UNIVERSIDAD DE ORIENTE

NÚCLEO DE ANZOÁTEGUI

ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

PROYECTOS DIGITALES AVANZADOS

PRÁCTICA N°11: SERVIDOR WEB Y CLIENTE

Profesor:
Pedro Rene Cabrera

Bachiller:
Claudia Rodríguez
C.I: 27.943.668

Barcelona, enero de 2025

1. OBJETIVOS DE LA PRÁCTICA

1.1. Establecer una red local entre dos Raspberry Pi Pico W, configurando uno como servidor y el otro como cliente, para permitir la comunicación y transferencia de datos entre ellos

1.2. Crear una página web en el servidor utilizando HTML para visualizar los datos recolectados y enviados por el cliente.

1.3. Configurar el Pico **'cliente'** para tomar muestras de temperatura usando su sensor interno, enviar los datos al servidor mediante solicitudes HTTP POST y explorar el protocolo HTTP para comprender su uso en la transferencia de datos

1.4. Configurar el Pico **'servidor'** para recibir y almacenar los datos enviados por el cliente, publicarlos en una página web accesible desde su dirección IP, y obtener datos adicionales de una API externa mediante una solicitud GET.

Observaciones: Este fue un añadido de la programadora, para hacer este dispositivo IoT un poco más dinámico, es decir, que reciba información del cliente pero que tambien sea capaz de obtener información de una página de internet en formato JSON.

1.5. Permitir que cualquier dispositivo conectado a la misma red local (computadora o teléfono) pueda acceder a la información publicada en la dirección IP del servidor, incluyendo los datos obtenidos de la API y los enviados por el cliente.

1.6. Utilizar el módulo OLED SSD1306 como periférico de salida en ambos dispositivos para visualizar la ejecución y monitoreo en tiempo real del programa.

2. DESARROLLO

2.1. PLANTEAMIENTO DEL PROBLEMA

En espacios o lugares como casas grandes, universidades o edificios grandes, el monitoreo de variables ambientales, como la temperatura, es crucial, pero a menudo complicado debido a la falta de herramientas económicas y autónomas que no dependan de internet. Esto plantea la necesidad de un sistema que funcione en una red local, permita recopilar y visualizar datos en tiempo real, y sea accesible para los usuarios conectados.

Para resolver este problema, la siguiente práctica busca implementar un sistema IoT utilizando Raspberry Pi Pico W, configurados en un modelo cliente-servidor. El dispositivo cliente tomará datos de temperatura y los enviará al servidor mediante solicitudes POST. El servidor, por su parte, almacenará los datos y los publicará en una página web accesible dentro de la red. Además, incluirá información adicional obtenida de una API externa, ofreciendo un sistema eficiente, dinámico y accesible para el monitoreo ambiental en tiempo real.

2.2. SOLUCIÓN

La solución se dividirá en tres partes, la del cliente, la del servidor y funciones utilizadas en ambos códigos:

2.2.1. Solución cliente:

- **Crear una función para el monitoreo de temperatura:** Se implementará la función **leer_sensor(),** que utiliza el sensor interno de temperatura del Raspberry Pi Pico W. Esta función no recibe parámetros y retorna la temperatura en grados Celsius (°C). Esta función es una mejora respecto a las prácticas anteriores, donde no se había definido de forma estructurada.
- **Definir una función para enviar los datos al servidor:** La función **enviar_datos_servidor**(ip_servidor, puerto, temperatura) recibe tres parámetros: ip_servidor (que es la dirección IP del servidor obtenida desde la ejecución del servidor), puerto (que es utilizado para la comunicación, que es el puerto 80 por el cual el servidor está escuchando) y por último temperatura (la variable que contiene la temperatura medida en grados Celsius). Esta función envía los datos al servidor mediante una solicitud POST con la siguiente línea de código: **s.send(f"POST / HTTP/1.1\r\nHost: {ip_servidor}\r\nContent-Type: text/plain\r\nContent-Length: {len(mensaje)}\r\n\r\n{mensaje}").** Después de enviar la solicitud, se cierra la conexión y se muestra en la consola que los datos fueron enviados correctamente.
- **Crear una función denominada cliente():** Esta es la función principal del código del cliente. Las acciones que realiza son las siguientes, nota: es una síntesis de los pasos que hace:
  - Se conecta a la red Wi-Fi clau-moto.
  - Se define la ip_servidor, que es la dirección IP del servidor (esta se obtiene a partir del programa del servidor)
  - Utiliza el GPIO14 como alerta visual durante el monitoreo de la temperatura (encendiendo un LED azul)
  - Realiza la toma de una muestra de temperatura usando la función **leer_sensor(),** y la envía al servidor utilizando **enviar_datos_servidor().**
  - Muestra un mensaje en la pantalla OLED que indica que la muestra se tomó correctamente.
  - Luego, espera 5 segundos antes de repetir el proceso.

2.2.2. Solución servidor:

- **Definir una función para obtener la fecha y hora actual:** Esta función es utilizada para obtener el momento exacto en el que se tomó la muestra de temperatura. Ya es conocida por la programadora debido a su uso en prácticas anteriores.
- **Se crea una función denominada servidor:** Esta función es el complemento de la función cliente(). Realiza las siguientes tareas:
  - Inicializa el GPIO14 como alerta visual cuando se recibe información del cliente.
  - Se conecta a la red Wi-Fi clau-moto.
  - Configura el servidor para escuchar en el puerto 80, que es el puerto utilizado por el cliente para enviar la solicitud POST.

- En el bucle principal (while):
  - Escucha y recibe solicitudes POST del cliente.
  - Al recibir los datos, muestra la temperatura en la pantalla OLED.
  - Realiza una solicitud GET a la API externa para obtener datos climáticos en formato JSON.
  - Publica la información tanto de la temperatura medida por el cliente como los datos obtenidos de la API en una página web accesible desde cualquier dispositivo conectado a la misma red local.
- Utiliza las funciones **obtener_datos_clima()** y **pagina_web()** para realizar las tareas anteriormente descritas.

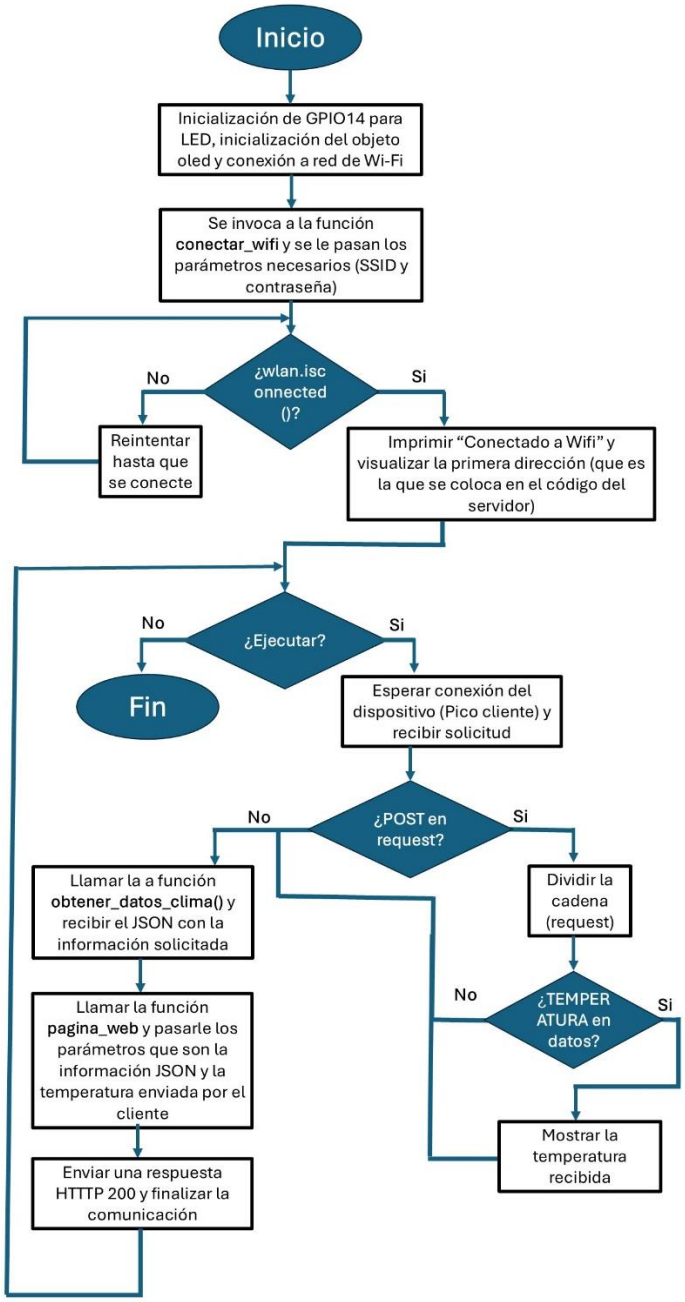
- **Definir la función obtener_datos_clima():** Esta función hace una solicitud GET a la API externa (https://api.weatherapi.com) para obtener datos climáticos (como la temperatura y humedad). Luego, los datos obtenidos se formatean para que puedan ser manipulados y utilizados en la página web.
- **Definir la función pagina_web():** Esta función recibe dos parámetros: ultima_temperatura, que es la última temperatura medida por el cliente y api_data que son los datos obtenidos de la API del clima. Con estos parámetros, genera una página web simple en HTML, mostrando primero los datos obtenidos de la API (como el clima actual) y luego la temperatura medida por el cliente. Esta página es accesible desde cualquier navegador dentro de la red local.

2.2.3. Funciones utilizadas en ambos códigos:

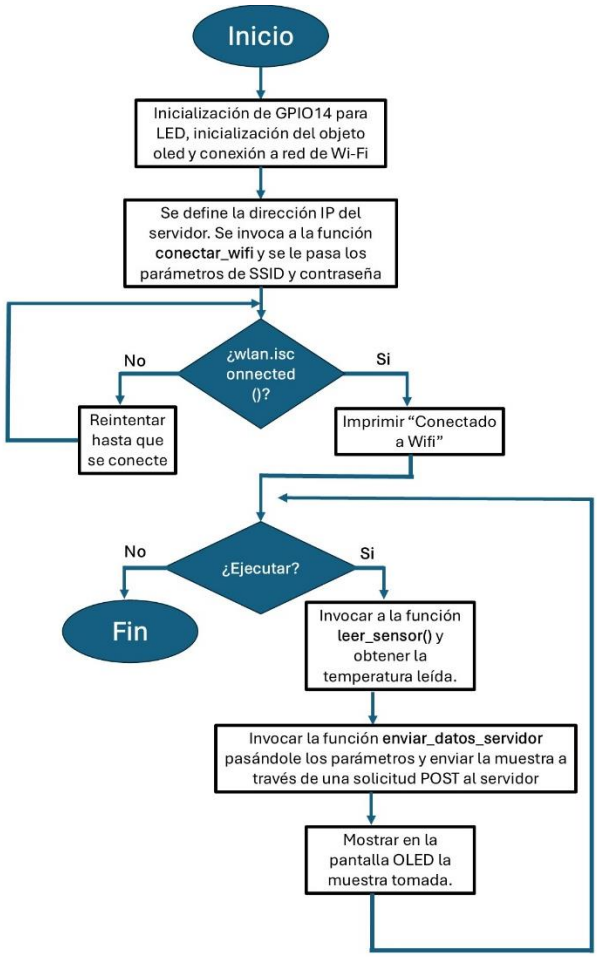
- **Función conectar_wifi(nombre_red, contraseña):** Esta función recibe dos parámetros: el nombre de la red Wi-Fi y la contraseña. La función permanece en un bucle while hasta que el dispositivo se conecta correctamente a la red. Una vez que la conexión es exitosa, muestra el mensaje "Conexión a WiFi exitosa".
- **Funciones para el módulo OLED:** Se reutilizan dos funciones de prácticas anteriores: crear_oled() que inicializa el módulo OLED SSD1306 y mostrar_oled(oled, mensaje, tiempo) que muestra un mensaje en el OLED durante el tiempo especificado.

2.3. Diagrama de Flujo

2.3.1. Diagrama de Flujo (servidor)

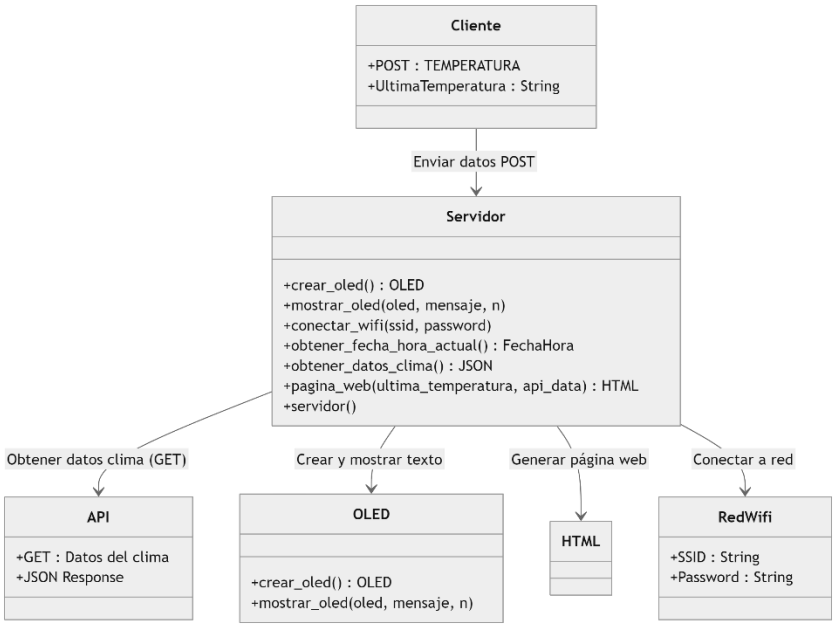


2.3.2. Diagrama de Flujo (cliente)

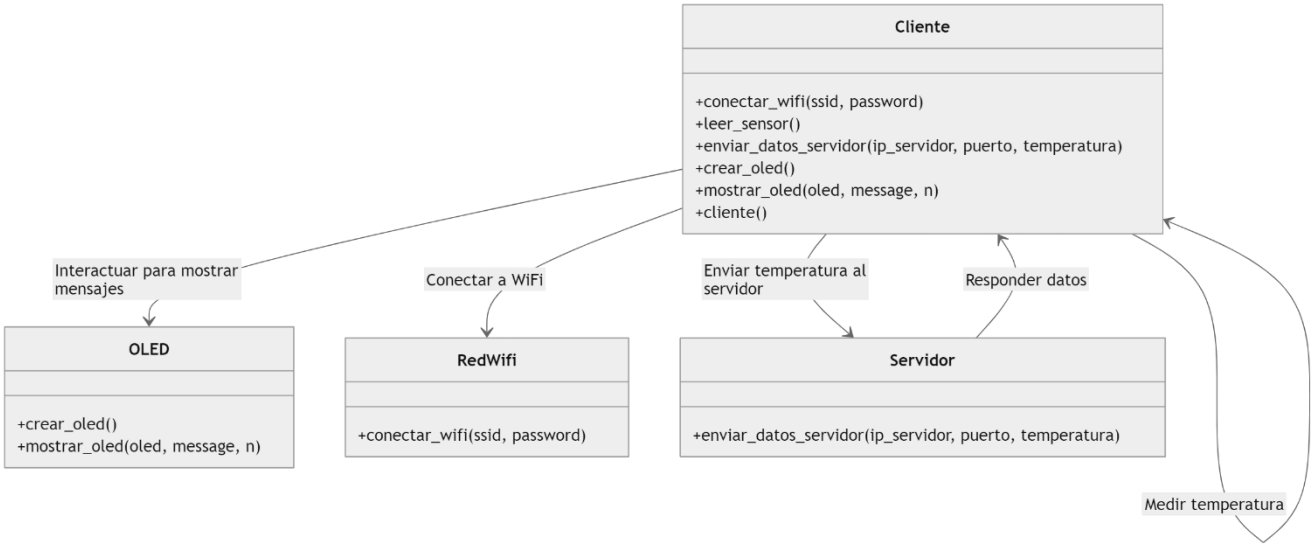


2.4. Diagrama UML.

2.4.1. Diagrama UML (servidor)



2.4.2. Diagrama UML (cliente)



3. ANEXOS

3.1. Código

3.1.1. Raspberry Pi Pico W (cliente)

```
# COM6 --> CLIENTE
# si le funciona el sensor
import usocket as socket
import network
import time
import re
from ssd1306 import SSD1306_I2C
from machine import Pin, I2C
import utime
from time import sleep

# -- crear objeto para manejar la pantalla oled -- #
def crear_oled():
  i2c = I2C(0, scl=Pin(17), sda=Pin(16), freq=400000) ## <-- pendiente
  oled = SSD1306_I2C(128,64,i2c)
  return oled

# -- funcion para mostrar mensajes en la pantalla OLED -- #
def mostrar_oled(oled, message, n):
  oled.fill(0)
  ancho_caracter = 7 # <-- tamaño en pixeles de un caracter
  max_columna = 120  # <-- long max de la pantalla
  fila = 0       # <-- primera linea
  columna = 0          # <-- primera columna
  palabras = message.split() # <-- string a lista
  for palabra in palabras:
    ancho_palabra = len(palabra) * ancho_caracter
    if columna + ancho_palabra > max_columna:
      fila += 16
      columna = 0
    if fila >= 50:
      oled.show()
      sleep(n)
      oled.fill(0)
      fila = 0
      columna = 0
    oled.text(palabra, columna, fila)
    columna = columna + 7 # --> espacio entre palabras porsia
    columna += ancho_palabra + ancho_caracter
  oled.show()
  sleep(n)

# -- funcion para conectar a wifi ---
def conectar_wifi(ssid, password):
  wlan = network.WLAN(network.STA_IF)
  wlan.active(True)
  wlan.connect(ssid, password)
  while not wlan.isconnected():
    time.sleep(1)
  print("Conectado a WiFi")
  return wlan.ifconfig()[0]

# -- leer sensor de temperatura del PICO W -- #
def leer_sensor():
  sensor_temp = machine.ADC(4)
  conversion_factor = 3.3 / (65535)
  reading = sensor_temp.read_u16() * conversion_factor
  temperature = 27 - (reading - 0.706)/0.001721
  print(temperature)
  temperature = round(temperature,2)
  return temperature

# -- funcion para enviar datos al servidor -- #
def enviar_datos_servidor(ip_servidor, puerto=80, temperatura=0):
  try:
    s = socket.socket()
    s.connect((ip_servidor, puerto))
    mensaje = f"TEMPERATURA={temperatura}"
    s.send(f"POST / HTTP/1.1\r\nHost: {ip_servidor}\r\nContent-Type: text/plain\r\nContent-Length: {len(mensaje)}\r\n\r\n{mensaje}")
```

```
    s.close()
    print("Datos enviados al servidor")
  except OSError as e:
    print(f"Error al conectar con el servidor: {e}")

# -- main -- #
def cliente():
  '''MODIFICAR DE ACUERDO SI ESTAS
  EN CASA O EN LA CLASE'''
  azul = Pin(14, Pin.OUT)
  azul.value(0)
  oled = crear_oled()
  oled.fill(0)
  oled.show()
  sleep(2)
  mostrar_oled(oled, "CLIENTE: MONITOREO DE TEMPERATURA", 3)
  ssid = 'clau-moto'
  password = 'tata4646'
  mostrar_oled(oled, f"SSID A CONECTAR: {ssid}", 4)
  ip_servidor = '192.168.181.74'
  conectar_wifi(ssid, password)
  azul = Pin(14, Pin.OUT)
  while True:
    temperatura = leer_sensor()
    azul.value(1)
    cadena = f"Temperatura medida: {temperatura} grados centigrados"
    print(cadena)
    mostrar_oled(oled, cadena, 1)
    azul.value(0)
    enviar_datos_servidor(ip_servidor, temperatura=temperatura)
    mostrar_oled(oled, "Esperando nueva muestra a tomar (cliente)", 1)
    time.sleep(5)  # espera 5 seg mas antes de tomar nueva muestra

if __name__ == "__main__":
  cliente()
```

3.1.2. Raspberry Pi Pico W (servidor)

```
# COM7 --> SERVER_SENSOR
# es porque el cliente si le funciona el sensor

import socket
import network
from machine import Pin, I2C
from time import sleep
import time
from ssd1306 import SSD1306_I2C
import utime
import requests

# -- funcion para crear el objeto OLED -- #
def crear_oled():
  i2c = I2C(0, scl=Pin(17), sda=Pin(16), freq=400000) ## <-- pendiente
  oled = SSD1306_I2C(128,64,i2c)
  return oled

# -- funcion para mostrar mensajes en la pantalla OLED -- #
def mostrar_oled(oled, message, n):
  oled.fill(0)
  ancho_caracter = 7 # <-- tamaño en pixeles de un caracter
  max_columna = 120  # <-- long max de la pantalla
  fila = 0       # <-- primera linea
  columna = 0            # <-- primera columna
  palabras = message.split() # <-- string a lista
  for palabra in palabras:
    ancho_palabra = len(palabra) * ancho_caracter
    if columna + ancho_palabra > max_columna:
      fila += 16
      columna = 0
    if fila >= 50:
      oled.show()
      sleep(n)
      oled.fill(0)
```

```
        fila = 0
        columna = 0
      oled.text(palabra, columna, fila)
      columna = columna + 7 # --> espacio entre palabras porsia
      columna += ancho_palabra + ancho_caracter
    oled.show()
    sleep(n)

# -- conectarse al wifi -- #
def conectar_wifi(ssid, password):
    station = network.WLAN(network.STA_IF)
    station.active(True)
    station.connect(ssid, password)
    while not station.isconnected():
        time.sleep(1)
    print("Conexion a WiFi exitosa")
    print(station.ifconfig())
    print("CONECTARSE EN EL SERVER A LA PRIMERA\nDIRECCION QUE IMPRIME")

# -- obtener fecha/hora actual -- #
def obtener_fecha_hora_actual():
    current_time = time.time()
    local_time = time.localtime(current_time)
    year = local_time[0]
    month = local_time[1]
    day = local_time[2]
    hour = local_time[3]
    minute = local_time[4]
    second = local_time[5]
    fecha = "{:02}/{:02}/{}".format(day, month, year)
    hora = "{:02}:{:02}:{:02}".format(hour, minute, second)
    return fecha, hora

# -- servidor (que es el main) -- #
def servidor():
    blanco = Pin(14, Pin.OUT)
    blanco.value(0)
    oled = crear_oled()
    oled.fill(0)
    oled.show()
    sleep(2)
    mostrar_oled(oled, "SERVIDOR: PROVEEDOR DE PAGINA WEB", 3)
    ultima_temperatura = "vacio"
    ssid = 'clau-moto'
    password = 'tata4646'
    mostrar_oled(oled, f"SSID A CONECTAR: {ssid}", 4)
    conectar_wifi(ssid, password)
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(('', 80))
    s.listen(5)
    cadena = "Esperando conexiones de posibles clientes.."
    mostrar_oled(oled, cadena, 4)
    print(cadena)

    while True:
        mostrar_oled(oled, "Esperando informacion del cliente.. (servidor)", 3)
        conn, addr = s.accept()
        print(f"Conexion desde: {addr}")
        print(f"Cliente detectado")
        request = conn.recv(1024).decode('utf-8')

        # solicitud POST
        if "POST" in request:
            contenido_inicio = request.find("\r\n\r\n") + 4
            datos = request[contenido_inicio:]
            if "TEMPERATURA=" in datos:
                ultima_temperatura = datos.split("=")[1].strip()
                blanco.value(1)
                cadena = f"Temperatura recibida: {ultima_temperatura} grados centigrados"
                mostrar_oled(oled, cadena, 2)
                blanco.value(0)
                mostrar_oled(oled, "Esperando nueva muestra (servidor)", 1)
                print(cadena)

        # obtener datos de la pagina
        api_data = obtener_datos_clima()
        # enviar los dos parametros con la info de la temp y de la api
```

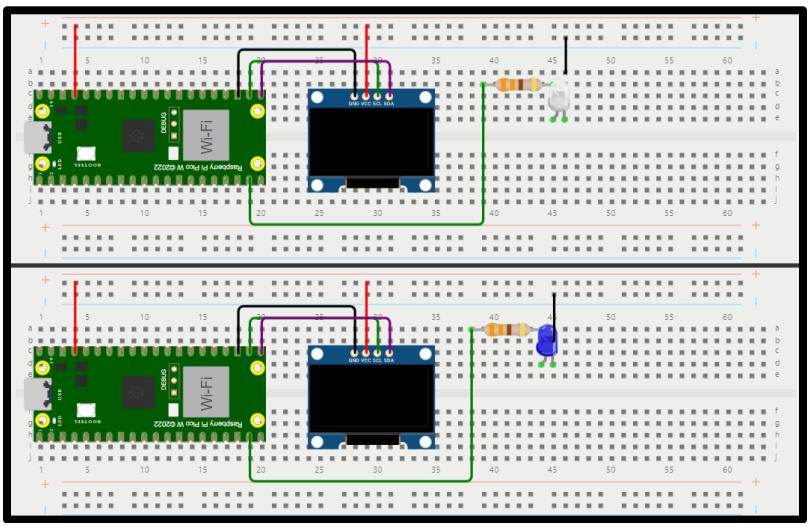
```
    respuesta = pagina_web(ultima_temperatura, api_data)
    conn.send("HTTP/1.1 200 OK\nContent-Type: text/html\nConnection: close\n\n" + respuesta)
    # finaliza la comunicacion
    conn.close()

# -- funcion para obtener datos del clima de la api -- #
def obtener_datos_clima():
  api_key = '2d5f2c6bf4d14d6c84d123921252201' # esta api-key es de mi perfil creado
  location = 'VENEZUELA' # puse venezuela porque no tomaba anzoategui
  url = f'https://api.weatherapi.com/v1/current.json?q={location}&key={api_key}'
  try:
    response = requests.get(url)
    if response.status_code == 200:
      weather = response.json() # obtiene el json de lagpina
      return {
        'weather_description': weather['current']['condition']['text'],
        'temperature_c': weather['current']['temp_c'],
        'humidity': weather['current']['humidity'],
        'precipitation': weather['current']['precip_mm'],
        'wind_speed': weather['current']['wind_kph']
      }
    else:
      return {'error': 'Unable to fetch data from API'}
  except Exception as e:
    return {'error': str(e)}

# -- funcion para crear pagina web -- #
def pagina_web(ultima_temperatura, api_data):
  api_info = """<p>Error obteniendo datos de la API.</p>"""
  if 'error' not in api_data:
    api_info = f"""
    <p>Clima actual en API:</p>
    <ul>
      <li>Descripcion: {api_data['weather_description']}</li>
      <li>Temperatura: {api_data['temperature_c']} °C</li>
      <li>Humedad: {api_data['humidity']}%</li>
      <li>Precipitacion: {api_data['precipitation']} mm</li>
      <li>Velocidad del viento: {api_data['wind_speed']} kph</li>
    </ul>
    """
  html = f"""
  <!DOCTYPE html>
  <html>
  <head>
    <title>Servidor y Clima</title>
  </head>
  <body>
    <h1>Datos recibidos</h1>
    <p>Temperatura ultima recibida del cliente: {ultima_temperatura} °C</p>
    {api_info}
  </body>
  </html>
  """
  return html

# -- main -- #
if __name__ == "__main__":
  servidor()
```

3.2. Diagrama circuital



3.3. Bibliografía

Estación metereológica con Raspberry Pi y sensor BME280. Mediremos temperatura, presión y humedad.

Enlace: https://www.youtube.com/watch?v=jSA5eM9fomE

Raspberry Pi Pico: Web Server (MicroPython). Enlace: https://randomnerdtutorials.com/raspberry-pi-pico-web-server-micropython/

Raspberry Pi Pico W: Getting Started with HTTP GET Requests (MicroPython). Enlace: https://randomnerdtutorials.com/raspberry-pi-pico-w-http-requests-micropython/