

# Guía – Interruptor con comando por voz desde Google Assistant o Alexa

Por: Camilo Andrés Segura Quintero

Desarrollo de Aplicaciones para Sistemas Ubicuos

Programa Ingeniería Electrónica y Telecomunicaciones

Universidad del Cauca

---

## Contenido

Introducción .....	1
Materiales .....	2
Diseño .....	2
Desarrollo .....	3
Conclusiones .....	7
Extras .....	8

## Introducción

Los sistemas de mando a distancia han representado una extensión a la capacidad humana con un valor elevado en ahorro de tiempo, comodidad y agilidad de procesos. Es así, como sistemas tan básicos como el control remoto, cambiaron la historia y rápidamente fueron adoptados como interfaces humano-máquina tan aceptados que ahora hacen parte de la vida cotidiana. En particular, la modernidad y el avance de la tecnología han permitido la creación de nuevas técnicas y mecanismos para comunicar comandos y controlar dispositivos de forma remota. Uno de estos mecanismos es el comando de voz.

El comando de voz es particularmente funcional para escenarios dónde un dispositivo interruptor tradicional o “apagador” no es accesible o resulta en un esfuerzo mayor para la persona que requiere el apagado o encendido de un dispositivo. Son casos tales como el de personas con discapacidad motriz, personas de viaje o individuos con alta carga laboral, que requieren el control sobre algún aparato a causa de sus limitaciones, seguridad de sus pertenencias o simple comodidad. El valor agregado parece no tener límites, según lo que denominamos como carga o dispositivo conectado al final del interruptor.

En esta guía se presenta un dispositivo interruptor comandado por voz, basado en el microcontrolador ESP32. El sistema consume múltiples servicios web de IFTTT, Adafruit, Amazon y Google, a través de diversas librerías e interfaces web que permiten el ensamble de la lógica en línea. El comando por voz es entregado a Google Assistant o a Amazon Alexa, según facilidad del usuario o del desarrollador para proponer soluciones más complejas.

Enlaces de interés para la práctica:

Video: <https://youtu.be/CBme-MJ9aC8>

Código: <https://github.com/Kseg97/RelaySwitchGoogleAssistant>

## Materiales

- 1x ESP32 (DOIT Esp32 Devkit v1 u otro) con cable
- 1x Resistencia 220 1/4W
- 1x Transistor NPN (BC548 o 2N2222)
- 1x LED 5mm
- 1x Relay 5VDC 5 pines
- Jumpers
- (opc.) Cable tomacorriente macho
- (opc.) Tomacorriente hembra
- Cuenta de google
- Smartphone con Google Assistant o Google Home
- Arduino IDE 1.8.1 o superior

## Diseño

El diseño del sistema HW corresponde a un montaje de circuito *relay*. Considérese usar los elementos opcionales de la lista de materiales para un acabado final. En la figura 1 se presenta el microcontrolador ESP32 (izquierda) y el circuito relé (derecha).

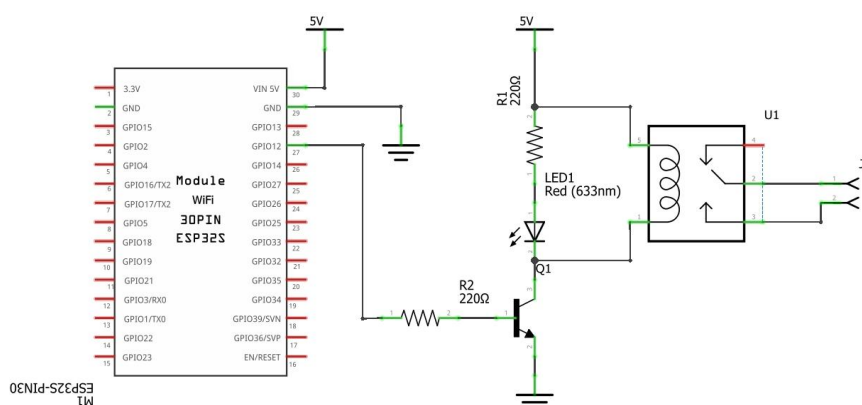


Figura 1. Diagrama esquemático de la electrónica.

La arquitectura de sistema es presentada en la figura 2. En la parte superior se encuentra los componentes de externos usados a nivel de servicios. IFTTT es el *middleware* clave que ensambla a Adafruit IO con Google Assistant. La práctica puede dividirse en dos secciones de desarrollo que serán diferenciadas en su momento. La primera parte consiste en comunicar la ESP32 con Adafruit IO (MQTT Broker) y permitir el encendido y apagado del relé remotamente a través de este protocolo e internet. Esto ya ha sido probado en otras prácticas del curso, pero se incluirá en esta guía para el público general. La segunda parte consiste en configurar IFTTT, Google Assistant y Alexa para tener la funcionalidad final.

La parte inferior de la arquitectura es dedicada a la ESP32 como un subsistema de control. Dentro de este subsistema, las librerías de WiFi (incluida por defecto en la instalación del microcontrolador en Arduino) y la lógica digital (cuya salida es un simple digitalWrite) son elementos de capa física. La librería Adafruit MQTT opera a nivel físico y de red, tomando ventajas embebidas en el microcontrolador.

La entrada del sistema es el comando de voz por las aplicaciones de Google Assistant o Home y Alexa, así como la alimentación del circuito que debe incluir la red eléctrica 120V o 220V y una alimentación de 5V para la tarjeta de desarrollo. La salida es a un tomacorriente cuya carga puede ser cualquier dispositivo de potencia soportado por la red eléctrica.

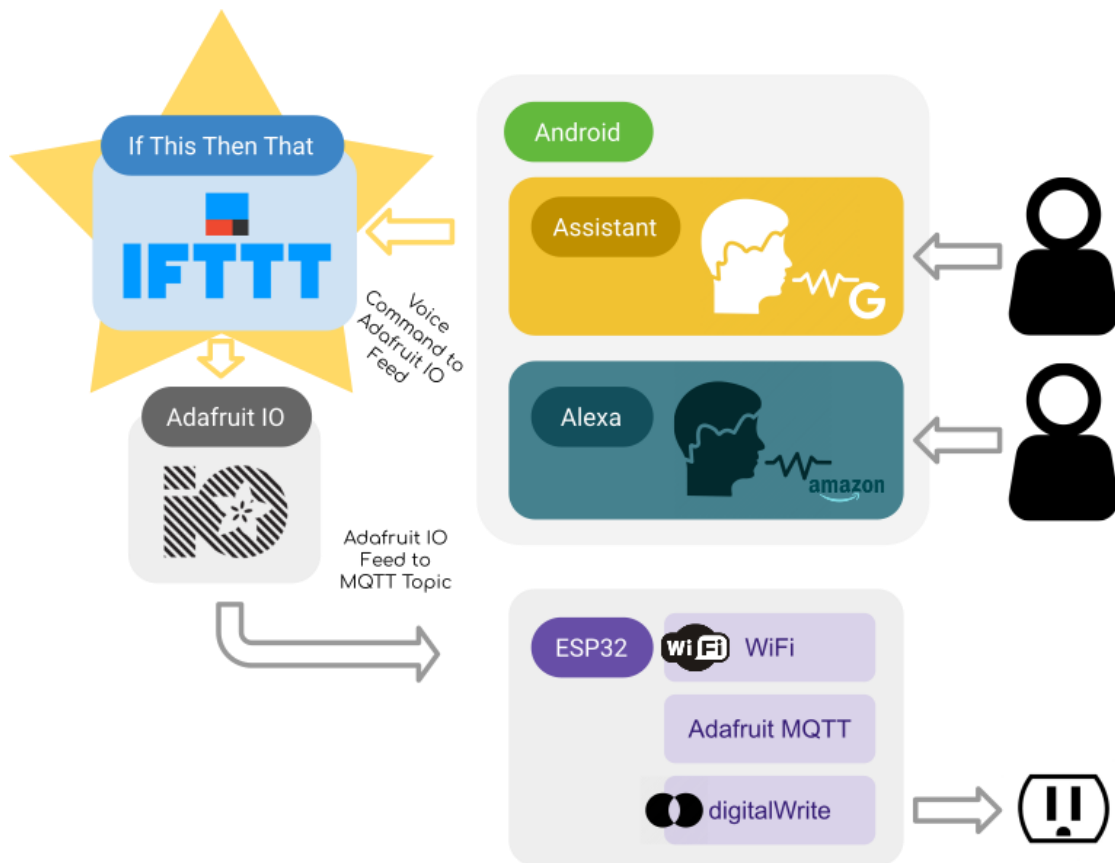


Figura 2. Diagrama arquitectónico de la solución.

## Desarrollo

### Parte I. Conexión MQTT.

1. Instalación de controladores y soporte para ESP32: <https://github.com/espressif/arduino-esp32#installation-instructions>. Se recomienda usar el instalador de tarjetas (parte superior: Herramientas >> Placa >> Gestor de tarjetas). En caso de no funcionar, seguir las instrucciones de la sección de instalación para el sistema operativo en que se instaló Arduino.
2. Instalación de la librería Adafruit MQTT (Programa >> Incluir Librería >> Administrar Bibliotecas). Otras librería MQTT Client que acepten usuario y contraseña podrían funcionar.

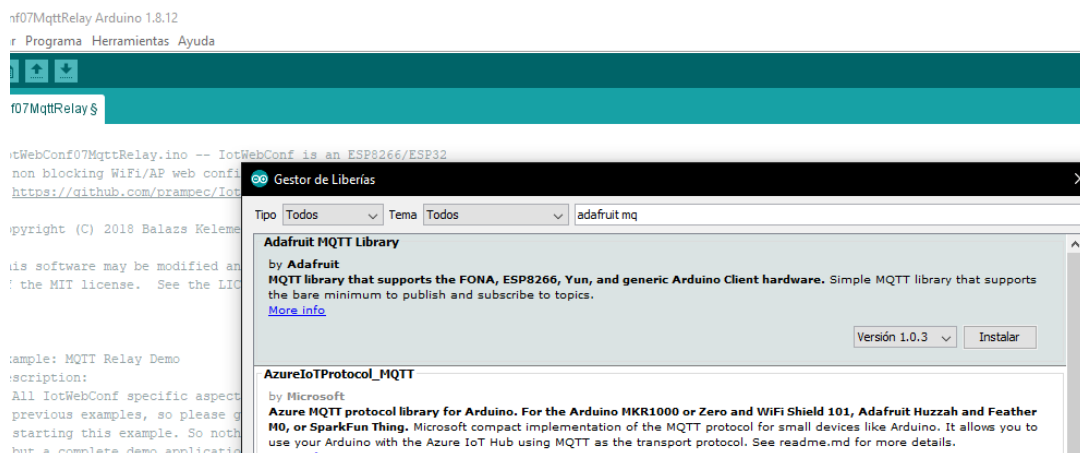


Figura 3. Administrar bibliotecas para la instalación de librerías.

3. Cree una cuenta en Adafruit IO. Adafruit hará preguntas sobre la compañía y otros detalles, estos campos pueden ser llenados con cualquier información, si lo ve pertinente. Cuando haya acabado, aparecerá el Dashboard.

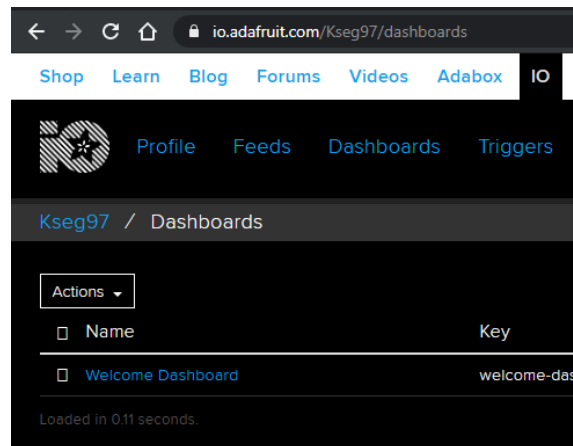


Figura 4. Dashboard de Adafruit IO (AIO).

4. Cree un nuevo Dashboard (en actions) con el nombre que desee, se recomienda "Esp32\_google\_assistant".
5. Cree un nuevo *feed* (arriba aparece Feeds, presione en él y diríjase a sus *feeds*). Use el nombre "onoff". Este feed será usado para conectar con IFTTT (ver figura 2).

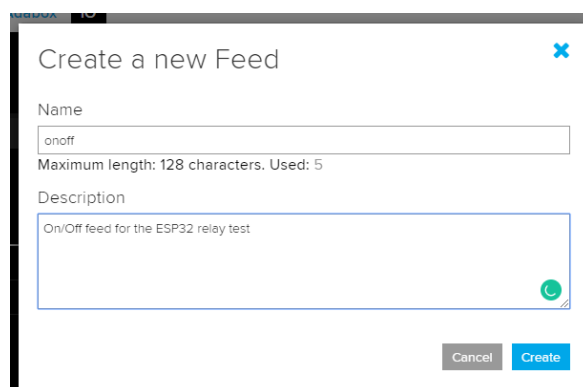


Figura 5. Dashboard de Adafruit IO (AIO).

6. Dentro del Dashboard creado, cree un nuevo bloque.



Figura 6. Dashboard de Adafruit IO (AIO).

7. Elija un *toggle* con el feed *onoff* creado. Presione en crear el bloque.
8. Obtenga las credenciales en AIO KEY (esquina superior derecha).
9. Descargar, clonar o copiar el código de:  
<https://github.com/Kseg97/RelaySwitchGoogleAssistant>.
10. Sustituir el SSID y contraseña por los de la red de preferencia.
11. Sustituir el usuario y la llave AIO por los provistos en la cuenta creada de Adafruit IO.
12. Montar el circuito de la figura x, basado en el diagrama esquemático de la figura1. La conexiones del *relay* pueden variar según el modelo, se recomienda realizar pruebas sin el relé (solo el LED) y verificar posteriormente el *datasheet* del *relay* para establecer su correcta conexión.

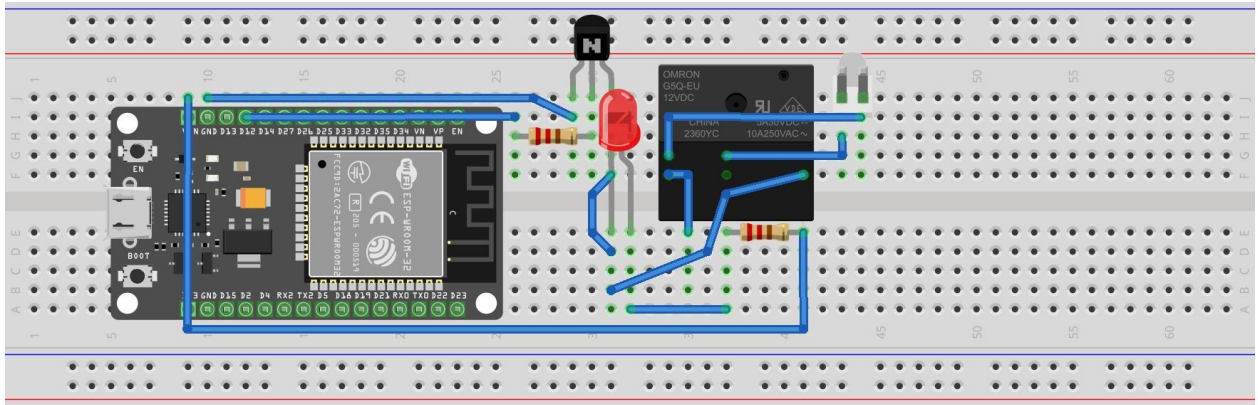


Figura 7. Montaje protoboard.

13. Verificar conexiones y pertinencia del código con las credenciales de cada plataforma.
14. Compilar y subir el sketch.
15. En el Dashboard de Adafruit IO, se debe ser capaz de mover el toggle relacionado a nuestro dispositivo. Cambio de estado.
16. Felicidades, puede controlar un relay a través de MQTT desde cualquier parte del mundo con acceso a internet.

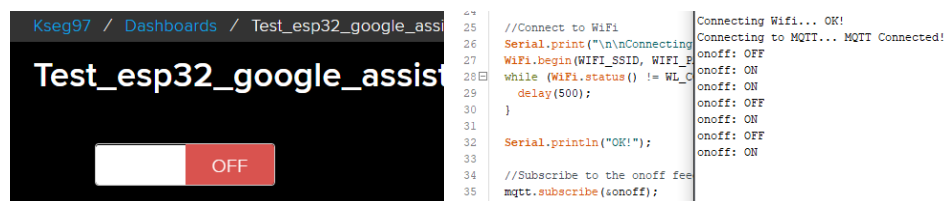
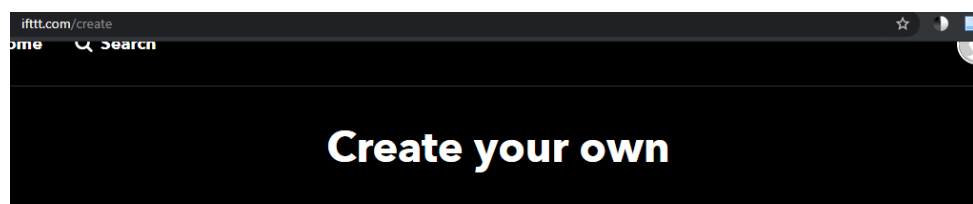


Figura 8. Toggle (derecha) y prueba del interruptor (izquierda).

## Parte II. Conexión con servicios avanzados.

17. Crearemos una cuenta de IFTTT. Al finalizar de crearla, nos pedirá el nombre para un servicio, podemos llenar este campo a gusto.
18. Creamos un applet con el creador <https://ifttt.com/create>



# If + This Then That

Build your own service on the **IFTTT** Platform

Figura 9. If This Then That – Creador.

Para Google Assistant.

19. Presionamos en *this*. Buscamos “Alexa”.
20. Presionamos sobre “Say a specific phrase”.
21. Escribimos una frase, como “aparatos on”, para que encienda el dispositivo. Permite ingresar varias frases.




Figura 10. Configuración de frases de Google desde IFTTT.

22. Buscamos por “Adafruit”, seleccionamos y elegimos “Send data to Adafruit IO”.
23. Elegimos el feed *onoff*.

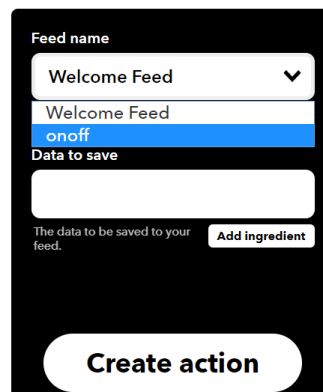


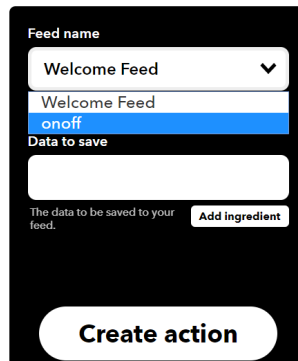
Figura 11. Configuración del feed en AIO desde IFTTT.

24. El dato a guardar es “ON”, para activar el interruptor.
25. Finalizamos el applet y conectamos las cuentas al activar el mismo. Es necesario que la cuenta de Google que se conecte a IFTTT sea la misma del *smartphone*.
26. Repite los pasos de creación pero ahora para el apagado. Usa el dato a guardar “OFF”.
27. Verificar que estén habilitados los applets de IFTTT.
28. Enciende el circuito de nuevo y repite el paso 15 para verificar que esté conectado a Adafruit.
29. “Ok, google”. Al *smartphone*.
30. “Aparatus on” y se encenderá el dispositivo.
31. Felicidades, tienes comando a voz de un interruptor.

Para Amazon Alexa.

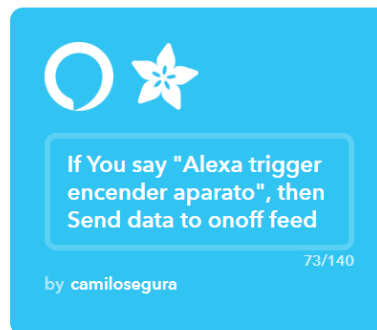
32. Crea una cuenta de Amazon y descarga Alexa en tu celular.
33. Ejecuta el paso 18 para ir al creador de IFTTT.
34. Presionamos en *this*. Buscamos “Alexa”.
35. Presionamos sobre “Say a specific phrase”.
36. Escribimos la frase, como “encender aparato”.
37. Continuamos y seleccionamos *that*.

38. Buscamos por “Adafruit”, seleccionamos y elegimos “Send data to Adafruit IO”.
39. Elegimos el feed *onoff*.



**Figura 12.** Configuración del feed de AIO, es igual en Alexa y en Google.

40. El dato a guardar es “ON”, para activar el interruptor.
41. Finaliza y activa el applet, te pedirá cuenta de una cuenta de Amazon.



**Figura 13.** Finalización de la configuración de Alexa.

42. Repite los pasos de creación pero ahora para el apagado. Usa el dato a guardar “OFF”.
43. Verificar que estén habilitados los applets de IFTTT.
44. Enciende el circuito de nuevo y repite el paso 15 para verificar que esté conectado a Adafruit.
45. Para Alexa debes descargar la aplicación e iniciar sesión con tu cuenta de Amazon.
46. Abre alexa y presiona sobre el asistente (icono del centro).
47. Di “Alexa trigger encender aparato”. Se encenderá el interruptor.
48. Felicidades, tienes Alexa (no solo la app de android sino cualquier dispositivo con el asistente) para controlar el interruptor.

## Conclusiones

El sistema presentado permite conocer el funcionamiento y articulación de un *middleware* para la incorporación de funcionalidades avanzadas y desacopladas de las capas inferiores. Cada subsistema es un bloque con distintos protocolos y mecanismos de comunicación que permiten construir una solución de valor funcional y amplio espectro de posibles aplicaciones o mejoras. Nuestro dispositivo permitirá el comando por voz a través de las plataformas mencionadas y podrá servir para el encendido y apagado de múltiples dispositivos eléctricos, independiente de su naturaleza.

## Extras

Hemos incluido un diseño PCB para la construcción y producción de este circuito. El archivo fritzing con los diagramas y la pcb estará en el repositorio de este proyecto.

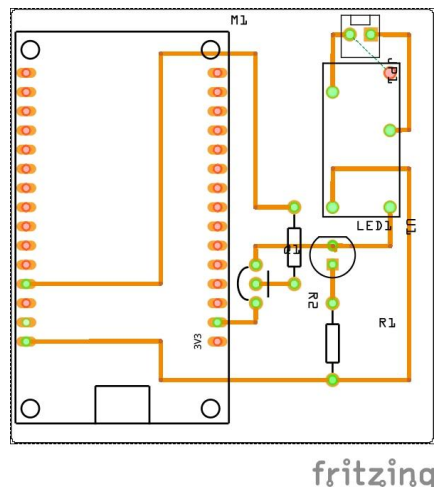


Figura 10. PCB.

Posibles mejoras pueden tomar las siguientes consideraciones:

- Acceso a la red. En la mayoría de casos el usuario va a disponer de su propia red y no tendrá conocimientos en programación, así que necesitará de una interfaz para configurar su red en el dispositivo, y que este se pueda conectar a internet para funcionar. Afortunadamente tanto la ESP32 como la ESP8266 disponen de modo AP, lo que les permite crear una red WiFi Hotspot. Existen librerías como WifiManager, IoTWebConfig (recomendada) y Autoconnect que gestionan el cambio de AP a WiFi normal y permiten la creación de menús para guardar datos en la EEPROM que configurarán el dispositivo en el siguiente dispositivo, de forma automática. Se recomienda usarlas en productos más complejos.
- Coste y complejidad de la tarjeta. La ESP32 tiene un costo muy elevado en comparación con la NodeMCU (ESP8266) o la ESP8266-01. El proyecto puede ser rediseñado fácilmente (cambiando la librería WiFi.h por ESP8266Wifi.h) entre otros detalles, para estas tarjetas. La ESP8266-01 es una opción tentativa por tener unos pocos GPIO y ser de muy bajo costo, pero su uso (instalación, conexión al computador) son más complicados y requieren pruebas y experiencia previas.