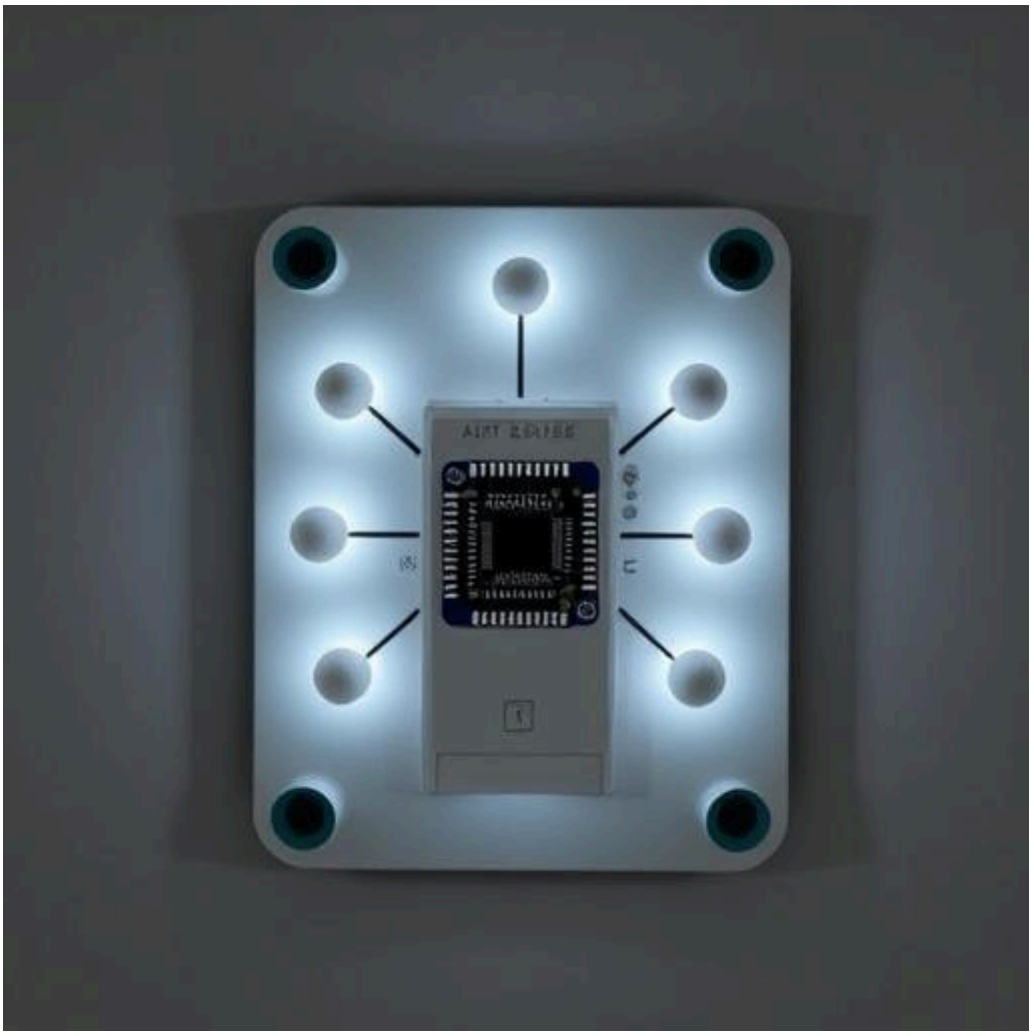
Specification Document Draft



Smart Light Switching System
Network Architecture Specification
For Light Switch Project

Prepared by:
Embedded Systems Team

Date: *22/12/2024*

Table of Contents

---

1. General Network Architecture

The proposed system architecture is designed to efficiently control and monitor lights across multiple classroom blocks using a mesh network. The following points summarize the network architecture:

1. **ESP32 as Master Gateways:**
    - Acts as a central communication bridge between the local network/WAN and the ESP8266 nodes.
    - Connects to library or administration routers for local and LAN network access.
    - Sends and receives MQTT packets to and from the main server hosting the central website.
2. **Main Server:**
    - Hosts the main website for user interaction and monitoring.
    - Sends control commands encapsulated in JSON packets to the ESP32 gateways.
    - Receives status updates from the network via MQTT for real-time monitoring and logging.
3. **ESP32 Placement:**
    - Strategically placed in the middle of the faculty building (INSAT) to maintain stable ESP-NOW connections.
    - Ensures seamless communication with nodes on both the left and right sides of the building.
4. **ESP8266 Nodes:**
    - Distributed as pairs within each 4-classroom block to maintain reliable connectivity.
    - Communicate via ESP-NOW to broadcast messages through the mesh network.
    - Relay messages to the specified node, ensuring commands reach the correct destination.

5. **Message Flow:**
    - Commands from the server are received by the ESP32, parsed, and broadcasted to the mesh network.
    - Nodes pass the message through the mesh until the specified ESP8266 processes it.
    - The addressed node updates its status, encapsulates it in a JSON packet, and sends it back to the ESP32.
    - The ESP32 transmits the updated status to the main server using MQTT.

6. **Network Stability:**
    - ESP32 maintains a strong ESP-NOW connection with nodes on both sides of the building.
    - Each ESP8266 node ensures redundancy by periodically broadcasting its status to neighboring nodes.

7. **Integration with LAN/WAN:**
    - ESP32 connects to routers via a predefined SSID and password.
    - Uses MQTT for robust and secure communication with the central server.

8. **Scalability:**
    - The architecture supports additional nodes and ESP32 gateways for expanded coverage.
    - Modular design ensures new classroom blocks can be added with minimal reconfiguration.

General network Sequence Diagram

Link:
https://www.plantuml.com/plantuml/png/VPDFRzf04CNl-occd41gSg1LLFbG1KDIfHM4OcYFrSIUa5NiNNTtd8X--kuVZ9O4STg1-RtHp7lojMKqt3Mr-53HerG4c-BcJ_7ZRh6Sh2Rp0jXq9EopL2qQRC11sDLygnKNyX_1-BCGMB8sCDXOCWD02xuG3cTPoXOLms2EKa51vYKIiNp3h7J7-tOiZzitnVBJ6BuXqolk0XN-UGTz6LzTmK9N11DFXN9oXieFL7w6cXwegH3-FhZukhyYSw-geDIAZQxTspJeVW-_5_UBaOXCII_2MbR819wgzRPN96YyFuDRRLxHL33_kLv3sptKqZx5mRre94I4dImQai6CQka6dLK5DD4iqLY2b84XNMulFy8QpPOukd0tIvJ1s0noex4gqJ8qP2rk2Lua-h4NY_IVC3oMPz5ejtLGzsFo35PKu-umWZLm52el7BvLveboQNJfPBtsNN_J1hSj8tUsx_Q7n-nY3nwmVEvp2-qqYnDkHjae9f1cV3Au9D-DzT1CsjOVtOK5-osEzv-US0QLvYSoUskqgdSW7y7GtquQSjbUkvFy1_qV

2. Slave Nodes (ESP8266)

The ESP8266 devices act as slave nodes within the mesh network, responsible for controlling lights and relaying information.

Communication Protocol:

- **ESP-NOW:**
  - **Range(Tested):** Effective up to 20 meters indoors, extendable with intermediate nodes.
  - **Encryption:** AES-128 encryption ensures secure communication.
  - **Signal Strength(Tested):** Signal strength can be monitored using RSSI (Received Signal Strength Indicator).
  - **Max Speed(Tested):** Communication speed is approximately 2 Mbps(Directly between two nodes).
  - **Packet Size:** Maximum payload size is 250 bytes.

Packet Structure(JSON):


```
"PacketStatusCode": 200,

"DestinationFlag": "ServerToNode",

"Body": {

 "Source": {

  "ID": "NodeA",

  "MAC": "AA:BB:CC:DD:EE:FF"

 },

 "LightStatus": "ON",

 "CommandFlag": "ToggleLight",

 "Destination": {

  "ID": "NodeB",

  "MAC": "FF:EE:DD:CC:BB:AA"

 },

 "ClassroomIndex": 2,

 "Path": ["NodeA", "NodeB", "NodeC","NodeD"]},

"Timestamp": "2024-12-22T14:30:00Z"
```

Explanation of JSON Packet Fields

PacketStatusCode:

- Represents the status of the communication process.
- Common values:
  - `200`: Success
  - `408`: Timeout
  - `404`: Connection Lost

DestinationFlag:

- Identifies the direction of communication:
  - `"ServerToNode"`: Packet sent from the server to a node.
  - `"NodeToServer"`: Packet sent from a node to the server.

Body:

- **Source**:
  - `ID`: Unique identifier of the sender.
  - `MAC`: Sender's MAC address for addressing and encryption purposes.
- **LightStatus**:
  - Indicates the current state of the light controlled by the node.
- **CommandFlag**:
  - Command for the node, such as toggling the light state or checking status.
- **Destination**:
  - `ID`: Target node's identifier.
  - `MAC`: Target node's MAC address.
- **ClassroomIndex**:
  - Specifies which classroom to control in case a node manages multiple rooms.
- **Path**:
  - Ordered list of nodes for relaying the packet to the destination.
- **Timestamp**:
  - timestamp for tracking and managing packet timeouts.

Control Flow of a Node (ESP8266)

Initialization

1. **Boot and Initialization:**
   - Initialize hardware peripherals (GPIO, Wi-Fi module).
   - Load the AES-128 encryption key from flash memory.
2. **Encryption Setup:**
   - Verify the integrity of the stored key.
   - Preload encryption configurations into the ESP-NOW module.

Operational States

1. **Listening Mode (Light Sleep):**
   - Enter light sleep mode to conserve power.

- Keep the radio module active to listen for ESP-NOW packets.
- Use interrupt service routine (ISR) to wake the CPU on packet reception.

2. **Packet Reception:**
   - Wake up the CPU upon receiving a packet.
   - Validate the packet structure and decrypt it using the shared AES-128 key.
   - Verify the integrity of the packet (e.g., using checksum or hash).
3. **Command Handling:**
   - Parse the `CommandFlag` and `DestinationFlag`:
     - If the node is the destination, execute the command (e.g., toggle light).
     - If the node is an intermediate relay, forward the packet along the defined path.
   - Log the event in memory for redundancy.
4. **Fault Handling:**
   - If the destination is unreachable, send a status packet (`PacketStatusCode: 404`) back to the server.
   - Retry failed transmissions up to three times.
   - If no acknowledgment is received, inform the neighboring node to assist.
5. **Light Control:**
   - Update GPIO states to toggle or adjust the light based on the command.
   - Confirm the light status and prepare a status packet for response.
6. **Packet Transmission:**
   - Construct a JSON packet with updated `PacketStatusCode` and `LightStatus`.
   - Encrypt the packet and send it to the next hop in the path.

Sleep and Maintenance

1. **Periodic Status Updates:**
   - Broadcast the node's current status (e.g., light state) to neighbors at regular intervals.
   - Ensure mesh integrity by validating connections with neighboring nodes.
2. **Sleep Mode Reentry:**
   - Reenter light sleep mode if no activity is detected for a defined period.

Control Flow Diagram of a single Node:

Link:

https://www.plantuml.com/plantuml/png/VPDHRzem58NV_Ikk-XIajjAwbmaUrWt2YjPs6O7ssOilmIBOqVNDrcpQVvyTDncmTho0p7VEkVQnhnrBucPVYUw762-CPYqh_HFXtYgSYF6jbGe-fFd5cwjta9g2sfgrDV0PsyaHVoD9_P24a7pA5b_EuL5VpFLe8iGuDOm4jxhSCkGLOWrtdNVj8tLdTlOeDSF64clos26VmIy1_ZC-MOTa4RoegHciwHUNM60tb4_xBRncXx2kOPgjG_g37qz9nfxEcPg26_83w0qapwiTzNu4l4K3IOjk50UOORVZNhnssfJ3SOGpYDW3ajwqFHM6B4bp2mbUbfVdCDrYiNFDVXJna3imhqLtlfaaXp2r-xqqQbx94lmtpD2nDZ9iEQn5qy7OUIVjklw2vo7zOafCIf-mQ7Z8WcHboxB2MDjeSWolw-v4Gz6GF1VBaXjt2cQ4IcM49RhQ6jUNS4Bbw0C7X2saUVRswYKefa3C3sIEz1q9ahklgzL1E5OEZ-LpI_vsghuYBxt79uORMn-eZD8RyOz-l4Ic5Yp1JBlYhqaKHb4K_1UEuE6jpv2qLReutjPm-Vk_PLMry96iL8Lq78kDYhWDV_xVBBbm-OMFly1NtfjZMujhDCe_-3y0

3. Master Node (ESP32)

The **ESP32 (Master Node)** now serves as a gateway between the ESP8266 slave nodes and the central server. It handles both ESP-NOW communication with the slave nodes and Wi-Fi communication with the central server via MQTT. By using multicore processing, the ESP32 ensures that both communication protocols are handled efficiently, maintaining the stability and responsiveness of the system.

Communication:

- **ESP-NOW:** Interfaces with slave nodes for data exchange.
- **Wi-Fi (Station Mode):**
    - Connects to AP using a predefined SSID and password.
    - Sends MQTT packets to the central server.

MQTT Configuration:

- **Broker:** Hosted on the central server.
- **Topics:** Organized by block and node ID (e.g.,`/nodeA/classroom1/status`).
- **Payload:** JSON-formatted data containing:
    - Node ID.
    - Light state.
    - Error codes.
    - RSSI.

Multicore Processing:

- **Core 0:** Handles Wi-Fi and MQTT communication.
- **Core 1:** Manages ESP-NOW interactions and generates JSON data.

Fault Handling:

- Retries MQTT connection if disconnected.
- Reports faulty nodes to the central server.
- Monitors signal strength and reconfigures node priorities if necessary.

4. Conclusion

This document presents the proposed architecture for the Smart Light Switching System, highlighting key components such as the ESP32 gateway, ESP8266 slave nodes, and their communication protocols (ESP-NOW and MQTT). The system ensures scalability, reliability, and secure communication through AES-128 encryption and MQTT packet handling. The fault-handling and redundancy mechanisms are designed to maintain a stable and fault-tolerant network, supporting real-time light control and monitoring across multiple classroom blocks.