introduccion.py clase09.py

```
Programming ‘Language’ = {
    Introducción = [Python, MicroPython]

}
```

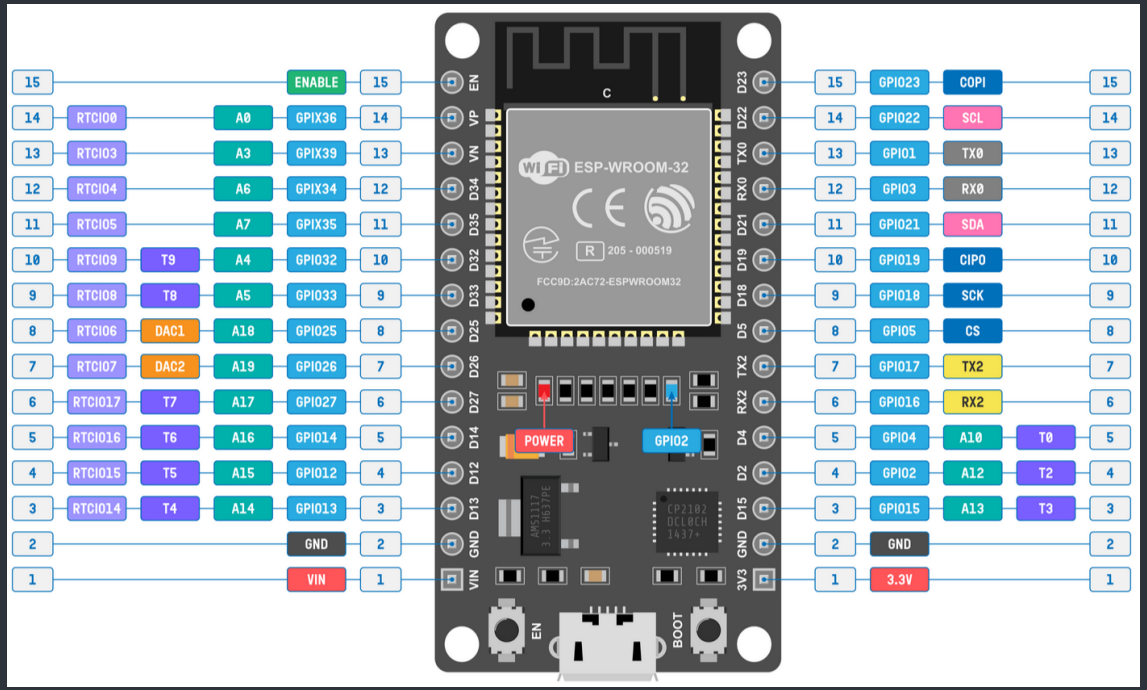
introduccion.py clase09.py

```
Clase 09 = {
    Presentación = [Les damos la bienvenida al curso]
}
```

```
Clases = {
```

- 👉 **Clase 09** **Estación. Punto de acceso. Web server.**
- Clase 10 Laboratorio 1: Control de LEDs y botones. Laboratorio 2: Monitorización de temperatura y humedad.

ESP32 DevKit V1 Pinout {



}

Wi-Fi {

Wi-Fi (Wireless Fidelity) es una tecnología de comunicación inalámbrica que permite la conexión de dispositivos electrónicos a una red de datos, como Internet, sin necesidad de cables físicos. Utiliza ondas de radio para transmitir y recibir datos entre dispositivos, como computadoras, teléfonos inteligentes, tabletas, y otros dispositivos habilitados para Wi-Fi, y un punto de acceso o router. Opera en bandas de frecuencia de 2.4 GHz y 5 GHz.

La banda de **2.4 GHz** ocupa el espectro desde **2.401 Ghz** hasta **2.485 GHz**. Cada canal tiene un ancho de 22 MHz. Es por ello que solo podremos usar solamente 3 canales de 20 MHz sin solaparse o 1 solo de 40 MHz.

La banda de **5 GHz** ocupa el espectro desde **5.180 GHz** hasta los **5.825 GHz**. Cada canal puede tener 20, 40, 80 y 160 MHz de ancho de banda. Soporta por ende 25 canales de 20 MHz, 12 canales de 40 MHz, 6 canales de 80 MHz y 2 canales de 160 MHz. Esta variedad nos permite descongestionar el espectro por lo que logramos conexiones mas estables.

}

Wi-Fi {

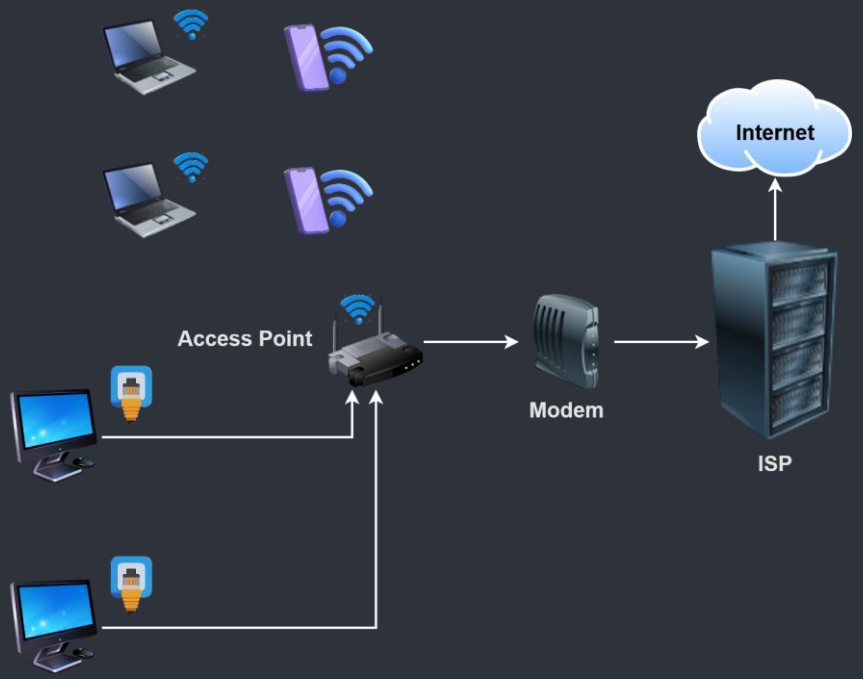
```
    # Algo muy importante que tiene Wi-Fi es que tiene estándares bien
    definidos que van a permitir una mayor compatibilidad entre dispositivos,
    como así también protocolos de seguridad establecidos para evitar que la
    información pueda ser interceptada fácilmente. Ejemplos de protocolos de
    seguridad son WEP, WPA y WPA2.

    # Dependiendo de la potencia del transmisor y receptor podemos obtener
    distancias de hasta 100 metros sin obstáculos.

    # Actualmente esta se plantea su reemplazo por Li-Fi pero con poco
    consenso internacional.
```

}

Red hogareña de Wi-Fi {

```
    # Un diagrama genérico de
    una conexión de red wifi
    puede verse de la siguiente
    manera.

    # Generalmente, nuestro ISP
    nos provee de Access Point
    y Modem en un solo equipo
    que solemos llamar
    habitualmente “Router”.
```

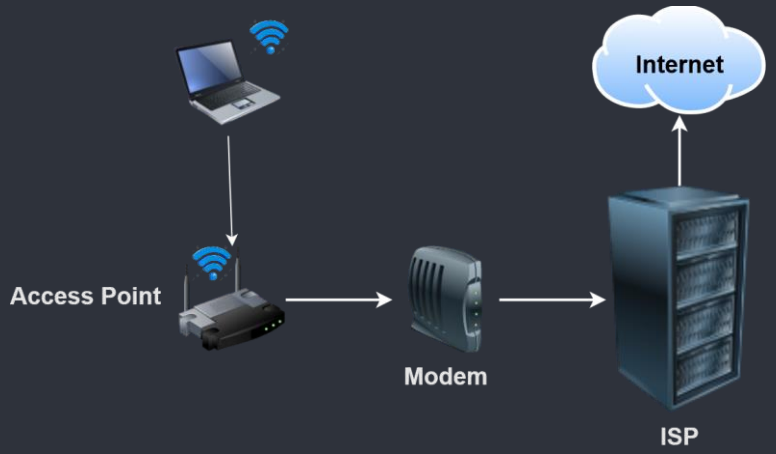
}



Red hogareña de Wi-Fi {

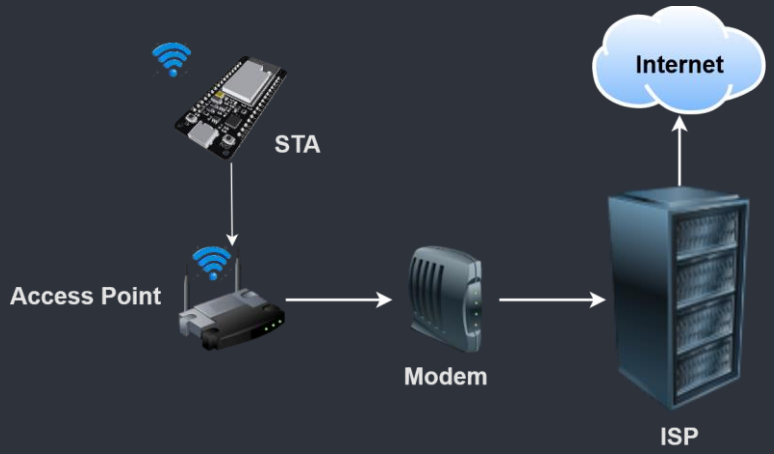
```
    # Para los ejemplos, utilizaremos
    un diagrama simplificado.

    # Vemos aquí un equipo portátil
    que se conecta a un Access Point
    (AP), el mismo se conecta al
    modem que es el que se vinculara
    con los servidores de nuestro ISP
    para entregarnos acceso a
    internet.
```

}



ESP32 en modo STA {

```
    # El primer modo de funcionamiento
    del modulo Wi-Fi del ESP32 es el
    modo station o estación (modo
    STA).

    # El modo Estación (STA) es uno de
    los modos de funcionamiento más
    utilizados del ESP32. En este
    modo, el ESP32 se conecta a una
    red Wi-Fi existente como un
    cliente, similar a cómo lo haría
    un teléfono inteligente o una
    computadora portátil. Este modo es
    fundamental para aplicaciones que
    requieren acceso a Internet o
    comunicación dentro de una red
    local.
}
```



ESP32 en modo STA {

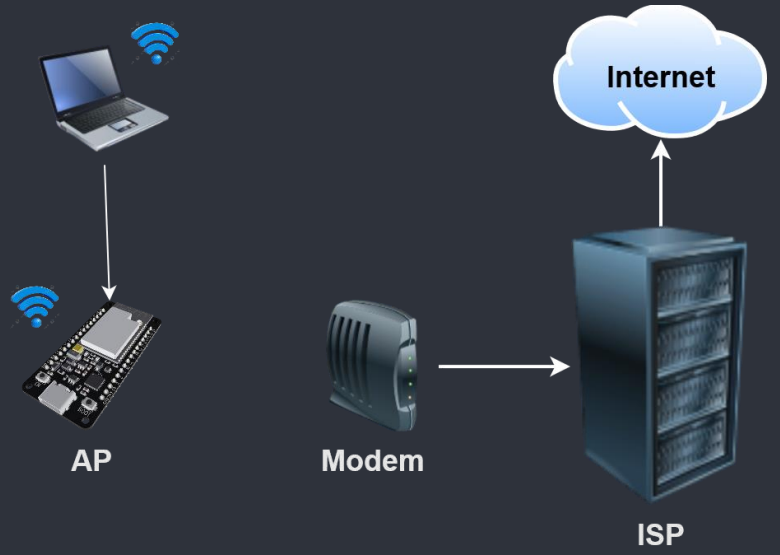
```
    # El modo Estación permite que el ESP32 se conecte a una red Wi-Fi
    existente, lo que le proporciona acceso a recursos y servicios disponibles
    en esa red, como ser acceso a internet, comunicación en la red local con
    otros dispositivos, actualización del firmware OTA (Over the Air), etc.

    # Casos de uso para este modo pueden ser monitoreo remoto de sensores,
    automatización de tareas, automatización del hogar, dispositivos portátiles
    con conexión a internet, control industrial, etc.

    # Permite conectarnos bajo diferentes protocolos de seguridad como WEP,
    WPA, WPA2, etc.
```

}

ESP32 en modo AP {

```
# El segundo modo de
funcionamiento del modulo Wi-Fi
del ESP32 es el modo access point
o punto de acceso (modo AP).

# En el modo Punto de Acceso, el
ESP32 crea su propia red Wi-Fi a
la que otros dispositivos pueden
conectarse. Este modo es útil para
crear redes locales temporales o
para configurar el ESP32 sin
necesidad de una red Wi-Fi
existente.
```

}



ESP32 en modo AP {

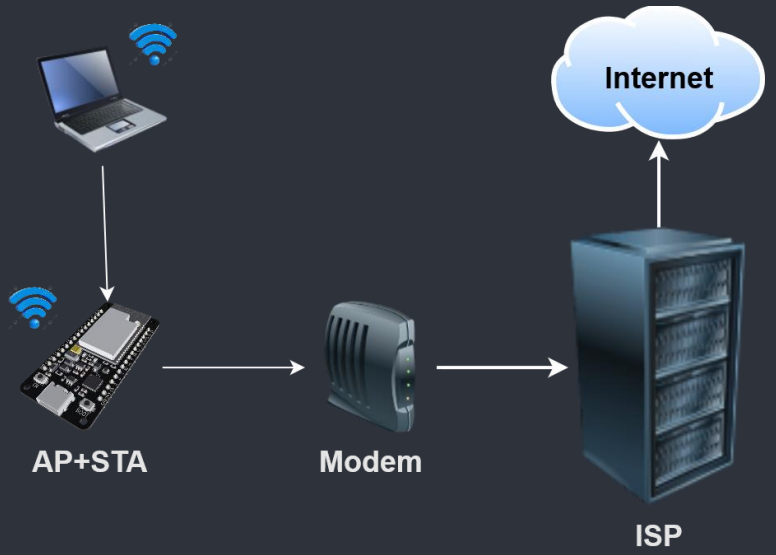
```
    # El modo Punto de Acceso (AP) es una de las funcionalidades más poderosas
    del ESP32. En este modo, el ESP32 actúa como si fuera un router Wi-Fi,
    creando su propia red inalámbrica a la que otros dispositivos pueden
    conectarse. Esto es especialmente útil en situaciones donde no hay una red
    Wi-Fi existente disponible o cuando se necesita una red local temporal para
    la configuración o comunicación entre dispositivos.

    # Este modo es especialmente útil para poder configurar por primera vez
    nuestro proyecto para que se adapte a las condiciones de trabajo (por
    ejemplo, instalación inicial del equipo) generando una red Wi-Fi temporal.

    # También sirve si necesitamos conectar dos dispositivos mediante una red
    de forma temporal de manera rápida o si necesitamos crear un red rápida
    para poder hacer pruebas y practicas.
```

}

ESP32 en modo dual STA+AP {

```
# El tercer modo de funcionamiento
del modulo Wi-Fi del ESP32 es el
modo dual (modo STA+AP).

# En el modo Dual, el ESP32 puede
funcionar simultáneamente como
estación y como punto de acceso.
Esto permite que el ESP32 esté
conectado a una red Wi-Fi
existente mientras crea su propia
red Wi-Fi.
```

}



ESP32 en modo dual STA+AP {

```
    # El modo Dual permite que el ESP32 actúe como un puente entre una red Wi-
    Fi existente y una red local creada por el propio ESP32. Bajo ciertas
    condiciones esto puede servir para extender la conexión de internet (no
    disponible aun en Micropython) o como puente entre redes.

    # Si bien el firmware todavía no tiene soporte para hacer NAT, podríamos
    hacer un manejo manual de paquetes entre los que llegan y los que salen, y
    si bien esto funcionaria, la performance seria muy mala, pero lo importante
    es que conceptualmente es posible hacerlo con mucho trabajo.

    # La función principal de este modo es poder brindar una red Wi-Fi para que
    todos los dispositivos se conectan a una red única mientras que el ESP32
    mantiene conexión a internet por si necesita recuperar información o datos
    (como sincronización de fechas y horas por NTP).
```

}

Comparativa {

Característica	Modo Estación (STA)	Modo Punto de Acceso (AP)	Modo Dual (STA + AP)
Descripción	Conecta el ESP32 a una red Wi-Fi existente	Crea una red Wi-Fi propia	Conecta a una red Wi-Fi y crea una red propia
Conectividad	Cliente de una red Wi-Fi	Punto de acceso para otros dispositivos	Cliente de una red Wi-Fi y punto de acceso
Acceso a Internet	Sí, a través de la red Wi-Fi conectada	No, a menos que esté conectado a otra red	Sí, a través de la red Wi-Fi conectada
Configuración Inicial	Necesita red Wi-Fi existente	No necesita red Wi-Fi existente	Necesita red Wi-Fi existente
Número de Dispositivos	Uno (el ESP32)	Múltiples dispositivos pueden conectarse	Múltiples dispositivos pueden conectarse
Rango de Cobertura	Depende del router Wi-Fi	Depende de la ubicación del ESP32	Depende de la ubicación del ESP32
Configuración de IP	Asignada por el router	Asignada por el ESP32	Asignada por el ESP32
Aplicaciones comunes	Acceso a Internet, comunicación en red local	Configuración de dispositivos, redes locales	Repetidor Wi-Fi, puente de red, configuración
Ventajas	Acceso a recursos de red e Internet	Fácil configuración y control local	Conectividad versátil y extendida
Desventajas	Necesita red Wi-Fi existente	Sin acceso a Internet	Complejidad en la configuración

}

Ejemplo 1 – Escaner de redes {

```python
import network

# Inicializar la interfaz de estación
estacion = network.WLAN(network.STA_IF)

# Activar la interfaz de estación
estacion.active(True)

# Escanear las redes disponibles
redes = estacion.scan()

print(redes)

# Desactivar la interfaz de estación
estacion.active(False)
```

}

SSID {

```
----------------------------------------
SSID: MiRedWiFi
BSSID: ac:cf:23:61:b7:95
Channel: 6
RSSI: -45
Authentication: WPA2-PSK
----------------------------------------
```

El **SSID** (Service Set Identifier) es el nombre de una red Wi-Fi. Es un identificador único que permite a los dispositivos distinguir entre diferentes redes Wi-Fi disponibles. El SSID es una cadena de texto que puede contener hasta 32 caracteres y es configurado por el administrador de la red. Cuando un dispositivo escanea las redes Wi-Fi disponibles, el SSID es uno de los principales identificadores que se muestran para que el usuario pueda seleccionar la red a la que desea conectarse.

}

BSSID {

```
----------------------------------------
SSID: MiRedWiFi
BSSID: ac:cf:23:61:b7:95
Channel: 6
RSSI: -45
Authentication: WPA2-PSK
----------------------------------------
```

El **BSSID** (Basic Service Set Identifier) es un identificador único que se asigna a cada punto de acceso (AP) en una red Wi-Fi. Es esencialmente la dirección **MAC** (Media Access Control) del punto de acceso y se utiliza para identificar de manera única cada AP en una red Wi-Fi.

El BSSID es una dirección MAC de 48 bits (6 bytes) que identifica de manera única un punto de acceso en una red Wi-Fi. Cada punto de acceso tiene un BSSID único, lo que permite a los dispositivos diferenciar entre múltiples puntos de acceso, incluso si tienen el mismo SSID. El BSSID tiene un formato de esta forma 00:1A:2B:3C:4D:5E

}

Canal {

```
----------------------------------------
SSID: MiRedWiFi
BSSID: ac:cf:23:61:b7:95
Channel: 6
RSSI: -45
Authentication: WPA2-PSK
----------------------------------------
```

El "**Channel**" (canal) en el contexto de redes Wi-Fi se refiere a la frecuencia específica en la que se transmite la señal Wi-Fi. Los canales son divisiones del espectro de frecuencia que permiten que múltiples redes Wi-Fi operen en la misma área sin interferir entre sí, siempre y cuando estén en canales diferentes o suficientemente separados.

Este dato nos sirve si vamos a armar una red Wi-Fi para poder ver cuantas otras redes hay en un canal determinado y de esta manera poder elegir algún canal libre para evitar interferencias o solapamientos, o en su defecto, elegir los canales en donde las otras redes tengan menor potencia de señal.

}

RSSI {

```
----------------------------------------
SSID: MiRedWiFi
BSSID: ac:cf:23:61:b7:95
Channel: 6
RSSI: -45
Authentication: WPA2-PSK
----------------------------------------
```

El **RSSI** (Received Signal Strength Indicator) es una medida de la potencia de la señal recibida por un dispositivo de red inalámbrica, como el ESP32, desde un punto de acceso Wi-Fi. El RSSI se expresa en decibelios-milivatios (dBm) y es un indicador clave de la calidad de la conexión inalámbrica.

Por ejemplo, un RSSI de -30 dBm es una señal muy fuerte, mientras que un RSSI de -90 dBm es una señal muy débil. En la practica se considera una señal de **-60dBm a -70dBm como débil**, de **-70dBm a – 80dBm como muy débil** y **menor a -80dBm** directamente se toma como que **no existe conexión**.

}

Authentication {

```
---------------------------------------
SSID: MiRedWiFi
BSSID: ac:cf:23:61:b7:95
Channel: 6
RSSI: -45
Authentication: WPA2-PSK
---------------------------------------
```

```
#  La authentication se refiere al proceso de verificar la identidad de un
dispositivo que intenta conectarse a la red. Los modos de autenticación
determinan cómo se realiza esta verificación y qué nivel de seguridad se
aplica a la conexión. Para WPA y WPA2 los ESP soportan TKIP y AES.

# 0 = Open
# 1 = WEP
# 2 = WPA-PSK
# 3 = WPA2-PSK
# 4 = WPA/WPA2-PSK
```

}

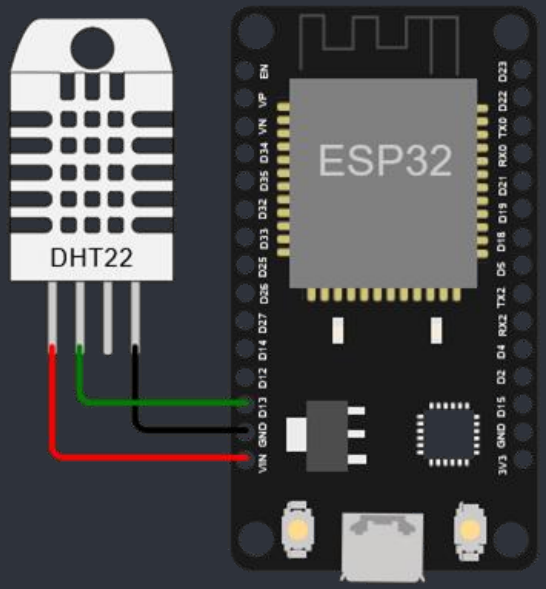
Ejemplo 3 - Conectarse {

```
import network, time

ssid = 'Nombre de Wi-Fi'
password = 'xxxxx'
estacion = network.WLAN(network.STA_IF)
estacion.active(True)
estacion.connect(ssid, password)

print('Conectando ...', end="")

while not estacion.isconnected():
    print('.', end="")
    time.sleep(0.1)

print('Conectado')
```

}

Ejemplo 10 – Respuesta DHT {

```
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(('0.0.0.0', 80))
    s.listen(5)

    while True:
        cliente, ip_cliente = s.accept()
        print('Cliente conectado desde', ip_cliente)
        sensor.measure()
        temperatura = sensor.temperature()
        humedad = sensor.humidity()
        response = f"""\
    <html>
    <body>
        <h1>## Aquí completar con lo que quiero mostrar ##</h1>
    </body>
    </html>
    """
        cliente.send(response)
        cliente.close()
}
```



```
1
2
3  Aprender a programar
4  es aprender a pensar.
5
6
7
8
9
10
11 { Steve Jobs; }
12
13
14
```


{ Nos vemos en la proxima clase }