**SMART CABIN AUTOMATION**

**Abstract**

This project presents an efficient and scalable home automation system built around the ESP32 microcontroller, integrating four independently controlled lighting circuits and a single-speed ceiling fan with ON/OFF control. The system combines local manual control via wall-mounted push switches with seamless remote operation through the ESP RainMaker IoT platform, offering users both convenience and flexibility. The implemented firmware utilizes AceButton for reliable debouncing and event handling of physical switches, ensuring stable and responsive toggling of appliances. All relay-driven loads are interfaced safely with proper power isolation. The fan is controlled with a single relay for maximum speed operation, simplifying hardware and software complexity. Provisioning of Wi-Fi connectivity is streamlined through BLE and QR code-based setup. A factory reset feature provides device recovery options. This solution demonstrates a practical and cost-effective approach to smart home control, balancing ease of installation with extensibility for future enhancements.

**Key Features**

* **Multi-Device Control:**  
  Four independent lighting circuits and one ceiling fan (ON/OFF max speed) controllable both locally and remotely.
* **Local Physical Control:**  
  Wall-mounted push switches for each light and fan allow manual toggling, with proper hardware debouncing via the AceButton library.
* **Cloud Integration via ESP RainMaker:**  
  Devices are registered with ESP RainMaker, enabling secure, reliable remote control, device monitoring, OTA firmware updates, and easy provisioning.
* **Simplified Fan Control:**  
  Fan controlled by a single relay for ON/OFF operation at maximum speed.
* **Reliable Input Handling:**  
  Internal pull-up resistors on input GPIOs with debouncing to prevent unintended toggles.
* **Factory Reset Facility:**  
  Physical reset button held for 5+ seconds triggers a factory reset, allowing easy recovery or device reprovisioning.
* **Power and Safety Considerations:**  
  Separate, stable power supplies for ESP32 and relays, common ground, relay coils driven via transistor stages with flyback diodes, isolating AC loads properly.
* **Serial Debugging:**  
  Serial output messages help monitor device behavior during development and troubleshooting.

**Objectives**

* **Provide seamless integration of local physical switches and cloud control:** Ensure state synchronization between manual inputs and app/cloud commands without conflict.
* **Create a lightweight, stable firmware:** Using efficient libraries (AceButton, RainMaker) to manage input events and cloud communication with minimal latency.
* **Simplify fan control hardware and logic:** Removing multi-speed speed control to maximize reliability and ease of use.
* **Enable easy device provisioning and management:** Using BLE QR code provisioning and OTA updates for user convenience.
* **Ensure system robustness and safety:** Including debouncing, power isolation, and protective relay driving circuits to prevent hardware damage and unintended behavior.
* **Build an extensible platform:** A modular software architecture that can be expanded later with sensors, multi-speed fans, or voice assistants as needed.

**Libraries**

1. **ESP RainMaker**

* **Library:** RMaker.h
* **Purpose:** Provides device registration, cloud connectivity, OTA (over-the-air) upgrades, parameter (ON/OFF/status) syncing with the RainMaker mobile app and cloud. Enables users to control and monitor devices remotely without building custom cloud infrastructure.
* **Source:** Official Espressif component and Arduino library.

2. **WiFi and WiFiProv**

* **Libraries:** WiFi.h, WiFiProv.h
* **Purpose:** Handles WiFi connections on the ESP32. WiFiProv is specifically for BLE-based provisioning, making initial setup and network changes via QR code simple for end-users.

3. **AceButton**

* **Library:** AceButton.h
* **Purpose:** Advanced button/switch manager for Arduino/ESP32. Handles:
  + Debounce logic on digital inputs (physical wall switches)
  + Detects events like press, release, long press, etc.
  + Makes button response robust and responsive in real-world (bouncy switches)

**Softwares Used**

1. **Arduino IDE**

* Used for writing, compiling, and uploading C++ firmware to the ESP32 microcontroller.
* Provides integration with board support packages, library management, and serial monitoring.

2. **ESP RainMaker Platform**

* Espressif’s official IoT platform, used for:
  + Device-cloud management
  + Mobile app control (iOS, Android)
  + Secure (OTA) over-the-air firmware updates
  + BLE-based provisioning (initial WiFi setup via QR code/app)
* Supported through the RainMaker Arduino library and mobile apps.

**Pin Connections**

|  |  |  |  |
| --- | --- | --- | --- |
| **ESP32 GPIO** | **Function** | **Description** | **Direction** |
| **32** | Wall switch for Light 1 | Switch input for Light 1 | Input |
| **35** | Wall switch for Light 2 | Switch input for Light 2 | Input |
| **34** | Wall switch for Light 3 | Switch input for Light 3 | Input |
| **39** | Wall switch for Light 4 | Switch input for Light 4 | Input |
| **33** | Wall switch for Fan | Switch input for Fan | Input |
| **15** | Relay for Light 1 | Controls relay to switch Light 1 | Output |
| **2** | Relay for Light 2 | Controls relay to switch Light 2 | Output |
| **4** | Relay for Light 3 | Controls relay to switch Light 3 | Output |
| **22** | Relay for Light 4 | Controls relay to switch Light 4 | Output |
| **18** | Relay for Fan | Controls relay to switch Fan | Output |

**Schematic Diagram**

**A computer diagram of a circuit board

AI-generated content may be incorrect.**

**System Architecture**

1. **Hardware Layer**

* **ESP32 Microcontroller:**
  + Central controller for all automation logic, wireless communication, and device control.
* **Relay Outputs (GPIOs 15, 2, 4, 22, 18):**
  + Drive relays for four lights and one fan (ON/OFF at maximum speed).
  + Each relay is isolated from the ESP32 via a driver circuit (typically NPN transistor + flyback diode).
* **Physical Inputs (GPIOs 32, 35, 34, 39, 33, 0):**
  + Wall-mounted push switches for local control of each light and fan.
  + Factory reset via dedicated button (GPIO0).
  + All switches connect between pin and ground; ESP32 uses internal pull-ups.
* **Relay Power Supply:**
  + Stable 5V supply for relays, separate from ESP32's 3.3V logic to protect against voltage dips on switching.

2. **Firmware/Software Layer**

* **ESP RainMaker SDK:**
  + Manages device communication with the cloud, handles provisioning (via BLE and QR code), exposes device state to the mobile app, and enables cloud-to-device control/feedback.
  + Supports OTA (over-the-air) firmware updates.
* **AceButton Library:**
  + Handles debouncing and event management for each physical input switch to ensure reliable toggling.
* **Event Loop Logic:**
  + Continuously polls physical switches.
  + Processes incoming cloud/app commands and updates device states.
  + Synchronizes local and remote state to prevent conflict.
* **Reset Handler:**
  + Detects long-press of reset button and triggers factory reset and re-provisioning if needed.

3. **Cloud & App Layer**

* **ESP RainMaker Cloud:**
  + Registers the device securely and maintains its state.
  + Relays user commands and device data between the ESP32 and mobile app.
  + Supports user management, device sharing, and notifications.
* **RainMaker Mobile Application (Android/iOS):**
  + Allows users to remotely monitor and operate all devices (lights, fan).
  + Supports provisioning, OTA updates, and device status feedback in real-time.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Conclusion and Results**

The ESP32-based home automation system integrating four independent lighting controls and a ceiling fan ON/OFF relay demonstrated effective real-time control, robustness, and user convenience. The combination of local wall-mounted switches with cloud connectivity via ESP RainMaker allowed seamless bi-directional operation: physical inputs were reliably detected and debounced using AceButton, and remote commands updated device states responsively.

**Results:**

* **Reliable Device Control:** Devices responded promptly to both local button presses and cloud app commands, with relay outputs switching lights and fan within 1-2 seconds. State synchronization between the ESP32 and the RainMaker cloud ensured consistent device status across all control interfaces.
* **Stable Operation:** The hardware design with dedicated relay driver circuits and separated power supplies maintained voltage stability and protected the microcontroller from electrical noise or resets during relay switching.
* **Provisioning and Management:** BLE-based Wi-Fi provisioning via QR codes simplified the initial setup process, while factory reset functionality enabled straightforward recovery and reprovisioning. OTA updates supported future firmware enhancements without physical access.
* **Simplified Fan Control:** By limiting fan control to an ON/OFF single-relay approach at maximum speed, hardware complexity was minimized, increasing system reliability.
* **Energy Efficiency and Safety:** The ability to automate lighting and fan control reduced unnecessary power consumption, and proper relay isolation ensured safe interaction with AC mains.

Overall, this project confirms that an ESP32-driven home automation system leveraging RainMaker cloud services and carefully designed local input handling can provide a cost-effective, extensible, and user-friendly solution for modern smart homes. With the modular firmware structure, the system is poised for future upgrades including multi-speed fan control or sensor integration.