital de tipo PWM (Modulación por Ancho de Pulso), que permite que una señal digital en forma de onda cuadrada aproxime un valor de voltaje equivalente a una salida analógica. Así, es posible usar un pin con salida PWM para controlar el brillo de un LED, la velocidad de un motor, la temperatura de una resistencia calefactora, entre otras posibilidades. Lamentablemente, no todos los pines son compatibles con la salida PWM, por lo que el LED ya incluido en la placa ESP8266 no está conectado a un pin PWM. Por lo tanto, necesitaremos un LED externo conectado a un pin con la característica de salida PWM para poder probar esta función. En el siguiente ejemplo, usamos el pin D3 (GPIO0).

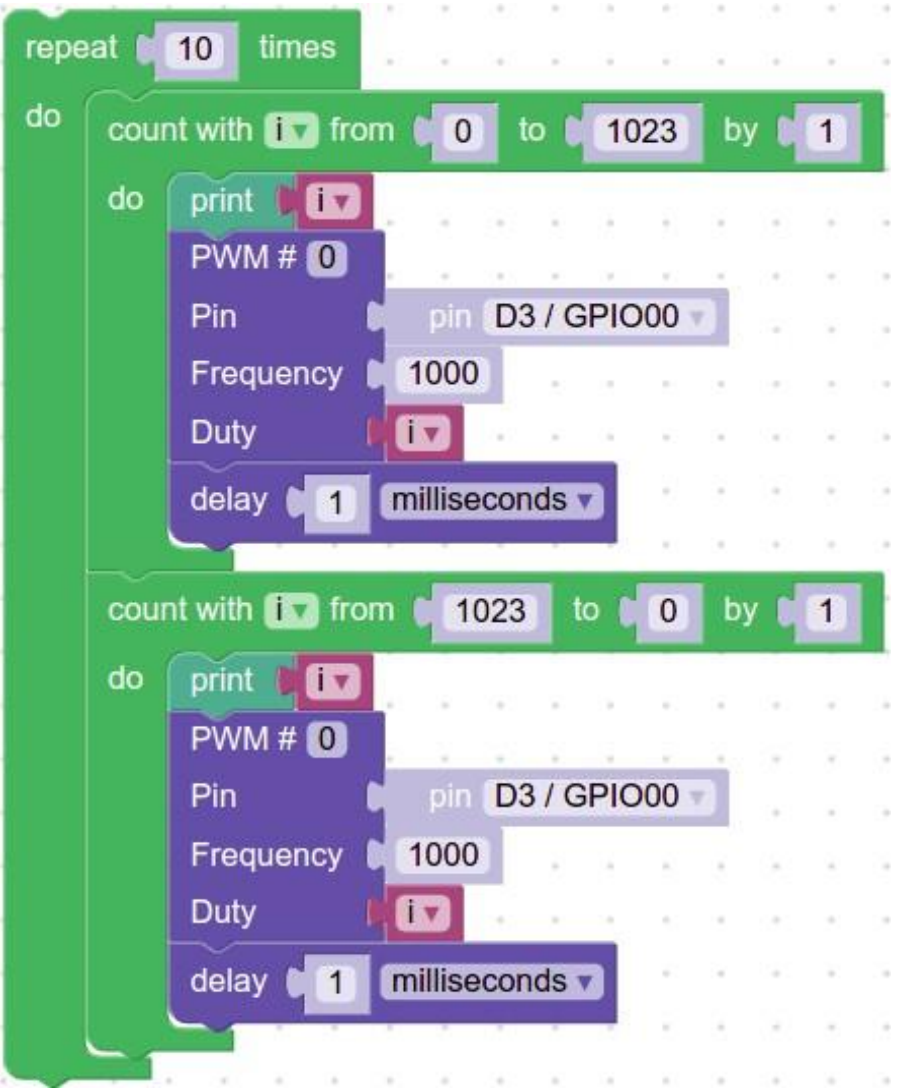
Montaje del circuito:

Precaución: Tenga en cuenta que el LED tiene polaridad, es decir, terminal positivo (el más largo) y terminal negativo (el más corto, junto al bisel en el LED). Por lo tanto, el terminal negativo del LED debe conectarse a G (tierra) y el terminal positivo a la resistencia, que está conectada al pin D3).



El PWM tiene dos parámetros principales: la frecuencia de la señal de onda cuadrada generada (Frequency) y el ciclo de trabajo (Duty). En este momento, nos interesa el ciclo de trabajo. Este parámetro indica la porción de la onda cuadrada que está en un nivel alto (encendido), lo cual puede variar de 0 a 100%. Así, un ciclo de trabajo del 100% significa máxima potencia, 50% equivale a la mitad de la potencia y 0% significa sin potencia. En el ESP8266, el PWM tiene una resolución de 10 bits (yendo de 0 para 0% hasta 1023 para 100%).

El siguiente programa hace que el LED aumente gradualmente su brillo desde 0 hasta el máximo y lo disminuya progresivamente hasta 0, repitiendo este proceso diez veces. Observa cómo, en esta aplicación, el valor del parámetro **duty** varía con el conteo de la variable i.



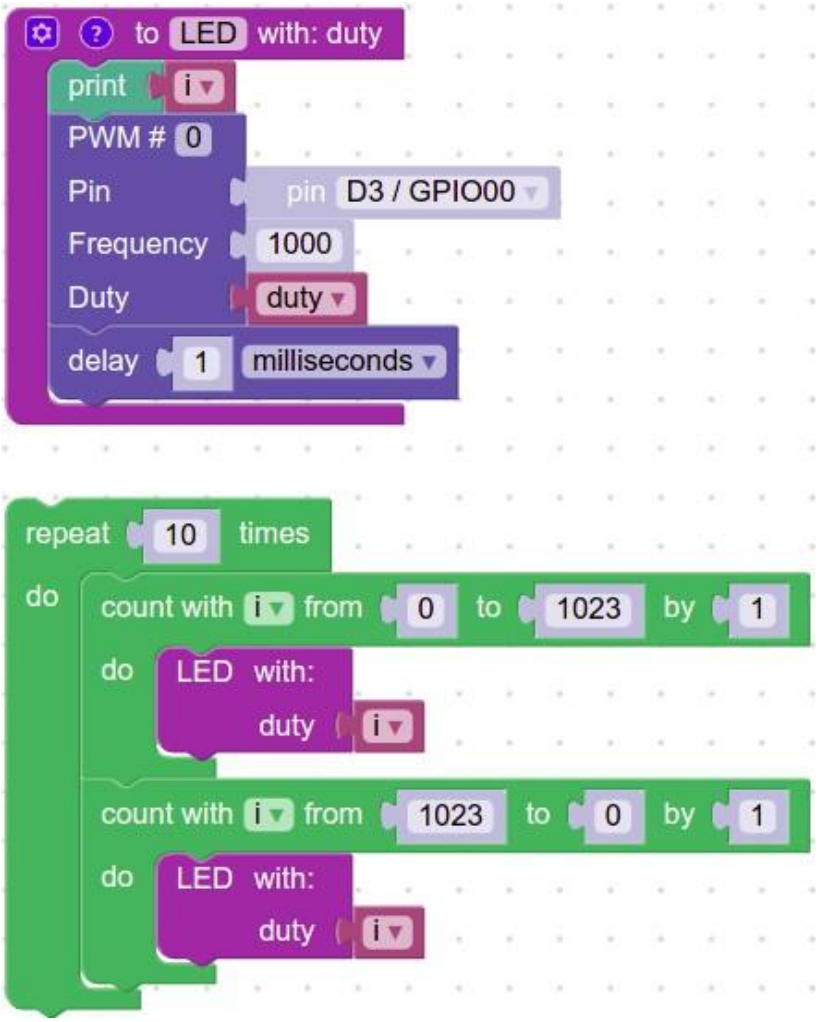
El programa anterior hace que el LED aumente su brillo de manera suave y lenta desde 0 hasta el máximo, y luego reduzca su brillo de forma gradual hasta 0, repitiendo este proceso diez veces.

Actividad:
El programa anterior hace que el LED aumente su brillo de manera suave y lenta desde 0 hasta el máximo, y luego reduce su brillo suavemente hasta 0, repitiendo este proceso diez veces.

Tip: Utiliza el bloque **subscribe to a topic** de los bloques EasyMQTT. Se ofrecen más detalles sobre esta práctica en la siguiente sección: “Control de dispositivos a través de Internet”.

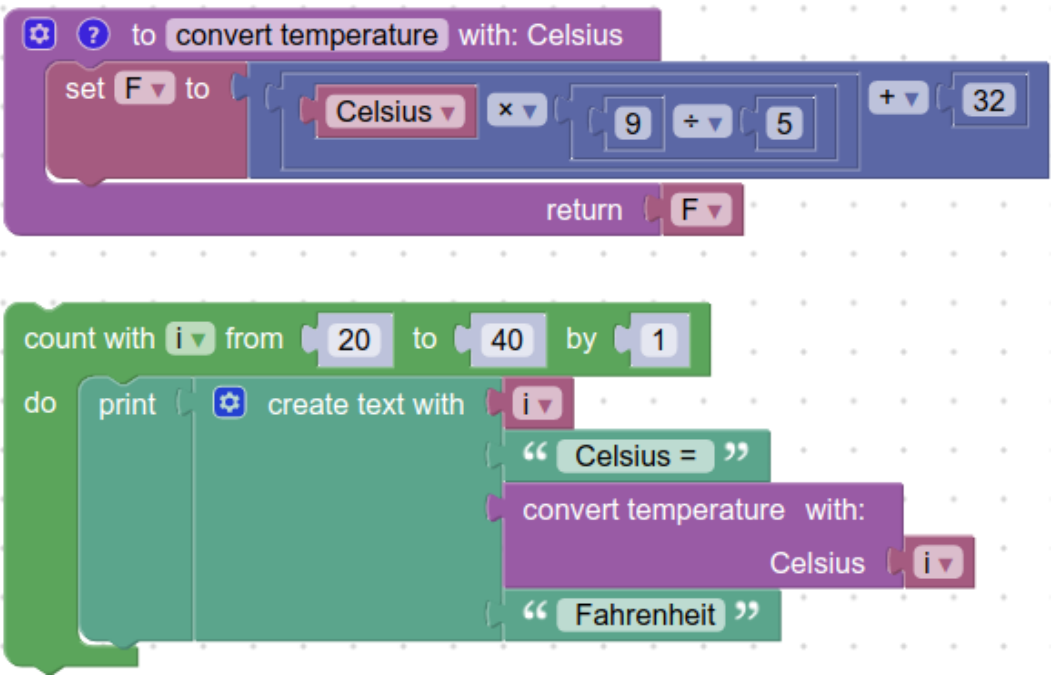
BIPES: Subrutinas / funciones

A medida que tu programa adquiere nuevas funciones, algunos fragmentos de código pueden ser útiles para reutilizar, lo que hace que valga la pena crear bloques reutilizables: estos son las funciones/subrutinas que pueden desarrollarse con las “funciones (**functions**)” de la caja de herramientas lateral. Si volvemos al ejemplo anterior de PWM y LED, nota que tres funciones se repiten dos veces. Estas tres funciones podrían agruparse en una sola función y reutilizarse varias veces, como en el siguiente ejemplo:



En el ejemplo anterior, hemos creado la función LED en el área de bloques superior, la cual recibe un parámetro duty que se utiliza internamente en esos bloques. Luego, en los bloques inferiores, que constituyen el programa principal, llamamos a la función LED con el parámetro deseado (valor de la variable i). En cada llamada, se ejecutan todos los bloques dentro de la función.

Puedes crear diversas funciones, de diferentes niveles de complejidad, con o sin recepción de parámetros, y crear funciones que devuelvan valores. Por ejemplo, se podría preparar y utilizar una función para convertir unidades de temperatura de la siguiente manera:



Y el resultado en la Consola será:

```
===
20 Celsius = 68.0 Fahrenheit
21 Celsius = 69.8 Fahrenheit
22 Celsius = 71.6 Fahrenheit
23 Celsius = 73.4 Fahrenheit
24 Celsius = 75.2 Fahrenheit
25 Celsius = 77.0 Fahrenheit
26 Fahrenheit = 78.8 Fahrenheit
27 Fahrenheit = 80.6 Fahrenheit
28 Fahrenheit = 82.4 Fahrenheit
29 Fahrenheit = 84.2 Fahrenheit
30 Fahrenheit = 86.0 Fahrenheit
31 Fahrenheit = 87.8 Fahrenheit
32 Fahrenheit = 89.6 Fahrenheit
33 Fahrenheit = 91.4 Fahrenheit
34 Fahrenheit = 93.2 Fahrenheit
35 Celsius = 95.0 Fahrenheit
36 Celsius = 96.8 Fahrenheit
37 Celsius = 98.6 Fahrenheit
38 Celsius = 100.4 Fahrenheit
39 Celsius = 102.2 Fahrenheit
40 Celsius = 104.0 Fahrenheit
>>>
```

Después de crear las funciones, puedes compartir tu programa con el botón y usa las funciones creadas en otros programas!

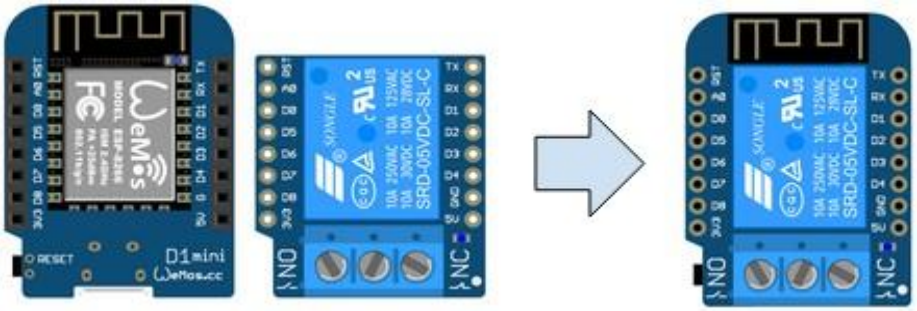
Control de dispositivos a través de Internet

En este ejercicio, utilizaremos un relé y enchufes eléctricos para encender o apagar cualquier dispositivo conectado a la toma de corriente (red eléctrica) mediante Web/Wifi. Para fines de prueba, sugerimos controlar una bombilla. De esta forma, el dispositivo puede ser una lámpara, un ventilador, una bomba de agua, un calefactor u otro aparato cuya potencia sea compatible con el relé. Por supuesto, necesitarás una red o router Wifi para la conexión. Sugerencia: investiga cómo funciona el relé (**relay**) y el histórico del primer "bug" informático.

Precaución:
Esta actividad incluye conectar el sistema a la red eléctrica con voltajes de 127 Voltios o 220 Voltios, lo que presenta un riesgo de descarga eléctrica u otras lesiones. Proceda solo si se siente seguro o consulte a un técnico especializado antes de conectar el sistema a la red eléctrica.

El relé es un dispositivo electromecánico que permite controlar aparatos eléctricos a partir de señales digitales. Así, la placa ESP8266 activa un módulo de relé, accionando una carga conectada a una toma de corriente. Para esta actividad, es necesario conectar el relé a la placa ESP8266 y la carga eléctrica externa al relé.

Una posibilidad es utilizar un módulo de relé del tipo "shield", que se ajusta directamente a la placa ESP8266 (modelo WeMos D1 mini), sin necesidad de cables de conexión, como en el ejemplo siguiente, donde las dos placas mostradas pueden apilarse, encajándolas a través de sus terminales de conexión.



Hay tres terminales de tornillo en la placa del relé, donde se puede conectar la carga que se va a controlar. Cabe señalar aquí que el relé funciona como un interruptor controlado electrónicamente, ya que tiene tres terminales de control de carga: C (Común), Normalmente Abierto (NO) y Normalmente Cerrado (NC). Debe usar 2 de estos terminales juntos (C con NO) o (C con NC). La combinación que elija definirá si el relé enciende o apaga la carga cuando sea activado por el ESP8266.

También es posible utilizar un módulo de relé simple o doble (lo que permite controlar dos o más dispositivos) conectado a la placa ESP mediante cables jumper.

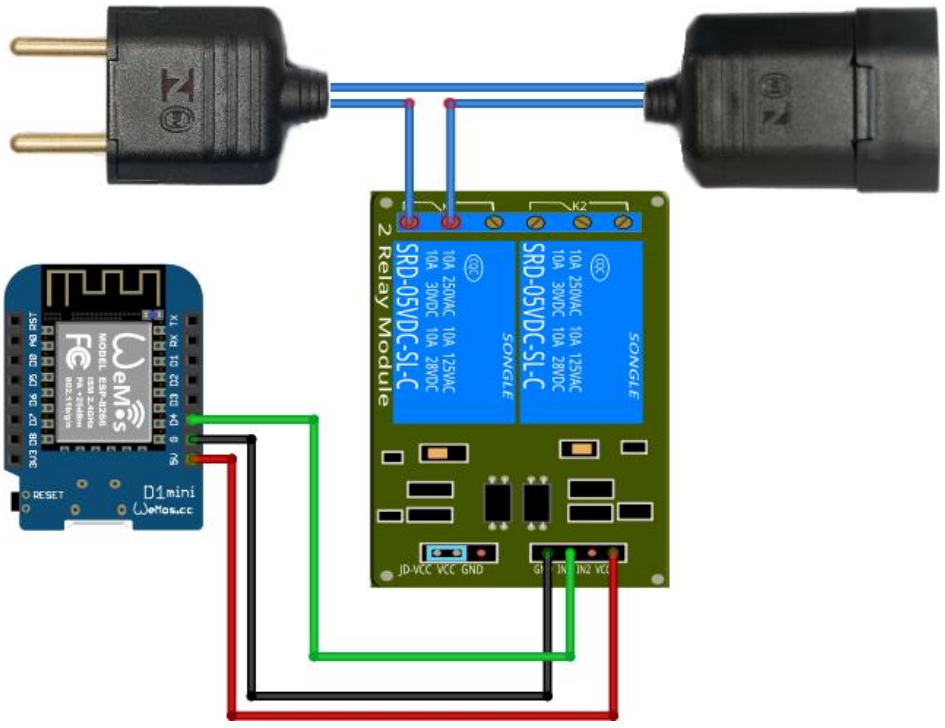
Conexiones de relé con el módulo ESP8266:

Placa relé	Placa ESP8266
S o IN1 de la placa relé	Pin D4 del ESP8266 (señal de control)
+ de la placa relé	Pin VIN (5 voltios) del ESP8266 (alimentación)
- de la placa relé	Pin GND (earth) del ESP8266

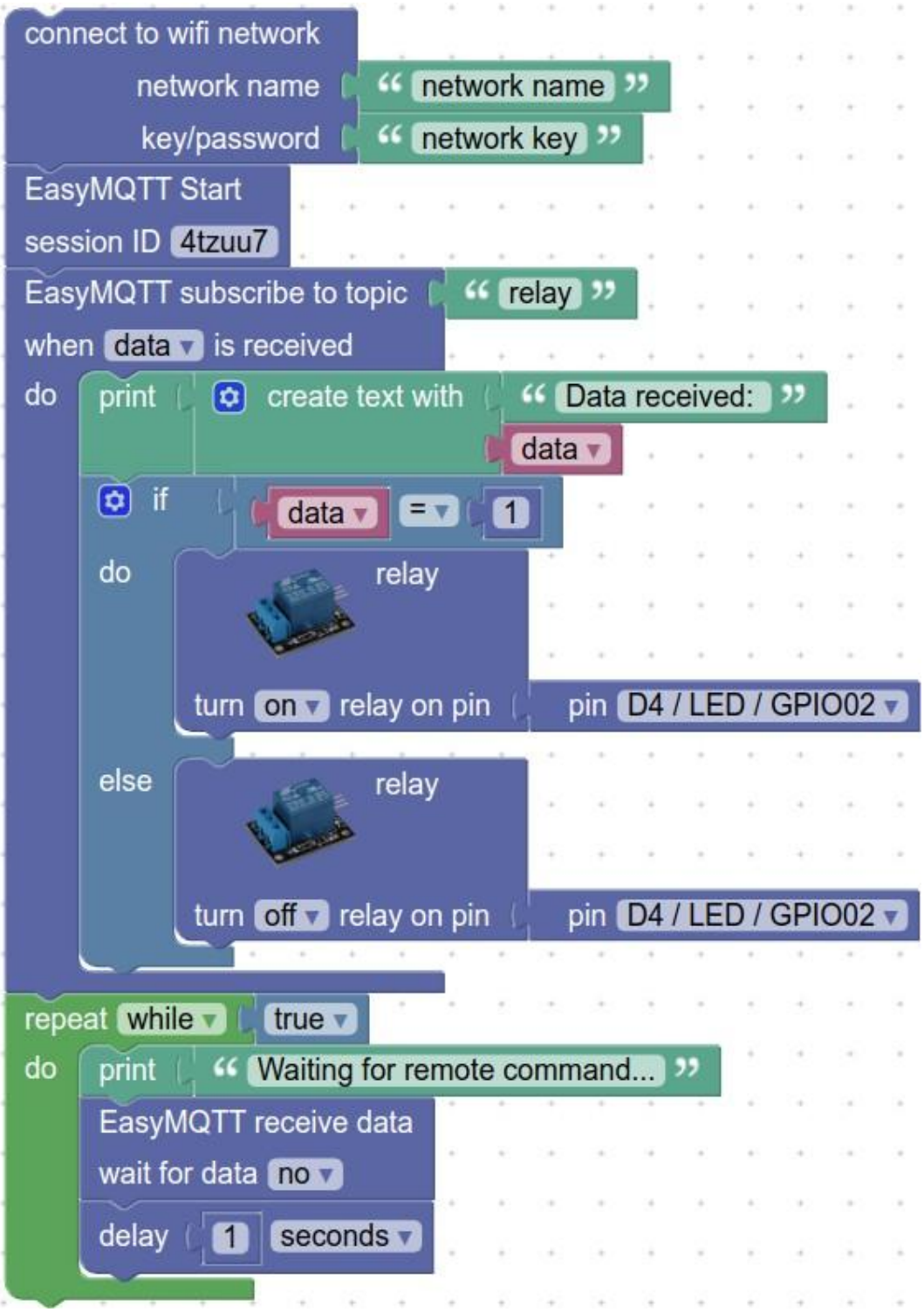
Atención: Todo el procedimiento de montaje debe realizarse con los dispositivos desconectados de la red eléctrica.

El diagrama a continuación ilustra un conjunto de enchufe y toma de dos cables, similar a una extensión de cable eléctrico. Un cable conecta directamente el enchufe y la toma, mientras que el otro cable pasa a través del relé (normalmente abierto: pines C y NO; o normalmente cerrado: pines C y NC).

Revise las conexiones cuidadosamente y, si es necesario, solicite la ayuda de un técnico en electrónica o electricidad. Luego, después de una verificación y validación minuciosas del ensamblaje, conecte la carga al enchufe (utilizaremos una lámpara en esta aplicación) y, finalmente, conecte el enchufe de alimentación a la red eléctrica.



Recuerda que el propósito de esta práctica es activar la bombilla a través de Internet. Por lo tanto, el programa en bloques para esta aplicación es el siguiente:



Para probar el programa, ejecútalo y ve a la pestaña EasyMQTT. Escribe “relay”, completa los datos con 0 o 1 y envía la información. Luego, verifica si el relé o el LED cambia su estado entre encendido y apagado según tus comandos.

Tenga en cuenta que el control remoto puede realizarse desde cualquier dispositivo conectado a Internet que conozca la sesión a través de una dirección web. BIPES también ofrece una Interfaz de Programación de Aplicaciones (API) con solicitudes web (HTTP) para integrar otras aplicaciones. Detallaremos este tópico en la siguiente sección.

Por ahora, vale la pena mencionar que la API HTTP de BIPES para integración permite que otros sistemas modifiquen los valores de los tópicos de EasyMQTT utilizando solicitudes HTTP. La actualización se realiza a través de la página **publish.php**, donde se deben proporcionar la sesión, el tópico y el valor deseados. Por ejemplo, para la sesión **4tzuu7** y el tópico **relay**, se pueden utilizar las siguientes solicitudes:

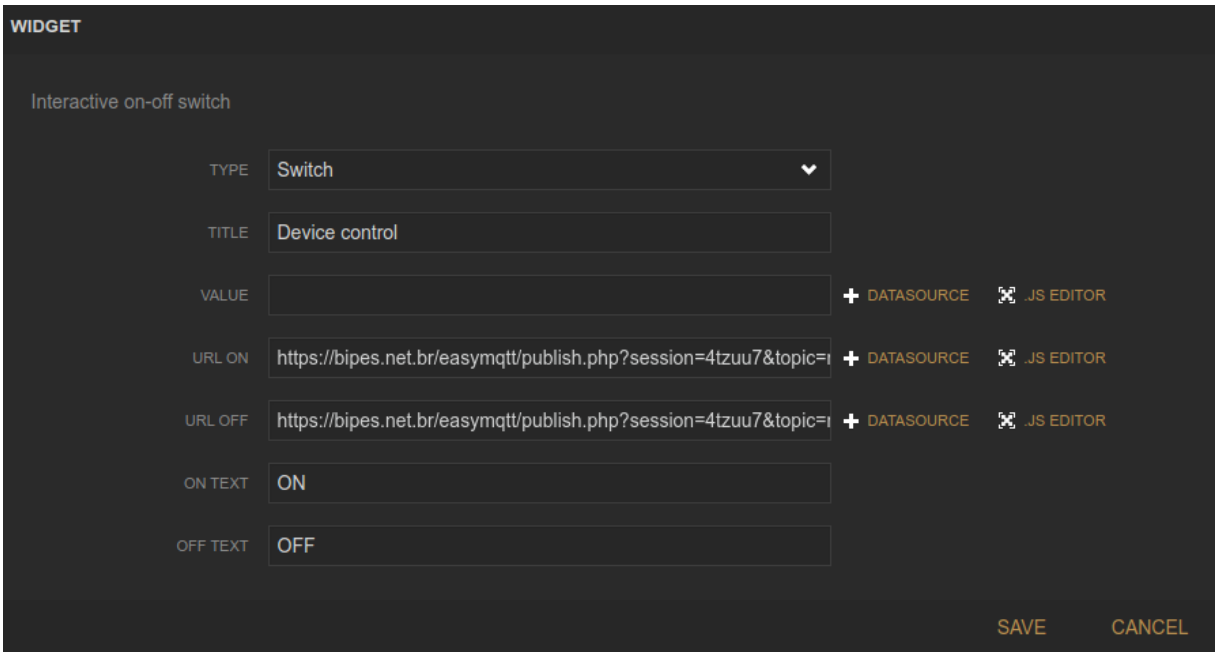
Para encender el dispositivo:

http://bipes.net.br/easymqtt/publish.php?session=4tzuu7&topic=relay&value=1

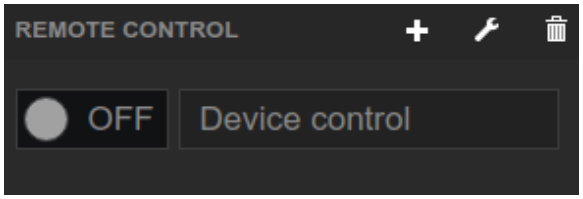
Para apagar el dispositivo:

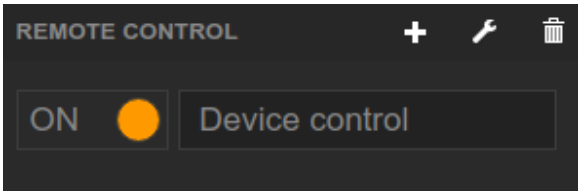
http://bipes.net.br/easymqtt/publish.php?session=4tzuu7&topic=relay&value=0

Tenga en cuenta que cada programa tendrá una sesión diferente (**4tzuu7in** en este caso). ¡Ajuste la sesión o cree la suya propia! Puede incluir estas direcciones web en las acciones de botones en la pestaña de IoT. Además, verifique si su acceso a BIPES fue a través de HTTP o HTTPS (revise la URL en el navegador). Esas direcciones de encendido/apagado mencionadas anteriormente deben comenzar con http o https, dependiendo de cómo se haya accedido a BIPES. Por ejemplo, la configuración de un widget Switch en el panel de freeboard (pestaña IOT) se vería así:



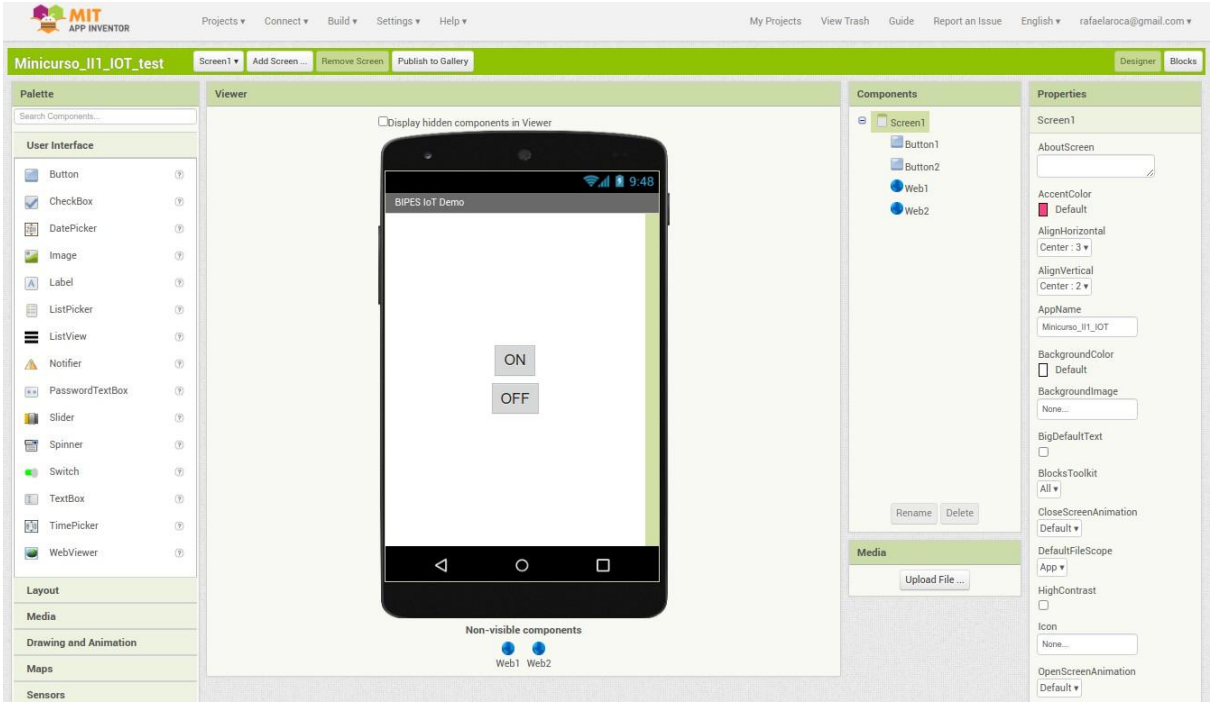
Ahora, al hacer clic en los botones de la pestaña IoT, controlarás el dispositivo conectado al relé, como por ejemplo una bombilla:





Si compartes este programa con otra persona a través del enlace, usando el botón , esa persona tendrá acceso al panel de control desde cualquier parte del mundo y podrá controlar el dispositivo de forma remota.

También puedes usar MIT App Inventor (http://ai2.appinventor.mit.edu/) para crear una aplicación para teléfonos inteligentes que controle el dispositivo. La aplicación completa se puede crear en línea utilizando el sitio web http://ai2.appinventor.mit.edu/:



Después de crear una cuenta e iniciar sesión en MIT App Inventor, en la pantalla del diseñador, crea un nuevo programa e incluye dos botones: ON y OFF, como se muestra en la figura. Si lo deseas, puedes agregar imágenes, ajustar tamaños, colores y textos para que la interfaz de usuario de tu programa sea más atractiva.

En bloques (haz clic en el botón en la parte superior derecha para acceder a “blocks”), incluye dos bloques que definan las acciones de los botones, cada uno configurado para realizar solicitudes HTTP a los enlaces mencionados anteriormente:

¡Hecho! Los bloques anteriores construyen una aplicación completa para teléfonos Android, que te permite encender y apagar dispositivos de forma remota. Observa lo similar que es a BIPES y lo fácil que es de usar. De hecho, tanto BIPES como MIT App Inventor están basados en el lenguaje de programación Google Blockly. Por eso los bloques son tan parecidos.

Actividad:

Recuerda el ejemplo mencionado en la introducción de este libro: el control de encendido/apagado de un refrigerador. ¡Ya tienes los conocimientos y las herramientas necesarias para implementar este tipo de control!

Utiliza la placa ESP8266 o ESP32 y el montaje con el relé, realizado en el ejercicio anterior, junto con un sensor de temperatura y humedad DHT11. Luego, prepara un programa que encienda un relé, encargado de activar un ventilador cuando la temperatura supere los 30 grados Celsius, y apague el relé cuando la temperatura sea inferior a 25 grados Celsius.

Después de probar el sistema, incluye la funcionalidad para monitorear la temperatura de forma remota. Agrega la opción de ajustar, a través del panel de control, la temperatura mínima y máxima que mantiene activado el relé que controla el ventilador.

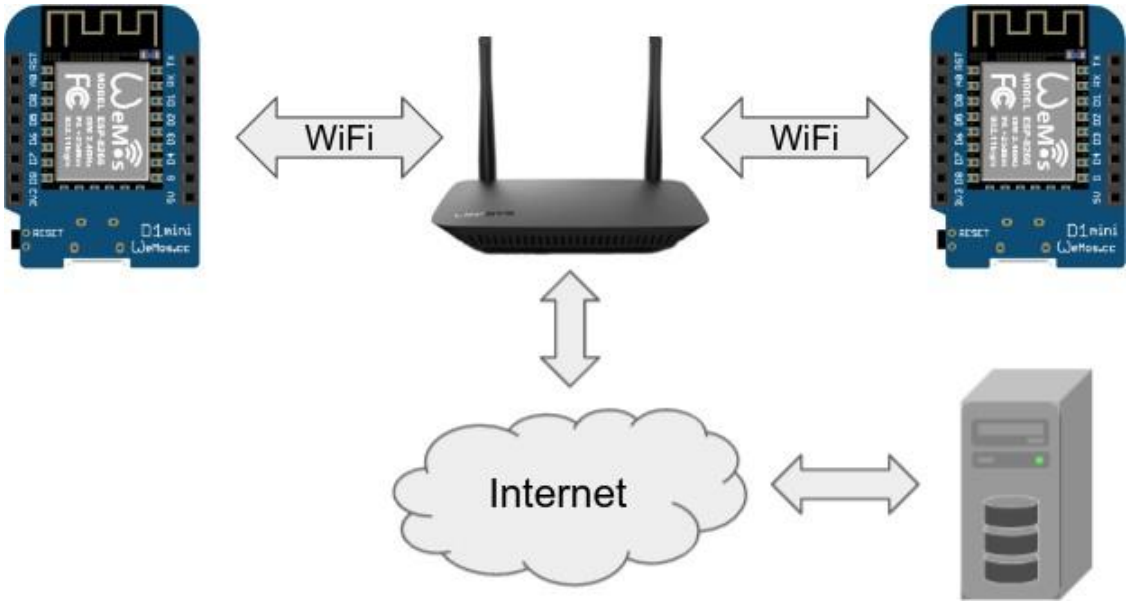
Cliente web y servidor web(HTTP)

Hoy en día, una actividad muy común es acceder a sitios web como http:///www.bipes.net.br/. Para este tipo de acceso, un navegador web como Google Chrome, por ejemplo, utiliza el protocolo de transferencia de hipertexto - HyperText Transfer Protocol (HTTP). El intercambio de información a través de HTTP es iniciado por un dispositivo cliente que comienza el proceso mediante una solicitud enviada al servidor. La siguiente figura ilustra una posible situación en la que una placa ESP8266 actúa como cliente, realizando una solicitud a un servidor a través de Internet. Cabe señalar que este escenario también podría invertirse, donde la ESP8266 podría actuar como servidor.

También son posibles otras arquitecturas (opciones de conexión), incluyendo escenarios sin conexión a Internet. Por ejemplo, la siguiente figura muestra la posibilidad de que dos placas ESP8266 intercambien datos directamente entre sí. En esta situación, una placa debe configurarse en modo punto de acceso, proporcionando una red WiFi para la conexión, y otra como una estación WiFi. Además, una tarjeta actúa como servidor HTTP y la otra como cliente HTTP. Este tipo de red inalámbrica se llama ad-hoc.

Se puede lograr una arquitectura similar utilizando un punto de acceso inalámbrico, que proporciona una red WiFi para conectar las dos placas. Desde el punto de vista del intercambio de mensajes HTTP, el funcionamiento es el mismo que el descrito anteriormente. Esta arquitectura de conexión se denomina modo infraestructura, ya que el punto de acceso inalámbrico se considera un equipo de infraestructura.

En algunas configuraciones de red, las dos placas pueden no comunicarse directamente. Por lo tanto, también es posible que dos sistemas embebidos se comuniquen a través de un servidor. Cada una de las placas ilustradas en la figura a continuación realiza solicitudes a un servidor. Además, las placas pueden publicar o consumir datos de este servidor para dejar mensajes que estarán disponibles para que otras aplicaciones en otras placas o dispositivos los consulten.



Para realizar la solicitud HTTP, el cliente necesita conocer el localizador uniforme de recursos -Uniform Resource Locator (URL), que, en resumen, tiene la siguiente estructura:

http://address:port/path/resource?query_string

Ya se ha utilizado un ejemplo de URL real en este libro en la actividad de activar un dispositivo a través de Internet:

http://bipes.net.br/easymqtt/publish.php?session=4tzuu7&topic=rele&value=1

Detallando cada parte de la URL:

Parte	Descripción
http://	Protocolo: HTTP o HTTPS (seguro, con transmisión encriptada) En el ejemplo: **http://**
address	Dirección del servidor en Internet. Esto podría, por ejemplo, tener el formato de una dirección IP o un nombre, como 34.95.149.23 o bipes.net.br En el ejemplo: **bipes.net.br**
:port	El puerto es un parámetro opcional, que normalmente se omite. Cuando se omite, se entiende que el puerto es 80, el puerto predeterminado para los servicios web. Utilizar otros valores de puerto es útil cuando más de un servidor web está asociado con la misma IP o dirección, o cuando algún firewall o sistema de seguridad bloquea el puerto 80. En el ejemplo, se omite el puerto.
/path/	La ruta al recurso en el servidor. En el ejemplo: **/easymqtt/**
resource	El nombre del recurso que se utilizará/accederá en ese servidor. En el ejemplo: **publish.php**
?query_string	El query_string te permite enviar parámetros al recurso. Se pueden incluir varios parámetros en el query_string, separados por el símbolo &. En el ejemplo: **?session=4tzuu7&topic=rele&value=1**

Cliente HTTP

Acceder a servicios web a través de HTTP no se trata solo del contenido. El protocolo HTTP se ha convertido en una forma estándar de acceder a miles de servicios y enviar y recibir datos entre dispositivos. Aquí, un concepto interesante es M2M o Máquina a Máquina, que significa comunicación directa entre máquinas. La URL para acceder a un servicio en los servicios web también se llama **endpoint**.

Conociendo la URL del servicio web que se utilizará, un cliente web puede realizar

solicitudes HTTP para acceder a contenido y ejecutar acciones asociadas con ese servicio web. Los clientes HTTP más comunes son los navegadores web. Por lo tanto, es posible utilizar Google Chrome o Firefox para acceder y probar las URLs de servicios web que se mencionan aquí.

En este sentido, se pueden utilizar servicios web gratuitos y de pago a través de HTTP, disponibles en un formato llamado Interfaz de Programación de Aplicaciones (API). Así, existen miles de opciones de servicios accesibles para clientes HTTP, como consultar el pronóstico del clima, realizar operaciones financieras, obtener cotizaciones de monedas y acciones, efectuar pagos, enviar mensajes por SMS, publicar mensajes en Twitter, enviar mensajes por Telegram, realizar consultas de geolocalización, controlar dispositivos de forma remota, acceder a datos de sensores, entre otros. Además, el sitio web https://any-api.com/ enumera más de 1400 API para la posible integración en otras aplicaciones.

Cliente HTTP: Envío de mensajes SMS

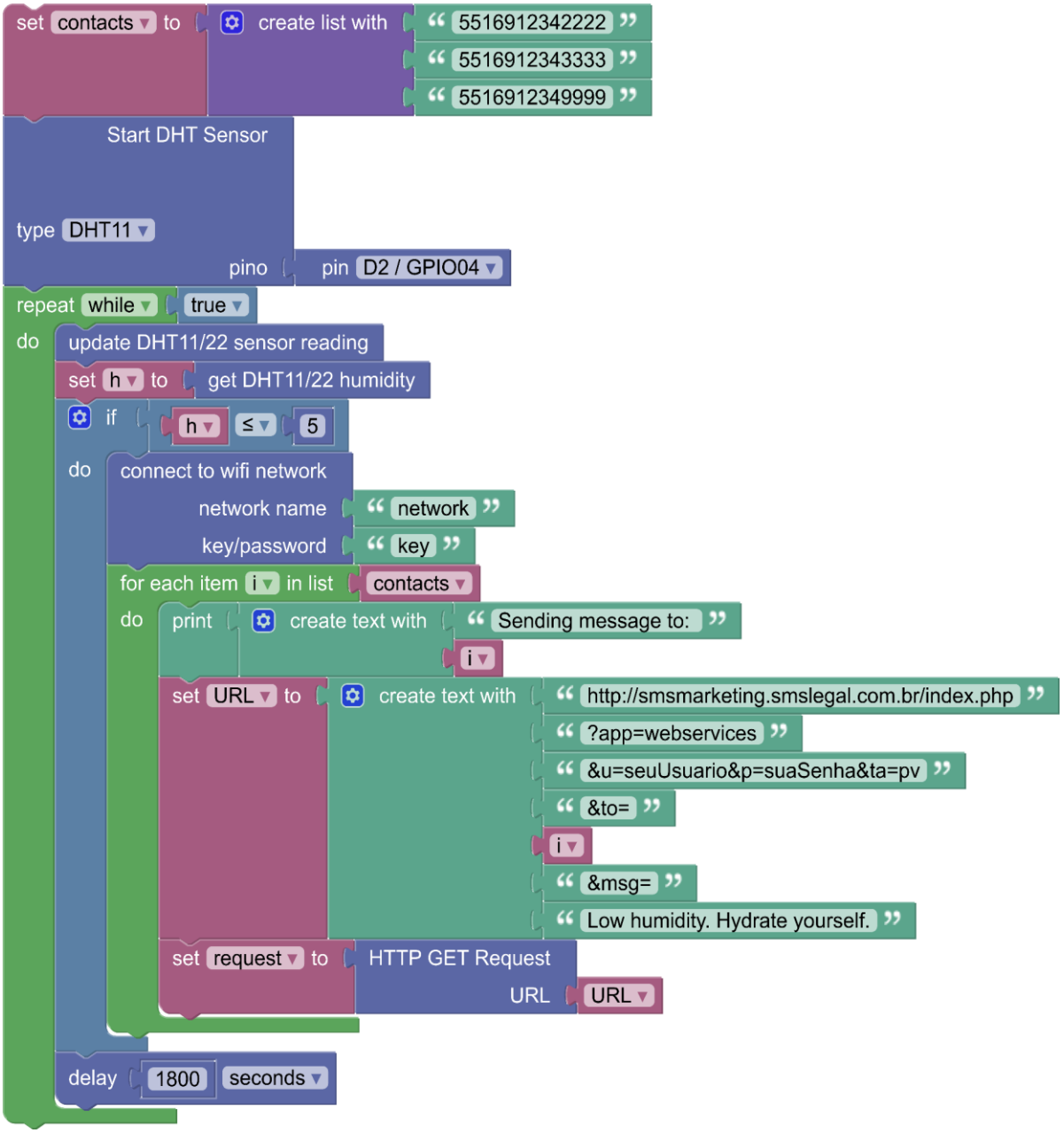
La siguiente figura muestra un ejemplo de un programa que envía mensajes SMS a una lista de teléfonos, advirtiendo sobre baja humedad cuando un sensor DHT11 mide una humedad relativa inferior al 5%. El sistema propuesto no cuenta con ningún módulo electrónico de teléfono celular adicional ni con una tarjeta SIM conectada a la placa ESP8266 para enviar mensajes SMS. En su lugar, el envío del mensaje SMS se realiza a través de un servicio web proporcionado por el siguiente recurso HTTP (endpoint): http://smsmarketing.smslegal.com.br/index.php. Al acceder a este recurso, es posible enviar parámetros (como el teléfono de destino y el mensaje) a través del query_string de la URL. Por ejemplo:

http://smsmarketing.smslegal.com.br/index.php?app=webservices&u=seuUsuario&p=suaSenha&ta=pv&to=5516997970000&msg=BIPES

La URL anterior envía un mensaje con “BIPES” al número de teléfono 5516997970000. Este servicio web es de pago, por lo que tanto este como otros servicios web requieren autenticación. Por esta razón, se pueden utilizar los parámetros u y p para informar el nombre de usuario y la contraseña.

El ejemplo mostrado está enfocado en teléfonos móviles de Brasil. Una búsqueda rápida en https://any-api.com/ para “SMS” arroja varios resultados de APIs de SMS. La primera que aparece, por ejemplo, es “zoomconnect”, que proporciona el endpoint de la API https://www.zoomconnect.com/app/api/rest/v1/sms/send para enviar SMS a una gran cantidad de países a bajo costo.

De todos modos, como se mencionó, la página siguiente muestra un ejemplo de cómo utilizar una solicitud HTTP para enviar mensajes SMS a varios teléfonos:



Cliente HTTP: Cambiando el color del gallo del clima

Quizás recuerdes esos regalos de “Gallo del Tiempo Portugués que Cambia de Color”, que cambian de color según las condiciones climáticas. Podemos construir una versión IoT de este gallo usando una placa ESP y un LED RGB de tres colores (rojo, verde y azul). La información climática se puede obtener de varias fuentes. Un ejemplo es el Instituto Nacional de Investigaciones Espaciales de Brasil del Ministerio de Ciencia, Tecnología e Innovación (INPE/MCTI), que ofrece un servicio web para consultar información climática. Además, es posible conectar la placa ESP a una batería externa, convirtiendo este gallo IoT en un artículo de decoración funcional.

Considerado un referente en investigación espacial, el INPE/MCTI es la principal fuente de información climática y pronósticos meteorológicos de Brasil. Además, el Centro de Previsión del Tiempo y Estudios Climáticos (CPTEC) del INPE/MCTI ofrece un servicio web gratuito para consultar información climática y pronósticos del tiempo para varias ciudades brasileñas. Más información sobre el servicio y sus términos de uso está disponible en: http://servicos.cptec.inpe.br/XML/ (información solo en portugués, disculpe).

De todos modos, existen servicios similares en la mayoría de los países. En Estados Unidos, por ejemplo, el Servicio Meteorológico Nacional también ofrece un servicio web para consultar datos meteorológicos. La información sobre este servicio web, sus puntos finales y cómo utilizarlos se puede obtener en este enlace: https://www.weather.gov/documentation/services-web-api#.

Una tercera y más amplia posibilidad es Open Weather Map (http://openweathermap.org/), que ofrece varios servicios web para obtener el clima actual y pronósticos, con opciones gratuitas y de pago.

Usando datos del INPE/MCTI (http://servicos.cptec.inpe.br)

Para el servicio web brasileño (INPE/MCTI), el primer paso para utilizar este servicio es verificar el código (ID) de la ciudad para la cual desea obtener información. Para este propósito, el servicio web http://servicos.cptec.inpe.br/XML/listaCidades ofrece una lista de las ciudades cubiertas por el servicio y sus respectivos códigos. Esta lista se proporciona en XML (Extensible Markup Language), un estándar internacional para el intercambio de información entre máquinas y servicios web.

Para este ejemplo, utilizaremos el caso de la ciudad de São Carlos - SP, un polo brasileño de ciencia y tecnología[3]. El código para la ciudad de São Carlos - SP es 4774. Así, es posible utilizar el recurso previsao.xml: http://servicos.cptec.inpe.br/XML/cidade/4774/previsao.xml. El acceso a este *endpoint* genera el siguiente resultado XML:

```
<cidade>
  <nome>São Carlos</nome>
  <uf>SP</uf>
  <atualizacao>2022-12-22</atualizacao>
  <previsao>
    <dia>2022-12-22</dia>
    <tempo>ci</tempo>
    <maxima>29</maxima>
    <minima>19</minima>
    <iuv>13.0</iuv>
  </previsao>
  <previsao>
    <dia>2022-12-23</dia>
    <tempo>ci</tempo>
    <maxima>27</maxima>
```

[3] Obtén más información sobre São Carlos - SP en https://www.reportsancahub.com.br/

```
    <minima>19</minima>
    <iuv>13.0</iuv>
  </previsao>
  <previsao>
    <dia>2022-12-24</dia>
    <tempo>c</tempo>
    <maxima>26</maxima>
    <minima>18</minima>
    <iuv>14.0</iuv>
  </previsao>
  <previsao>
    <dia>2022-12-25</dia>
    <tempo>ci</tempo>
    <maxima>24</maxima>
    <minima>18</minima>
    <iuv>14.0</iuv>
  </previsao>
</cidade>
```

Fuente: CPTEC/INPE (Acceso en Diciembre/2021)

El campo ***tempo*** ofrece la información necesaria para implementar nuestra aplicación entre la información disponible. ***Tempo*** es la palabra portuguesa para tiempo, pero también significa clima. ¡Sé feliz! Este libro incluso enseña un poco de portugués.

En formato XML, cada pieza de información está encerrada entre marcadores especiales. Por ejemplo, la información meteorológica siempre está delimitada entre <tempo> y </tempo>. En el ejemplo anterior, los datos más recientes indican: *<tempo>ci</tempo>.* La documentación de este servicio web del INPE/MCTI también describe las posibilidades de información meteorológica. La tabla a continuación muestra algunas de estas siglas. Consulte el enlace http://servicos.cptec.inpe.br/XML/ para obtener una lista completa de siglas y descripciones.

Código	Descripción
ci	Chubascos aislados
c	Lluvia
pc	Chubascos de lluvia
t	Tormenta
cl	Cielo despejado

Usando datos de Open Weather Map (http://openweathermap.org/)

OpenWeatherMap tiene varias API y endpoints con cientos de posibilidades. Nos interesa la API que devuelve el clima actual basado en una consulta que proporciona el nombre de la

ciudad sobre la que queremos información. Se puede encontrar más información, uso y documentación sobre este ***endpoint*** aquí: https://openweathermap.org/current#name. OpenWeatherMap permite hasta 60 llamadas a la API por minuto y 1,000,000 de llamadas a la API por mes en la versión gratuita. Para usar las APIs de OpenWeatherMap, necesitas crear una cuenta y generar una clave API para autorizar las solicitudes a la API: https://home.openweathermap.org/api_keys.

El punto final de nuestro interés es:

http://api.openweathermap.org/data/2.5/weather?q=Marrakesh&appid=KEY

Tenga en cuenta que la URL anterior incluye un parámetro q=Marrakesh, donde especificamos que queremos información para la ciudad de Marrakesh. Además, debe proporcionar una clave de API después de appid=. Simplemente genere una clave de API y reemplácela por la palabra KEY. La clave de API será algo similar a *1b1111124567fafabcdee3aaae02dce0*.

Aquí tienes algunos ejemplos de cómo utilizar dicha API:

API URL: **http://api.openweathermap.org/data/2.5/weather?q=Miami&appid=KEY**

Resultado para la ciudad de Miami:

{"coord":{"lon":-80.1937,"lat":25.7743},"weather":[{"id":801,"main":"Clouds",
"description":"few clouds",
"icon":"02n"}],"base":"stations","main":{"temp":295.62,"feels_like":296.01,"temp_min":294.
15,"temp_max":297.16,"pressure":1015,"humidity":80},"visibility":10000,"wind":{"speed":0.
45,"deg":225,"gust":1.34},"clouds":{"all":20},"dt":1641429138,"sys":{"type":2,"id":2009435,"
country":"US","sunrise":1641384498,"sunset":1641422623},"timezone":-18000,"id":416413
8,"name":"**Miami**","cod":200}

API URL: **http://api.openweathermap.org/data/2.5/weather?q=São Carlos&appid=KEY**

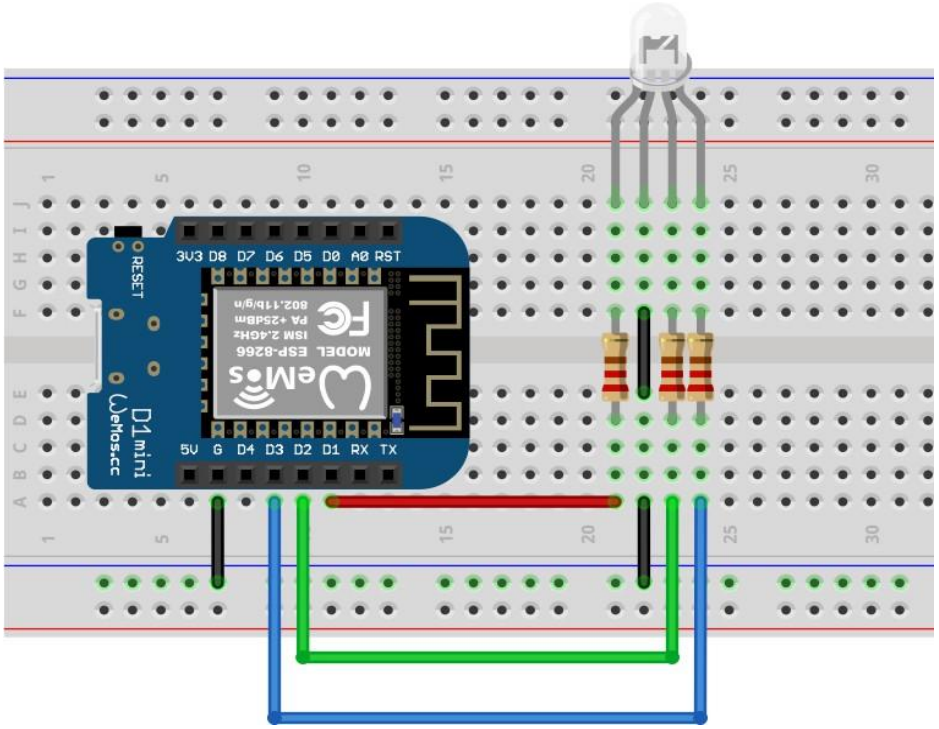
Resultado para la ciudad de São Carlos, SP, Brasil:

{"coord":{"lon":-47.8908,"lat":-22.0175},"weather":[{"id":500,"main":"Rain",
"description":"light rain",
"icon":"10n"}],"base":"stations","main":{"temp":295.61,"feels_like":296.34,"temp_min":295.
61,"temp_max":295.61,"pressure":1008,"humidity":93,"sea_level":1008,"grnd_level":915},"
visibility":10000,"wind":{"speed":1.48,"deg":348,"gust":1.53},"rain":{"1h":0.15},"clouds":{"all
":100},"dt":1641429064,"sys":{"type":1,"id":8462,"country":"BR","sunrise":1641371643,"sun
set":1641419968},"timezone":-10800,"id":3449319,"name":"**São Carlos**","cod":200}

Nota en los dos ejemplos que el campo “description” tiene la información que

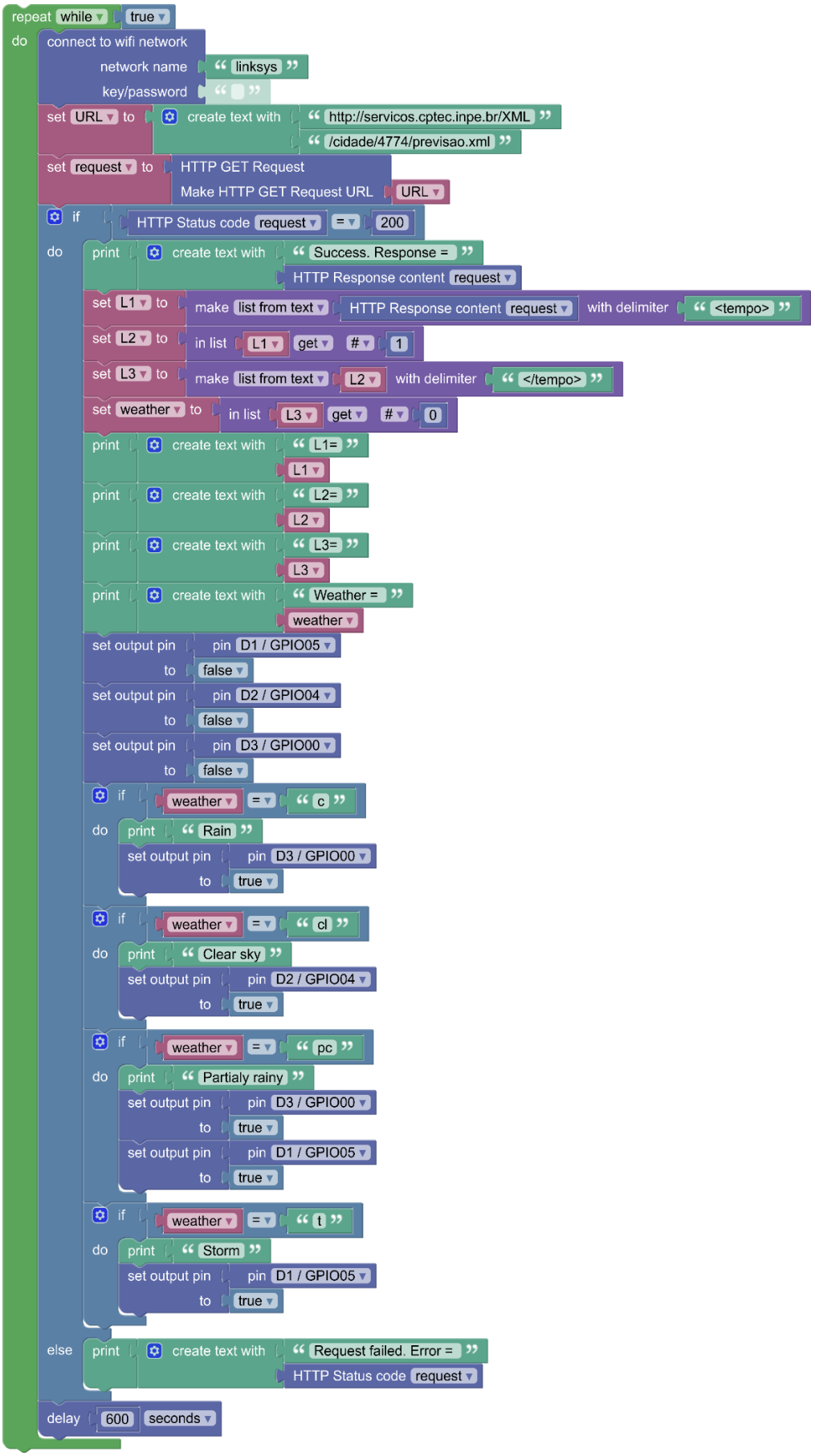
necesitamos: “few clouds” y “light rain”.

Basándose en la información proporcionada por las APIs mencionadas, es posible implementar el "Gallo del clima que cambia de color". La siguiente figura ilustra un posible montaje con un LED tricolor conectado al módulo ESP. Este LED tiene un terminal de tierra (GND) y tres terminales más, uno para cada color. Al controlar los pines relacionados con cada color del LED, es posible encender el LED en rojo, verde, azul o en combinaciones de estos colores.



Para el servicio INPE/MCTI: El programa presentado tiene un bucle de repetición infinito con una espera de 10 minutos en la página siguiente. Así, el sistema consulta al servidor INPE/MCTI cada 10 minutos y actualiza los colores de los LEDs según las condiciones meteorológicas de la ciudad de São Carlos. Justo después de realizar la solicitud HTTP GET al servidor, el programa verifica si el servidor respondió con el código 200, lo que significa que la solicitud fue recibida, procesada correctamente y la respuesta está disponible. En este caso, la respuesta es un texto en formato XML, como se ilustra arriba.

El programa luego realiza una secuencia de operaciones sobre las variables L1, L2, L3 y weather para obtener solo la abreviatura con la información sobre el clima. A continuación, se realizan varias pruebas condicionales IF para verificar algunas condiciones y encender los LED correspondientes: rojo (pin D1) para tormenta, verde (pin D2) para cielo despejado, azul (pin D3) para lluvia. También se implementó una posibilidad para la abreviatura pc (parcialmente lluvioso), la cual enciende simultáneamente los LED conectados a los pines D1 y D3 (rojo y azul), dando como resultado un color similar al morado.


repeat while true
do
connect to wifi network
network name linksys
key/password
set URL to create text with http://servicos.cptec.inpe.br/XML
/cidade/4774/previsao.xml
set request to HTTP GET Request
Make HTTP GET Request URL URL
if HTTP Status code request = 200
do
print create text with Success. Response =
HTTP Response content request
set L1 to make list from text HTTP Response content request with delimiter <tempo>
set L2 to in list L1 get # 1
set L3 to make list from text L2 with delimiter </tempo>
set weather to in list L3 get # 0
print create text with L1=
L1
print create text with L2=
L2
print create text with L3=
L3
print create text with Weather =
weather
set output pin pin D1 / GPIO05
to false
set output pin pin D2 / GPIO04
to false
set output pin pin D3 / GPIO00
to false
if weather = c
do print Rain
set output pin pin D3 / GPIO00
to true
if weather = cl
do print Clear sky
set output pin pin D2 / GPIO04
to true
if weather = pc
do print Partialy rainy
set output pin pin D3 / GPIO00
to true
set output pin pin D1 / GPIO05
to true
if weather = t
do print Storm
set output pin pin D1 / GPIO05
to true
else print create text with Request failed. Error =
HTTP Status code request
delay 600 seconds

La siguiente lista muestra un ejemplo de la salida de la Consola al ejecutar el programa. Ten en cuenta que imprimir los valores de las variables L1, L2, L3 y weather no sería necesario en la versión de producción. En su lugar, añadimos estas impresiones para mostrar cada paso en la obtención de la información deseada mediante los bloques de manipulación de listas. Algunos fragmentos se han resaltado en negrita para facilitar la comprensión de los mensajes que aparecen a continuación.

```
Waiting for Wifi connection Connected

Success. Response = b"<?xml version='1.0' encoding='ISO-8859-1'?><cidade><nome>São
Carlos</nome><uf>SP</uf><atualizacao>2021-12-22</atualizacao><previsao><dia>202 1-
12-22</dia><tempo>c</tempo><maxima>29</maxima><minima>19</minima><iuv>13
.0</iuv></previsao><previsao><dia>2021-12-23</dia><tempo>ci</tempo><maxima>27</
maxima><minima>19</minima><iuv>13.0</iuv></previsao><previsao><dia>2021-12-24</
dia><tempo>c</tempo><maxima>26</maxima><minima>18</minima><iuv>14.0</iuv></p
revisao><previsao><dia>2021-12-25</dia><tempo>ci</tempo><maxima>24</maxima><
minima>18</minima><iuv>14.0</iuv></previsao></cidade>"

L1=['b"<?xml version=\'1.0\' encoding=\'ISO-8859-1\'?><cidade><nome>São
Carlos</nome><uf>SP</uf><atualizacao>2021-12-22</atualizacao><previsao><dia>2021
-12-22</dia>',
'c</tempo><maxima>29</maxima><minima>19</minima><iuv>13.0</iuv></previsao><pr
evisao><dia>2021-12-23</dia>',
'ci</tempo><maxima>27</maxima><minima>19</minima><iuv>13.0</iuv></previsao><pr
evisao><dia>2021-12-24</dia>',
'c</tempo><maxima>26</maxima><minima>18</minima><iuv>14.0</iuv></previsao><pr
evisao><dia>2021-12-25</dia>',
'ci</tempo><maxima>24</maxima><minima>18</minima><iuv>14.0</iuv></previsao></ci
dade>"']

L2=c</tempo><maxima>29</maxima><minima>19</minima><iuv>13.0</iuv></previsao>
<previsao><dia>2021-12-23</dia>

L3=['c',
'<maxima>29</maxima><minima>19</minima><iuv>13.0</iuv></previsao><previsao><di
a>2021-12-23</dia>']

Weather = c Rain
```

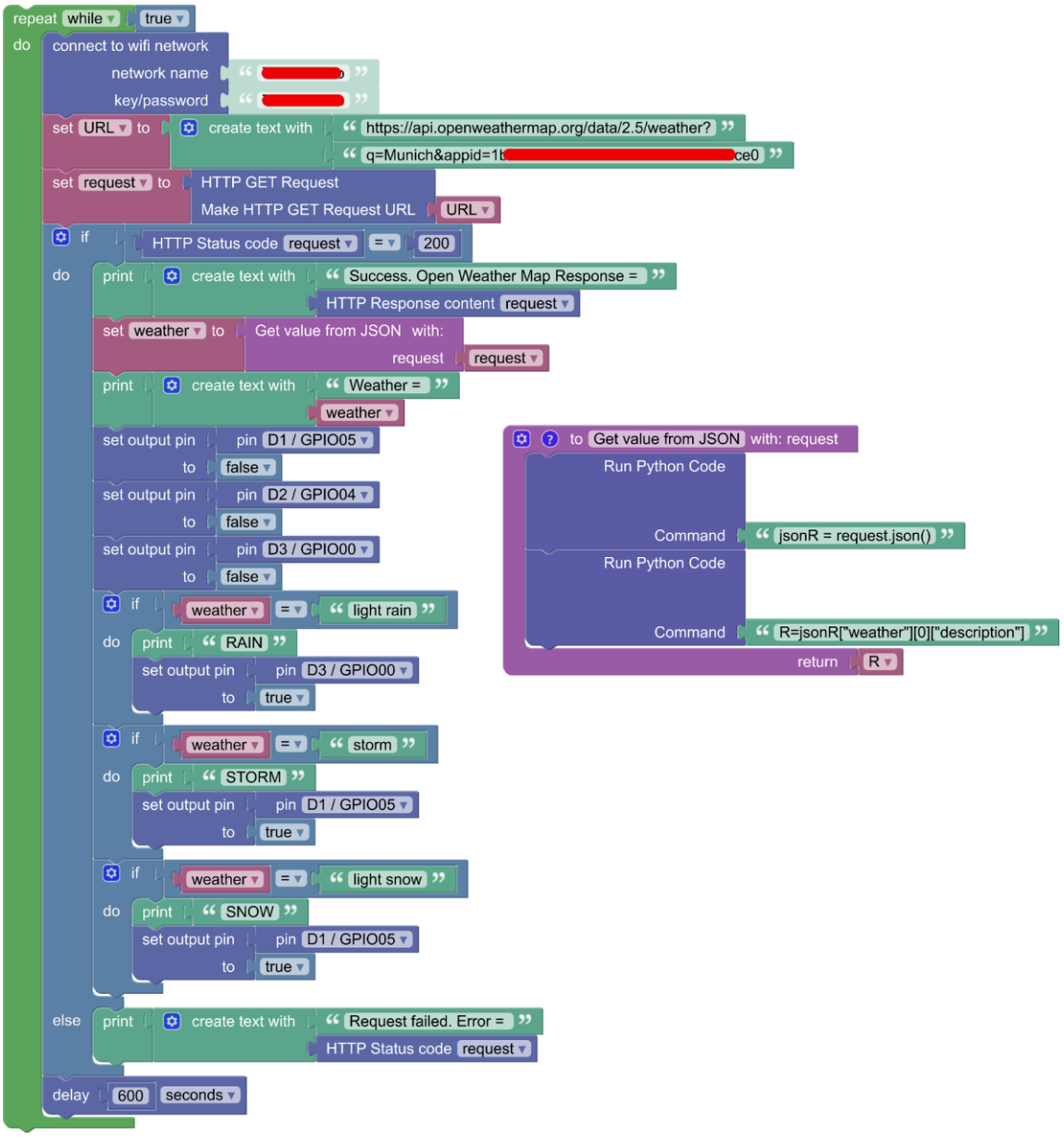
El siguiente programa muestra un ejemplo que utiliza la API de OpenWeatherMap para consultar las condiciones meteorológicas en Múnich. Tenga en cuenta que OpenWeatherMap devuelve un JSON (Notación de Objetos de JavaScript), que debe manejarse de manera diferente al XML. Además, hemos ocultado el valor de la clave de la API, así como el nombre y la clave de la red, utilizando una marca de bolígrafo rojo.

Este programa también nos ayuda a hablar de otra función de BIPES: ejecutar comandos de Python directamente dentro de los bloques.

Micropython tiene una función para manejar datos en formato JSON de manera automática. Sin embargo, no hemos preparado un bloque para eso en BIPES. Así que, con dos sencillas instrucciones en Python, podemos acceder fácilmente a cualquier información en la respuesta JSON:

variableJSON = request.json()
variableData = variableJSON[“item”][index][“subitem”].

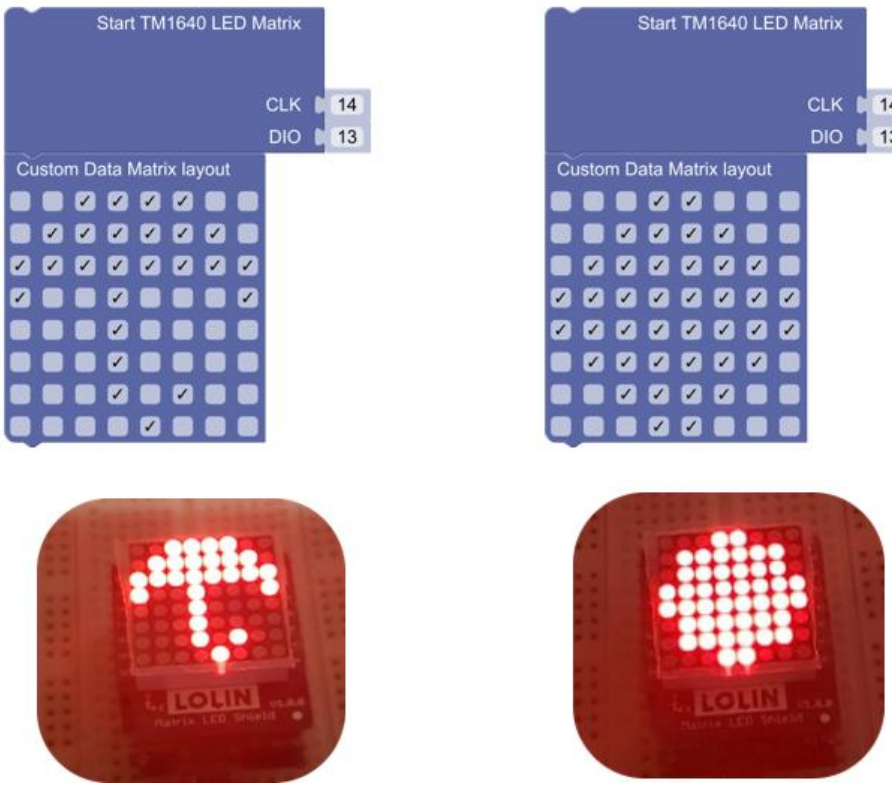
De esta manera, hemos creado una función Obtener valor de JSON (**Get value from JSON**) que recibe la solicitud como parámetro y, de forma interna, ejecuta estas dos sentencias de Python mencionadas anteriormente y devuelve el valor de interés al programa principal. ¡Está nevando en Múnich ahora!



Finalmente, la siguiente figura muestra el sistema funcionando en una placa Franzininho WiFi (https://franzininho.com.br/), una placa de hardware abierto desarrollada en Brasil y basada en el procesador ESP32S2. La placa cuenta con un LED RGB alimentado por un power bank USB, que muestra el clima actual en su LED RGB, junto a un tradicional “gallo meteorológico”.

Opcionalmente, incluso es posible incluir una matriz de LED y mostrar un icono de paraguas o de sol, dependiendo de las condiciones climáticas. En el ejemplo siguiente, se utilizó una matriz de LEDs con el controlador TM1640, y cada píxel de la matriz se definió en el bloque “Custom Data Matrix Layout” de BIPES.

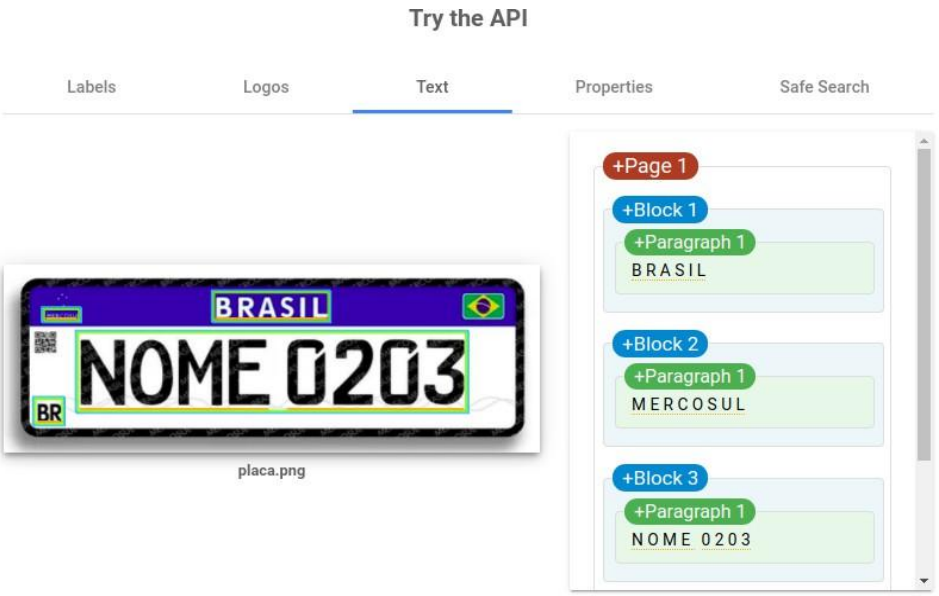
Estas matrices LED se pueden encontrar en formato shield, un accesorio que se conecta directamente a la placa ESP8266, evitando la necesidad de cables de conexión. En este caso, se utilizó la placa WeMos D1 mini con el shield TM1640 LED Matrix. La siguiente figura muestra los bloques utilizados para activar la matriz LED y su resultado.



Cliente HTTP: Otras posibilidades

Como se mencionó, existen miles de opciones de API para la integración mediante solicitudes HTTP. Por ejemplo, trabajamos con la API de condiciones meteorológicas, pero otro servicio útil incluye el envío de mensajes por Telegram a través del *endpoint* https://api.telegram.org/bot<token>/sendMessage.

Otra posibilidad aún más avanzada es agregar funcionalidades de inteligencia artificial y visión por computadora a un sistema embebido mediante el uso de servicios web de aprendizaje automático. Así, incluso en un sistema embebido con recursos computacionales limitados, es posible aprovechar toda la potencia de procesamiento de los servicios en la nube. Un ejemplo de esta posibilidad se encuentra en sistemas para analizar imágenes e identificar objetos, personas y caracteres. De esta forma, un circuito de bajo costo, como un módulo ESP32-CAM, puede capturar una foto y enviarla vía HTTP a un servicio en la nube. Google, Amazon e IBM cuentan con servicios avanzados de procesamiento de imágenes en la nube, capaces incluso de inferir si una persona está feliz o triste, entre otras características. La siguiente figura muestra un ejemplo de una placa de matrícula de automóvil, que se puede enviar, mediante una solicitud HTTP, a la *API Google Cloud Vision*. Como resultado, el servicio de Google devuelve una lista de los caracteres presentes en la placa. También se observa que el sistema detectó la palabra “MERCOSUR” presente en la placa, en su parte superior izquierda, en una dimensión pequeña. Puede encontrar más información sobre el uso de esta API en: https://cloud.google.com/vision/docs/drag-and-drop.



Tenga en cuenta que este es un ejemplo de reconocimiento de texto (también llamado OCR: reconocimiento óptico de caracteres), y el sistema podría incluir reconocimiento de objetos, emociones y otras características en la imagen. También puede buscar en Internet ejemplos y proyectos que utilicen el módulo ESP32-CAM con Google Cloud Vision. La comunidad de desarrolladores de sistemas embebidos ya ha implementado y puesto a disposición algunas opciones. Estas funciones en la nube también pueden ser útiles en aplicaciones de robótica, inspección óptica automatizada (AOI) y mecatrónica. Para obtener más detalles sobre el uso de la nube integrada con robots, una posibilidad de lectura es:

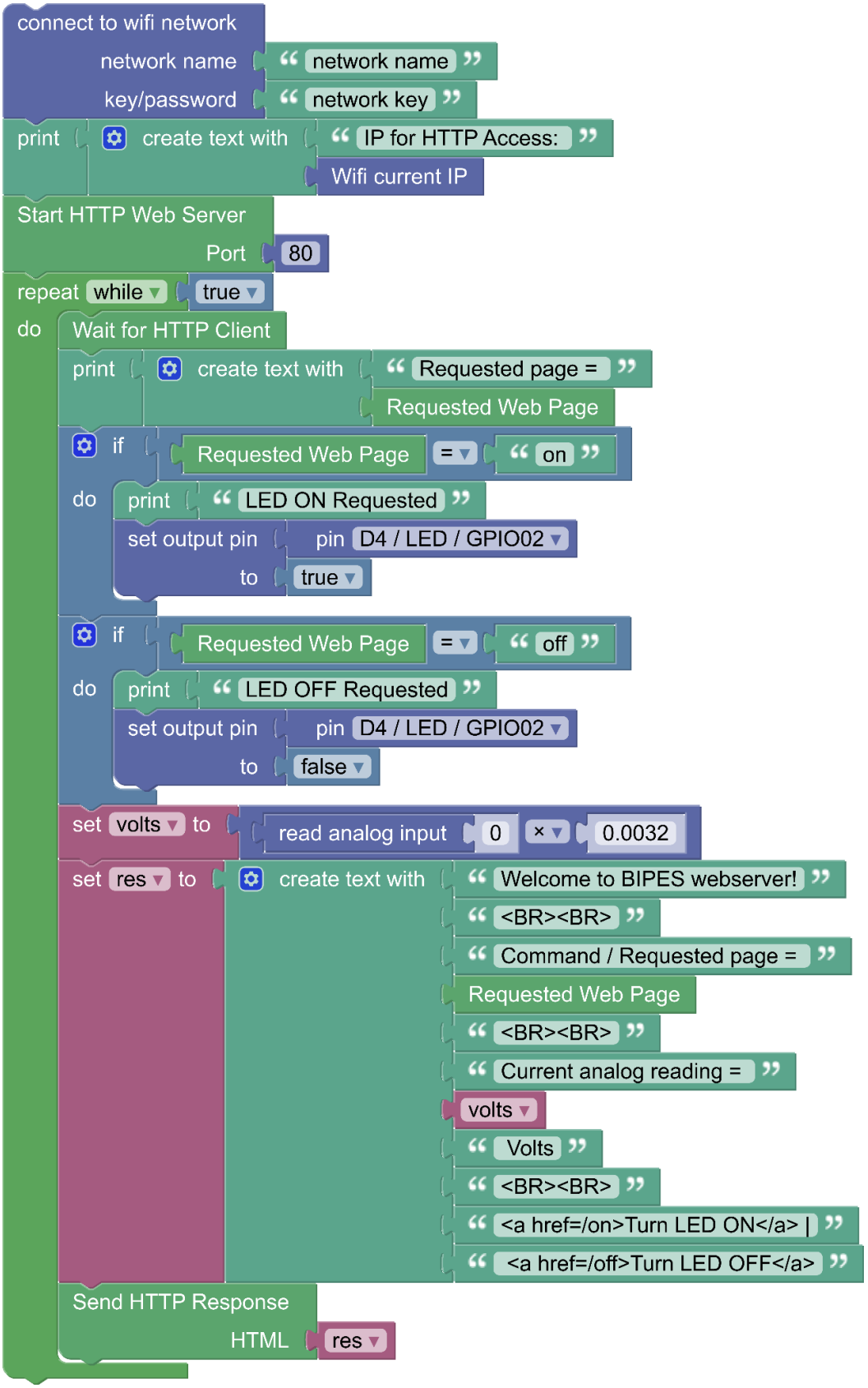
Aroca, RV, Péricles, A., de Oliveira, BS, Marcos, L., & Gonçalves, G. (2012, June). Towards smarter robots with smartphones. In the 5th workshop in applied robotics and automation, Robocontrol (pp. 1-6). sn.

Servidor HTTP

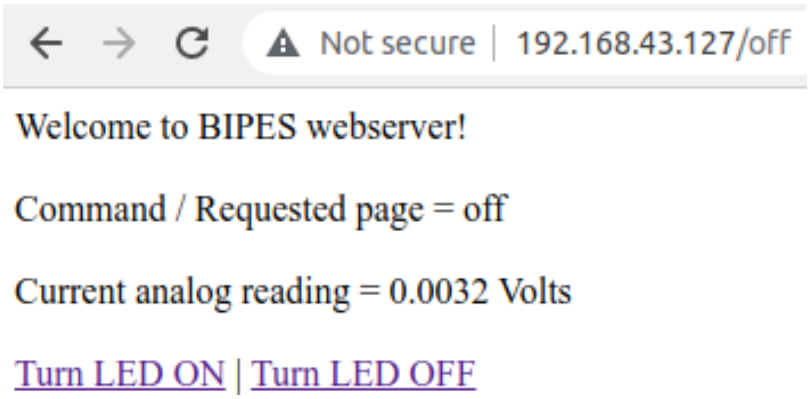
BIPES también permite que una placa ESP funcione como un servidor HTTP, lo que posibilita el acceso directo desde cualquier cliente HTTP, como un navegador web en una PC, un teléfono inteligente o incluso otra placa ESP32 o ESP8266 utilizando el bloque de solicitud *HTTP GET*.

La siguiente figura muestra la implementación de un servidor HTTP en BIPES. El servidor se inicia en el puerto 80 y espera continuamente solicitudes HTTP de los clientes. Al acceder mediante su dirección IP, el servidor muestra la lectura de la entrada analógica de la placa ESP8266 y ofrece dos enlaces a páginas web en este servidor: /on y /off. El acceso a estas páginas se verifica mediante los bloques "**IF Requested Web Page = on**" (u **off**) y genera una acción en el pin GPIO para encender o apagar el LED conectado al pin D4. Mediante códigos HTML, es posible insertar figuras, cajas de texto para solicitar datos del usuario, entre otras posibilidades para el desarrollo web / HTML. El bloque "Wifi IP Actual"

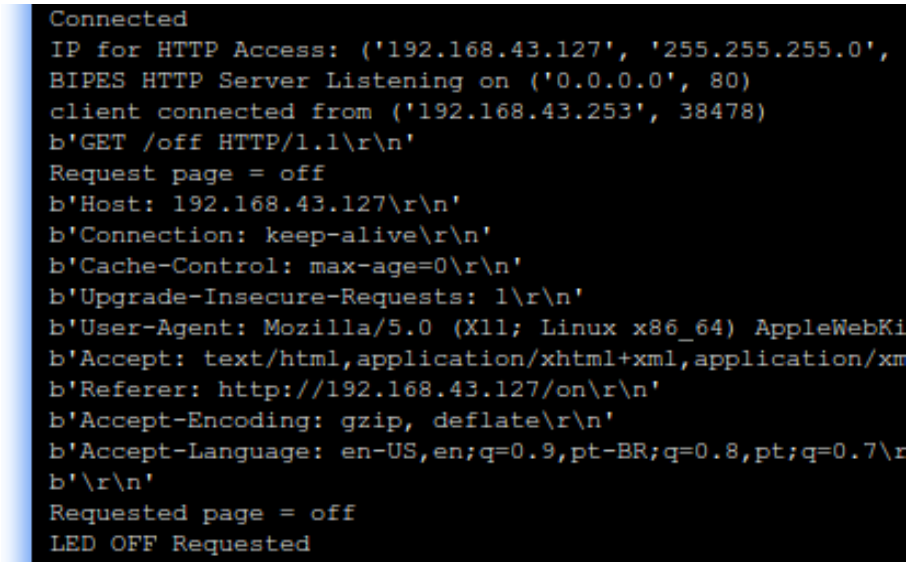
también se utilizó para verificar la dirección IP de tu tarjeta, facilitando la identificación de la IP para acceder a la tarjeta desde otros dispositivos en tu red.



El resultado se puede ver desde un navegador web, accediendo a la IP de la placa ESP:



Y, como se mencionó, puede ser supervisado por la Consola:



En la mayoría de los casos, este servidor web solo recibirá solicitudes HTTP de dispositivos conectados a la misma red WiFi en la que la placa ESP está ejecutando la aplicación del servidor HTTP.

Esta limitación depende de la arquitectura de red utilizada: en la mayoría de los casos, el punto de acceso inalámbrico o router obtiene una dirección de Internet (IP) única del proveedor de Internet, que es pública y accesible para recibir conexiones provenientes de Internet. Dicho router o punto de acceso distribuye direcciones IP internas/privadas que no son accesibles desde ningún otro dispositivo en Internet. Estas IP internas tienen la forma 192.168.x.y, 172.x.y.z o 10.x.y.z. La única IP pública se comparte con todos los dispositivos internos mediante la Traducción de Direcciones de Red (NAT). Así, como el módulo ESP tiene una IP privada, para que pueda ser accesible desde Internet, es necesario configurar una estrategia de redirección de puertos, la cual se presentará en la siguiente sección.

De todos modos, incluso con la IP con acceso solo en la red interna privada, todos los dispositivos en la misma red pueden acceder al servidor web de la placa ESP. Así, la IP de la placa podría configurarse, por ejemplo, en una aplicación hecha en MIT App Inventor, permitiendo integrar aplicaciones de teléfono inteligente con la placa ESP a través de solicitudes HTTP.

Para obtener más detalles sobre el proceso NAT y la asignación de direcciones IP, consulte un libro especializado en redes informáticas, como:

TANENBAUM, AS – Computer Networks – 4th Ed., Editor Campus (Elsevier)

Integración con Google Home o Amazon Alexa

El Internet de las Cosas ha experimentado un crecimiento significativo en la automatización del hogar, también conocida como domótica. En este contexto, dos dispositivos, que también son sistemas embebidos, ofrecen opciones de asistentes personales de alta calidad y flexibilidad: *Google Home* y *Amazon Alexa*. Entre las diversas funciones de estos dispositivos, la principal es la interacción con el usuario a través de comandos y respuestas de voz, donde el usuario puede realizar solicitudes de manera sencilla utilizando la voz. Por ejemplo, algunos de los comandos que se pueden utilizar son: “Agendar una cita”, “Encender la luz”, “Enviar un mensaje a Matheus”, “Recuérdame sacar mi ropa de la lavadora en 10 minutos” o “Haz una llamada telefónica a Lilian”.

La flexibilidad de estos asistentes personales incluye la posibilidad de integrarlos con nuevos servicios y aplicaciones, así como personalizar acciones. Utilizaremos el servicio IFTTT entre las diversas posibilidades de integración: ***If This Then That*** (https://ifttt.com/), gratuito hasta 5 acciones de automatización, denominadas Applets. Además, IFTTT ofrece cientos de opciones de eventos (***If This***) que pueden desencadenar cientos de otras opciones de acción (***Than That***). Además de contar con cientos de opciones, IFTTT también es compatible con Google Home y Amazon Alexa. Y, en particular, IFTTT dispone de la acción ***WebHook***, que le permite realizar una solicitud HTTP cuando se detecta algún evento: es decir, la solicitud HTTP se ejecuta en el contexto de la acción (Than That).

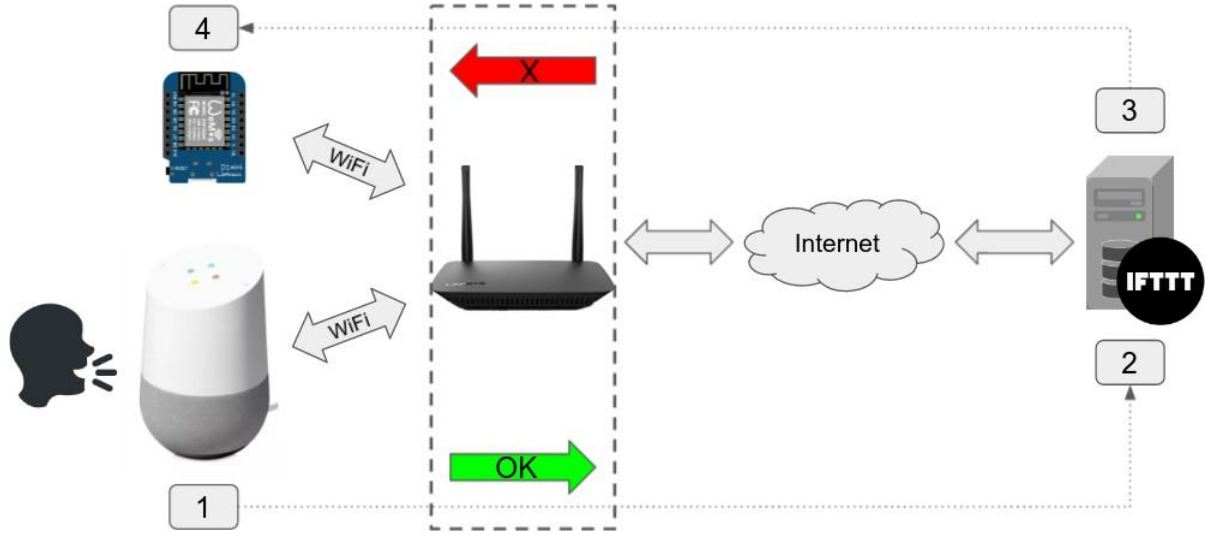
Así, es posible crear un Applet de IFTTT que espere una frase como “Enciende la luz”, y cuando se pronuncie esta frase, realice una solicitud HTTP al servidor web de BIPES integrado en la propia placa BIPES. Sin embargo, como ya se mencionó, el servidor HTTP integrado por BIPES en la placa ESP será inaccesible para el servidor del servicio IFTTT, por lo que será necesario realizar un ajuste técnico para que la solicitud llegue a la placa ESP. A continuación, se presentan dos opciones para que los datos de Google Home o Amazon Alexa, pasando por IFTTT, lleguen a la placa ESP que controla el dispositivo deseado.

Opción 1: Redirección de puertos

La siguiente figura ilustra la arquitectura de un sistema en el que un comando de voz realizado a través de Google Home resulta en encender o apagar un electrodoméstico mediante una placa ESP8266. En la figura, el usuario da el comando de voz que hemos

definido como “Enciende mi luz (*Shine my light*)”.

Así, Google Home recibe el comando de voz y envía (1) datos al servidor de IFTTT (2), un servicio en la nube, que procesa el comando en su Applet y realiza una solicitud HTTP (3) que es recibida por la placa ESP8266 (4). Ten en cuenta que el flujo de datos [(1) a (2)] tiene una flecha verde, donde, en la mayoría de los casos, no hay restricciones para enviar solicitudes desde una dirección IP privada interna a una dirección IP pública externa. Sin embargo, la solicitud del servidor de IFTTT a la placa ESP8266 [(3) a (4)] puede que no llegue a la ESP8266. Para que eso ocurra, es necesario configurar el reenvío de puertos en el router, punto de acceso o firewall (mencionamos varias ***o*** porque la arquitectura de cada red podría requerir configuraciones diferentes).

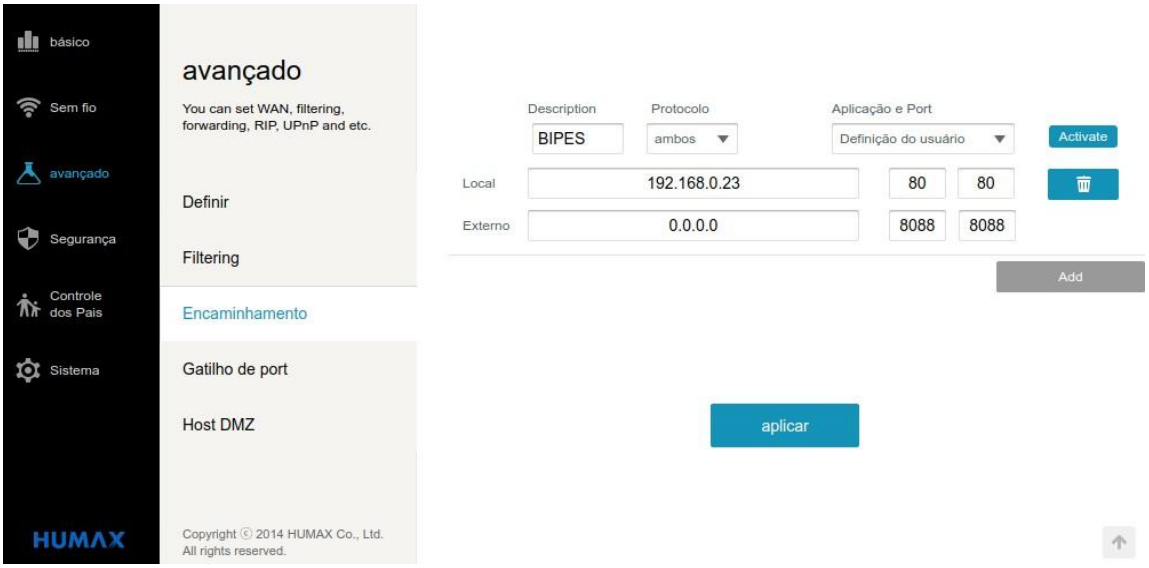


Para esta configuración, es necesario configurar un reenvío de puerto TCP desde la IP pública/externa de tu red hacia la IP interna de la placa ESP8266. Primero, consulta el manual de tu equipo de red y sus opciones. Como ejemplo, la siguiente figura muestra la configuración de reenvío de puertos en un módem/enrutador de cable integrado HUMAX.

Según la configuración que ya realizamos en la sección anterior, una placa ESP8266 recibió la IP privada 192.168.0.23 y se puede acceder a ella, desde la red interna, en la dirección http://192.168.0.23/. Ten en cuenta que el router HUMAX ha sido configurado en Opciones Avanzadas → Redirección, habilitando un reenvío de puertos desde la dirección externa 0.0.0.0 (cualquier dirección externa) hacia la dirección interna 192.168.0.23. Además, también se ha configurado para mapear el puerto externo 8088 al puerto interno 80 en la IP interna 192.168.0.23.

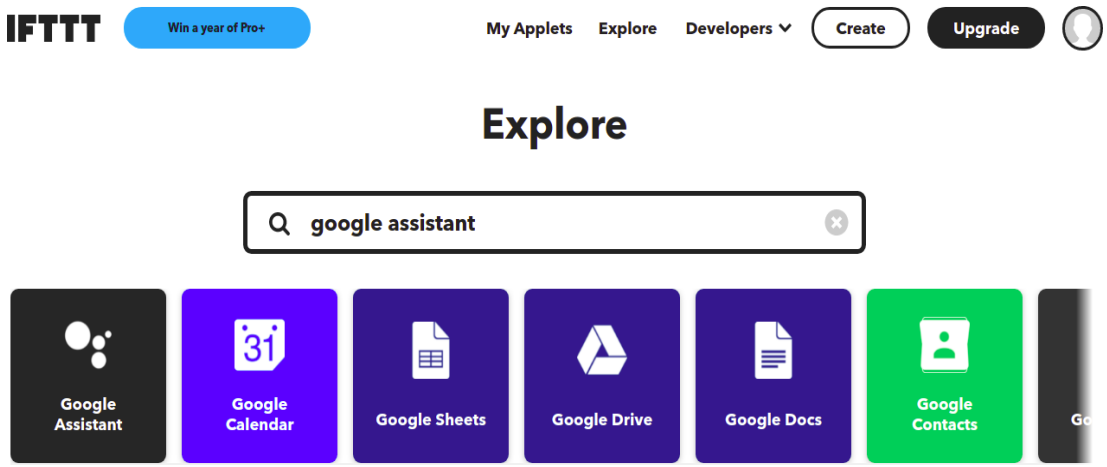
El siguiente paso es comprobar cuál es tu dirección IP pública. Una posibilidad es buscar en Google “*What is my IP address*”. El propio Google mostrará una dirección IP identificada como “Tu dirección IP pública”. Con el reenvío de puertos activado, puedes acceder a tu IP desde cualquier lugar de Internet, seguida del puerto 8088. Por ejemplo, si la IP es 187.66.80.187, utiliza un navegador web para ir a la dirección

http://187.66.80.187:8088/. A continuación, asegúrate de que el programa del servidor HTTP esté activo en la placa ESP8266. Si puedes acceder a la placa a través de esta dirección IP pública, IFTTT también podrá acceder a ella.

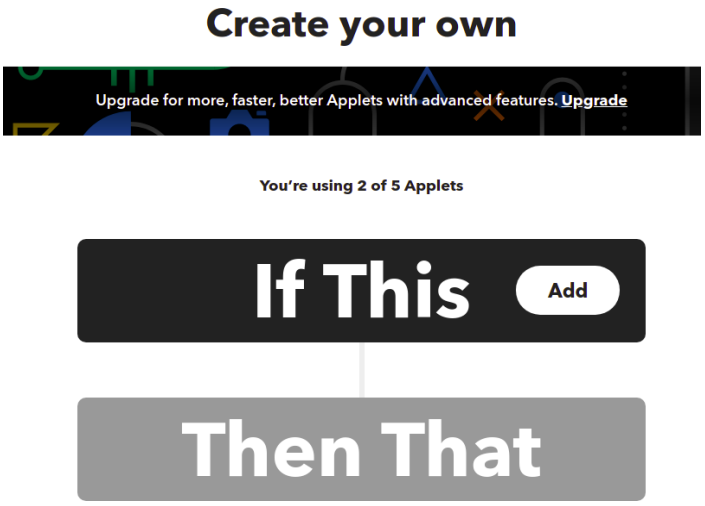


Antes de continuar, una pequeña nota: tu dirección IP pública puede cambiar dinámicamente después de algunas horas, días o semanas. Esta posibilidad de cambio puede ocurrir según tu proveedor de servicios de Internet (ISP). Por lo tanto, si necesitas usar el sistema con frecuencia, es interesante configurar un servicio de DNS dinámico (https://www.noip.com/pt-BR), que te proporcionará un nombre en Internet, como *mi_casa.noip.com*. Este nombre se actualizará y sincronizará continuamente con tu dirección IP pública, incluso si la IP cambia.

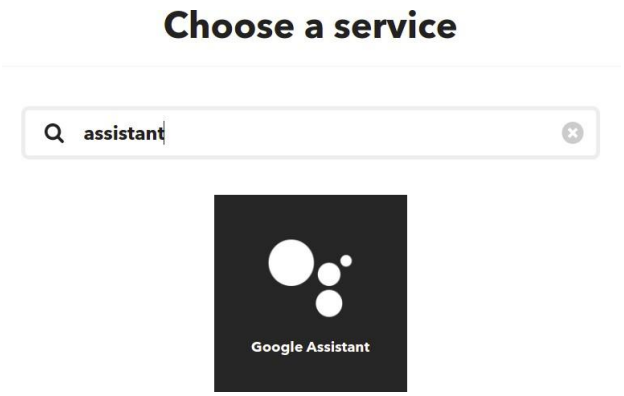
Ahora que sabemos la IP pública para realizar solicitudes HTTP, podemos configurar el Applet de IFTTT. Primero, ve a https://ifttt.com/ y crea una cuenta si no tienes una. Es importante usar el mismo correo electrónico/cuenta que ya utilizas con tu Google Home o Alexa. A continuación se muestra una configuración para Google Home, similar a la de Amazon Alexa. Después de crear la cuenta, haz clic en ***Explore*** y busca Google Assistant, como se muestra:



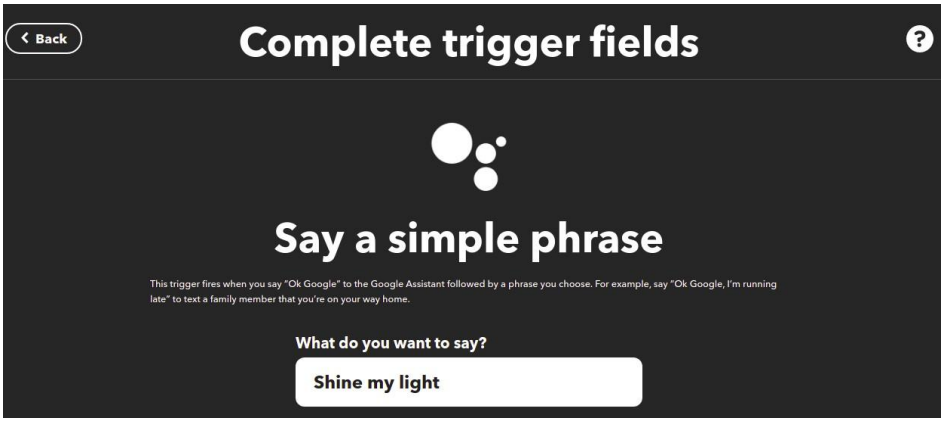
Sigue los siguientes pasos y confirma la asociación de IFTTT con Google Home. Este procedimiento solo necesita realizarse una vez. Luego, es posible crear un Applet de IFTTT. Para hacerlo, haz clic en *Create* en la pantalla principal de IFTTT:



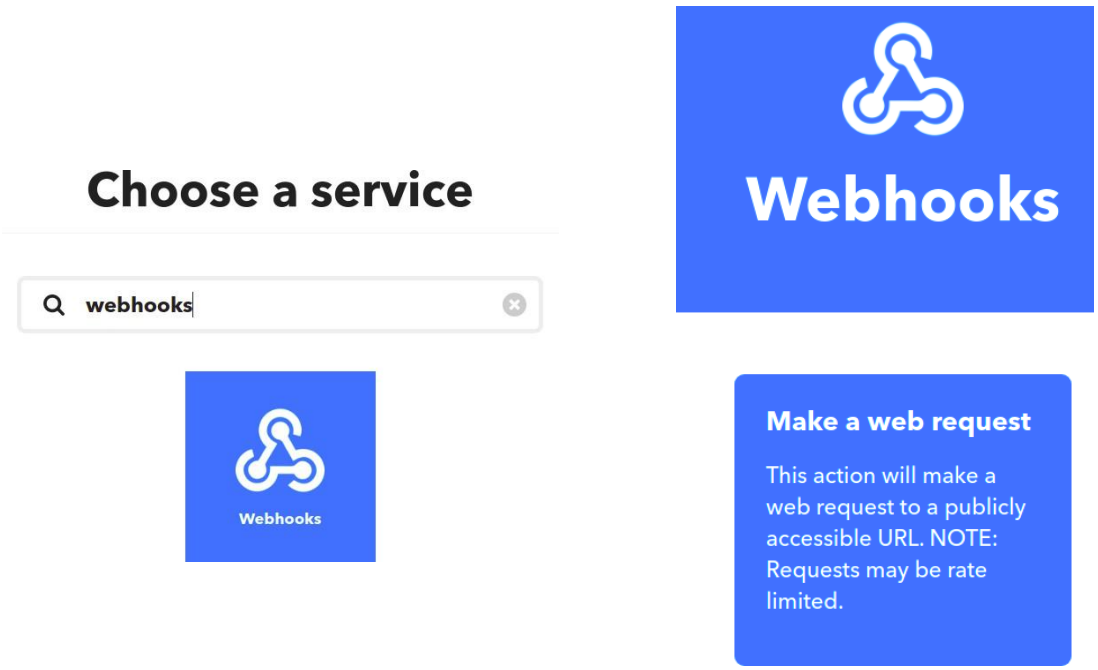
Haz clic en "If This" y elige Google Assistant:



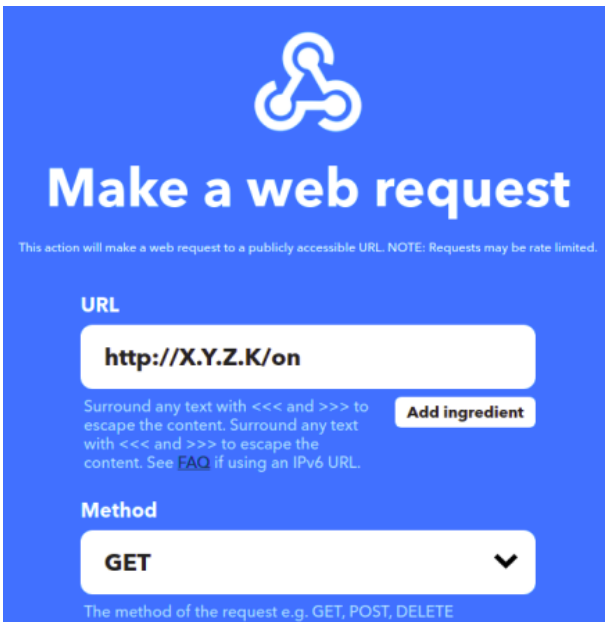
Elige la opción "*Say a simple phrase*" e ingresa una frase. En este ejemplo, usamos "*Shine my light*". También es posible especificar lo que el sistema dirá en respuesta a este comando mediante el campo "*What do you want the Assistant to say in response?*".



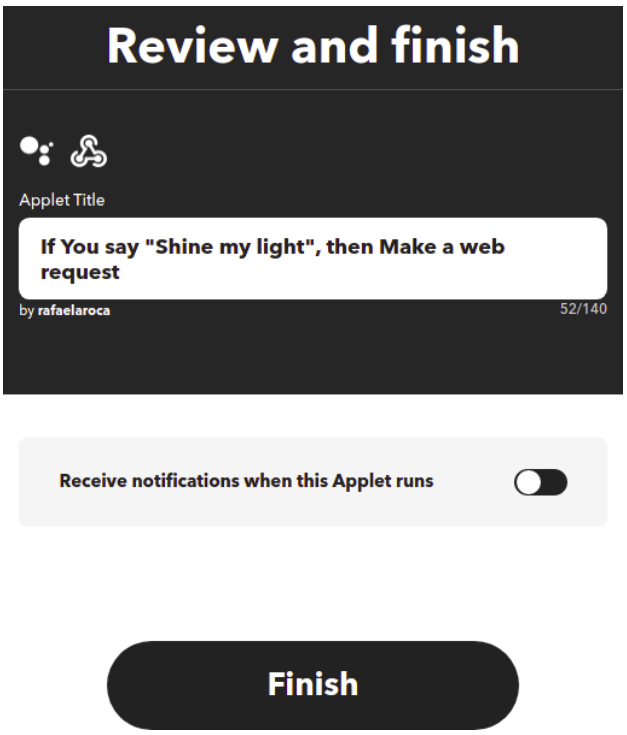
Haz clic en Crear desencadenador. A continuación, vamos a añadir la acción que se debe realizar. Haz clic en "*Then That*" y elige *Webhook*, luego selecciona la opción "*Make a web request*".



En "*Make a web request*", introduce la URL completa del servicio HTTP que llega al ESP8266. La URL debe incluir tu dirección IP pública junto con la ruta y el nombre del recurso; por ejemplo, http://187.66.80.187:8088/on o http://187.66.80.187:8088/off. La figura a continuación muestra la URL http://XYZK/on, de manera genérica. Ten en cuenta que en este campo se debe ingresar la dirección pública correcta. Además, selecciona el método GET para que se realice la solicitud HTTP.



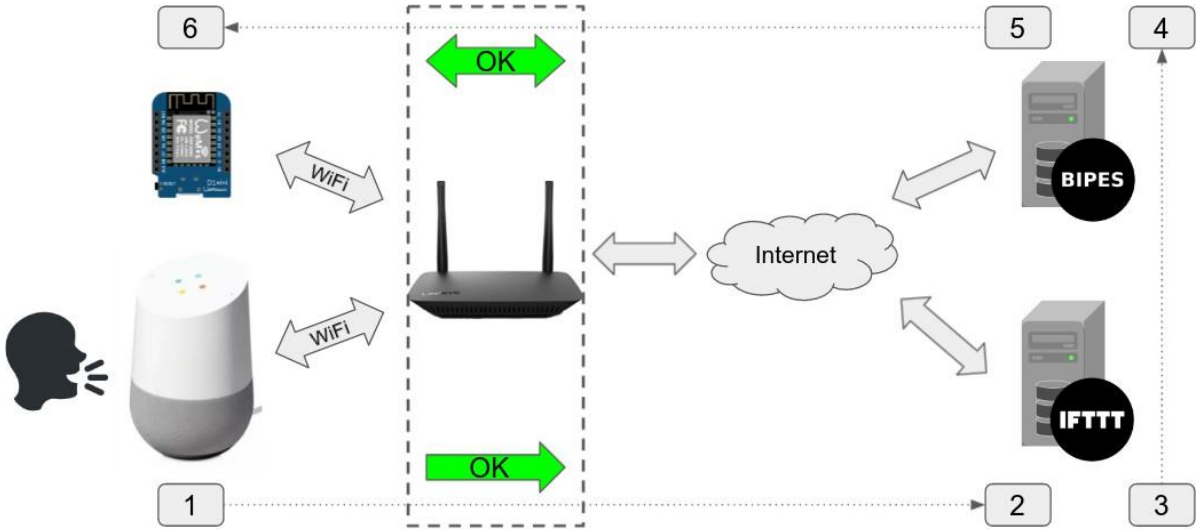
Finalmente, haz clic en “*Create action*” y luego en “*Continue*”, seguido de *Finish*. Por último, para probarlo, ve a Google Home y di: “*Ok Google, ¡Shine mi luz!*”.



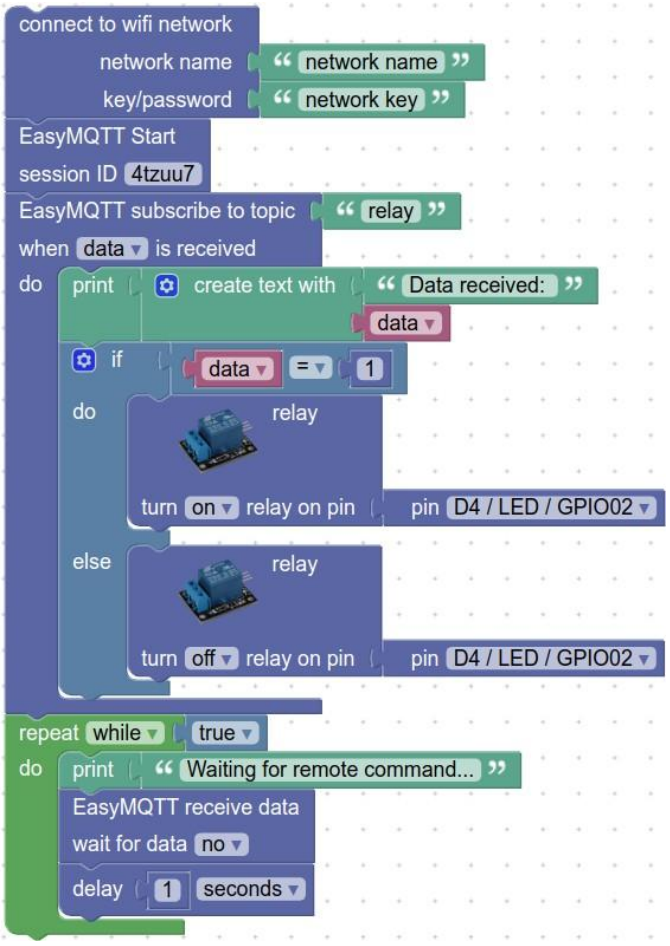
Opción 2: Integración con IFTTT a través de EasyMQTT

Puede ocurrir que la contraseña del router/punto de acceso o del firewall no esté disponible, y no sea posible configurar el reenvío de puertos. Algunos proveedores de servicios de Internet también pueden no proporcionar direcciones IP públicas a sus clientes o, por razones de seguridad, ofrecer conexiones a Internet con firewalls muy restrictivos, limitando la recepción de flujos de datos de Internet que no hayan sido iniciados desde su red. Para estos y otros casos, es posible integrar BIPES e IFTTT a través del servidor web http://bipes.net.br del propio proyecto BIPES.

La figura siguiente ilustra esta arquitectura del sistema: el usuario da un comando de voz “Shine my light” a Google Home (1), que realiza una solicitud al servidor de IFTTT (2). IFTTT ejecuta entonces un Applet (3) que efectúa una solicitud HTTP GET al servidor web del proyecto BIPES (http://bipes.net.br) (4), publicando un valor en un tópico de una sesión EasyMQTT de BIPES. El servidor BIPES, a su vez, actualiza esa sesión (5) de manera síncrona con la placa ESP8266, la cual debe estar ejecutando un programa suscrito a un tópico EasyMQTT, que luego recibe los datos y realiza una acción (6).



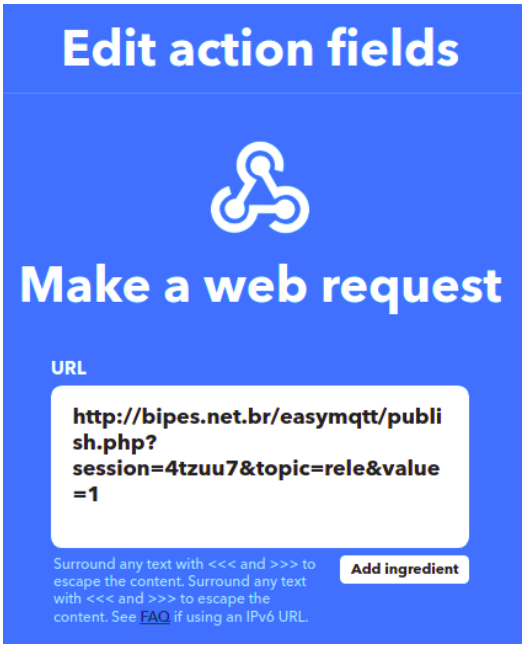
Tenga en cuenta que esta vez no hay un servidor HTTP en la placa ESP8266, sino únicamente un programa suscrito a un tópico de una sesión EasyMQTT. Este programa ya se ha ilustrado en la sección "***Control de Dispositivos a través de Internet***". Sin embargo, presentamos el mismo programa aquí nuevamente para facilitar su comprensión. Este programa se suscribe al tópico del relé de la sesión *EasyMQTT 4tzuu7* y, ante cualquier cambio en este tópico, realiza la acción correspondiente (bloque "***suscribe to the topic***").



Recuerda que ya hemos utilizado una URL del servidor de proyectos BIPES para cambiar una sesión de EasyMQTT y controlar un dispositivo desde una aplicación para smartphones creada con MIT App Inventor. La URL utilizada para la integración con una aplicación de smartphone fue:

http://bipes.net.br/easymqtt/publish.php?session=4tzuu7&topic=relay&value=1

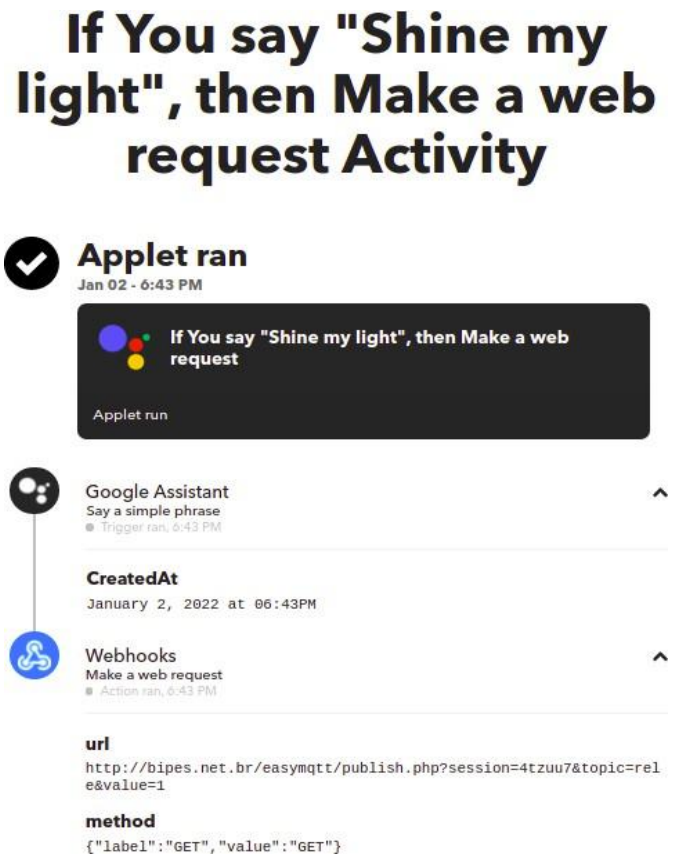
Ahora, esta misma URL se puede utilizar para la integración con IFTTT. El Applet de IFTTT es idéntico al anterior, solo cambia la URL que activa IFTTT cuando ocurre el evento:



Guarda los cambios y pruébalos pidiéndole a Google Home: “Shine my light”. En este caso, no se necesita configuración de reenvío de puertos/paquetes, como en la opción 1. Por lo tanto, el sistema funciona independientemente de la configuración y propiedades de la red, pero depende del servidor BIPES.

Nota: Recuerda que BIPES es un proyecto abierto y gratuito, y la sesión utilizada como ejemplo aquí (4tzuu7) puede ser utilizada por otros usuarios/lectores, lo que podría causar efectos inesperados (cambios en el estado de un tópico). Por lo tanto, define una sesión única para tu proyecto. Puedes verificar si una sesión de EasyMQTT está vacía/disponible mediante la URL: https://bipes.net.br/beta2/easymqtt/getsession.php?session=12, donde 12 es la sesión deseada a verificar.

Finalmente, puede ser útil saber que IFTTT ofrece la opción “***View Activity***”, que le permite comprobar los registros (logs) de recepción de eventos y los resultados. En particular, en el caso de *Webhooks*, es posible verificar si el comando fue enviado al servidor y el resultado de la solicitud HTTP. La figura a continuación muestra un ejemplo de una solicitud realizada al servidor EasyMQTT del proyecto BIPES.



Actividad: Cliente - Servidor: control remoto a través de WiFi

Utiliza un par de placas ESP8266, programando una para que funcione como punto de acceso inalámbrico y servidor HTTP. Luego, configura la otra como estación WiFi y cliente, de modo que esta segunda placa, el cliente, envíe una solicitud HTTP cada vez que se active un pin digital (por un sensor, un botón u otros dispositivos). Cuando la placa configurada como servidor reciba la solicitud HTTP, debe cambiar un pin de salida para encender o apagar cualquier dispositivo. De este modo, una placa ESP8266 actuará como un control remoto inalámbrico vía WiFi de un dispositivo conectado a la placa ESP8266 que funciona como “servidor”.

*** * * * ¡Felicidades!**

¡Has desarrollado aplicaciones completas de Internet de las Cosas (IoT) con nube, sensores, control remoto y paneles para visualización y control!

*** * * ***

Actividades extra

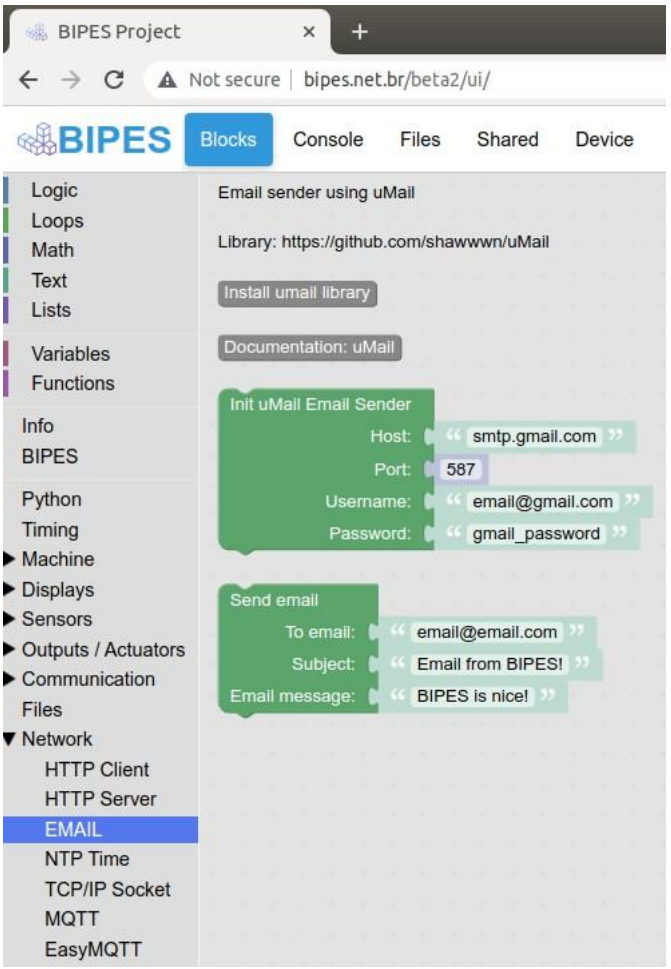
Enviando un correo electrónico

Actividad:
Diseña un programa que envíe un correo electrónico a ti cuando un pin de entrada digital (GPIO) en el ESP8266 detecte un cambio en tu señal de entrada (por ejemplo, de 0 a 1). Este pin puede estar conectado a un sensor de presencia o a un sensor de puerta abierta (interruptor de láminas). Al detectar el cambio, el programa debe enviar el correo electrónico.

Tips:

1. Para esta actividad, la placa ESP8266 debe estar conectada a Internet;
2. BIPES tiene un sistema de librerías modular. Por ejemplo, necesitas instalar la librería uMail para enviar correos electrónicos. La instalación se puede realizar rápidamente accediendo a Network -> EMAIL y luego haciendo clic en "*Install mail library*".

a.



3. Después de instalar la biblioteca, puedes utilizar sus bloques de lanzamiento y los bloques para enviar correos electrónicos:

a. 

GMAIL

BIPES utiliza la biblioteca umail de MicroPython, que le permite enviar correos electrónicos usando prácticamente cualquier proveedor de correo. Sin embargo, cada proveedor puede requerir configuraciones específicas.

Gmail, por ejemplo, requiere que crees una contraseña de aplicación para que el ESP8266 pueda enviar correos electrónicos directamente. Por lo tanto, la cuenta de GMAIL debe tener la autenticación en dos pasos habilitada y la opción de “Contraseña de aplicación” activada.

El siguiente enlace contiene una explicación adicional:https://myaccount.google.com/apppasswords

En este enlace, selecciona:

App -> Email,
Device -> Other

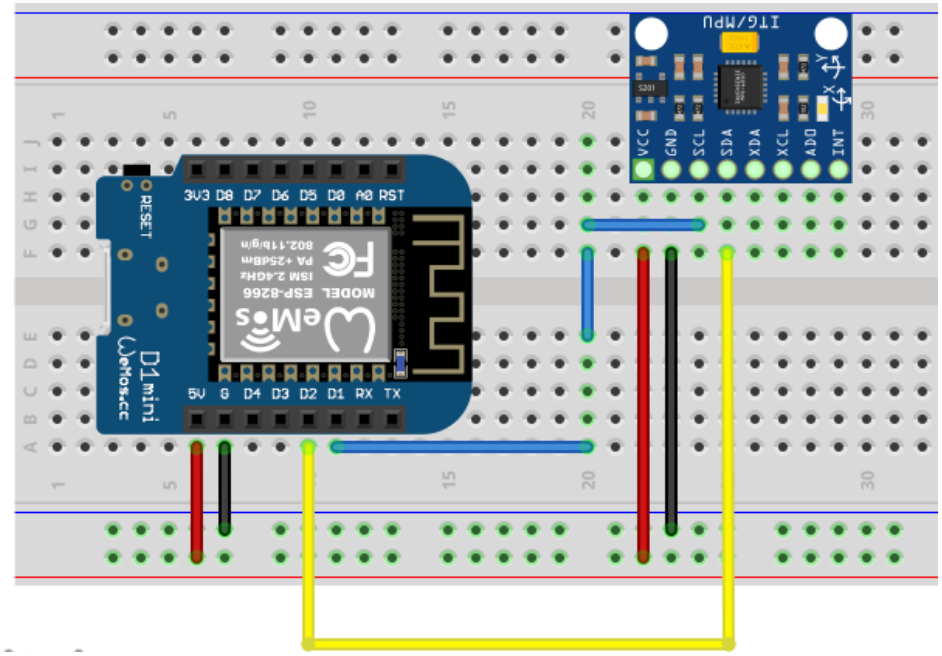
Gmail generará una contraseña dedicada para el ESP8266. Copia esta contraseña y pégala en el bloque Init uMail.

Unidad de Medición Inercial - MPU6050

Si tienes una unidad de medición inercial MPU6050, puedes utilizarla según los diagramas a continuación y usar el bloque IMU / MPU6050:

La siguiente tabla muestra las conexiones:

MPU6050	ESP8266 pin
GND	GND
VCC	5V
SCL	5 (D1)
SDA	4 (D2)



Actividad:
Crea un panel de IoT que muestre la variación de los ángulos de orientación de la placa en gráficos en tiempo real.

Desafío:
Implementa un programa para calcular los ángulos (en este caso, utilizando únicamente acelerómetros) de inclinación en los tres ejes a partir de las mediciones de aceleración gravitacional y prepara una flecha en el panel de control IoT que indique la inclinación de la placa.

Motor servo RC (para aeromodelismo)

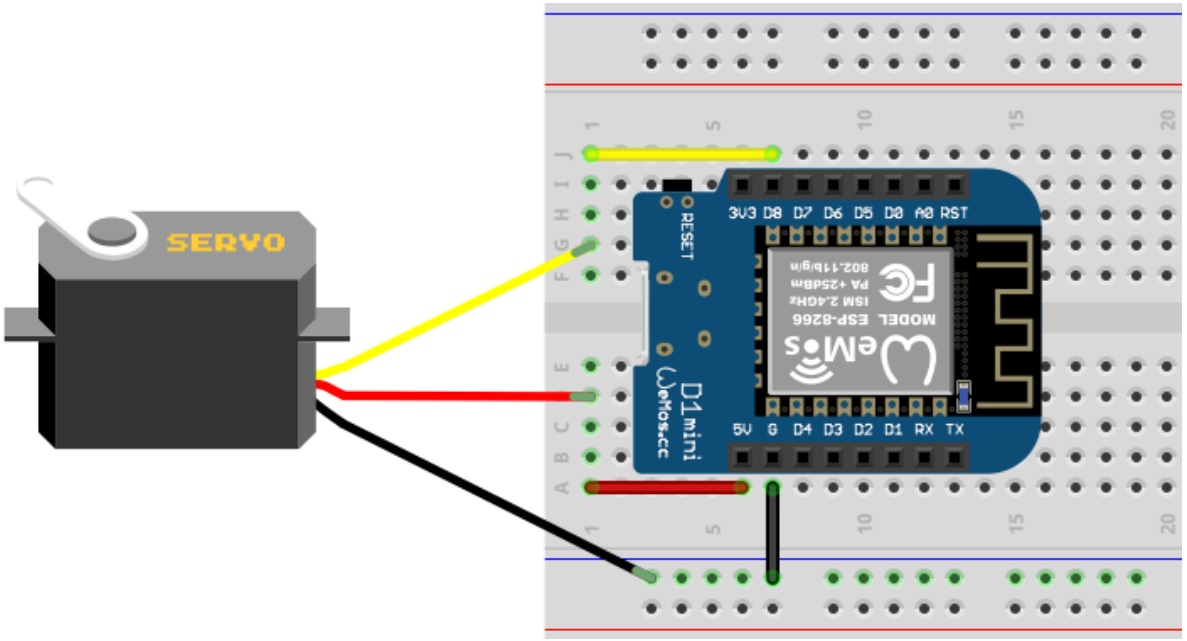
Actividad:

Desarrolle un programa con un panel de control IoT que contenga un control deslizante. Cuando el usuario cambie el valor del control deslizante, el servo debe moverse al ángulo establecido en el control deslizante.

Tip: utiliza el bloque RC Servo Motor en la caja de herramientas a la izquierda, en Salidas y Actuadores.

La siguiente tabla muestra las conexiones:

Servo	ESP8266 Pin
GND (marrón)	GND
VCC (rojo)	5V
S / Señal (naranja)	15 (D8)



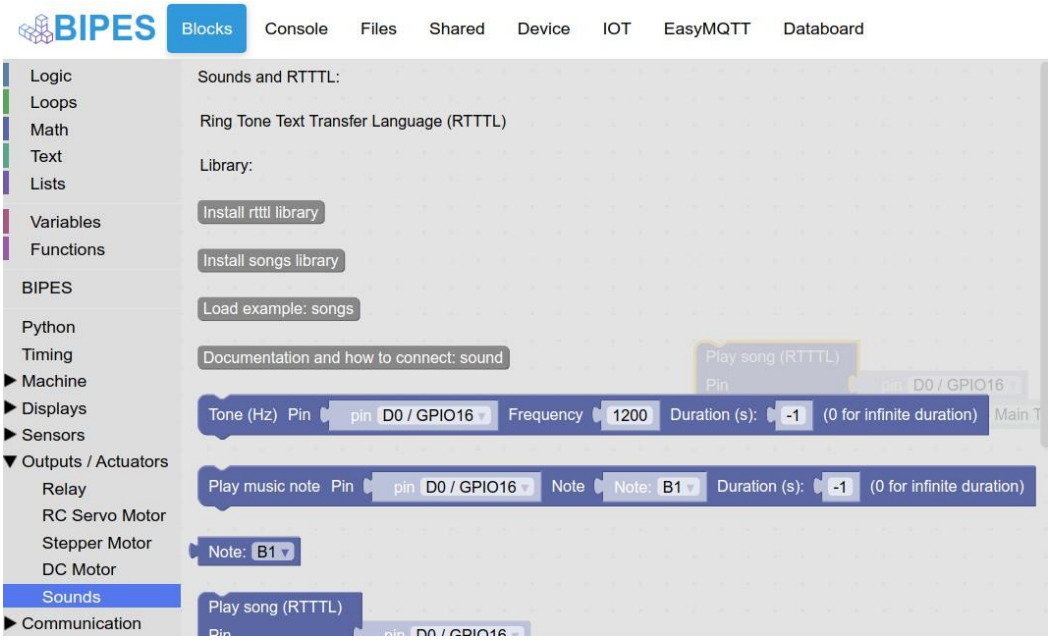
Desafío:

Implemente un sistema tipo “gimbal” que ajuste la posición del servo según el ángulo medido por la unidad inercial MPU6050 (en este caso, utilizando únicamente acelerómetros).

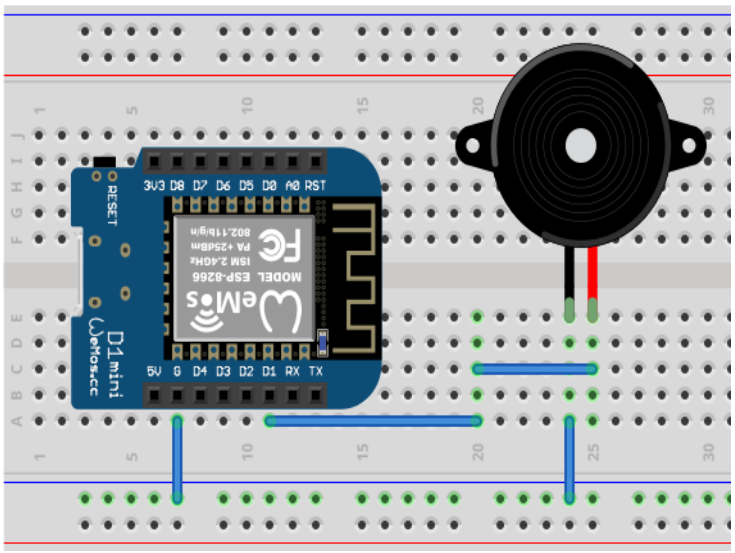
Música

BIPES también ofrece bloques para generar tonos de sonido y reproducir canciones completas. Consulta los bloques y opciones en la barra lateral de herramientas de *outputs and actuators >> sounds*. En cuanto al hardware para reproducir una canción, solo necesitas un zumbador.

Para producir sonidos, BIPES utiliza una biblioteca externa. Ve a la pestaña lateral *Outputs and Actuators >> Sounds*, haz clic en *Install rtttl library* y luego en *Install songs library*. Se descargarán e instalarán dos archivos en la memoria de las placas para la reproducción de audio. La placa debe estar conectada a Internet para ejecutar estos comandos.

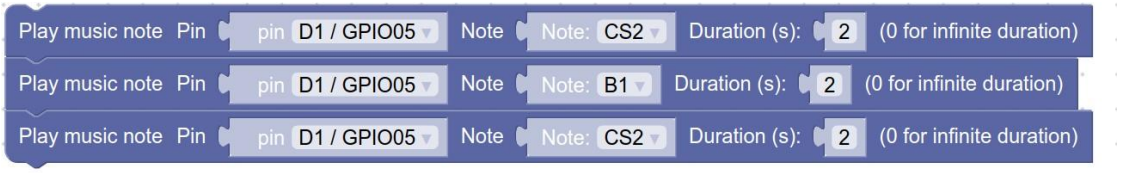


Después de eso, puedes ensamblar un circuito con la placa ESP8266 y un zumbador, como se sugiere a continuación:



¡Luego podrás reproducir canciones! Ten en cuenta que hay una biblioteca con varias canciones ya preparadas, que puedes usar directamente, o incluso incluir/crear tus propias canciones editando el archivo songs.py.

También es posible reproducir una secuencia de notas musicales tocando tu música:



Tenemos un video en YouTube con ejemplos de algunas canciones reproducidas por una tarjeta ESP32: https://www.youtube.com/watch?v=a7U-sz70Wrg

BIPES con otras plataformas

BIPES está disponible para varias otras plataformas y ofrece muchas otras posibilidades, incluyendo bloques de control PID en lazo cerrado, RFID, visión por computadora con OpenCV, entre otras. ¡Así que explora, disfruta, comparte y contribuye al proyecto!

Python con Arduino - Snek

BIPES también funciona con Arduino, gracias a Snek Lang **(https://sneklang.org/)** de Keith Packard. Consulta más en**:** https://bipes.net.br/snek-web-uploader/.

Otros

BIPES es compatible con múltiples plataformas de microcontroladores y SoCs (Sistemas en un Chip), como micro:bit, STM32 y otros. BIPES también puede utilizarse con placas Raspberry Pi Pico con MicroPython o Raspberry Pi (y similares) con Linux. Además, BIPES fue el primer proyecto en permitir la programación basada en bloques para la Raspberry Pi Pico. Consulta el sitio web del proyecto BIPES para obtener más detalles y conocer las plataformas de hardware compatibles: https://bipes.net.br/.

Obtener ayuda y ayudar

El proyecto BIPES cuenta con una comunidad de soporte para usuarios en GitHub. Si tienes alguna pregunta, crítica o sugerencia, visita nuestra comunidad y siéntete libre de preguntar, ayudar y contribuir al proyecto a través del enlace:

https://github.com/BIPES/BIPES/discussions/

También puedes contribuir a **BIPES** de varias maneras. Obtén más información sobre el proyecto en https://github.com/BIPES and https://bipes.net.br/.

Comentarios finales

Los sistemas embebidos y las aplicaciones de Internet de las Cosas (IoT) pueden ser útiles en muchas ocasiones. BIPES tiene como objetivo facilitar la creación rápida de prototipos de aplicaciones embebidas e IoT, y contribuir a un acceso más accesible e intuitivo a la programación de estos sistemas para usuarios con menos experiencia. Además, se espera que BIPES también incremente la productividad de los usuarios experimentados.

Los proyectos de Ciencia, Tecnología, Ingeniería, Artes y Matemáticas (STEM o STEAM) pueden beneficiarse de BIPES. Por ejemplo, puede contribuir a proyectos de robótica educativa y proyectos maker, ofreciendo programación intuitiva, recopilación de datos, almacenamiento de datos, una plataforma de visualización y eliminando la necesidad de instalar software o realizar configuraciones específicas, lo que facilita su uso en diferentes tipos de computadoras y dispositivos. También ofrece una forma diferente de aprender programación mediante código, ya que los programas basados en bloques se convierten automáticamente a Python. Además, el código Python puede editarse, modificarse y probarse rápidamente.

En el entorno académico, se puede utilizar para automatizar experimentos de laboratorio, recopilar datos o facilitar prácticas de laboratorio que requieran algún tipo de monitoreo o automatización.

Este libro también se basó en el uso de dispositivos de bajo costo, como los módulos ESP8266 y ESP32, ya ampliamente utilizados en aplicaciones de automatización del hogar, comerciales, educativas, académicas y, en algunos casos, en monitoreo de máquinas en industrias, en el campo y otros entornos.

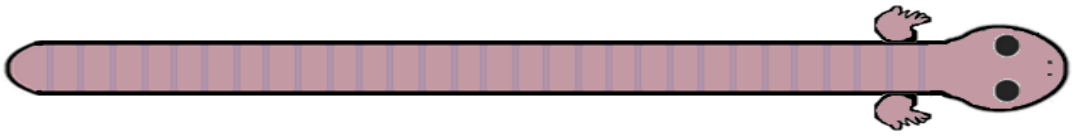
Otro pilar del proyecto BIPES es MicroPython o su variante, CircuitPython. Aunque es una versión de Python optimizada para microcontroladores, existen varios casos de éxito en aplicaciones industriales que utilizan MicroPython e incluso proyectos espaciales basados en MicroPython. La Agencia Espacial Europea (ESA), por ejemplo, tiene un proyecto para usar MicroPython en aplicaciones espaciales4. También existen varios kits educativos y de robótica que utilizan MicroPython. Por ejemplo, kits de satélites educativos basados en procesadores ESP32, como los satélites educativos de PION, MySatKit e IdeaSpace.

Esperamos que este texto haya sido útil y ¡agradecemos su atención!

[4] https://essr.esa.int/project/micropython-for-leon-pre-qualified-version

Agradecimientos

El proyecto BIPES integra varias herramientas de software de código abierto: Google Blockly, MicroPython, freeboard, CodeMirror, bibliotecas de MicroPython, esp-web-tools, entre otras. El equipo de BIPES ofrece un gran agradecimiento a todos los desarrolladores de estas herramientas libres. Agradecemos a los colegas del equipo del proyecto BIPES, incluyendo a Gustavo Tamanaka, por el logotipo del proyecto, y a Caio Augusto Silva por la implementación del módulo EasyMQTT. Estamos agradecidos con el proyecto Fritzing, que hizo posibles los dibujos y diagramas en este texto. También agradecemos el apoyo de la Universidad Federal de São Carlos a través de su Pro-rectoría de Extensión. Agradecemos igualmente al Consejo Nacional de Desarrollo Científico y Tecnológico del Ministerio de Ciencia, Tecnología e Innovación (CNPq/MCTI) por apoyar el proyecto BIPES (beca CNPq 306315/2020-3). Adicionalmente, los autores agradecen el apoyo brindado por el Ministerio de Ciencia, Tecnología e Innovación de Colombia (MINCIENCIAS) y por el Fondo Francisco José de Caldas a través de la asignación de recursos en el marco de la Convocatoria 934 de 2023, contrato 332-2023, proyecto 99209.



Plataforma integrada basada en bloques para sistemas embebidos

http://bipes.net.br

Con el apoyo de:

CNPq FUNDAÇÃO VINCULADA AO MCTI

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES

PATRIA AMADA BRASIL GOVERNO FEDERAL

Una introducción al Internet de las Cosas y los Sistemas Embebidos utilizando programación basada en bloques con BIPES y ESP8266 / ESP32.

Esta traducción fue realizada con fines académicos y educativos. El texto original es de libre acceso en el sitio oficial del proyecto BIPES (http://bipes.net.br).

Diciembre/2021 – Primera Edición:
Rafael Vidal Aroca
Wesley Flavio Gueta
Jorge André Gastmaier Marques
Tatiana de Figueiredo Pereira Alves Taveira Pazelli

Traducción al español:
Henry B. Guerrero y Danna Celena Montenegro Orjuela
Primera edición en español – 2025

ISBN: 978-65-01-79539-3

9 786501 795393