**In the name of God**

**Parsa Aghaali 400521072**



**ESP32 OTA Firmware Update Project Report**

**1. Project Overview**

The objective of this project is to implement an **Over-the-Air (OTA) firmware update** system for an ESP32 microcontroller. OTA updates allow the ESP32 to download and install new firmware versions from a server without requiring physical access, reducing downtime and maintenance costs.

**2. Goals and Objectives**

The project aims to:

1. Connect the ESP32 to a Wi-Fi network.
2. Check for new firmware availability hosted on a remote HTTP server.
3. Download and install the new firmware only if a newer version is detected.
4. Store the firmware version persistently using **NVS (Non-Volatile Storage)**.
5. Restart the ESP32 automatically after a successful update to apply changes.
6. Display system information, including the boot reason, firmware version, and update status.

**3. System Requirements**

**Hardware:**

* ESP32 Development Board.

**Software Tools:**

* Arduino IDE for programming and flashing the ESP32.
* An HTTP Server to host the firmware binary file (newfirm.bin).

**Libraries:**

1. **WiFi.h**: Establishes a connection to the Wi-Fi network.
2. **HTTPClient.h**: Facilitates HTTP communication with the server.
3. **HTTPUpdate.h**: Handles the OTA update process.
4. **Preferences.h**: Stores data persistently in the ESP32's NVS.
5. **esp\_system.h**: Provides system-level functions, such as restarting the device.

**4. Project Workflow**

**4.1 Setup Phase**

* **Wi-Fi Connection**:
  + The ESP32 connects to a predefined Wi-Fi network using the SSID and password.
  + If the connection is unsuccessful, it retries until it connects.
* **Firmware Version Check**:
  + The firmware version is hardcoded into the code using #define FIRMWARE\_VERSION.
  + The ESP32 retrieves the previously stored firmware version from NVS.
* **Reset Reason Display**:
  + The system displays the boot reason (e.g., Power-on reset, Software reset) using esp\_reset\_reason() for debugging.

**4.2 OTA Update Mechanism**

* **Firmware Comparison**:
  + If the stored firmware version does not match the current version:
    - The ESP32 initiates the OTA update process.
  + If versions match, it skips the update and reports that the firmware is up-to-date.
* **OTA Update Execution**:
* The HTTPUpdate library downloads the firmware from the HTTP server.
* Three possible outcomes are handled:
  1. **HTTP\_UPDATE\_FAILED**: Prints an error message with the HTTP update failure details.
  2. **HTTP\_UPDATE\_NO\_UPDATES**: Indicates no new updates are available.
  3. **HTTP\_UPDATE\_OK**: Confirms a successful update, stores the new firmware version in NVS, and restarts the ESP32.
*  **Restarting After Update**:
  + After a successful update, the ESP32 restarts using ESP.restart() to apply the new firmware.

**5. Code Flow**

**setup() Function**

1. Initializes the serial communication.
2. Displays the current firmware version.
3. Prints the reset reason for system debugging.
4. Connects the ESP32 to the Wi-Fi network.
5. Checks for firmware updates and performs OTA if required.

**loop() Function**

* Contains a simple delay(1000) to prevent unnecessary reboots or tight loops.

**checkAndUpdateFirmware()**

1. Initializes NVS to retrieve the stored firmware version.
2. Compares the stored version with the current firmware version.
3. Triggers the OTA update if versions differ.

**performOTAUpdate()**

1. Connects to the server and downloads the new firmware.
2. Handles possible outcomes (OK, NO\_UPDATES, FAILED).
3. On success, updates the stored version and restarts the ESP32.

**printResetReason()**

* Displays the reason for the ESP32's last reset for diagnostic purposes.

**6. Testing and Validation**

**Test Cases:**

|  |  |  |
| --- | --- | --- |
| Test Condition | Expected Result | Outcome |
| Connect to Wi-Fi | ESP32 connects to the Wi-Fi network | Successful |
| Firmware Version Comparison | If versions differ, OTA update starts | Successful |
| OTA Update with Server Hosting New Firmware | ESP32 downloads and applies new firmware | Successful |
| OTA Update with Same Firmware | ESP32 reports firmware is up-to-date | Successful |
| Restart After Update | ESP32 restarts automatically | Successful |

**7. Results**

The project met all objectives:

* **Wi-Fi Connectivity**: Successfully connects to a Wi-Fi network.
* **Firmware Update**: The ESP32 fetches and applies new firmware if available. The firmware that updated was LED blinking that completely configured in the ESP32 hardware.
* **NVS Storage**: Ensures updates are not redundantly applied.
* **Automatic Restart**: The ESP32 restarts after a successful update, ensuring the new firmware takes effect.
* **Reset Logging**: The ESP32 provides a detailed reset reason for debugging.

**8. Challenges and Solutions**

1. **Repeated Firmware Update Issue**:
   * Issue: The ESP32 repeatedly applied the firmware update due to a missing persistent version check.
2. **Handling Update Failures**:
   * Implemented error handling for failed updates using HTTP error codes.
3. **Wi-Fi Connection Reliability**:
   * Added a retry loop to ensure a stable Wi-Fi connection.