单LED灯实现呼吸灯亮度变化

#include <stdio.h>

#include <unistd.h>

#include "ohos\_init.h"

#include "cmsis\_os2.h"

#include "iot\_gpio.h"

#include "iot\_watchdog.h"

#include "iot\_pwm.h"

#include "hi\_io.h"

#define STACK\_SIZE (1024)

#define IOT\_PWM\_DUTY\_MAX  100

#define PWM\_FREQ\_DIVISION 64000

#define RED\_LED\_PIN\_NAME 3

#define RED\_LED\_PIN\_FUNCTION HI\_IO\_FUNC\_GPIO\_3\_PWM5\_OUT

#define RED\_PWM\_PORT\_NUM 5

static void SetRedLedBrightness(int dutyCycle)

{

    IoTPwmStart(RED\_PWM\_PORT\_NUM, dutyCycle, PWM\_FREQ\_DIVISION);

}

static void PWMLedDemoTask(void)

{

    hi\_io\_set\_func(RED\_LED\_PIN\_NAME, RED\_LED\_PIN\_FUNCTION);

    IoTPwmInit(RED\_PWM\_PORT\_NUM);

    int dutyCycle = 0;

    int increment = 1;

    while (1) {

        while (dutyCycle <= IOT\_PWM\_DUTY\_MAX) {

            SetRedLedBrightness(dutyCycle);

            dutyCycle += increment;

            osDelay(5); // 每10毫秒调整一次占空比

        }

        while (dutyCycle >= 0) {

            SetRedLedBrightness(dutyCycle);

            dutyCycle -= increment;

            osDelay(5); // 每10毫秒调整一次占空比

        }

    }

}

static void PWMLedDemo(void)

{

    osThreadAttr\_t attr;

    IoTGpioInit(RED\_LED\_PIN\_NAME);

    attr.name = "PWMLedDemoTask";

    attr.attr\_bits = 0U;

    attr.cb\_mem = NULL;

    attr.cb\_size = 0U;

    attr.stack\_mem = NULL;

    attr.stack\_size = STACK\_SIZE;

    attr.priority = osPriorityNormal;

    if (osThreadNew((osThreadFunc\_t)PWMLedDemoTask, NULL, &attr) == NULL) {

        printf("[LedExample] Failed to create PWMLedDemoTask!\n");

    }

}

APP\_FEATURE\_INIT(PWMLedDemo);

RGB灯实现呼吸灯色彩变化

#include <stdio.h>

#include <unistd.h>

#include "ohos\_init.h"

#include "cmsis\_os2.h"

#include "iot\_gpio.h"

#include "iot\_watchdog.h"

#include "iot\_pwm.h"

#include "hi\_io.h"

#define STACK\_SIZE (1024)

#define IOT\_PWM\_DUTY\_MAX  100

#define PWM\_FREQ\_DIVISION 64000

#define RED\_LED\_PIN\_NAME 3

#define GREEN\_LED\_PIN\_NAME 4

#define BLUE\_LED\_PIN\_NAME 5

#define RED\_LED\_PIN\_FUNCTION HI\_IO\_FUNC\_GPIO\_3\_PWM5\_OUT

#define GREEN\_LED\_PIN\_FUNCTION HI\_IO\_FUNC\_GPIO\_4\_PWM1\_OUT

#define BLUE\_LED\_PIN\_FUNCTION HI\_IO\_FUNC\_GPIO\_5\_PWM2\_OUT

#define RED\_PWM\_PORT\_NUM 5

#define GREEN\_PWM\_PORT\_NUM 1

#define BLUE\_PWM\_PORT\_NUM 2

void SetRGBLedBrightness(int redDutyCycle, int greenDutyCycle, int blueDutyCycle)

{

    if (redDutyCycle < 0) redDutyCycle = 0;

    if (redDutyCycle > IOT\_PWM\_DUTY\_MAX) redDutyCycle = IOT\_PWM\_DUTY\_MAX;

    if (greenDutyCycle < 0) greenDutyCycle = 0;

    if (greenDutyCycle > IOT\_PWM\_DUTY\_MAX) greenDutyCycle = IOT\_PWM\_DUTY\_MAX;

    if (blueDutyCycle < 0) blueDutyCycle = 0;

    if (blueDutyCycle > IOT\_PWM\_DUTY\_MAX) blueDutyCycle = IOT\_PWM\_DUTY\_MAX;

    IoTPwmStart(RED\_PWM\_PORT\_NUM, redDutyCycle, PWM\_FREQ\_DIVISION);

    IoTPwmStart(GREEN\_PWM\_PORT\_NUM, greenDutyCycle, PWM\_FREQ\_DIVISION);

    IoTPwmStart(BLUE\_PWM\_PORT\_NUM, blueDutyCycle, PWM\_FREQ\_DIVISION);

}

static void PWMLedDemoTask(void)

{

    hi\_io\_set\_func(RED\_LED\_PIN\_NAME, RED\_LED\_PIN\_FUNCTION);

    hi\_io\_set\_func(GREEN\_LED\_PIN\_NAME, GREEN\_LED\_PIN\_FUNCTION);

    hi\_io\_set\_func(BLUE\_LED\_PIN\_NAME, BLUE\_LED\_PIN\_FUNCTION);

    IoTPwmInit(RED\_PWM\_PORT\_NUM);

    IoTPwmInit(GREEN\_PWM\_PORT\_NUM);

    IoTPwmInit(BLUE\_PWM\_PORT\_NUM);

    int dutyCycle = 0;

    int increment = 1;

    while (1) {

    for (int i = 0; i < 7; i++) { // 循环七次，每次设置不同颜色

        int redDutyCycle = 0;

        int greenDutyCycle = 0;

        int blueDutyCycle = 0;

        switch (i) {

            case 0: // 红色

                redDutyCycle = dutyCycle;

                break;

            case 1: // 红绿

                redDutyCycle = dutyCycle;

                greenDutyCycle = dutyCycle;

                break;

            case 2: // 绿色

                greenDutyCycle = dutyCycle;

                break;

            case 3: // 绿蓝

                greenDutyCycle = dutyCycle;

                blueDutyCycle = dutyCycle;

                break;

            case 4: // 蓝色

                blueDutyCycle = dutyCycle;

                break;

            case 5: // 红蓝

                redDutyCycle = dutyCycle;

                blueDutyCycle = dutyCycle;

                break;

            case 6: // 红绿蓝

                redDutyCycle = dutyCycle;

                greenDutyCycle = dutyCycle;

                blueDutyCycle = dutyCycle;

                break;

            default:

                break;

        }

        while (dutyCycle <= IOT\_PWM\_DUTY\_MAX) {

            SetRGBLedBrightness(redDutyCycle, greenDutyCycle, blueDutyCycle);

            dutyCycle += increment;

            osDelay(2); // 每2毫秒调整一次占空比

            switch (i) {

                case 0: // 红色

                    redDutyCycle = dutyCycle;

                    break;

                case 1: // 红绿

                    redDutyCycle = dutyCycle;

                    greenDutyCycle = dutyCycle;

                    break;

                case 2: // 绿色

                    greenDutyCycle = dutyCycle;

                    break;

                case 3: // 绿蓝

                    greenDutyCycle = dutyCycle;

                    blueDutyCycle = dutyCycle;

                    break;

                case 4: // 蓝色

                    blueDutyCycle = dutyCycle;

                    break;

                case 5: // 红蓝

                    redDutyCycle = dutyCycle;

                    blueDutyCycle = dutyCycle;

                    break;

                case 6: // 红绿蓝

                    redDutyCycle = dutyCycle;

                    greenDutyCycle = dutyCycle;

                    blueDutyCycle = dutyCycle;

                    break;

                default:

                    break;

            }

        }

        while (dutyCycle >= 0) {

            SetRGBLedBrightness(redDutyCycle, greenDutyCycle, blueDutyCycle);

            dutyCycle -= increment;

            osDelay(2); // 每2毫秒调整一次占空比

            switch (i) {

                case 0: // 红色

                    redDutyCycle = dutyCycle;

                    break;

                case 1: // 红绿

                    redDutyCycle = dutyCycle;

                    greenDutyCycle = dutyCycle;

                    break;

                case 2: // 绿色

                    greenDutyCycle = dutyCycle;

                    break;

                case 3: // 绿蓝

                    greenDutyCycle = dutyCycle;

                    blueDutyCycle = dutyCycle;

                    break;

                case 4: // 蓝色

                    blueDutyCycle = dutyCycle;

                    break;

                case 5: // 红蓝

                    redDutyCycle = dutyCycle;

                    blueDutyCycle = dutyCycle;

                    break;

                case 6: // 红绿蓝

                    redDutyCycle = dutyCycle;

                    greenDutyCycle = dutyCycle;

                    blueDutyCycle = dutyCycle;

                    break;

                default:

                    break;

            }

        }

    }

}

}

static void PWMLedDemo(void)

{

    osThreadAttr\_t attr;

    IoTGpioInit(RED\_LED\_PIN\_NAME);

    IoTGpioInit(GREEN\_LED\_PIN\_NAME);

    IoTGpioInit(BLUE\_LED\_PIN\_NAME);

    attr.name = "PWMLedDemoTask";

    attr.attr\_bits = 0U;

    attr.cb\_mem = NULL;

    attr.cb\_size = 0U;

    attr.stack\_mem = NULL;

    attr.stack\_size = STACK\_SIZE;

    attr.priority = osPriorityNormal;

    if (osThreadNew((osThreadFunc\_t)PWMLedDemoTask, NULL, &attr) == NULL) {

        printf("[LedExample] Failed to create PWMLedDemoTask!\n");

    }

}

APP\_FEATURE\_INIT(PWMLedDemo);

RGB灯实现呼吸灯色彩变化+触碰按钮调节光效

#include <stdio.h>

#include <unistd.h>

#include "ohos\_init.h"

#include "cmsis\_os2.h"

#include "iot\_gpio.h"

#include "iot\_watchdog.h"

#include "iot\_pwm.h"

#include "hi\_io.h"

#define STACK\_SIZE (1024)

#define IOT\_PWM\_DUTY\_MAX  100

#define PWM\_FREQ\_DIVISION 64000

#define RED\_LED\_PIN\_NAME 3

#define GREEN\_LED\_PIN\_NAME 4

#define BLUE\_LED\_PIN\_NAME 5

#define RED\_LED\_PIN\_FUNCTION HI\_IO\_FUNC\_GPIO\_3\_PWM5\_OUT

#define GREEN\_LED\_PIN\_FUNCTION HI\_IO\_FUNC\_GPIO\_4\_PWM1\_OUT

#define BLUE\_LED\_PIN\_FUNCTION HI\_IO\_FUNC\_GPIO\_5\_PWM2\_OUT

#define RED\_PWM\_PORT\_NUM 5

#define GREEN\_PWM\_PORT\_NUM 1

#define BLUE\_PWM\_PORT\_NUM 2

void SetRGBLedBrightness(int redDutyCycle, int greenDutyCycle, int blueDutyCycle)

{

    if (redDutyCycle < 0) redDutyCycle = 0;

    if (redDutyCycle > IOT\_PWM\_DUTY\_MAX) redDutyCycle = IOT\_PWM\_DUTY\_MAX;

    if (greenDutyCycle < 0) greenDutyCycle = 0;

    if (greenDutyCycle > IOT\_PWM\_DUTY\_MAX) greenDutyCycle = IOT\_PWM\_DUTY\_MAX;

    if (blueDutyCycle < 0) blueDutyCycle = 0;

    if (blueDutyCycle > IOT\_PWM\_DUTY\_MAX) blueDutyCycle = IOT\_PWM\_DUTY\_MAX;

    IoTPwmStart(RED\_PWM\_PORT\_NUM, redDutyCycle, PWM\_FREQ\_DIVISION);

    IoTPwmStart(GREEN\_PWM\_PORT\_NUM, greenDutyCycle, PWM\_FREQ\_DIVISION);

    IoTPwmStart(BLUE\_PWM\_PORT\_NUM, blueDutyCycle, PWM\_FREQ\_DIVISION);

}

static void PWMLedDemoTask(void)

{

    hi\_io\_set\_func(RED\_LED\_PIN\_NAME, RED\_LED\_PIN\_FUNCTION);

    hi\_io\_set\_func(GREEN\_LED\_PIN\_NAME, GREEN\_LED\_PIN\_FUNCTION);

    hi\_io\_set\_func(BLUE\_LED\_PIN\_NAME, BLUE\_LED\_PIN\_FUNCTION);

    IoTPwmInit(RED\_PWM\_PORT\_NUM);

    IoTPwmInit(GREEN\_PWM\_PORT\_NUM);

    IoTPwmInit(BLUE\_PWM\_PORT\_NUM);

    int dutyCycle = 0;

    int increment = 1;

    while (1) {

    for (int i = 0; i < 7; i++) { // 循环七次，每次设置不同颜色

        int redDutyCycle = 0;

        int greenDutyCycle = 0;

        int blueDutyCycle = 0;

        switch (i) {

            case 0: // 红色

                redDutyCycle = dutyCycle;

                break;

            case 1: // 红绿

                redDutyCycle = dutyCycle;

                greenDutyCycle = dutyCycle;

                break;

            case 2: // 绿色

                greenDutyCycle = dutyCycle;

                break;

            case 3: // 绿蓝

                greenDutyCycle = dutyCycle;

                blueDutyCycle = dutyCycle;

                break;

            case 4: // 蓝色

                blueDutyCycle = dutyCycle;

                break;

            case 5: // 红蓝

                redDutyCycle = dutyCycle;

                blueDutyCycle = dutyCycle;

                break;

            case 6: // 红绿蓝

                redDutyCycle = dutyCycle;

                greenDutyCycle = dutyCycle;

                blueDutyCycle = dutyCycle;

                break;

            default:

                break;

        }

        while (dutyCycle <= IOT\_PWM\_DUTY\_MAX) {

            SetRGBLedBrightness(redDutyCycle, greenDutyCycle, blueDutyCycle);

            dutyCycle += increment;

            osDelay(2); // 每2毫秒调整一次占空比

            switch (i) {

                case 0: // 红色

                    redDutyCycle = dutyCycle;

                    break;

                case 1: // 红绿

                    redDutyCycle = dutyCycle;

                    greenDutyCycle = dutyCycle;

                    break;

                case 2: // 绿色

                    greenDutyCycle = dutyCycle;

                    break;

                case 3: // 绿蓝

                    greenDutyCycle = dutyCycle;

                    blueDutyCycle = dutyCycle;

                    break;

                case 4: // 蓝色

                    blueDutyCycle = dutyCycle;

                    break;

                case 5: // 红蓝

                    redDutyCycle = dutyCycle;

                    blueDutyCycle = dutyCycle;

                    break;

                case 6: // 红绿蓝

                    redDutyCycle = dutyCycle;

                    greenDutyCycle = dutyCycle;

                    blueDutyCycle = dutyCycle;

                    break;

                default:

                    break;

            }

        }

        while (dutyCycle >= 0) {

            SetRGBLedBrightness(redDutyCycle, greenDutyCycle, blueDutyCycle);

            dutyCycle -= increment;

            osDelay(2); // 每2毫秒调整一次占空比

            switch (i) {

                case 0: // 红色

                    redDutyCycle = dutyCycle;

                    break;

                case 1: // 红绿

                    redDutyCycle = dutyCycle;

                    greenDutyCycle = dutyCycle;

                    break;

                case 2: // 绿色

                    greenDutyCycle = dutyCycle;

                    break;

                case 3: // 绿蓝

                    greenDutyCycle = dutyCycle;

                    blueDutyCycle = dutyCycle;

                    break;

                case 4: // 蓝色

                    blueDutyCycle = dutyCycle;

                    break;

                case 5: // 红蓝

                    redDutyCycle = dutyCycle;

                    blueDutyCycle = dutyCycle;

                    break;

                case 6: // 红绿蓝

                    redDutyCycle = dutyCycle;

                    greenDutyCycle = dutyCycle;

                    blueDutyCycle = dutyCycle;

                    break;

                default:

                    break;

            }

        }

    }

}

}

static void PWMLedDemo(void)

{

    osThreadAttr\_t attr;

    IoTGpioInit(RED\_LED\_PIN\_NAME);

    IoTGpioInit(GREEN\_LED\_PIN\_NAME);

    IoTGpioInit(BLUE\_LED\_PIN\_NAME);

    attr.name = "PWMLedDemoTask";

    attr.attr\_bits = 0U;

    attr.cb\_mem = NULL;

    attr.cb\_size = 0U;

    attr.stack\_mem = NULL;

    attr.stack\_size = STACK\_SIZE;

    attr.priority = osPriorityNormal;

    if (osThreadNew((osThreadFunc\_t)PWMLedDemoTask, NULL, &attr) == NULL) {

        printf("[LedExample] Failed to create PWMLedDemoTask!\n");

    }

}

APP\_FEATURE\_INIT(PWMLedDemo);