**Hardware Arduino Code**

#define LEFT\_BUTTON 2

#define RIGHT\_BUTTON 3

#define LEFT\_LED 9

#define RIGHT\_LED 10

#define BRIGHTNESS 128

#define PRESS\_DURATION 1000 // 1 second

#define BLINK\_INTERVAL 300 // LED blink rate

#define SCHEDULER\_INTERVAL 100 // 100 ms base scheduler

// States

bool leftActive = false;

bool rightActive = false;

bool hazardActive = false;

bool leftLedState = false;

bool rightLedState = false;

// Button timing

unsigned long leftPressStart = 0;

unsigned long rightPressStart = 0;

bool leftPressed = false;

bool rightPressed = false;

// Blinking timing

unsigned long lastBlinkTime = 0;

// Scheduler

unsigned long lastSchedulerTime = 0;

void setup() {

pinMode(LEFT\_BUTTON, INPUT); // Use INPUT\_PULLUP if using GND

pinMode(RIGHT\_BUTTON, INPUT);

pinMode(LEFT\_LED, OUTPUT);

pinMode(RIGHT\_LED, OUTPUT);

Serial.begin(9600);

Serial.println("System Init");

}

void loop() {

unsigned long now = millis();

// ===== Scheduler: Every 100 ms =====

if (now - lastSchedulerTime >= SCHEDULER\_INTERVAL) {

lastSchedulerTime = now;

Base\_Task(); // GPIO + UART + PWM service

Application\_Task(); // Logic based on button state

}

// ===== Blinking Logic: Every 300 ms =====

if (now - lastBlinkTime >= BLINK\_INTERVAL) {

lastBlinkTime = now;

if (leftActive || hazardActive) {

leftLedState = !leftLedState;

analogWrite(LEFT\_LED, leftLedState ? BRIGHTNESS : 0);

}

if (rightActive || hazardActive) {

rightLedState = !rightLedState;

analogWrite(RIGHT\_LED, rightLedState ? BRIGHTNESS : 0);

}

}

}

// ========== BASE LAYER ==========

void Base\_Task() {

unsigned long now = millis();

// === LEFT BUTTON ===

if (digitalRead(LEFT\_BUTTON) == HIGH) {

Serial.print("left");

if (!leftPressed) {

leftPressed = true;

leftPressStart = now;

}

} else {

leftPressed = false;

}

// === RIGHT BUTTON ===

if (digitalRead(RIGHT\_BUTTON) == HIGH) {

if (!rightPressed) {

rightPressed = true;

rightPressStart = now;

}

} else {

rightPressed = false;

}

// UART Reporting

Serial.print("LEFT:");

Serial.print(leftPressed ? "PRESSED, " : "RELEASED, ");

Serial.print(leftActive ? "ON | " : "OFF | ");

Serial.print("RIGHT:");

Serial.print(rightPressed ? "PRESSED, " : "RELEASED, ");

Serial.print(rightActive ? "ON | " : "OFF | ");

Serial.print("HAZARD:");

Serial.println(hazardActive ? "ACTIVE" : "INACTIVE");

}

// ========== APPLICATION LAYER ==========

void Application\_Task() {

unsigned long now = millis();

// === Hazard Activation ===

if (leftPressed && rightPressed &&

(now - leftPressStart >= PRESS\_DURATION) &&

(now - rightPressStart >= PRESS\_DURATION) &&

!hazardActive) {

hazardActive = true;

leftActive = false;

rightActive = false;

Serial.println("HAZARD ON");

return;

}

// === Hazard Deactivation ===

if (hazardActive) {

if ((leftPressed && (now - leftPressStart >= PRESS\_DURATION)) ||

(rightPressed && (now - rightPressStart >= PRESS\_DURATION))) {

hazardActive = false;

analogWrite(LEFT\_LED, 0);

analogWrite(RIGHT\_LED, 0);

Serial.println("HAZARD OFF");

}

return;

}

// === LEFT Indicator Logic ===

if (leftPressed && (now - leftPressStart >= PRESS\_DURATION)) {

leftPressed = false;

if (rightActive) {

rightActive = false;

analogWrite(RIGHT\_LED, 0);

Serial.println("Right OFF");

}

leftActive = !leftActive;

if (!leftActive) analogWrite(LEFT\_LED, 0);

Serial.println(leftActive ? "Left ON" : "Left OFF");

}

// === RIGHT Indicator Logic ===

if (rightPressed && (now - rightPressStart >= PRESS\_DURATION))

{

rightPressed = false;

if (leftActive) {

leftActive = false;

analogWrite(LEFT\_LED, 0);

Serial.println("Left OFF");

}

rightActive = !rightActive;

if (!rightActive) analogWrite(RIGHT\_LED, 0);

Serial.println(rightActive ? "Right ON" : "Right OFF");

}

}