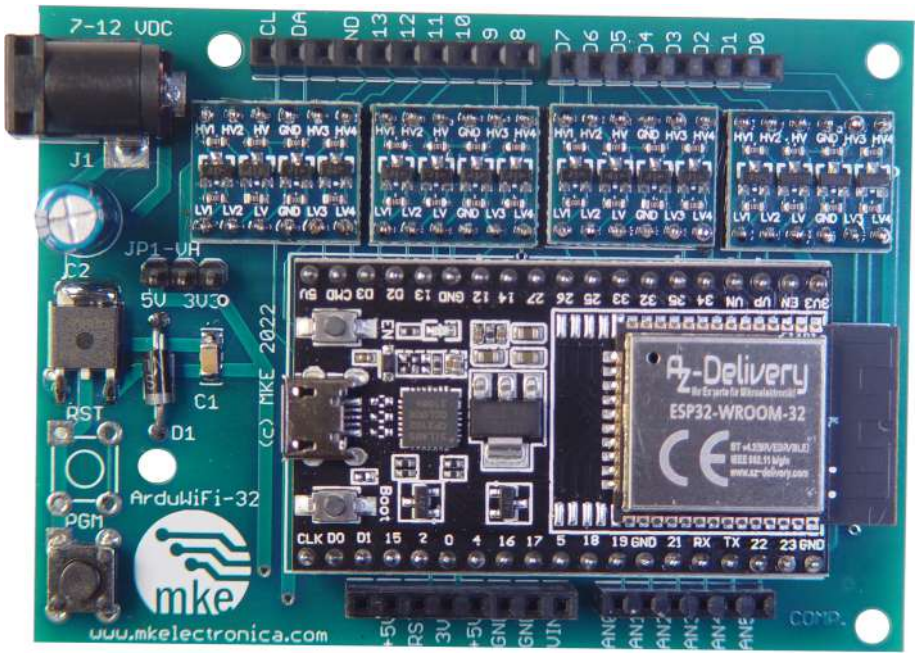
ArduWiFi-32

Manual de usuario



Rev. Febrero 2022

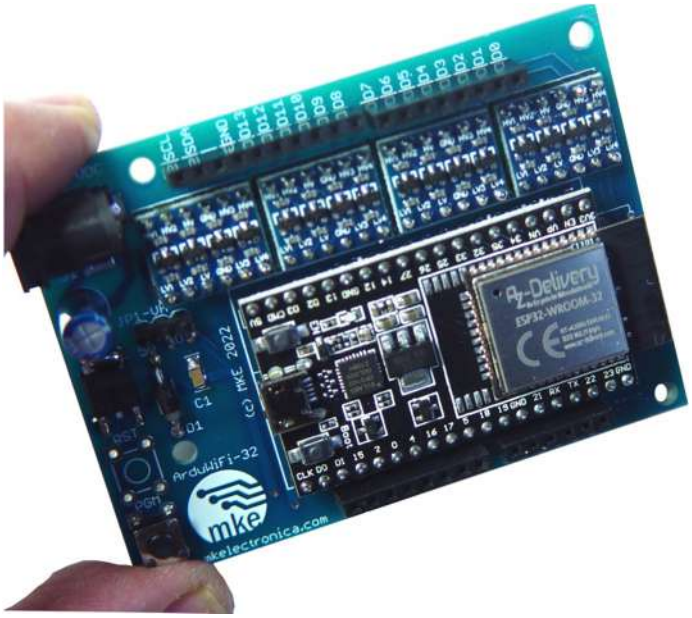
MK Electrónica

Tfno. 34-944 04 80 89

Email: info@mkelectronica.com
www.mkelectronica.com

ArduWiFi-32: Manual de usuario

1.- INTRODUCCIÓN



¿Quieres seguir avanzando y dar un paso adelante en el desarrollo de proyectos y aplicaciones? Quizá nuestra tarjeta controladora ArduWiFi-32 sea la solución. Se trata de una tarjeta que incorpora un micro controlador de última generación, el ESP32, con excelentes características y capacidad de comunicación Bluetooth y WiFi.

ArduWiFi-32 se presenta en un formato compatible con la popular tarjeta controladora Arduino UNO en cuanto a la distribución de patillas se refiere. Se dice que a nivel de hardware es compatible con buena parte de los shields desarrollados para Arduino UNO. En **MK Electrónica** la venimos usando en nuestro robot ArduBot y con los shields:

- BASIC I/O (https://mkelectronica.com/producto/basic-i-o/)
- Shield de relés: (https://mkelectronica.com/producto/shield-de-4-reles-para-arduino-y-compatibles/)
- Shield LCD: (https://mkelectronica.com/producto/shield-lcd-arduino-compatibles/)
- Servo : (https://mkelectronica.com/producto/mini-servo-motor-emax-es08a/)

Existen diferentes herramientas, lenguajes y plataformas para la programación del micro controlador ESP32, 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 ArduWiF32. 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, la tarjeta ArduWiFi-32 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 ella. !!

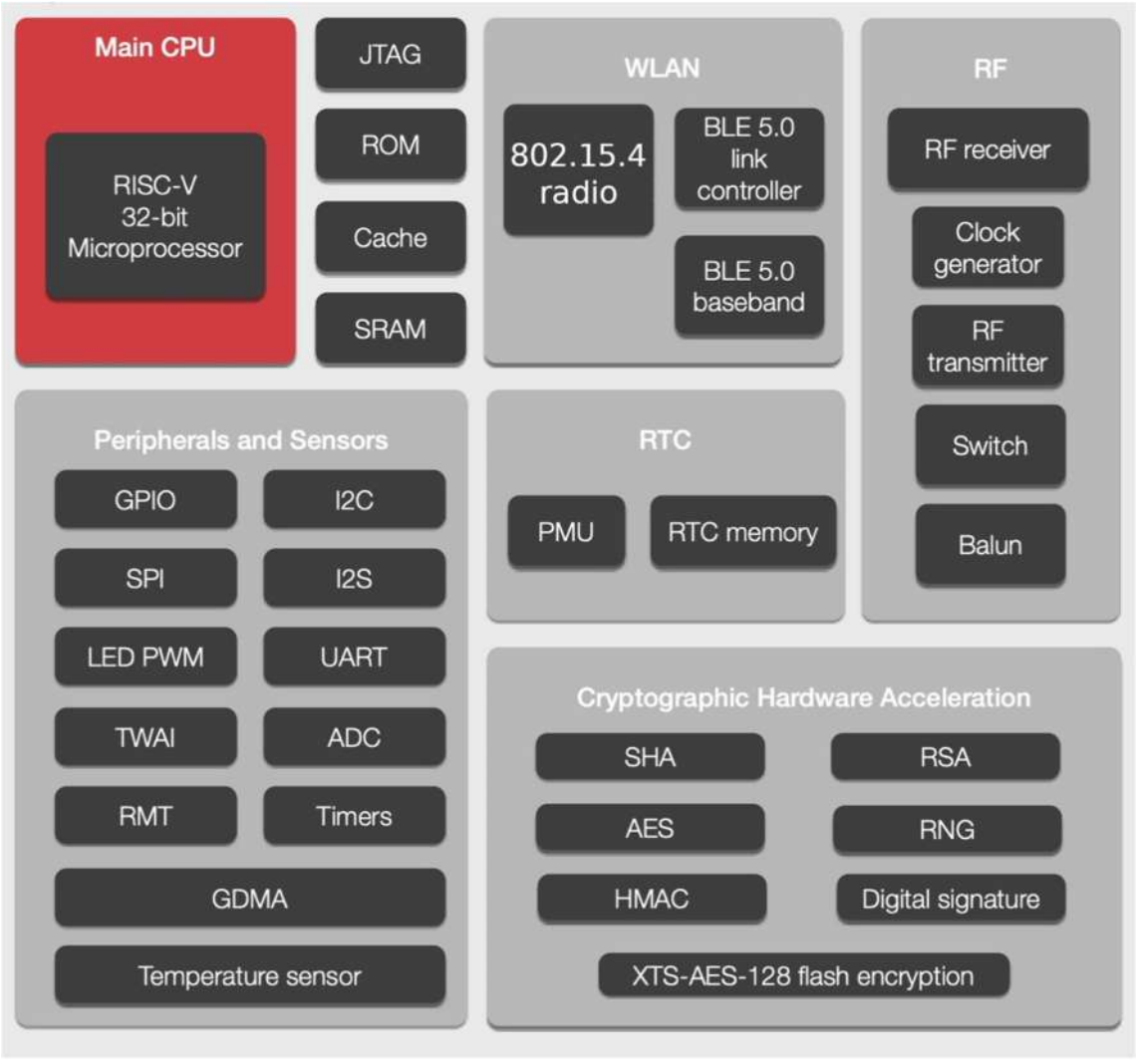
2.- ¿EN QUÉ CONSISTE?

Para explicar la razón de ser de ArduWiFi-32 debemos empezar por el principio y hablar de los tres componentes fundamentales que la 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.

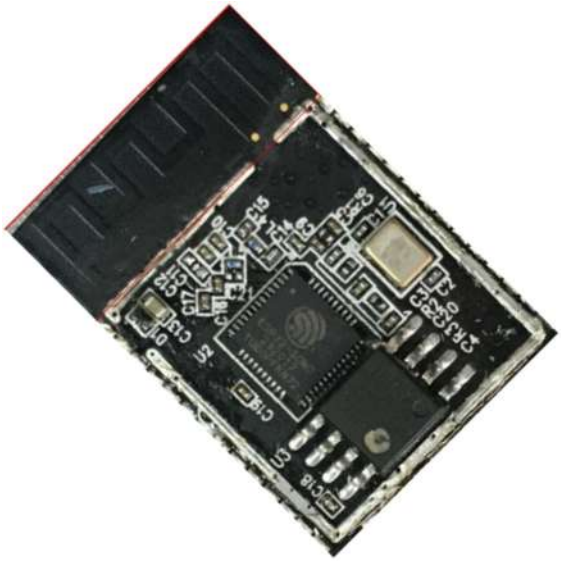
2-2 Un módulo

Una característica de los SoC ESP32 es que **NO** integran la memoria donde almacenaremos 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 ESP8266 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 las siguientes características:

- 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 ·"*tarjeta 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.

La solución se presenta en forma de "*Device kit*", que no es ni más ni menos que un circuito electrónico 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.

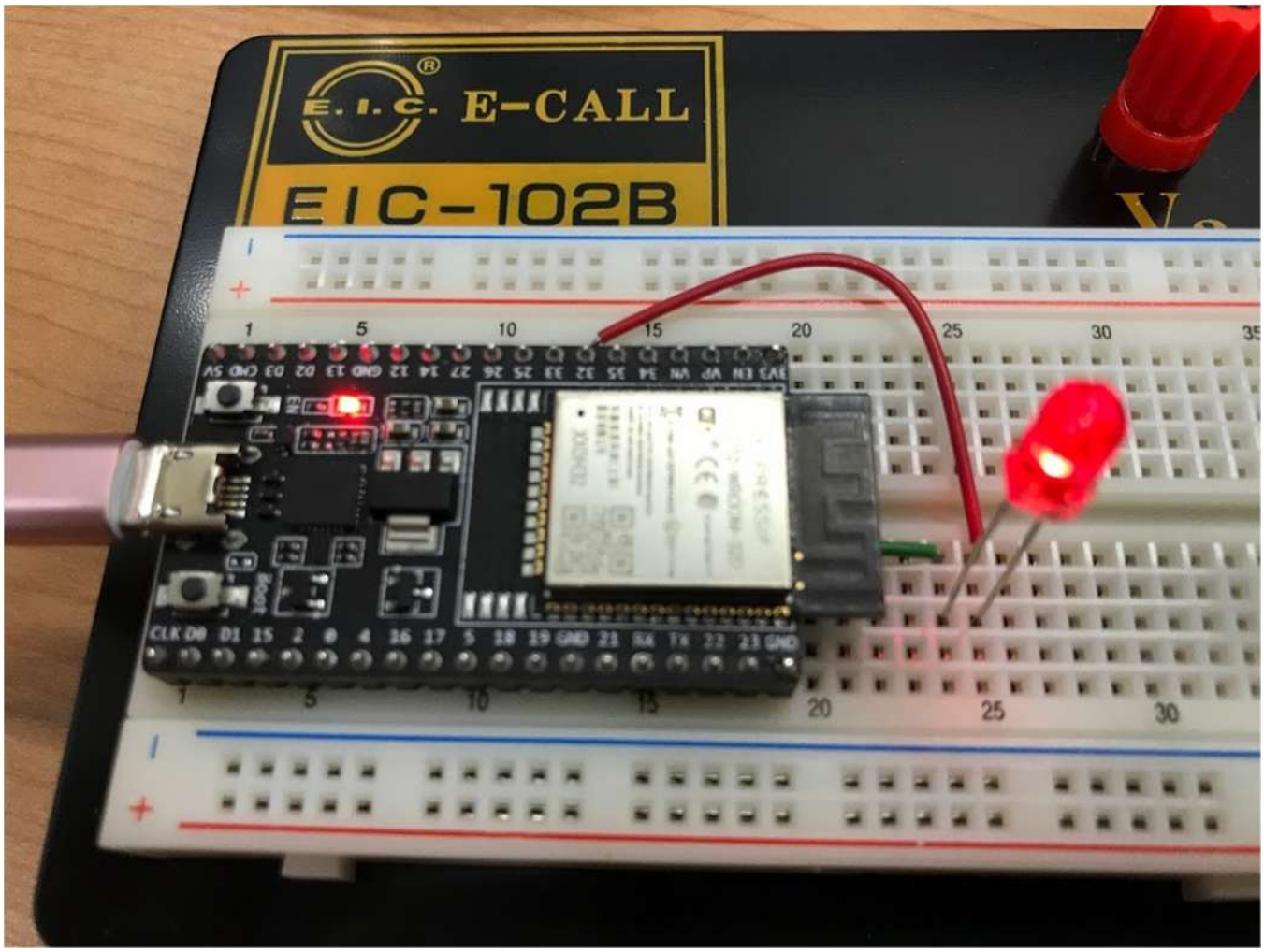
Aquí también nos podemos encontrar con multitud de fabricantes. La ventaja es que hay un amplio comercio de este tipo de tarjetas de desarrollo, pero no todos son compatibles entre sí al 100%. Puede haber pequeñas diferencias y hay que estar atentos.

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.

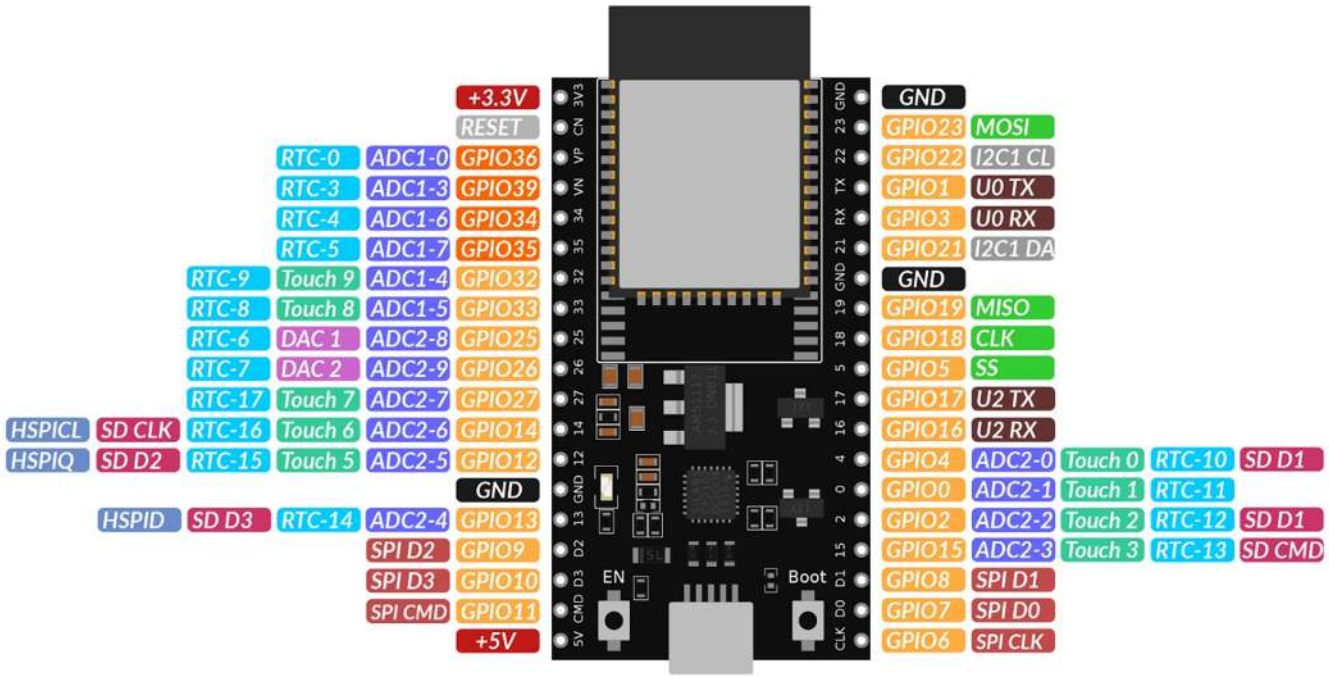
El ESP32-DevKitC incorpora y hereda por tanto todas las características del módulo ESP32-WROOM-32 que a su vez hereda las del SoC ESP32. Se programa desde un puerto USB y admite una tensión de alimentación d +5V, a partir de los cuales se obtienen los +3.3V que necesita el controlador ESP32.

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 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 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

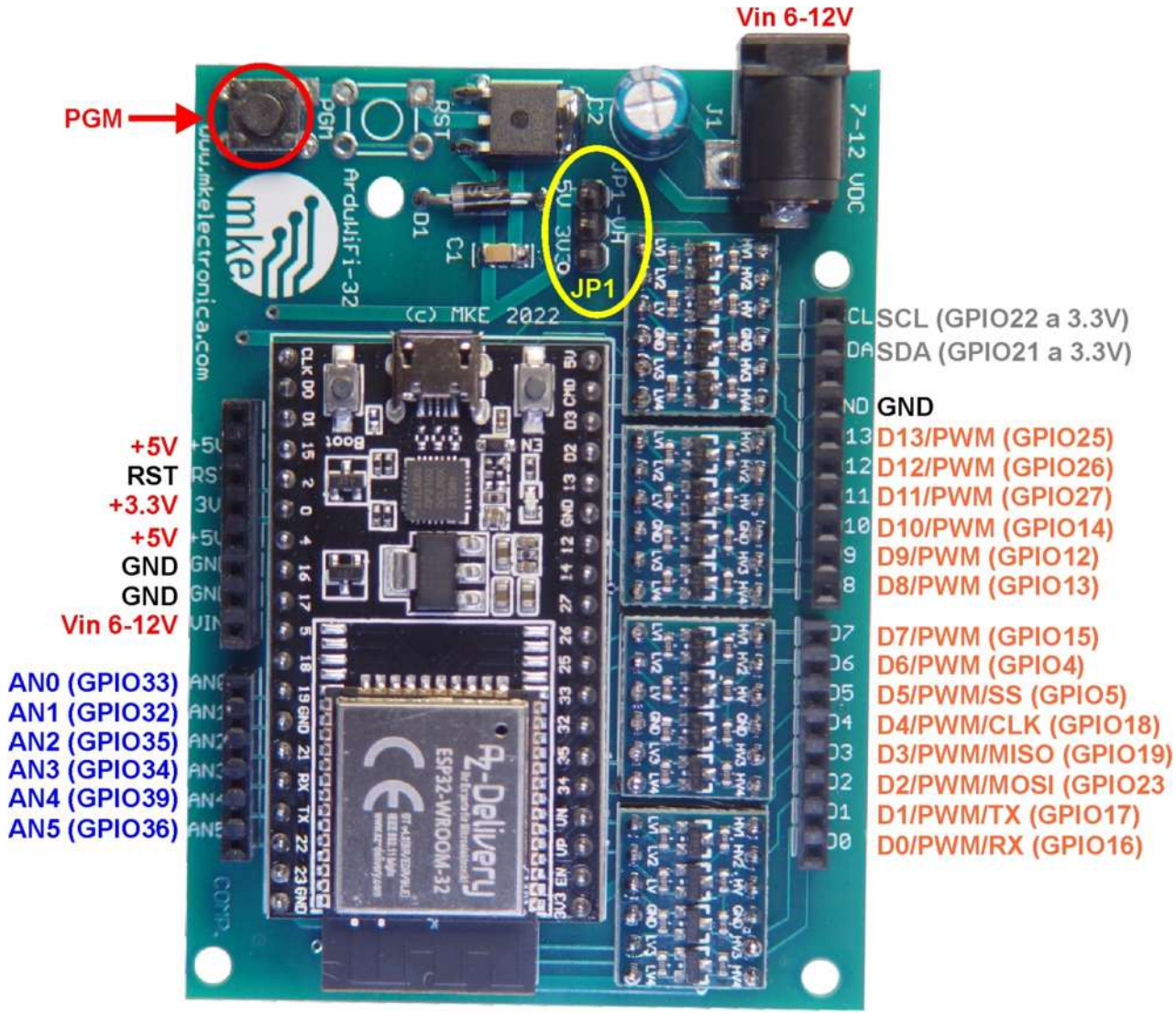
2-4 ArduWiFi-32

Y así llegamos a nuestra tarjeta controladora. 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 ArduWiFi tiene su espacio:

- Utiliza el kit de desarrollo ESP32-DevKitC, con lo que hereda las características del módulo ESP23-WROOM-32 y del controlador SoC ESP32.
- Admite señales de E/S con niveles de 5V como las que emplean la mayor parte de shields disponibles actualmente.
- Permite usar una alimentación más flexible comprendida entre 6 y 12V.
- Las señales de E/S tienen la misma disposición que en la popular tarjeta controladora Arduino UNO, y es compatible con un buen número de shields.

Todo no puede ser… Al tratar de mantener una compatibilidad con las señales de Arduino UNO, hemos tenido que sacrificar algunas características y prestaciones propias del controlador ESP32, sin embargo tenemos la gran ventaja de poder aprovechar un buen número de los shields existentes en la actualidad. En la figura puedes ver la relación de señales entre ArduWiFi-32/Arduino UNO y el SoC ESP32.



A diferencia de Arduino UNO, hay que destacar que:

- Podemos usar una alimentación flexible comprendida entre 6 y 12V (Vin)
- Cualquiera de las señales D0-D13 pueden generar señales PWM.
- Son compatibles con niveles de 5V o de 3.3V según el jumper JP1.
- El ESP32 dispone de un 2º canal serie tipo UART al que se puede acceder mediante las señales D0 (Rx) y D1 (Tx), La comunicación con el IDE se realiza exclusivamente mediante el UART del canal 1.

- De igual manera, las señales SDA y SCL del bus I2C son exclusivas y no están compartidas con A4 y A5 como ocurre en Arduino UNO. **Atención, solo admite niveles de 3.3V** en colector abierto.
- Las señales D2, D3, D4 y D5 se pueden configurar como señales MOSI, MISO, CLK y SS para un interface SPI.
- Las entradas analógicas AN0-AN5 admite tensiones entre 0 y 5V gracias a un divisor de tensión resistivo, pero se pierde algo de precisión debido a las tolerancias.

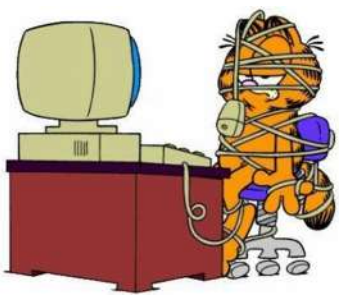
Si a todo lo anterior le añadimos las características propias del controlador ESP32 y el módulo que lo contiene, podemos hacer una sencilla comparación con Arduino UNO:

- CPU con arquitectura RISC de doble núcleo y de 32 bits. Arduino UNO 8 bits.
- Velocidad de trabajo de 240 MHz frente a 16 MHz en Arduino UNO.
- Memoria Flash de programa de 4 MB frente a los 32 KB de Arduino UNO.
- Memoria RAM de datos de 520 KB frente a los 2 KB de Arduino UNO.
- Memoria ROM de 448 KB con el firmare residente, bootloader, etc…
- Convertidor ADC de 12 bits de resolución frente a los 10 bits en Arduino UNO.
- 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

Resumiendo. Aunque en ArduWiFi-32 hemos usado las mismas señales de E/S, la misma distribución, empleamos el mismo IDE y el mismo lenguaje de programación, **NO TIENE NADA QUE VER** con Arduino.

3.- INSTALACIÓN Y CONFIGURACIÓN

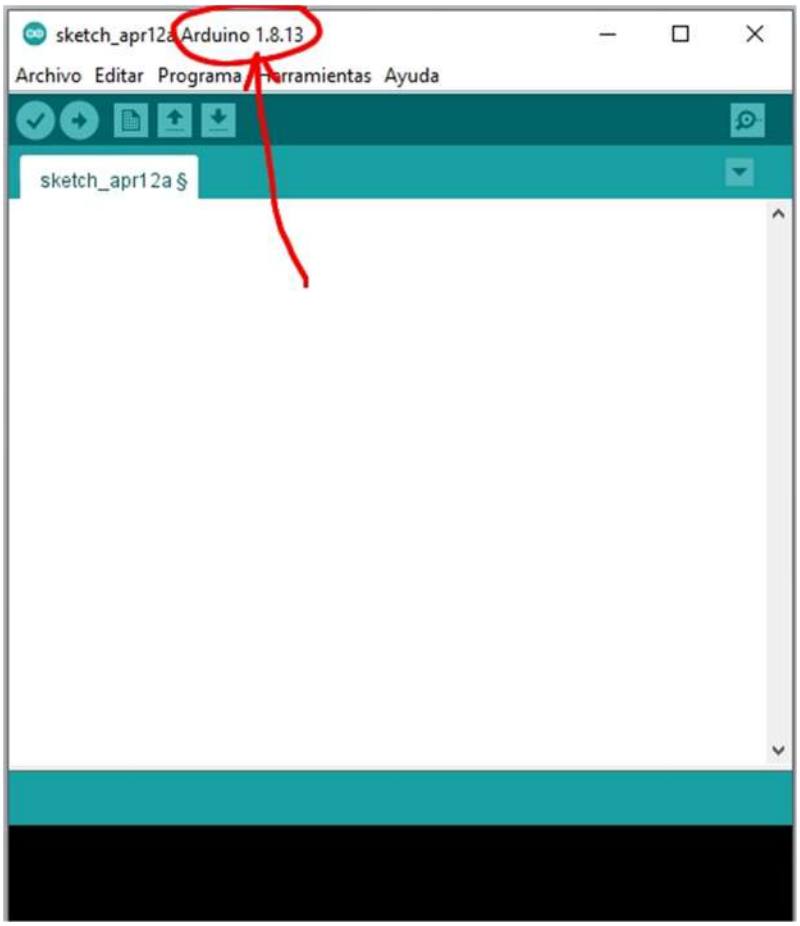
No me cansaré de repetir que vamos a trabajar con un microcontrolador que tecnológicamente hablando nada tiene que ver con el Atmega 328P empleado en la tarjeta Arduino UNO u otras similares. Vamos a trabajar con una tarjeta, ArduWiFi-32, que emplea el controlador ESP32, mucho más evolucionado, de mayores 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.



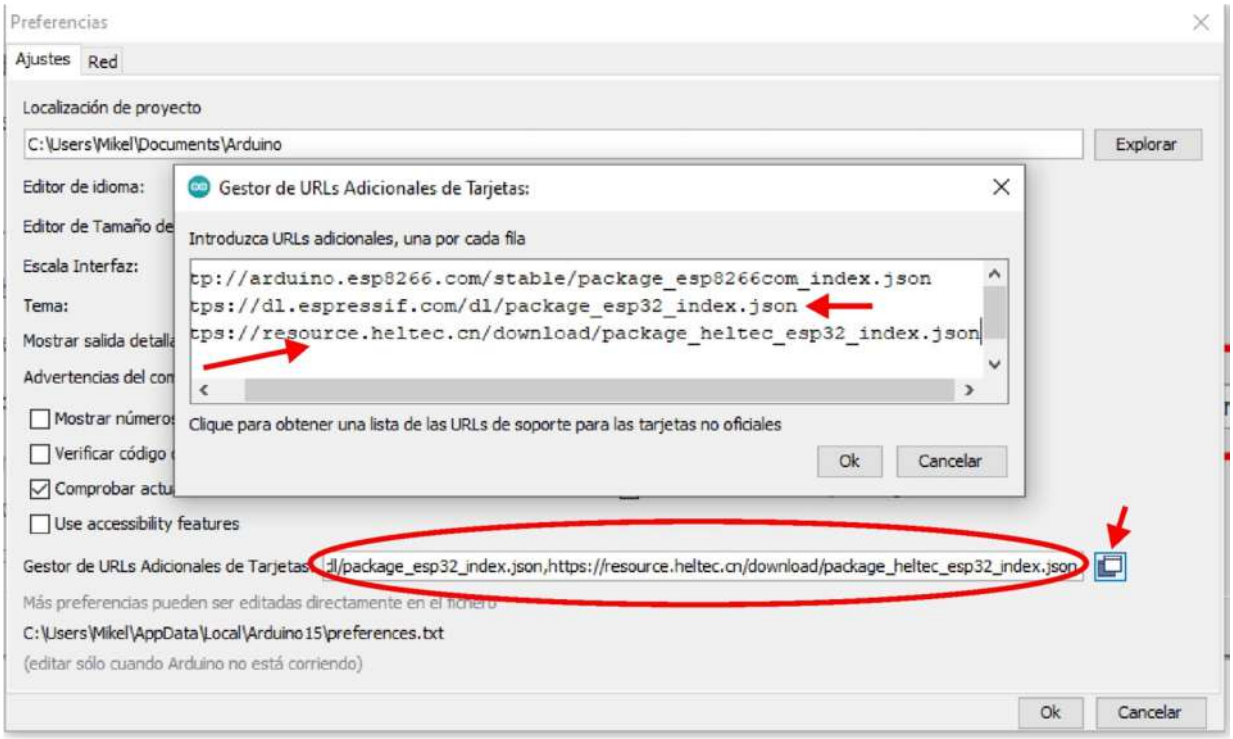
3-1 Gestor de placas

Quizá esto sea una novedad para ti, pero las actuales versiones del IDE permiten trabajar con muchísimas tarjetas controladoras sean de Arduino o no. 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 y programas necesarios que sean capaces de gestionar esas tarjetas.

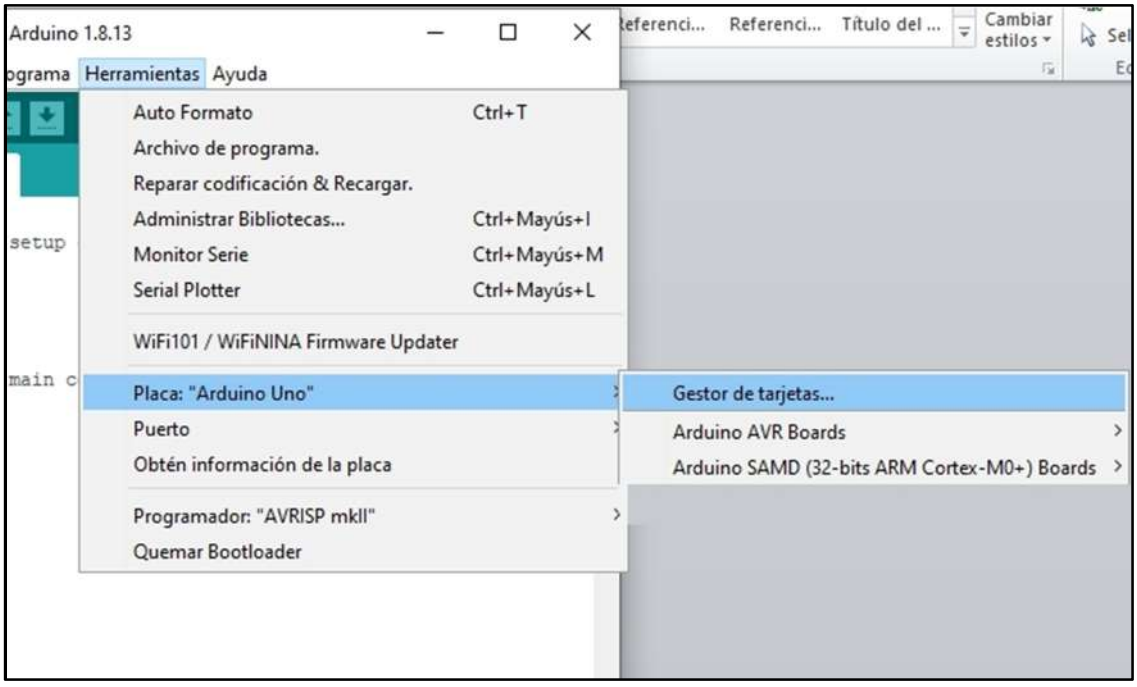
En el campo **Gestor de URLs** de la opción **Archivo → Preferencias** del IDE debemos copiar estas direcciones, que son donde se encuentran esas librerías y programas:

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

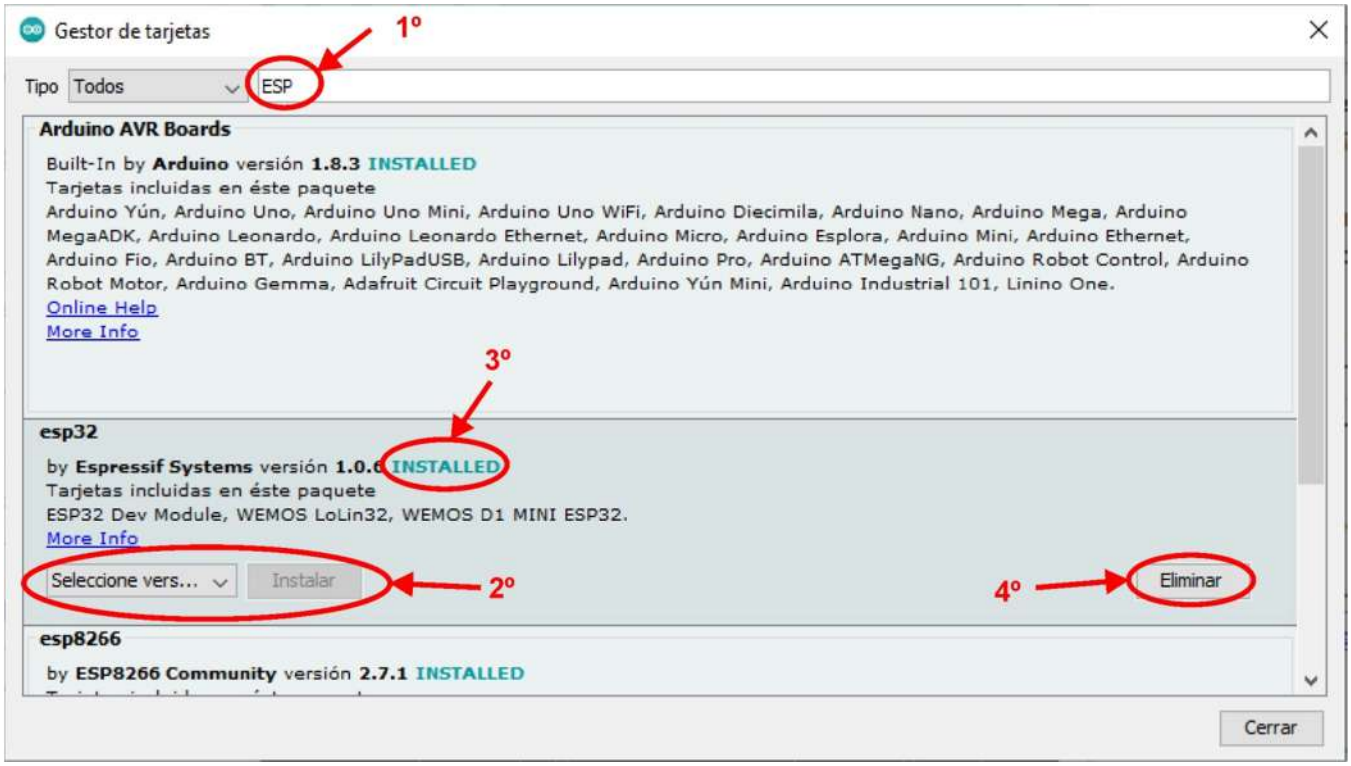


Como es posible que quizá tengas otras direcciones o URL's adicionales, puedes pulsar el botón de la derecha. Se abre una ventana donde puedes insertar y editar tantas direcciones como necesites.

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.

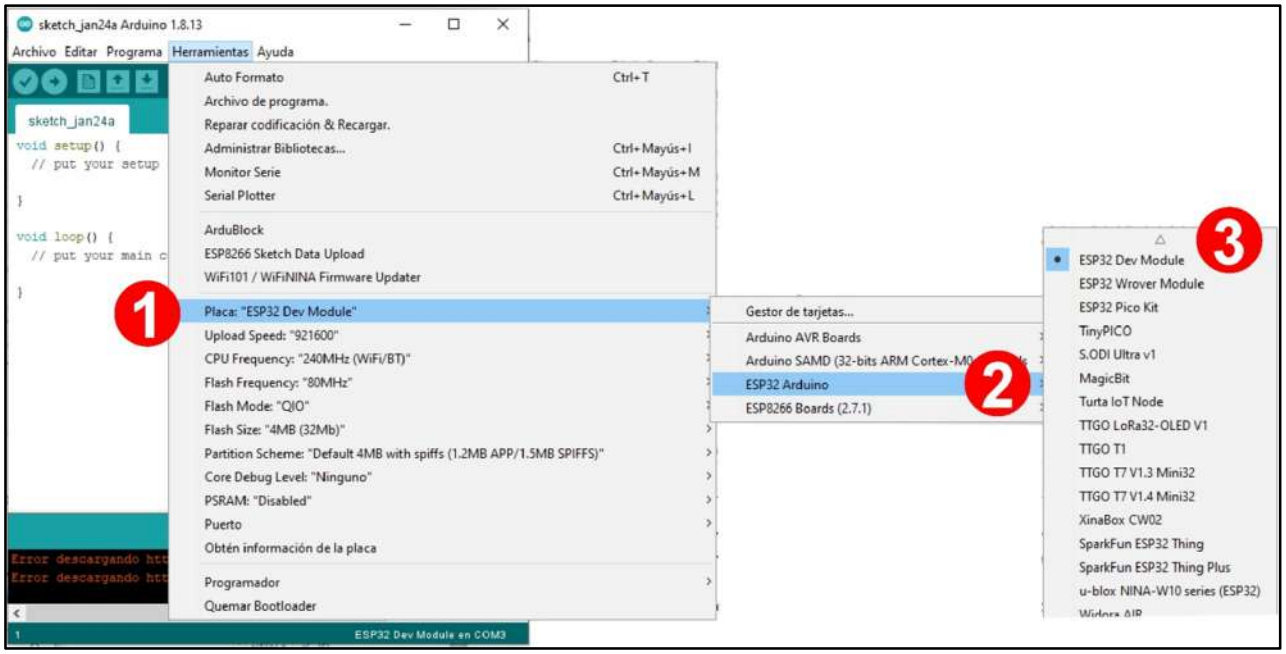
3º En mi caso aparece que ya tengo instalado las librerías y accesorios relacionados con las tarjetas basadas en el SoC ESP32.

4º En esta misma ventana también podemos actualizar y/o borrar esas librerías.

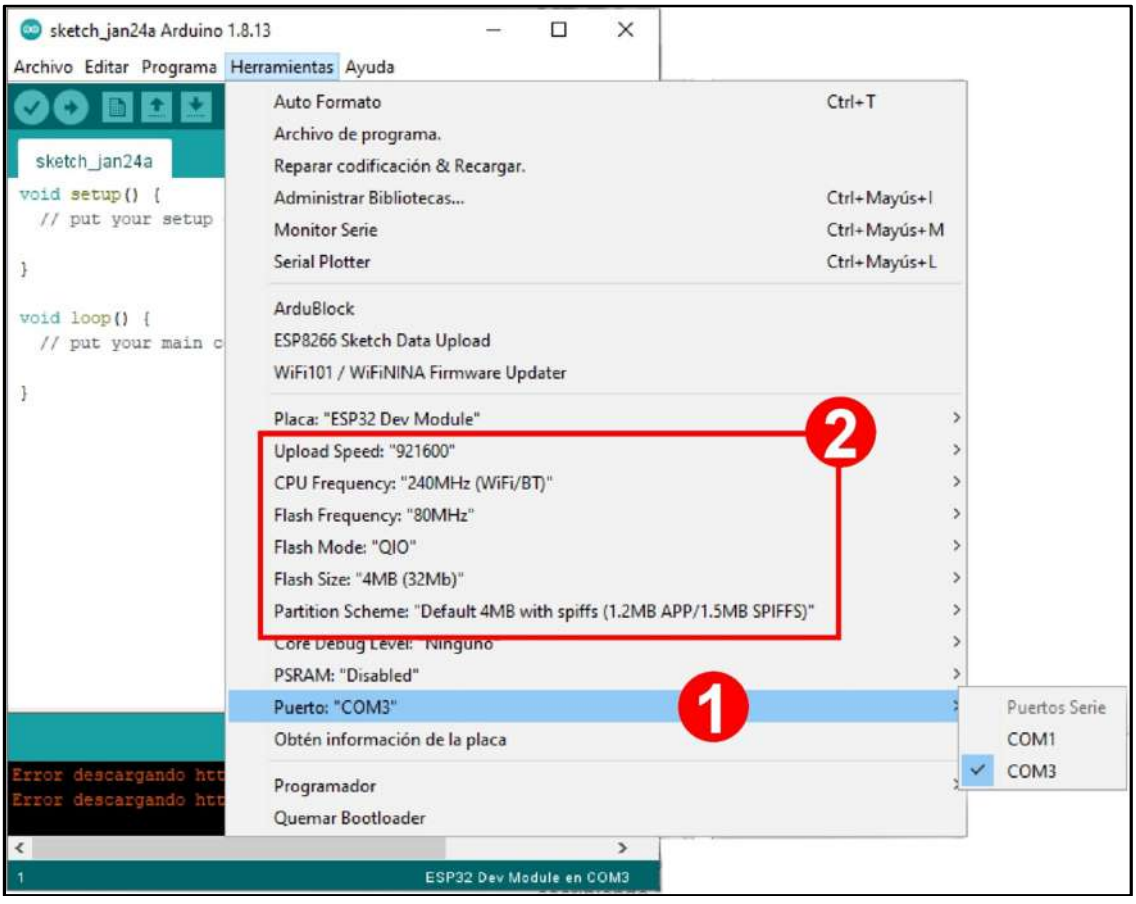
3-2 Configuración

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.



Por último, mediante **Herramientas → Puerto** (1), debemos configurar el puerto COM que el sistema operativo nos asigne cuando conectemos ArduWiFi-32 al puerto USB. También podemos ver un resumen de las características del controlador (2).



4.- EJEMPLOS

La mejor forma de ver funcionando a ArduWiFi-32 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 ArduFiFi32 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 algunas tarjetas o shields bastante comunes, aunque esto no garantiza que todas las que hay en el mercado sean compatibles con ArduWiFi-32:

- Shield BASIC I/O: https://mkelectronica.com/producto/basic-i-o/
- Shield de relés: https://mkelectronica.com/producto/shield-de-4-reles-para-arduino-y-compatibles/
- Shield LCD: https://mkelectronica.com/producto/shield-lcd-arduino-compatibles/
- Mini Servo: https://mkelectronica.com/producto/mini-servo-motor-emax-es08a/

4-1 Ejemplos básicos

Trataremos de usar la mayor parte de recursos que ofrece ArduWiFi 32 y su controlador, el SoC ESP32. En principio vamos a usar la tarjeta básica de periféricos BASIC I/O, que se inserta en ArduFiWi32 como cualquier otro shield. En la red puedes encontrar información de todas las funciones del lenguaje de programación así como abundantes ejemplos

IMPORTANTE !! Por razones de compatibilidad eléctrica es necesario cambiar red SIL de resistencias marcada en la figira, por un valor comprendido entre 470 Ω y 2K2 Ω. Todos los shields comercializados a partir de enero de 2022 ya tienen la modificación.

4-4-1.- Blink

Es un clásico. Es el ejemplo más sencillo y consiste en hacer una intermitencia en el led rojo de BASIC I/O. La única novedad está en fichero auxiliar "**definiciones.h**" que se incluye en todo los ejemplos. Define la relación que hay entre las señales de E/S digitales de Arduino y las correspondientes en el SoC ESP32. Así, D0 se corresponde con la señal GPIO16, D1 con GPIO17, D2 con GPIO23, etc… Revísalo ¡!

4-1-2.- Entradas/Salidas

Otro clásico. En esta ocasión se trata de leer el estado de los 4 pulsadores del shield BASIC I/O y reflejarlo sobre los 4 leds. No hay novedades.

4-1-3.- Timbre

Trata de emular un timbre electrónico. Cada vez que se acciona el pulsador D12 se emite un tono por el altavoz de BASIC I/O.

A diferencia del lenguaje de Arduino, en el ESP32 no existen las funciones **tone()** y **noTone()**, sin embargo, alguien ha creado la librería **Tone32.h** que las contiene. En este caso, además de los parámetros típicos como el pin de salida del tono, su frecuencia y su duración, hay que indicar el canal. El ESP32 tiene 16 canales PWM, del 0 al 15, que se pueden usar para, entre otras cosas, generar tonos. En nuestro ejemplo usamos el canal 0.

4-1-4.- Do Re Mi

Siendo capaces de generar un tono, también podremos generar cualquier melodía. Solo hace falta conocer las frecuencias de las diferentes notas. En la librería **Tone32.h** disponemos del fichero **pitches.h** que las contiene. Este ejemplo emite el sonido de las 7 notas principales: do, re, mi, fa, sol, la y si cada vez que se acciona el pulsador D12 de BASIC I/O.

4-1-5.- PWM

Enseña cómo se genera una señal PWM por cualquiera de los 16 canales que dispone el ESP32. En este caso se produce una señal PWM que regula el brillo del led rojo del shield BASIC I/O. Hay que seguir estos tres pasos con sus correspondientes funciones:

1. Se define el nombre del canal (CanalR), su frecuencia (1000) y los bits de resolución entre 1 y 16 (8):

ledcSetup(CanalIR, 1000, 8);

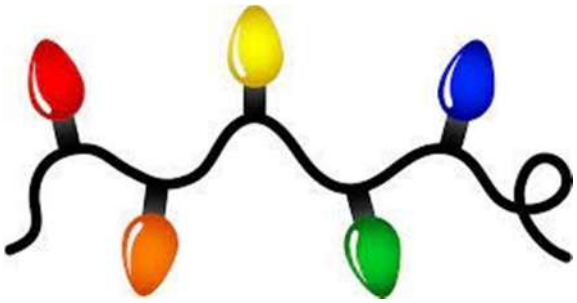
2. Se vincula ese canal (CanalR) con una de las patillas (LED_Rojo):

ledcAttachPin(LED_Rojo, CanalR);

3. La señal se produce indicando el canal deseado (CanalR) y la anchura del ciclo útil (ciclo):

ledcWrite(canalR, ciclo);

4-1-6.- PWM Aleatorio



Seguimos con las señales PWM. En esta ocasión vamos a hacer un juego de luces con los leds de BASIC I/O de forma que el brillo de los mismos varíe aleatoriamente. Se definen 4 canales con una frecuencia de 1000 Hz y 8 bits de resolución cada uno. Esos canales se conectan con los leds y, finalmente, mediante la función **random();** se obtiene el valor aleatorio del ciclo útil para cada canal.

4-1-7.- Servo

Para controlar un servomotor se emplean las mismas funciones que se usabann en Arduino, pero la librería que las contiene es "**ESP32Servo.h**". La debemos incluir en nuestros programas. El ejemplo hace que el servo se desplace hasta un extremo a intervalos de 5º y luego retorna directamente al extremo opuesto.

4-1-8.- Serial

Otro clásico… Cada vez que se acciona el pulsador D12 se envía un mensaje por el puerto serial del canal 1 del ESP32. Dicho mensaje se puede ver directamente mediante el monitor serie del IDE.

4-1-9.- Medir Pulso

El ejemplo trata de medir el tiempo que el pulsador D12 se mantiene activado y mostrarlo en el monitor serie del IDE.

4-1-10.- Monitor Remoto

Ejemplo práctico que permite conocer de forma remota el estado de unas entradas digitales como son los pulsadores de BASIC I/O, En este caso dicho estado se monitoriza en el monitor serie del IDE.

4-1-11.- Control Remoto

Otro ejemplo práctico. En esta ocasión se trata de controlar el estado de los leds del shield BASIC I/O. Usaremos el monitor serie del IDE para enviar los comandos '**r**', '**a**', '**v**', y '**b**'. ArduWiFi-32 recibe y decodifica esos comandos para que los leds rojo, ámbar, verde y blanco cambien de estado respectivamente.

4-1-12.- Servo Control

El servomotor conectado con D3 (GPIO19) se controla de forma remota, desde el monitor serie del IDE, desde donde se envía el número de grados que se desea girar. Conviene configurar al monitor serie del IDE con la opción de "**Sin ajuste de línea**".

4-1-13.- Conversión ADC

Para realizar una conversión con el ADC del controlador ESP32, se emplea la misma función **analogRead(canal)** que seguro conoces. Sin embrago recuerda que la resolución es de 12 bits en lugar de 10 como ocurre en Arduino UNO. El ejemplo realiza la conversión de canal A0 en el que está conectado el potenciómetro Volt1 de BASIC I/O, cada vez que se acciona el pulsador D12. En el monitor serie del IDE se visualiza el resultado de la conversión en decimal, en binario y en hexadecimal, así como la tensión equivalente.

Hay que indicar que ESP32 solo admite tensiones analógicas de entrada comprendidas entre 0 y 3.3. Por mantener la compatibilidad con Arduino UNO, la tarjeta ArduWiFi-32 permite tensiones de entrada comprendidas entre 0 y 5V, aunque para ello emplea unos divisores de tensión a base de resistencias cuyas tolerancias producen ciertos errores o falta de precisión en las conversiones. ¡¡ Nada es perfecto !!

4-1-14.- Dimmer

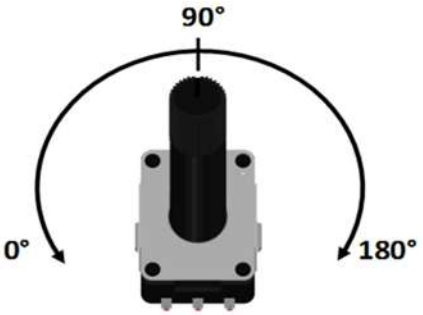
Tenemos los conocimientos suficientes como para hacer un control regulado de potencia o "*Dimmer*":

- Configurar y conectar un canal PWM.
- Realizar una conversión ADC del sensor de luz.
- Producir una señal PWM en función del sensor de luz.

El brillo del led blanco de BASIC I/O se regula mediante una señal PWM que varía en función de la luz ambiente que detecta el sensor de luz conectado en la entrada analógica A2. Cuanta más luz ambiente haya, menor es el ciclo útil de la señal PWM y por tanto el brillo del led es menor e incluso se apaga. Puedes usar una linterna para alumbrar directamente el sensor y comprobar el efecto. También puedes usar un potenciómetro en lugar del sensor de luz para hacer una regulación manual.

4-1-15.- Servo Control

Lo mismo que regulamos el brillo de un led, podemos regular el posicionamiento de un servomotor movimiendo un simple potenciómetro. Es lo que hacemos en este ejemplo. Según giramos el eje del potenciómetro Volt2 de BASIC I/O conectado en A1, el eje del servomotor conectado en D3 se desplaza entre 0 y 180º.



4-1-16.- Climatizador

En esta ocasión usamos el sensor de temperatura de la tarjeta BASIC I/O conectado en A4. Simulamos el funcionamiento de un termostato de forma que, cuando la temperatura está por debajo de un valor mínimo, se activa el led rojo a modo de

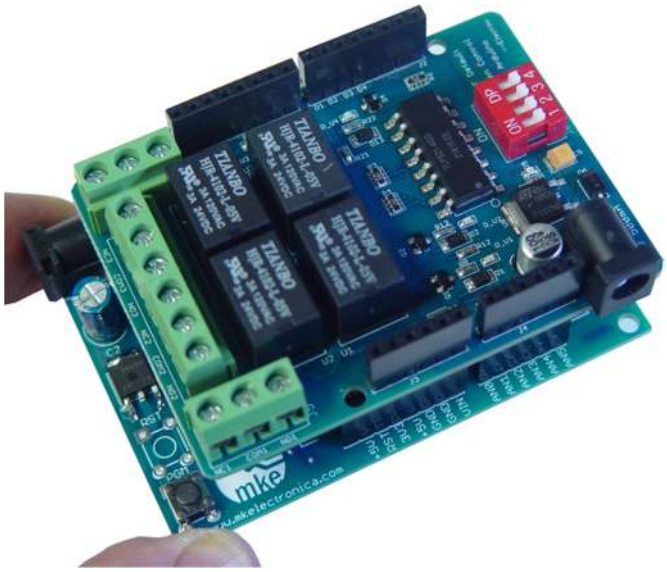
calefactor. Si está por encima de un valor máximo se activa el led verde simulando un aparato de refrigeración.

Como decíamos en el ejemplo 4-1-13 hemos tenido que sacrificar la precisión de las conversiones a cambio de la compatibilidad. Es por esto que en el ejemplo usamos un valor experimental para calibrar y mejorar las medidas a partir de una temperatura conocida.

4-1-17.- Relé ON/OFF

En este y en los siguientes ejemplos vamos a usar el shield de 4 relés MCS01584R: (https://mkelectronica.com/producto/shield-de-4-reles-para-arduino-y-compatibles/).

En esta ocasión simplemente vamos a activar/desactivar el relé 1.



4-1-18.- Secuencia de relés

No hay ninguna novedad. En este ejemplo se produce una determinada secuencia en el encendido y apagado de los cuatro relés del shield MCS01584R.

4-1-19.- Relés Remotos

Tampoco hay grandes novedades. Se trata de controlar de forma remota el estado de los 4 relés. Desde el monitor serie del IDE se envían los comandos '**1**', '**2**', '**3**' y '**4**'. ArduWiFi-32 recibe el comando y el relé correspondiente cambia de estado.

Es recomendable configurar el monitor serie del IDE con la opción de "**Sin ajuste de línea**".

4-1-20.- Saludos LCD

Ahora vamos a dedicar un par de ejemplos al shield MCS01602M y su pantalla LCD. En este ejemplo emplearemos la conocida librería "**LiquidCrystal.h**" que contiene diversas funciones relacionadas con el manejo de una pantalla LCD. Usaremos algunas de esas funciones para visualizar un saludo desde MK Electrónica.

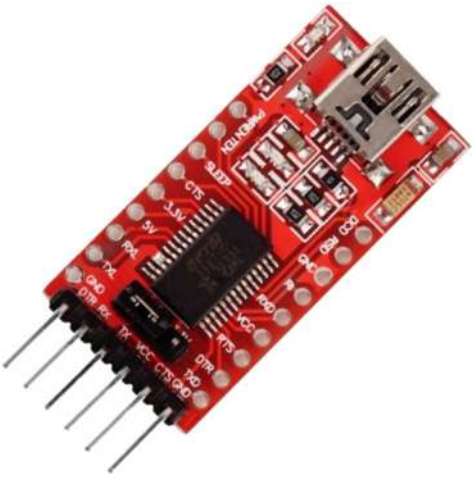
4-1-21.- Pulsadores LCD

El Shield MCS01602M además de la pantalla LCD dispone de 5 pulsadores que muy bien pudieran servir como entradas para desplazarnos sobre un hipotético menú visualizado en la pantalla: SELECT, LEFT, UP, DOWN y RIGH, o desencadenar diferentes acciones cuando se detecta la activación de cualquiera de ellos.

Los pulsadores están conectados a una única entrada analógica, la A0. Gracias a un divisor resistivo, el convertidor ADC recibe una tensión analógica que varía en función del pulsador accionado. Esto nos permite determinar y visualizar cuál de ellos se ha activado.

4-1-22.- UART

Con este ejemplo se demuestra que el ESP32 tiene dos canales serie tipo UART totalmente independientes entre sí. El canal 1 es el que venimos usando hasta el momento para la descarga de los programas a través del puerto USB y la comunicación con el monitor serie del IDE. El canal 2, a diferencia de Arduino UNO, está totalmente disponible en las patillas D0 (Rx) y D1 (Tx) y no necesita de librería alguna.



Este ejemplo conecta ambos canales formando un "*puente*" de manera que, lo que se transmite por el canal 1 (el IDE), se recibe en el canal 2 y viceversa. Para probarlo necesitas conectar en el canal 2 un adaptador UART a USB entre las patillas D0 y D1 (Rx y Tx), así como un software de comunicaciones como puede ser SSCOM32 o cualquier otro. Para el canal estándar, el 1, basta con el propio monitor serie del IDE.

4-1-23.- Scan I2C

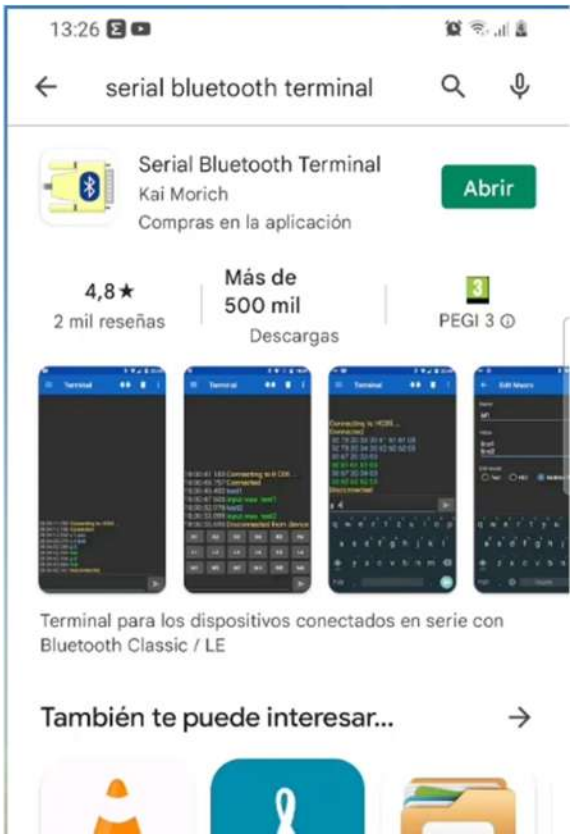
Acabamos esta serie de ejemplos básicos. Se trata de un ejemplo experimental que permite comprobar la comunicación con dispositivos I2C. Necesitas uno de ellos y conectarlo con las señales SDA y SCL de la tarjeta ArduWiFi-32. El programa averigua cual es la dirección del dispositivo conectado.

Importante!! Las señales I2C son en colector abierto y compatibles con 3.3 V.

4-2 Ejemplos con Bluetooth

Ya se ha explicado que el controlador ESP32, y por tanto ArduWiFi-32, tiene conectividad WiFi. Esto permitirá comunicarnos por radio frecuencia (RF) con cualquier dispositivo que emplee esta misma tecnología: Smartphone's, tablet's, PC's e infinidad de dispositivos.

El objetivo de este manual es simplemente proponer una serie de ejemplos que permitan la comunicación entre ArduWiFi 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.



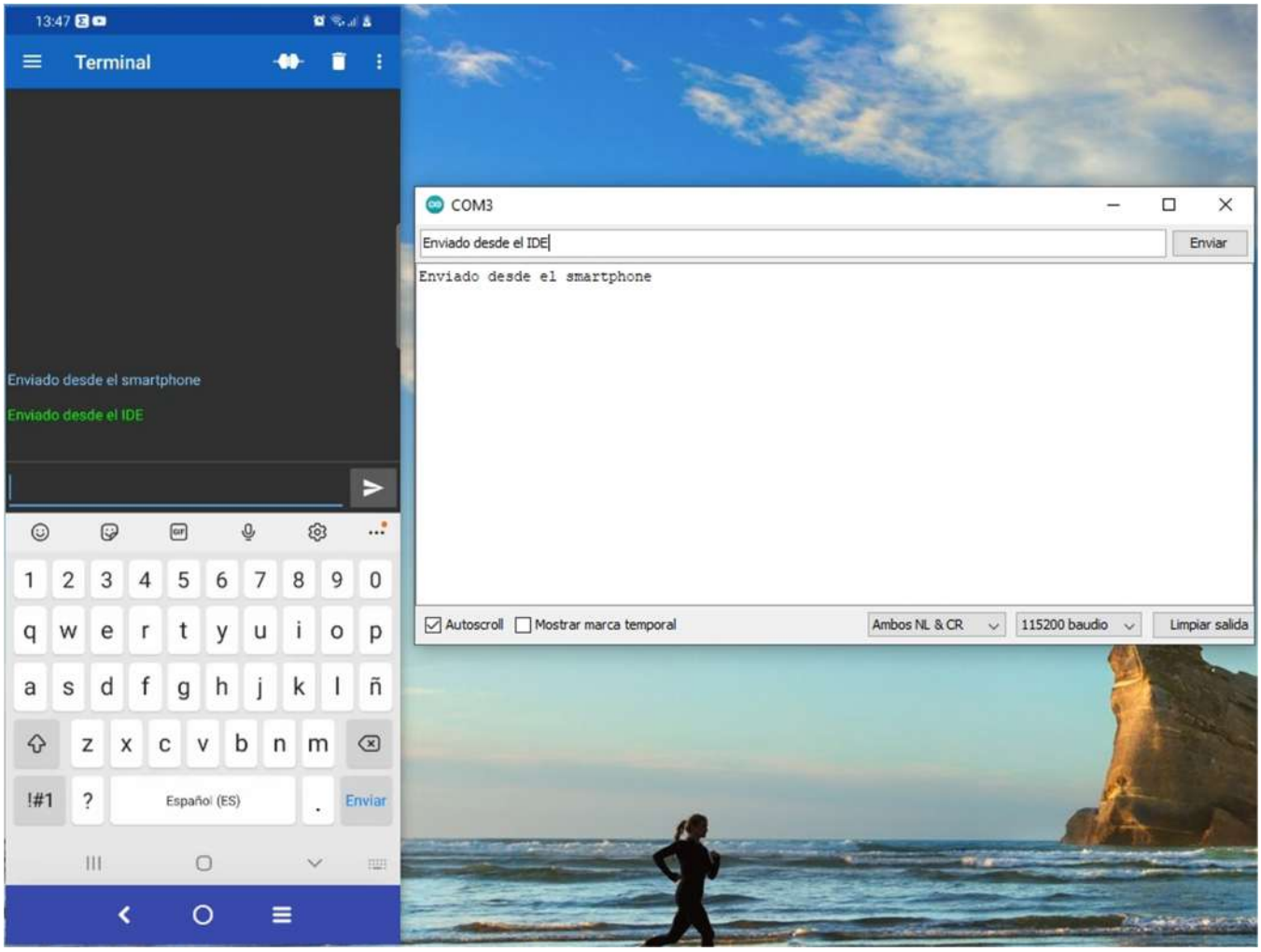
Para estos ejemplos hemos usado un Smartphone dotado de Bluetooth y al que le hemos instalado la aplicación gratuita "**Serial Bluetooth Terminal**" que puedes descargar desde Play Store. Nos gusta mucho y funciona muy bien, pero hay muchas más. Tú eliges.

En los ejemplos a nuestro dispositivo, la controladora ArduWiFi-32, le hemos denominado "**MK-ELECTRONICA**". Puedes 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.

4-2-1.- Comunicación BT

Es el más sencillo. Realiza una conexión entre la app de comunicaciones en el Smartphone remoto y ArduWiFi-32. Lo que se envía desde el primero se visualiza en el monitor serie del IDE y viceversa.

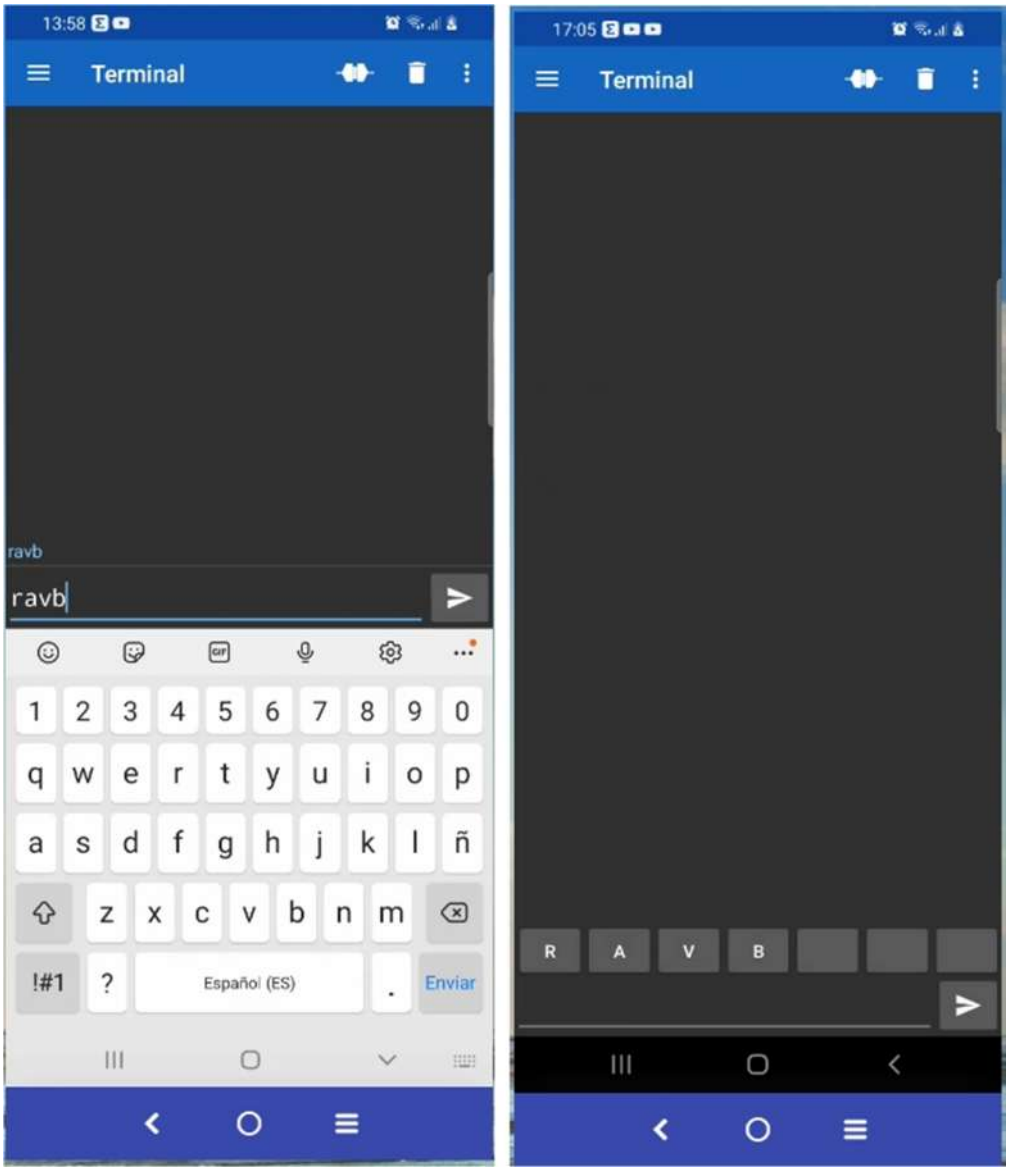
Empleamos la librería "**BluetoothSerial.h**" cuyas funciones facilitan enormemente la conexión y comunicación con cualquier dispositivo remoto. Se instaló durante el proceso de instalación que vimos en el apartado 3. En estos ejemplos usaremos las funciones más importantes, no obstante sería interesante que buscaras toda la información disponible relativa a la misma y sus funciones.



Observa que hay una comunicación bidireccional e **INALAMBRICA** entre el dispositivo remoto, el Smartphone, y la tarjeta controladora ArduWiFi-32 mediante el monitor del IDE.

4-2-2 Control Remoto

A partir de aquí las posibles aplicaciones y proyectos dependen de tu imaginación. En este ejemplo controlaremos el estado de los leds de la tarjeta BASIC I/O desde nuestro móvil. Para ello enviaremos de forma remota y sin cables los conocidos comandos '**r**', '**a**', '**v**', y '**b**', y además… puedes crear botones con macros que se encarguen de enviar lo que desees con una simple pulsación. Mira esta imagen.



A la izquierda enviamos los comandos tecleándolos en el terminal. A la derecha hemos creado 4 botones, R, A, V y B, que cada vez que los pulsas envían, en este ejemplo, los comandos '**r**', '**a**', '**v**', y '**b**' respectivamente. Es una opción muy interesante y cómoda. Te invito a que explores las posibilidades de esta app.

4-2-3 Entradas digitales

En este ejemplo, nuestro Smartphone remoto va a visualizar el estado de los pulsadores de la tarjeta BASIC I/O. Cada vez que uno de ellos cambie de estado ArduWiFi-32 transmite el nuevo valor ON/OFF.

4-2-4 Entradas analógicas

De la misma manera que monitorizamos las entradas digitales de los pulsadores de BASIC I/O, en el Smartphone también podemos monitorizar sus entradas analógicas. Es lo que hacemos en este ejemplo. Usamos los comandos '**0**', '**1**', '**2**', '**3**' y '**4**' . Cuando ArduWiF32 los recibe, responde con el valor actual de las entradas analógicas A0 – A4 respectivamente.

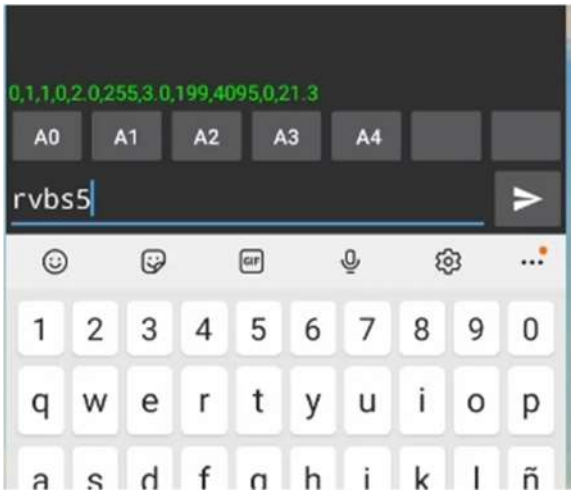
En la izquierda los comandos los introducimos mediante el teclado del terminal del Smartphone. A la derecha se aprecia la creación de 5 botones: A0, A1, A2, A3 y A4. Cuando se accionan envían automáticamente los comandos '**0**', '**1**', '**2**', '**3**' y '**4**' respectivamente.

4-2-5 Estado BASIC I/O

En este ejemplo basta con enviar cualquier carácter para que ArduWiFi-32 devuelva el estado tanto de las entradas digitales como de las analógicas, que son mostradas en el Smartphone remoto.

4-2-6 BASIC I/O

Este es el último ejemplo con la tarjeta BASIC I/O. Se trata de tener un control remoto total de la misma. Mediante los comandos '**r**', '**a**', '**v**', y '**b**', se controla el estado de los leds. El comando '**s**' sirve para producir un tono sobre el altavoz y el comando '**5**' para obtener es estado de las entradas digitales y analógicas separados entre sí mediante ','.



En la imagen hemos enviado los comandos 'rvbs5' mediante el teclado del terminal del Smartphone, lo que origina que los leds rojo, verde y blanco cambien de estado, se genere un sonido en el altavoz y ArduWiFi-32 devuelva una cadena o string con el estado de las entradas y salidas digitales separados por ','. Esta se puede decodificar para determinar que (en el ejemplo):

- Los pulsadores: D12=OFF, D8=ON, D7=ON y D6=OFF
- A0 = 2.0V y el eje de del potenciómetro Volt1 está en la posición 255
- A1 = 3.0V y el eje de del potenciómetro Volt2 está en la posición 199
- El sensor de luz vale 4095.
- El sensor IR vale 0
- La temperatura es de 21.3ºC

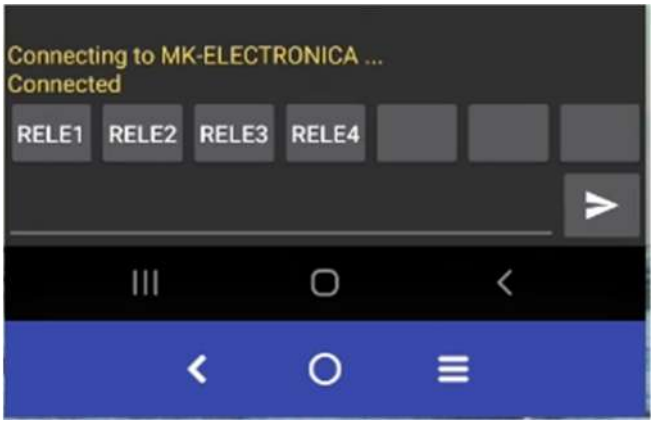
También puedes descargar desde Play Store la aplicación gratuita "**Basic I/O monitor**". Básicamente hace lo mismo que lo que hacemos con la aplicación "**Bluetooth Serial Terminal**" pero de una forma mucho más intuitiva. Muestra una representación de la tarjeta de periféricos BASIC I/O.

Cada vez que tocamos los leds D11, D10, D9 y D6 la aplicación transmite los comandos '**r**', '**a**', '**v**', y '**b**' respectivamente, lo que origina que el led correspondiente en la tarjeta BASIC I/O cambie de estado.

Si tocamos en el Speaker se envía el comando '**s**' y ArduWiFi-32 genera un tono sobre el altavoz de BASIC I/O.

Cada cierto tiempo y de forma periódica, el Smartphone envía el comando '**5**' con lo que ArduWiFi-32 devuelve el estado de las entradas digitales y analógicas. En la imagen se puede ver que:

- Los pulsadores D12 y D8 están en ON.
- V1 = 2.0V y V2 = 5.0V
- El sensor de luz = 417
- El sensor IR = 112
- La temperatura es de 22.3ºC



4-2-7 Control Relés

No hay grandes novedades. En esta ocasión se trata de controlar los cuatro relés del shield MCS01584R mediante el envío de los comandos '**1**', '**2**', '**3**,' y '**4**'. En la imagen hemos creado sendos botones para ello con la app "**Bluetooth Serial Terminal**"

4-2-8 Control LCD

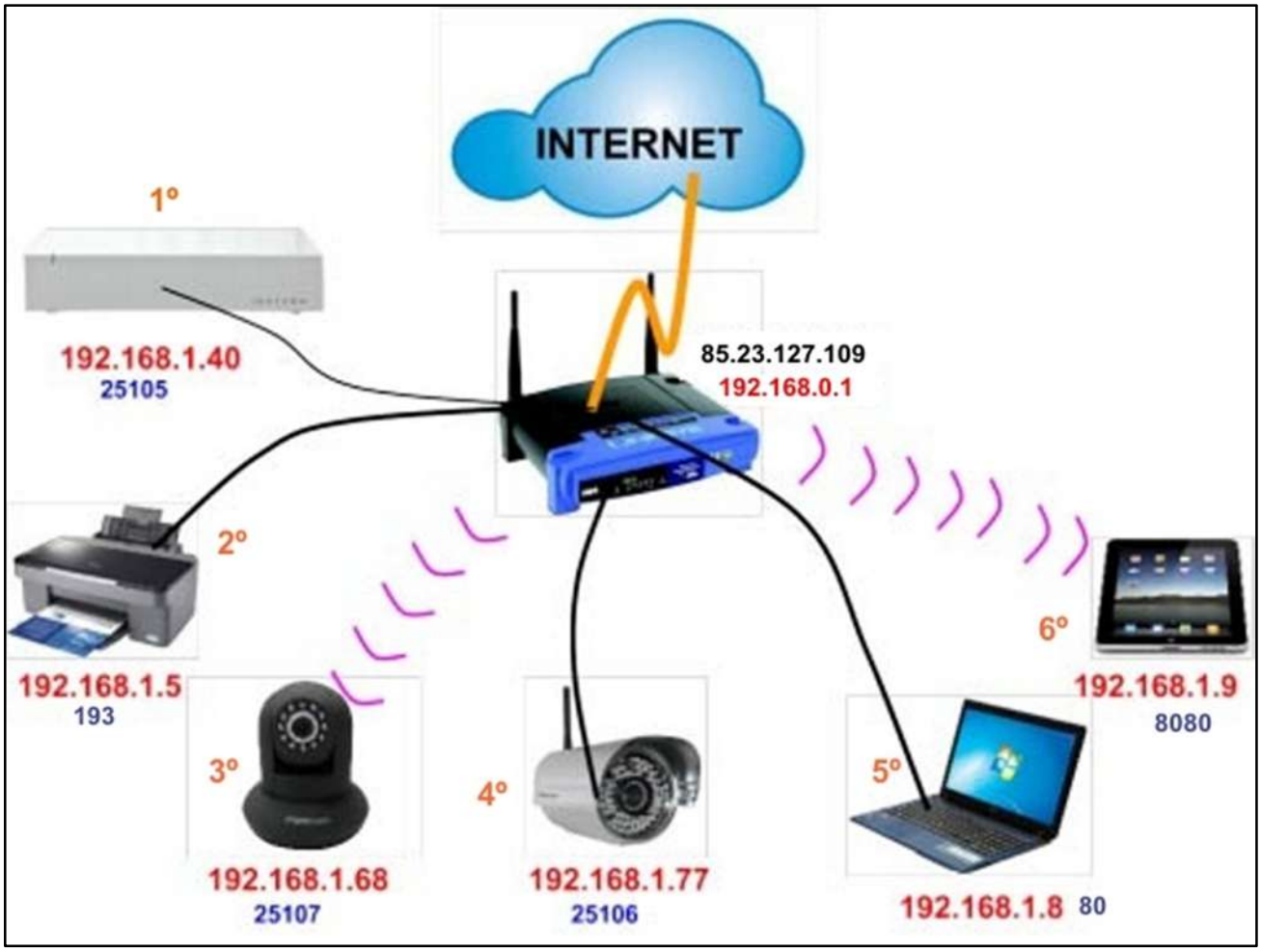
Como último ejemplo dedicado a la comunicación Bluetooth, vamos a enviar mensajes de texto desde nuestro Smartphone remoto. Estos serán recibidos en la tarjeta ArduWiFi-32 y mostrados sobre el shield LCD MCS01802M.

4-3 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 conectes /desconectes a 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 conoces 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**".



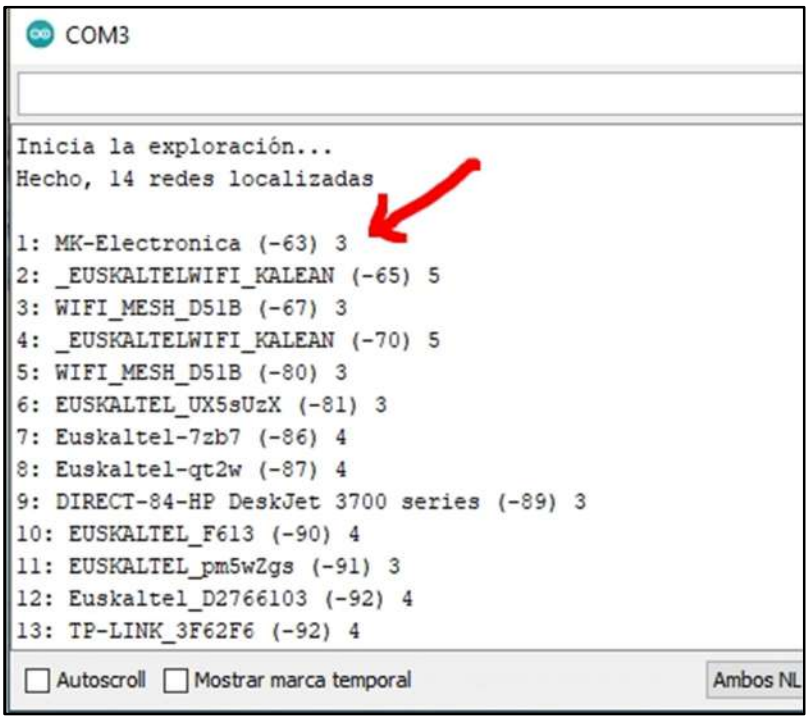
También hay que hablar de la existencia de empresas que proporcionan plataformas que facilitan la interacción entre el usuario y el dispositivo de una forma sencilla, intuitiva y amigable. Algunas de estas empresas son: Thingspeak, Blynk, Adafruit IO, Amazon web, Arduino Cloud IoT, y muchas más.

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/)

4-3-1 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 nuestra tarjeta ArduWiFi-32 y su controlador 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.

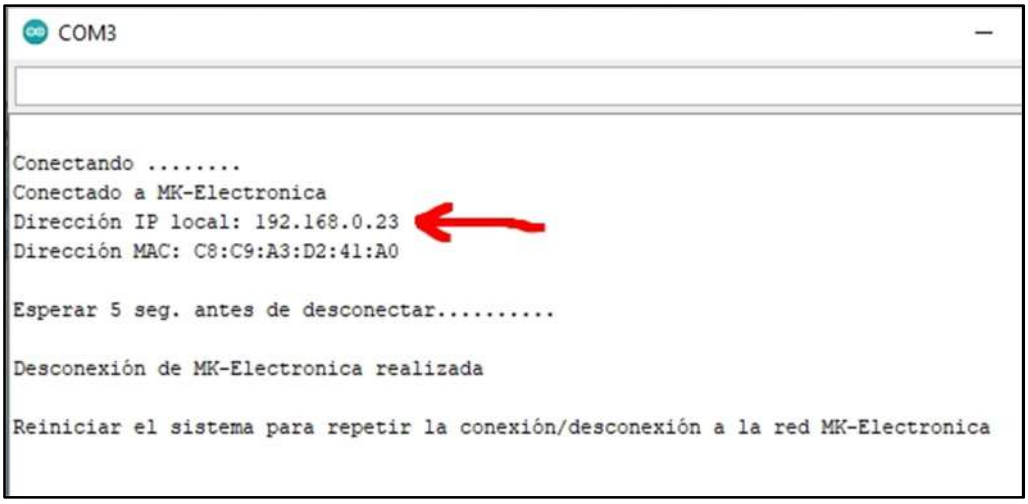
Empleamos la librería "**WiFi.h**" que también se instaló en el proceso de instalación y configuración explicado en el apartado 3. Usaremos las más importantes, pero sería deseable que buscaras información de todas las funciones contenidas en esta librería.

4-3-2 Conexión Red WiFi

El ejemplo nos enseña cómo ArduWiFi-32 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 asigna el router. Es la que usaremos cada vez que queramos conectar con ArduWiFi-32. En mi caso ha sido la IP = 192.168.0.23, en tu caso seguro que es otra. **Toma nota de ella**. Si ahora desconectamos ArduWiFi-32 de la red y conectamos otro dispositivo, por ejemplo, un móvil, seguro que al volver a conectar ArduWiFi-32 el router asigna otra dirección diferente. Por eso se suele hablar de IP dinámica.

El ejemplo también nos muestra la dirección MAC del dispositivo. Esto es un código único en el mundo que identifica al dispositivo, en este caso al ESP32 de la tarjeta ArduWiFi-32. A partir de su MAC ciertos ruters te permiten asignar a ese dispositivo una dirección IP fija. Investiga si el tuyo lo permite.

Finalmente, cinco segundos después ArduWiFi-32 se desconecta de la red.



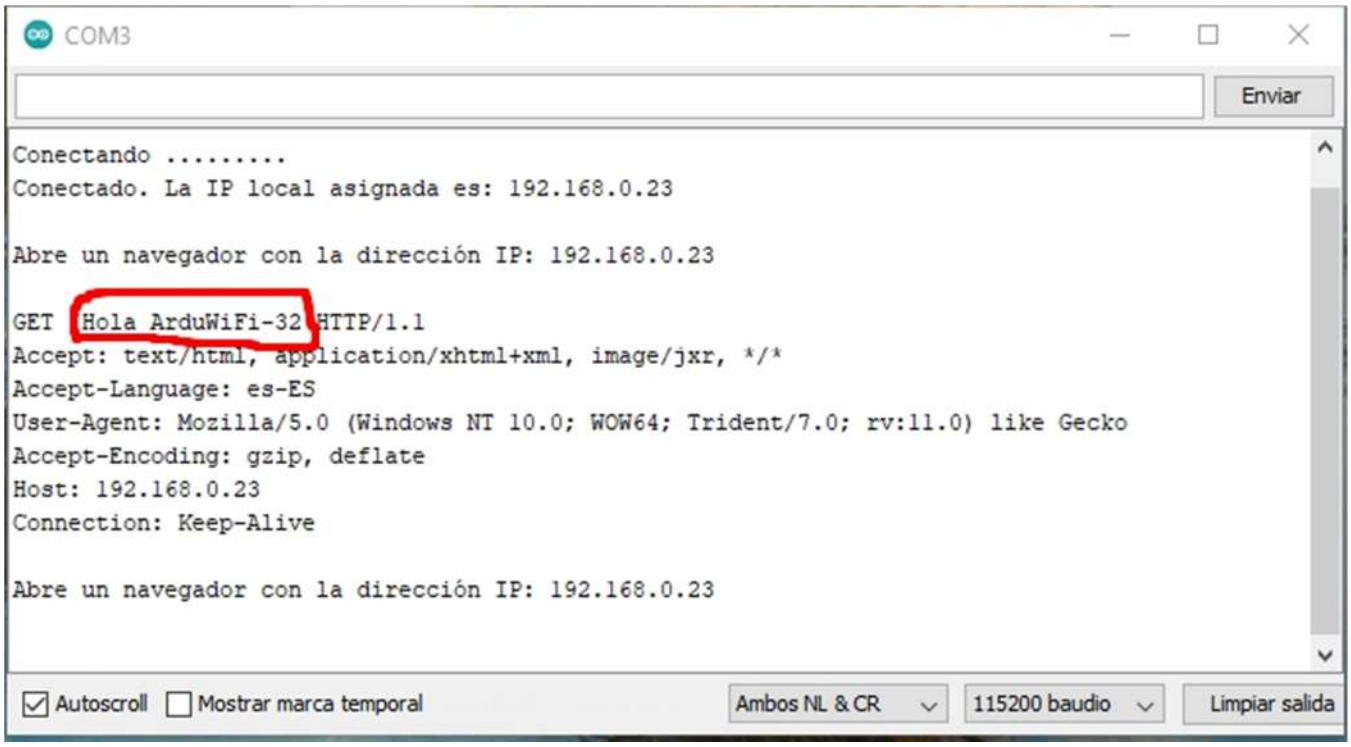
4-3-3 Servidor

En este ejemplo vamos a transferir datos entre un navegador como es el Internet Explorer (el cliente) y nuestra tarjeta ArduWiFi-32 (el servidor).

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 (ArduWiFi-32) responde con el mensaje "**HTTP/1.1 200 OK**".
6. Se cierra la conexión.

Para probarlo basta con abrir un navegador (Internet Explorer), indicar la dirección IP que el router asignó a nuestra tarjeta controladora ArduWiFi-32 y escribir el mensaje que quieres enviarle. En la imagen el cliente escribe: **192.168.0.23/Hola_ArduWiFi-32**. Es lo que se conoce como una "**petición cliente → servidor**".

En la siguiente imagen aparece todo lo que recibe ArduWiFi-32 como consecuencia de esa petición por parte del cliente: lenguaje, tipo de navegador, sistema operativo, versiones, etc… De todo ello lo más importante para nosotros se encuentra en la 1ª línea, justo tras la palabra "**GET**"", el mensaje: "**Hola_ArduWiFi-32**".



Resumiendo, desde un navegador instalado en un PC, Smartphone, Tablet, o similar hemos conseguido mandar un mensaje a ArduWiFi-32 usando la red WiFi.

Si abrimos un puerto en el router, por ejemplo el 80, y lo asociamos a la IP local de nuestro dispositivo (ArduWiFi-32), y en el navegador escribimos por ejemplo…

192.168.0.23:80 / Hola_ArduWiFi-32

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

4-3-4 Ejecutar Comandos

Si sabemos mandar mensajes desde un navegador (el cliente) hasta nuestra tarjeta ArduWiFi-32 (el servidor), ya podemos empezar a hacer muchas cosas. Por ejemplo, podemos mandar comandos para actuar sobre los periféricos de BASIC I/O. Es lo que hacemos en este caso.

Desde el navegador el cliente manda la petición con uno de los comandos: '**r1**' / '**r0**' para encender o apagar el led rojo de BASIC I/O, '**a1**' / '**a0**`' para el led ámbar, '**v1**' / '**v0**' para el verde y '**b1**' / '**b0**' para encender o apagar el led blanco.

ArduWiFi-32, el servidor, recibe la petición. De todas las líneas recibidas se queda con la primera. Es la que nos interesa ya que transporta los comandos. Lugo la analiza para ver si contiene uno de ellos y actúa sobre el led correspondiente activándolo o desactivándolo. Por ejemplo, en mi caso…

- **192.168.0.23/r1 → Activa led rojo**
- **192.168.0.23/b0 → desactiva led blanco**
- **192.168.0.23/a1 → Activa led ámbar**

4-3-5 Uso de botones

Podemos mejorar el interface con el usuario. En este ejemplo, cuando ArduWiFi-32 recibe una petición desde el cliente (navegador), le devuelve una sencilla página web que muestra 4 botones, uno por cada led de BASIC I/O. Cada vez que el cliente pulsa uno de ellos origina una nueva petición con un comando para que el servidor que la recibe (ArduWiFi-32) cambie de estado al led correspondiente. El resultado es el siguiente:

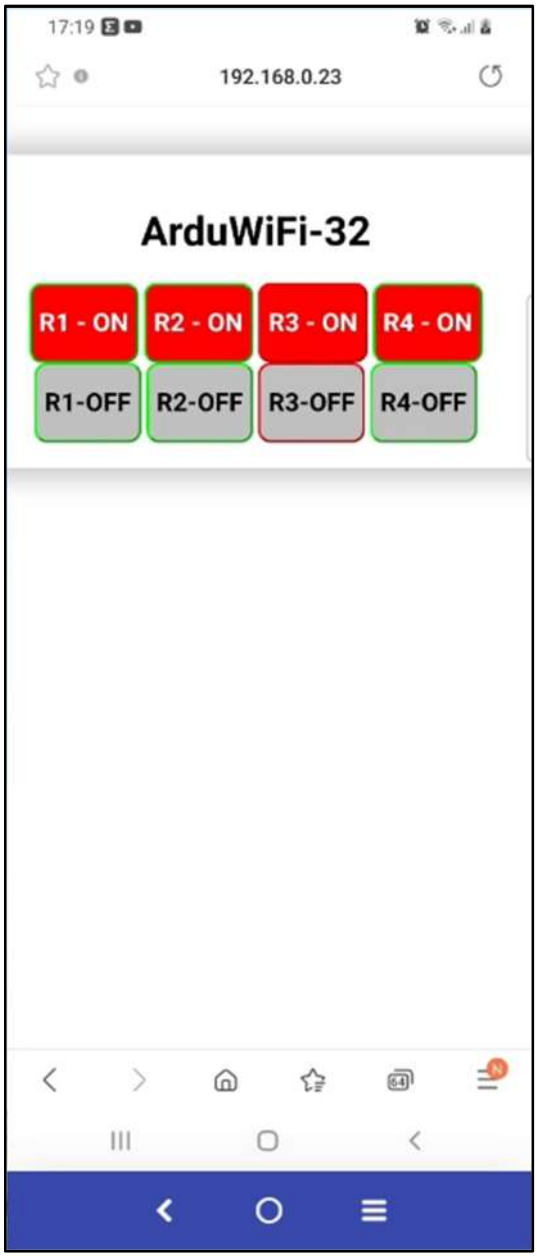
En este ejemplo tenemos un nuevo fichero auxiliar: "**paginaHTML.hpp**". Contiene el código que se envía al cliente y que define la página web que debe aparecer en su navegador. Está escrita en un lenguaje llamado HTML. Aunque en este caso ese código es bastante sencillo e intuitivo, te puedo asegurar que aquí tienes otro campo al que dedicar tus estudios: el diseño de páginas web.

Por supuesto que también puedes utilizar el navegador de tu Smartphone. De igual manera indicas la IP local de ArduWiFi-32 y recibirás la misma página desde la que podrás controlar los mismos leds.

Te recuerdo que estas usando tu red WiFi interna para establecer la comunicación entre el PC / Smartphone / Tablet y la controladora ArduWiFi-32 con su tarjeta BASIC I/O. Si quisieras controlar esos mismos leds desde cualquier lugar del mundo, tendrás que indicar tu dirección pública para que la información llegue al router y este la reenvíe al puerto asociado a la dirección IP local de tu ArduWiFi-32, y que previamente has tenido que abrir.

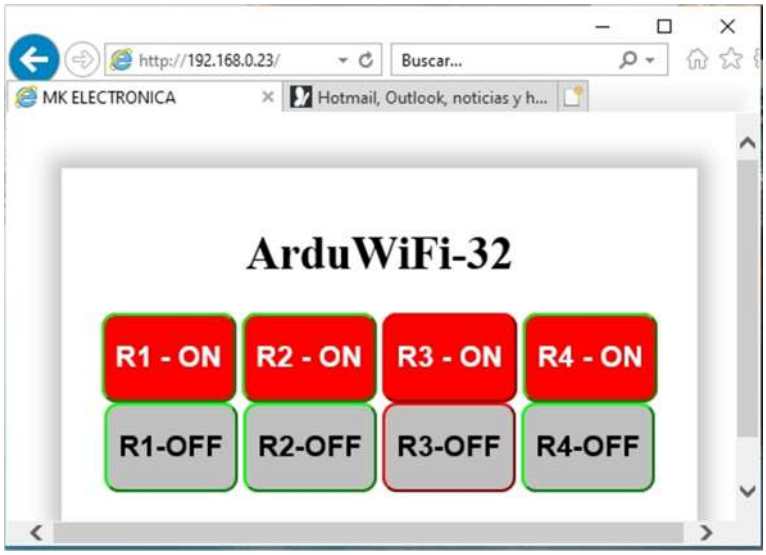
4-3-6 Control Relés

Las páginas escritas en el lenguaje HTML que el servidor manda al cliente pueden ser tan complejas como quieras o como sepas. En este caso, cada vez que el cliente envíe una petición, el servidor va a enviar una página con 8 botones con los que se podrá activar o desactivar cada uno de los 4 relés del shield MCS01584R que hemos usado en ocasiones anteriores.



Cada vez que se pulsa uno de esos botones se realiza una nueva petición con uno de los 8 comandos posibles: Rx0 para desactivar o Rx1 para activar, donde x es el número del relé (1 a 4).

En las figuras puedes ver el resultado desde el navegador Internet Explorer en un PC y en el navegador de un Smartphone.



4-3-7 Medir Tensión

No solo podemos controlar periféricos digitales del tipo led o tipo relé. En este ejemplo el cliente va a solicitar desde el navegador que ArduWiFi-32, el servidor, le envíe el valor actual presente en el potenciómetro Volt1 de BASIC I/O conectado en la entrada analógica AN0. También le envía la tensión equivalente.

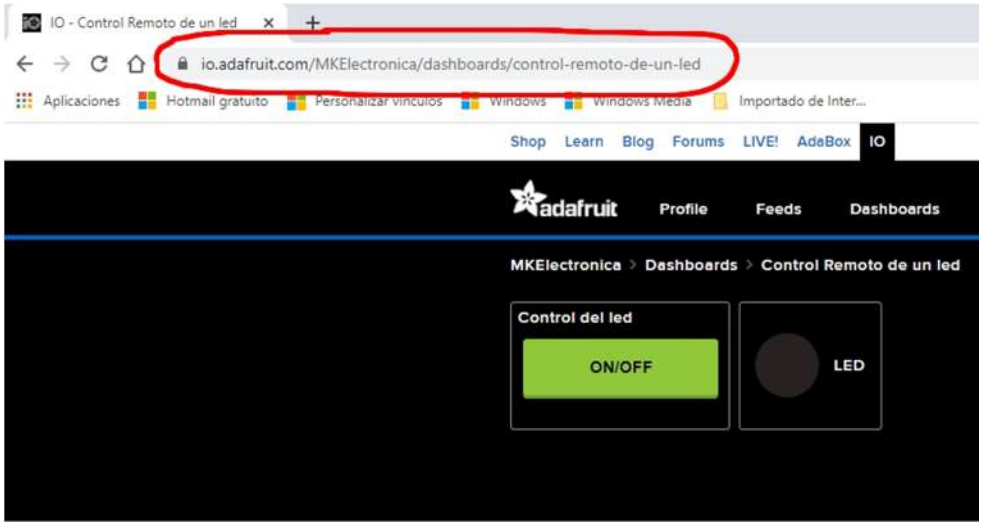
4-3-8 Led Remoto

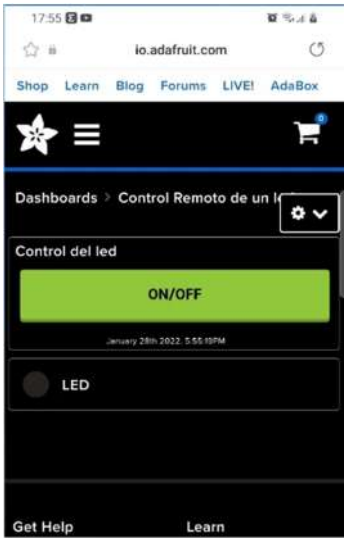
En este y en los siguientes ejemplos vamos a hacer algo parecido a lo anterior, pero empleando la plataforma Adufruit IO que nos permite crear un interface con el usuario más sencillo, potente e intuitivo.

La idea consiste en la transferencia de datos entre la plataforma y la controladora ArduWiFi-32. Esos datos van empaquetados en lo que se conoce como "*feeds*". Esos feeds están vinculados a diferentes tipos de bloques 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 dos bloques: un pulsador y un led. El primero está vinculado al feed '*pulsador*'. Cada vez que lo accionas se envía su estado a ArduWiFi-32 que lo analiza y activa/desactiva el led blanco de BASIC I/O. Inmediatamente después el estado lógico de ese led se envía de vuelta a la plataforma, a través del feed '*estadoLed*', que está vinculado al bloque led dibujado en el dashboard. Este se ilumina o no.

Hacer todo esto requiere conocer el funcionamiento de Adafruit IO. Para ello te 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 cualquier dispositivo móvil y desde cualquier lugar del mundo. El resultado para este ejemplo lo tienes en las siguientes figuras.

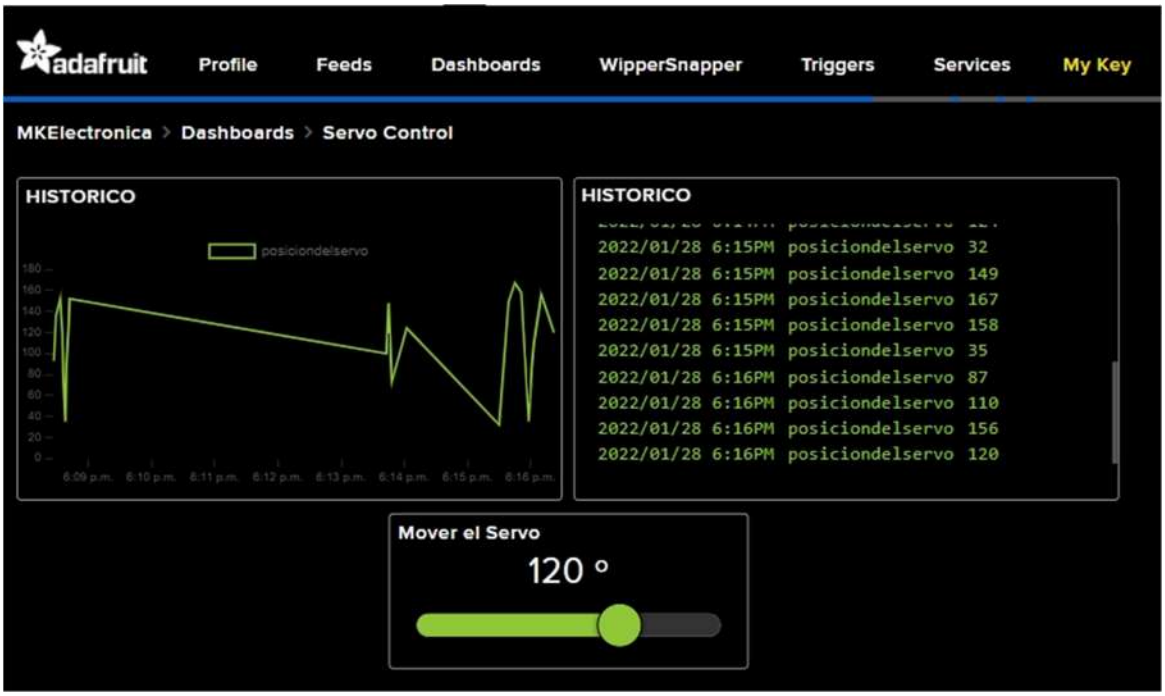




4-3-9 Servo Control

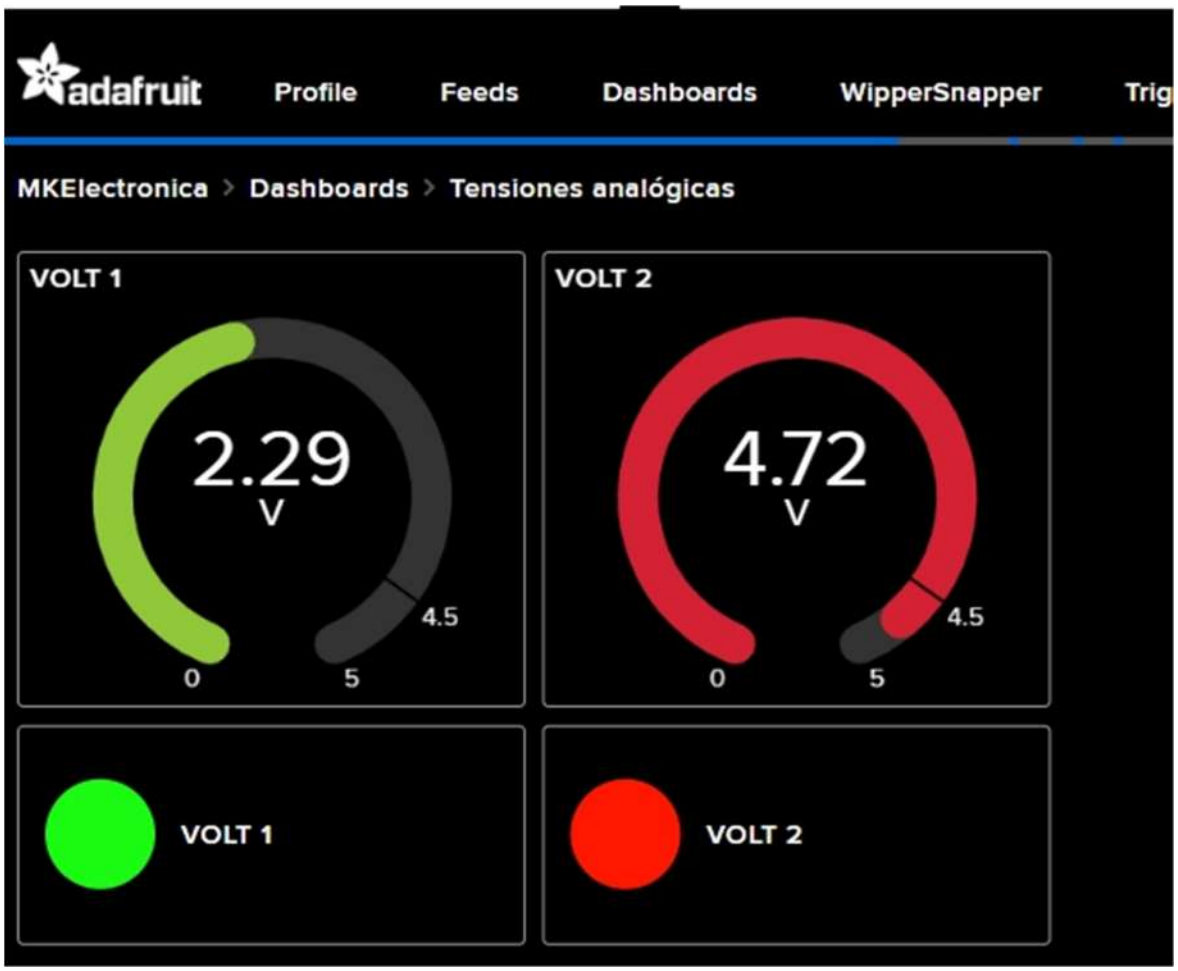
En esta ocasión vamos a mover el servomotor conectado en D3 de la tarjeta BASIC I/O. Para ello usaremos un panel de control o dashboard con un bloque de mando deslizante o slider asociado a un feed llamado '*posiciondelservo*'. Todo movimiento realizado con ese slider desde el PC o Smartphone remoto, se envía a ArduWiFi-32 que lo traduce en movimiento físico del servo.

En el mismo dashboard y asociados al mismo feed tenemos un bloque gráfico y otro de texto que irán mostrando el histórico de movimientos producidos.



4-3-10 Tensiones analógicas

En esta ocasión vamos a mostrar de forma remota las tensiones que producen los potenciómetros Volt1 y Volt2 de la tarjeta BASIC I/O conectados en AN0 y AN1.



Hemos creado un dashboard con cuatro bloques. A la izquierda tenemos un bloque indicador analógico o "*gauge*" y un indicador luminoso. Ambos están asociados al feed '*volt1*' a través del cual la controladora ArduWifi-32 transmite el valor actual de la tensión del canal analógico AN0. Si esa tensión es superior a 4.5V el indicador analógico se pone en rojo y el luminoso se activa.

A la derecha tenemos dos bloques idénticos pero vinculados al feed '*volt2*', que recibe desde ArduWiFi-32, el valor de la tensión presente en el canal analógico AN1.

4-3-11 Control de relés

Desde la plataforma Adfruit IO vamos a controlar los relés del shield MCS01584R. En el dashboard hemos colocado cuatro bloques del tipo interruptor y asociados a los feeds '*rele1*', '*rele2*', '*rele3*' y '*rele4*'. Cuando un interruptor cambia de estado, la plataforma lo transmite a través del feed correspondiente. ArduFiFi-32 recibe ese nuevo estado y actúa sobre el relé físico.

4-3-12 Enviar texto

Vamos con el último ejemplo. Ahora vamos a usar el shield MCS01602M con la pantalla LCD para visualizar los mensajes que le llegan desde la plataforma Adafruit IO.

Hemos usado dos bloques. Arriba hemos puesto el bloque texto. Permite introducir el texto que queremos enviar. Abajo hemos puesto un bloque del tipo stream que permite visualizar cualquier texto.

Ambos textos están vinculados al mismo feed llamado '*texto*', por lo que el bloque stream siempre visualiza lo que escribimos en el bloque texto. El contenido de ese feed llega hasta ArduWiFi-32 que lo recoge y visualiza sobre la pantalla LCD del shield.

Recuerda que todo esto lo puedes hacer desde cualquier lugar del mundo usando cualquier dispositivo con conexión a internet y un simple navegador para acceder a la plataforma. Adafruit IO es una plataforma sencilla de manejar y potente, pero hay muchas más. Te recomiendo que investigues sobre ellas porque su uso te facilitarán el diseño de aplicaciones.

Y así llegamos al final de este manual de usuario de la controladora ArduWiFi-32. Con estos ejemplos simplemente se pretende que veas sus casi ilimitadas posibilidades ya que disponemos de un potente controlador con capacidad de conexión tanto bluetooth como WiFi. Todo en uno !!