**Analysis of ESP32 SIM7600 IoT Device Firmware**

**Overview**

This document analyzes an Arduino sketch designed for an ESP32 microcontroller interfaced with a SIM7600 cellular modem. The firmware enables secure IoT functionality, including MQTT communication over cellular networks, Over-The-Air (OTA) firmware updates, and device provisioning. The code leverages libraries such as TinyGSM, Adafruit\_NeoPixel, and mbedTLS for hardware control, communication, and cryptography. Below, we detail the implementation, its advantages, and its security features.

**1. Implementation Details**

**1.1 Hardware Integration**

* **SIM7600 Modem**:
  + Controlled via the TinyGSM library over UART (pins 16 and 17).
  + Supports GPRS for internet connectivity and AT commands for MQTT over SSL/TLS.
* **Peripherals**:
  + **RGB LED**: A single NeoPixel on pin 48 for status indication.
  + **LCD (Optional)**: An I2C LCD (address 0x27, pins 35 and 36) for displaying status messages.
  + **GPIO Pins**: Power control (pin 21), factory reset (pin 4), and LED status (pin 13).
* **Serial Communication**:
  + HardwareSerial for modem interaction and SerialMon for debugging at 115200 baud.

**1.2 State Machine**

* **Purpose**: Manages the device lifecycle through a finite state machine (SetupState).
* **States**:
  + STATE\_INIT\_MODEM: Initializes the modem.
  + STATE\_WAIT\_NETWORK: Waits for cellular network registration.
  + STATE\_CONNECT\_GPRS: Establishes GPRS connection.
  + STATE\_UPLOAD\_CERTIFICATE: Uploads SSL certificate.
  + STATE\_SETUP\_SSL: Configures SSL for MQTT.
  + STATE\_SETUP\_MQTT: Initializes MQTT service.
  + STATE\_CONNECT\_MQTT: Connects to the MQTT broker.
  + STATE\_SUBSCRIBE\_MQTT: Subscribes to MQTT topics.
  + STATE\_RUNNING: Normal operation mode.
  + STATE\_WAIT\_PROVISION: Awaits provisioning response.
  + STATE\_ERROR: Handles failures with cleanup and reset.
  + Recovery states (STATE\_RECOVER\_NETWORK, STATE\_RECOVER\_GPRS, STATE\_RECOVER\_MQTT): Restores connectivity.
* **Operation**: Transitions between states with retry logic (max 10 retries, 2-second delay).

**1.3 MQTT Communication**

* **Broker**: Connects to u008dd8e.ala.dedicated.aws.emqxcloud.com on port 8883 (SSL/TLS).
* **Topics**:
  + **Publish**: esp32\_status (device status), dev\_pass\_req (provisioning request).
  + **Subscribe**: server\_cmd (commands), OTA\_Update (firmware updates), dev\_pass\_res (provisioning response).
* **Features**:
  + Handles Unsolicited Result Codes (URCs) for asynchronous MQTT events (e.g., connection loss, message receipt).
  + Supports encrypted message exchange.

**1.4 OTA Firmware Updates**

* **Mechanism**:
  + Receives base64-encoded firmware chunks via MQTT.
  + Uses ESP32 OTA APIs (esp\_ota\_\*) to write to the next partition.
* **Features**:
  + Tracks chunk numbers and sizes (max 1024 bytes per chunk).
  + Verifies integrity with SHA-256 hashing.
  + Supports rollback to the previous firmware if validation fails.
  + Requests missing chunks if incomplete.

**1.5 Provisioning**

* **Process**:
  + Requests credentials from the server using the device’s IMEI.
  + Subscribes to dev\_pass\_res and publishes to dev\_pass\_req.
  + Updates credentials upon receiving an encrypted response.
* **Timeouts**: 20-minute provisioning timeout, 10-second request interval.

**1.6 Security Features**

* **SSL/TLS**: Uses a preloaded CA certificate for secure MQTT communication.
* **AES-256 Encryption**: Encrypts/decrypts messages with a static key and IV.
* **Credential Storage**: Stores MQTT credentials in NVS (Non-Volatile Storage).
* **Factory Reset**: Resets to default credentials and firmware via a physical pin or MQTT command.

**1.7 Error Handling and Recovery**

* **Retries**: Attempts failed operations up to 10 times with a 2-second delay.
* **Reset**: Resets modem and state machine on persistent failures.
* **Monitoring**: Periodically checks network and GPRS connectivity (every 5 seconds).

**1.8 Diagnostics**

* **Logging**: Outputs detailed status via SerialMon.
* **Visual Feedback**: Uses RGB LED and optional LCD for status indication.

**2. Advantages**

**2.1 Reliability**

* **State Machine**: Ensures a structured setup with retries and recovery, reducing downtime from network or modem issues.
* **OTA Rollback**: Prevents bricking by reverting to the previous firmware if the new one fails validation.

**2.2 Flexibility**

* **Provisioning**: Supports both provisioned and unprovisioned states, suitable for initial deployment or reconfiguration.
* **Modularity**: Optional LCD and configurable pins allow adaptation to different hardware setups.

**2.3 Scalability**

* **MQTT**: Enables cloud integration for managing multiple devices remotely.
* **OTA**: Facilitates firmware updates without physical access, ideal for large deployments.

**2.4 Security**

* **Encrypted Communication**: Protects data in transit and at rest.
* **Integrity Checks**: Ensures firmware updates are authentic and complete.

**3. Security Analysis**

**3.1 Implemented Security Features**

**3.1.1 SSL/TLS for MQTT**

* **Details**:
  + Uploads a CA certificate (iot\_inverter2.pem) to the SIM7600 modem.
  + Configures SSL with AT+CSSLCFG commands (SSLv4, CA certificate, auth mode 1).
  + Connects to the MQTT broker over port 8883.
* **Advantage**:
  + Ensures end-to-end encryption between the device and the broker.
  + Prevents eavesdropping and man-in-the-middle (MITM) attacks on cellular networks.
* **Limitation**:
  + Relies on a single, hardcoded certificate. If compromised or expired, the device cannot connect securely without an update.

**3.1.2 AES-256 Encryption**

* **Details**:
  + Uses AES-256-CBC with a 32-byte static key and 16-byte static IV.
  + Implements PKCS7 padding for block alignment.
  + Encrypts messages before publishing and decrypts received payloads.
* **Advantage**:
  + Adds an additional layer of encryption beyond SSL/TLS, protecting message content even if SSL is bypassed.
  + Ensures confidentiality of sensitive data (e.g., provisioning responses).
* **Limitation**:
  + **Static Key and IV**: Using fixed values reduces security. A static IV allows replay attacks, and a compromised key affects all devices.
  + **Key Management**: The key is hardcoded, making it difficult to rotate or update securely.

**3.1.3 Credential Management**

* **Details**:
  + Stores MQTT username, password, and client ID in NVS using the Preferences library.
  + Supports provisioning via MQTT, updating credentials dynamically.
  + Resets to defaults (ESP32, 12345) on factory reset.
* **Advantage**:
  + Persistent storage prevents credential loss on reboot.
  + Provisioning allows secure initial setup without hardcoded sensitive data.
* **Limitation**:
  + Default credentials (ESP32, 12345) are weak and predictable, posing a risk if provisioning fails or is delayed.

**3.1.4 OTA Security**

* **Details**:
  + Verifies firmware integrity with SHA-256 hashing.
  + Encrypts OTA payloads (assumed via AES-256, as messages are encrypted).
  + Supports rollback to the previous partition if validation fails.
* **Advantage**:
  + Prevents installation of corrupted or malicious firmware.
  + Rollback ensures device availability even after a failed update.
* **Limitation**:
  + No signature verification (e.g., RSA), so an attacker with the AES key could craft a valid update.

**3.1.5 Factory Reset**

* **Details**:
  + Triggered by a physical pin or MQTT command (RESET\_PASSWORD).
  + Clears NVS and reboots to the factory partition.
* **Advantage**:
  + Allows recovery from compromised states or misconfiguration.
* **Limitation**:
  + Physical access requirement for pin-based reset; MQTT-based reset requires prior authentication.

**3.2 Security Strengths**

* **Layered Encryption**: Combining SSL/TLS and AES-256 provides defense-in-depth.
* **Integrity Verification**: SHA-256 hashing ensures firmware authenticity.
* **Secure Provisioning**: Encrypting provisioning requests/responses protects credentials in transit.

**3.3 Security Weaknesses**

1. **Static AES Key and IV**:
   * Hardcoded values are vulnerable to extraction via reverse engineering or physical attacks.
   * Static IV enables replay attacks, undermining encryption security.
2. **Weak Default Credentials**:
   * ESP32 and 12345 are easily guessable, risking unauthorized access before provisioning.
3. **Lack of Certificate Validation**:
   * No client certificate or mutual TLS; relies solely on server-side CA validation.
4. **No Firmware Signing**:
   * OTA updates lack cryptographic signatures, relying only on hashing and encryption.
5. **Physical Security**:
   * Factory reset pin could be exploited if an attacker gains physical access.

**3.4 Recommendations for Improvement**

1. **Dynamic Key Management**:
   * Generate unique AES keys per device (e.g., derived from IMEI and a secret) and use a random IV per message.
   * Implement key rotation via provisioning.
2. **Stronger Defaults**:
   * Use a random, device-specific default password instead of 12345.
3. **Mutual TLS**:
   * Add client certificates for mutual authentication with the MQTT broker.
4. **Firmware Signing**:
   * Implement RSA or ECDSA signatures for OTA updates to verify authenticity, not just integrity.
5. **Secure Boot**:
   * Enable ESP32 secure boot to prevent unauthorized firmware execution.
6. **Rate Limiting**:
   * Limit provisioning requests to prevent brute-force attacks on the server.

**4. Conclusion**

The firmware implements a robust IoT solution with cellular connectivity, MQTT communication, OTA updates, and provisioning. Its advantages include reliability, flexibility, and scalability, making it suitable for remote device management. Security is a strong focus, with SSL/TLS, AES-256 encryption, and OTA integrity checks providing significant protection. However, weaknesses such as static cryptographic keys, weak defaults, and lack of firmware signing limit its resilience against advanced attacks. Implementing the recommended improvements would elevate its security posture, making it more suitable for critical applications.