#include <FastLED.h>

// Định nghĩa các thông số của dãy LED

#define NUM\_LEDS 24

#define DATA\_PIN 6

CRGB leds[NUM\_LEDS];

// Định nghĩa nút nhấn

#define BUTTON\_PIN 5//chan 10 atmega

int BUT=2;//chan 32 atmega,Khong dung;

// Biến để lưu trữ trạng thái của nút nhấn

bool buttonState = false;

bool lastButtonState = false;

// Biến để lưu trữ độ sáng của LED (giá trị từ 0 đến 255)

uint8\_t brightness = 120; // Độ sáng mặc định

// Biến để lưu trữ hiệu ứng hiện tại

int currentEffect = 0;

void setup() {

// Cài đặt chân điều khiển dãy LED

FastLED.addLeds<WS2812B, DATA\_PIN, GRB>(leds, NUM\_LEDS);

// Cài đặt nút nhấn

pinMode(BUTTON\_PIN, INPUT);

pinMode(BUT, INPUT);//khong dung

// Đặt độ sáng ban đầu cho dãy LED

FastLED.setBrightness(brightness);

// Mặc định các LED tắt khi khởi động

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

leds[i] = CRGB::Black;

}

FastLED.show();

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

void loop() {

// Đọc trạng thái của nút nhấn

buttonState = digitalRead(BUTTON\_PIN);

// Kiểm tra nếu trạng thái của nút nhấn thay đổi từ không nhấn sang nhấn

if (buttonState == HIGH && lastButtonState == LOW) {

// Tắt các LED khi nhấn nút

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

leds[i] = CRGB::Black;

}

FastLED.show();

delay(200); // Debounce

}

// Kiểm tra nếu nút đã được nhả

if (buttonState == LOW && lastButtonState == HIGH) {

// Thay đổi hiệu ứng khi nút nhấn đã được nhả

changeEffect();

}

// Cập nhật trạng thái cuối của nút nhấn

lastButtonState = buttonState;

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Hiển thị hiệu ứng hiện tại

switch (currentEffect) {

case 0:

cauVong(); // Hiệu ứng cầu vồng

break;

case 1:

tatled();

break;

case 2:

xoayTronDoiMau(); // Hiệu ứng xoay tròn đổi màu

break;

case 3:

sangDon(CRGB::Red); // Hiệu ứng sáng đơn màu đỏ

break;

case 4:

sangDon(CRGB::Green); // Hiệu ứng sáng đơn màu xanh lá

break;

case 5:

sangDon(CRGB::Blue); // Hiệu ứng sáng đơn màu xanh dương

break;

case 6:

sangDon(CRGB(255, 0, 200)); // Hiệu ứng sáng đơn màu tím

break;

case 7:

lenxuong\_doimau();

break;

case 8:

ledTho();// Hiệu ứng led thở

break;

case 9:

ledDuoi(); // Hiệu ứng 6 LED đuổi

break;

case 10:

cauVongTatCa(); // Hiệu ứng cầu vồng tất cả LED

break;

case 11:

nhapNhay8Mau(); // Hiệu ứng nhấp nháy 8 màu

break;

case 12:

quetMau();

break;

case 13:

ledDuoiRED();

break;

}

FastLED.show();

delay(20); // Thời gian trễ cho mỗi lần lặp để giảm tốc độ hiển thị hiệu ứng

}

void changeEffect() {

// Thay đổi hiệu ứng

currentEffect = (currentEffect + 1) % 14;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//HIEU UNG 1

void cauVong() {

// Hiệu ứng cầu vồng

fill\_rainbow(leds, NUM\_LEDS, millis() / 20);

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//TAT LED

void tatled()

{

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

leds[i] = CRGB::Black;

}

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//HIEU UNG 3

void xoayTronDoiMau() {

// Hiệu ứng xoay tròn đổi màu

uint8\_t wheelPos = (millis() / 2) % 255;

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

leds[i] = CHSV((wheelPos + i \* 255 / NUM\_LEDS) & 255,255, 255);

}

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//HIEU UNG 4,5,6,7

void sangDon(CRGB color) {

// Hiệu ứng sáng đơn

fill\_solid(leds, NUM\_LEDS, color);

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//HIEU UNG 8

void lenxuong\_doimau() {

CRGB colors[] = {

CRGB::Red, // Màu đỏ

CRGB::Blue, // Màu xanh

CRGB::Yellow, // Màu vàng

CRGB::Green, // Màu xanh lá

CRGB::Purple, // Màu tím

CRGB::Cyan, // Màu xanh da trời

CRGB::Orange, // Màu cam

CRGB(255, 105, 180) // Màu hồng

};

int half\_leds = NUM\_LEDS / 2;

// Biến lưu trữ trạng thái nút nhấn

bool buttonPressed = false;

for (int cycle = 0; cycle < 8; cycle++) {

// Chọn màu sắc dựa trên vòng lặp

CRGB color = colors[cycle % 8];

// Hiển thị từ đầu đến giữa và từ cuối đến giữa

for (int i = 0; i < half\_leds; i++) {

fadeToBlackBy(leds, NUM\_LEDS, 20); // Giảm độ sáng một lượng nhất định

if (i < half\_leds ) {

leds[i] = color; // Sử dụng màu được chọn

leds[NUM\_LEDS - 1 - i] = color; // Sử dụng màu được chọn

} else {

leds[i] = colors[(cycle + 4) % 8]; // Sử dụng màu được chọn khác

leds[NUM\_LEDS - 1 - i] = colors[(cycle + 4) % 8]; // Sử dụng màu được chọn khác

}

FastLED.show();

FastLED.delay(40);

// Kiểm tra nút nhấn sau mỗi lần hiển thị

buttonState = digitalRead(BUTTON\_PIN);

if (buttonState == HIGH) {

buttonPressed = true; // Nút đã được nhấn

break; // Thoát vòng lặp hiển thị

}

}

// Kiểm tra nếu nút nhấn đã được nhấn và nhả

if (buttonPressed) {

changeEffect(); // Thay đổi hiệu ứng nếu nút được nhấn

delay(200);

return; // Thoát khỏi hàm để áp dụng hiệu ứng mới

}

// Đảo chiều từ giữa ra 2 đầu

for (int i = half\_leds - 1; i >= 0; i--) {

fadeToBlackBy(leds, NUM\_LEDS, 10); // Giảm độ sáng một lượng nhất định

if (i < half\_leds) {

leds[i] = color; // Sử dụng màu được chọn

leds[NUM\_LEDS - 1 - i] = color; // Sử dụng màu được chọn

} else {

leds[i] = colors[(cycle + 4) % 8]; // Sử dụng màu được chọn khác

leds[NUM\_LEDS - 1 - i] = colors[(cycle + 4) % 8]; // Sử dụng màu được chọn khác

}

FastLED.show();

FastLED.delay(40);

// Kiểm tra nút nhấn sau mỗi lần hiển thị

buttonState = digitalRead(BUTTON\_PIN);

if (buttonState == HIGH) {

buttonPressed = true; // Nút đã được nhấn

break; // Thoát vòng lặp hiển thị

}

}

// Kiểm tra nếu nút nhấn đã được nhấn và nhả

if (buttonPressed) {

changeEffect();

// Thay đổi hiệu ứng nếu nút được nhấn

delay(200);

return; // Thoát khỏi hàm để áp dụng hiệu ứng mới

}

}

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//HIEU UNG 9

void ledTho() {

// Hiệu ứng led thở với 3 màu: xanh lá, đỏ, xanh dương

static uint8\_t brightness = 0;

static int phase = 0;

static int colorIndex = 0;

// Định nghĩa các màu

CRGB colors[] = {

CRGB(0, 255, 0), // Xanh lá

CRGB(255, 0, 0), // Đỏ

CRGB(0, 0, 255),// Xanh dương

CRGB(255, 255, 0),//Vang

CRGB(255,0,255),//Hong

CRGB(0, 255,255)//Xanh da troi

};

int numColors = sizeof(colors) / sizeof(colors[0]);

if (phase == 0) {

brightness += 5;

if (brightness >= 255) {

brightness = 255;

phase = 1;

}

} else if (phase == 1) {

brightness -= 5;

if (brightness <= 0) {

brightness = 0;

phase = 2;

}

} else {

phase = 0;

colorIndex = (colorIndex + 1) % numColors; // Chuyển sang màu tiếp theo

delay(500); // Đợi 1 giây trước khi bắt đầu lại vòng lặp

return;

}

// Thiết lập độ sáng cho toàn bộ dãy LED

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

leds[i] = colors[colorIndex];

leds[i].fadeLightBy(255 - brightness); // Điều chỉnh độ sáng

}

FastLED.show();

delay(50);

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//HIEU UNG 10

void ledDuoi() {

// Hiệu ứng 12 LED đuổi TRON

static uint8\_t startIndex = 0;

startIndex = (startIndex + 1)%24; /\* Tốc độ di chuyển \*/

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

leds[i] = CRGB::Black;

}

leds[(startIndex +0)% 24] = CRGB(0,0,255); // BLUE(0,0,255)

leds[(startIndex + 1) % 24] =CRGB(0,255,255); // DATROI

leds[(startIndex + 2) % 24] = CRGB(0,255,0);//XANH LA

leds[(startIndex + 3) % 24] = CRGB(255,255,0);//VANG

leds[(startIndex + 4) % 24] = CRGB(255,165,0);//CAM(255,165,0)

leds[(startIndex + 5) % 24] = CRGB(255,0,0);//DO(255,0,0)

leds[(startIndex +12)% 24] = CRGB(0,0,255); // BLUE(0,0,255)

leds[(startIndex + 13) % 24] =CRGB(0,255,255); // DATROI

leds[(startIndex + 14) % 24] = CRGB(0,255,0);//XANH LA

leds[(startIndex + 15) % 24] = CRGB(255,255,0);//VANG

leds[(startIndex + 16) % 24] = CRGB(255,165,0);//CAM(255,165,0)

leds[(startIndex + 17) % 24] = CRGB(255,0,0);//DO(255,0,0)

delay(20);

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//HIEU UNG 11

void cauVongTatCa() {

// Hiệu ứng cầu vồng đồng màu

uint8\_t hue = (millis() /35) % 255;

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

leds[i] = CHSV(hue, 255, 255);

}

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//HIEU UNG 12

void nhapNhay8Mau() {

// Hiệu ứng nhấp nháy 8 màu

static bool isOn = false;

isOn = !isOn;

if (isOn) {

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

switch (i % 8) {

case 0:

leds[i] = CRGB::Red;

break;

case 1:

leds[i] = CRGB::Green;

break;

case 2:

leds[i] = CRGB::Blue;

break;

case 3:

leds[i] = CRGB::Yellow;

break;

case 4:

leds[i] = CRGB::Cyan;

break;

case 5:

leds[i] = CRGB::Magenta;

break;

case 6:

leds[i] = CRGB::Orange;

break;

case 7:

leds[i] = CRGB::Purple;

break;

}

delay(8);//toc do nhay sang

}

} else {

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

leds[i] = CRGB::Black;

delay(8);

}

}

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//HIEU UNG 13

void quetMau() {

// Biến lưu trữ trạng thái nút nhấn

bool buttonPressed = false;

// Định nghĩa các màu

CRGB colors[] = {

CRGB(255, 0, 0), // Đỏ

CRGB(0, 0, 255), // Blue

CRGB(255, 255, 0), // Vàng

CRGB(0, 255, 0), // Xanh lá

CRGB(0, 255, 255), // Xanh da trời

CRGB(255, 165, 0), // Cam

CRGB(128, 0, 128) // Tím

};

int numColors = sizeof(colors) / sizeof(colors[0]);

static int colorIndex = 0;

static bool direction = true; // true: từ 1 đến 24, false: từ 24 về 1

if (direction) {

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

leds[i] = colors[colorIndex];

FastLED.show();

delay(50); // Thời gian trễ giữa mỗi LED

// Kiểm tra nút nhấn sau mỗi lần hiển thị

buttonState = digitalRead(BUTTON\_PIN);

if (buttonState == HIGH) {

buttonPressed = true; // Nút đã được nhấn

break; // Thoát vòng lặp hiển thị

}

}

// Kiểm tra nếu nút nhấn đã được nhấn và nhả

if (buttonPressed) {

changeEffect();

// Thay đổi hiệu ứng nếu nút được nhấn

delay(200);

return; // Thoát khỏi hàm để áp dụng hiệu ứng mới

}

} else {

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

leds[i] = colors[colorIndex];

FastLED.show();

delay(50); // Thời gian trễ giữa mỗi LED

// Kiểm tra nút nhấn sau mỗi lần hiển thị

buttonState = digitalRead(BUTTON\_PIN);

if (buttonState == HIGH) {

buttonPressed = true; // Nút đã được nhấn

break; // Thoát vòng lặp hiển thị

}

}

}

// Kiểm tra nếu nút nhấn đã được nhấn và nhả

if (buttonPressed) {

changeEffect();

// Thay đổi hiệu ứng nếu nút được nhấn

delay(200);

return; // Thoát khỏi hàm để áp dụng hiệu ứng mới

}

delay(150); // Thời gian chờ giữa các lần quét

//direction = !direction; // Đổi chiều

colorIndex = (colorIndex + 1) % numColors; // Chuyển sang màu tiếp theo

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////

//HIEU UNG 14

void ledDuoiRED() {

// Hieu ung 8 led cach nhau 2 led duoi nhau

static uint8\_t startIndex = 0;

startIndex = (startIndex + 1)%24; /\* Tốc độ di chuyển \*/

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

leds[i] = CRGB::Black;

}

leds[(startIndex + 0) % 24] = CRGB(0,0,255);//

leds[(startIndex + 3) % 24] = CRGB(0,0,255);//

leds[(startIndex + 6) % 24] = CRGB(0,0,255);//

leds[(startIndex + 9) % 24] = CRGB(0,0,255);//

leds[(startIndex + 12) % 24] = CRGB(0,0,255);//

leds[(startIndex + 15) % 24] = CRGB(0,0,255);//

leds[(startIndex + 18) % 24] = CRGB(0,0,255);//

leds[(startIndex + 21) % 24] = CRGB(0,0,255);//

delay(30);

}