In the name of God

Parsa Aghaali 400521072

Question Report: ESP32 Deep Sleep with Wi-Fi and HTTP Server Communication

Introduction:

The goal of this project was to program an ESP32 to:

1. Wake up from deep sleep.
2. Connect to a Wi-Fi network.
3. Send a **heartbeat message** to an HTTP server.
4. Return to deep sleep for a specified duration (10 minutes) to save power.

The solution was tested successfully using a **custom Python Flask server** to handle the /heartbeat endpoint.

Steps Taken:

1. **ESP32 Configuration:**
    - Programmed the ESP32 to use its **deep sleep mode** for 10 minutes after sending a heartbeat signal.
    - Used the WiFi.h and HTTPClient.h libraries to connect to Wi-Fi and communicate with the server.
2. **Server Setup Using Flask:**
    - A custom Python server using the **Flask** framework was implemented to handle the /heartbeat HTTP GET request.
    - The Flask server provided a clean and efficient way to test the ESP32's heartbeat functionality without relying on static files.
3. **Flask Server Code:** The following Flask server code was used:

```
4. from flask import Flask
5.
6. app = Flask(__name__)
7.
8. @app.route('/heartbeat', methods=['GET'])
9. def heartbeat():
10.     return "Heartbeat received!", 200
```

```
11.
12.if __name__ == '__main__':
13.    app.run(host='0.0.0.0', port=80)
14.
```

- The server listens on port 80 and responds with **"Heartbeat received!"** when the ESP32 sends a request to /heartbeat.

4- **Wi-Fi and HTTP Communication:**

- The ESP32 connects to the Wi-Fi network using the provided SSID and password.
- It sends an HTTP GET request to the Flask server's /heartbeat endpoint.
- The server responds with a **200 OK** status and a success message.

5 **Deep Sleep Implementation:**

- After successfully sending the heartbeat message, the ESP32 enters deep sleep for 10 minutes using:

```
•     ESP.deepSleep(600000000);
```

Results

1. **Server Communication Success:**
    - The ESP32 successfully sent the HTTP GET request to the Flask server's /heartbeat endpoint.
    - The server responded with **200 OK** and the message "Heartbeat received!".

Example ESP32 Serial Output:

ESP32 Waking Up...

Connecting to Wi-Fi...

Wi-Fi connected!

ESP32 IP Address: 192.168.1.4

Heartbeat sent. Server response: Heartbeat received!

Going to deep sleep for 10 minutes...

2. **Deep Sleep Functionality:**
    - After sending the heartbeat message, the ESP32 entered deep sleep for 10 minutes.
    - Upon waking up, it repeated the process (Wi-Fi connection → Heartbeat → Sleep).
3. **Flask Server Output:** When the ESP32 sent the heartbeat, the Flask server logged the request:

    192.168.1.4 - - [17/Dec/2024 16:04:54] "GET /heartbeat HTTP/1.1" 200 -

---

Conclusion

The ESP32 deep sleep functionality with Wi-Fi communication was successfully implemented. The combination of the ESP32 and a **custom Python Flask server** allowed for efficient testing of the heartbeat signal. The server handled HTTP requests cleanly, and the ESP32 efficiently entered deep sleep to save power.