Programa Código Escuela 4.0

Arduino Wi-Fi en redes locales.

Índice

Introducción

En este documento vamos a trabajar **programas sencillos** para que el alumnado empiece a manejar la tarjeta **Arduino R4 WiFi**.

Programaremos la tarjeta para que actúe como servidor, utilizando directamente el IDE de Arduino para generar el HTML. La conexión se realizará a través de redes locales: **tanto el portátil como la tarjeta Arduino debe estar conectados a la misma red para que puedan interactuar**.

Los programas **se pueden copiar de este documento y pegar directamente en el IDE de Arduino**. Al hacerlo hay que tener cuidado, ya que puede suceder que el texto explicativo (lo que va a continuación de // en letra gris) se cambie de línea y el IDE lo interprete como código, dando errores de compilación.

Este documento puede servir para repasar y profundizar en conceptos como DHCP, Backend, ...

Conectando a Wi-Fi

Para poder conectar Arduino necesitamos una red con un **protocolo de seguridad WPA2** (es decir, red con contraseña). No se puede conectar a redes como WEDU que acceden a través de un usuario.

También es necesario usar **redes de 2.4 GHz**, no se podrán usar las de 5 GHz.

Las pizarras digitales de la dotación de la CAM permiten realizar esta conexión usando el HOTSPOT que encontramos dentro de la configuración de internet.

Las pizarras digitales suelen permitir entre 15-20 conexiones. Hay que tener en cuenta que necesitamos dos conexiones (IPs) por cada grupo: una para la tarjeta Arduino y otra para el portátil que usaremos para manejarla. Si la clase es numerosa y se necesitan más conexiones se puede utilizar el móvil del profesor para completar las conexiones (cada móvil permite unas 5 conexiones).

Normalmente el tráfico de datos no es un factor limitante, pero si el alumnado utiliza la conexión para ver videos, jugar,... puede saturar la red.

Conectando a Wi-Fi. Cosas a tener en cuenta.

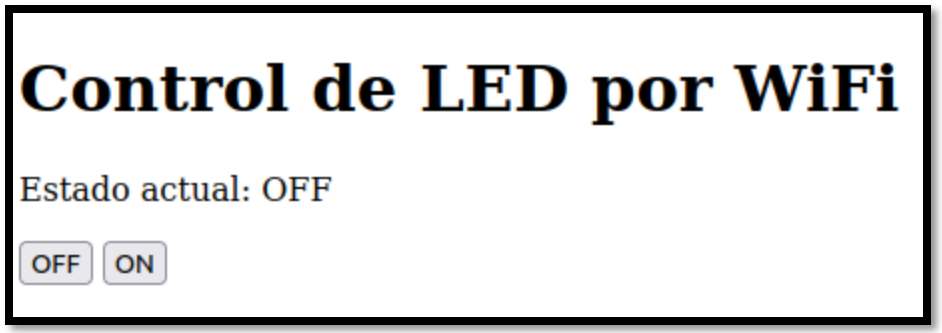
Para hacer programa el alumnado debe tener acceso al nombre y contraseña de la red. Por seguridad es recomendable usar el hotspot de la pizarra u otro dispositivo para poder cambiar la contraseña fácilmente o desactivar la red cuando se termine la práctica.

Cuando la pizarra asigna IPs a través del DHCP normalmente reserva esas direcciones durante un tiempo, de manera que, aunque ese dispositivo no la esté usando no estará disponible para conectar otros.

Es frecuente que sea necesario resetear la tarjeta Arduino para que podamos visualizar la IP en el puerto serie. Para algunas prácticas es conveniente que la tarjeta siga conectada al ordenador ya que se proporcionará información a través de su puerto serial.

Práctica I. Encender un LED.

Vamos a empezar haciendo el programa más sencillo posible. Nos va a permitir encender y apagar un LED de manera remota. Conectaremos el LED al pin 13 de manera que podamos usar el LED interno si lo deseamos.



Al realizar la programación nos conectaremos a una página web como la que aparece a la izquierda y desde la cual podremos encender y apagar el LED. Arduino está actuando como servidor.

Cuando presionamos el botón ON la página manda la petición a la tarjeta, que encenderá el LED y actualizará el estado. Análogamente podemos apagar el LED usando el botón OFF.

Recuerda que estamos trabajando con redes locales y para poder ver la página el portátil debe estar conectado a la misa red que la tarjeta Arduino.

Práctica I. Encender un LED.

Es importante tener en cuenta que **el flujo que sigue la información no es el mismo que seguiremos programando**.

Flujo de información

1. **Setup**: Configura LED, WiFi, servidor.
2. **Espera**: Arduino espera petición.
3. **Petición**: conexión, GET 0/ 1/
4. **Proceso**: decide ON/OFF
5. **Acción**: LED cambia estado
6. **Respuesta**: Envía web HTML

Flujo de programación

1. **Setup**: Configura LED, WiFi, servidor.
2. **Espera**: espera cliente en bucle, sale cuando recibe petición.
3. **Petición**: lee petición y decide si encender o apagar el LED, cambia el estado de la variable ON/OFF
4. **HTML**: programamos la página web que enviaremos posteriormente con el estado actualizando, incluyendo los botones con GET 0/ 1/ que envían la petición.

Para realizar de programación **primero debemos hacer la respuesta a la petición y luego crear los botones que generarán la petición** a través de la página web que se ha cargado en el dispositivo conectado a la red local de la pizarra.

Práctica I. Encender un LED.

El esquema de la programación que vamos a realizar es el siguiente:

1. Conectamos Arduino a WI-Fi y abrimos puerto de comunicación serie.
2. Respuesta a los botones (en este caso ON/OFF). Utilizaremos digitalWrite para encender y apagar el LED.
3. Página web usando HTML, incluyendo los botones.

Para crear un botón utilizaremos el código: `<a href=\"/0\"><button>OFF</button></a>`

En este caso el botón se llamará OFF (es lo que visualizaremos en la página web) y enviará una petición HTTP a Arduino.

```
if (request.indexOf("GET /0") >= 0)
```

Con esta línea comprobamos si la petición es de encendido o apagado y continuamos la programación en consecuencia.

Práctica I. Encender un LED.

En este enlace puedes descargarte un archivo tipo texto con el programa explicado paso a paso. Puedes copiar y pegar el programa directamente en Arduino y se debería compilar sin problemas.

- **Recuerda que debes introducir el nombre de red y la contraseña que te indique tu profesor.**
- **Una vez subido el programa a la placa ve al puerto serie para comprobar la IP.**
- **Es posible que tengas que reiniciar la placa (pulsador blanco al lado del cargador USB C).**
- **Teclea el número de IP que te aparece en el monitor serie en la barra del navegador.**
- **Recuerda que tu dispositivo debe estar en la misma red (WiFi) que la tarjeta Arduino.**
- **Comprueba su funcionamiento.**

Práctica II. Servo controlado por navegador.

Ahora vamos a utilizar lo aprendido en la práctica anterior para controlar un servo y que se mueva a 0º, 90º y 180º según el botón que pulsemos en la página web suministrada por Arduino.

Es importante que mostremos la posición del servo motor en el navegador para poder controlarlo remotamente. Para realizar la práctica necesitarás utilizar lo que ya sabes sobre cómo mover un servo con Arduino junto con lo que ya has aprendido sobre conexión WiFi y cómo programar el servidor.

Una vez realizada puedes modificarla para que el servo vaya avanzado de y retrocediendo con un paso de 20º. Puedes utilizar programaciones similares para controlar otros actuadores: los motores del coche-robot (si lo tenemos construido podremos manejarlo de manera remota), el anillo de LEDs, la matriz LED de la propria R4,...

En la siguiente página tienes indicaciones para ver los cambios que debes realizar en la programación.

Práctica II. Servo controlado por navegador.

Para realizar el programa correctamente debes seguir las siguientes indicaciones:

- Debes **definir el pin** del servo y cargar la librería `#include <Servo.h>` al comienzo de la programación e incluir `miServo.attach(pinServo)` en el Void SetUp.
- La conexión al puerto serie es igual a la práctica anterior.
- **Debes incluir 3 botones** en la página HTML. Para identificar cada caso la variable entera deberá tener **3 valores posibles** (de 0 a 2o de 1 a 3, como prefieras).
- Recuerda que debe ser coherente con el GET/ para que funcione correctamente.
- Puedes modificar el HTML para que sea más atractivo visualmente.
- Puedes usar la variable ángulo para almacenar y mostrar el estado a la vez que para mover el servo a la posición deseada.

Práctica III. Leer sensor desde el navegador.

El único sensor analógico incluido en el kit de Arduino es el de humedad de tierra. En el comparador tenemos indicada la salida A0 que es la que utilizaremos para obtener la lectura y que conectaremos al pin del mismo nombre.

Para realizar la práctica necesitarás utilizar lo que ya sabes sobre cómo realizar la lectura de un sensor analógico con Arduino junto con lo que ya has aprendido sobre conexión WiFi y cómo programar el servidor. La programación será análoga a las realizadas anteriormente y la página nos mostrará el valor de lectura cuando presionemos el botón del navegador.

Una vez realizada la práctica puedes modificarla para que la página muestre si hace falta regarla o no. Ten en cuenta que cuando la lectura se aproxima el cero quiere decir que la tierra conduce (es decir, mucha humedad) y viceversa.

En la siguiente página tienes indicaciones para ver los cambios que debes realizar en la programación.

Práctica III. Leer sensor desde el navegador.

Para realizar el programa correctamente debes seguir las siguientes indicaciones:

- Debes **definir el pin** al que está conectado el sensor (A0) e incluir que será un pin de entrada en el Void SetUp.
- Es conveniente crear una variable para almacenar el valor de la lectura analógica del pin 0.
- La conexión al puerto serie es igual a la práctica anterior.
- **Debes incluir 1 botón** en la página HTML.
- Recuerda que debe ser coherente con el GET/ para que funcione correctamente.
- Puedes modificar el HTML para que sea más atractivo visualmente.

Práctica IV. Leer sensor desde el navegador II.

Normalmente lo que queremos es que **la página muestre la lectura del sensor actualizada**, sin tener que apretar un botón ni recargar la página. La manera más habitual de conseguirlos es incluyendo un **script que actualice el valor de la variable**.

Un script es un pequeño programa que realizamos utilizando java y que podemos incluir en las páginas web para hacerlas más dinámicas e interactivas.

Para que el script funcione debemos coger el valor de la variable que nos da Arduino y convertirla a texto, almacenándola en una especie de variable (en realidad es un contenedor pequeño, una etiqueta de HTML). Le diremos dónde debe mostrar ese dato y haremos que lo actualice cada segundo. Además, debemos decirle a la página que cargue la función del script

Dado que es posible que todavía no hayas trabajado mucho con java puedes descargar el archivo directamente en este enlace y posteriormente probara a hacer pequeñas modificaciones, como que te indique cuando el valor es muy alto y es necesario regar las plantas

Práctica V. Backend I. Identificando la IP.

Vamos a enfocarnos ahora en la programación como servidor de la tarjeta Arduino. Utilizaremos de base el programa que hicimos para controlar el LED ya que es el más sencillo de realizar y comprobar su funcionamiento.

La **programación backend** de un servidor consiste en hacer la parte "oculta" de una web o app, donde se procesa la información. Se utiliza para guardar datos en bases de datos, gestionar usuarios o tomar decisiones antes de mostrar resultados al usuario.

En nuestro caso nos centraremos en proteger el funcionamiento de nuestro servidor y su garantizar su correcto funcionamiento. El primer paso es identificar las IPs que están haciendo las peticiones a la tarjeta.

En la siguiente página tienes indicaciones para ver los cambios que debes realizar en la programación.

Práctica V. Backend I. Identificando la IP.

Para realizar el programa correctamente debes seguir las siguientes indicaciones:

- La instrucción client.remoteIP() nos permite obtener la IP que está utilizando el dispositivo que realiza la petición
- Normalmente no queremos que está información aparezca en la página web, así que lo más lógico es mandarla por puerto serie. Esto quiere decir que debemos tener la tarjeta conectada al ordenador para poder verla.
- Recuerda incluir un pequeño texto que indique de que trata la información que va a mostrar, por ejemplo "Cliente conectado desde:"
- Puedes modificar el HTML para que sea más atractivo visualmente.

Práctica VI. Backend II. Bloquear una IP.

Vamos a enfocarnos ahora en la programación como servidor de la tarjeta Arduino. Utilizaremos de base el programa que hicimos para controlar el LED ya que es el más sencillo de realizar y comprobar su funcionamiento.

La **programación backend** de un servidor consiste en hacer la parte "oculta" de una web o app, donde se procesa la información. Se utiliza para guardar datos en bases de datos, gestionar usuarios o tomar decisiones antes de mostrar resultados al usuario.

En nuestro caso nos centraremos en proteger el funcionamiento de nuestro servidor y su garantizar su correcto funcionamiento. El primer paso es identificar las IPs que están haciendo las peticiones a la tarjeta.

En la siguiente página tienes indicaciones para ver los cambios que debes realizar en la programación.

Práctica VI. Backend II. Bloquear una IP.

Puedes descargar el código completo en este **enlace**. Para realizar el programa correctamente debes seguir las siguientes indicaciones:

- La instrucción `IPAddress` nos permite crear un objeto (instancia) que indica que el tipo de dato es una IP (4 digitos separados por punto). En este caso podemos llamarla ipBloqueada aunque cualquier nombre servirá siempre y cuando seamos coherentes en la programación.
- Antes del void setup () debemos incluir una línea del tipo: `IPAddress ipBloqueada(192,168,91,114);`
- Para que comprobar que el programa funciona debemos usar una de las IPs que está conectada a nuestro dispositivo (la puedes ver en el puerto serie).
- Para bloquear a la IP cuando se recibe una petición comprobamos con un condicional si la ip del cliente es igual que la ip bloqueada. En caso positivo debemos cerrar la conexión y hacer un **return** (sirve para **salir inmediatamente de la función loop()** y no seguir ejecutando el resto del código).

Práctica VII. Backend III. Bloquear por peticiones.

Es común que los servidores bloqueen una IP por realizar demasiadas peticiones. Es un método de seguridad que garantiza su correcto funcionamiento y lo protege de ataques maliciosos, cómo bots que piden información constantemente para saturarlo.

Haremos un programa sencillo basado en el anterior, de manera que cuando una misma IP hace más de dos peticiones en un segundo pasa a estar bloqueada, usando un código muy parecido al anterior.

Por simplificación sólo podemos bloquear una IP y el programa no funcionará correctamente si son varios dispositivos (distintas IP) las que realizan muchas peticiones. Puedes intentar modificar el programa para que sea más completo, pero ten en cuenta que la programación será bastante más larga y compleja.

En la siguiente página tienes indicaciones para ver los cambios que debes realizar en la programación.

Práctica VII. Backend III. Bloquear por peticiones.

Puedes descargar el código completo en este **enlace**. Para realizar el programa correctamente debes seguir las siguientes indicaciones:

- Debemos crear dos instancias tipo IPAddress : una para almacenar la ipBloqueada y otra para ultimaIP. También una variable tipo entero (**int**) para contar las peticiones y otra tipo **unsigned long** (número entero largo sin signo, siempre positivo) para almacenar el tiempo de inicio.
- Es necesario usar la función millis () que nos dice el tiempo de ejecución del programa.
- Cuando se realiza una petición, primero comprobamos y si la IP coincide con la bloqueada cerramos conexión. Después usamos millis () para establecer el tiempo actual.
- Sino es así comprobamos si la IPcliente es igual a ultimaIP. Después si el tiempo actual menos el tiempo de inicio es mayor de 1 segundo ponemos el contador de peticiones a cero y el tiempo inicial=tiempo actual y luego aumentamos el contador de peticiones.
- Si la IPcliente no es la misma a la actual, hacemos que IPcliente = ultimaIP, establecemos tiempo inicial a tiempo actual y ponemos el contador de peticiones a uno.

Práctica VIII. Backend III. Bloquear por conexiones.

En este caso vamos a limitar el número de dispositivos que se pueden conectar al servidor. Esta acción se realiza para garantizar que el servidor va a poder dar una respuesta adecuada a las peticiones.

Haremos un programa sencillo basado en lo aprendido en las prácticas anteriores y que nos permitía bloquear una IP, de manera que cuando un dispositivo se intente comprobar compruebe si es uno de los que están conectados o si hay sitio libre para conectarse.

Aunque normalmente utilizaríamos arrays (estructura de datos que permite guardar **varios valores en una sola variable**, organizados en una especie de lista)para configurar el programa por simplicidad utilizaremos variables parecidas a las usadas hasta ahora.

En la siguiente página tienes indicaciones para ver los cambios que debes realizar en la programación.

Práctica VIII. Backend III. Bloquear por conexiones.

Puedes descargar el código completo en este **enlace**. Para realizar el programa correctamente debes seguir las siguientes indicaciones:

- Debemos crear dos instancias tipo IPAddress : ip1 e ip2, ya que sólo permitiremos que se conecten dos dispositivos a nuestro servidor. También crearemos dos variables booleanas (**bool**): ocupada1 y ocupada2 que iniciaremos en false y que usaremos para saber si el número de dispositivos conectados.
- Al recibir una petición comprobaremos si la IPcliente corresponde a la ip1 y ocupada1 está en true permitiremos acceso, sino comprobamos lo mismo para la 2 .
- De no cumplirse ninguno de los casos anteriores, si ocupada1 está en fale (vacía) definimos esa IPcliente como la ip1 y permitimos acceso. Sino lo intentamos con la 2 y hacemos lo mismo.
- En el caso de que la IPcliente no se corresponda con ip1 o ip2 y esté ambas cogidas (ocupada1 y ocupada2 en true) bloquearemos el paso.