#include <FastLED.h>

// === Cấu hình LED WS2812B ===

#define LED\_PIN     17        // Chân kết nối dải LED

#define NUM\_LEDS    60        // Số lượng LED trong dải

#define LED\_TYPE    WS2812B   // Loại LED

#define COLOR\_ORDER GRB       // Thứ tự màu sắc

CRGB leds[NUM\_LEDS];

// === Cấu hình âm thanh ===

#define AUDIO\_PIN   15        // Chân nhận tín hiệu âm thanh

#define NUM\_MODES   9         // Số hiệu ứng LED

int audioThreshold = 200;     // Ngưỡng tín hiệu để kích hoạt hiệu ứng

int currentMode = 0;          // Chế độ hiệu ứng hiện tại

int audioValue = 0;           // Biến lưu giá trị âm thanh

void setup() {

  Serial.begin(115200);

  FastLED.addLeds<LED\_TYPE, LED\_PIN, COLOR\_ORDER>(leds, NUM\_LEDS);

  FastLED.clear();

  FastLED.show();

}

void loop() {

  audioValue = analogRead(AUDIO\_PIN);

  Serial.println(audioValue);

  if (audioValue > audioThreshold) {

    currentMode = (currentMode + 1) % NUM\_MODES;

    runEffect(currentMode);

  } else {

    FastLED.clear();

    FastLED.show();

  }

  delay(20); // Giảm tải CPU

}

// Khai báo các hàm hiệu ứng

void cylonEffect();

void fadeEffect();

void expandEffect();

void shiftLeftRight();

void blinkEffect();

void rainbowChase();

void waveEffect();

void meteorEffect();

void explosionEffect();

// === Hàm chạy hiệu ứng ===

void runEffect(int mode) {

  switch (mode) {

    case 0: cylonEffect(); break;

    case 1: fadeEffect(); break;

    case 2: expandEffect(); break;

    case 3: shiftLeftRight(); break;

    case 4: blinkEffect(); break;

    case 5: rainbowChase(); break;

    case 6: waveEffect(); break;

    case 7: meteorEffect(); break;

    case 8: explosionEffect(); break;

  }

}

// === Định nghĩa các hiệu ứng ===

void cylonEffect() {

  for (int i = 0; i < NUM\_LEDS; i++) {

    FastLED.clear();

    leds[i] = CRGB::Red;

    FastLED.show();

    delay(20);

  }

  for (int i = NUM\_LEDS - 1; i >= 0; i--) {

    FastLED.clear();

    leds[i] = CRGB::Red;

    FastLED.show();

    delay(20);

  }

}

void fadeEffect() {

  for (int brightness = 0; brightness <= 255; brightness += 5) {

    fill\_solid(leds, NUM\_LEDS, CHSV(160, 255, brightness));

    FastLED.show();

    delay(20);

  }

  for (int brightness = 255; brightness >= 0; brightness -= 5) {

    fill\_solid(leds, NUM\_LEDS, CHSV(160, 255, brightness));

    FastLED.show();

    delay(20);

  }

}

void expandEffect() {

  FastLED.clear();

  for (int i = 0; i < NUM\_LEDS / 2; i++) {

    leds[i] = CHSV(100, 255, 255);

    leds[NUM\_LEDS - 1 - i] = CHSV(100, 255, 255);

    FastLED.show();

    delay(30);

  }

  FastLED.clear();

  FastLED.show();

}

void shiftLeftRight() {

  for (int i = 0; i < NUM\_LEDS; i++) {

    FastLED.clear();

    leds[i] = CHSV(0, 255, 255);

    if (i > 0) leds[i - 1] = CHSV(0, 255, 128);

    FastLED.show();

    delay(30);

  }

  for (int i = NUM\_LEDS - 1; i >= 0; i--) {

    FastLED.clear();

    leds[i] = CHSV(0, 255, 255);

    if (i < NUM\_LEDS - 1) leds[i + 1] = CHSV(0, 255, 128);

    FastLED.show();

    delay(30);

  }

}

void blinkEffect() {

  for (int i = 0; i < 10; i++) {

    fill\_solid(leds, NUM\_LEDS, CHSV(random(0, 255), 255, 255));

    FastLED.show();

    delay(100);

    FastLED.clear();

    FastLED.show();

    delay(100);

  }

}

void rainbowChase() {

  static uint8\_t hue = 0;

  for (int i = 0; i < NUM\_LEDS; i++) {

    leds[i] = CHSV(hue + (i \* 10), 255, 255);

  }

  FastLED.show();

  hue += 5;

  delay(50);

}

void waveEffect() {

  static uint8\_t wavePos = 0;

  for (int i = 0; i < NUM\_LEDS; i++) {

    int brightness = sin8(wavePos + (i \* 10));

    leds[i] = CHSV(160, 255, map(brightness, 0, 255, 0, 255));

  }

  FastLED.show();

  wavePos += 4;

  delay(30);

}

void meteorEffect() {

  FastLED.clear();

  for (int i = 0; i < NUM\_LEDS; i++) {

    leds[i] = CHSV(0, 255, 255);

    FastLED.show();

    delay(30);

    leds[i].fadeToBlackBy(50);

  }

}

void explosionEffect() {

  int center = NUM\_LEDS / 2;

  for (int radius = 0; radius < center; radius++) {

    leds[center + radius] = CHSV(random(0, 255), 255, 255);

    leds[center - radius] = CHSV(random(0, 255), 255, 255);

    FastLED.show();

    delay(50);

  }

  FastLED.clear();

  FastLED.show();

}