Exercise 2: Wi-Fi and Wireless Data Transmission

Learning Outcome: To get familiar with wireless data transmission.

Todo:

- Setup HTTP Web Server on Raspberry Pi Pico
- Interact with on-board LED via web interface
- Display real-time BMP values on web interface

Implementation:

1. Setup HTTP Web Server on Pico W

NOTE:

- If you face issue with connecting the Pico to Wifi (at home or university), it may due to router settings and configuration
- Hence, you better to create a Hotspot from your mobile and then connect *the Pico and computer with the* ***same*** *wifi*.

- **Connect Pico W to Wi-Fi**
  - Import libraries, here, you need *network*, *socket*
  - Define SSID (wifi name), password
  - Define WLAN with network.WLAN(network.STA_IF)
  - Make it active, then try to connect with SSID and password.
  - Check connection status with .status() function (can skip, yet still recommend to do so). If connect successfully, print out the assigned IP for the Pico W.
    (At this point, you can test the connection between your computer and Pico W via ping or telnet command.)
- **Setup socket and listen**
  - Define address to make socket listen later on port 80 for all network interfaces (0.0.0.0).
  - Open socket, bind to address and start listen

- **Setup HTTP Web Server**
  - Make a While loop to check connection from client

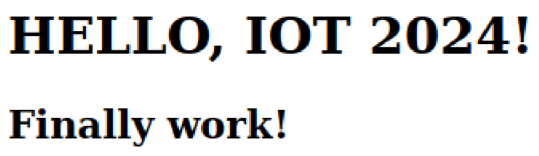
  - Receive request

  - Send response to client. Here, the response need to be in string of HTML. For example:
```
html = """\
HTTP/1.1 200 OK
Content-Type: text/html
<!DOCTYPE html>
<html>
  <head><title>Raspberry Pi Pico Web Server</title></head>
  <body>
      <h1>HELLO, IOT 2024!</h1>
      <h2>Finally work!</h2>
  </body>
</html>
"""
```

  - Close connection

- **Run and test**

- **Sample output on webpage**


HELLO, IOT 2024!

Finally work!

2. Interact with on-board LED via web interface

In this task, we keep almost the same code, but just do some small modifications

- **Define LED with Pin**

- **Create web interface with Toggle button**
  - Your HTML should display:
    - LED status
    - Toggle button

- Example code of HTML:

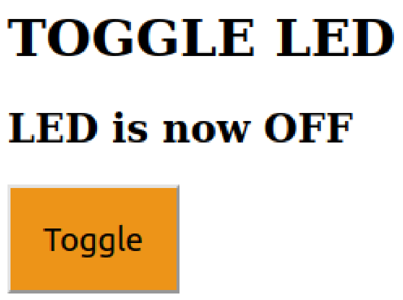
```
def generate_html(status):
    html = f"""\
    HTTP/1.1 200 OK
    Content-Type: text/html

    <!DOCTYPE html>
    <html>
      <head><title>Raspberry Pi Pico Web Server</title></head>
      <body>
          <h1>TOGGLE LED</h1>
          <h2>LED is now {status}</h2>
          <p><a href='/toggle'><button style="background-color: #ed9418; padding: 20px; font-size:20px">Toggle</button></a></p>
      </body>
    </html>
    """
    return str(html)
```

- **Write code to toggle the LED**
    - When LED status = ON
        - Pressing "Toggle" button gonna turn the LED off
        - On the web, LED status should shown as OFF

    - And vice versa

- **Run and test**

- **Sample output on webpage**

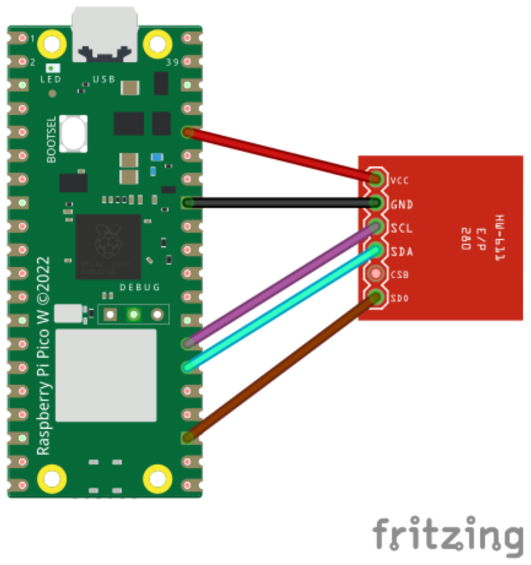

TOGGLE LED

LED is now OFF

3. Display real-time BMP values on web interface

In this task, we also keep almost the same code, but just do some small modifications

- **Connect BMP280 to Pico W**
    - Look back at Exercise 1. Check out the (a) Fritzing draw, (b) Pinout, and (c) exercise PDF.
      <u>**Note:**</u>
        - SCL and SDA pins can be connect to any SCL and SDA pins on Pico, but need to be in **the same channel**
        - GND can be to any GND pin
        - VCC to 3V3(OUT) not the (_EN)
        - SDO must be connected to (any) GND pin.



- **Read sensor data**
    - Define BMP280 instance

    - Read pressure and temperature values

- **Create web interface**
    - Your HTML should display pressure (in Pa) and temperature (in °C) values.

    - Example code of HTML

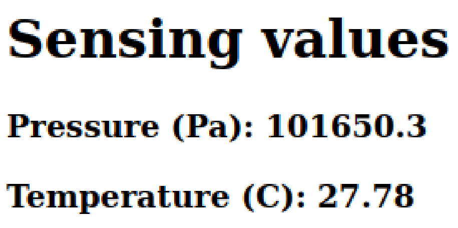
Auto-reload page	Auto fetching data with JS (No-reload page)

def generate_html(press, temp):
    html = f"""\
    HTTP/1.1 200 OK
    Content-Type: text/html
    <!DOCTYPE html>
    <html>
	def generate_html():
	    html = """\
	    HTTP/1.1 200 OK
	    Content-Type: text/html
	    <!DOCTYPE html>
	    <html>

```
  <head>
    <title>Raspberry Pi Pico Web Server</title>
    <meta http-equiv="refresh" content="2">
  </head>
  <body>
    <h1>Sensing values</h1>
    <h3>Presure (Pa): {press}</h3>
    <h3>Temperature (C): {temp}</h3>
  </body>
</html>
"""
return str(html)
```

```
  <head>
    <title>Raspberry Pi Pico Web Server</title>
    <script>
      function fetchData() {
          fetch('/data')
          .then(response => response.json())
          .then(data => {
              document.getElementById("press").textContent = data.pressure;
              document.getElementById("temp").textContent = data.temperature;
          })
          .catch(error => console.error('ERROR fetching data:', error));
      }
      setInterval(fetchData, 1000); // Fetch every 1s
    </script>
  </head>
  <body>
      <h1>Sensing values</h1>
      <h3>Pressure (Pa): <span id="press">Loading...</span></h3>
      <h3>Temperature (C): <span id="temp">Loading...</span></h3>
  </body>
</html>
"""
return str(html)
```

- **Run and test**
- **Sample output on webpage**



Sensing values

Pressure (Pa): 101650.3

Temperature (C): 27.78