// --- Integrated Code with WiFi Failover and Active-Low Logic ---

// WiFi Credentials

const char\* ssid1 = "Airtel\_prad\_5900";

const char\* password1 = "air97139";

const char\* ssid2 = "Deva";

const char\* password2 = "Vittala@26";

// device names (change names as per requirement)

String Device\_1\_Name = "Moon Light";

String Device\_2\_Name = "Balcony Light";

String Device\_3\_Name = "Light";

String Device\_4\_Name = "Fan";

//Update the HEX code of IR Remote buttons 0x<HEX CODE>

#define IR\_Button\_1 0x1FEE01F

#define IR\_Button\_2 0x1FE10EF

#define IR\_Button\_3 0x1FE906F

#define IR\_Button\_4 0x1FE50AF

#define IR\_Button\_5 0x1FEE0BF

#define IR\_Button\_6 0x1FE10BF

#define IR\_Button\_7 0x1FE806F

#define IR\_Button\_8 0x1FE50BF

#define IR\_All\_On 0x1FE807F

#define IR\_All\_Off 0x1FE48B7

#include <WiFi.h>

#include <Espalexa.h>

#include "BluetoothSerial.h"

#include <IRremote.h>

#include <Preferences.h>

#include <AceButton.h>

using namespace ace\_button;

#if !defined(CONFIG\_BT\_ENABLED) || !defined(CONFIG\_BLUEDROID\_ENABLED)

#error Bluetooth is not enabled! Please enable it in menuconfig

#endif

BluetoothSerial SerialBT;

Espalexa espalexa;

Preferences pref;

// define the GPIO connected with Relays and switches

#define RelayPin1 26 //fan

#define RelayPin2 25 //light

#define RelayPin3 33 //moon light

#define RelayPin4 32 // socket

// Use INPUT\_PULLUP for active-low buttons

#define SwitchPin1 35 // fan green

#define SwitchPin2 34 //light blue

#define SwitchPin3 39 //moon light // purple

#define SwitchPin4 36 //socket //white

#define wifiLed 22

#define IR\_RECV\_PIN 21

#define WiFiResetButton 0 //BOOT button

// Relay State (Active-low: LOW is ON, HIGH is OFF)

bool toggleState\_1 = HIGH;

bool toggleState\_2 = HIGH;

bool toggleState\_3 = HIGH;

bool toggleState\_4 = HIGH;

ButtonConfig config;

AceButton button1(&config, SwitchPin1);

AceButton button2(&config, SwitchPin2);

AceButton button3(&config, SwitchPin3);

AceButton button4(&config, SwitchPin4);

void handleEvent(AceButton\*, uint8\_t, uint8\_t);

int wifiFlag = 0;

char bt\_data;

IRrecv irrecv(IR\_RECV\_PIN);

decode\_results results;

// --- WiFi Failover Variables ---

int currentWiFi = 0;

unsigned long wifiRetryTimer = 0;

unsigned long bootTime = 0;

const unsigned long WIFI\_RETRY\_DELAY = 30000;

const unsigned long BOOT\_SCAN\_DELAY = 30000; // 30 seconds after boot

int wifiRetryCount = 0;

const int MAX\_WIFI\_RETRIES = 3;

bool bootScanDone = false;

// --- Function Prototypes ---

void firstLightChanged(uint8\_t brightness);

void secondLightChanged(uint8\_t brightness);

void thirdLightChanged(uint8\_t brightness);

void fourthLightChanged(uint8\_t brightness);

void scanWiFiNetworks();

void connectWifi();

void handleWiFiFailover();

void all\_SwitchOn();

void all\_SwitchOff();

void addDevices();

void sendFeedback(int relay, int value);

void ir\_remote();

void bluetooth\_control();

void WiFiEvent(WiFiEvent\_t event) {

// This event handler is used to prevent the built-in auto-reconnect

// so we can manage it with our failover logic

}

// ====================== Device Control Callbacks ======================

void firstLightChanged(uint8\_t brightness) {

if (brightness == 255) {

digitalWrite(RelayPin1, LOW); // LOW turns relay ON for active-low

toggleState\_1 = LOW;

} else {

digitalWrite(RelayPin1, HIGH); // HIGH turns relay OFF for active-low

toggleState\_1 = HIGH;

}

pref.putBool("Relay1", (toggleState\_1 == LOW)); // Store the logical state (true for ON)

}

void secondLightChanged(uint8\_t brightness) {

if (brightness == 255) {

digitalWrite(RelayPin2, LOW); // LOW turns relay ON

toggleState\_2 = LOW;

} else {

digitalWrite(RelayPin2, HIGH); // HIGH turns relay OFF

toggleState\_2 = HIGH;

}

pref.putBool("Relay2", (toggleState\_2 == LOW));

}

void thirdLightChanged(uint8\_t brightness) {

if (brightness == 255) {

digitalWrite(RelayPin3, LOW); // LOW turns relay ON

toggleState\_3 = LOW;

} else {

digitalWrite(RelayPin3, HIGH); // HIGH turns relay OFF

toggleState\_3 = HIGH;

}

pref.putBool("Relay3", (toggleState\_3 == LOW));

}

void fourthLightChanged(uint8\_t brightness) {

if (brightness == 255) {

digitalWrite(RelayPin4, LOW); // LOW turns relay ON

toggleState\_4 = LOW;

} else {

digitalWrite(RelayPin4, HIGH); // HIGH turns relay OFF

toggleState\_4 = HIGH;

}

pref.putBool("Relay4", (toggleState\_4 == LOW));

}

void addDevices() {

espalexa.addDevice(Device\_1\_Name, firstLightChanged);

espalexa.addDevice(Device\_2\_Name, secondLightChanged);

espalexa.addDevice(Device\_3\_Name, thirdLightChanged);

espalexa.addDevice(Device\_4\_Name, fourthLightChanged);

espalexa.begin();

}

void sendFeedback(int relay, int value) {

EspalexaDevice\* d = espalexa.getDevice(relay);

d->setPercent(value);

}

// ====================== IR Remote ======================

void ir\_remote() {

if (irrecv.decode(&results)) {

switch (results.value) {

case IR\_Button\_1:

toggleState\_1 = !toggleState\_1; // Toggle the state (HIGH/LOW)

digitalWrite(RelayPin1, toggleState\_1); // Apply the new state

pref.putBool("Relay1", (toggleState\_1 == LOW)); // Save logical state

(toggleState\_1 == HIGH) ? sendFeedback(0, 0) : sendFeedback(0, 100);

break;

case IR\_Button\_2:

toggleState\_2 = !toggleState\_2;

digitalWrite(RelayPin2, toggleState\_2);

pref.putBool("Relay2", (toggleState\_2 == LOW));

(toggleState\_2 == HIGH) ? sendFeedback(1, 0) : sendFeedback(1, 100);

break;

case IR\_Button\_3:

toggleState\_3 = !toggleState\_3;

digitalWrite(RelayPin3, toggleState\_3);

pref.putBool("Relay3", (toggleState\_3 == LOW));

(toggleState\_3 == HIGH) ? sendFeedback(2, 0) : sendFeedback(2, 100);

break;

case IR\_Button\_4:

toggleState\_4 = !toggleState\_4;

digitalWrite(RelayPin4, toggleState\_4);

pref.putBool("Relay4", (toggleState\_4 == LOW));

(toggleState\_4 == HIGH) ? sendFeedback(3, 0) : sendFeedback(3, 100);

break;

case IR\_All\_Off: all\_SwitchOff(); break;

case IR\_All\_On: all\_SwitchOn(); break;

}

irrecv.resume();

}

}

// ====================== Bluetooth Control ======================

void bluetooth\_control() {

if (SerialBT.available()) {

bt\_data = SerialBT.read();

switch (bt\_data) {

case 'A': digitalWrite(RelayPin1, LOW); toggleState\_1 = LOW; sendFeedback(0, 100); pref.putBool("Relay1", true); break;

case 'a': digitalWrite(RelayPin1, HIGH); toggleState\_1 = HIGH; sendFeedback(0, 0); pref.putBool("Relay1", false); break;

case 'B': digitalWrite(RelayPin2, LOW); toggleState\_2 = LOW; sendFeedback(1, 100); pref.putBool("Relay2", true); break;

case 'b': digitalWrite(RelayPin2, HIGH); toggleState\_2 = HIGH; sendFeedback(1, 0); pref.putBool("Relay2", false); break;

case 'C': digitalWrite(RelayPin3, LOW); toggleState\_3 = LOW; sendFeedback(2, 100); pref.putBool("Relay3", true); break;

case 'c': digitalWrite(RelayPin3, HIGH); toggleState\_3 = HIGH; sendFeedback(2, 0); pref.putBool("Relay3", false); break;

case 'D': digitalWrite(RelayPin4, LOW); toggleState\_4 = LOW; sendFeedback(3, 100); pref.putBool("Relay4", true); break;

case 'd': digitalWrite(RelayPin4, HIGH); toggleState\_4 = HIGH; sendFeedback(3, 0); pref.putBool("Relay4", false); break;

case 'Z': all\_SwitchOn(); break;

case 'z': all\_SwitchOff(); break;

}

}

}

// ====================== Relay Control ======================

void all\_SwitchOn() {

toggleState\_1 = LOW; digitalWrite(RelayPin1, LOW); pref.putBool("Relay1", true); sendFeedback(0, 100);

toggleState\_2 = LOW; digitalWrite(RelayPin2, LOW); pref.putBool("Relay2", true); sendFeedback(1, 100);

toggleState\_3 = LOW; digitalWrite(RelayPin3, LOW); pref.putBool("Relay3", true); sendFeedback(2, 100);

toggleState\_4 = LOW; digitalWrite(RelayPin4, LOW); pref.putBool("Relay4", true); sendFeedback(3, 100);

}

void all\_SwitchOff() {

toggleState\_1 = HIGH; digitalWrite(RelayPin1, HIGH); pref.putBool("Relay1", false); sendFeedback(0, 0);

toggleState\_2 = HIGH; digitalWrite(RelayPin2, HIGH); pref.putBool("Relay2", false); sendFeedback(1, 0);

toggleState\_3 = HIGH; digitalWrite(RelayPin3, HIGH); pref.putBool("Relay3", false); sendFeedback(2, 0);

toggleState\_4 = HIGH; digitalWrite(RelayPin4, HIGH); pref.putBool("Relay4", false); sendFeedback(3, 0);

}

// ====================== AceButton Switch Control ======================

void handleEvent(AceButton\* button, uint8\_t eventType, uint8\_t buttonState) {

if (eventType == AceButton::kEventPressed) {

bool newState;

int relayPin;

int deviceIndex;

if (button->getPin() == SwitchPin1) {

toggleState\_1 = !toggleState\_1;

newState = (toggleState\_1 == LOW);

relayPin = RelayPin1;

deviceIndex = 0;

} else if (button->getPin() == SwitchPin2) {

toggleState\_2 = !toggleState\_2;

newState = (toggleState\_2 == LOW);

relayPin = RelayPin2;

deviceIndex = 1;

} else if (button->getPin() == SwitchPin3) {

toggleState\_3 = !toggleState\_3;

newState = (toggleState\_3 == LOW);

relayPin = RelayPin3;

deviceIndex = 2;

} else if (button->getPin() == SwitchPin4) {

toggleState\_4 = !toggleState\_4;

newState = (toggleState\_4 == LOW);

relayPin = RelayPin4;

deviceIndex = 3;

} else {

return;

}

digitalWrite(relayPin, newState ? LOW : HIGH);

pref.putBool(("Relay" + String(deviceIndex + 1)).c\_str(), newState);

sendFeedback(deviceIndex, newState ? 100 : 0);

}

}

// ====================== WiFi Failover Functions ======================

void scanWiFiNetworks() {

Serial.println("Scanning for WiFi networks...");

int n = WiFi.scanNetworks();

if (n == 0) {

Serial.println("No networks found");

} else {

Serial.printf("%d networks found:\n", n);

for (int i = 0; i < n; ++i) {

Serial.printf("%d: %s (%d dBm) %s\n", i + 1, WiFi.SSID(i).c\_str(),

WiFi.RSSI(i), (WiFi.encryptionType(i) == WIFI\_AUTH\_OPEN) ? "Open" : "Encrypted");

}

}

WiFi.scanDelete();

}

void connectWifi() {

WiFi.disconnect(true);

delay(1000);

WiFi.onEvent(WiFiEvent);

if (currentWiFi == 0) {

Serial.println("Connecting to " + String(ssid1));

WiFi.begin(ssid1, password1);

} else {

Serial.println("Connecting to " + String(ssid2));

WiFi.begin(ssid2, password2);

}

wifiRetryTimer = millis();

wifiRetryCount++;

}

void handleWiFiFailover() {

// Boot scan after 30 seconds

if (!bootScanDone && millis() - bootTime > BOOT\_SCAN\_DELAY) {

scanWiFiNetworks();

bootScanDone = true;

}

// Check if we need to retry or switch networks

if (WiFi.status() != WL\_CONNECTED &&

millis() - wifiRetryTimer > WIFI\_RETRY\_DELAY) {

if (wifiRetryCount >= MAX\_WIFI\_RETRIES) {

currentWiFi = (currentWiFi == 0) ? 1 : 0;

wifiRetryCount = 0;

Serial.println("Switching to alternate WiFi network");

}

connectWifi();

}

}

// ====================== Setup Function ======================

void setup() {

Serial.begin(115200);

bootTime = millis();

// Initialize relays as outputs and start in the OFF (HIGH) state

pinMode(RelayPin1, OUTPUT);

pinMode(RelayPin2, OUTPUT);

pinMode(RelayPin3, OUTPUT);

pinMode(RelayPin4, OUTPUT);

digitalWrite(RelayPin1, HIGH);

digitalWrite(RelayPin2, HIGH);

digitalWrite(RelayPin3, HIGH);

digitalWrite(RelayPin4, HIGH);

// Initialize switches with pull-up resistors for active-low buttons

pinMode(SwitchPin1, INPUT\_PULLUP);

pinMode(SwitchPin2, INPUT\_PULLUP);

pinMode(SwitchPin3, INPUT\_PULLUP);

pinMode(SwitchPin4, INPUT\_PULLUP);

// Configure AceButton for active-low switches

config.setFeature(ButtonConfig::kFeatureSuppressAfterClick);

config.setEventHandler(handleEvent);

// Initialize IR remote

irrecv.enableIRIn();

// Initialize preferences for state persistence

pref.begin("smart\_home", false);

toggleState\_1 = pref.getBool("Relay1", false) ? LOW : HIGH; // Read and apply last state

toggleState\_2 = pref.getBool("Relay2", false) ? LOW : HIGH;

toggleState\_3 = pref.getBool("Relay3", false) ? LOW : HIGH;

toggleState\_4 = pref.getBool("Relay4", false) ? LOW : HIGH;

digitalWrite(RelayPin1, toggleState\_1);

digitalWrite(RelayPin2, toggleState\_2);

digitalWrite(RelayPin3, toggleState\_3);

digitalWrite(RelayPin4, toggleState\_4);

// Initialize Bluetooth

SerialBT.begin("Pradip Trial");

// Connect to WiFi

pinMode(wifiLed, OUTPUT);

connectWifi(); // Start initial WiFi connection

}

// ====================== Loop Function ======================

void loop() {

handleWiFiFailover();

if (WiFi.status() == WL\_CONNECTED) {

espalexa.loop();

// Update the WiFi LED status

digitalWrite(wifiLed, LOW);

} else {

// Turn off WiFi LED when not connected

digitalWrite(wifiLed, HIGH);

}

// Check for Bluetooth data

bluetooth\_control();

// Check for IR remote signals

ir\_remote();

// Check for button presses (using AceButton)

button1.check();

button2.check();

button3.check();

button4.check();

}