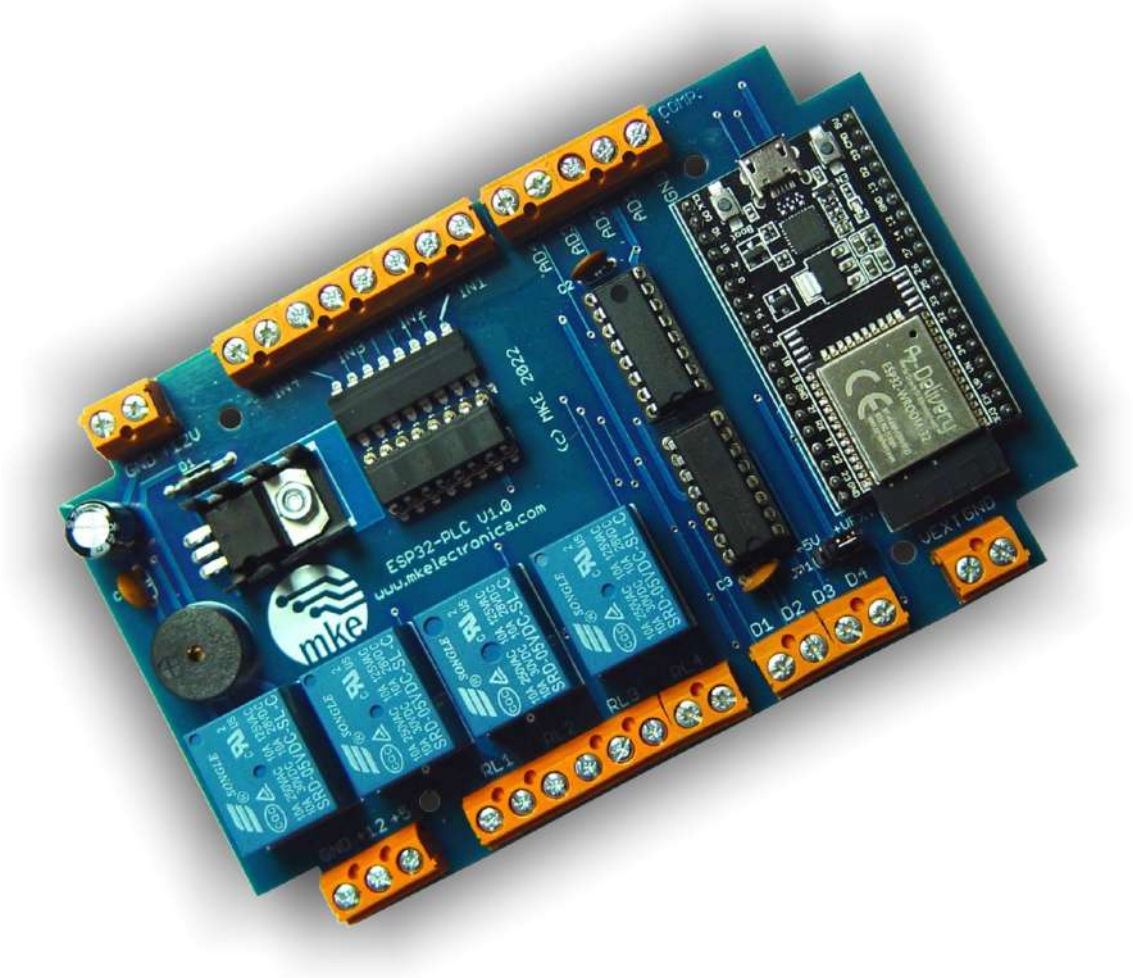
ESP32-PLC

Manual de usuario



Rev. Abril 2022

MK Electrónica

Tfno. 34-944 04 80 89

Email: info@mkelectronica.com

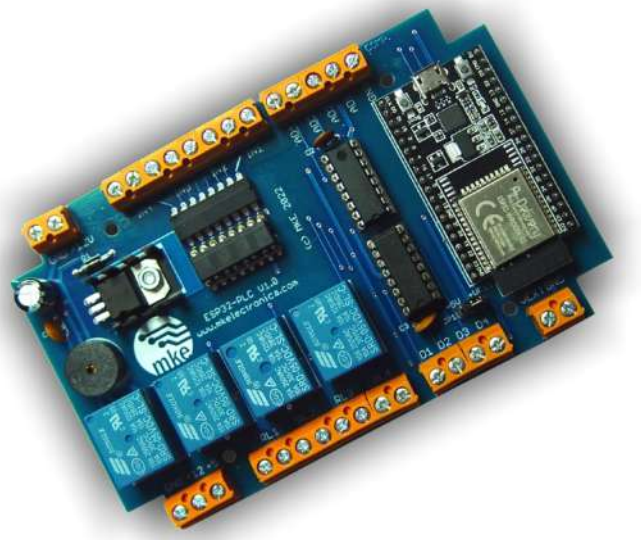
www.mkelectronica.com

ESP32-PLC: Manual de usuario

1.- INTRODUCCIÓN

En múltiples ocasiones y proyectos nos encontramos con la necesidad de controlar periféricos con unos requerimientos eléctricos que ningún micro controlador es capaz de proporcionar directamente.

En el mundo del Internet de las Cosas (IoT) esto es más acusado. Piensa en la posibilidad de controlar actuadores tales como sistemas de refrigeración, calefacción, electroválvulas, motores, contactores, etc… Por otra parte no siempre, ni todos los sensores, nos producen señales digitales con niveles de 3.3V o de 5V. A veces pueden ser señales con niveles de diferentes valores de tensión o de polaridad. Incluso podemos encontrarnos con sensores que proporcionan señales alternas.



En otras palabras, la mayor parte de las veces nos encontramos con la necesidad de acompañar al micro controlador con una cierta electrónica que permita adaptar niveles lógicos y generar señales amplificadas en tensión y/o en corriente. Esta es la razón de nuestro sencillo y económico Controlador Lógico Programable ESP32-PLC, basado en el micro controlador ESP32.

Existen diferentes herramientas, lenguajes y plataformas para su programación, pero lo mejor de todo es que podemos emplear el mismo entorno de trabajo o IDE de Arduino, y "*casi*" el mismo lenguaje. Al fin y al cabo se trata de un lenguaje C++ estándar ANSI. No se puede decir que los programas y librerías hechos para Arduino sean compatibles al 100% con el controlador ESP32 de nuestro PLC. En ocasiones tendrás que hacer sencillas modificaciones o buscar las librerías apropiadas para el ESP32, pero en la red puedes encontrar abundante información. No obstante te vamos a proporcionar una buena colección de ejemplos que esperamos te sean de gran utilidad y despejen tus dudas.

En resumidas cuentas, nuestro Controlador Lógico Programable ESP32-PLC, te ofrece un controlador de altas prestaciones pero trabajando en un entorno tan conocido y amigable como es el IDE de Arduino. ¡¡ Seguro que te sientes muy cómodo con él !!

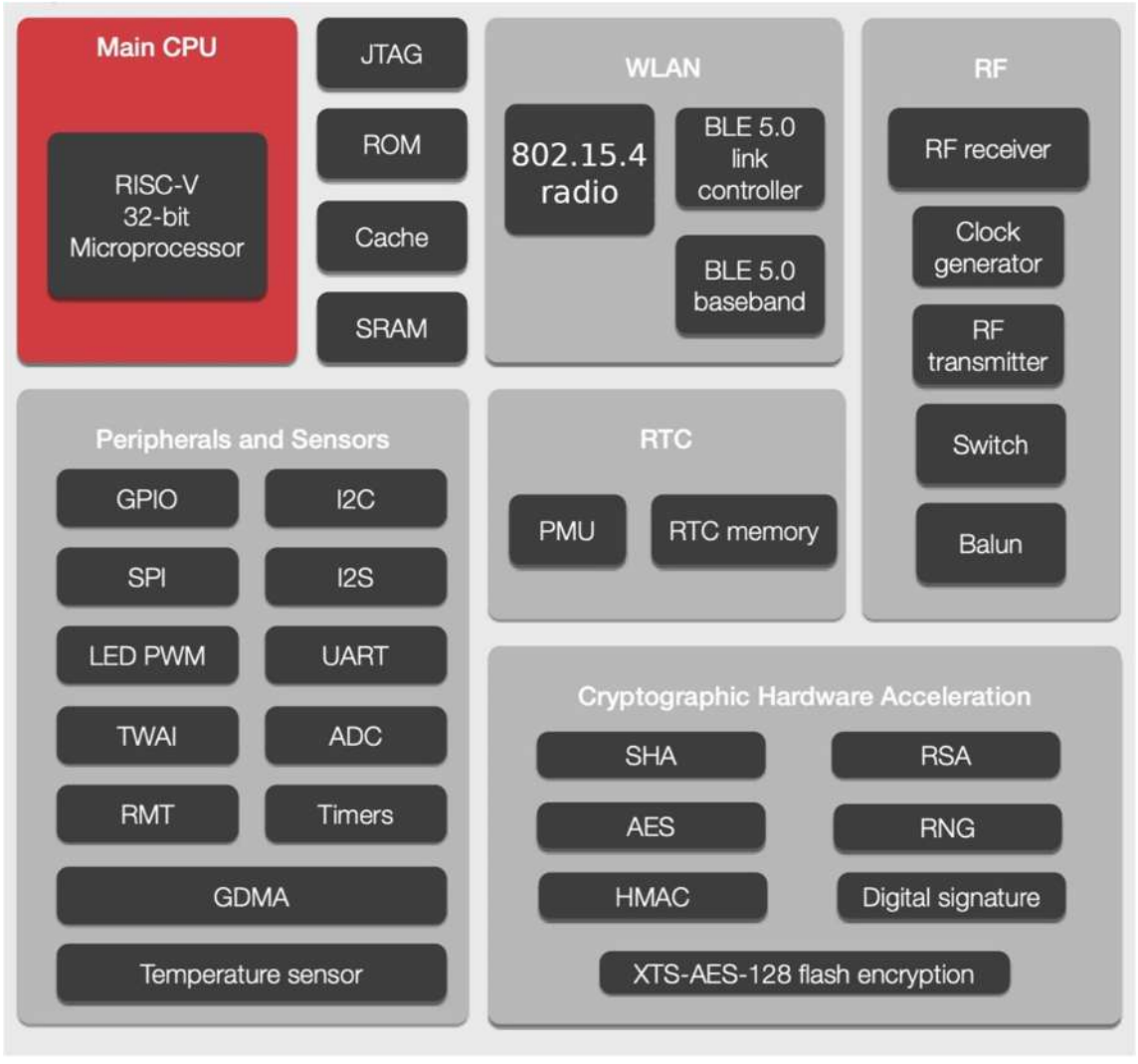
2.- ¿EN QUÉ CONSISTE?

Para explicar la razón de ser de ESP32-PLC debemos empezar por el principio y hablar de los tres componentes fundamentales que lo forman.

2-1 El SoC ESP32

Estamos hablando del microcontrolador ESP32 desarrollado por la firma China Espressif System, Puedes visitarla en www.espressif.com , y descargar todo tipo de información técnica, aplicaciones, ejemplos, herramientas de desarrollo, etc…

Consiste en un sistema completo en un único chip (**S**ystem **o**n **C**hip – **SoC**) con una CPU tipo RISC de doble núcleo Xtensa LX6 de 32 bits. Esta es su arquitectura interna…



El chip ESP32 integra, además de la CPU, la RAM, los circuitos de RF para comunicaciones Bluetooth y WiFi, circuitos para la encriptación, diferentes interfaces y periféricos como ADC, PWM, UART SPI, eI2C, etc., así como múltiples señales de E/S conocidas como GPIO's. Su fabricante propone diferentes versiones de este chip.

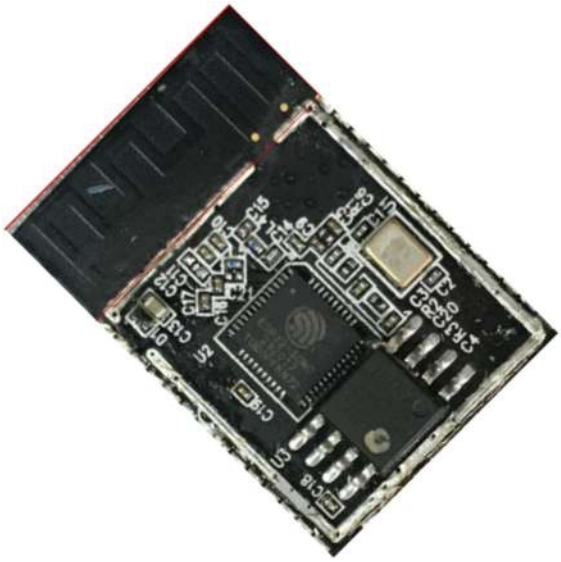
2-2 Un módulo

Una característica de los SoC ESP32 es que **NO** integran la memoria donde almacenamos nuestros programas. Se debe conectar externamente. También se le debe conectar el cristal de cuarzo del oscilador y la antena para las comunicaciones WiFi y Bluetooth.

A partir de aquí aparecen diferentes fabricantes que, usando el SoC ESP32 y heredando por tanto todas sus características, producen diferentes tipos de módulos con diferentes velocidades y tamaños de memoria.

Por ejemplo, uno de estos módulos, aunque hay infinidad de ellos y son más o menos compatibles entre sí, es el ESP32-WROOM-32 fabricado también por Espressif. Además de las que hereda del propio ESP32, añade lo siguiente:

- Antena impresa en el propio módulo.
- Oscilador para una velocidad de trabajo de 240 MHz.
- Memoria Flash para nuestros programas de usuario de 4MB.
- Acceso a todas las señales del SoC ESP32. Son 32 patillas de E/S o GPIO's.



En la imagen puedes ver uno de estos módulos. Bajo la carcasa metálica o blindaje se encuentra el SoC ESP32, el oscilador y la memoria Flash para los programas del usuario. También se puede apreciar la antena de comunicaciones WiFi y Bluetooth impresa en el propio módulo, así como algunos componentes auxiliares.

2-3 Device kit

También conocido como "*kit de desarrollo*". Trabajar directamente con uno de los módulos comentados anteriormente presenta varios inconvenientes:

- No disponen de una conexión directa con el PC como puede ser un interface USB a través del cual podamos descargar nuestros programas.
- Trabajan a +3.3V y no disponen de circuitos de regulación/estabilización de tensión. Es por ello que no se pueden alimentar directamente, por ejemplo, desde un puerto USB que suministra +5V.
- Dado el reducido tamaño de los módulos, el acceso a las patillas de E/S o GPIO's es muy complicado. Hay que ser muy hábil para poder hacer las conexiones con los sensores y actuadores que queremos controlar.

La solución se presenta en forma de "*Device kit*", que no es ni más ni menos que una tarjeta electrónica que resuelve esos inconvenientes. Dispone de un interface que permite conectar el módulo con un PC a través de un puerto USB, y un regulador/estabilizador que permite alimentar al conjunto con tensiones más normalizadas como pueden ser los +5V. También adapta el acceso a las patillas de E/S o GPIO's para facilitar las conexiones con los periféricos.

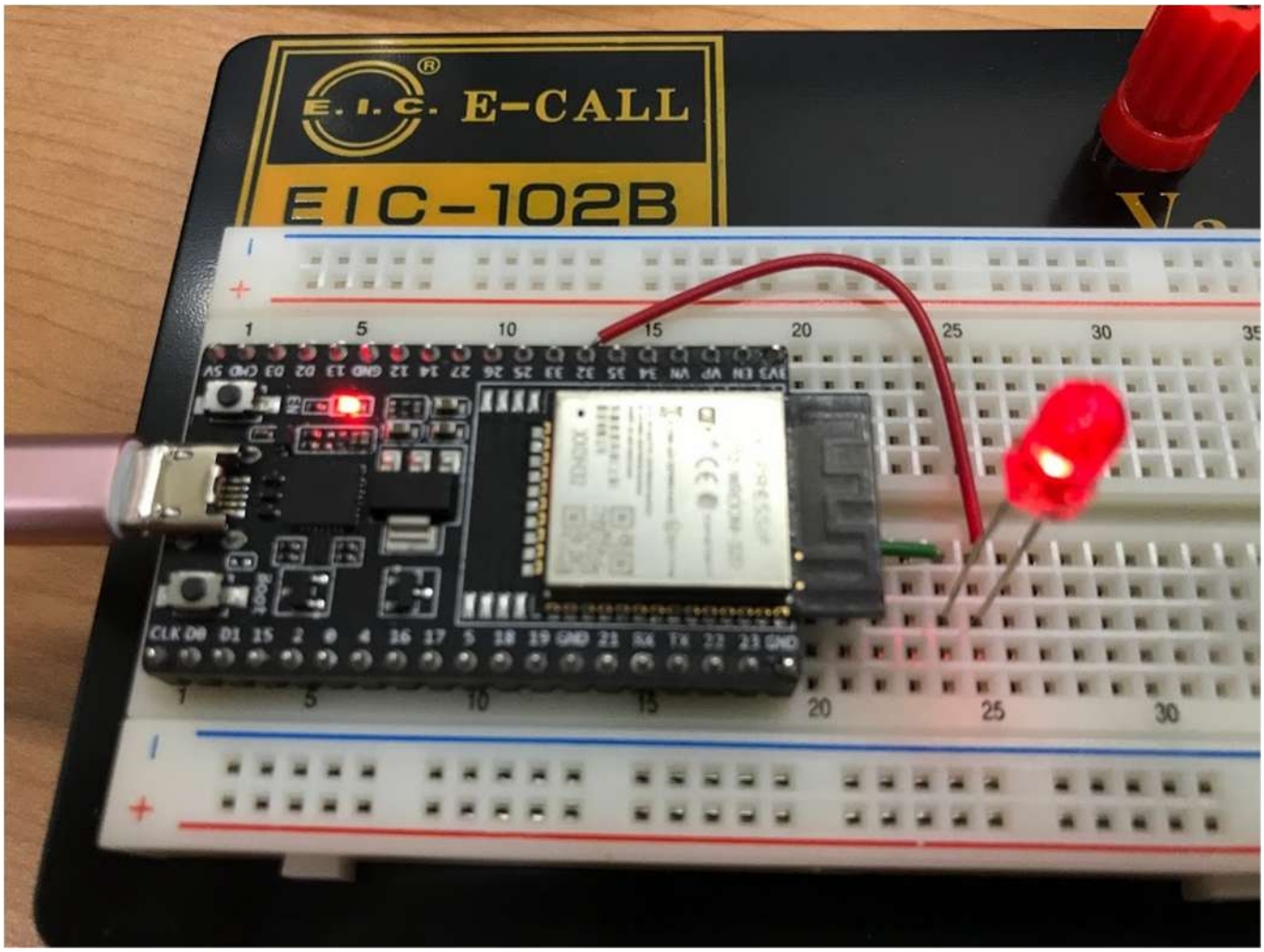
Aquí también nos podemos encontrar con multitud de fabricantes. La ventaja es que hay un amplio comercio de este tipo de tarjetas de desarrollo, pero no todos son compatibles entre sí al 100%. Puede haber pequeñas diferencias y hay que estar atentos.

Una de las tarjetas de desarrollo más conocidas es la ESP32-DevKitC de la imagen. Originalmente fabricada también por Espressif, hoy en día tiene infinidad de "*clones*" producidos por otros tantos fabricantes.

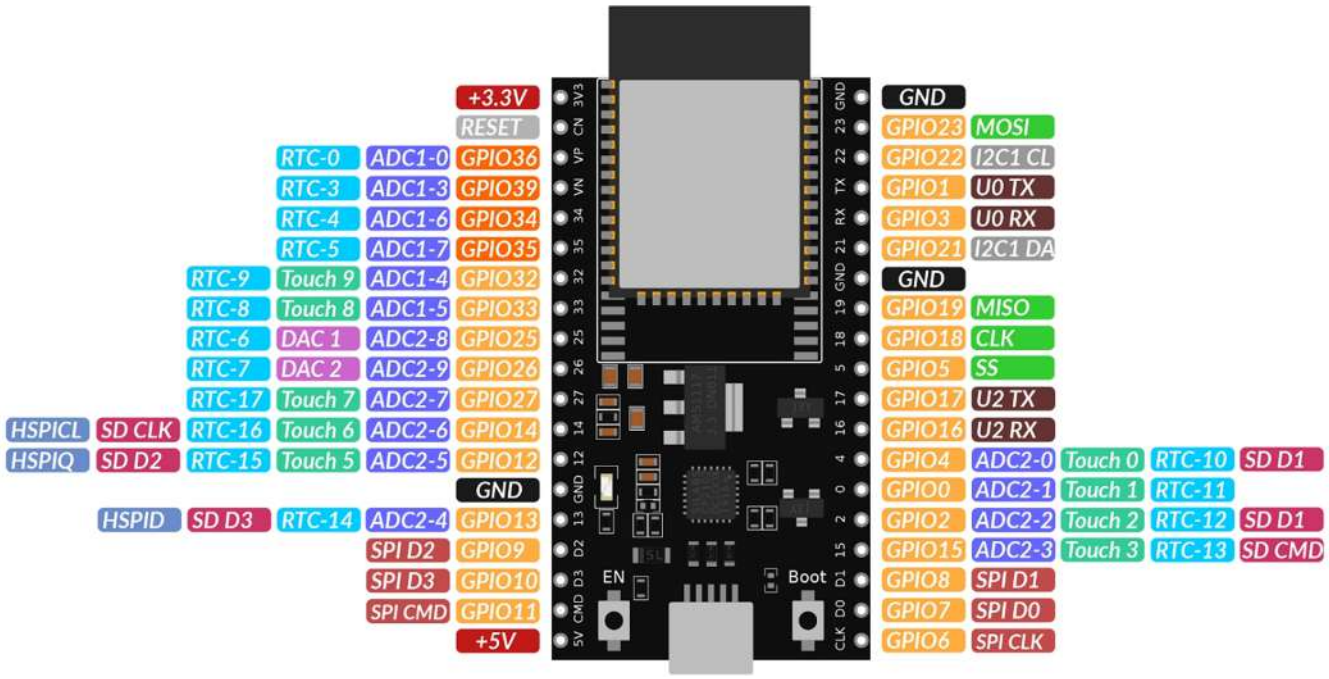
El ESP32-DevKitC incorpora, y hereda por tanto, todas las características del módulo ESP32-WROOM-32 que a su vez hereda las del SoC ESP32. Se programa desde un puerto USB y admite una tensión de alimentación de +5V, a partir de los cuales se obtienen los +3.3V que necesita el controlador ESP32.

Esta tarjeta de desarrollo también incluye dos pulsadores. Uno de ellos es el pulsador **EN** que origina el reinicio del sistema. Equivale al RESET. El otro pulsador se le conoce como "**BOOT**" o también como "**PGM**". En ocasiones es necesario activarlo para que la tarjeta entre en el modo de programación cada vez que intentemos descargar un nuevo programa sobre ella.

Esta tarjeta de desarrollo o Device Kit también facilita la conexión con las señales de E/S o GPIO's del ESP32. Tiene las dimensiones apropiadas como para poder insertarla sobre un módulo protoboard, lo que te permite realizar las conexiones de todo tipo de experimentos y montajes de prototipos de forma rápida y sencilla.



Llegados a este punto vamos a mostrar la distribución de las señales de E/S o GPIO's en la tarjeta de desarrollo ESP32- DevKitC, que son las mismas que las disponibles en el módulo ESP32-WROOM-32 (o equivalente) y que a su vez son las que proporciona directamente el SoC ESP32. Mira la figura…



Como ocurre con todos los controladores modernos, la mayor parte de esas señales pueden tener diferentes funciones dependiendo de cómo se configuren. Se resumen en la siguiente tabla…

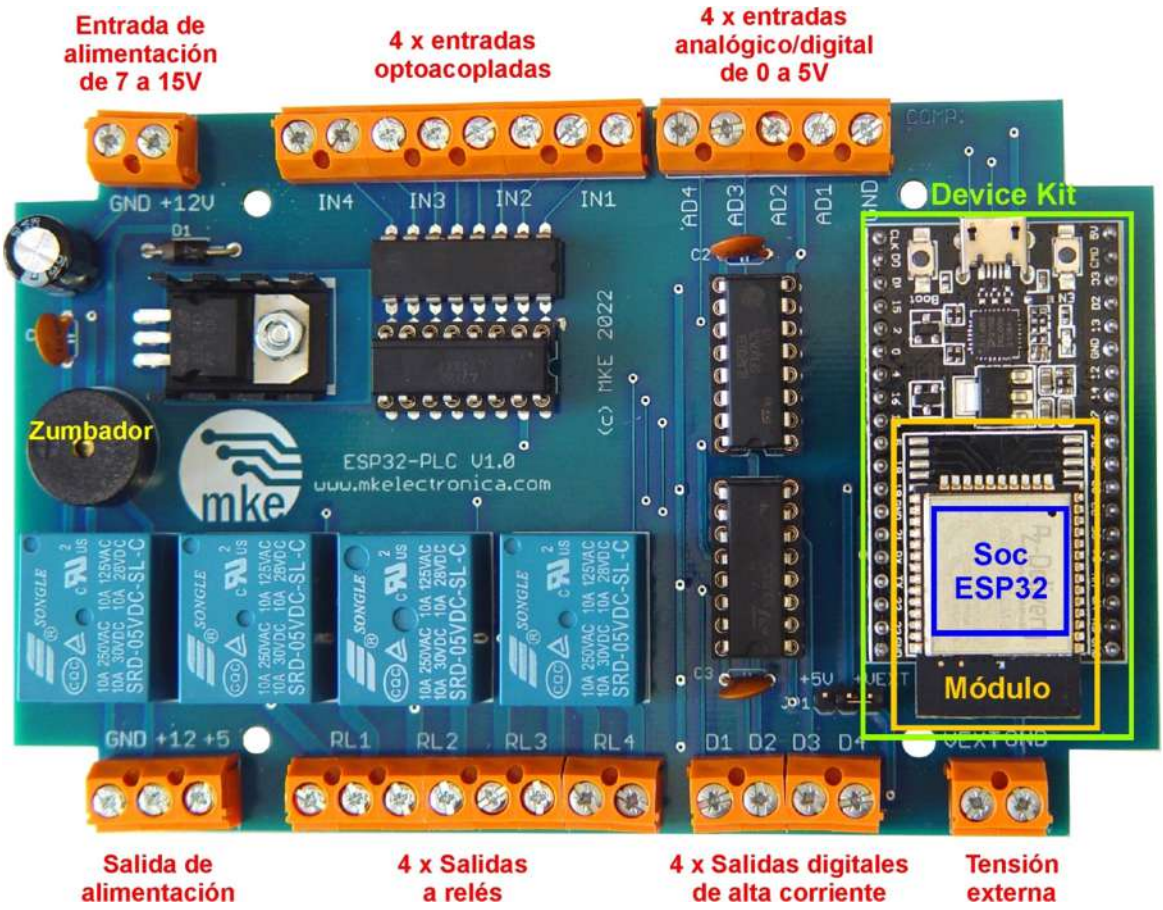
COLOR	NOMBRE	DESCRIPCIÓN
	GND	Tierra de alimentación
	+3.3V - +5V	Tensiones positivas de alimentación
	GPIO0-GPI033	E/S digitales de propósito general. Todas puede actuar como salidas PWM
	GPIO34-GPIO36	Sólo puedes actuar como entradas digitales.
	ADCX-Y	Entradas analógicas del ADC, de 0 a 3.3 V y con 12 bits de resolución
	DAC1-DAC2	Salidas analógicas del DAC, de 0 a 3.3V y con 8 bits de resolución
	Touch 0-Touch 9	Entradas con sensor capacitivo
	RTC0 – RTC17	Señales de E/S en modo de bajo consumo. Se usan en el modo sleep.
	SD xxx	Interface para tarjetas de memoria SD
	SPI xxx	Interface para la memoria Flash de programa. NO USAR
	I2C1XX	Interface CL y DA del bus I2C
	VSPI	Interface SPI de baja velocidad
	TX / RX	Interface serie UART. No usar U0 TX-U0 RX
	HSPI	Interface SPI de alta velocidad

2-4 ESP32-PLC

Y así llegamos a nuestro controlador programable ESP32-PLC. El caso es que el SoC ESP32 trabaja a 3.3V, y todos los módulos y kits de desarrollo que lo usan trabajan con la misma tensión. Esto implica que tanto las señales digitales de entrada o salida, como las entradas analógicas, **NUNCA** deben de superar esta tensión. Ello provocaría la destrucción total o parcial del ESP32.

Aquí es donde ESP32-PLC hace sus aportaciones:

- Utiliza el kit de desarrollo ESP32-DevKitC, con lo que hereda las características del módulo ESP23-WROOM-32 y del controlador SoC ESP32.
- Se alimenta a partir de una tensión de alimentación comprendida entre 7 y 15 VDC. Se puede obtener fácilmente a partir de alimentadores o de pilas y baterías.
- Emplea entradas digitales optoacopladas, entradas analógico/digitales de 0 a 5V, salidas a relés y salidas digitales de alta corriente.
- Conexiones rápidas mediante bornes de paso 5.08.
- Zumbador piezoeléctrico para emitir señales sonoras.



Aunque vamos a emplear el mismo entorno de trabajo o IDE que Arduino, y el mismo lenguaje de programación (o casi), no debemos de olvidar que nuestro micro controlador es un potente y moderno ESP32 con las siguientes prestaciones:

- CPU con arquitectura RISC de doble núcleo y de 32 bits.
- Velocidad de trabajo de 240 MHz.
- Memoria Flash de programa de 4 MB.
- Memoria RAM de datos de 520.
- Memoria ROM de 448 KB con el firmare residente, bootloader, etc…
- Convertidor ADC de 12 bits de resolución.
- Comunicación WiFi 802.11 b/g/n a 2.4GHz y 150 Mbps.
- Comunicación Bluetooth V4.2 BR/EDR y especificaciones LE de bajo consumo

2-4-1 Entrada de alimentación

ESP32-PLC se puede alimentar con una tensión continua de 7 a 15 VDC. Trabajar con 12 VDC es una buena opción y haremos referencia a ella. Esa tensión se regula a 5 Vcc para alimentar al propio ESP32-DevKit C y a toda la electrónica del sistema.

2-4-2 Salida de alimentación

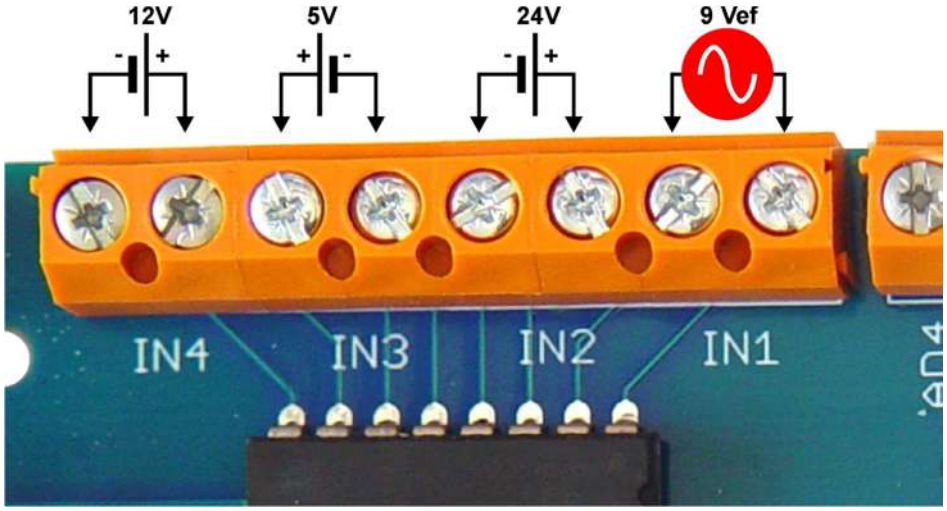
En estos bornes tenemos la misma tensión de entrada junto con la estabilizada de +5 Vcc. Se puede utilizar para alimentar a otros dispositivos externos. La corriente máxima aconsejable para la salida de +5 Vcc es de unos 300mA. La corriente de salida para la de +12 VDC dependerá de la corriente que entregue la fuente de alimentación conectada a la entrada.

2-4-3 Entradas optoacopladas x 4

Proporcionan un aislamiento a las entradas digitales del microcontrolador, permitiendo introducir señales digitales desde sensores o dispositivos con diferentes características eléctricas:

- 4 x Entradas independientes entre sí y denominadas IN1-IN4.
- Admiten niveles lógicos con tensiones de entrada en continua de 2.5V a 30 VDC.
- La polaridad es indiferente.
- Admiten tensiones alterna de 2.5 VAC a 30 VAC ef.

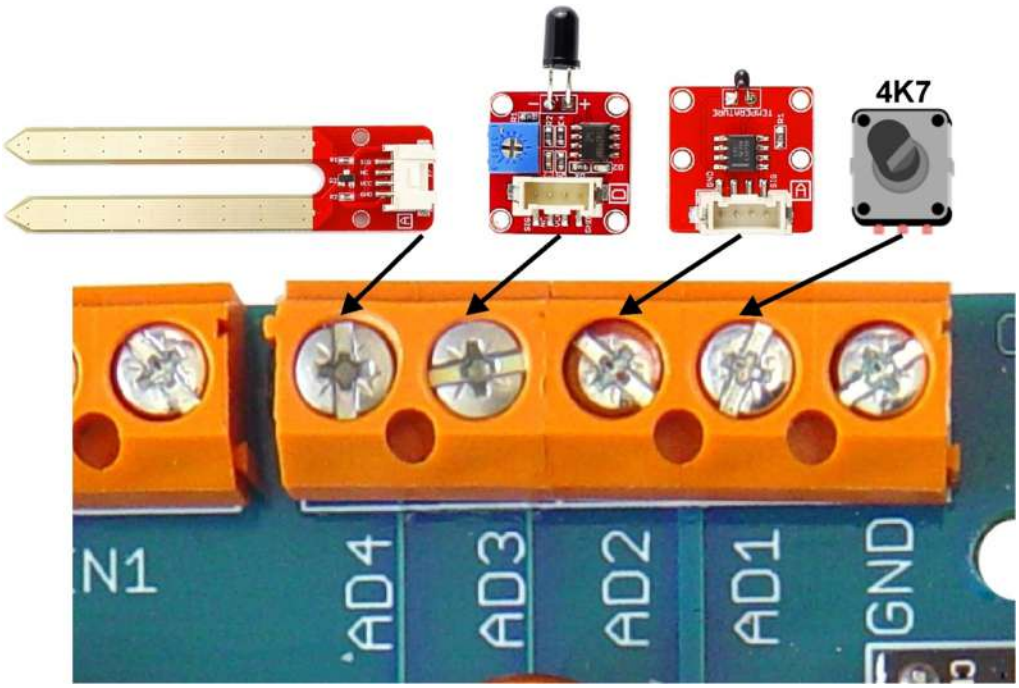
Todo ello nos permite conectar sensores industriales que proporcionan señales digitales con niveles diferentes de tensión y polaridad, tanto en continua como en alterna.



2-4-4 Entradas analógicas/digitales x 4.

Están denominadas como AD1-AD4 y, según se empleen las funciones **analogRead()** o **digitalRead()**, se tratan como entradas analógicas o digitales respectivamente. Son tolerantes a niveles de 0 a 5 VDC y **NO SE DEBE SUPERAR ESTE VALOR**.

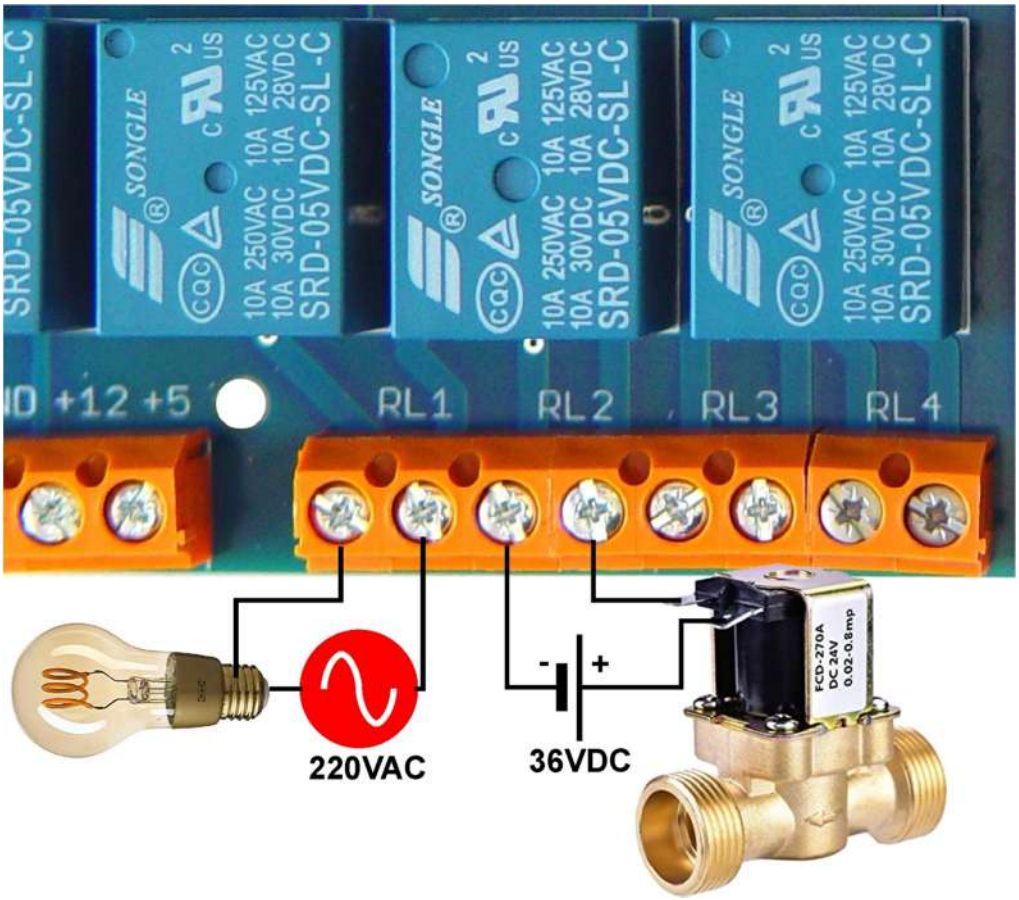
Estas entradas son referenciadas a una GND común y en ellas podemos conectar los típicos sensores que producen señales con esos niveles de tensión.



2-4-5 Salidas a relés x 4

Se denominan RL1-RL4 y proporcionan unos contactos normalmente abiertos que se cierran cuando se activa el correspondiente relé. Según su fabricante los contactos de cada uno de ellos pueden soportar corrientes de hasta 10A/250VAC o 10A /30VDC.

Con esto ya se pueden activar cargas de mayor envergadura como puede ser un sistema de alumbrado, de calefacción, un motor, una electroválvula, etc… También podemos activar a otros relés o contactores que a su vez puedan controlar cargas aún mayores.



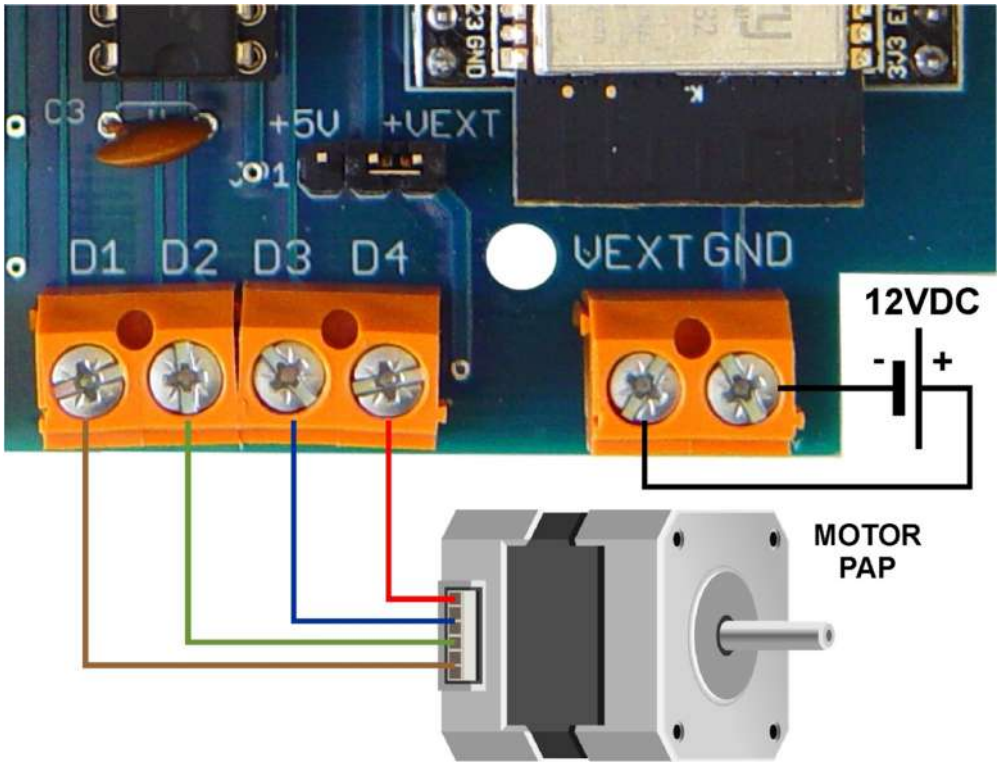
2-4-6 Salidas de alta corriente

Son 4 salidas digitales denominadas DIG1-DIG4 que también pueden proporcionar señales de salida moduladas en anchura o PWM. Están amplificadas tanto en corriente como en tensión con las siguientes características:

- Corriente máxima de 600 mA por salida. Soporta picos de hasta 1.2 A.
- Tensión de alimentación de las cargas entre 4.5 VDC y 36 VDC.
- Frecuencia máxima de conmutación 5 KHz.

Dentro de sus límites, estas salidas se pueden usar para activar todo tipo de cargas de corriente continua como relés, lámparas, motores DC, motores PAP, etc…

Se dispone de un jumper con el que seleccionar la tensión de alimentación de las cargas entre los 5 VDC disponibles en el propio ESP32-PLC (máximo 300 mA), o una tensión externa máxima de 36 V que se debe aplicar entre los bornes VEXT y GND.



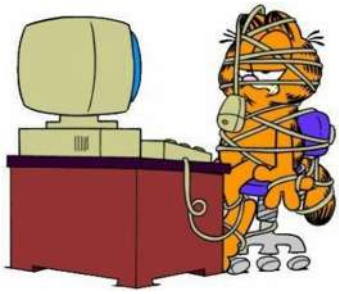
3.- Y ADEMAS…

Si vas a utilizar nuestro ESP32-PLC más allá de los ambientes educacionales, experimentales o de aficionados, y le quieres dar un carácter más profesional, te sugerimos el empleo del kit opcional "**ESP32-PLC-KIT**" que hemos preparado en MK Electrónica. Aunque seguro que encuentras otras opciones, nuestro kit consta de:

1. Carril DIN de 35 mm. Se trata de una barra metálica normalizada empleada en el montaje de elementos y cuadros eléctricos
2. Caja para carril DIN donde se aloja la tarjeta ESP32-PLC y proporciona protección contra el polvo, golpes y humedad, y se puede montar sobre el carril DIN.
3. Alimentador de 12V / 2 A (24 W) que se monta sobre el carril DIN y que alimenta a la tarjeta ESP32-PLC.
4. Juego de pegatinas recortables de vinilo con la distribución de las señales de E/S del ESP32-PLC.

4.- INSTALACIÓN Y CONFIGURACIÓN

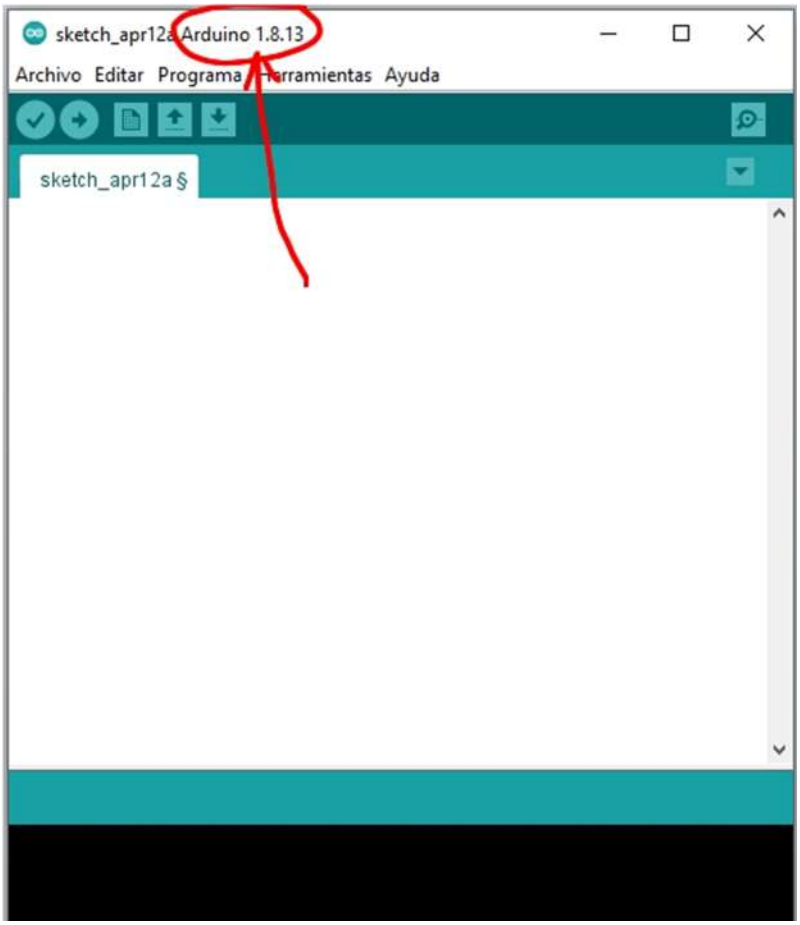
No me cansaré de repetir que vamos a trabajar con un micro controlador que tecnológicamente hablando nada tiene que ver con el Atmega 328P empleado en la tarjeta Arduino UNO u otras similares. Vamos a trabajar con una tarjeta, ESP32-PLC, que emplea el controlador ESP32, mucho más evolucionado, de mayores prestaciones y con capacidades tanto WiFi como Bluetooth.



Ocurre simplemente que para programarla, vamos a seguir usando el mismo lenguaje que empleábamos con Arduino. Al fin y al cabo se trata de una adaptación del lenguaje estándar C++. También trabajaremos con el mismo entorno de desarrollo o IDE. Además, todo ello irá acompañado de librerías diseñadas expresamente para trabajar con el ESP32 y que nos facilitarán la comunicación WiFi y Bluetooth. No te hagas un lío.

Lo primero es descargar ese IDE. Ahora sí que es importante instalar la última versión disponible, pues puede haber diferencias notables con versiones antiguas. A la hora de escribir estas líneas nos encontramos que la versión actual es la 1.8.13 y la puedes descargar desde la página oficial:

https://www.arduino.cc/en/software. Está disponible para Windows, OS X y Linux.



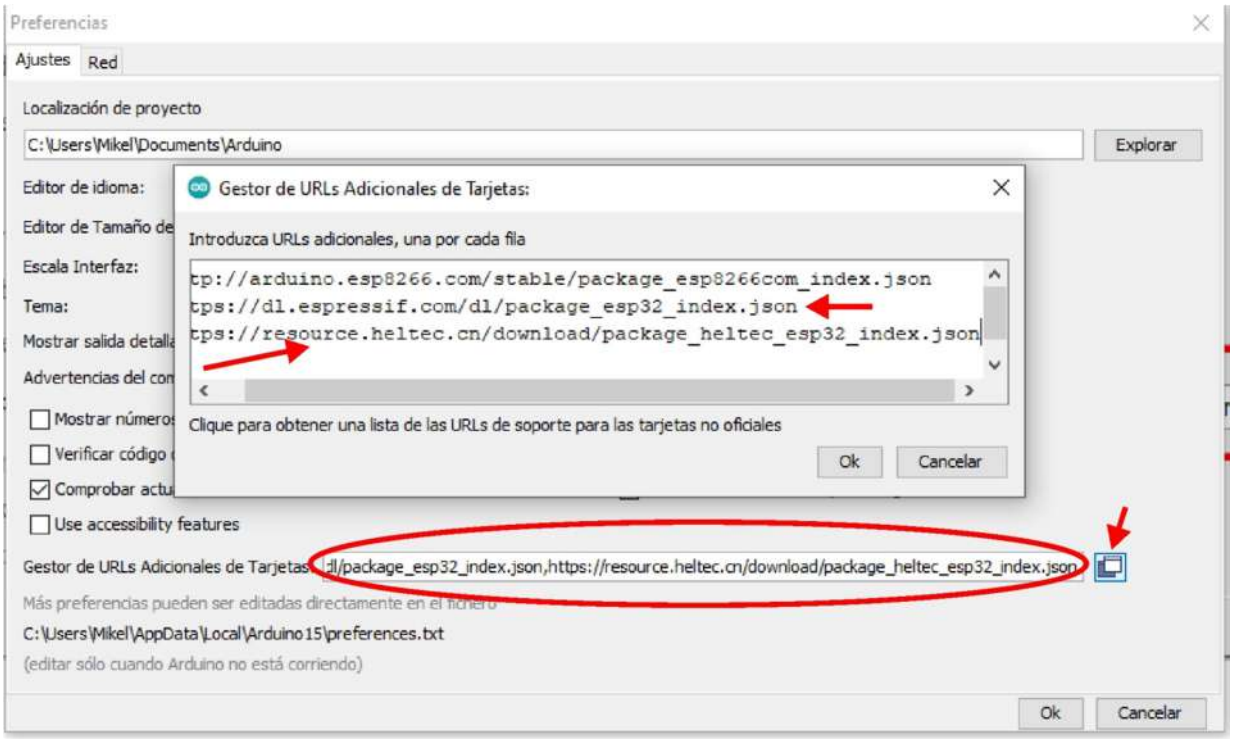
4-1 Gestor de placas

Quizá esto sea una novedad para ti, pero las actuales versiones del IDE permiten trabajar con muchísimas tarjetas controladoras sean o no de Arduino. Lo único que hace falta es instalar lo que se conoce como un "*plugin*" en el que está incluido todo lo necesario para compilar programas escritos en el lenguaje Arduino (*.INO) y que luego se graban en tarjetas controladoras que **NO** son Arduino.

En nuestro caso vamos a emplear módulos y tarjetas basadas en el SoC ESP32. Debemos por tanto instalar las librerías, drivers y programas necesarios que sean capaces de gestionar esas tarjetas.

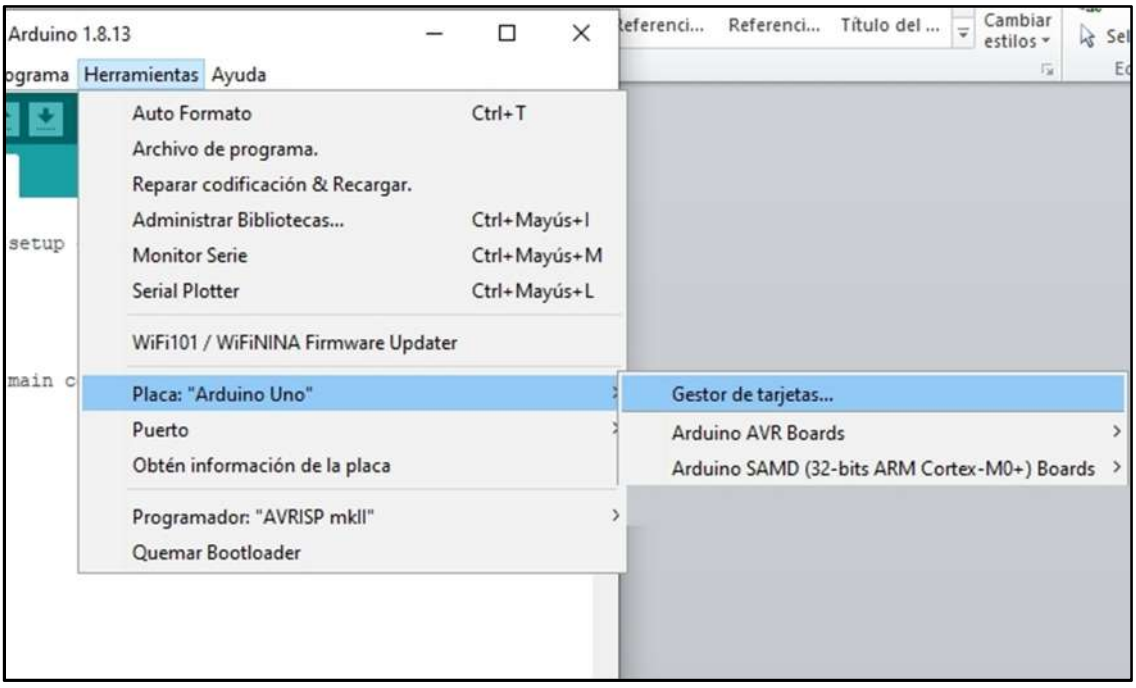
En el campo **Gestor de URLs** de la opción **Archivo → Preferencias** del IDE debemos copiar estas direcciones, que son donde se encuentran esas librerías y programas:

https://dl.espressif.com/dl/package_esp32_index.json
https://resource.heltec.cn/download/package_heltec_esp32_index.json

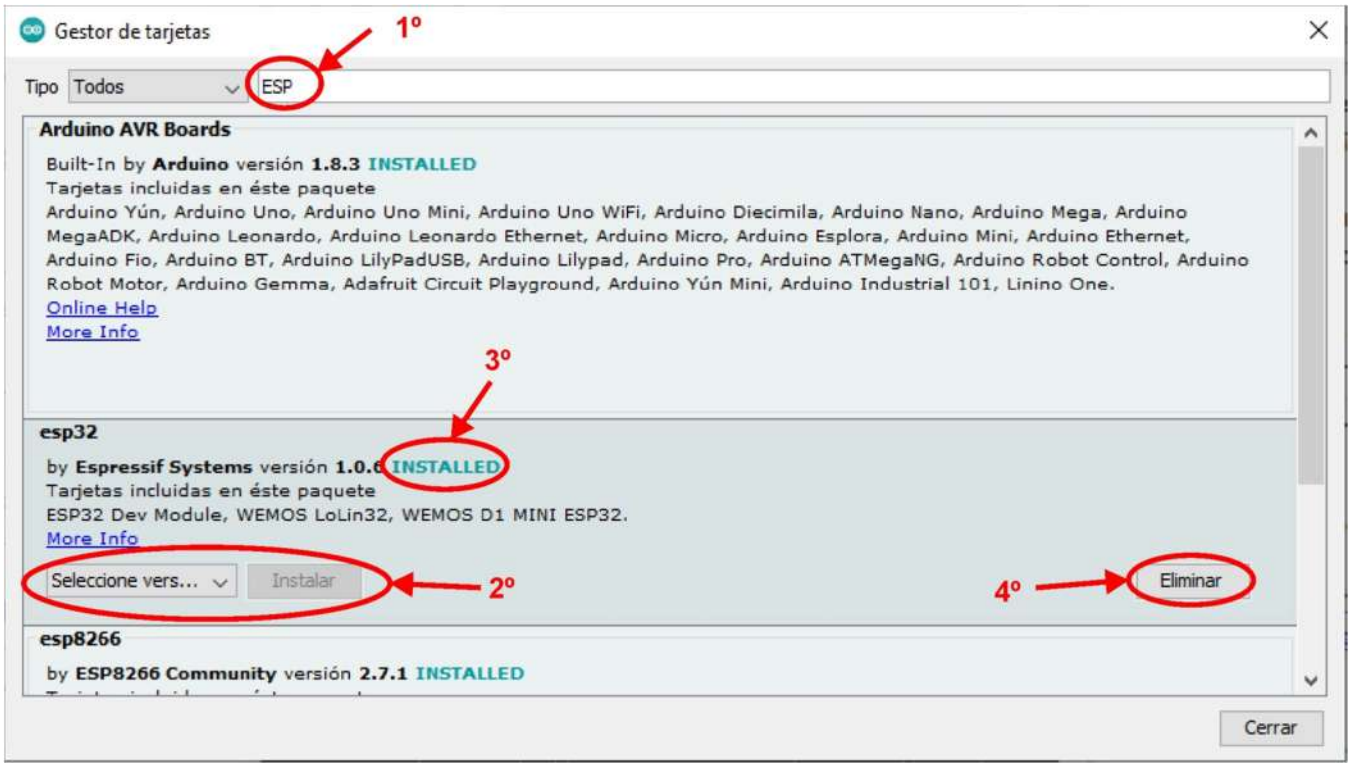


Como es posible que quizá tengas otras direcciones o URL's adicionales, puedes pulsar el botón de la derecha. Se abre una ventana donde puedes insertar y editar tantas direcciones como necesites.

Ahora, vamos a hacer que el IDE busque e instale esas librerías mediante la opción **Herramientas → Placa → Gestor de tarjetas**…



Se abre una ventana como la de la imagen, que representa una especie de buscador.

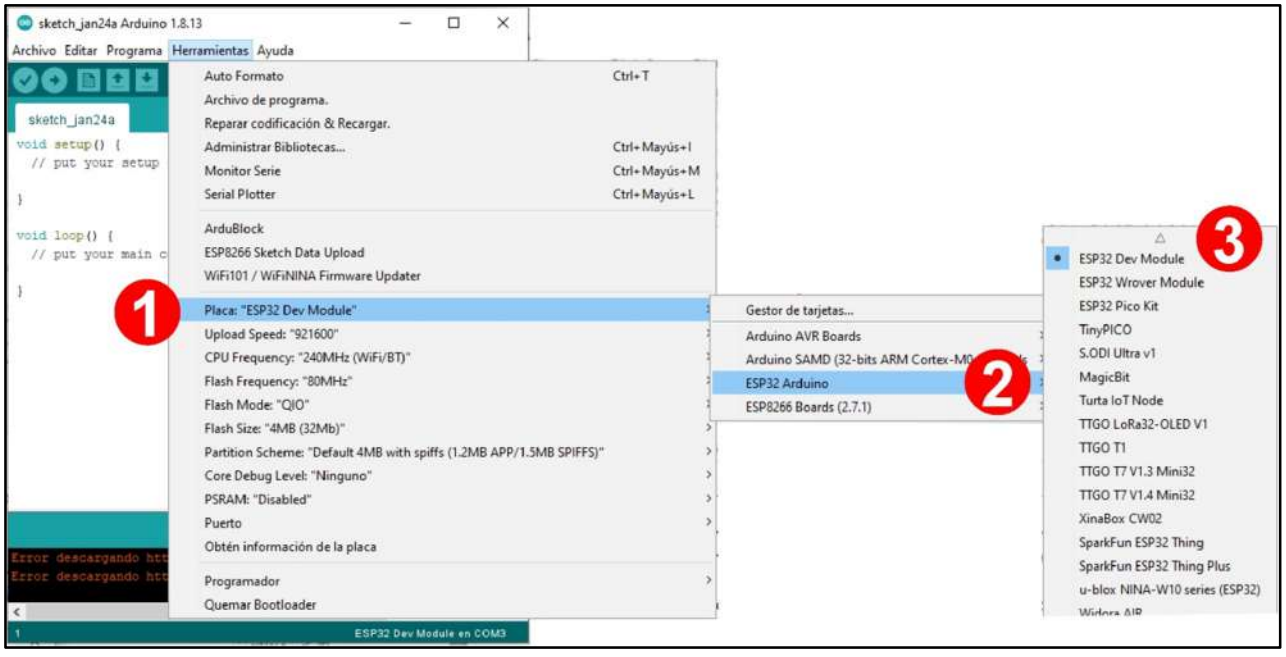


1º En el campo de filtro superior escribimos lo que queremos buscar, en nuestro caso "**ESP**". Inmediatamente aparece todo lo relacionado con tarjetas basadas en ese SoC.
2º Si hay varias versiones podemos seleccionar la que deseamos instalar.
3º En mi caso aparece que ya tengo instalado las librerías y accesorios relacionados con las tarjetas basadas en el SoC ESP32.
4º En esta misma ventana también podemos actualizar y/o borrar esas librerías.

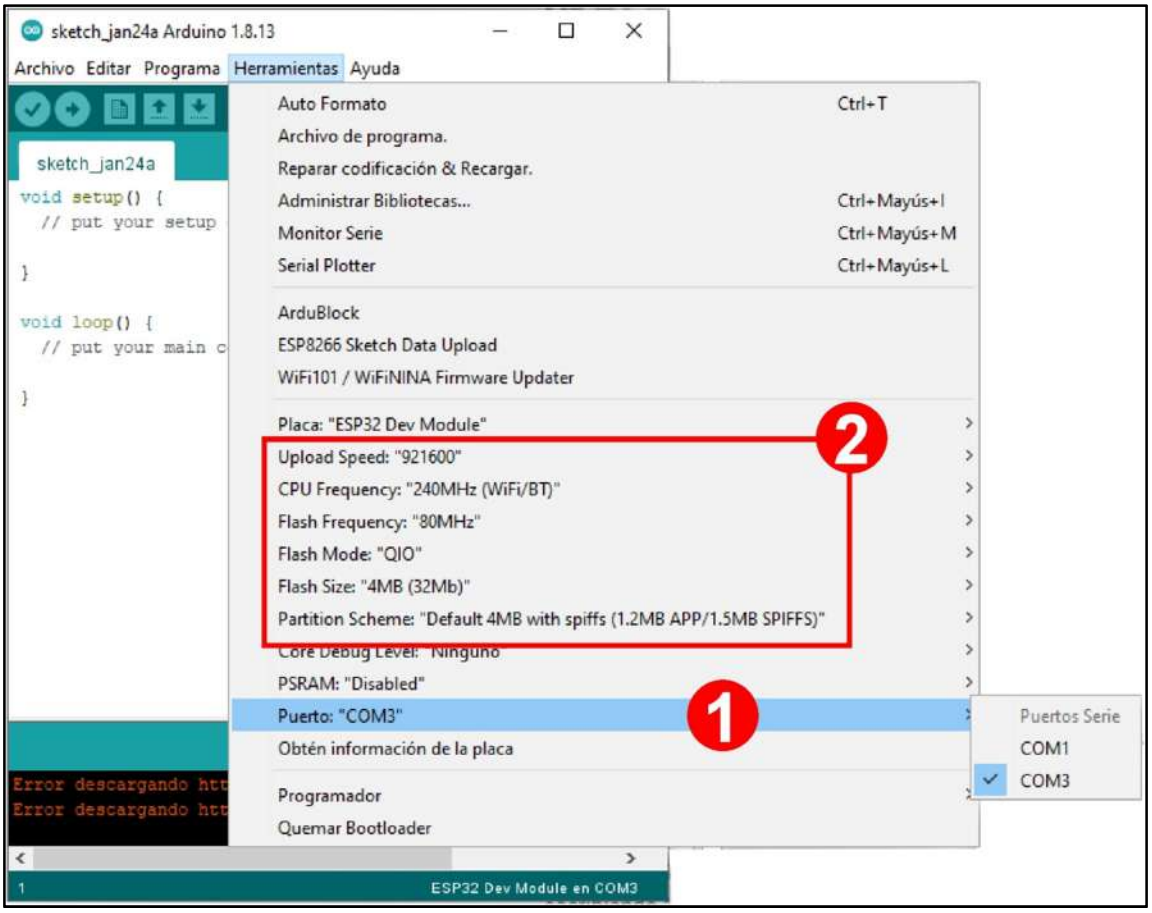
4-2 Configuración

Volviendo nuevamente a **Herramientas → Placas** (1) ahora vemos que aparece el nuevo grupo **ESP32 Arduino** (2). Desde él se despliega un buen número de tarjetas basadas en este SoC y producidas por diferentes fabricantes.

En nuestro caso vamos a seleccionar la tarjeta ESP32 Dev Module (3).



Por último, mediante **Herramientas → Puerto** (1), debemos configurar el puerto COM que el sistema operativo nos asigne cuando conectemos ESP32-PLC al puerto USB. También podemos ver un resumen de las características del controlador (2).

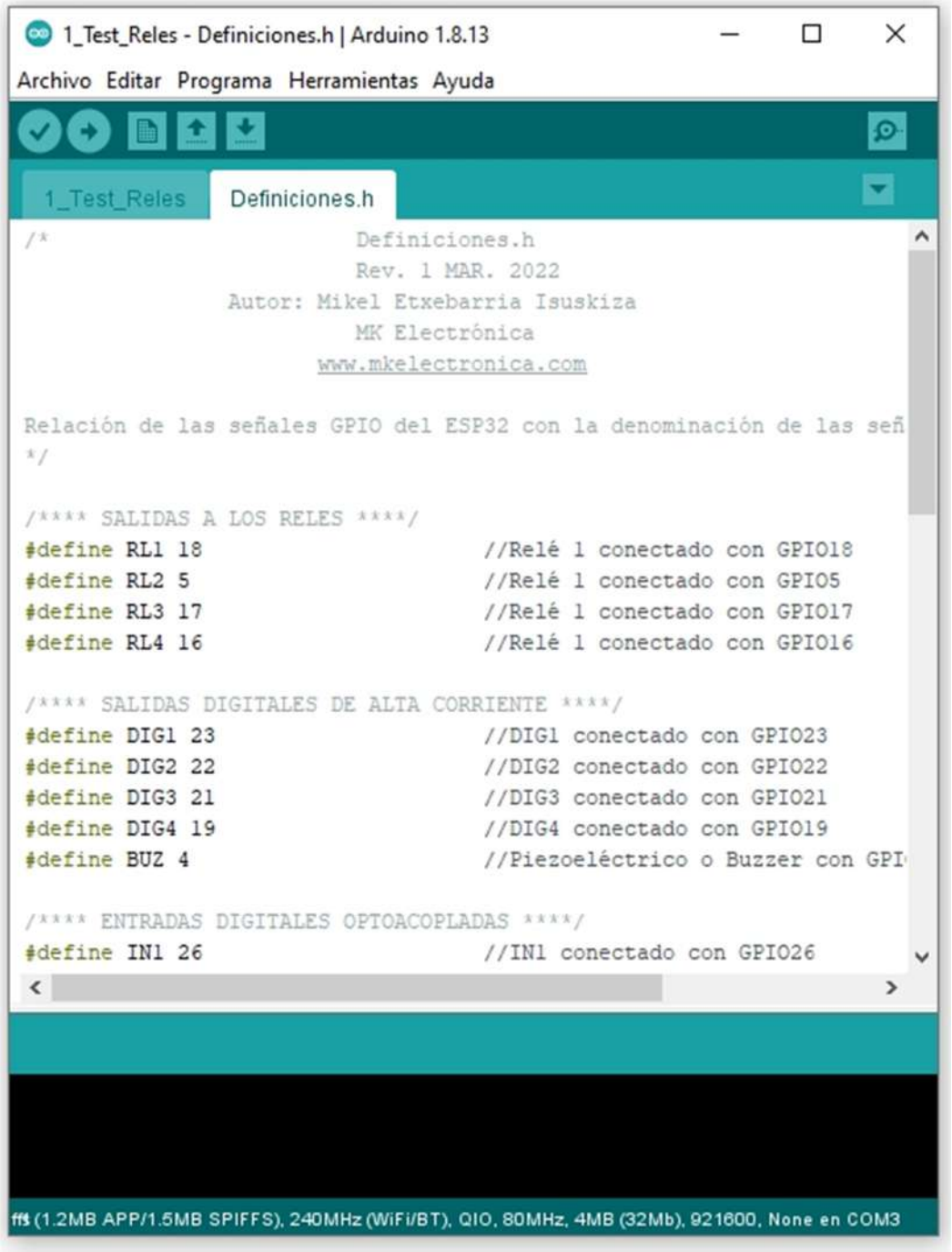


5.- EJEMPLOS

La mejor forma de ver funcionando ESP32-PLC es mediante una serie de ejemplos como los que proponemos en este apartado. No es nuestra intención ofrecer soluciones completas ni ejemplos complejos. Preferimos facilitar programas sencillos pero comprensibles. Que sirvan para ver las posibilidades de ESP32-PLC y su controlador ESP32, así como destacar algunas de las diferencias de programación respecto al popular Arduino.

Hemos dividido estos ejemplos 3 grupos: Básicos, usando comunicaciones Bluetooth y usando comunicaciones WiFi. También hemos empleado algunos sensores y actuadores bastante comunes comercializados por MK Electrónica. Naturalmente podrás emplear otros similares.

En todos esos ejemplos hemos incluido un fichero auxiliar llamado "**Definiciones.h**". Contiene la definición que relaciona las señales empleadas por las E/S de ESP32-PLC con las señales GPIO propias del controlador ESP32.



Podemos ver cómo las señales RL1-RL4 que gobiernan las salidas a los relés, están asociadas a GPIO18, GPIO5, GPIO17 y GPIO16 respectivamente del SoC ESP32. Las salidas de alta corriente DIG1-DIG4 se conectan con GPIO23, GPIO22, GPIO21 y GPIO19, la salida al zumbador BUZ se conecta con GPIO4 y así sucesivamente.

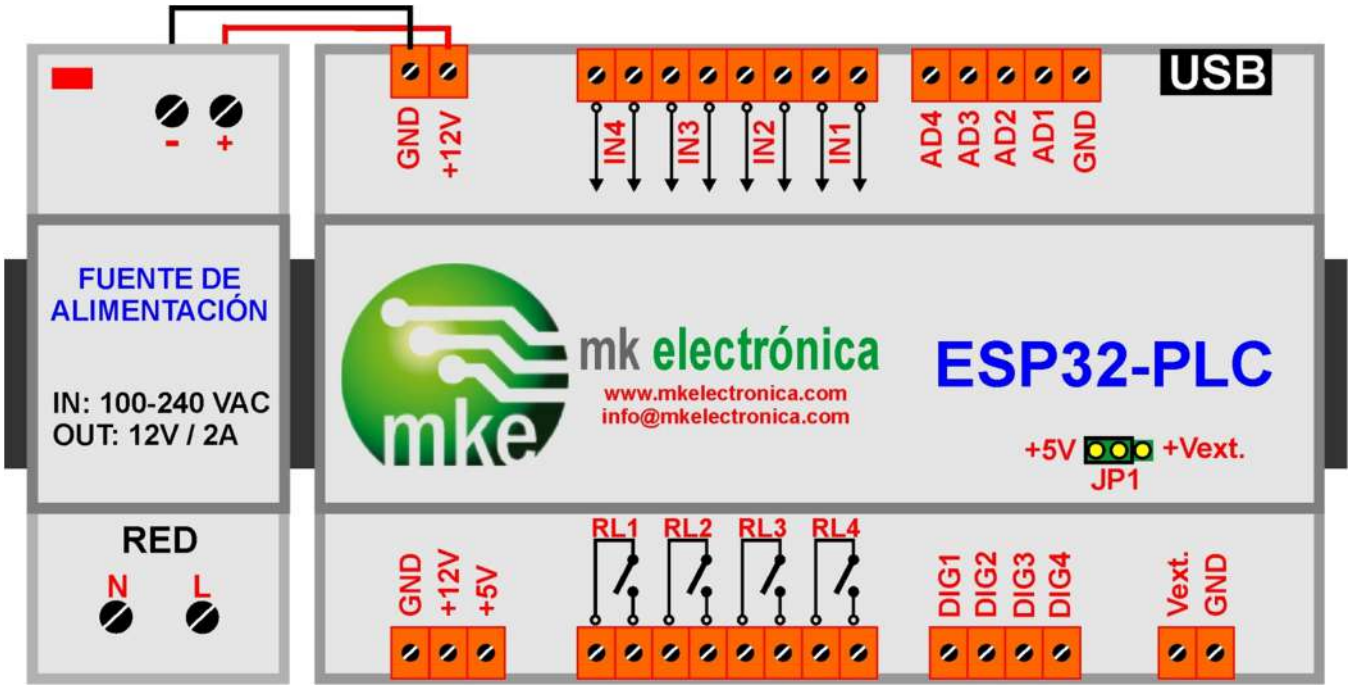
Este mismo fichero auxiliar también contiene una única función, **configurar_ES()**, que la emplearemos en el **setup()** de nuestros programas. Con ella podremos configurar rápidamente todas las señales de E/S del ESP32-PLC.

5-1 Ejemplos básicos

Trataremos de usar la mayor parte de recursos que ofrece el ESP32-PLC y su controlador, el SoC ESP32. También mostraremos cómo conectar diferentes periféricos. Esperamos que te sean de utilidad y te sirvan de referencia para tus propios proyectos.

5-1-1 Test_Relés

Este primer ejemplo consiste en probar el funcionamiento de los 4 relés RL1-RL4 de ESP32-PLC así como su zumbador. Se irán activando/desactivando secuencialmente. En este caso lo único que tenemos que hacer es conectar la tensión de alimentación.

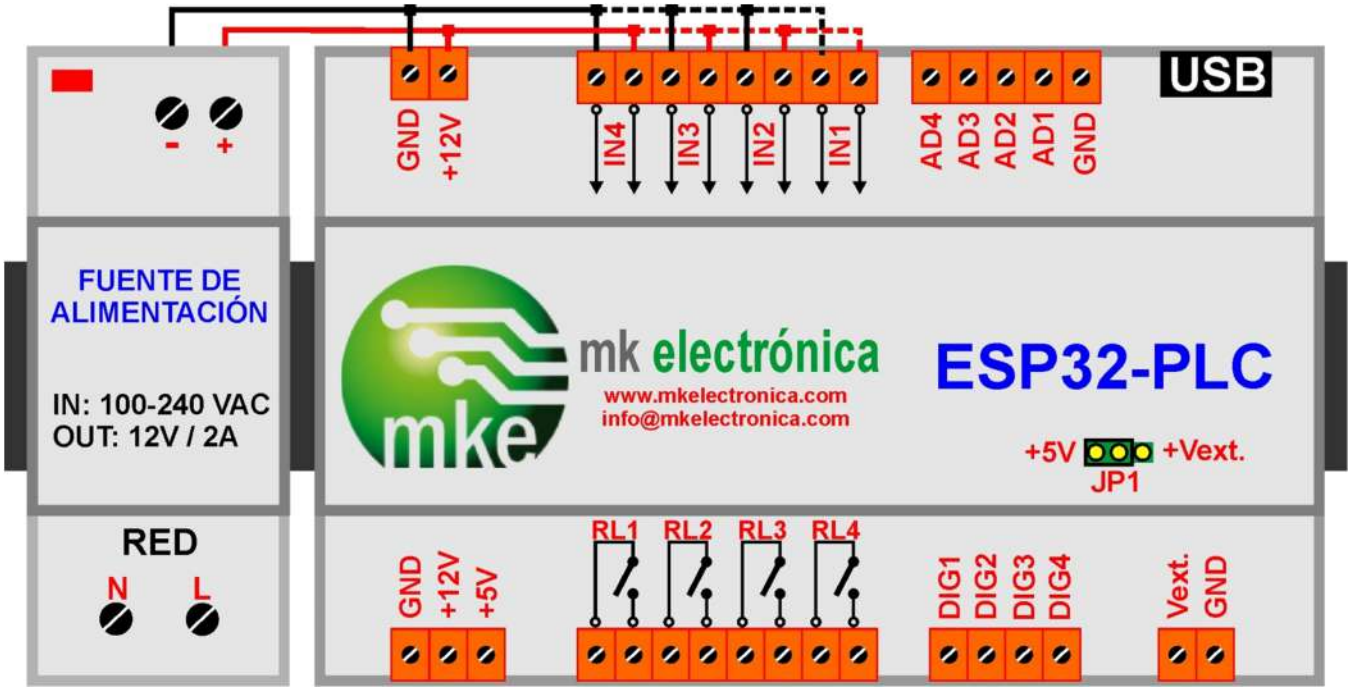


Basta con escuchar el sonido de cada relé ("*clack*") cuando se cierran sus contactos. También podemos conectar en los bornes un polímetro midiendo continuidad. La resistencia será de 0 Ω cada vez que se active el relé correspondiente.

5-1-2 Test Entradas Optoacopladas

Una vez que sabemos que los relés funcionan correctamente, podemos comprobar el funcionamiento de cada una de las entradas optoacopladas IN1-IN4 que activarán los relés RL1-RL4 respectivamente. El monitor serie del IDE también monitoriza el estado de esas entradas.

Basta con conectar la misma tensión que empleamos para alimentar el equipo, por ejemplo 12 VDC, e ir aplicándola sucesivamente en las entradas IN1-IN4.



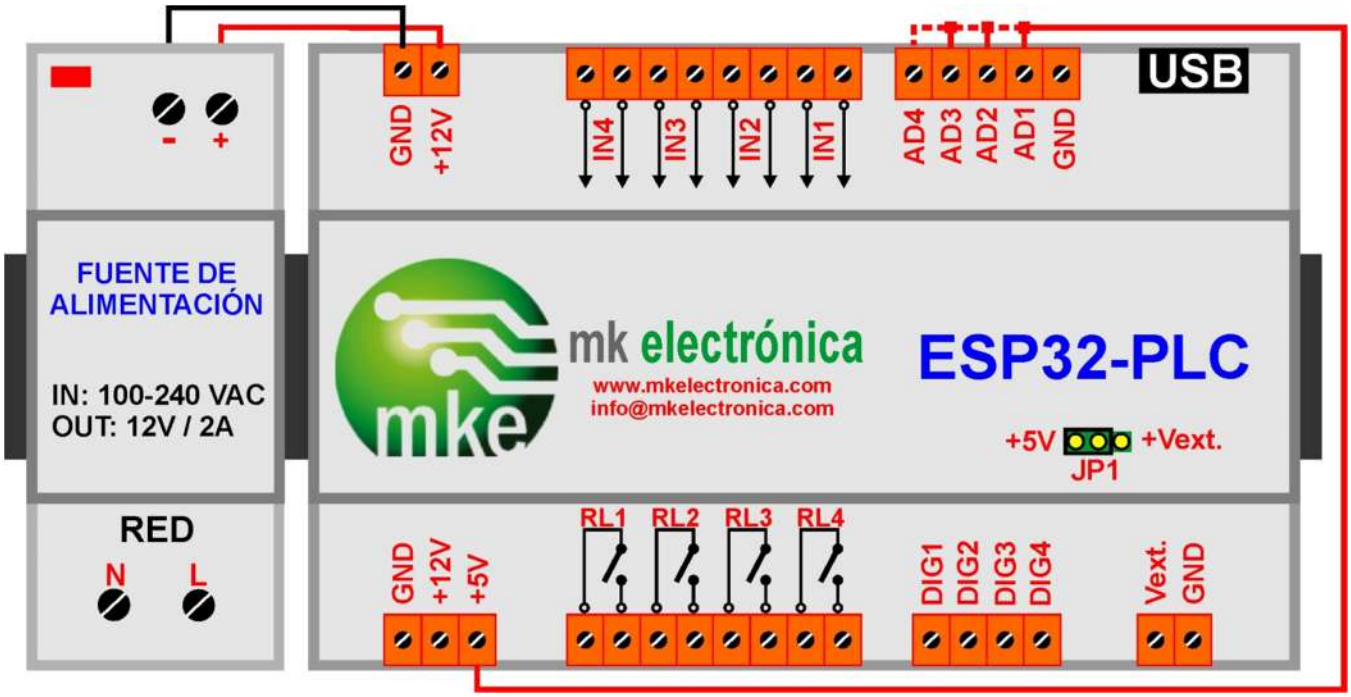
No importa el valor de dicha tensión ni tampoco su polaridad. Basta con que esté dentro de los límites de 2.5V a 30V sean en continua (DCV) o en alterna (ACV). Puedes probar con distintos valores.

Las entradas optoacopladas producen señales en colector abierto. Esas señales se configuran como entradas con resistencia pull-up y funcionan con lógica inversa. Se lee un nivel "0" cuando la entrada tiene tensión y un nivel "1" cuando no hay tensión.

5-1-3 Test Entradas Digitales

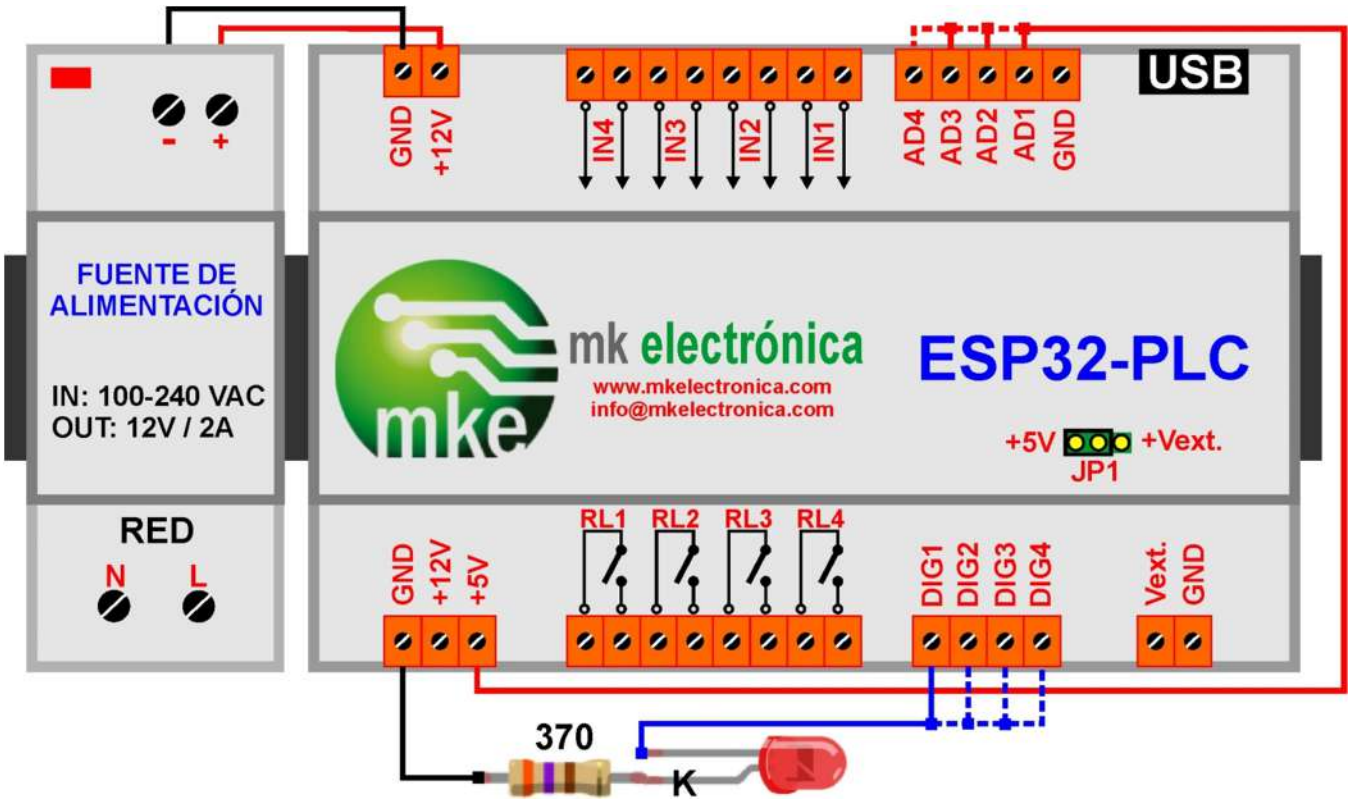
En este ejemplo se trata de comprobar las entradas AD1-AD4 en el modo digital activando/desactivando los relés RL1-RL4 respectivamente. El monitor serie del IDE también monitoriza el estado de esas entradas.

Hay que recordar que estas entradas únicamente son tolerantes a señales de 0 y 5V con respecto a GND. Conectadas con GND o sin conexión alguna, la entrada produce un nivel "0". Conectadas a +5V produce un nivel "1". En ningún caso se debe superar esta tensión. Para hacer las pruebas podemos tomar la tensión de 5V que produce el propio ESP32-PLC y la vamos conectando sucesivamente a cada uno de las entradas.



5-1-4 Test Salidas Digitales

Para probar las salidas digitales DIG1-DIG4 de alta corriente podemos usar un simple led que se conecta sucesivamente con cada una de ellas. Mira la figura.



El ejemplo lee las entradas AD1-AD4 en el modo digital, como en el ejemplo anterior, y activa/desactiva cada salida DIG1-DIG4 respectivamente.

El valor de la tensión de estas salidas digitales de alta corriente depende de la posición del jumper JP1. Puede ser de +5V como en este ejemplo, o la tensión que se aplica desde el exterior a través de los bornes VEXT- GND. **No te olvides de él**.

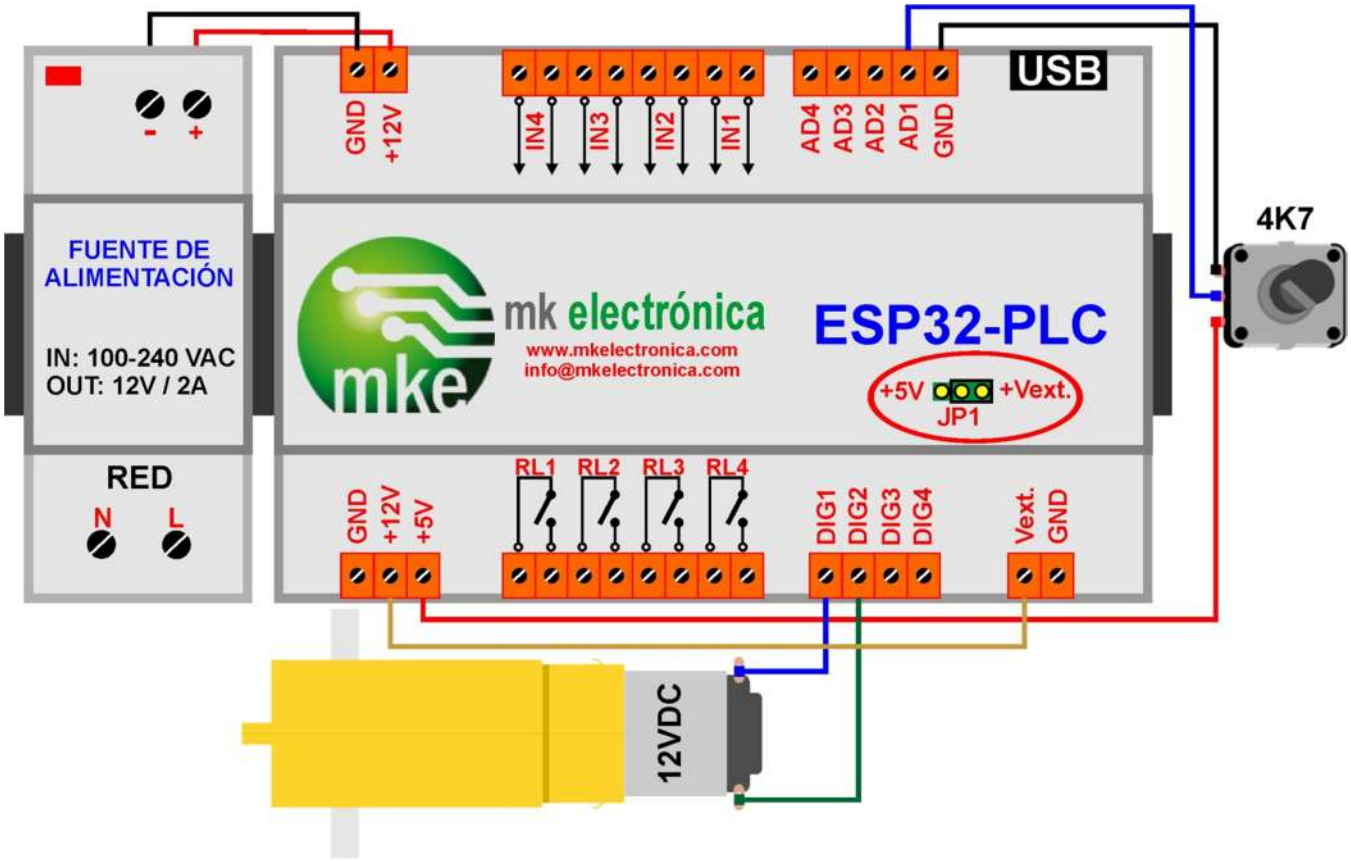
5-1-5 Test Entradas Analógicas

Ahora comprobamos las entradas AD1-AD4 en el modo analógico. Basta con conectar a +5V un potenciómetro cuyo cursor lo vamos conectando sucesivamente con cada una de ellas. En el monitor serie se visualizan los valores leídos en cada entrada.

5-1-6 Motor DC PWM

En este caso vamos a usar el mismo potenciómetro conectado con la entrada analógica AD1, para generar una señal PWM por la salida DIG1 de alta corriente. Con ello controlaremos la velocidad de un motor DC conectado entre DIG1 y en DIG2. Observa que:

1. El potenciómetro produce una tensión de 0 a 5V en AD1
2. El jumper JP1 se coloca en la posición +VEXT
3. La salida de +12 V se conecta con la entrada VEXT. de tensión externa. El motor trabaja a 12VDC.



Este ejemplo muestra cómo se generan señales PWM en un controlador ESP32. Es diferente a como se hace en Arduino:

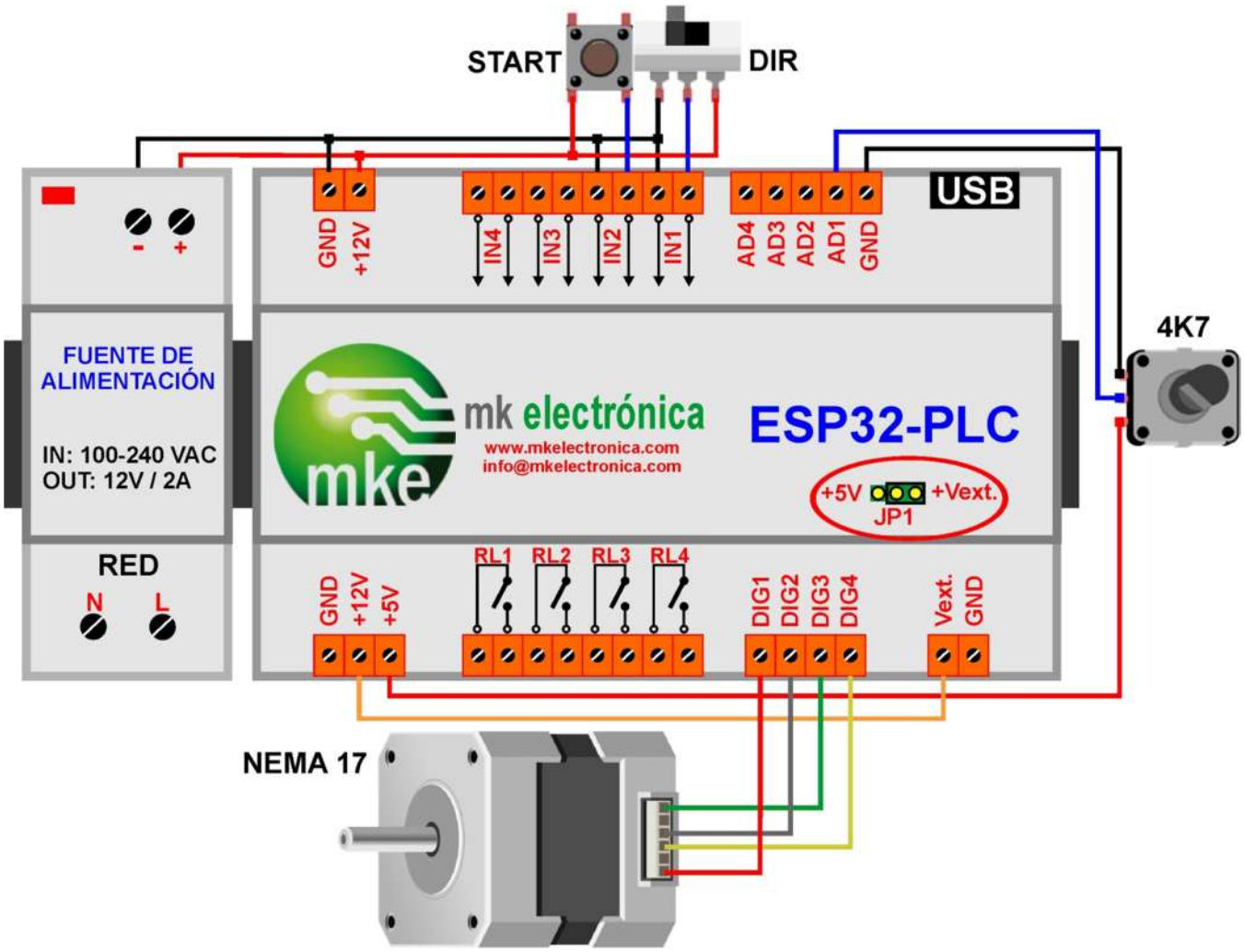
1. Se define un canal PWM: **#define C1 1**
2. Se ajusta la frecuencia y resolución del canal PWM: **ledcSetup(C1, 1000, 10);**
3. Se establece la conexión entre el canal PWM y un pin: **ledcAttachPin(DIG1, C1);**
4. Se genera la señal PWM por ese canal: **ledcWrite(C1,cicloUtil);**

5-1-7 Motor PAP

Vamos a ir un poco más lejos controlando un motor paso a paso (PAP) bipolar. En este caso el montaje se complica un poco:

- El conmutador DIR conectado en la entrada IN1 permite seleccionar el sentido de giro.
- El pulsador START conectado en la entrada IN2 inicia el ciclo o secuencia de movimiento.
- El potenciómetro conectado en la entrada AD1 en el modo analógico introduce una tensión de 0 a 5V que determina el Nº de pasos a girar entre 0 y 4095

- Las dos bobinas del motor se conectan con las salida digitales de alta corriente DIG1-DIG4.
- Mediante el jumper JP1 se selecciona una tensión externa de 12V que se obtienen directamente de la salida del propio ESP32-PLC.

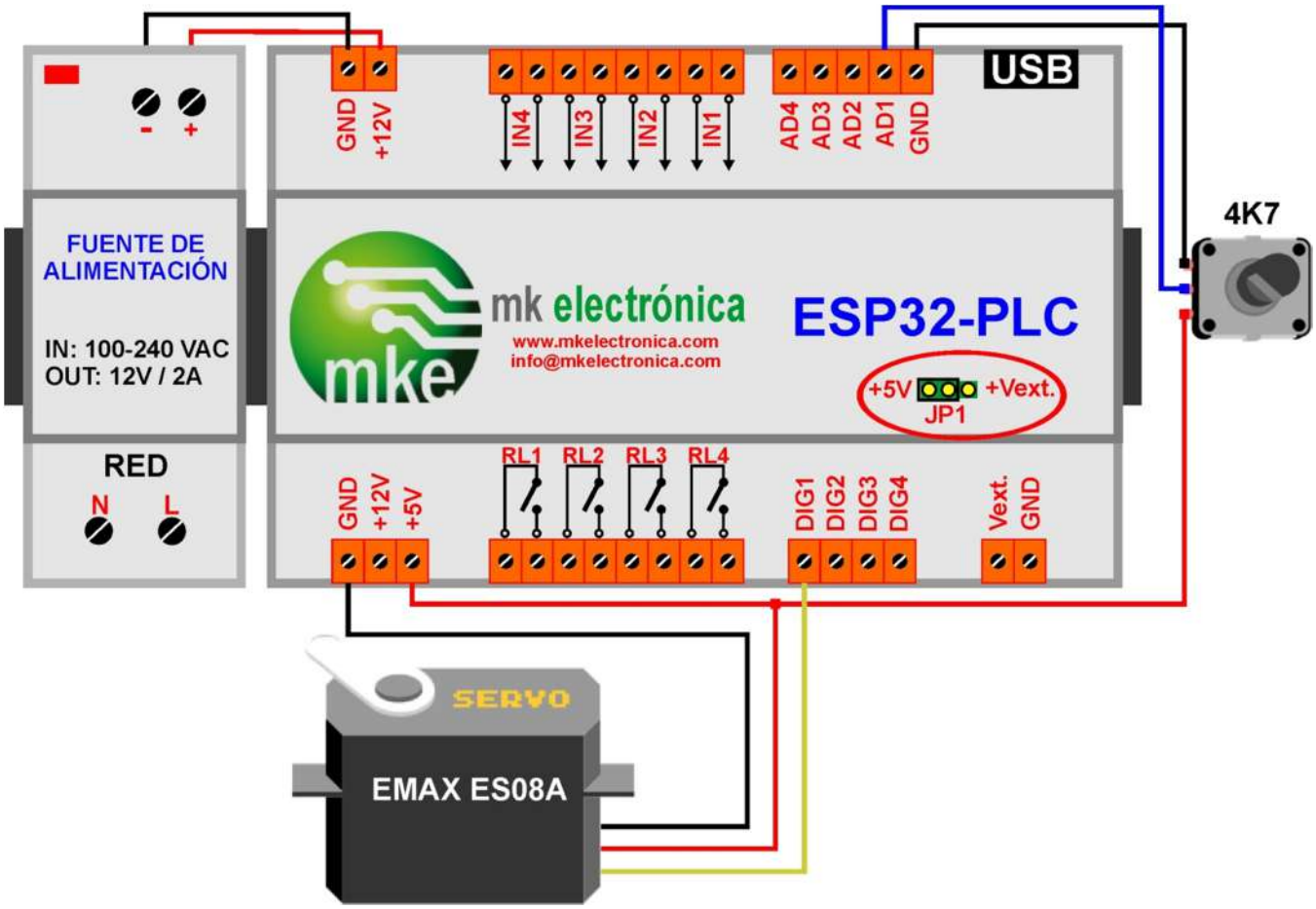


El monitor serie del IDE muestra un informe con los pasos a girar, los grados y las vueltas que girará el eje del motor

5-1-8 Servo Motor

Este sencillo ejemplo nos muestra la posibilidad de controlar un servomotor conectado en la salida DIG1. Mediante el potenciómetro conectado en la entrada AD1 en el modo analógico producimos el giro del servo entre 0º y 180º.

El servo empleado trabaja a 5V que se toman directamente desde la salida +5V del ESP32-PLC. Es conveniente que JPI se ponga en la posición +5V por defecto.



En este caso debemos usar la librería "**ESP32Servo.h**" específica para el controlador ESP32.

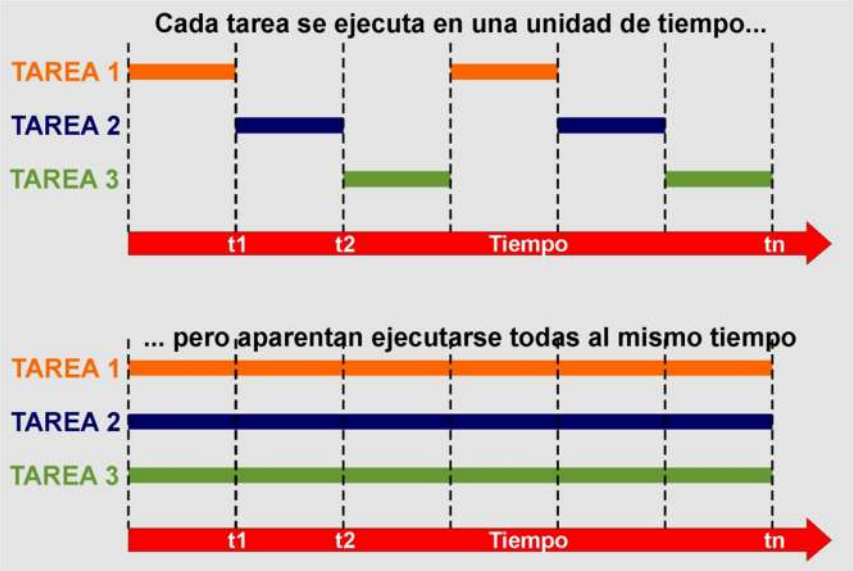
5-1-9 Multitarea

Quizá este ejemplo no tenga una utilidad práctica inmediata, pero trata de un tema muy interesante que seguro te puede ser útil en el futuro. Se trata de la "**MULTITAREA**" o capacidad de hacer varias tareas al mismo tiempo (o casi).

En principio cualquier procesador puede hacer varias tareas aparentemente de forma simultánea. Basta con un sistema planificador o "*Scheduler*" que se encarga de gestionar qué tarea se ejecuta en cada momento y durante cuánto tiempo.

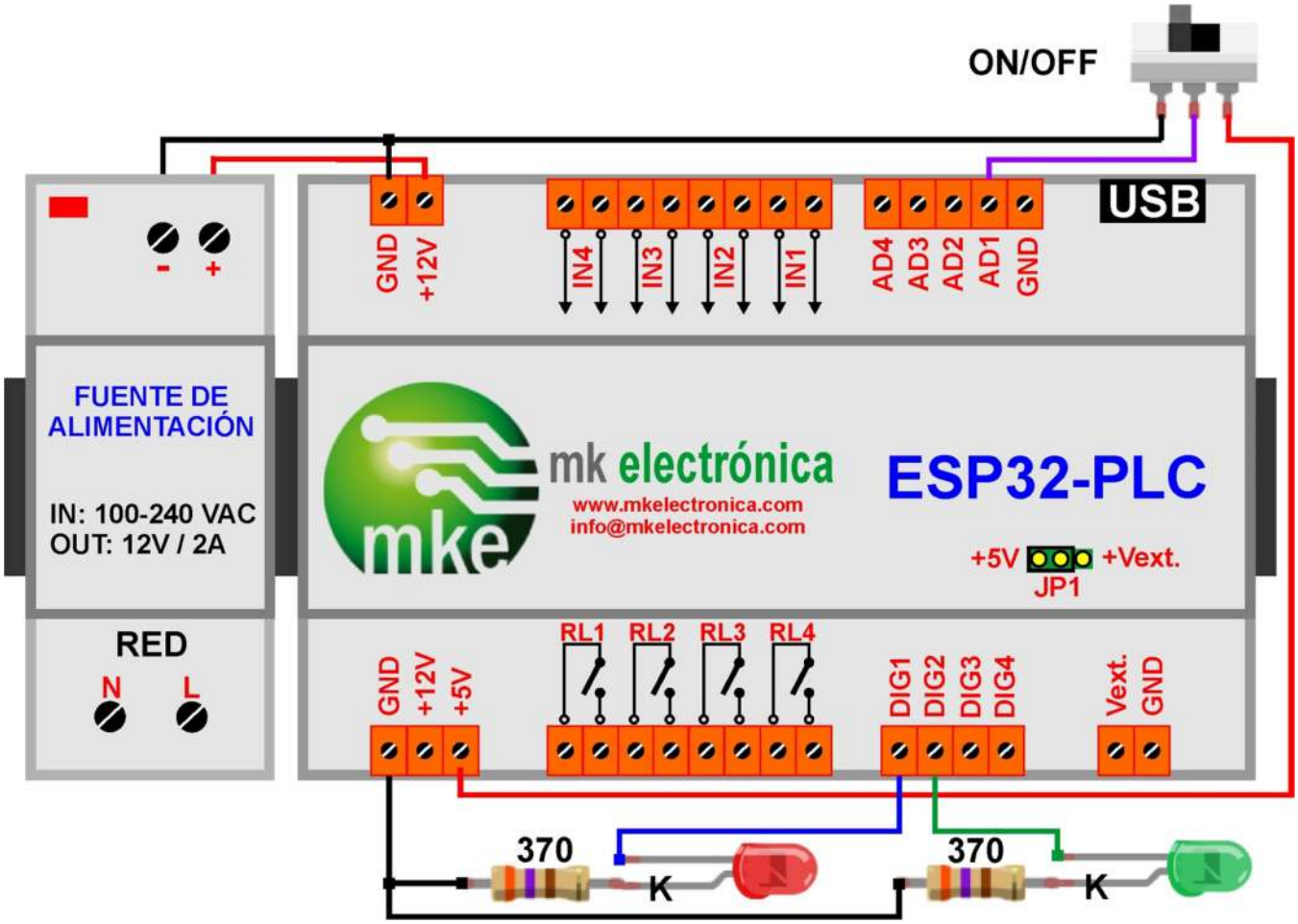
En la figura se muestra la ejecución de 3 tareas que se ejecutan en unidades de tiempo controladas por el planificador. Dependiendo de él y de la velocidad del controlador parece aparentar que todas se ejecutan al mismo tiempo.

¿Qué se consigue con esto? Al dividirlo en tareas independientes, se mejora la comprensión de un programa. Si una tarea se bloquea las demás siguen ejecutándose. En definitiva se gana en seguridad y eficiencia.



Mediante el planificador adecuado, esto se puede hacer con cualquier micro controlador, pero obviamente cuanto más veloz sea, mejor. Recuerda que nuestro ESP32 es un doble núcleo, de 32 bits y a 240 MHz,

Vamos con el ejemplo para el que proponemos el siguiente montaje…



- El led rojo se conecta con la salida DIG1 a +5V. Para él hemos creado una tarea, "**blink1**", que ejecuta la función "**blinkDIG1()**" haciéndolo parpadear cada segundo.
- El led verde se conecta con la salida DIG2. Para él hemos creado una tarea, "**blink2**", que ejecuta la función "**blinkDIG2**", que lo hace parpadear cada 0.5 segundos.
- El interruptor se conecta en la entrada AD1 en el modo digital. Para él hemos creado una tarea, "**testeo**", que ejecuta la función "**testAD1()**" que habilita o suspende la tarea "blink1" según su estado.

Cuando grabes el programa verás que…

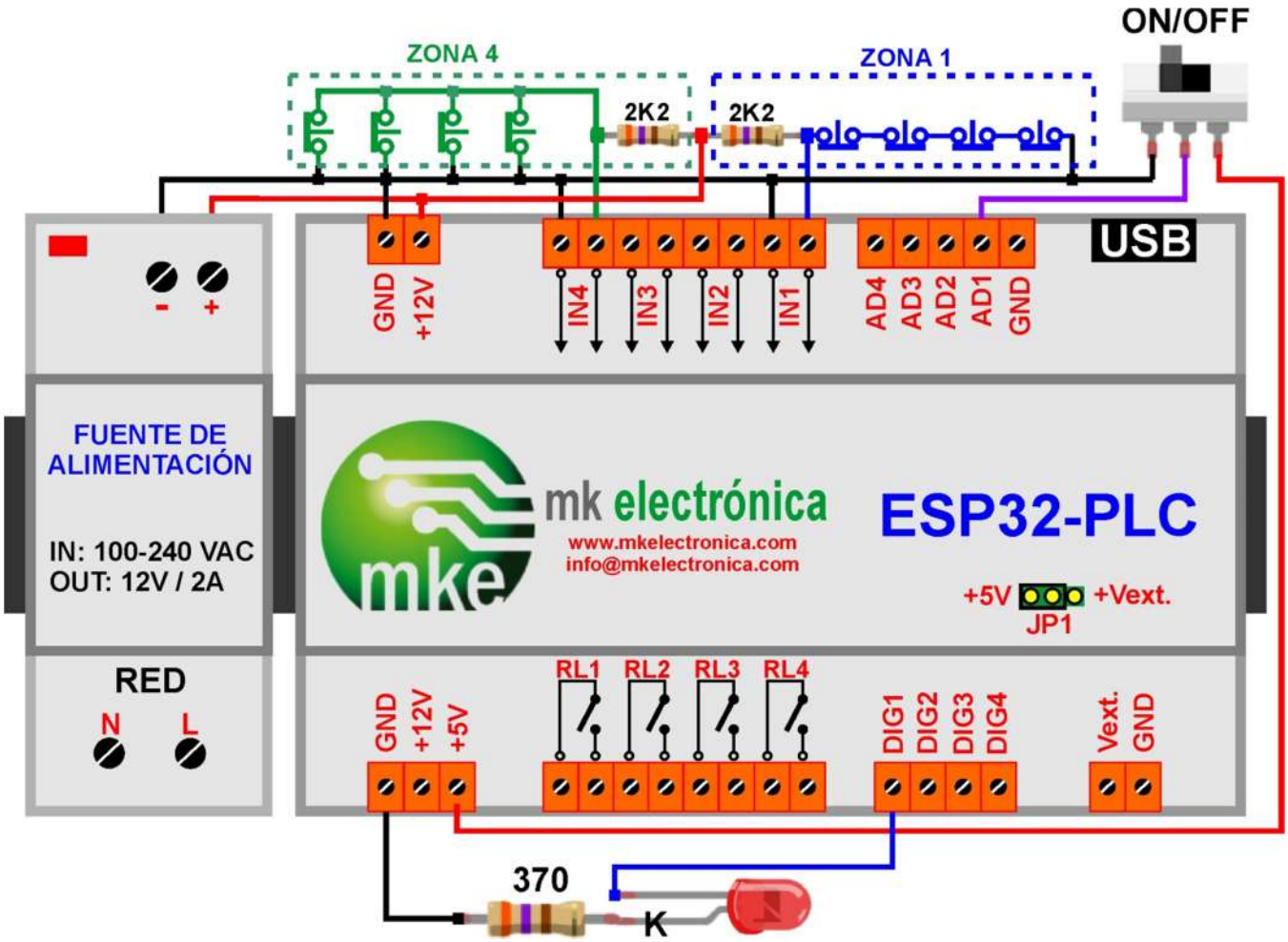
- Ambos leds parpadean simultáneamente y cada uno a su ritmo.
- Si mueves el interruptor, la respuesta es inmediata y la tarea **blink1** se suspende o se habilita.
- Si eliminas los comentarios de las funciones **while(1);** y **{delay(1);]** simulas que la función **blinkDIG2()** de la tarea **blink2** se ha bloqueado en un bucle sin fin, pero las otras dos tareas siguen funcionando.

Si quieres comprobar la funcionalidad y eficiencia de la multitarea te invito a que diseñes un programa que haga exactamente lo mismo que el de este ejemplo, pero por el método tradicional conocido como "**Super loop**".

5-1-10 Alarma

Un ejemplo práctico que usa la multitarea para emular una alarma. Empezamos por explicar las conexiones que hemos hecho sobre el ESP32-PLC.

- El interruptor ON/OFF permite activar (armar) la alarma cuando se pone a "1" y desactivarla o dejarla fuera de servicio cuando se pone a "0".
- El led conectado en la salida DIG1 junto con el zumbador de ESP32-PLC emiten señales luminosas y sonoras cuando la alama entra en servicio y/o se dispara.
- En la entrada optoacoplada IN1 se ha conectado un conjunto de pulsadores en serie y normalmente cerrados. Cuando se abre **cualquiera** de ellos la alarma "ZONA 1" se dispara. Emulan los interruptores magnéticos para puertas y ventanas.
- En la entrada optoacoplada IN4 se ha conectado un conjunto de pulsadores en paralelo y normalmente cerrados. Si se abren **todos** ellos se dispara la alarma por "ZONA 4".
- La alarma en la ZONA 1 produce una señal sonora en el zumbador, y la alarma en la ZONA 4 dispara el relé RL1.

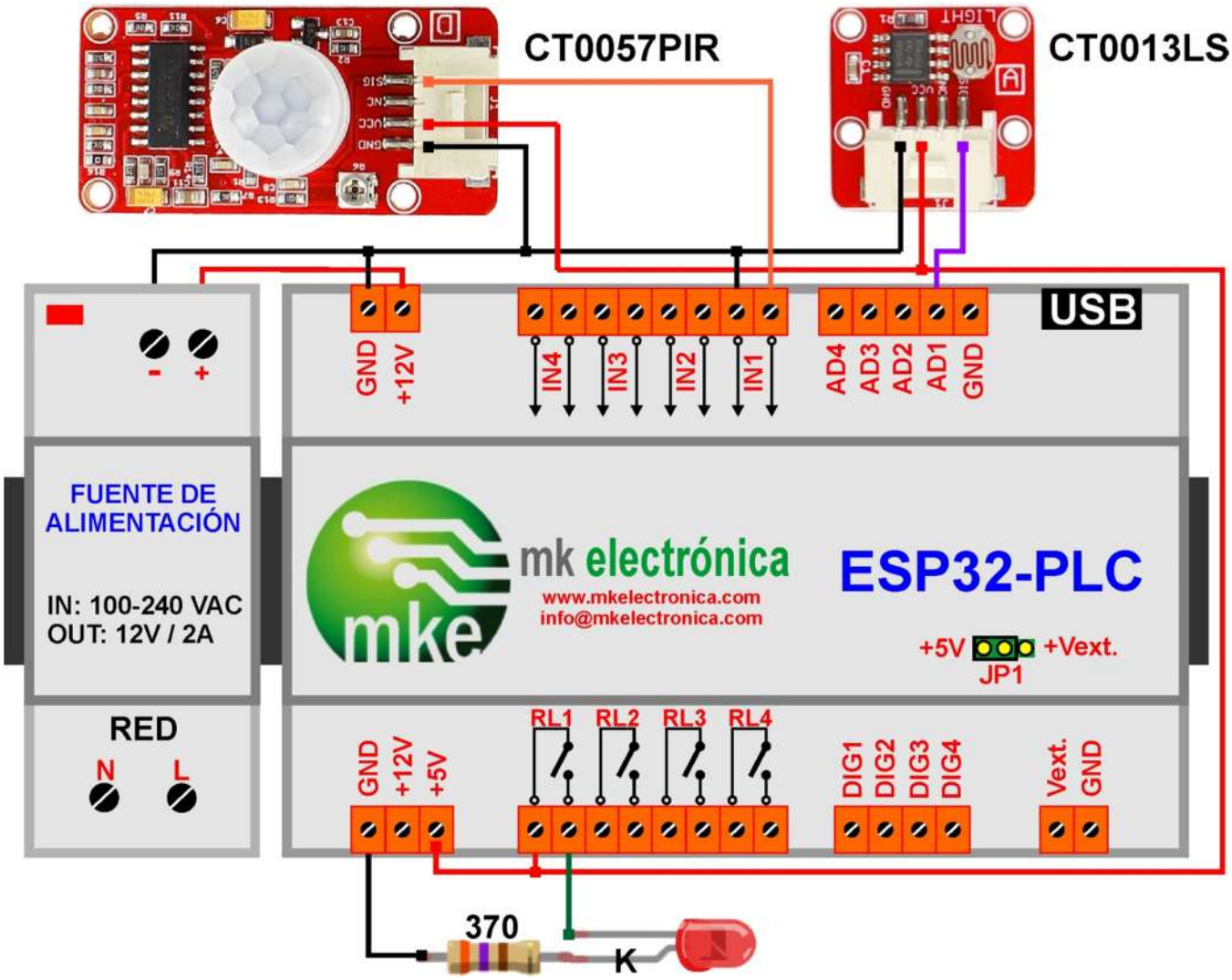


Hemos creado tres tareas: **testeo**, **armar** y **testAlarma**. Para descongestionar el programa principal cada tarea se ha guardado en sus correspondientes ficheros auxiliares independientes: "*Tarea_testeo.hpp*", "*Tarea_armar.hpp*" y "*Tarea_testAlarma.hpp*". Se incluyen mediante sendas directivas **#include**. Vamos a explicar cada tarea:

- **testeo**: Es la única tarea activa al iniciar el sistema. Ejecuta la función **testAD1()** que comprueba el estado del interruptor en la AD1. Si está a "1" se habilita la tarea **armar** para activar la alarma. En caso contrario se suspenden las tareas **armar** y **testAlarma** y se desconectan las salidas. Alarma desactivada o fuera de servicio.
- **armar:** Ejecuta la función **armarAlarma()** que habilita la tarea **testAlarma**, genera un aviso sonoro con 8 beeps por el zumbador y mantiene al led en DIG1 en una intermitencia constante cada 0,1" como aviso acústico luminoso de que la alarma está armada en vigilancia.
- **testAlarma:** Ejecuta la función **testSensores()**. Lee constantemente los sensores de las entradas IN1 (ZONA 1) y de IN4 (ZONA 4). Si se activa la ZONA 1 se produce una intermitencia sonora en el zumbador. Si se detecta una alarma en la ZONA 4 se activa el relé RL1.

5-1-11 Alumbrado automático

Con objeto de mostrar el empleo y conexión de algunos sensores comerciales, vamos emular un sistema automático de alumbrado como el que puede haber en un escalera comunitaria, local, habitación, etc… En esta ocasión vamos a usar un sensor PIR de movimiento y otro sensor de luz. (los puedes ver en el catálogo de MK Electrónica)-

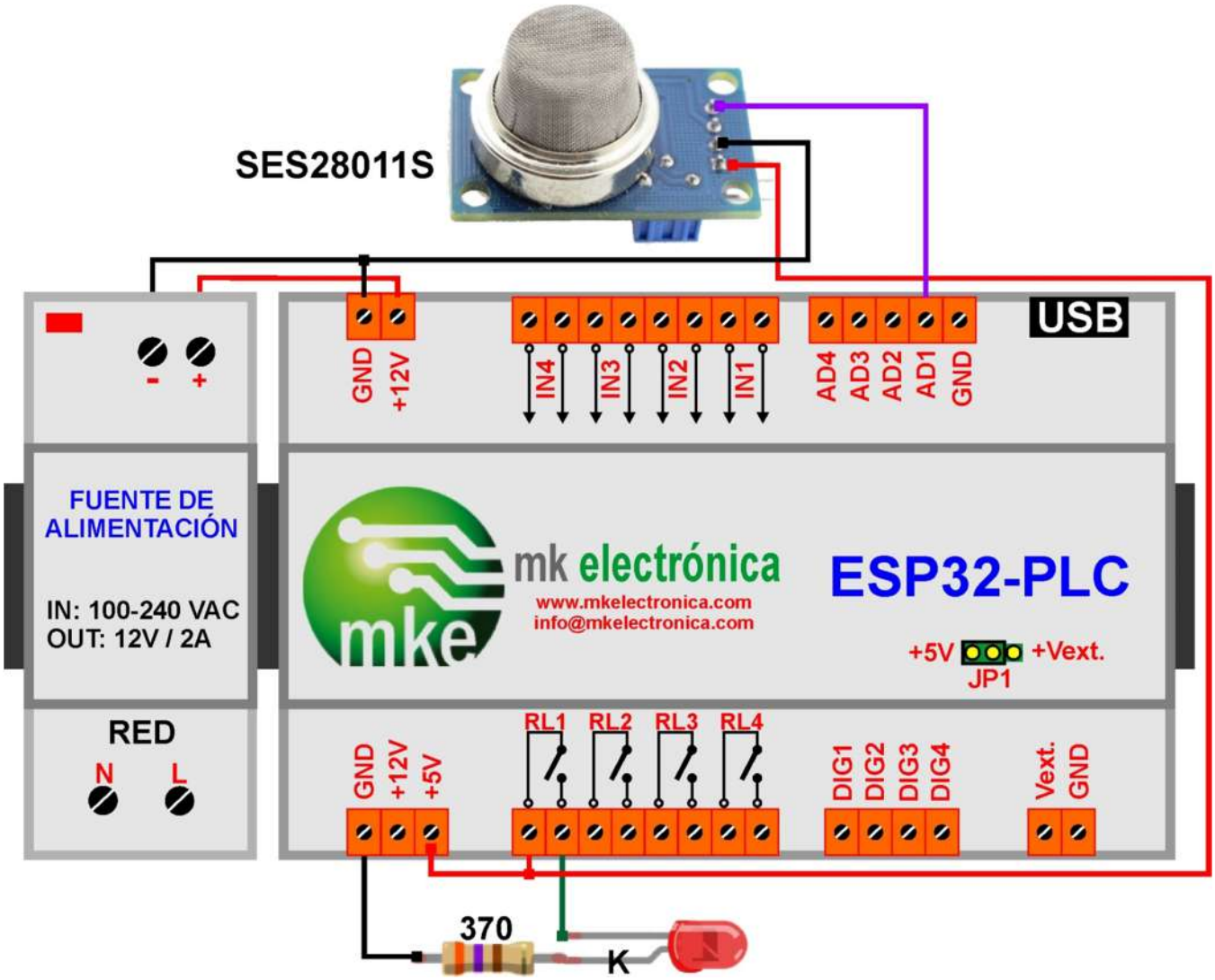


El sensor de movimiento CT0057PIR se conecta con la entrada optoacoplada IN1 mientras que el sensor de luz CT0013LS se conecta con la entrada AD1 en el modo analógico.

Cuando se detecte un movimiento y la luz ambiente esté por debajo de un umbral, se activa el relé RL1 durante un tiempo. Los contactos de este relé se pueden utilizar para conectar un sistema de alumbrado (p.e. el led).

5-1-12 Alarma Gas

En este ejemplo vamos a usar un sensor de gas que detectará la concentración de butano/propano. Cuando se supera un determinado umbral se produce una señal sonora de alarma y se activa el relé RL1 que pone en marcha un sistema de ventilación (simulamos con un led).

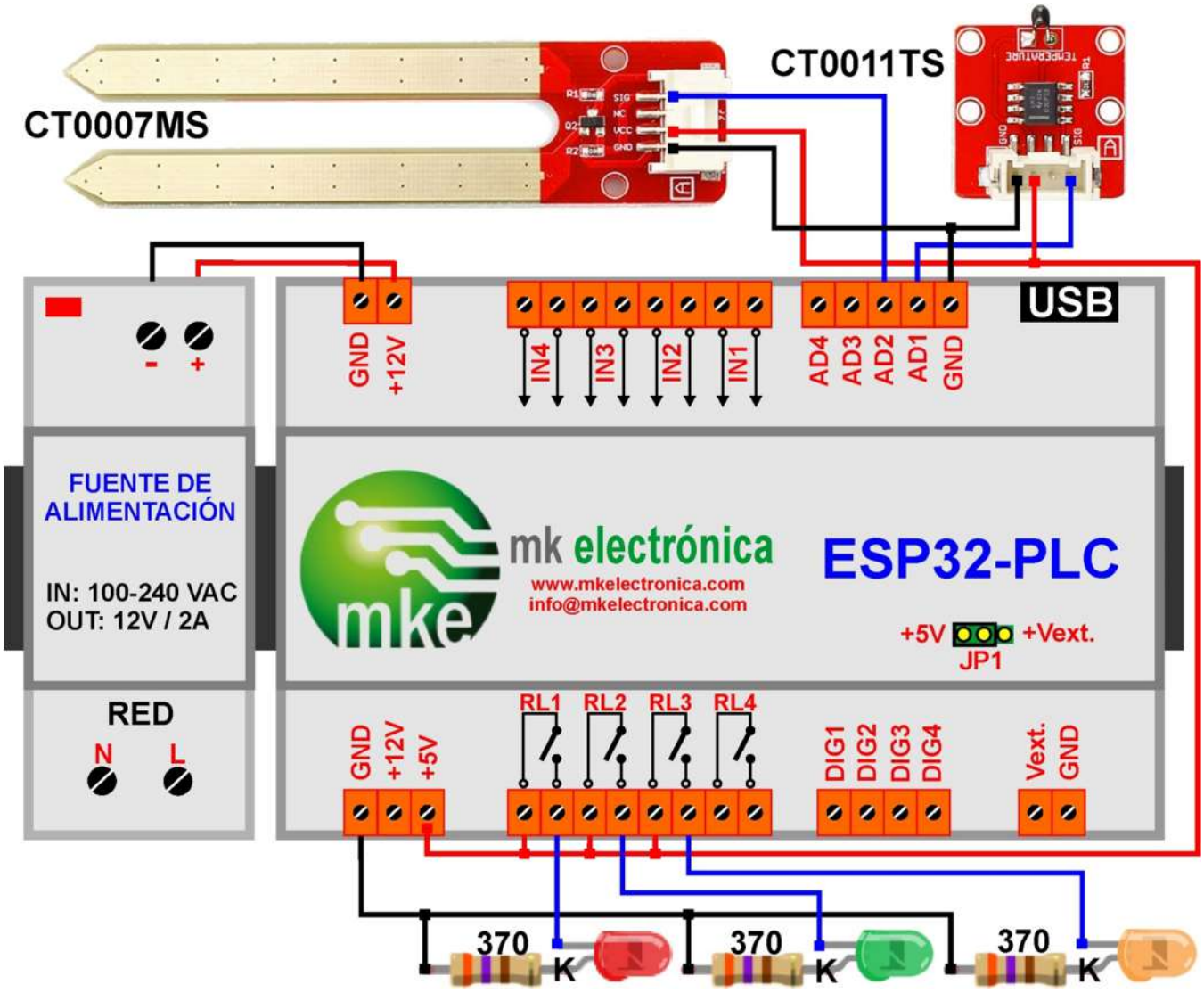


Nuevamente hacemos uso de la multitarea. En esta ocasión el programa principal toma muestras del sensor y comprueba si la concentración excede o no del umbral establecido. Si así fuera se habilita la tarea **alarma** que ejecuta la función **alarmaON()** encargada de generar una señal acústica intermitente. Al "*mismo tiempo*", aún con la alarma sonando, el programa principal sigue tomando muestras de la concentración. Cuando esté por debajo del valor permitido la tarea **alarma** se suspende y la señal acústica desaparece.

El sensor SES28011S está basado en la cápsula MQ2. Se emplea el fichero auxiliar **MQ2.hpp** donde una serie de funciones desarrolladas por www.sandboxelectronics.com proporcionan la concentración de diferentes tipos de gases a partir de los datos y gráficas del fabricante de la cápsula MQ2.

5-1-13 Invernadero

Vamos con el último de esta serie de ejemplos básicos. En esta ocasión se trata de emular el control de un invernadero. Para ello disponemos de un sensor de temperatura y otro que mide la humedad del suelo. Se conectan con las entradas AD1 y AD2 en el modo analógico respectivamente.



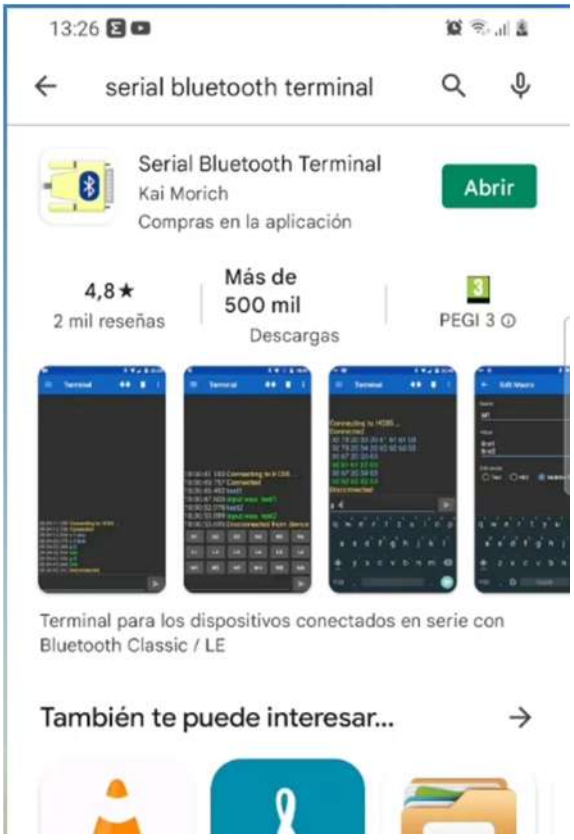
- Cuando la temperatura está por debajo de un determinado umbral se activa el relé RL11 que pone en marcha un sistema calefactor (led rojo).
- Cuando la temperatura supera un umbral máximo se activa el relé RL2 que pone en marcha un sistema refrigerador (led verde).
- Cuando la humedad del suelo está por debajo de un umbral se activa el relé RL3 que pone en marcha un sistema de regadío (led ámbar).

A estas alturas se trata de un ejemplo muy sencillo que simplemente nos abre la puerta a más aplicaciones.

5-2 Ejemplos con Bluetooth

Ya se ha explicado que el SoC ESP32, y por tanto nuestro ESP32-PLC, tiene conectividad Bluetooth. Esto permitirá comunicarnos por radio frecuencia (RF) con cualquier dispositivo que emplee esta misma tecnología: Smartphone's, tablet's, PC's, etc…

El objetivo de este manual es simplemente proponer una serie de ejemplos que permitan la comunicación entre el ESP32-PLC y esos dispositivos. Si quieres profundizar sobre la tecnología Bluetooth, en la red tienes información técnica, normativas, aplicaciones comerciales, etc… También hay tutoriales y cursos que tratan el tema como es el caso del "Curso online de comunicaciones con Arduino" que se imparte en el Campus Tecnológico. (https://cursocomunicaciones.es/)

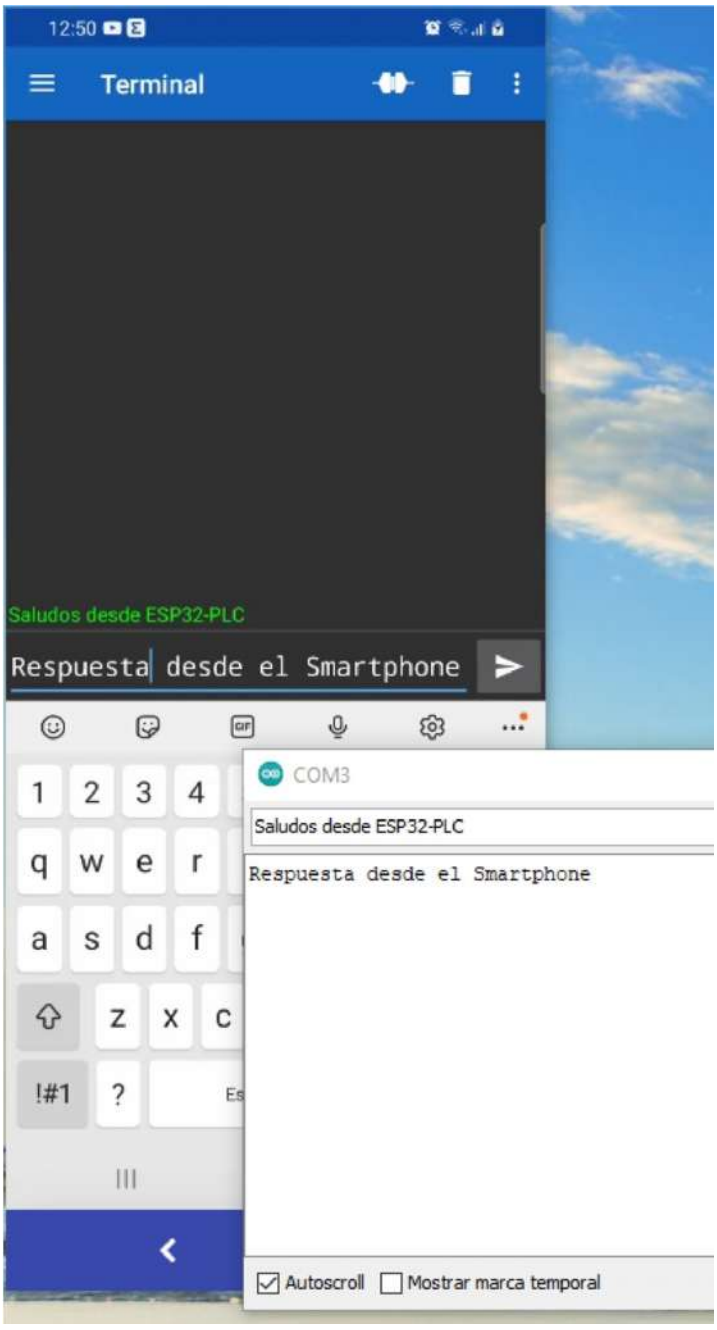


Para estos ejemplos hemos usado un Smartphone dotado de Bluetooth y al que le hemos instalado la aplicación gratuita "**Serial Bluetooth Terminal**" que puedes descargar desde Play Store. Nos gusta mucho y funciona muy bien, pero hay muchas más. Tú eliges.

En los ejemplos, a nuestro ESP32-PLC, le hemos denominado "**MK-ELECTRONICA**". Puedes (debes) cambiarle el nombre. Como harías con cualquier otro dispositivo, tu Smartphone debe localizarlo de entre todos aquellos que estén dentro de su radio de acción y luego vincularlo. Ahora puedes abrir la app "**Serial Bluetooth Terminal**" o la que hayas elegido.

5-2-1 Comunicación BT

Es el más sencillo. Realiza una conexión entre la app de comunicaciones en el Smartphone remoto y el ESP32-PLC. Lo que se envía desde el primero se visualiza en el monitor serie del IDE y viceversa.



Empleamos la librería "**BluetoothSerial.h**" cuyas funciones facilitan enormemente la conexión y comunicación con cualquier dispositivo remoto. Se instaló durante el proceso de instalación que vimos en el apartado 4. En estos ejemplos usaremos las funciones más importantes, no obstante sería interesante que buscaras toda la información disponible relativa a la misma y sus funciones.

Desde el IDE escribimos el mensaje "*Saludos desde ESP32-PLC*" que se recibe en el móvil. Desde este escribimos el mensaje "*Respuesta desde el Smartphone*" que se recibe en el IDE.

Observa que hay una comunicación bidireccional e **INALAMBRICA** entre el dispositivo remoto, el Smartphone, y el ESP32-PLC a través del monitor del IDE.

5-2-2 Control Relés

A partir de aquí todas las posibles aplicaciones y proyectos dependen ya de tu imaginación y buen hacer. Esperamos que estos ejemplos, y los anteriores, te sirvan de modelo e inspiración.

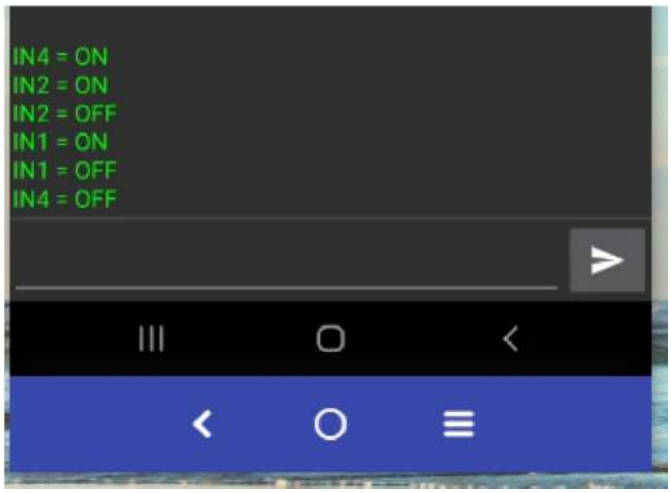
En este caso vamos a usar el dispositivo remoto (p.e. el móvil) para controlar de forma remota e inalámbrica los relés de ESP32-PLC. Para ello vamos a enviar 4 comandos muy sencillos que harán cambiar de estado a los relés correspondientes y también el zumbador: "RL1", "RL2", "RL3", "RL4" y "BUZ".

Una de las cosas que me gusta de la App "**Serial Bluetooth Terminal**" es que además de enviar cadenas que escribimos desde el teclado, como hicimos en el ejemplo anterior, también tenemos la posibilidad de crear botones cuya pulsación desencadena el envío de la cadena que desees. La verdad, resulta muy cómodo. Te recomiendo que instales esta aplicación y estudies todas sus posibilidades.

El programa en el ESP32-PLC recibe la cadena (el comando) hasta que le llega el código de retorno de carro o CR ('\r'). Asegúrate de que dicho código también se envía desde el móvil (revisa la configuración de la App). La cadena recibida se va explorando en busca de cada uno de los comandos posibles para actuar en consecuencia.

5-2-3 Entradas optoacopladas

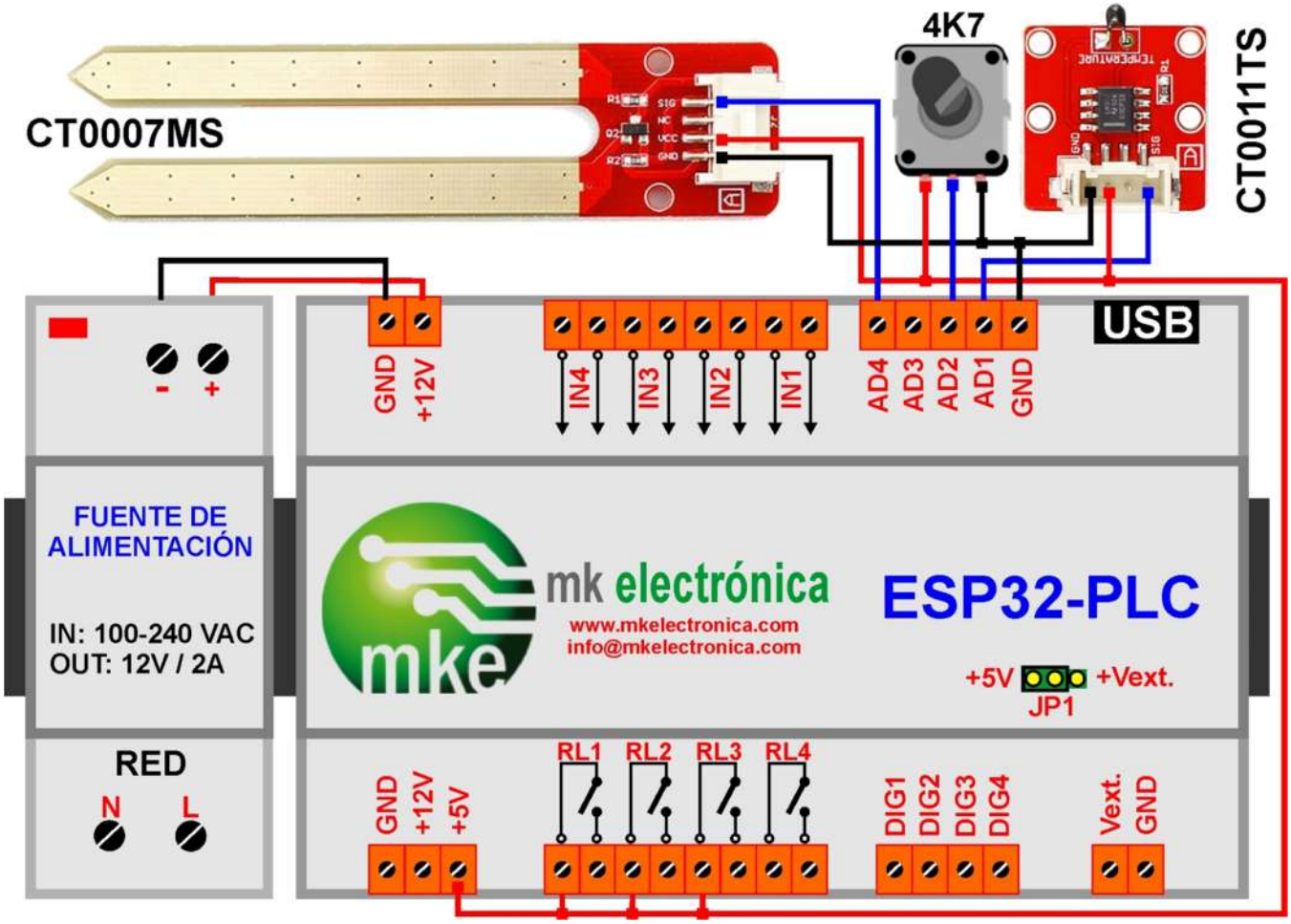
Además de controlar de forma remota las salidas a los relés del ESP32-PLC, también es posible monitorizar remotamente el estado de sus entradas. Es lo que hacemos en este ejemplo. Vamos a monitorizar el estado de las entradas optoacopladas IN1-IN4 sobre la pantalla del dispositivo remoto, que en nuestro caso es un móvil al que le hemos instalado la aplicación "**Serial Bluetooth Terminal**".



Cada vez que una de esas entradas cambia de estado, se envía al dispositivo remoto la cadena "*ON*" u "*OFF*" según esté a nivel "1" o a nivel "0" .

5-2-4 Entradas analógicas

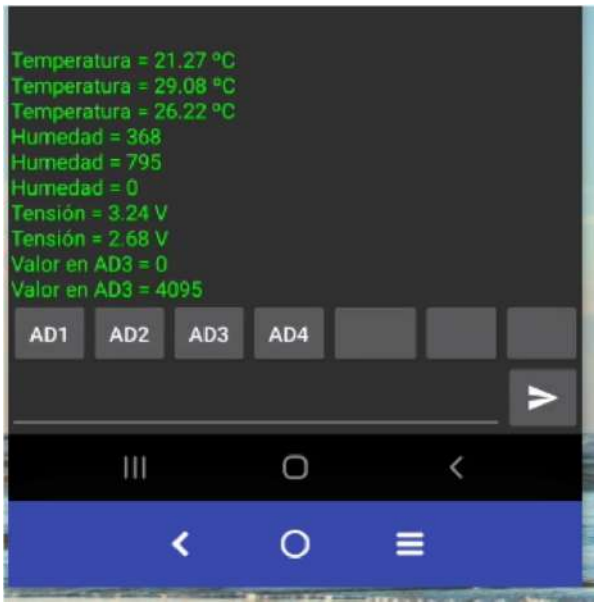
Lo mismo podemos hacer con las entradas analógicas a las que le hemos conectado, como en ejemplos anteriores, una serie de sensores analógicos: sensor de humedad del suelo, potenciómetro y sensor de temperatura.



Desde el dispositivo remoto (el móvil) enviamos cadenas con los comandos "AD1", "AD2", "AD3" y "AD4". Recuerda que las cadenas deben acabar con CR ('\r') y que debes realizar la configuración apropiada en la App "**Serial Bluetooth Terminal**".

Nuestro ESP32-PLC recibe el comando y lo procesa. Si se recibe…

- "**AD1**": Se lee a entrada analógica AD1, se obtiene el valor de la temperatura y se transmite el mensaje "*Temperatura = XX ºC*".
- "**AD2**": Se lee la entrada analógica AD2, calcula la tensión que produce el potenciómetro y transmite el mensaje "*Tensión = XX V*".



- "**AD3**": Se lee la entrada analógica AD3 y se muestra el mensaje con el valor actual "*Valor en AD3: XXXX*".
- "**AD4**": Se lee la entrada analógica AD3 donde se conecta el sensor de humedad del suelo. Se transmite el mensaje "*Humedad = XXX*".

Mediante la App "**Serial Bluetooth Terminal**" podemos configurar 4 botones para que envíen la correspondiente cadena o comando cada vez que se accionan.

En la pantalla vemos las respuestas enviadas desde el ESP32-PLC

5-2-5 Salidas digitales

Como no puede ser menos también podremos controla de forma remota, desde nuestro móvil, las salidas digitales DIG1-DIG4. Por ellas, mediantes sus correspondientes comandos, podremos sacar niveles lógicos "1" o "0" o bien señales PWM con un determinado porcentaje de ciclo útil.

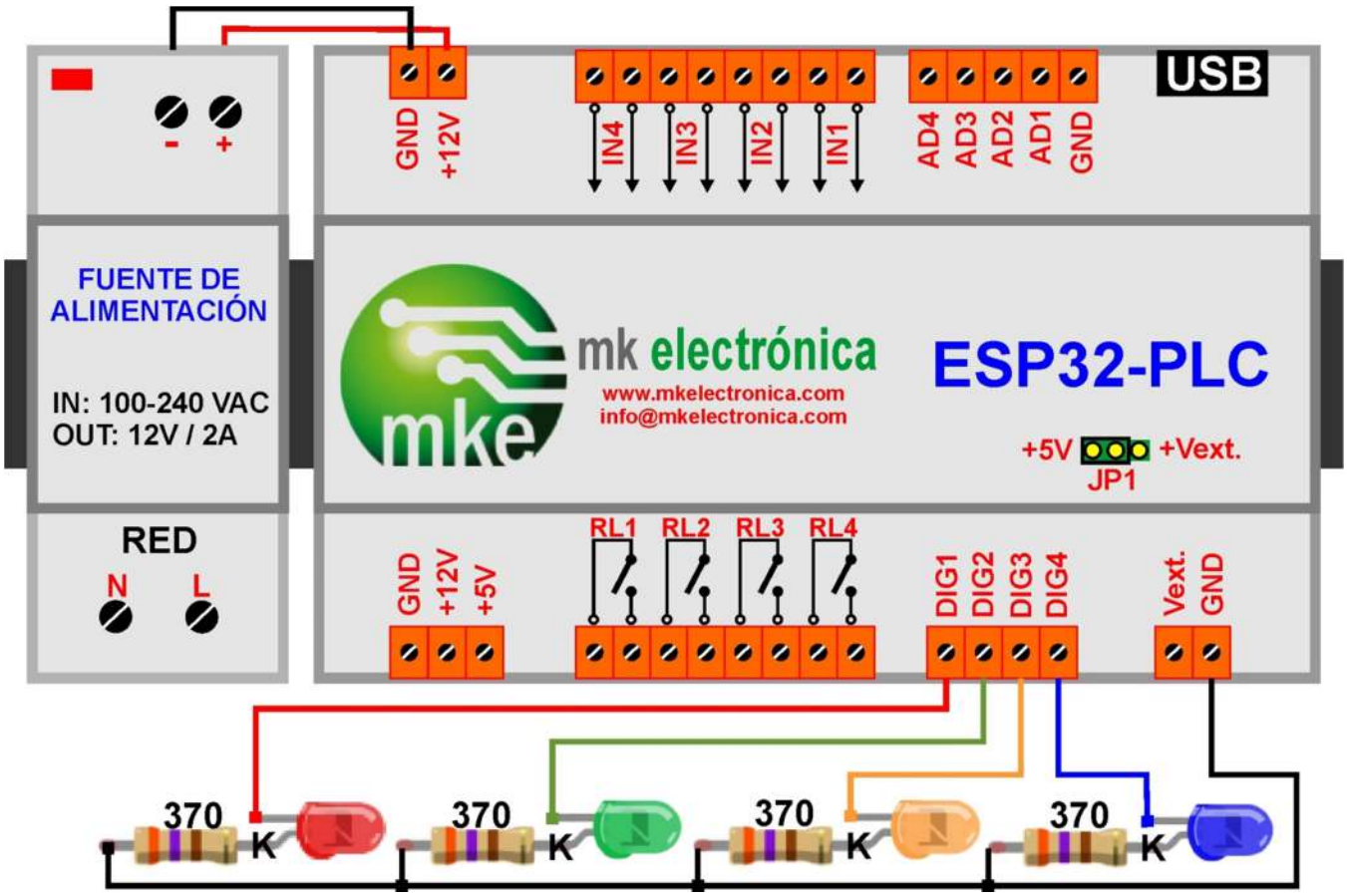
Disponemos de cuatro comandos con un parámetro '*p*' separado por ',' que son:

- "**D1,p**" : Para actuar sobre la salida DIG1.
- "**D2,p**" : Para actuar sobre la salida DIG2.
- "**D3,p**" : Para actuar sobre la salida DIG3.
- "**D4,p**" : Para actuar sobre la salida DIG4.

Con el parámetro '**p**' establecemos qué queremos sacar por cada una de las salidas: **H** = nivel "1", **L** = nivel "0" y **0-100** es el porcentaje del ciclo útil de la señal PWM. Por ejemplo:

- **D1,L**: Pone a nivel "0" la salida DIG1
- **D3,H**: Pone a nivel "1" la salida DIG3
- **D4,85**: Señal PWM al 85% por DIG4
- **D2,10**: Señal PWM al 10% por DIG2

Para comprobar el funcionamiento del ejemplo hemos conectado cuatro sencillos diodos led sobre esas salidas. Los podremos activar, desactivar o regular la intensidad de su brillo.

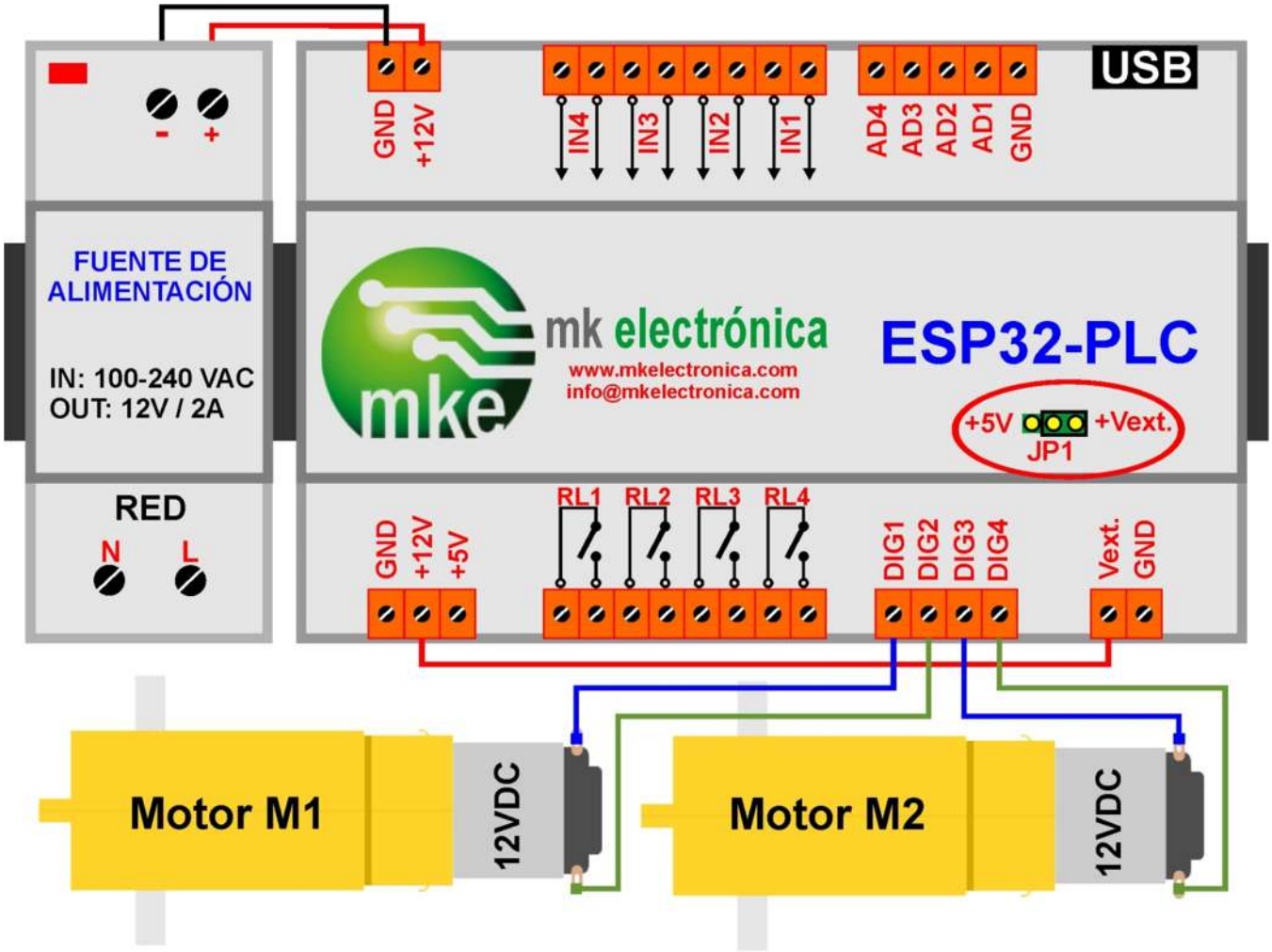


En el ejemplo destacamos la función **recibe()** que hemos guardado en el archivo auxiliar "*recibir.h*" para que el programa principal esté más despejado. Esta función espera a recibir desde el móvil remoto una cadena que finaliza con ('\r'). De esa cadena se extrae el comando propiamente dicho y el parámetro que lo acompaña.

Con ese parámetro se establece si la salida referida es de tipo digital y el valor de la misma ("1" o "0"), o si es de tipo PWM y el porcentaje del ciclo útil de la señal (0-100%). Con estos datos el programa principal actúa directamente sobre la salida oportuna y retransmite al dispositivo remoto la acción recién ejecutada.

5-2-6 Motores DC por PWM

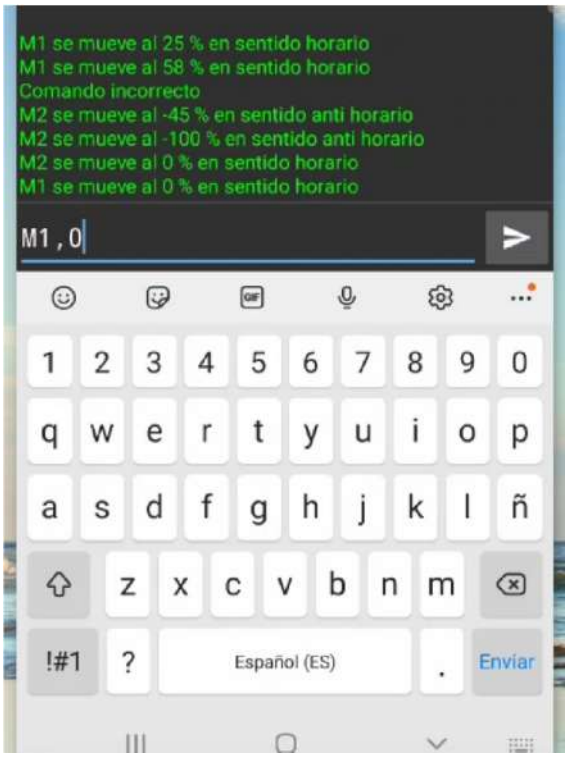
El ejemplo anterior nos sirve como base para este nuevo ejemplo que nos permitirá controlar mediante PWM la velocidad y el sentido de giro de dos motores DC.



En nuestro montaje hemos empleado motores de 12VDC que se aplican por VEXT y se toman desde la salida +12V del ESP32-PLC. No olvides configurar el Jumper JP1 .

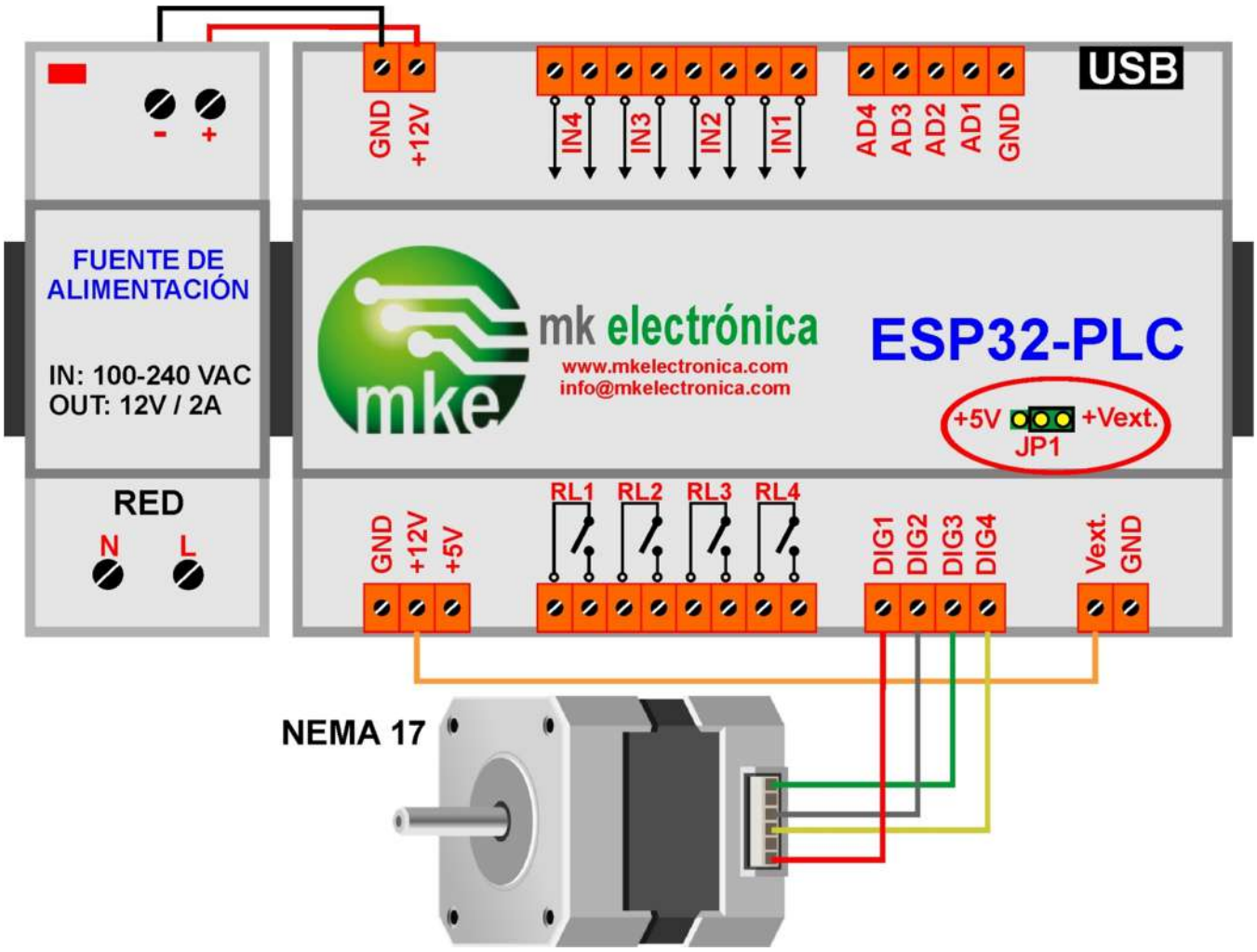
En este caso empleamos dos únicos comandos acompañado de sus respectivos parámetros separados por ',': "**M1,p**" y "**M2,p**". Donde 'p' es el parámetro con el que indicamos el porcentaje del ciclo útil. Está comprendido entre 0 y 100 % y con su signo establecemos el sentido de giro.

ESP32-PLC retransmite al dispositivo móvil remoto el movimiento que se está produciendo, como puedes ver en la imagen.



5-2-7 Motor PAP

De forma parecida podemos controlar de forma remota la velocidad, el sentido de giro y el desplazamiento que debe realizar un motor paso a paso (PAP) bipolar que conectamos en el ESP32-PLC como se muestra en la figura.



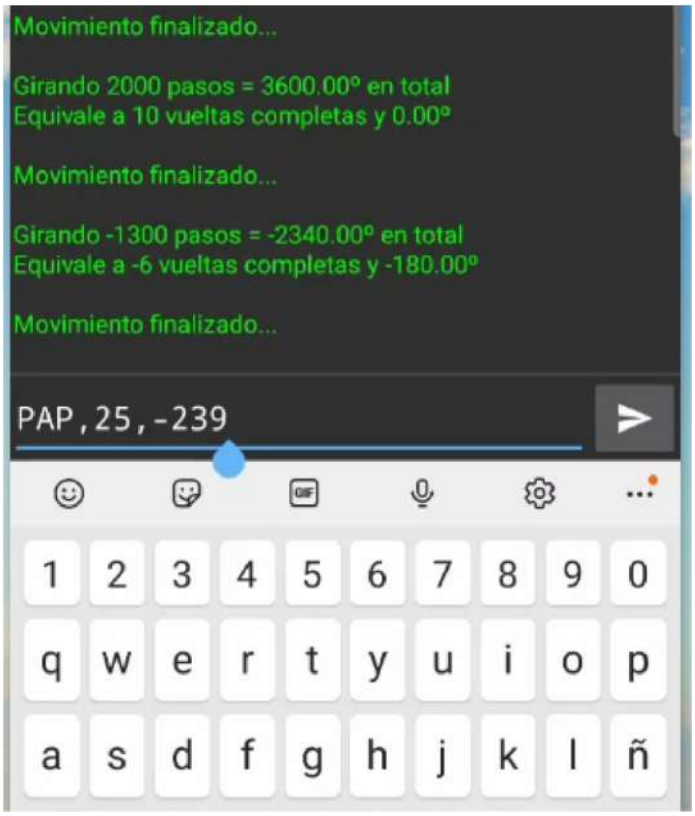
El comando que debemos enviar desde el móvil es "**PAP,p1,p2**", donde…

- **PAP**: es el comando propiamente dicho.
- **p1**: Representa la velocidad de giro en rpm. Depende del modelo empleado, pero en nuestro caso, un NEMA17, no es aconsejable una velocidad mayor de 120 rpm.
- **p2**: Representa el número de pasos que debe girar el motor. Se trata de un número entero con signo de un rango -32768 a 32767 según sea el sentido de giro. Por ejemplo: "**PAP, 30,-23769**" o "**PAP, 105,11546**".

La idea es muy parecida a la de los dos ejemplos anteriores. En este caso la función **recibe()** contenida en el archivo auxiliar "*recibir.h*", que espera a recibir una cadena compuesta del comando "PAP" y los dos parámetros separados por ','. La cadena debe finalizar con ('\r').

El programa principal o **loop()** hace uso de las funciones contenidas en la librería oficial de Arduino "*Stepper.h*". Con ellas establecemos las conexiones del motor PAP, así como la velocidad y el Nº de pasos según el comando recibido.

El ESP32-PLC retransmite al dispositivo remoto un informe con el movimiento realizado.



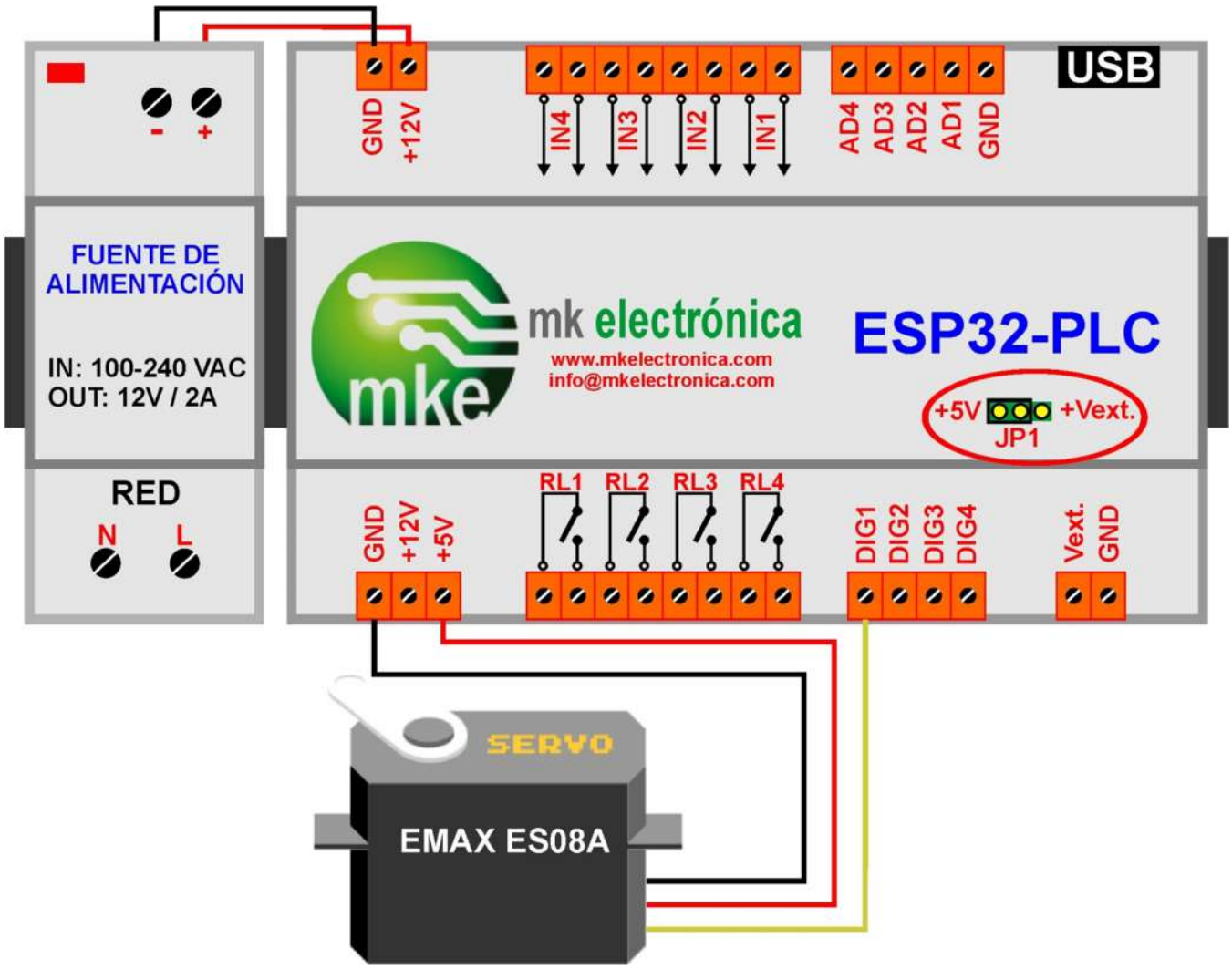
5-2-8 Servomotor

Siguiendo la línea de los ejemplos anteriores, ahora vamos a controlar de forma remota el posicionamiento de un servomotor. Se usa la librería "ESP32Servo.h" empleada en ejemplos anteriores.

En este caso tenemos el comando "**S,p**", donde '*p*' es el parámetro que indica los grados en los que el servo se debe posicionar (0º - 180º).

El ESP32-PLC retransmite al dispositivo remoto la posición actual en la que se encuentra el eje del servomotor.

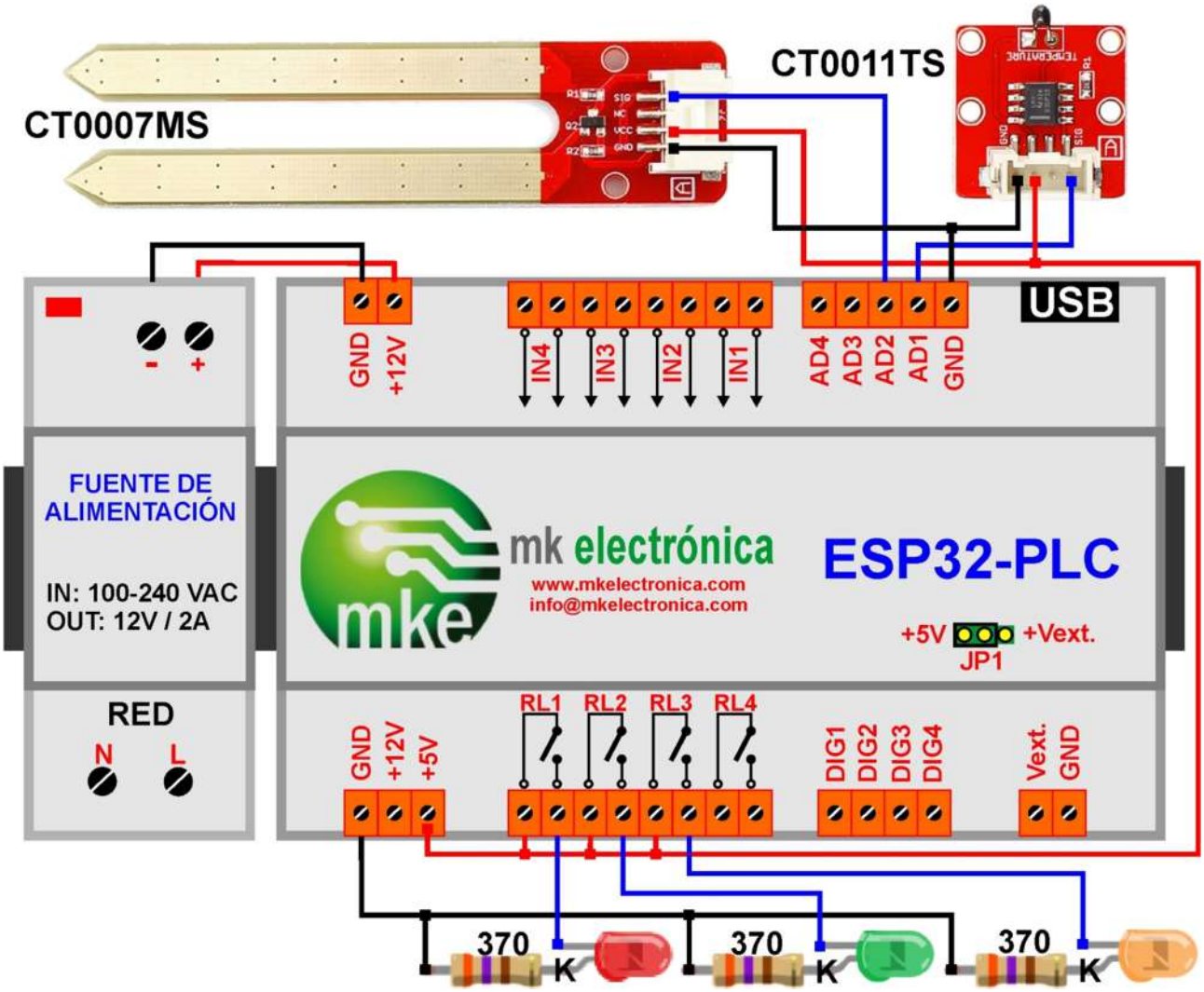
En nuestro montaje hemos empleado un mini servomotor EMAXES08A controlado desde la salida DIG1 y alimentado a +5V. Si fuera necesario configura el JP1 a la posición de +5V ya que en los ejemplos anteriores lo pusimos en la posición de VEXT.

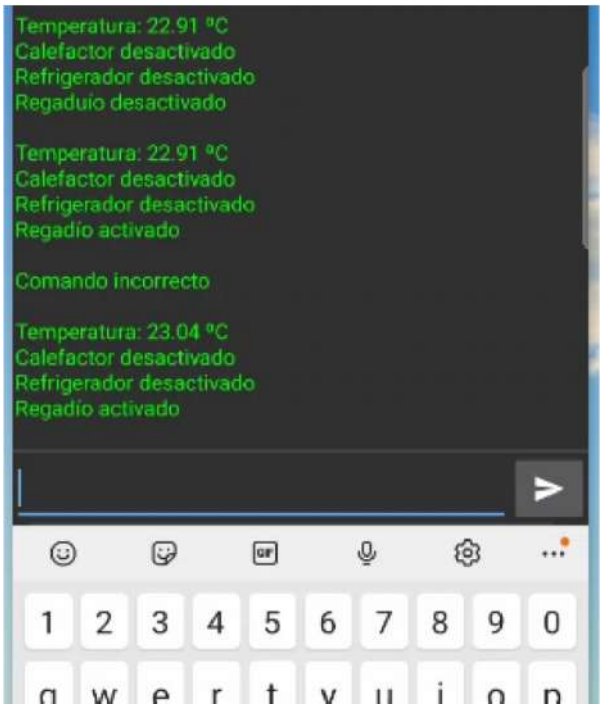


5-2-9 Invernadero

Volvemos a nuestro conocido invernadero, pero ahora vamos a tener la posibilidad, gracias a la comunicación bluetooth, de conocer de forma remota tanto la humedad del suelo como la temperatura.

El programa principal se limita a controlar constantemente el invernadero: lee los valores de temperatura y humedad, los compara con los umbrales establecidos y actúa sobre los relés que controlan los sistemas de calefacción, refrigeración, y regadío. En el esquema los contactos de los relés activan los leds rojo, verde y ámbar respectivamente. En este sentido es muy parecido al ejemplo 5-1-13.



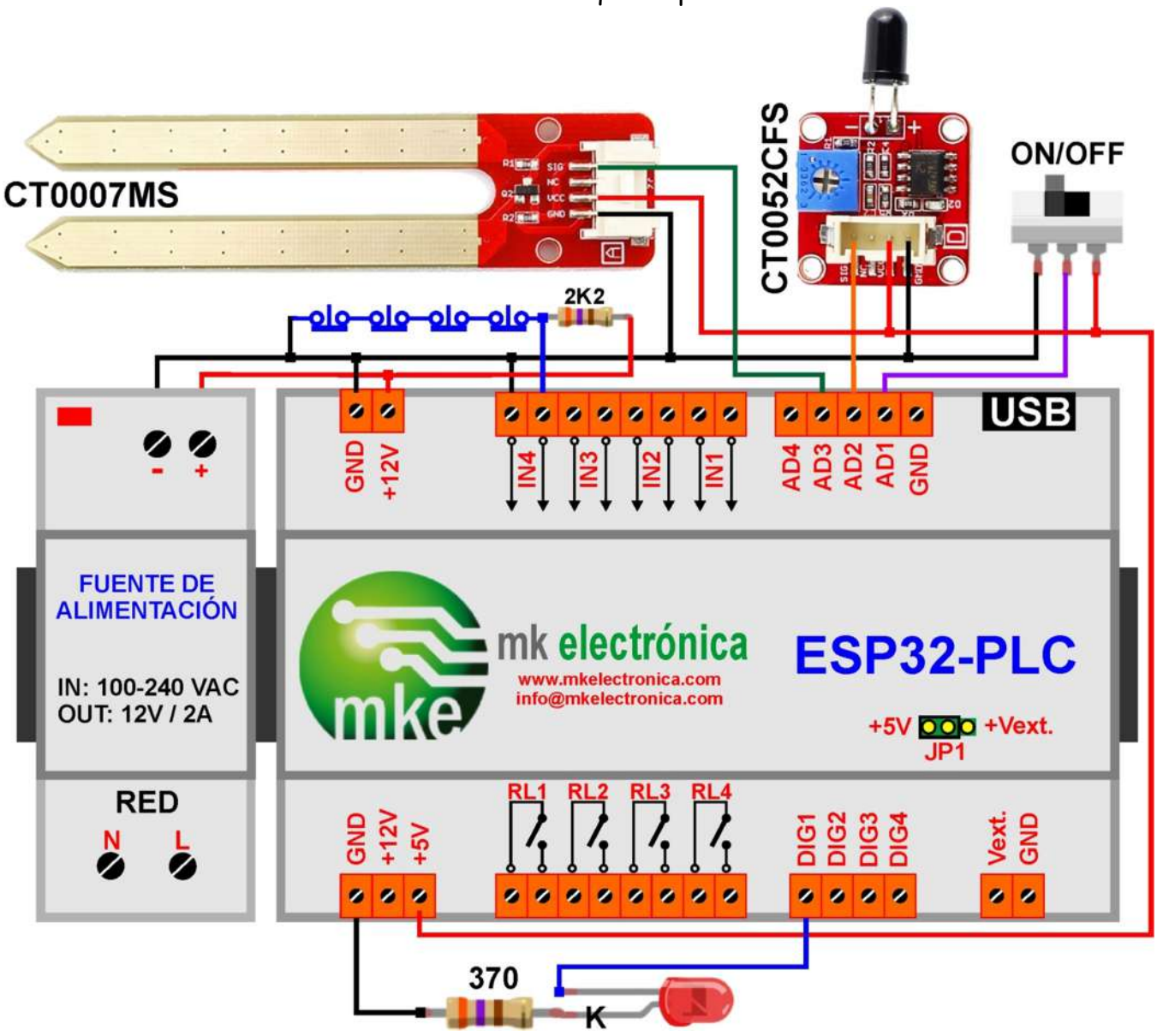


La diferencia consiste en el empleo de la multitarea. Mientras el programa principal se ejecuta sin interrupción, al "*mismo*" tiempo se está ejecutando la tarea **Rx**. Está contenida en el fichero auxiliar "*recibir.hpp*" y formada por la función "**recibe()**".

En cualquier momento puede recibir desde el dispositivo remoto el comando '**?**'. El ESP32-PLC responde con el valor de la temperatura, y el estado actual de los sistemas de calefacción, refrigeración y regadío.

5-2-10 Alarma

Vamos a ver otra versión de una alarma que emplea también la multitarea.



En esta ocasión tenemos…

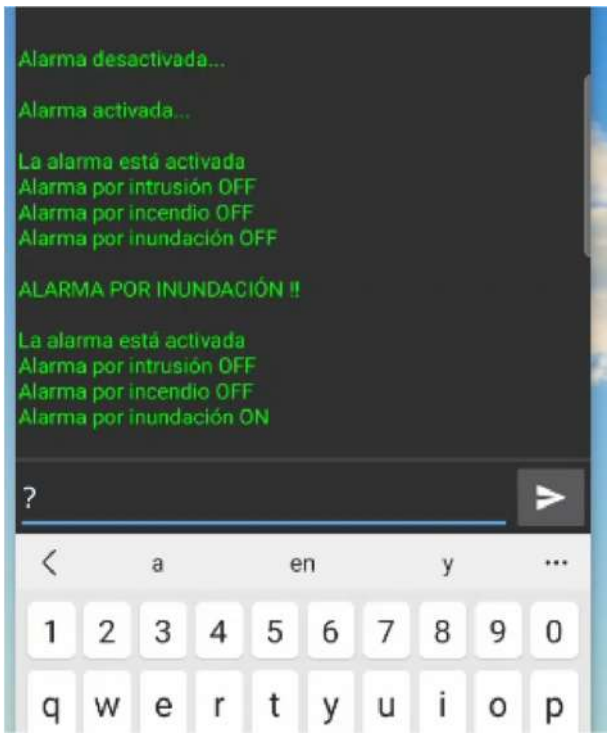
- Una zona vigilada mediante pulsadores normalmente cerrados conectados en serie con la entrada optoacoplada IN4. Cuando se abre **cualquiera** de ellas se dispara la alarma por intrusión y el zumbador emite una señal sonora intermitente.
- Un sensor de llama o incendio. Se conecta con la entrada AD2 del ESP32-PLC y cuando se detecta un incendio se activa el relé RL1 para poner en marcha un sistema de rociadores de espuma.

- Un sensor de humedad conectado con AD3 detecta si la humedad sobrepasa un determinado umbral. En caso de inundación se activa el relé RL2que pone en marcha las bombas de achique.
- Un conmutador conectado en AD1 permite activar/desactivar la alarma de forma local. También podremos hacerlo de forma remota.
- Un led conectado en la salida DIG1 parpadea cuando la alarma está activada y en funcionamiento en modo vigilancia.

También emplea la multitarea y para ello se han creado cuatro tareas que se ejecutan en paralelo "*casi*" al mismo tiempo:

- **Rx:** Está contenida en el fichero auxiliar "*recibir.hpp*" y ejecuta la función **recibe()** que, en cualquier momento, puede recibir uno de estos dos comandos seguidos de ('\r'):
    - "**s**": activa o desactiva la alarma desde el dispositivo móvil remoto.
    - "**?**": Se solicita el estado de la alarma. El ESP32-PLC devuelve el estado actual de la alarma: "*activada*", "*desactivada*", "*Alarma por intrusión*", "*Alarma por incendio*" y "*Alarma por inundación*".

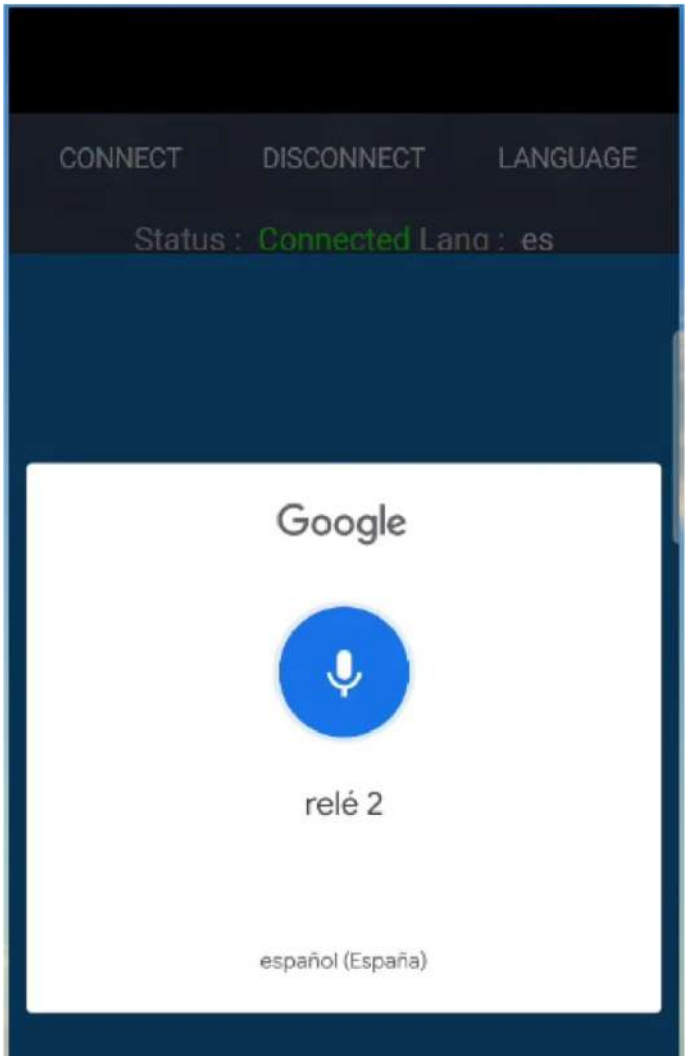
- **testeo**: Está contenida en el fichero "*Tarea_testeo.hpp*" y ejecuta la función **testAD1()**. Comprueba si la alarma está accionada bien mediante el conmutador local en AD1 o bien remotamente mediante el comando "**s**". Si así fuera se habilita la tarea "**armar**". En caso contrario se desactivan todas las salidas y las tareas "**armar**" y "**testAlarma**".
- **armar**: Está contenida en el fichero auxiliar "*Tarea_armar.hpp*" y ejecuta la función **armarAlarma()**. Se habilita la tarea "**testAlarma**", se genera una señal sonora de pre aviso en el zumbador y el led conectado en DIG1 se pone en modo intermitente. La alarma está armada y vigilante.
- **testAlarma**: Está contenida en el fichero auxiliar "*Tarea_testAlarma.hpp*" y ejecuta la función **testSensores()**. Comprueba constantemente el estado de los sensores conectados: de intrusión en IN4, de fuego en AD2 y de inundación en AD3. Según se



detecte una de esas alarmas…

- **Intrusión**: Envía el mensaje "*ALARMA POR INTRUSIÓN*" al dispositivo remoto y produce una alarma sonora en el zumbador.
- **Incendio**: Envía el mensaje "*ALARMA POR INCENDIO*" al dispositivo remoto activa el relé RL1.
- **Inundación**: Envía el mensaje "*ALARMA POR INUNDACIÓN*" al dispositivo remoto y activa el relé RL2.

5-2-11 Control por Voz



Aquí tienes el último ejemplo dedicado a la comunicación bluetooth de nuestro ESP32-PLC. Quizá te parezca el más espectacular pero es muy sencillo. Proponemos controlar cualquier dispositivo mediante el empleo de comandos dictados por voz.

Para ello es necesario instalar en el Smartphone o en la Tablet la App gratuita "**Arduino Voice Control**", aunque seguramente haya otras muchas similares. Esta App utiliza Google para sintetizar la voz en texto. Necesitas por tanto estar conectado a internet. Convierte una voz en una cadena de caracteres de texto y la transmite vía bluetooth. ¡¡Impresionante !!

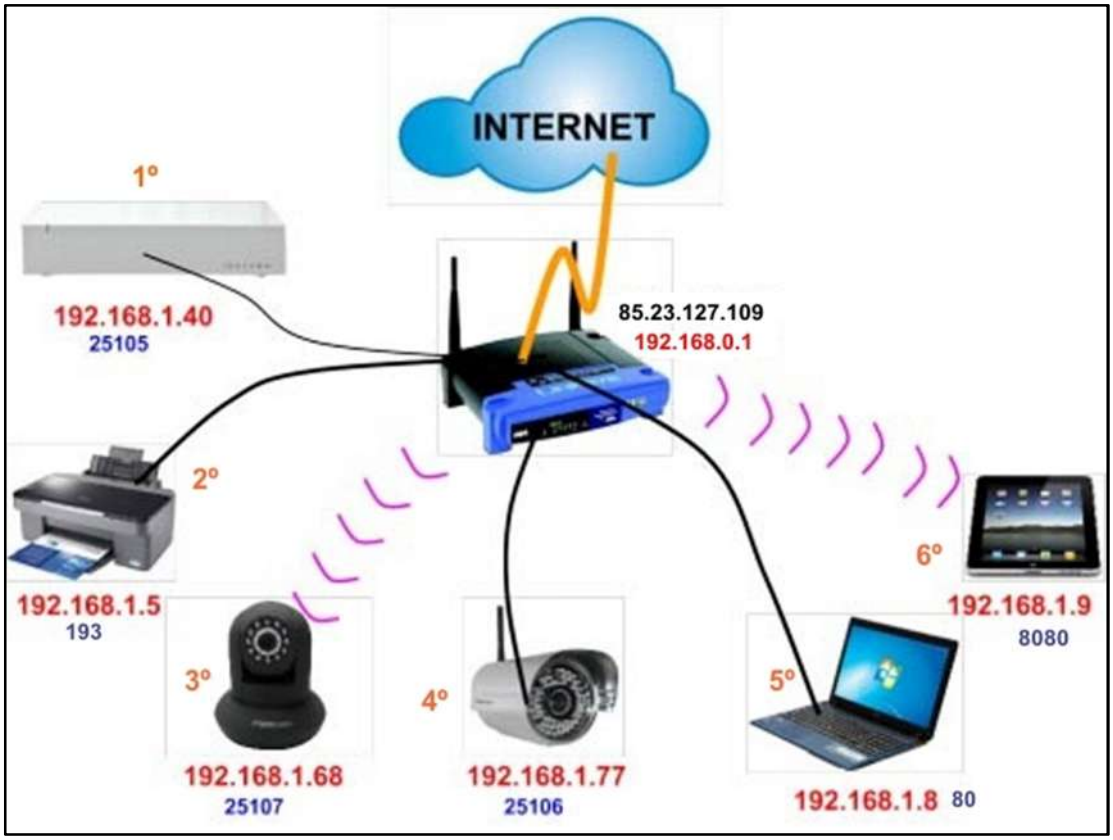
Nuestro ejemplo utiliza las voces "relé 1", "relé 2", "relé 3" y "relé 4" que dictamos en nuestro móvil. Estas se codifican en otras tantas cadenas de texto o comandos y se transmiten vía bluetooth y se reciben en el ESP32-PLC. Aquí se decodifican y se actúa sobre el relé correspondiente. Imagino que esto te dará un buen número de ideas para tus propios proyectos y aplicaciones.

5-3 Ejemplos con WiFi

La tecnología WiFi permite la comunicación inalámbrica entre todos aquellos dispositivos que estén conectados a la misma red. Para elegir el dispositivo con el que queremos comunicarnos debemos de conocer su dirección IP, que es una dirección compuesta de 4 bytes separados por '.'. Esta dirección, conocida como "**IP local**", la asigna automáticamente el router y no tiene porqué ser siempre la misma. Depende de la cantidad de dispositivos que vayas conectando /desconectando en tu red local (LAN).

También puedes configurar el router para abrir/cerrar puertos asignados a esas direcciones IP locales de los dispositivos de tu red local. Si abres un puerto podrás acceder a su correspondiente dispositivo desde el exterior de tu red local, desde internet o red WAN.

Para ello necesitas conocer la dirección **IP pública** que tu compañía o proveedor de servicios te ha asignado. Hay páginas especializadas que te permiten conocerla como son www.cualesmiip.com o www.miip.es . Si conoces tu IP pública y el puerto que abriste en tu router para un determinado dispositivo con una determinada IP, podrás acceder a ese dispositivo desde cualquier lugar del mundo. Basta con indicar desde un navegador: "**IP pública : puerto**". (p.e. 85.23.127.109:193 accede a la impresora)



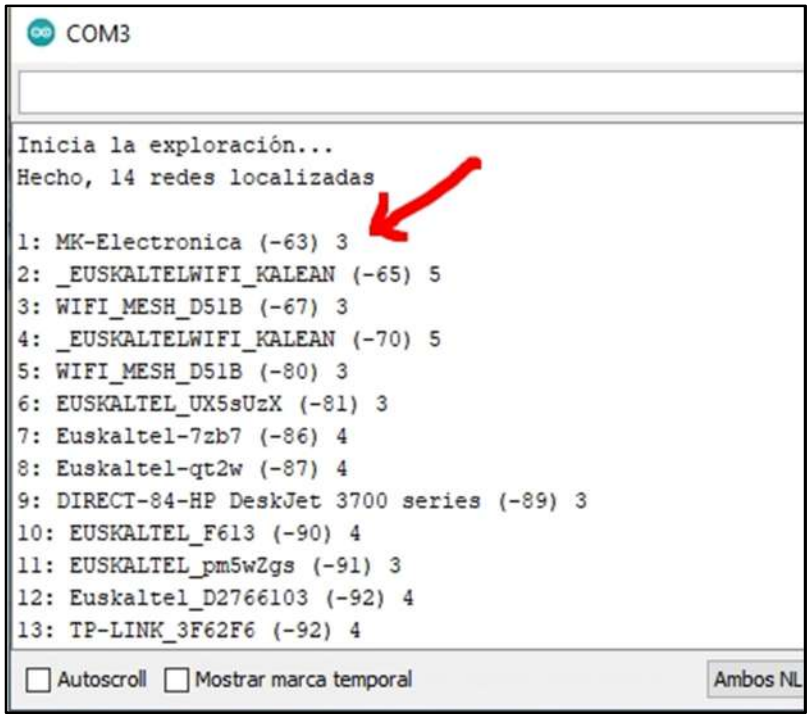
También hay que hablar de la existencia de empresas que proporcionan plataformas que facilitan la interacción o interface entre el usuario y el dispositivo de una forma sencilla, intuitiva y amigable. Algunas de estas empresas son: Thingspeak, Blynk, Adafruit IO, Amazon web, Arduino Cloud IoT, y muchas más.

La mayor parte de ellas tienen una versión limitada pero gratuita que vienen muy bien para familiarizarnos con su empleo. Tendrás que registrarte para poder acceder, obtener información y crear tus propios proyectos. Nosotros, en nuestros ejemplos, hemos usado **Adafruit IO**. Te invitamos a que la pruebes.

Es el principio del **Internet de las Cosas o IoT**, pero lógicamente hay mucho más detrás de todo esto, y este manual no es el lugar apropiado para explicarlo. En la red tienes abundante información, normativas, tutoriales, etc… también hay cursos específicos como los que imparte el Campus Tecnológico en la modalidad online:

- Comunicaciones con Arduino (https://cursocomunicaciones.es/)
- Internet de las Cosas (https://cursointernetdelascosas.es/)

5-3-1 Redes WiFi



Este sería el ejemplo más evidente y sencillo. Localiza y muestra todas las redes WiFi que se encuentren dentro de la cobertura de nuestro ESP32-PLC y su controlador SoC ESP32. De entre todas las redes que aparecen en el lista, se puede apreciar la red WiFi de **MK-Electrónica** que, lógicamente, es la que vamos a usar nosotros.

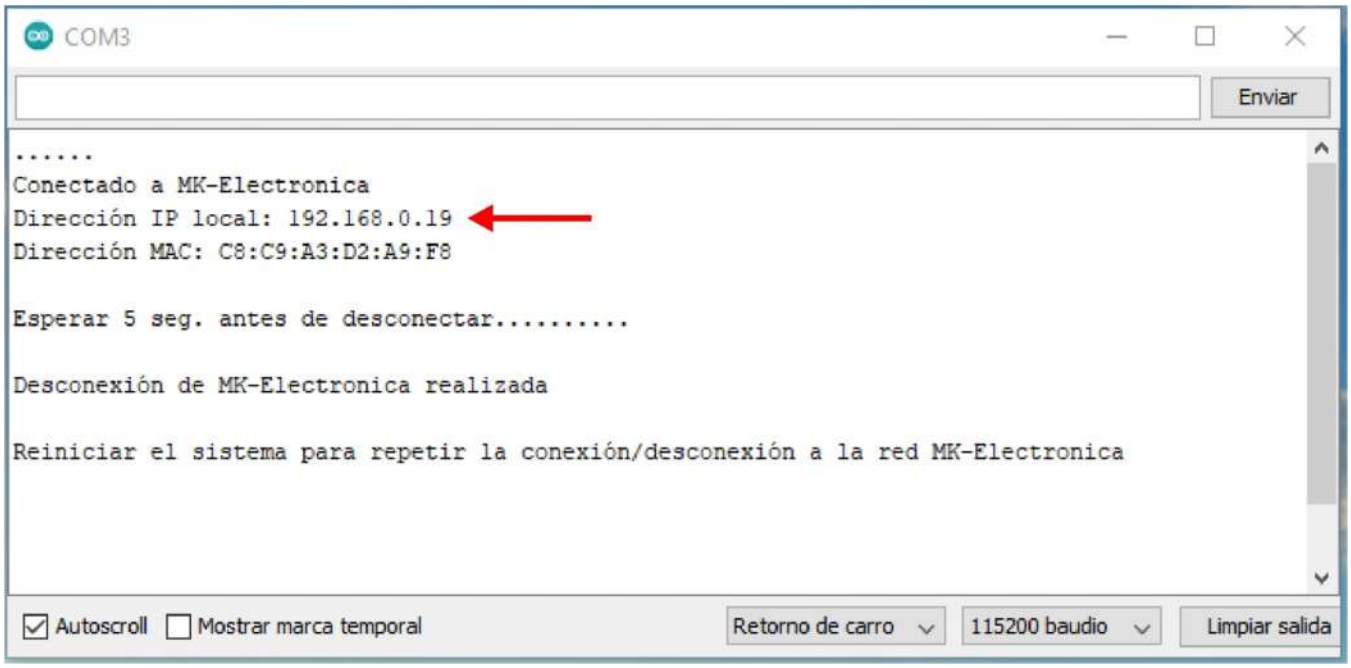
Empleamos la librería "**WiFi.h**" que también se instaló en el proceso de instalación y configuración explicado en el apartado 4. Usaremos las más importantes, pero sería deseable que buscaras información de todas las funciones contenidas en esta librería. Quien sabe lo que puedes encontrar…

5-3-2 Conexión Red WiFi

El ejemplo nos enseña cómo el ESP32-PLC se conecta/desconecta a una red WiFi siempre y cuando conozcamos sus credenciales: nombre (SSID) y clave (PASS). El proceso de conexión dura unos segundos y una vez concluido nos muestra la dirección IP local que le asigna el router. Es la que usaremos cada vez que queramos conectar con el ESP32-PLC. En mi caso ha sido la IP = 192.168.0.19, en tu caso seguro que es otra. **Toma nota de ella**. Si ahora desconectamos el ESP32-PLC de la red y conectamos otro dispositivo, por ejemplo, un móvil, es muy probable que al volver a conectarlo el router le asigna otra dirección diferente. Por eso se suele hablar de dirección IP dinámica.

El ejemplo también nos muestra la dirección MAC del dispositivo. Esta es un código único en el mundo que identifica al dispositivo, en este caso al ESP32 que incorpora el ESP32-PLC. A partir de la MAC ciertos ruters te permiten asignar a ese dispositivo una dirección IP fija que no cambiará aunque conectes/desconectes otros dispositivos a tu red local. Investiga si el tuyo lo permite.

Finalmente, cinco segundos después el ESP32-PLC se desconecta de la red.



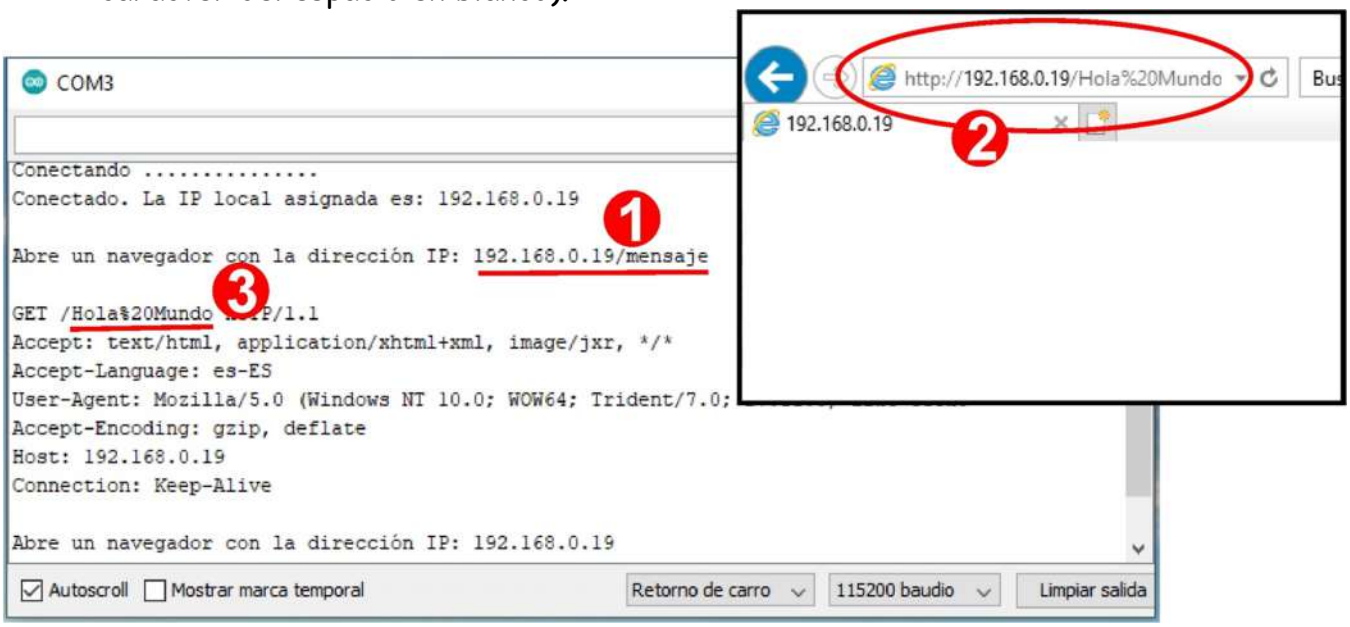
5-3-3 Servidor

En este ejemplo vamos a transferir datos entre un navegador como es el Internet Explorer (el cliente) y nuestro ESP32-PLC (el servidor). Para ello:

1. Se establece la conexión con la red WiFi.
2. Esperamos a recibir datos desde el cliente (el navegador).
3. Vamos recibiendo y visualizando en el monitor del IDE todas las líneas según van llegando.
4. Cuando llega una línea en blanco finaliza la recepción de datos desde el cliente.
5. El servidor (ESP32-PLC) responde entonces con el mensaje "**HTTP/1.1 200 OK**".
6. Se cierra la conexión.

Para probarlo basta con seguir los siguientes pasos:

1. Al ejecutar el programa del ejemplo, el monitor serie del IDE nos muestra la dirección IP que nos asigna el router (**192.168.0.19**).
2. Abrimos un navegador (p.e. Internet Explorer) e indicamos esa IP seguida del mensaje que queremos llegue al servidor ESP32-PLC (**192.168.0.19/Hola mundo**).
3. En el monitor serie se van visualizando todas las líneas que llegan desde el navegador. En la primera de ellas, tras la palabra o método **GET**, aparece el mensaje "*Hola%20Mundo*" enviado desde el navegador (el código %20 corresponde al carácter del espacio en blanco).



Es lo que se conoce como una "**petición cliente → servidor**". Desde un navegador instalado en un PC, Smartphone, Tablet, o similar hemos conseguido mandar un mensaje a ESP32-PLC usando la red WiFi.

Si abrimos un puerto en el router, por ejemplo el 850, lo asociamos a la IP local de nuestro dispositivo ESP32-PLC (p.e. 192.168.0.19), y en el navegador escribimos la dirección pública que nos asigna nuestro proveedor…

xxx.xxx.xxx.xxx:850/Hola mundo

…el mensaje lo podremos mandar desde cualquier lugar del mundo.

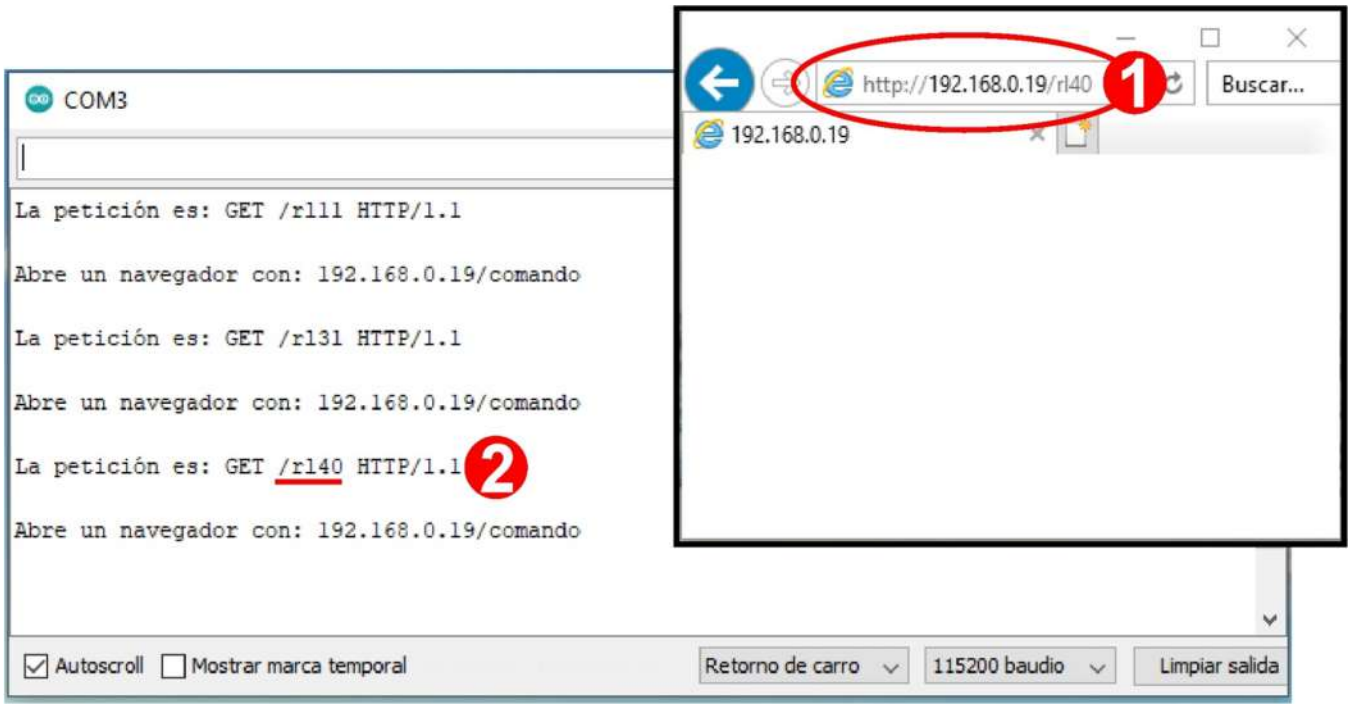
5-3-4 Ejecuta comandos

Si sabemos mandar mensajes desde un navegador (el cliente) y recogerlos en nuestro ESP32-PLC (el servidor), ya podemos empezar a hacer muchas cosas usando la red WiFi y, si procede, la red de redes Internet. En este ejemplo desde el navegador mandaremos 8 comandos que permiten activar/desactivar cada uno de los relés:

- **rl11/rl10** → Relé RL1 ON/OFF
- **rl21/rl20** → Relé RL2 ON/OFF
- **rl31/rl30** → Relé RL3 ON/OFF
- **rl41/rl40** → Relé RL4 ON/OFF

Como hicimos en el ejemplo anterior:

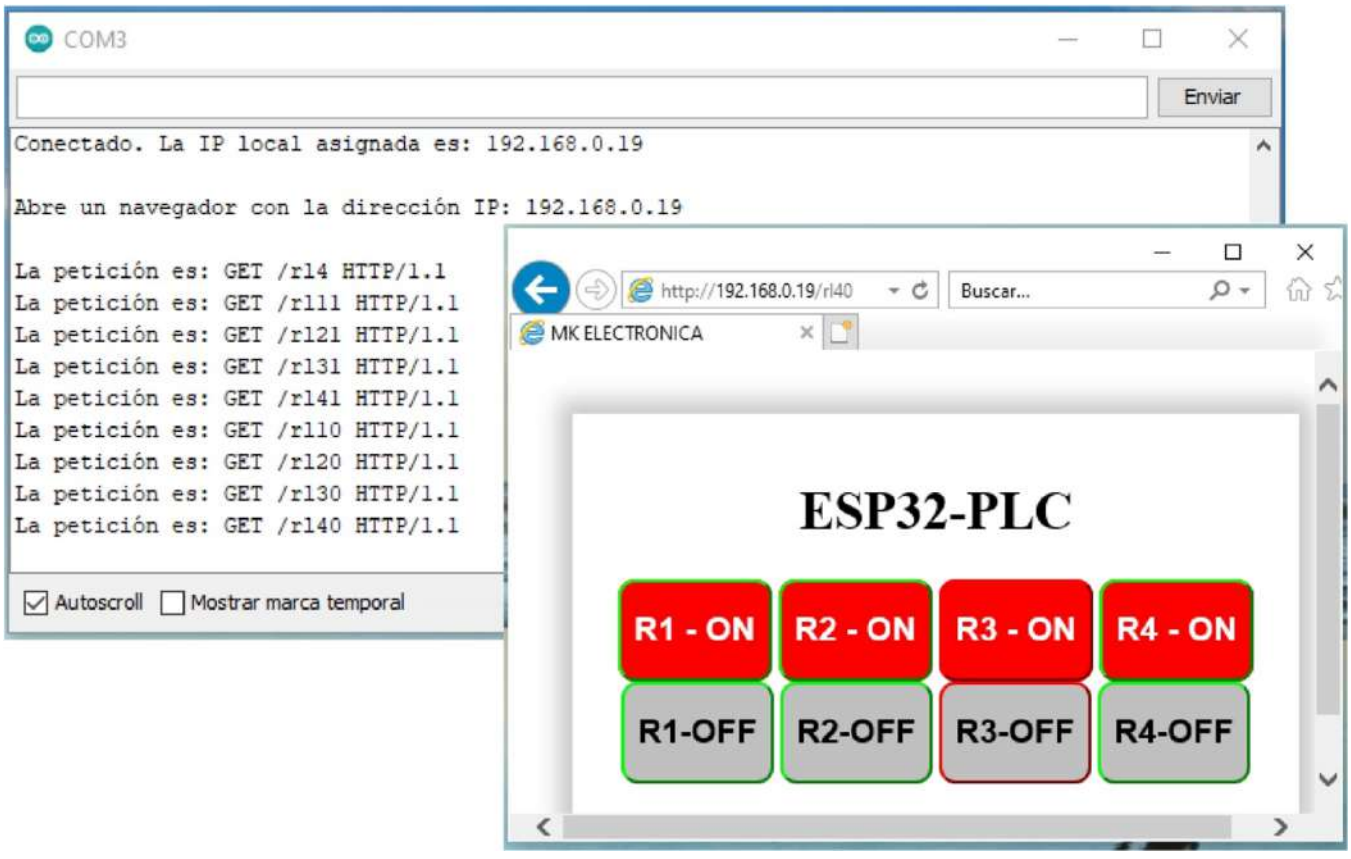
1. Abrimos un navegador e indicamos la dirección IP seguida de uno de esos comandos (**p.e.192.168.0.19/rl40**).
2. El programa en ESP32-PLC (servidor) extrae y visualiza en el monitor serie del IDE la petición realizada desde el navegador (cliente). A partir de ella actúa sobre el relé correspondiente.



5-3-5 Uso de botones

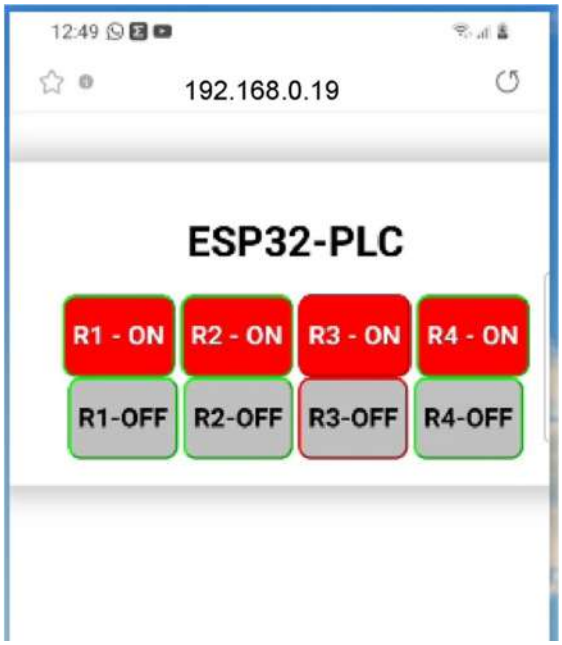
Espero que este ejemplo sí que te resulte cuando menos llamativo. El cliente envía los mismos comandos pulsando los correspondientes botones que le aparecen en el navegador. No es muy diferente al ejemplo anterior. En esta ocasión tenemos el fichero auxiliar "*paginaHTML.hpp*" que contiene la función **paginaHTML()** a la que se llama con cada petición del cliente.

Dicha función transmite al navegador (cliente) una serie de líneas escritas en el lenguaje de programación **HTML**. Este es uno de los lenguajes que se emplean para el desarrollo de páginas web. En este caso para hacer una sencilla página que muestra precisamente los 8 botones.



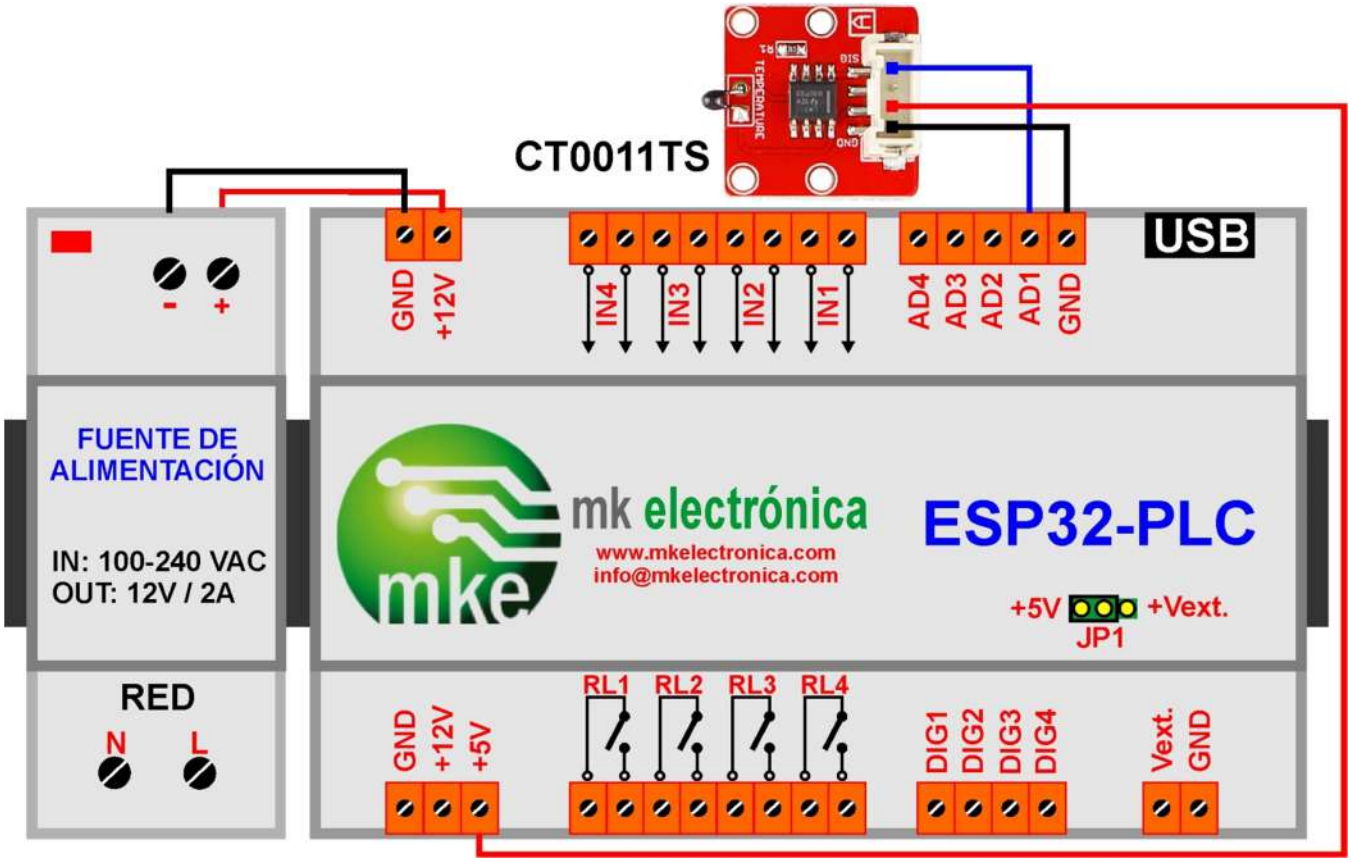
Cada vez que se pulsa uno de ellos el cliente envía una petición con uno de los comandos posibles. El servidor ESP32-PLC lo recibe, lo visualiza en el monitor serie del IDE, decodifica y actúa sobre el relé correspondiente.

Seguro que ya lo has deducido, pero que sepas que todo esto se puede hacer desde el navegador de un dispositivo móvil como puede ser un Smartphone o una Tablet. Simplemente debe de estar conectado a la misma red WiFi que el ESP32-PLC. Y cómo no, si conoces la IP pública que te asigna tu proveedor y abres el oportuno puerto en tu router, este control de los relés lo puedes hacer desde cualquier lugar del mundo.



5-3-6 Medir Temperatura

Imaginación al poder!! A partir de ahora puedes adaptar todos los ejemplos que hemos propuesto y todos los que se te ocurran. Con este vamos a medir de forma remota la temperatura que capta el sensor conectado a nuestro ESP32-PLC.



En esta ocasión la función **paginaHTML()** contenida en el fichero auxiliar "*paginaHTML.hpp*" envía al cliente una sencilla página web con un botón. Cada vez que se pulsa se envía el comando "**ReadTa**". El servidor ESP32-PLC lo recibe, mide el valor analógico del sensor en AD1, calcula la temperatura y retransmite al cliente el valor de la misma.

5-3-7 Buzzer

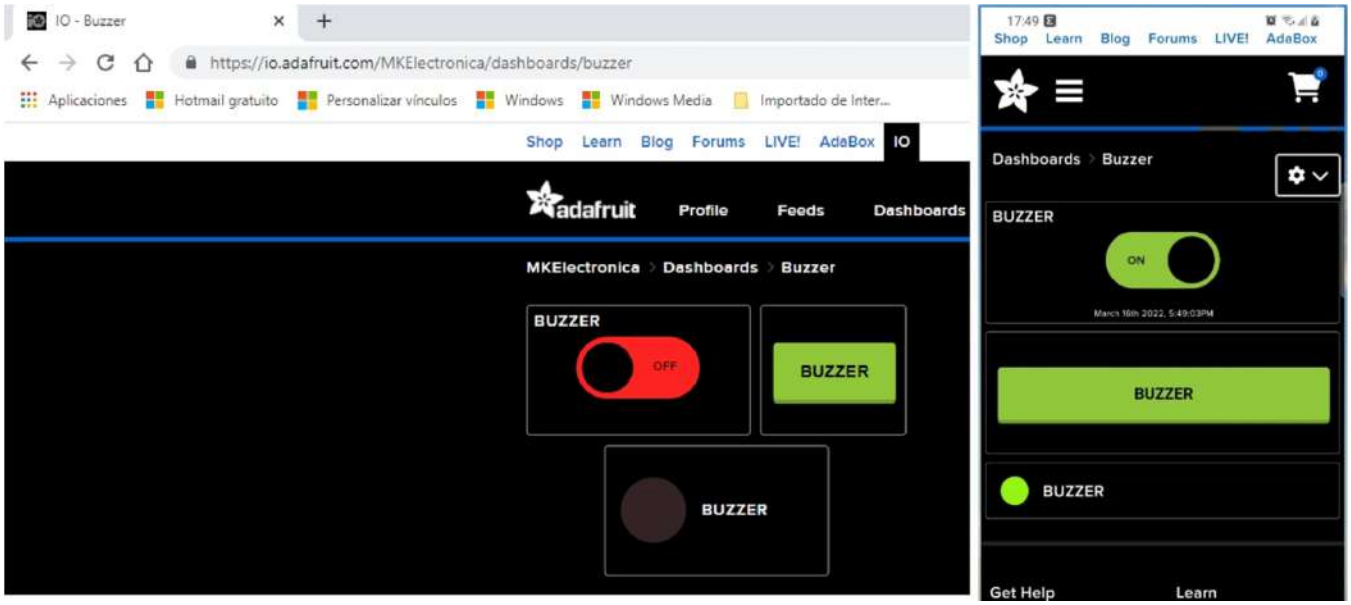
De los dos ejemplos anteriores seguro que deduces la importancia de servir páginas web que sean intuitivas, fáciles de usar y con cierta estética. Se trata de conseguir un buen interface entre el usuario cliente y el servidor. Sin embargo los lenguajes de programación para el desarrollo de servidores y páginas web, como el HTML que hemos empleado, son muy potentes y complejos, y no son objeto de este manual.

Para solucionarlo, en el mercado existen empresas comerciales que ofrecen plataformas que facilitan enormemente la creación de esos interfaces entre la aplicación y el usuario. Una de ellas es Adafruit IO (www.io.adafruit.com) que usaremos en este y en los siguientes ejemplos.

La idea consiste en la transferencia de datos entre la plataforma y el ESP32-PLC. Esos datos van empaquetados en lo que se conoce como "*feeds*". Esos feeds están vinculados a diferentes tipos de bloques o "*widgets*" configurables en un panel de trabajo o "*Dashboard*" que desarrollas en tu cuenta de Adafruit IO: pulsadores, interruptores, leds, gráficos, mandos deslizantes, visualizadores, etc…

En este ejemplo hemos creado un dashboard con tres bloques: un interruptor, un pulsador y un indicador luminoso. Están vinculados al mismo feed "**interruptor**" que refleja su estado. Cada vez que accionas el interruptor o el pulsador el valor del feed se refleja en el indicador luminoso y se transmite al ESP32-PLC, que lo analiza para activar/desactivar el zumbador.

Hacer todo esto requiere conocer el funcionamiento de Adafruit IO. Para ello tienes que crear una cuenta (gratuita) y revisar los múltiples tutoriales y ejemplos disponibles. Adafruit IO te asigna un nombre de usuario y una clave que deberás usar en todos tus programas. Es muy sencillo y los resultados son espectaculares. Todos los dashboard que hayas creado en tu cuenta son accesibles desde un PC o desde un dispositivo móvil y desde cualquier lugar del mundo. El resultado para este ejemplo lo tienes en las siguientes figuras.



Observa que hemos usado la librería "***AdafruitIO_WiFi.h***". Aunque adjuntamos una copia ZIP de la misma, La puedes descargar desde:

https://github.com/adafruit/Adafruit_IO_Arduino

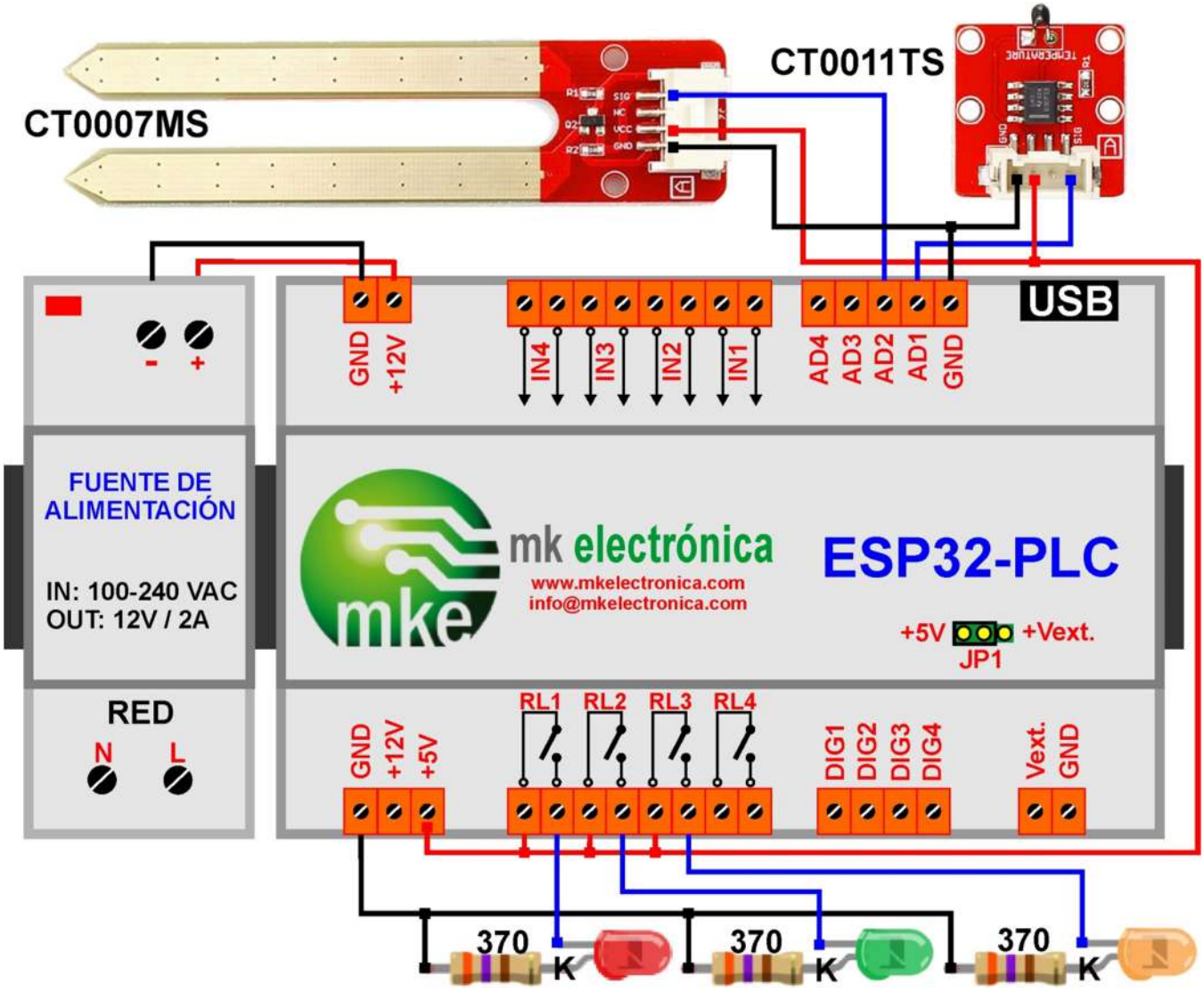
También puedes usar el administrador de bibliotecas del IDE de Arduino: **Programa → Incluir Librería → Administrar Bibliotecas**

5-3-8 Invernadero

Vamos con una nueva versión, la tercera, de nuestro ejemplo de invernadero. Vamos a utilizar el mismo montaje que empleábamos en el ejemplo 5-2-9. El sensor de humedad del suelo se conecta con la entrada AD2 en el modo analógico mientras que el sensor de temperatura se conecta en la entrada AD1.

Cuando la temperatura está por debajo de un umbral se activa el relé RL1 que pone en marcha el sistema de calefacción. Si está por encima de un umbral máximo se activa el

relé R2 que pone en marcha el sistema de refrigeración. Si la humedad del suelo está por debajo del valor establecido, se activa el relé RL3 activando un sistema de regadío. Hemos puesto unos leds en los contactos de los relés para representar estas tres situaciones.



El funcionamiento del programa ya lo conoces del ejemplo 5-2-9. La novedad consiste en que de forma remota y mediante la plataforma Adafruit IO, vamos a poder monitorizar en todo momento el estado de nuestro invernadero desde cualquier lugar del mundo.

Para ello empleamos dos feeds, "**humedad**" y "**temperatura**", a través de los cuales el ESP32-PLC envía a la plataforma (publica) el valor de estos parámetros detectados por los sensores. Por su parte en la plataforma hemos creado un panel de control o "*Dashboard*" con los 7 bloques que puedes ver en la figura. Cada bloque está asociado a

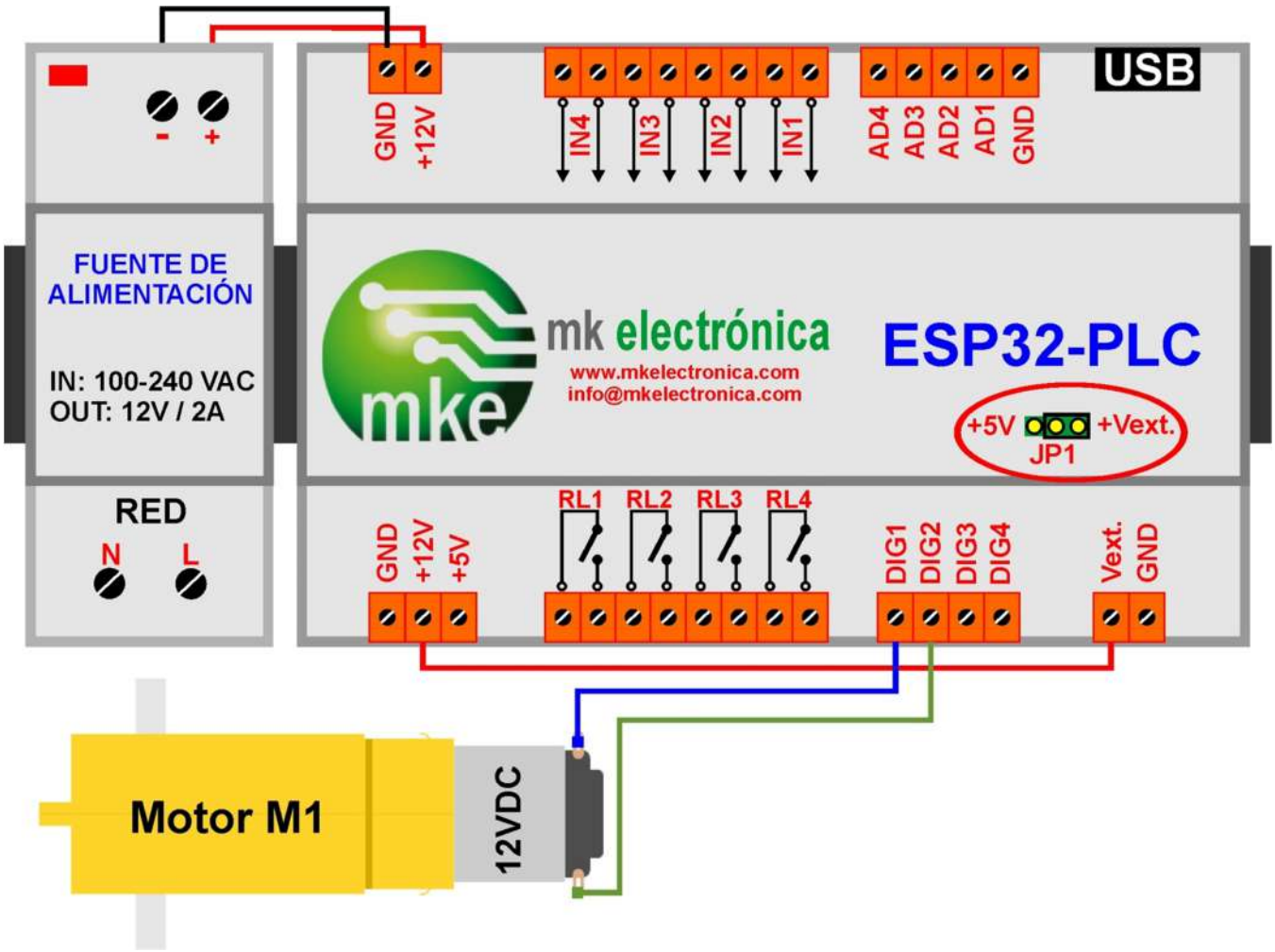
esos feeds para poder así representar de forma elegante e intuitiva los valores de los mismos, Es decir, lo que se envía desde nuestro invernadero controlado por el ESP32-PLC.

1. Indicador gráfico de línea que representa el feed de temperatura. Está configurado para visualizar un rango de 0º a 60º de forma que si la temperatura es inferior a 15º el indicador se visualiza en azul. Si está entre los 15º y 30º se visualiza en verde y si es superior a 30º se visualiza en color rojo.
2. Indicador gráfico de línea que representa al feed la humedad. Está configurado para una rango de 0 a 100 % de forma que si esta está por debajo de 20 se representa en color marrón, si está 60 y 20 se visualiza en verde y si la humedad está por encima de 60 se visualiza en azul.
3. Este bloque es del tipo gráfico y representa simultáneamente ambos feeds de temperatura y humedad. Es una forma muy intuitiva de ver la evolución a lo largo del tiempo de esos dos parámetros procedentes de los sensores conectados al ESP32-PLC.
4. Visualizador numérico que representa la temperatura. Otra forma de representar un feed.

5. Visualizador luminoso que representa al feed de la temperatura y que se activa en color verde si es superior a 30º. Indica la puesta en marcha del sistema refrigerador según el umbral establecido.
6. Visualizador luminoso que también representa el feed de la temperatura procedente el sensor. Se activa en color rojo si es inferior a 15º e indica la puesta en marcha del sistema calefactor.
7. Visualizador luminoso que representa el feed de la humedad medida por el sensor del ESP32-PLC. Se activa en color azul si está por debajo de 20 indicando así la puesta en marcha del sistema de regadío.

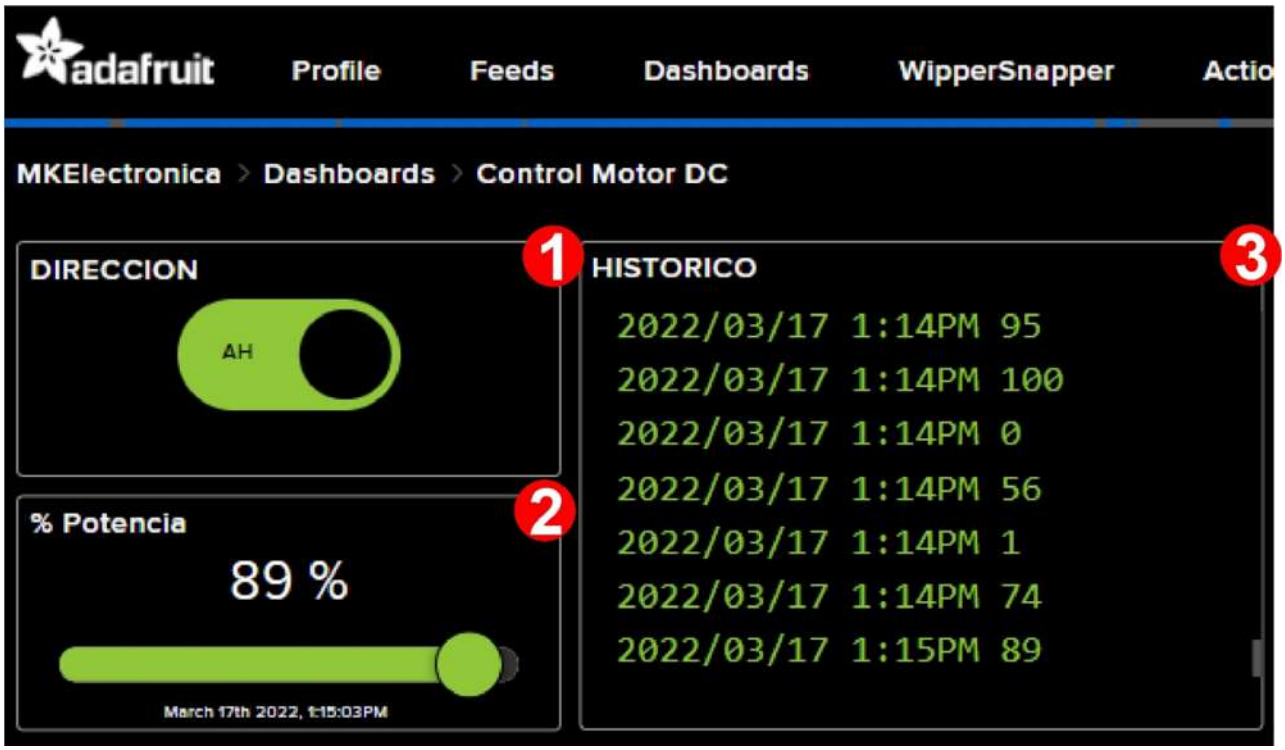
5-3-9 Control Motor DC

Nuevamente controlaremos un motor DC, pero esta vez lo haremos de forma remota, mediante la plataforma Arduino IO y desde cualquier lugar del mundo.



La generación de una señal PWM para controlar la potencia aplicada al motor no es ninguna novedad. Ya lo hemos hecho en anteriores ejemplos. Nos centramos en la novedad.

En Adafruit IO hemos creado un panel de control como el de la figura. Gestiona dos feeds cuyos valores serán transmitidos al ESP32-PLC para que actúe sobre el motor. El feed "**potencia**" determina la duración del ciclo útil entre 0 y 100% y el feed "**dirección**" determina el sentido de giro. El panel tiene tres bloques o widgets.

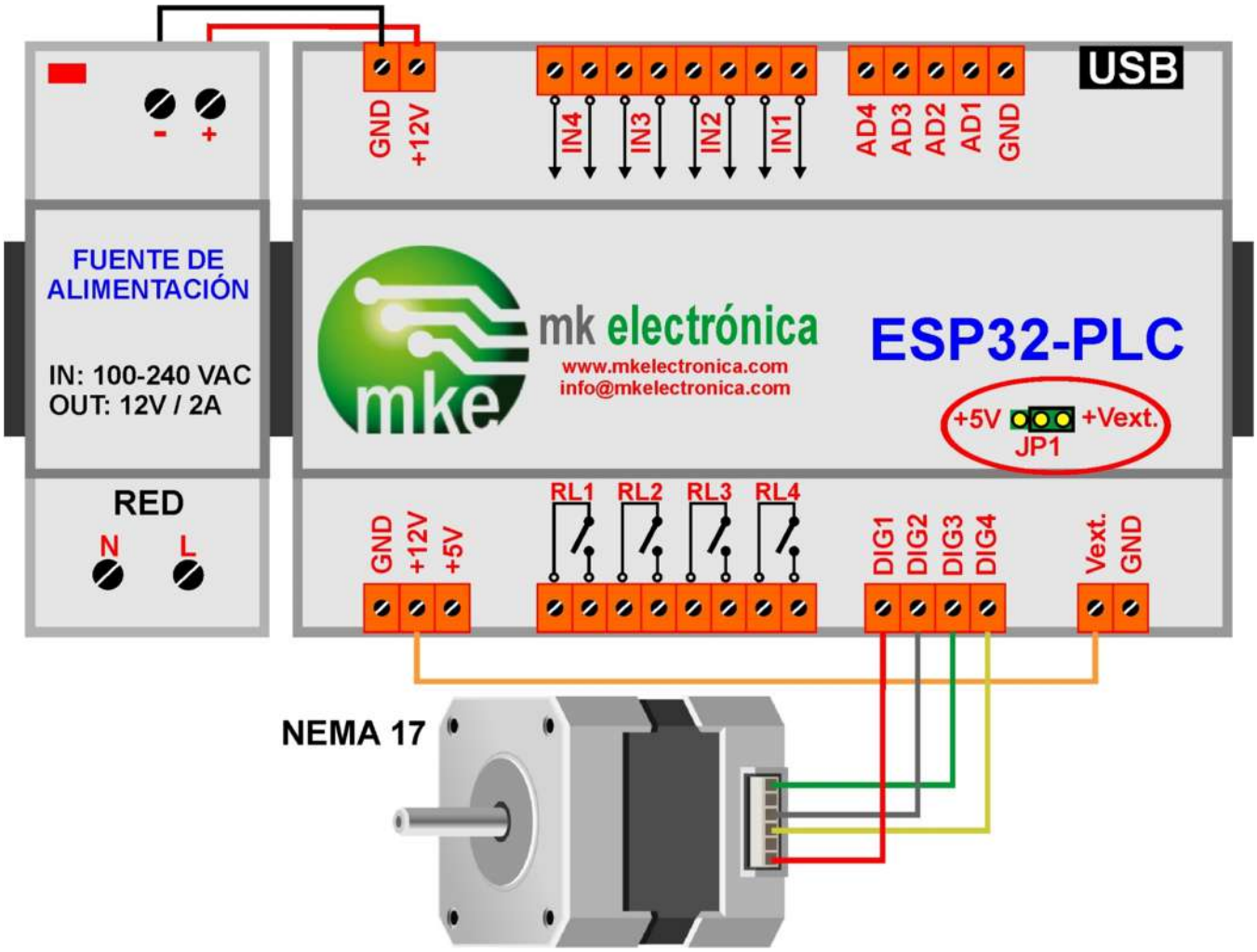


1. Interruptor que está asociado al feed "**dirección**". Cuando cambia de estado se transmite al ESP32-PLC el estado actual "1" o "0". El programa en el ESP32-PLC recibe este nuevo valor y determina el sentido de giro que debe producirse.
2. Slider o mando deslizante asociado al feed "**potencia**". Está configurado en un rango de 0 a 100 en pasos de uno en uno. Cada vez que se modifica Adafruit IO transmite el nuevo valor. El ESP32-PLC lo recibe y determina el ciclo útil de la señal PWM. El motor se mueve a la velocidad y sentido de giro indicado.
3. Bloque stream que se asocia a ambos feeds. Cada vez que cualquiera de ellos cambia de estado, este bloque visualiza el nuevo valor y la fecha/hora en que se produjo. Es una buena forma de registrar los movimientos producidos en el motor.

NOTA: Recuerda que el motor se alimenta a 12V y que debes de poner el jumper JP1 en la posición VEXT.

5-3-10 Motor PAP

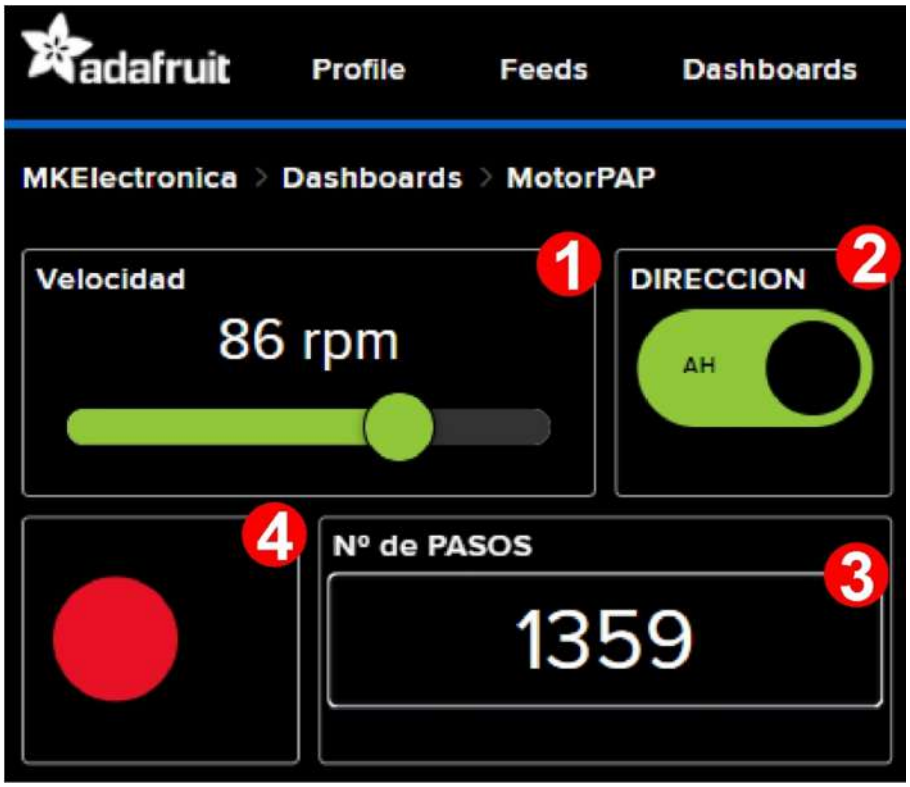
De igual manera usaremos la plataforma Adafruit IO para controlar los movimientos de un motor paso a paso como el usado en anteriores ejemplos.



NOTA: Recuerda que el motor se alimenta a 12V y que debes de poner el jumper JP1 en la posición VEXT.

En este caso empleamos tres feeds diferentes "**velocidad**" envía al ESP32-PLC la velocidad con la que se deben realizar los movimientos, "**dirección**" envía el sentido de giro y "**Npasos**" contiene los pasos que el motor debe realizar.

Para interactuar con el usuario mediante un interface intuitivo hemos creado un panel de control o "*Dashboard*" con cuatro bloques que detallamos a continuación.

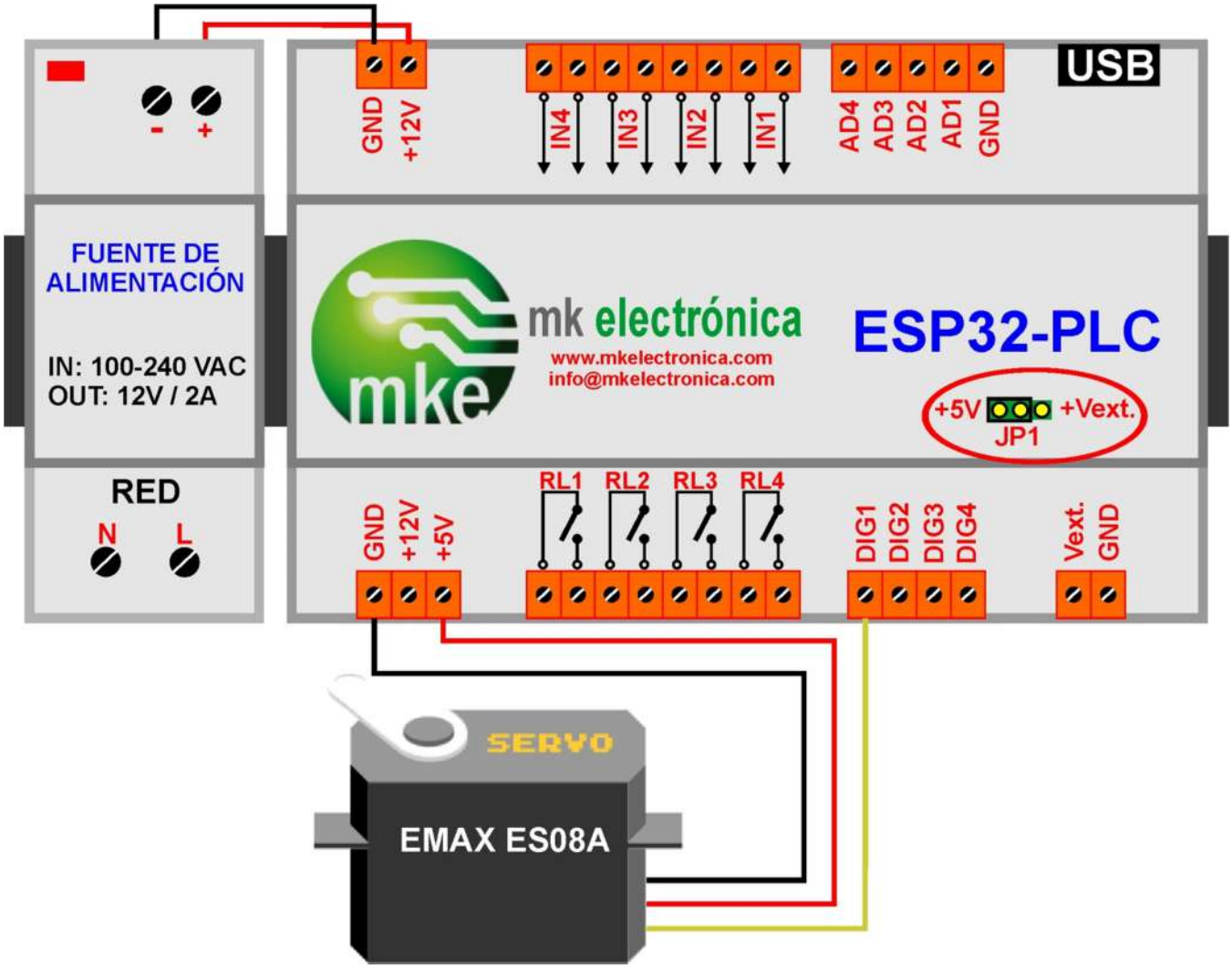


1. Slider asociado al feed "**velocidad**". Cada vez que se mueve transmite al ESP32-PLC la nueva velocidad. Está configurado para enviar un valor de un rango entre 1 y 125 rpm.
2. Interruptor asociado al feed "**dirección**". Cuando cambia de estado transmite el nuevo valor "1" o "0" para que el ESP32-PLC determine el sentido o dirección de giro.
3. Bloque de texto asociado al feed "**Npasos**". Cada vez que introducimos un nuevo valor numérico se transmite el número de pasos que deben realizarse. El ESP32-PLC ha recibido las instrucciones necesarias para ejecutar un nuevo movimiento: velocidad, dirección y número de pasos.
4. Indicador luminoso asociado al feed "*Npasos*". Se activa en color verde cuando el Nº de pasos es 0, es decir, cuando se ha completado un movimiento. En caso contrario se mantiene en color rojo.

5-3-11 Servo Control

Vamos a acabar con este último ejemplo para controlar desde Adafruit IO un servomotor. El montaje es el mismo que el usado en otros ejemplos.

NOTA: Recuerda que el servo motor se alimenta a 5V y que debes de poner el jumper JP1 en la posición +5V.



Tenemos un único feed, "**posiciondelservo**", que determina la posición entre 0º y 180º en la que el servo se debe posicionar.

El panel de control que hemos creado emplea tres bloques asociados a ese mismo feed:

1. Slider o mando deslizante. Está configurado para un rango de 0º a 180º en pasos de uno en uno. Cada vez que se mueve se transmite el nuevo valor del feed "**posiciondelservo**" al ESP32-PLC para que ejecute el movimiento.
2. Grafica que representa los valores del feed. Es una forma de representar los movimientos que ha realizado el servo motor.
3. Bloque Stream que representa también el valor del feed "**posiciondelservo**". Es otra forma de representar el historial con los movimientos que se han producido así como la fecha y hora de los mismos.

Y así llegamos al final de este manual de usuario de nuestro controlador lógico programable ESP32-PLC. En estos últimos ejemplos hemos propuesto el uso de una plataforma como Adafruit IO para intercambiar información mediante feeds o campos de datos entre el panel de control o Dashboard que hemos creado, y nuestra aplicación en el ESP32-PLC.

Hemos usado una versión gratuita de demostración que te permite tener hasta 5 paneles diferentes y 10 feeds. Dispones de una versión de pago que te permite tener un número ilimitado de dashboards, feeds y grupos de feeds, puede almacenar más datos y durante más tiempo en la nube, y permite una mayor velocidad de transferencia de feeds entre ESP32-PLC y la plataforma. También puedes configurar un potente sistema que permite disparar eventos como enviar un emáil o publicar un mensaje cuando el valor de un determinado feed cumple una condición.

También hay otras plataformas como Blynk, ThingSepeak, Arduino IoT, Zetta, Google Cloud, etc… Te invito a que investigues y te quedes con la que mejor te venga. En el fondo todas ellas tienen por objeto la transferencia de datos con dispositivos y facilitar el interface con el usuario. **Es una excelente solución para diseñar aplicaciones de Internet de las Cosas IoT**.