README

**ESP32 BLE Provisioning Assignment**

**1. Introduction**

This project is focused on setting up Wi-Fi provisioning for an ESP32 using Bluetooth Low Energy (BLE). The main idea is to let users send their Wi-Fi credentials over BLE using the ESP BLE Provisioning app. Once the credentials are received, the ESP32 connects to the Wi-Fi network. This implementation follows the assignment's requirements, and I've customized the process using the ESP-IDF framework.

**2. Tools I Used**

* **Hardware:**
  + ESP32 Development Board
* **Software:**
  + ESP-IDF Framework (v4.x+)
  + ESP BLE Provisioning Android App (for testing the provisioning process)
* **Other Tools:**
  + VSCode for editing and compiling the code
  + GitHub for version control and sharing the final project

**3. Setting Up the Environment**

Before diving into the code, I made sure everything was set up properly. Here's how I did it:

1. **Installed ESP-IDF:**  
   Followed the standard setup guide from Espressif to get ESP-IDF up and running. I used this installation guide to make sure everything was configured correctly.
2. **Tested the Development Environment:**  
   After setting up ESP-IDF, I tested it by flashing a basic example to my ESP32 board to confirm everything was working as expected.

**4. BLE Provisioning Setup and Code Explanation**

Here’s a breakdown of how the BLE provisioning is set up:

* **NVS Flash Initialization:**  
  Before any Wi-Fi-related operations, I initialize the NVS (Non-Volatile Storage) to store the Wi-Fi credentials. This step ensures that the ESP32 remembers the credentials even after a reset.
* **BLE Provisioning Service:**  
  I initialized the BLE service using wifi\_prov\_mgr\_init() and started provisioning with wifi\_prov\_mgr\_start\_provisioning(). The device advertises itself under the name "MyESP32\_Provisioning", which can be picked up by the mobile app.
* **Wi-Fi Configuration and Event Handling:**  
  I used event handlers to manage key events like provisioning start, receiving Wi-Fi credentials, and connecting to the Wi-Fi network. Everything is handled efficiently by the wifi\_prov\_mgr and custom event handlers I added to log the actions.
* **Wi-Fi Event Management:**  
  I also set up handling for events like disconnection from Wi-Fi. The device automatically tries to reconnect, and logs the events for easy debugging.
* **Event Group:**  
  An event group is used to manage the connection state. This ensures the system waits for provisioning to finish before proceeding with other tasks.

**5. Customizations I Made**

To make the code different from existing solutions and blogs, I implemented several customizations:

* Changed the BLE device name to "MyESP32\_Provisioning".
* Added extra logs to track the entire provisioning process more effectively.
* Enhanced the Wi-Fi reconnect logic to handle disconnections more gracefully.
* Optimized error handling for better stability in different scenarios.

**6. Testing and Results**

For testing, I used the **ESP BLE Provisioning App** on my Android phone to send the Wi-Fi credentials to the ESP32 device. The entire process worked seamlessly:

1. Connected to the ESP32 device via BLE.
2. Entered the Wi-Fi credentials in the app.
3. The ESP32 received the credentials and successfully connected to the Wi-Fi network.
4. I logged the entire process and captured screenshots for reference.

**7. How to Run the Project**

If you want to run this project on your ESP32, follow these steps:

**Clone the Project:**  
  
git clone <https://github.com/FastSloth8008/esp32-ble-provisioning.gi>

cd esp32-ble-provisioning

**Set Up ESP-IDF:** Ensure ESP-IDF is installed and set up as per the official Espressif guide.

**Build and Flash:**  
  
idf.py build

idf.py flash

idf.py monitor

**Provision Wi-Fi:**  
Download the **ESP BLE Provisioning App** on your phone and connect to the ESP32 device using the BLE interface. Follow the instructions in the app to provide Wi-Fi credentials.

**8. Conclusion**

This BLE provisioning project worked exactly as expected, allowing users to send Wi-Fi credentials to an ESP32 over BLE. I’ve ensured the code is well-documented and follows the assignment’s requirements, with customisations to make it unique.