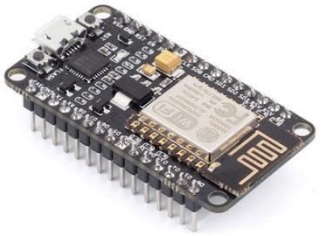
# Módulo 4: Plataformas MQTT y Adafruit.io





## ÍNDICE

4 Plataformas MQTT **2**

4.1 Introducción. 2

4.2 El protocolo MQTT 6

4.3 Plataformas MQTT 9

4.4 MQTT de Adafruit 10

4.5 Ejemplo 1: Control de un LED 14

PASO 1: Monitorizar el led 16

Paso2: Controlando el LED desde la placa y Adafruit IO. 21

4.6 Ejemplo 2: Temperatura y humedad 22

4.7 Bibliografía 28

# 4 Plataformas MQTT

## 4.1 Introducción.

El primer dato conocido sobre una "cosa", que no fuera un ordenador de propósito general, conectado a Internet, ARPANET se llamaba entonces, fue una máquina expendedora de bebidas gaseosas de la marca Coca-Cola. Se conectó a la red local en 1970 del campus de la Escuela de Ciencias de la Computación de la Universidad Carnegie Mellon y en 1982 a la red global. Querían comprobar las existencias de la máquina antes de recorrer medio campus universitario en balde si estaba vacía, teniendo en cuenta que se cargaba en horario irregular por estudiantes voluntarios de posgrado. Se trataba de una máquina con 6 columnas de carga a las que instalaron micropulsadores para detectar si había botellas y los conectaron a un servidor. El mensaje que ofrecía el programa guardaba relación con el orden de los botones selectores de cada columna dispuestos como en la tabla siguiente:

| | | |
|---|---|---|
| columna 1 | columna 2 | columna 3 |
| columna 4 | columna 5 | columna 6 |

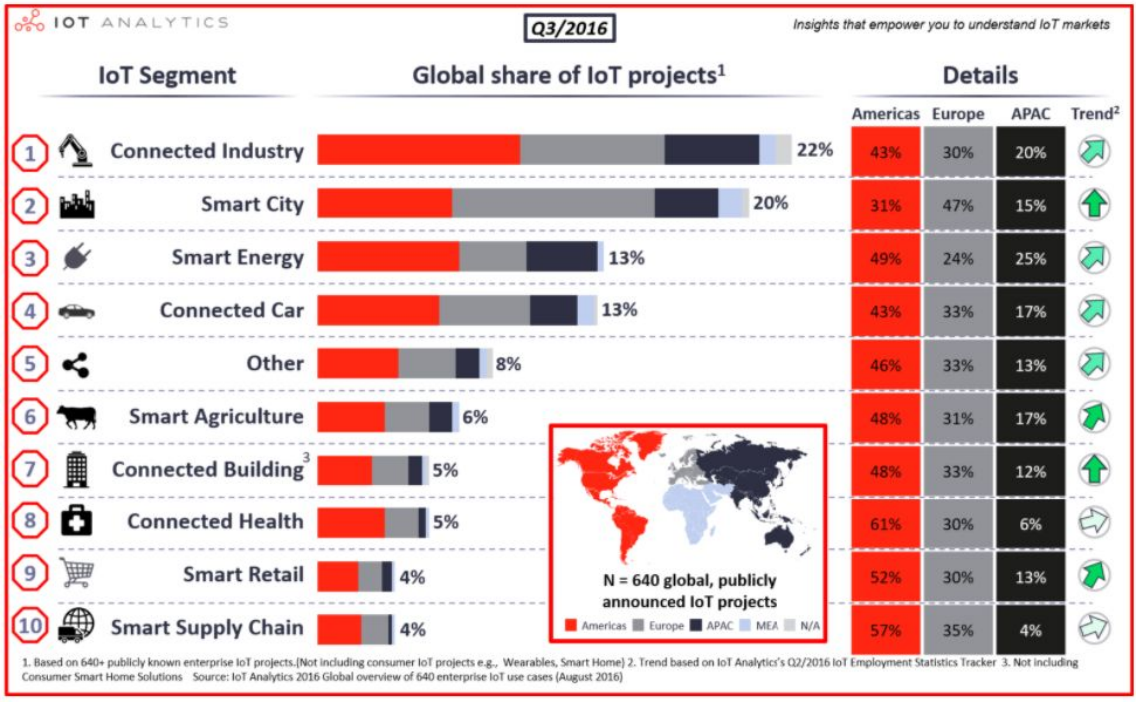
de manera que en cada "columna X" podía indicar EMPTY, COLD o el tiempo desde la recarga de forma que tenían actualizada la carga de la máquina. Si el mensaje era, por ejemplo:

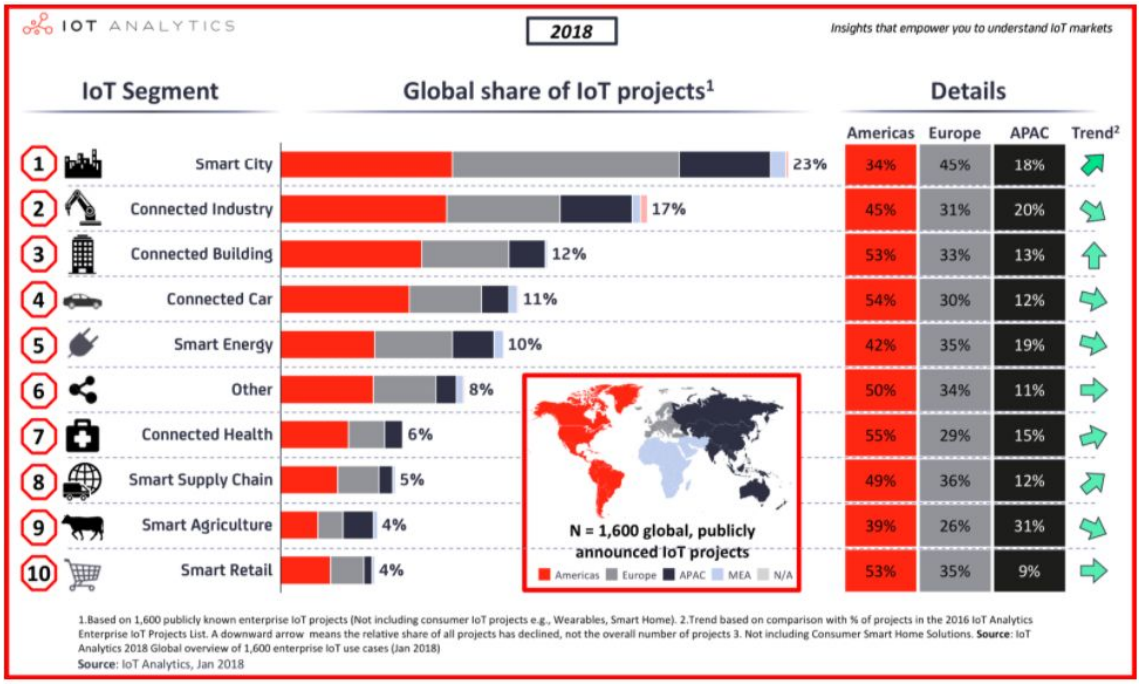
| | | |
|---|---|---|
| EMPTY | COLD | 1h5m |
| 1h4m | EMPTY | COLD |

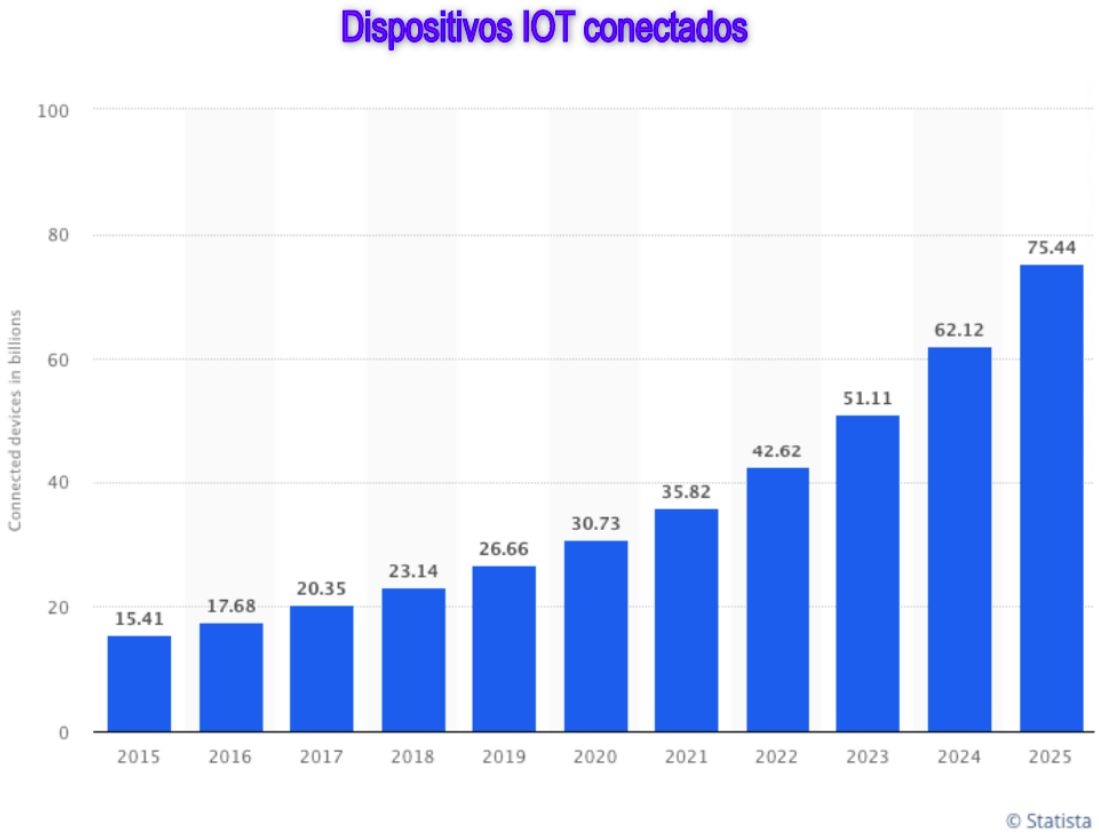
Indicaba que no quedaban botellas en las columnas 1 y 5, que para sacar una botella fría había que pulsar el segundo botón o el último y que en la tercera y cuarta columna habían cargado botellas hacía poco más de una hora por lo que no estarían frías del todo. Este acto de "vandalismo estudiantil", se hizo eco entre departamentos de otras universidades y fueron muchas las que copiaron la idea, tanto en Estados Unidos de América como en Europa, conectando dispensadoras de refrescos o máquinas de café. Sin embargo, no es hasta 1999 que no se acuña el término de IOT por parte de Kevin Ashton.

Internet ha permitido que la información fluya sin distancias ni fronteras. Nos hemos acostumbrado rápido a buscar en los navegadores, mandar y recibir mensajes en las redes sociales, a comprar la comida o artículos sin salir de casa y a ver series, películas o la televisión en vivo. Lo que nos sigue pareciendo novedoso, pese a que existe desde hace más de 30 años,

es la automatización de lo doméstico. Gracias al desarrollo tecnológico, los sistemas de IOT son más accesibles, hay mayor variedad y son más robustos y seguros. Esto ha propiciado un cambio en la tendencia en el crecimiento de los dispositivos conectados a Internet, duplicándose el número en los últimos cinco años y con unas perspectivas de futuro exponenciales. No cabe duda que las grandes compañías de nuevas tecnologías como IBM (Watson IOT Platform), Microsoft (Azure IOT), Google (Cloud IOT), Amazon (AWS IOT), Intel (Intel IOT) o Alibaba (Cloud IOT) están apostando fuerte por sus sistemas integrados con Inteligencia Artificial (IA).

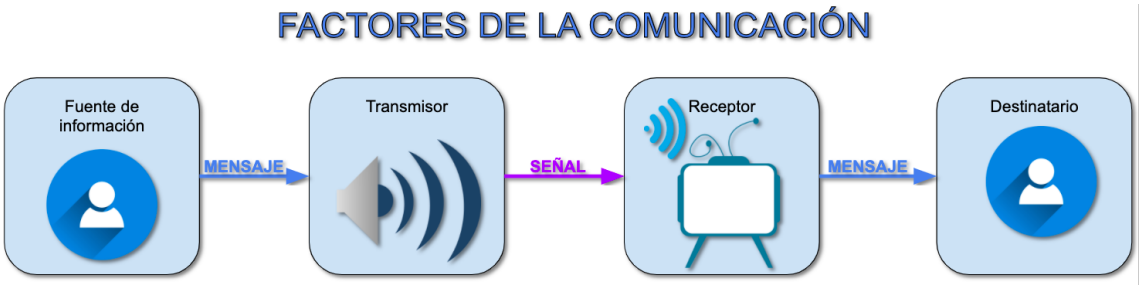


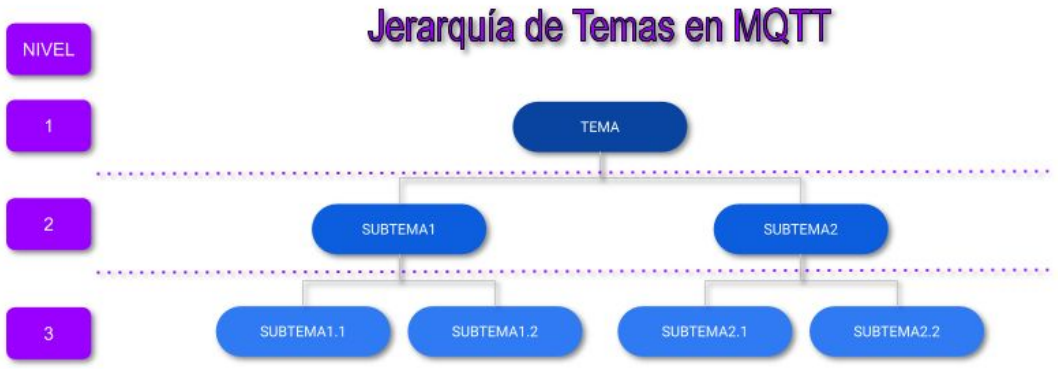


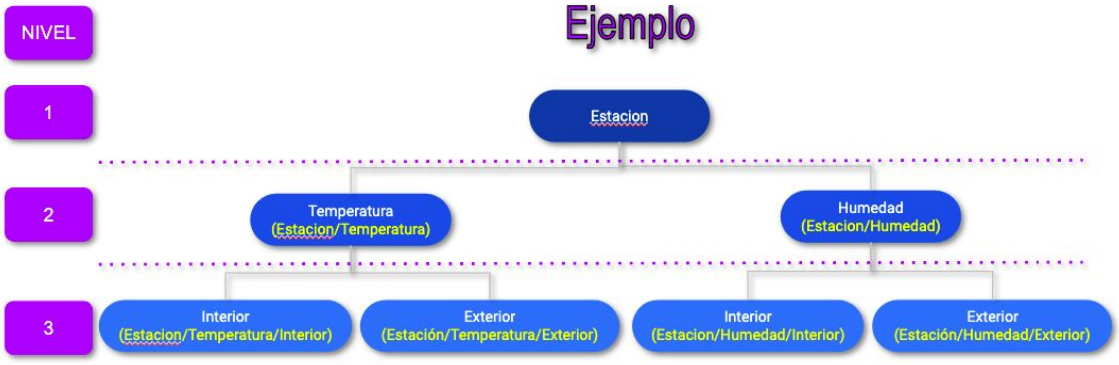

Con la tecnología MQTT y los dispositivos de bajo coste basados en los SOC de Espressif ESP8266 y ESP32 se ha vuelto accesible, independientemente de la formación en programación o en electrónica. Es importante, antes de seguir, definir MQTT y toda la terminología asociada.

- **MQTT** (Message Queue Telemetry Transport): se trata de un protocolo de transporte de mensajes para la telemetría tipo Cliente/Servidor basado en publicaciones y suscripciones a tópicos o temas. Fue inventado y desarrollado por IBM en 1999 con el fin de conectar sensores de oleoductos vía telefónica o satélite. En 2013, lo libera a OASIS (organización sin ánimo de lucro de estandarización avanzada) y se declara un estándar ISO (ISO/IEC 20922). Actualmente existen librerías para emplear MQTT en la mayoría de los lenguajes de programación.
- **Mensaje**: es el objeto de la comunicación, en general. En el caso de un sistema MQTT, un cliente publica un mensaje bajo un tema al broker o servidor y este se lo suministra a todos los clientes que están suscritos a dicho tema. Para




- **Tema** o tópico (**topic**): en MQTT, nombre, en UTF-8, que emplea el broker para filtrar mensajes a los clientes. Corresponde al valor de una señal proveniente de un sensor, de monitorización o la orden a un actuador concreto. Los temas o tópicos pueden estar a varios niveles. Si un cliente se suscribe a un nivel superior, recibe mensajes del tema en cuestión y de aquellos que están a niveles inferiores. En la plataforma Adafruit se llaman "Feeds".



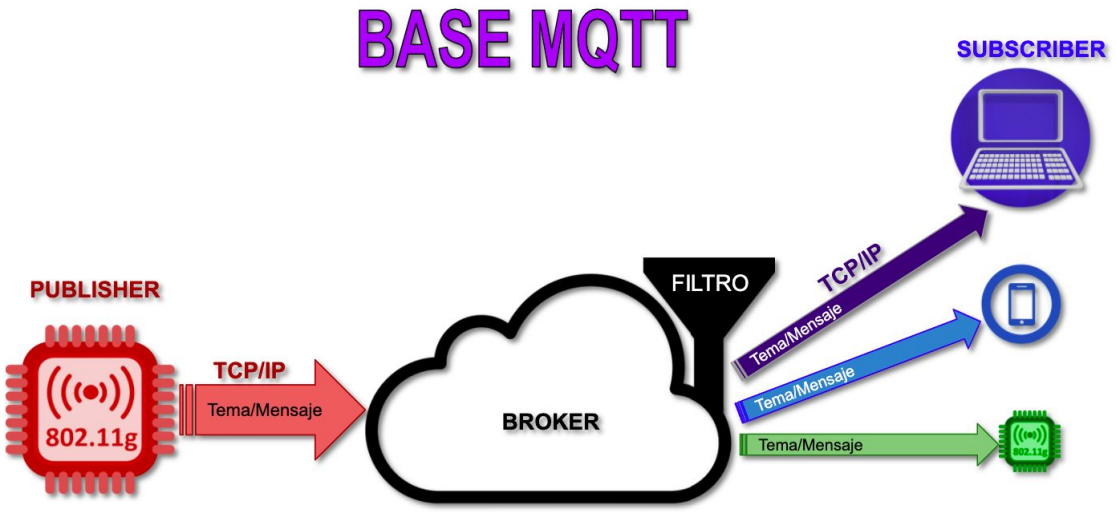




- **Publicación**: un cliente manda el mensaje de un único tema al servidor o broker.
- **Suscripción**: registro, por parte de un cliente, en el servidor para recibir notificaciones o mensajes en un determinado tema. Los clientes pueden suscribirse a unos o varios temas.

- **Cliente** MQTT: cualquier dispositivo en contacto con el servidor o broker. En nuestro caso hay dos tipos: el teléfono u ordenador que sirve para monitorizar temas o mandar órdenes y placas microcontroladoras que publican temas al servidor.
- **Broker** MQTT: es un programa servidor de MQTT, acepta mensajes publicados por clientes y los difunde entre los clientes suscritos. Publicar: cuando un cliente envía un mensaje al broker. Tópico: los mensajes deben estar etiquetados con algún tópico o tema. Con esta figura, se separa el emisor del receptor tanto en el tiempo como en el espacio, es decir, que al almacenarse el valor, cuando el emisor envía el mensaje, el receptor no tiene porqué estar conectado y viceversa ni en el mismo lugar.
- **Telemetría** (Telemetry): las señales publicadas por los dispositivos, se monitorizan de manera remota en Tableros o (Dashboards).
- **Comando** (Command or State change): desde el Dashboard se puede enviar órdenes de cambio de estado en los dispositivos.

## 4.2 El protocolo MQTT

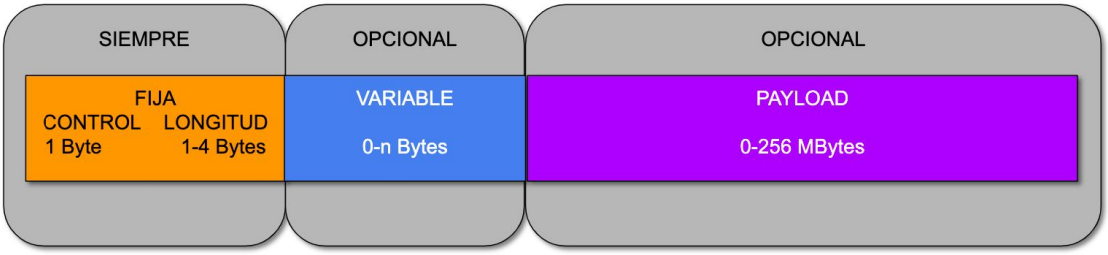
MQTT es un servicio de mensajería de máquina a máquina M2M (Machine to Machine). El servidor se denomina Broker y los clientes se conectan a él publicando o suscribiendose a temas. A esto se denomina mensajería tipo push con patrón pub-sub. El broker filtra los mensajes enviados a cada cliente como temas que se organizan jerárquicamente. Dicho de otra manera, un cliente (PUBLISHER) manda (publica) al servidor (BROKER) un mensaje en un "Tema" (Topic) y el Broker lo almacena, cualquier cliente suscriptor (SUBSCRIBER) de ese "Tema" que lo requiera, recibirá, de parte del Broker, un mensaje que ha sido filtrado.



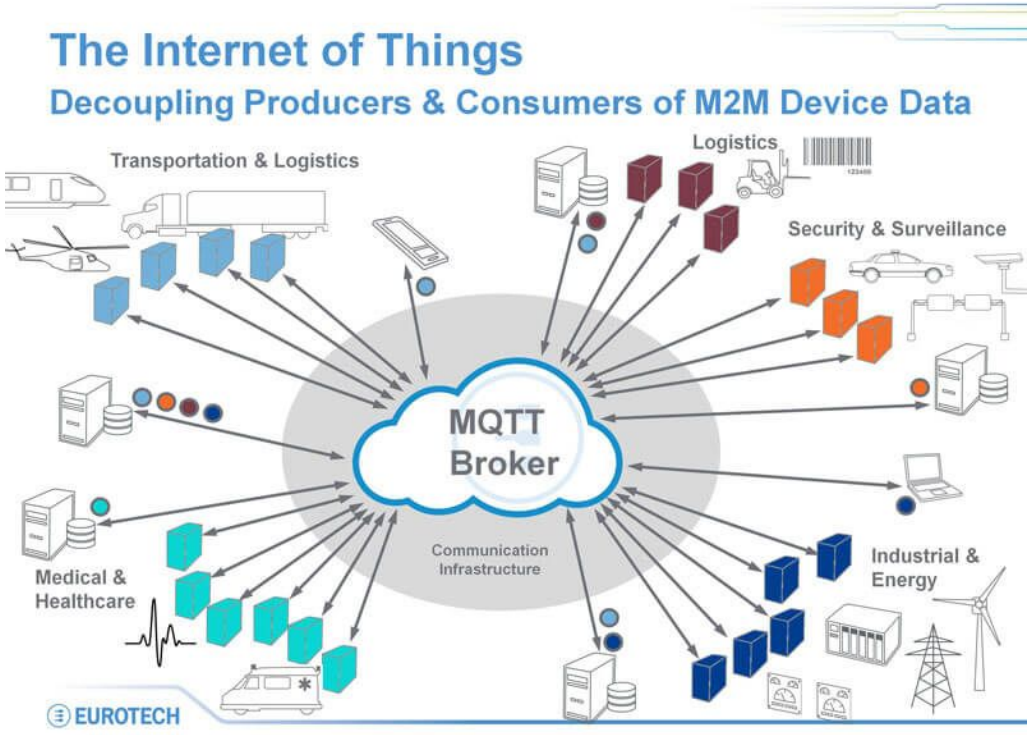

La estructura de los mensajes MQTT consta de 3 partes:

- Cabecera fija: está compuesto por un código de control (1 byte) que identifica el tipo de mensaje y la longitud del mensaje (longitud entre 1 y 4 Bytes).
- Cabecera variable: es opcional y aporta información adicional que acompaña a determinados mensajes.
- Contenido (payload): es el contenido del mensaje en sí. Puede tener un máximo de 256MB aunque, normalmente, no suele sobrepasar 2-4 KBytes.

# ESTRUCTURA DE UN MENSAJE MQTT




Lo cierto es que este sistema tiene multitud de aplicaciones puesto que consume muy pocos datos y no necesita, en la mayoría de las ocasiones, grandes anchos de banda (velocidad de comunicación). Esto repercute en la posibilidad de emplear dispositivos de bajo consumo y aparatos alimentados por batería. Dispone, además, de medidas de seguridad y calidad del servicio (QoS) que le da robustez y fiabilidad. Es pues, uno de los sistemas preferidos para el Internet de las cosas (IOT).




MQTT tiene tres niveles de calidad de servicio (QoS) a usar en función de las características y necesidades de fiabilidad de nuestro sistema:

1. QoS 0: el mensaje se envía una única vez. Al no requerir el OK del receptor, puede que el mensaje se pierda. (Unacknowledged at most one)
2. QoS 1: El mensaje se reenvía las veces que haga falta hasta que el receptor indica que lo ha recibido correctamente. El receptor puede recibir más de una vez el mensaje. (Acknowledged at least one)

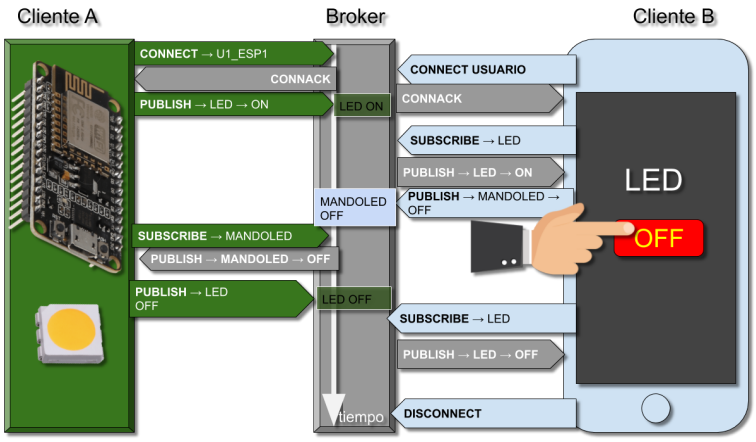
3. QoS 2: Se asegura la recepción del mensaje una vez (assured exactly one)

Mayor nivel de calidad, supone una mayor carga del sistema con un mayor número de mensajes de verificación intercambiados.

Por otro lado, en cuanto a la seguridad, es un factor importante si nuestro sistema IOT no sólo sirve para monitorizar señales o si estas deben mantenerse dentro del ámbito privado. MQTT permite y normalmente emplea más de una de las opciones siguientes:

- Transporte SSL/TLS. SSL (Secure Sockets Layer) es una capa de conexión segura por medio de un protocolo que hace uso de certificados digitales para establecer comunicaciones seguras a través de Internet. TLS (Transport Layer Security) es una capa de seguridad de transporte, es el sucesor del SSL. Funcionan de manera parecida cifrando la información para proteger la transferencia entre dos entes, el cliente y el servidor. Por defecto, emplea el puerto 1883 para SSL y el 8883 cuando funciona sobre TLS,
- Autenticación por usuario y contraseña y
- Autenticación por certificado: consiste en el uso de un certificado digital (credencial) para identificar a un dispositivo concreto utilizado por un usuario conocido de la red. Es decir, permite comprobar que los dispositivos conectados a la red de la organización cuentan con las autorizaciones pertinentes.

La secuencia en la comunicación, empieza con el cliente que envía un mensaje (CONNECT) que contiene la información necesaria: usuario, contraseña, client-id…. Entonces, el Broker responde con un mensaje (CONNACK), indicando si la conexión ha sido aceptada o rechazada.




Una vez conectado el cliente al Broker, para enviar mensajes al servidor se emplea PUBLISH, que contienen el Tema (topic) y el contenido (payload).

Los clientes que quieren suscribirse, emplean mensajes (SUBSCRIBE) indicando el tema objeto de la suscripción, debiendo, el Broker responderle (SUBACK) con el contenido (payload) del tópico.

De vez en cuando, para saber si los clientes están activos, estos mandan mensajes (PINGREQ) y el servidor les responde con un “PINGRESP”.

Finalmente, si un cliente se quiere desconectar envía un “DISCONNECT”.

Para eliminar la suscripción, el cliente envía un “UNSUBSCRIBE”, respondiendo el Broker como “UNSUBACK”.

## 4.3 Plataformas MQTT

En el Internet de las cosas, se emplean varios protocolos. HTTP, CoAP y MQTT son los más usuales. Últimamente las plataformas basadas en MQTT han cobrado cierta popularidad y la oferta es amplia, las hay Open Source como Mosquitto para instalar en servidor propio o de terceros, las hay comerciales con plan gratuito (con limitaciones) como Adafruit.io o Thingspeak y otras con planes escalables pero sin plan gratuito como AWS IOT de Amazon. Algunos de ellos son:

- OpenSource: Mosquitto, Aedes (Mosca), Thingsboard, DeviceHive (HiveMQ CE), HBMQTT, EMQTT, RabbitMQ, ActiveMQ, Moquette, MQTTnet...
- Comerciales con planes gratuitos: Adafruit.io, Thingspeak, Cayene, MyQttHub.com, Kaa Cloud...
- Comerciales para grandes empresas: IBM Bluemix, Amazon AWS IOT SDK, UBIDOTS, TheThings.io...

La escalabilidad en los planes de precio de estas plataformas reside en varios factores como:

- Número de dispositivos clientes.
- Número de usuarios.
- Número de temas.
- Tráfico de mensajes por unidad de tiempo: minutos, horas, días o mes.
- Interacción con bases de datos externas y capacidad de almacenamiento.

Para instalar en proyectos domésticos y locales, el más conocido es Mosquitto. Se puede instalar, además de en servidores, en PC sobre Windows, Linux y Mac y sobre miniPC como RaspberryPi o BeagleBone-Black. En este curso, nos centraremos en las plataformas online de Adafruit.io y Thingspeak. Las dos tienen soporte para placas con microcontrolador Expressif ESP8266, pero sólo Adafruit admite para ESP32.

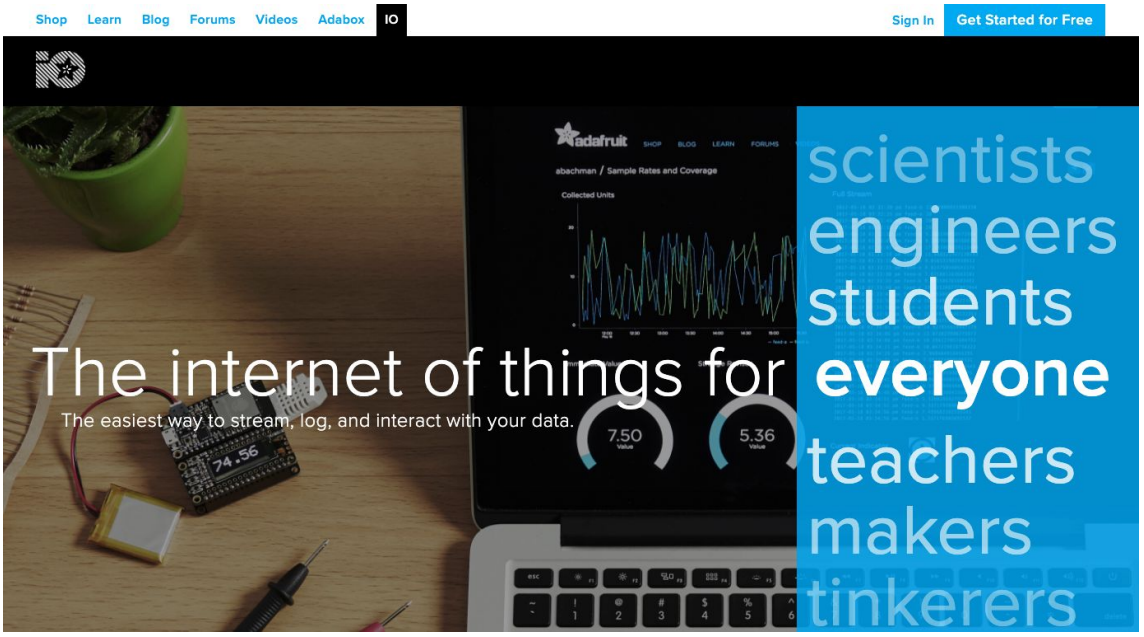
| Plataformas WEB | Adafruit IO | | ThingSpeak | |
|---|---|---|---|---|
| WEB | https://io.adafruit.com/ | | https://thingspeak.com/ | |
| Denominación comercial del plan | IO Free | IO+ | Gratuito | Estudiante |
| Precio | 0€ | 99$/año | 0€ | 250€/año |
| Monitorización de topics | Sí | Sí | Sí | Sí |
| Control de cambio de estado de topics | Sí | Sí | Sí | Sí |
| Nº de topics | 10 | ilimitado | 8200/día | 90000/día |
| Nº de Dashboards | 5 | ilimitado | 4 | 250 |
| Nº de datos por minuto | 30 | 60 | 4 | 30 |
| Nº de Triggers (notificaciones) | cada 15 min | cada 5s | cada 15 s | cada 15s |
| Tiempo de almacenamiento de los datos (días) | 30 | 60 | 1 año | ilimitado |

## 4.4 MQTT de Adafruit

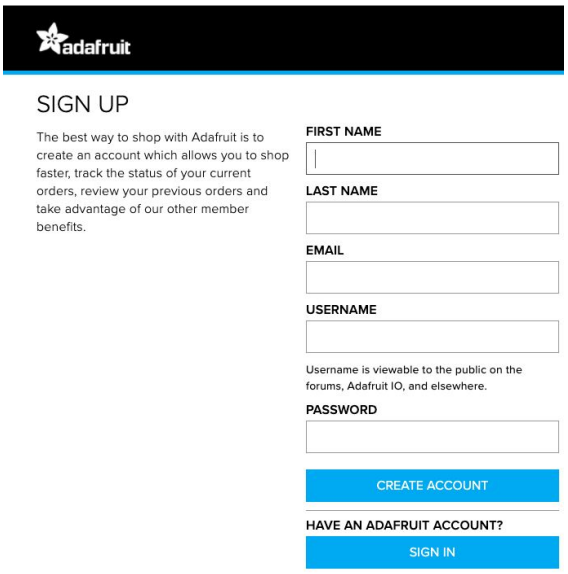
La empresa Adafruit fue fundada en 2005 por Limor Fried, conocida hasta entonces por su blog de electrónica y programación así como en las redes sociales por "Ladyada". Es ingeniera por el MIT (Instituto Tecnológico de Massachuset, Boston) y miembro del Consejo Asesor de la ciudad de Nueva York. Su objetivo era crear un portal WEB para aprender electrónica y facilitar los productos adaptando y ofreciendo diseños para todas las edades y niveles de habilidad en el montaje y construcción de proyectos en base a microcontroladores Arduino compatibles. En los últimos 10 años, Adafruit ha crecido a más de 100 empleados en el corazón de Nueva York con una fábrica de más de 4500m$^2$. En la página web de Adafruit podemos encontrar la tienda pero también una sección de aprendizaje con tutoriales, blogs y foros que es de obligada lectura para todos los aficionados e iniciados en montajes con Arduino o derivados como las placas basadas en ESP8266 que estamos empleando en este curso.


Durante el brote de COVID-19, Adafruit Industries está operando como un servicio esencial y negocio de fabricación de EPIs y componentes de dispositivos médicos.

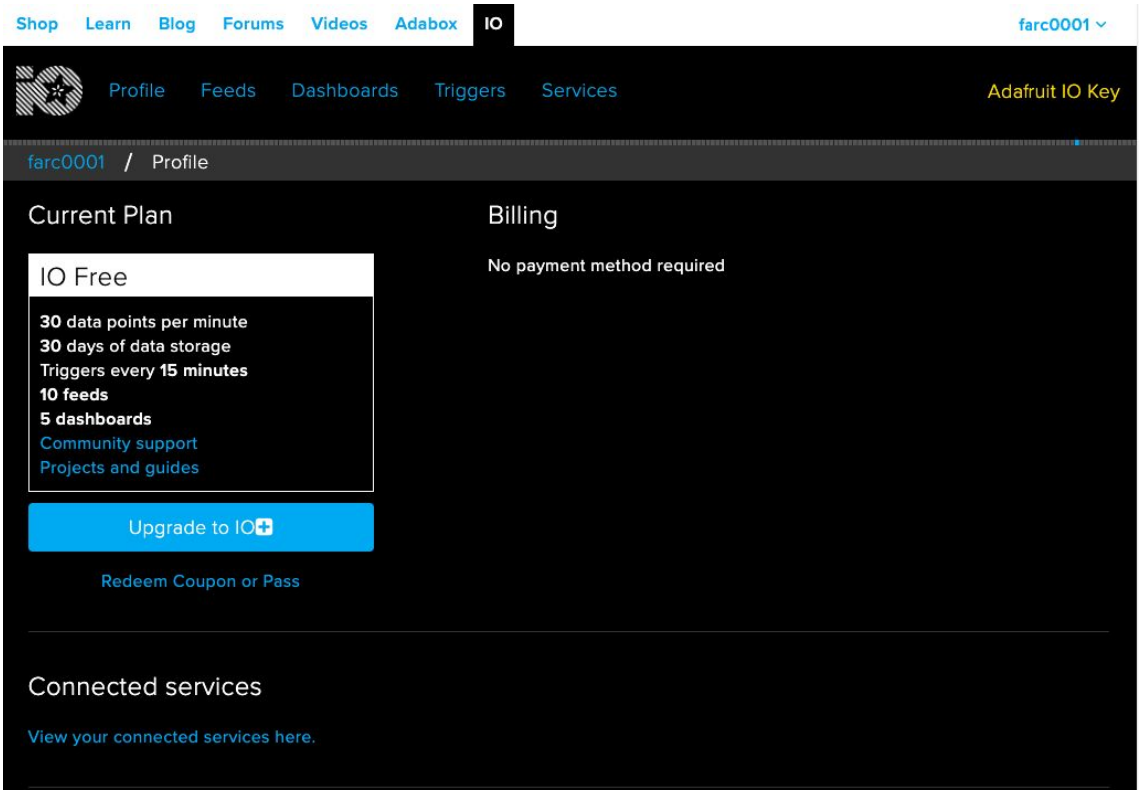
La plataforma MQTT que ofrece Adafruit se puede acceder a través de su WEB corporativa pinchando en la pestaña "IO" o bien desde la dirección: https://io.adafruit.com.




El siguiente paso consiste en registrarse seleccionando "Get Started for Free" y rellenando los campos que aparecen y pulsando sobre "Create account".




Una vez hecho y activado, se puede entrar en el entorno de la plataforma, seleccionando "Sign In" de la página principal.




Al entrar lo que nos encontramos la página principal (Home) del perfil (Profile) de la cuenta con el Plan gratuito "IO Free" y sus características: hasta 30 datos por minuto, hasta 30 días de almacenamiento, Triggers (notificaciones) cada 15 minutos, hasta 10 Feeds (temas o topics) y 5 dashboards (tableros o entornos de monitorización). Tienes también la posibilidad de ampliar "IO+" el plan pagando 99$/año, hasta 60 datos por minuto, 2 meses de almacenamiento, triggers cada 5 segundos y sin límite de Feeds y Dashboards.

Las pestañas de navegación dentro la plataforma "IO" son:

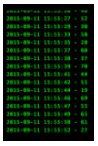
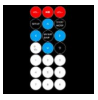
- **Profile:** es el perfil de la cuenta. Tiene varias opciones:

  - Home: la página principal del plan contratado,
  - Public: donde se muestra los Dashboards y Feeds que tenemos compartidos en la red, es decir, que los tenemos publicados,
  - Sharing: Feeds compartidos con otros usuarios o que me han compartido a mi.
  - Monitor: indica el consumo de datos, Feeds, Dashboards y, en directo, informa de errores, conexiones activas y datos transmitidos.
  - Activities: registra actividades en la cuenta como si se ha cambiado de **IO Key** (clave de cifrado única para cada usuario).
- **Feeds:** para crear y configurar los temas o topics. También se pueden descargar los datos en un fichero ".csv",
- **Dashboards:** para diseñar los tableros o entornos de monitorización,
- **Triggers:** para configurar las situaciones en las que deba realizarse una notificación. Por ejemplo si el valor del tema "temperatura" llega a 30ºC, salta un trigger.
- **Services:** conectar tu sistema a servicios externos como:
  - IFTTT o Zapier: interacción entre App de dispositivos inteligentes como teléfonos, tablets y sistemas ,
  - Weather: ofrece datos del tiempo atmosférico cada 20 minutos,
  - Randomizer: para generar datos o claves aleatorias o
  - Time: ofrece sincronización de fecha y hora similar a NTP, esto permite que no sea necesario incorporar a todos los montajes circuitos RTC (Real Time Clock) ya que la sincronización se gestiona como un tema o Feed construyendo un reloj, un cronómetro o sincronizando varios dispositivos a la vez.

El procedimiento de funcionamiento, para cualquier proyecto, es sencillo:

1. Diseñar el Tablero "Dashboard" con los elementos de monitorización y control de los "Feed":
   a. Crear los elementos y asociarlos a los Feed
   b. Organizar (ordenar espacialmente) los elementos

| | | |
|---|---|---|
|  | Editar aspecto del tablero. En este apartado podremos ordenar los elementos del tablero y guardarlo al final. | |
| |  | Si está activado, permite un ancho de alta definición con 1300px |
| |  | Si está activado, pone la organización de los elementos en vertical compacto. |
| |  | Salir sin guardar los cambios realizados. |
| |  | Salir guardando los cambios realizados. |
|  | Añadir elemento del tablero | |
| |  | Interruptor (Toggle): permite enviar una orden remota para cambiar algo en el dispositivo. Con dos posiciones estables. Por ejemplo el estado de un LED o encender y apagar un aparato ayudándonos de un relé. Permite poner un texto para encendido y otro para apagado así como asignarle un nombre al elemento. |

| | | |
|---|---|---|
| |  | Pulsador (Momentary Button): Permite enviar una orden remota para cambiar algo en el dispositivo. Es monoestable. |
| |  | Barra de valor (Slider): para establecer referencias y enviarlas al dispositivo. |
| |  | Indicador circular (Gauge): indicador de valor con límites mínimo y máximo que si se traspasan, cambia de color. |
| |  | Texto (Text): para poner texto en el tablero |
| |  | Datos en tiempo real (Stream): muestra los datos conforme se van recibiendo del tema en cuestión. Se puede mostrar fecha, hora, tema (Feed) y valor recibido así como localización geogra |
| |  | Imagen (Image): Permite colocar imágenes, hasta 64, que van cambiando en función del valor del Feed. |
| |  | Gráfico histórico (Line Chart) de los datos de un determinado tema o Feed. |
| |  | Selector de color(Color Picker): sirve para mandar un color seleccionado con el ratón en formato hexadecimal. |
| |  | Mapa (Map): permite geolocalizar sobre el mapa los datos que se van recibiendo por el Broker. |
| |  | Control remoto (Remote control): Reproduce un mando a distancia de los que se utilizan en los Kit de Arduino para enviar información. |
| |  | Icono (Icon): permite mostrar un icono de la lista o ir cambiando porque se identifican por un nombre. |
| |  | Indicador (Indicator): es de tipo ON - OFF pudiendo asignar un color a cada estado. |
| |  | Teclado numérico (Number Pad): Transmite al dispositivo un Byte con el valor del botón pulsado de 0 a 9, “*” y “#”. |
| |  | (Multiline Text): muestra cadenas de texto transmitido por un Feed. |
| |  | Enlace a Tablero (Dashboard link): permite saltar a otro tablero que tengas. |

| | |
|---|---|
|  | Borrar el tablero (Dashboard) entero. |
|  | Muestra la contraseña secreta del usuario “IO Key” |
|  | Indica que el tablero es de uso privado, si pinchamos y aceptamos el mensaje siguiente, lo volvemos público. |
|  | Vista a toda página |

2. Elaborar el programa para la tarjeta ESP8266 teniendo en cuenta que:
    a. En el Setup(): Configuración de la conexión del cliente con el Broker (bloque Iniciar dentro de los bloques de MQTT). Los datos que necesitas para hacerlo son:
        i. Relativos a la WiFi donde conectes el dispositivo cliente NodeMCU: SSID y clave.
        ii. Dirección WEB del broker: “io.adafruit.com”
        iii. Puerto: “1883” (para seguridad SSL)
        iv. Cliente Id: es el nombre que le quieras dar a la placa o dispositivo dentro del sistema MQTT, debe ser único luego suele incluir el nombre deusuraio.
        v. Usuario: es el nombre de usuario que tienes en tu cuenta de Adafruit IO.
        vi. Clave: corresponde a la “Adafruit IO Key”. La obtienes desde cualquier ventana de Adafruit IO pinchando en la esquina superior derecha.
    b. Comunicación de los temas o Feed: Necesitas saber el nombre que le has puesto a los Feed y ponerlo en “Feed Key”. Por otro lado, si:
        i. Si envías un valor al broker → Coge el bloque de “Publicar tema” o Feed y el correspondiente de “adafruit.io publish”.
        ii. Si quieres recibir un valor del broker → Coge el bloque de “Suscribirse tema” o Feed y el de “adafruit.io subscribe”.

## 4.5 Ejemplo 1: Control de un LED

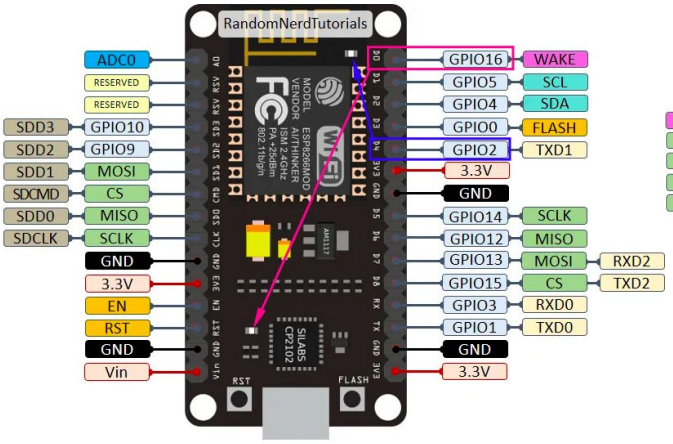
Este ejemplo consiste en la monitorización y el control de un LED desde Adafruit.io y se realizará en dos pasos: primeramente, se realizará el control y monitorización del LED de la propia placa del NodeMCU desde Adafruit IO y, después, se incluirá un pulsador que permita su conmutación en la placa y un elemento de control remoto en el Dashboard.

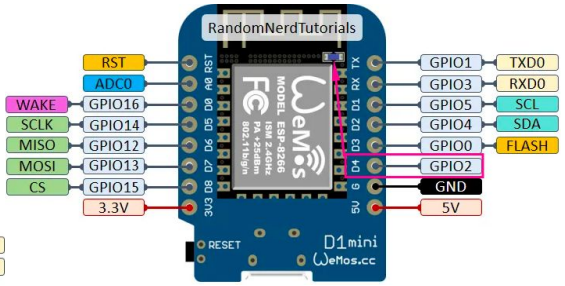
Antes de seguir, indicar que en la placa NodeMCU hay 2 LED azules:

- GPIO 16 (D0) → Exclusivo de la NodeMCU
- GPIO 2 (D4) → del módulo ESP12, que lleva el ESP8266, soldado a la placa. El mismo que tienen las placas: NodeMCU, WEMOS, WEMOS D1 mini.

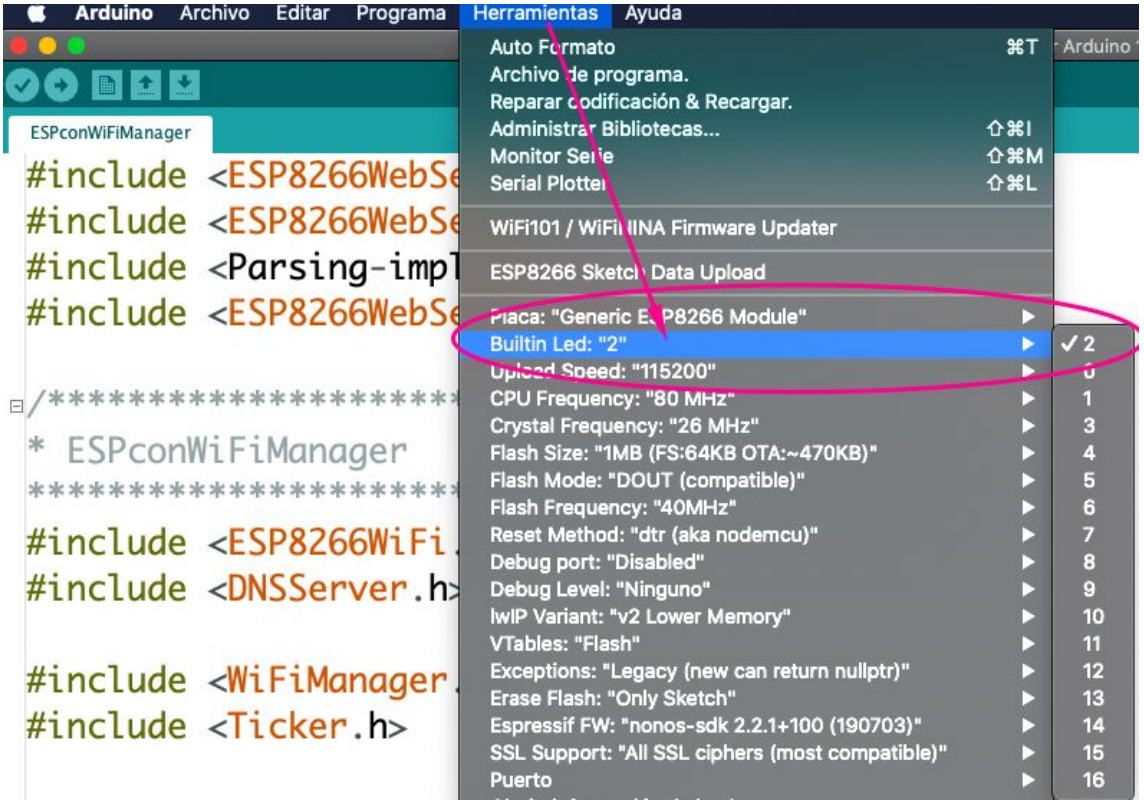
Ambos están montados en lógica inversa, es decir, cuando mandas un “ON” se apagan y cuando mandas un “OFF” se encienden.

En la placa ESP01, para conectar con Arduino UNO vía serie, el LED de la placa, en función de la variante o versión, está en GPIO1 (D10) o en GPIO2 (D4), por lo que deberás probar antes de conectarte a IOT.




Puedes configurar el LED_BUILTIN en el IDE de Arduino para no tener que introducirlo manualmente en los programas.




El material necesario para los pasos del ejemplos son:

- Placa NodeMcu o WEMOS D1 mini.
- Cable USB de alimentación.
- Módulo pulsador que incorpore resistencia[1] o Pulsador + Resistencia de 10K-1M Ohmios (1/4W).
- Protoboard
- Cables

[1] ArduinoBlocks no contempla pines INPUT_PULLUP. Por lo que conectar un pulsador diréctamente entre un pin y GND no será bien interpretado por el programa. Si se quiere, se puede descargar el código ".ino" y modificar el tipo de entrada en el IDE de Arduino.

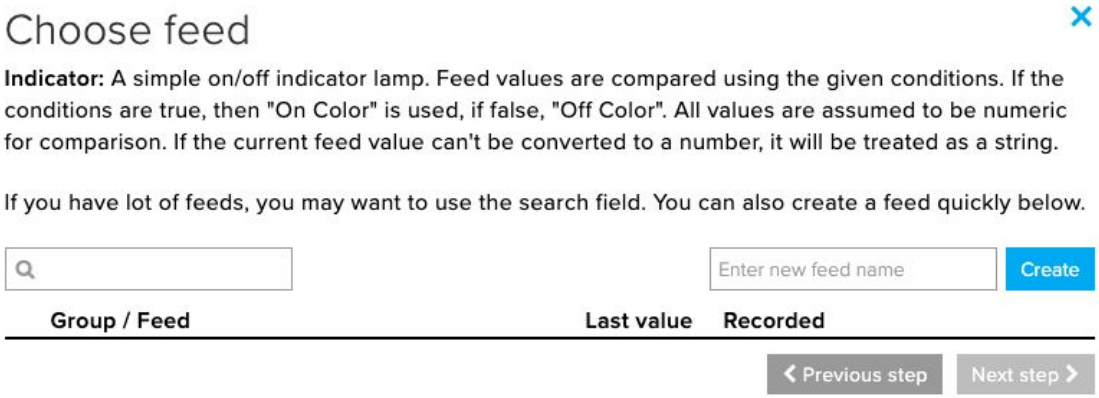
## PASO 1: Monitorizar el led

**Esquema de conexión:**

Para el primer paso, sólo necesitaremos la placa y el cable de datos USB que nos servirá, también, de alimentación. No hay que hacer ningún montaje.

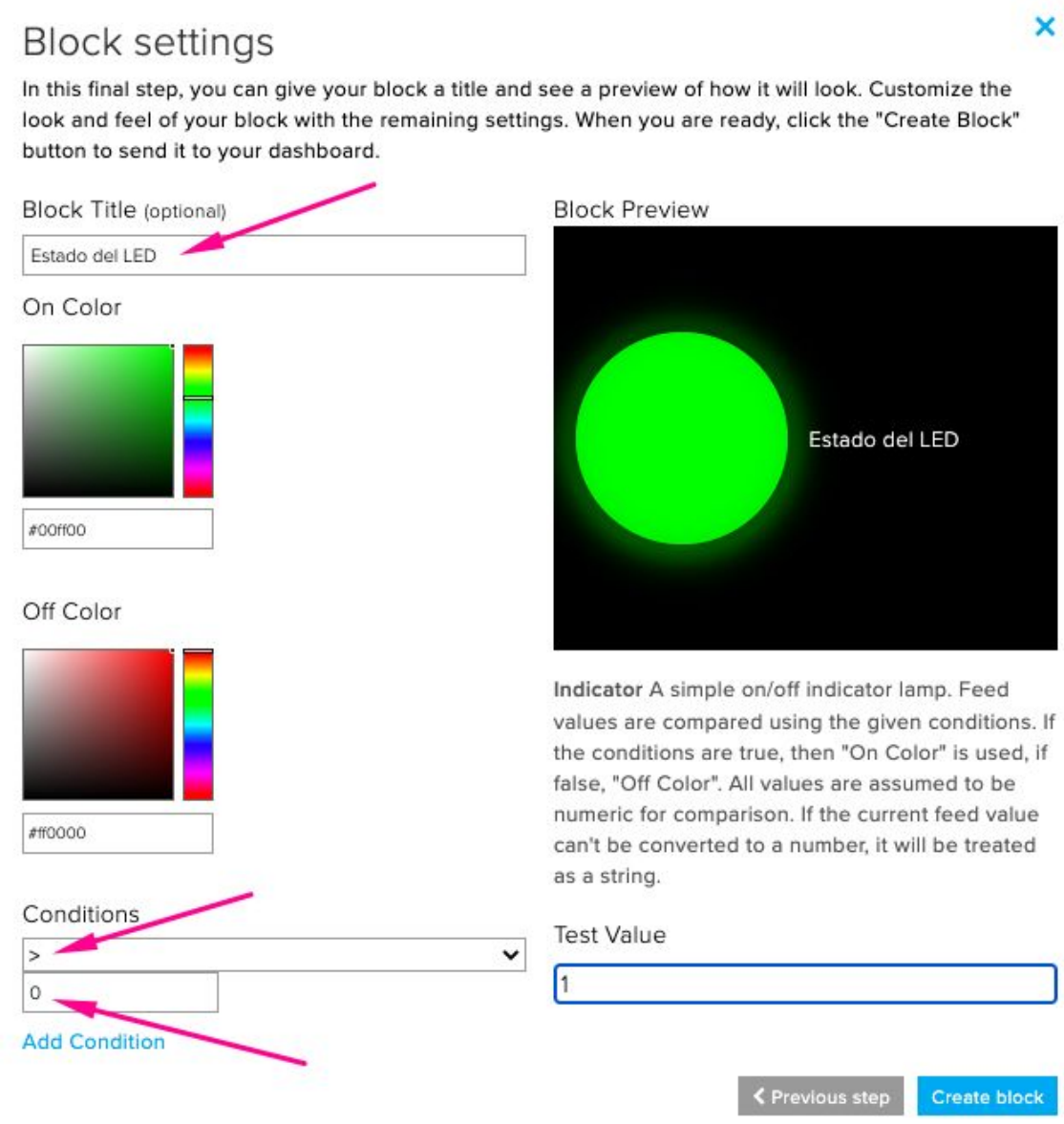
**Solución propuesta:**

- Primero configuraremos la parte del Broker MQTT, por lo que nos iremos a Adafruit IO.
  - Crearemos un tablero de monitorización (Dashboard). Para ello nos iremos a la opción de navegación **"Dashboards"** y seleccionamos **"view all"**, nos mostrará una nueva pantalla y en **"Actions"** podremos optar por **"Create a new dashboard"**, acto seguido, le pondremos el nombre al tablero, para su posterior identificación, una descripción y pulsamos el botón **"Crear"**.
  - Una vez creado el tablero, fijaos que nos ha creado una tabla con "Name", "Key" y "Created At". En las filas subsiguientes están los nombres que le hemos puesto a los Dashboards creados, el nombre llave y la fecha de creación, correspondientemente.
  - Pinchamos en el tablero que acabamos de crear y aparece la pantalla de edición del tablero con una serie de iconos y seleccionaremos + seguido de "Indicator". Aparecerá entonces la siguiente pantalla.



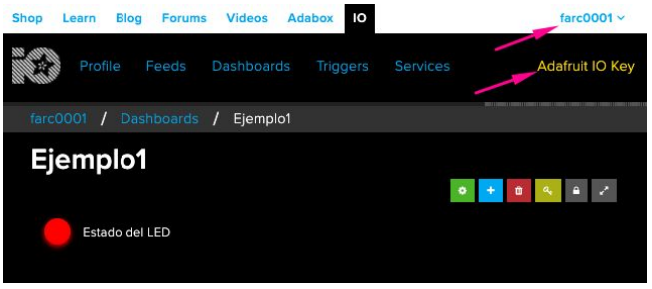

Indica que el elemento del dashboard escogido debe estar asociado a un Feed. Como es la primera vez que hacemos algo, no tenemos ningún Feed. Para crear uno, Hay que ir al campo de texto de la derecha y poner un nombre al Feed, en este caso lo llamaré "LED" y pulsaré sobre el botón "Create". Ahora me aparecerá la opción "LED". La selecciono y pulso al paso siguiente "Next step".

  - En el PopUp que aparece el pongo título al elemento ("Block Title") y modifico las condiciones con ">" y "0" para que al poner en:
    - "Test Value" → 0 el icono esté rojo
    - "Test Value" → 1 el icono esté verde
  - Ya sólo queda pulsar en "Create block".




El resultado será algo tan sencillo como lo que se muestra en la imagen siguiente.

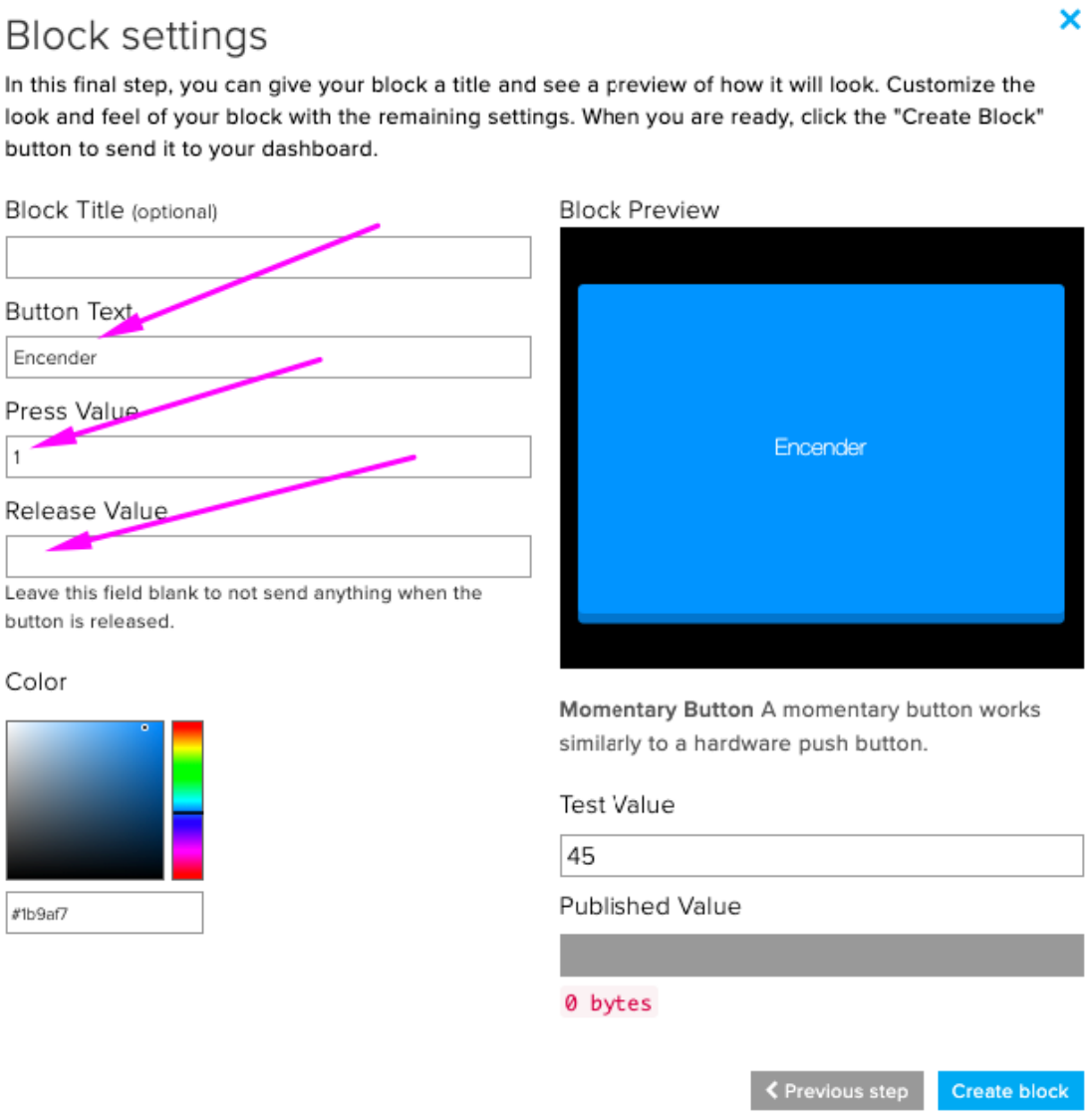
Ahora, de la plataforma de Adafruit tenemos que preparar la siguiente información para el programa de la placa:




- Nombre de usuario, en mi caso “farc0001”
- Adafruit IO Key
- Nombre del Feed, en este caso, “LED”

Para enviar los comandos de encender y apagar el LED, emplearemos dos botones

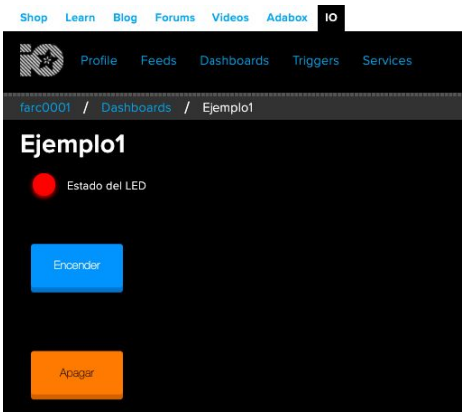
- Pulsamos en + y seleccionamos RESET, creamos un Feed nuevo “EnciendeLED” y hacemos click en “Next step”
- Ponemos “Encender” en el cuadro de texto de “Button text” para que aparezca en el pulsador.
- Borramos el valor, que mandaría si no se está pulsando, de “Release Value” porque nos saturaría el plan de datos y dejamos el valor de pulsado a “1” (Press Value).
- Le damos a “Create Block”




Hacemos lo mismo con el botón de apagado:

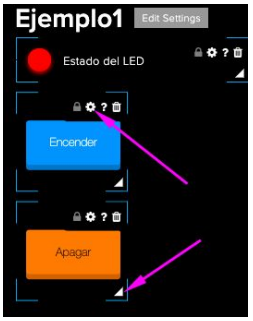
- Pulsamos en + y seleccionamos RESET, creamos un Feed nuevo “ApagaLED” y hacemos click en “Next step”
- Ponemos “Apagar” en el cuadro de texto de “Button text” para que aparezca en el pulsador.

- Borramos el valor, que mandaría si no se está pulsando, de “Release Value” porque nos saturaría el plan de datos y dejamos el valor de pulsado a “0” (Press Value).
- Cambiamos el color del botón y lo ponemos en un rojo apagado.
- Le damos a “Create Block”




Una vez creados todos los elementos del tablero los ordenamos para que esté mejor presentado.

- Seleccionamos  y el aspecto cambia porque cada elemento es editable en tamaño y ubicación, símplemente moviéndolos con el ratón. También se pueden editar las características de los elementos.




Con lo que, al final, queda como en la siguiente imagen tras guardarlo con “Save” 

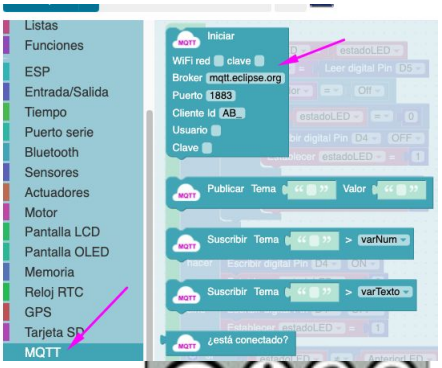
.


Una vez personalizada la interfaz de monitorización y control, pasamos al programa de la placa, para lo cual iremos a ArduinoBlocks.

Antes de seguir hay que hacer dos consideraciones relacionadas con el número de valores que puedes transmitir por minuto con el plan gratuito, que son 30, es decir, 1 cada 2 segundos y que si empleamos intrucciones como “delay(ms)” el procesador bloquea toda ejecución y se puede desconectar de Adafruit y hasta de la comunicación WiFi:

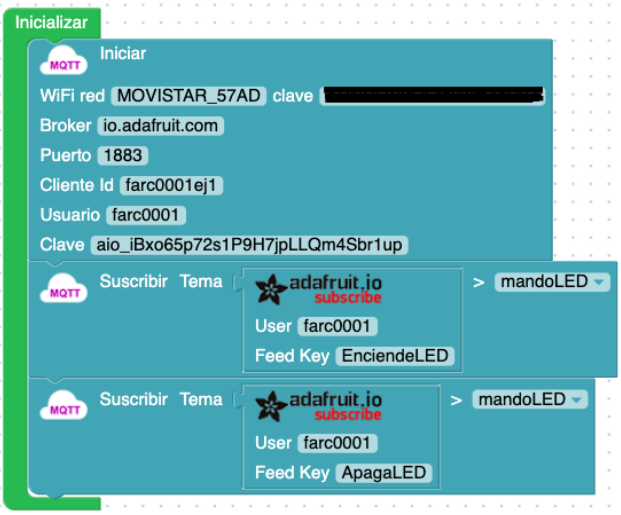
- Para que no dé problemas, es mejor separar más de 2 segundos las transmisiones.
- Evita el uso de la sentencia “delay(ms)”.
- Emplea técnicas de paso de testigo como la del semáforo, que sólo envíe en caso de cambio de estado o usa funciones como “Ejecutar cada (ms)” que no bloquean nada.

Para este ejemplo, sólo transmitimos desde el cliente NodeMCU, el estado del LED cada 5 segundos. Pero primero empecemos por programar la conexión a la WiFi y a la plataforma MQTT sacando el bloque de “MQTT Iniciar” del catálogo “MQTT”.



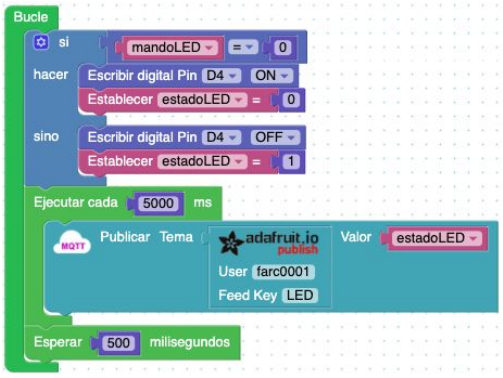

- En WiFi red ponemos el SSID y la Clave de la WiFi a la que vamos a conectar el dispositivo.

- En Broker, hay que poner la dirección del servidor de Adafruit → "io.adafruit.com"
- En Puerto → "1883" para SSL.
- En Cliente, hay que poner el nombre que tendrá el dispositivo en la plataforma MQTT, debe ser único en Adafruit IO, independientemente del usuario (si no fuera así no se podría compartir).
- En Usuario, es el nombre de usuario de la cuenta de Adafruit IO
- En Clave, corresponde a la Adafruit IO Key. (Esta se puede cambiar en caso necesario)




Después, se indica al Broker, los Feed a los que el cliente NodeMCU se quiere suscribir, es decir, los valores de los temas que quiere recibir y en qué variable del programa los vamos a guadar. Los bloques se sacan, también, del catálogo MQTT. En el ejemplo presentado correspondiente a la imagen anterior, se indica que se suscribe a los feed: "EnciendeLED" y "ApagaLED" guardando los valores en "mandoLED".

Tras configurar la conexión, se aborda el resto del programa de la siguiente manera:




- Cuando la NodeMCU recibe un valor correspondiente a un tema, del Broker, se guarda en la variable "mandoLED" y evalúa si se ha recibido un "0" (ApagaLED) o un "1" (EnciendeLED).
- Si la orden es de encender el LED, al pin D4 (GPIO2) se le manda una señal baja ya que el LED está configurado en inversa.
- Si la orden, por el contrario, es de apagar el LED, el pin D4 se pone a nivel alto.
- Después, cada 5 segundos se publica al Broker el estado actualizado el estado del LED.
- Finalmente esperamos 500ms (pueden ser menos) para que se produzcan 2 bucles por segundo.

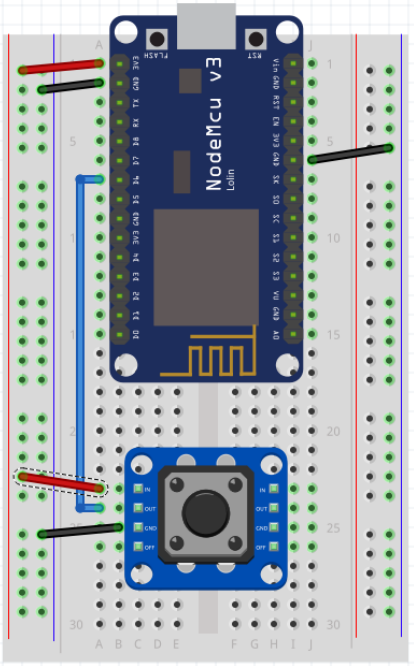
Podemos comprobar que, al alimentar la placa NodeMCU y, tras un instante para conectarse, al pulsar el botón de encender, el LED de la placa se enciende y que, al pulsar el botón Apaga, el LED se apaga.

## Paso2: Controlando el LED desde la placa y Adafruit IO.

El siguiente paso, que puede resultar, a priori, sencillo, consiste en añadir al circuito electrónico un botón que permita el cambio de estado del LED desde la misma placa. El esquema de conexión.


Ahora el objetivo no es sólo la monitorización del LED, sino también el cambio de su estado tanto en local ordenador con un pulsador conectado a la NodeMCU, como con un elemento en el Dashboard de Adafruit IO.

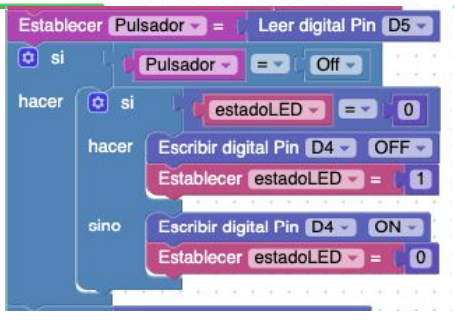
Para ello vamos a seguir el procedimiento propuesto: primero, el diseño del Dashboard que reutilizaremos del paso anterior y, después, montamos el circuito (imagen siguiente) y elaboramos el programa sobre la base del anterior en el Arduinoblocks pero cambiando la estrategia,




En este caso, emplearemos la transmisión de datos desde la placa únicamente cuando cambie el estado del LED y este puede cambiar pulsando o los botones del Dashboard o el pulsador del circuito.

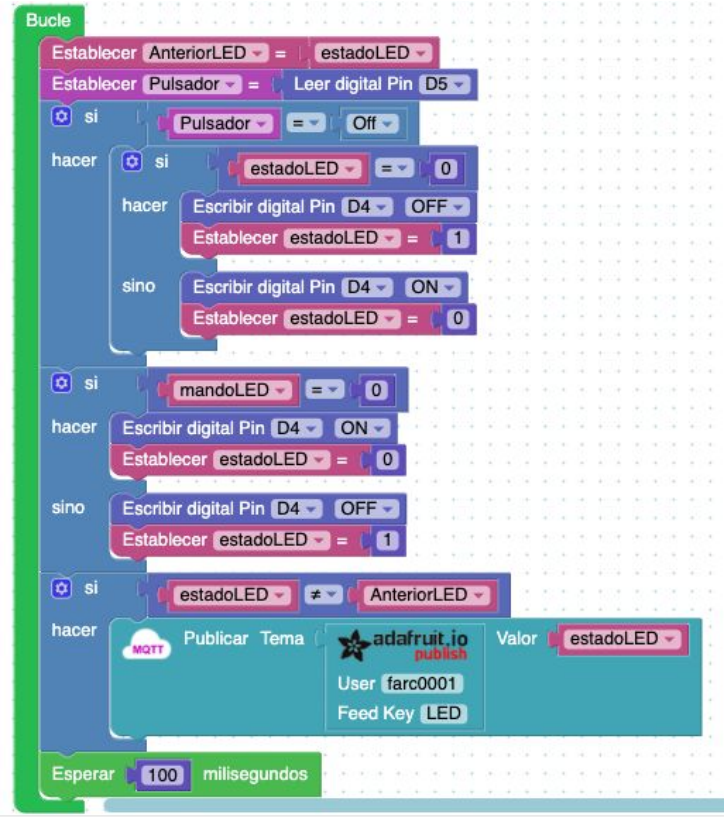
El contenido del bloque “Iniciar”, “MQTT Iniciar”, no se modifica y podemos utilizarlo tal cual lo empleamos en el paso anterior de este ejemplo.

Sin embargo, la función “Bucle” sufre modificaciones en la parte de la publicación del estado del LED. Para ello incorporamos una variable auxiliar “AnteriorLED” que guarda el estado del LED anterior, con el fin de detectar cambios de estado comparándola con la que almacena el actual “estadoLED”.

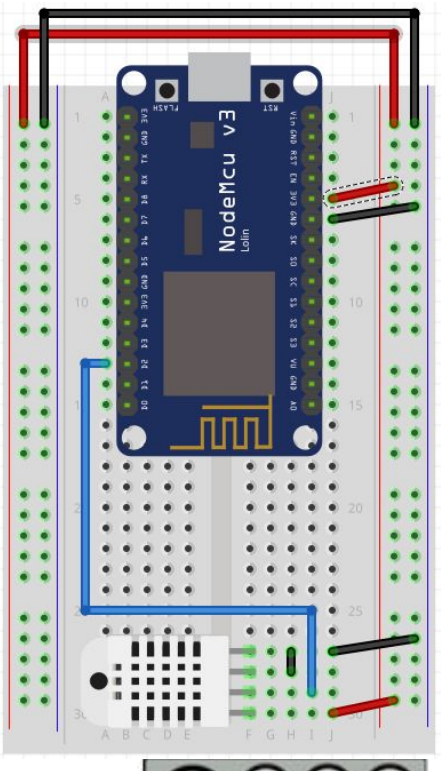

Incorporamos la detección del pulsador del circuito y su efecto sobre el LED. Lo que hace es conmutar el encendido cada vez que se pulsa.

De tal manera que la función “Bucle” completa queda como se muestra en la siguiente imagen.




PRÁCTICA_1: A partir del ejemplo anterior, realizar las modificaciones necesarias para conseguir que se monitorice el ciclo de un semáforo y le puedas controlar el estado de las luces

## 4.6 Ejemplo 2: Temperatura y humedad




Con este ejemplo construiremos un dispositivo que funcione como una miniestación meteorológica remota, recogiendo los datos que manda en la plataforma Adafruit IO. Vamos a introducir un módulo que integra dos sensores y se comunica por OneWire con el microcontrolador, el DHT11 (vale también el DHT22).

**Esquema de conexión:**

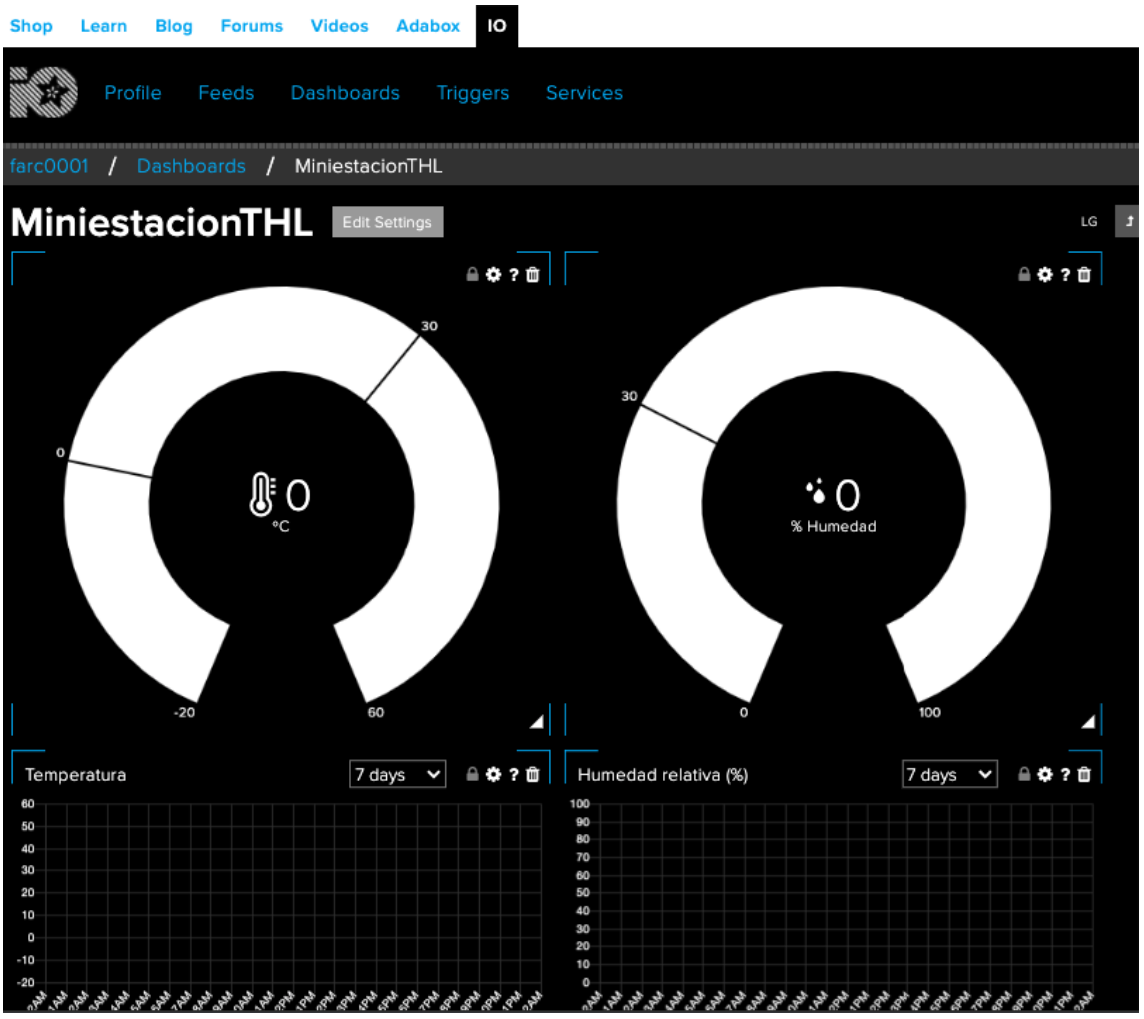
Por lo tanto, necesitaremos:

- Una placa NodeMCU
- Un módulo DHT11 o DHT22. → Pin D6
- Cables de conexión.

**Solución propuesta:**

Siguiendo el procedimiento propuesto, empezaremos diseñando el tablero o Dashboard de monitorización.

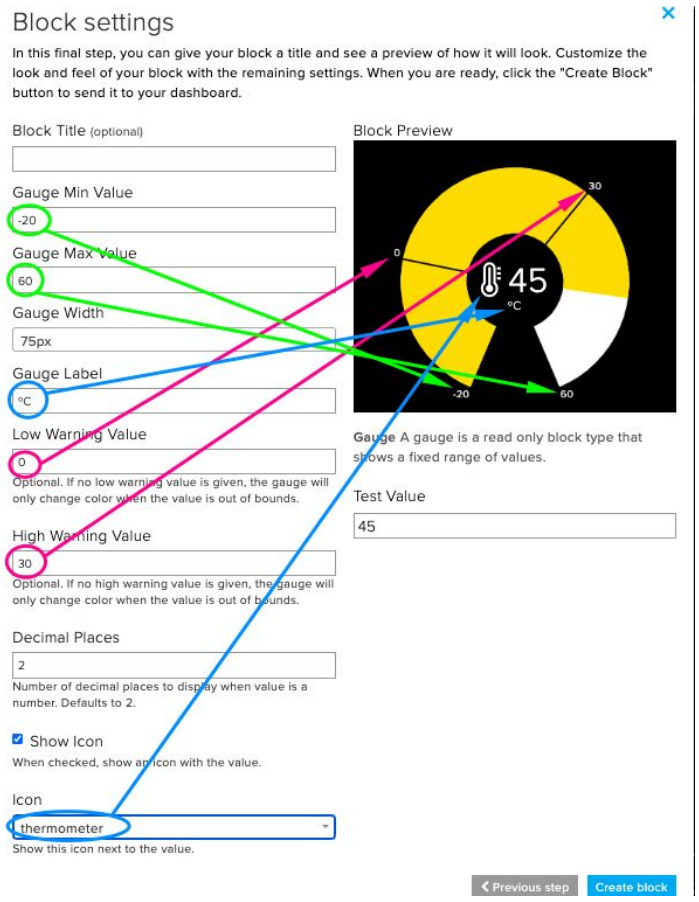
En el diseño, vamos a contemplar, no solo, el valor actualizado, sino también, el histórico. Para ello, emplearemos 2 elementos tipo reloj (gauge) y otros 2 Líneas de tiempo (linechart) con la siguiente disposición. Le hemos incorporado unos valores límite normales y un icono acorde con la magnitud a medir. Aquí sólo monitorizamos, no comandamos nada.




Empezaremos con la configuración del primer reloj indicador, el de la temperatura. Desde la pantalla principal de Dashboards→ Actions →Create a new dashboard, le ponemos un nombre, añadimos una descripción → Create.

- Entramos en Tablero creado y pulsamos sobre  seguido de  , creamos un nuevo Feed “Temperatura”, lo seleccionamos → “Next step”.
- Cambiamos los valores mínimo “-20”, y máximo “60” (color verde) del indicador que irá expresado en grados centígrados con un icono de “thermometer” (azul)
- Pondremos unos límites de alerta, mínimo en 0ºC y máximo en 30ºC (Rosa).
- Le damos un grosor al arco del indicador de 75px.

- Comprobamos con “Test Value” poniéndole varios valores si cambia de color al traspasar los límites de alerta y pulsamos sobre “Create block”




Una vez creado el indicador de temperatura, haremos lo mismo con el de humedad. En la imagen del Dashboard diseñado para este ejemplo, los parámetros para el reloj de humedad relativa son:

- Gauge Min Value → 0
- Gauge Max Value → 100
- Gauge Width → 75px
- Gauge Label → % Humedad
- Low Warning Value → 30
- Icon → w:raindrops

Ya sólo nos queda incorporar las gráficas de datos históricos. Para ello pulsamos en + y seleccionamos  , escogemos el Feed de “Temperatura” que habíamos creado antes→ “Next step”.

Nos tiene que salir una pantalla en la que tenemos que elegir: el título del bloque, el periodo de monitorización en horas o días, etiquetas para los ejes, valores mínimo y máximo del eje Y.... Una vez relleno se pulsa en “Create block”.

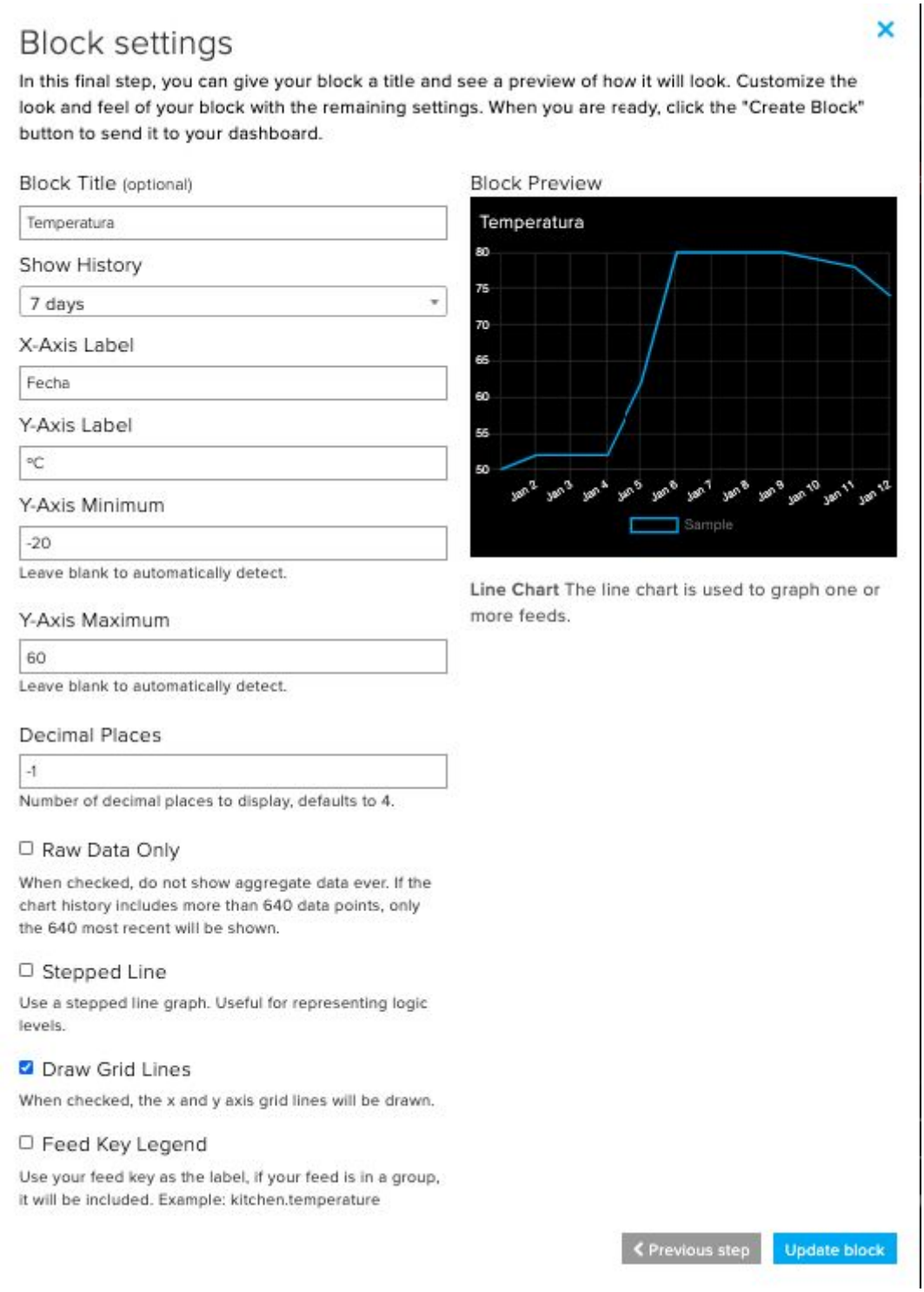
En el ejemplo los valores para el histórico de temperatura son:

- Feed → “Temperatura”
- Block Title → Temperatura
- Show History → 7 days
- X-Axis Label → Fecha
- Y-Axis Label → ºC
- Y-Axis Minimum → -20
- Y-Axis Maximum → 60
- Marcado de “Draw Grid Lines”

Para el de Humedad, son:

- Feed → “Humedad”
- Block Title → Humedad

- Show History → 7 days
- X-Axis Label → Fecha
- Y-Axis Label → %
- Y-Axis Minimum → 0
- Y-Axis Maximum →100
- Marcado de “Draw Grid Lines”




Como nos coloca un elemento debajo del anterior, ordenamos todo pulsando en  , situando con ayuda del ratón en el tamaño adecuado y la organización que más nos guste. Cuando lo tengamos, lo guardamos y ya tenemos la plataforma MQTT preparada.

La siguiente fase consiste en realizar el programa, en este caso se ha optado por la técnica del semáforo o de paso de turno que consiste en iniciar un “timer” o cronómetro y publicar el primer tema cuando llegue a cero, reiniciarlo y pasar el turno al tema siguiente de manera que

se publique, reiniciarlo otra vez y pasar el turno al siguiente feed, si lo hay, o al primero otra vez, cerrando el ciclo.

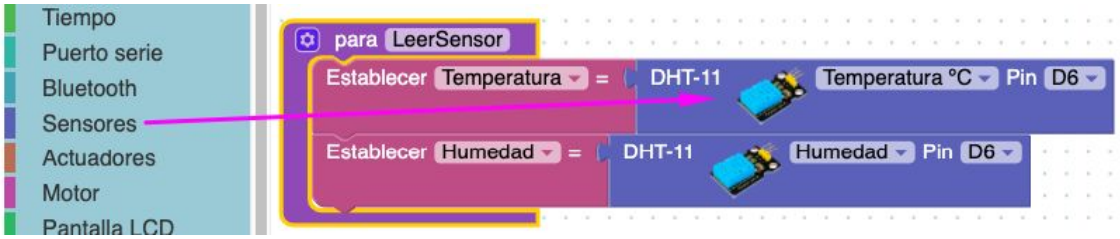
El sensor DHT ofrece valores significativos 1 vez por segundo, aproximadamente, mucho más rápido de lo necesario en circunstancias ambientales normales, donde la variación térmica importante ocurre en minutos salvo que, de repente, le den los rayos solares) y el tema de la humedad pasa algo parecido excepto en el momento que rompe a llover. Por lo tanto, este tipo de plataformas ofrece un tráfico de valores suficiente sin necesidad de saturar el plan gratuito.

Vamos a ArduinoBlocks y creamos un nuevo proyecto. En la función "Inicializar", pondremos el bloque de:

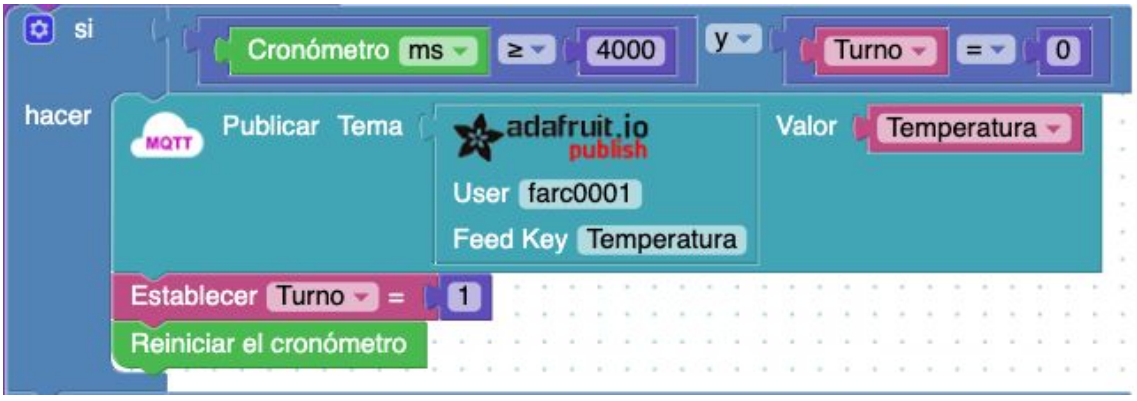
- "MQTT Iniciar" con los datos de la WiFi y de conexión a Adafruit IO, así como el nombre del cliente que es el del dispositivo.
- Inicializar la variable "Turno" a cero. Es la que da el testigo para la publicación de cada Feed. En nuestro ejemplo "0" es para la Temperatura y "1" para la Humedad relativa.
- Iniciamos el cronómetro.


Para organizar lo que sería un proyecto genérico, haremos uso de funciones, lo que hace mucho más fácil seguir y entender los programas. En nuestro ejemplo, el bloque principal "Bucle" tiene una función "LeerSensor" otra "PublicarValores" y una pequeña espera de 100ms para evitar el bloqueo del dispositivo.

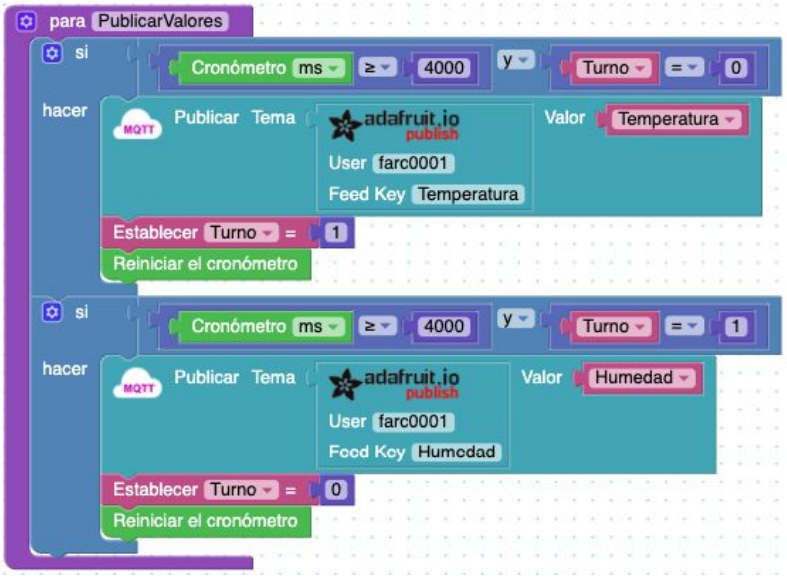
La función "LeerSensor", recoge en sendas variables los valores del DHT11(o DHT22) de Temperatura y Humedad. Si tuvieramos más sensores, los secuenciaríamos aquí.




Cada turno se ha programado cada 4 segundos (4000ms) y el proceso es: comparar si se ha agotado el tiempo y el turno es el correcto, entonces se publica el feed, se actualiza la variable del turno y se reinicia el cronómetro. Esto se realiza de manera sistemática, tengas 2 o 10 publicaciones de temas distintos.




Quedando la función “PublicarValores” completa




**NOTA:** si no te funciona el ABConnector en el ordenador o compilas desde el IDE de Arduino, debes descargarte las librerías que empla ArduinoBlocks desde la opción de Recursos → Librerías Arduino. Después, tendrás que descomprimir el “zip” para, posteriormente, comprimir la carpeta “ABlocks_DHT” en “zip” e incorporarla al IDE de Arduino.

PRÁCTICA_2: A partir del ejemplo anterior, realizar las modificaciones necesarias para conseguir que se monitorice cuando ha empezado a llover o cuando le empieza a dar el sol al sensor. Crea nuevos Feed “haSalidoSol” y “Llueve”. Comprueba la diferencia entre los valores anteriormente publicados y los que tengas que publicar y, si la diferencia es superior a 2ºC o aumenta considerablemente la humedad, que mande la señal al Broker.

Puedes emplear el método del semáforo sólo (sistemático) o añadirle el del cambio de valor (optimizado).

## 4.7 Bibliografía

[1]https://www.espressif.com/en/home

[2] https://programarfacil.com/podcast/nodemcu-tutorial-paso-a-paso/

[3]https://didactronica.com/arduino-desde-cero-arduinoblocks/