### 项目代码说明

项目代码分为arduino代码和树莓派代码分别为智能厕所、智能门锁。

### Arduino代码

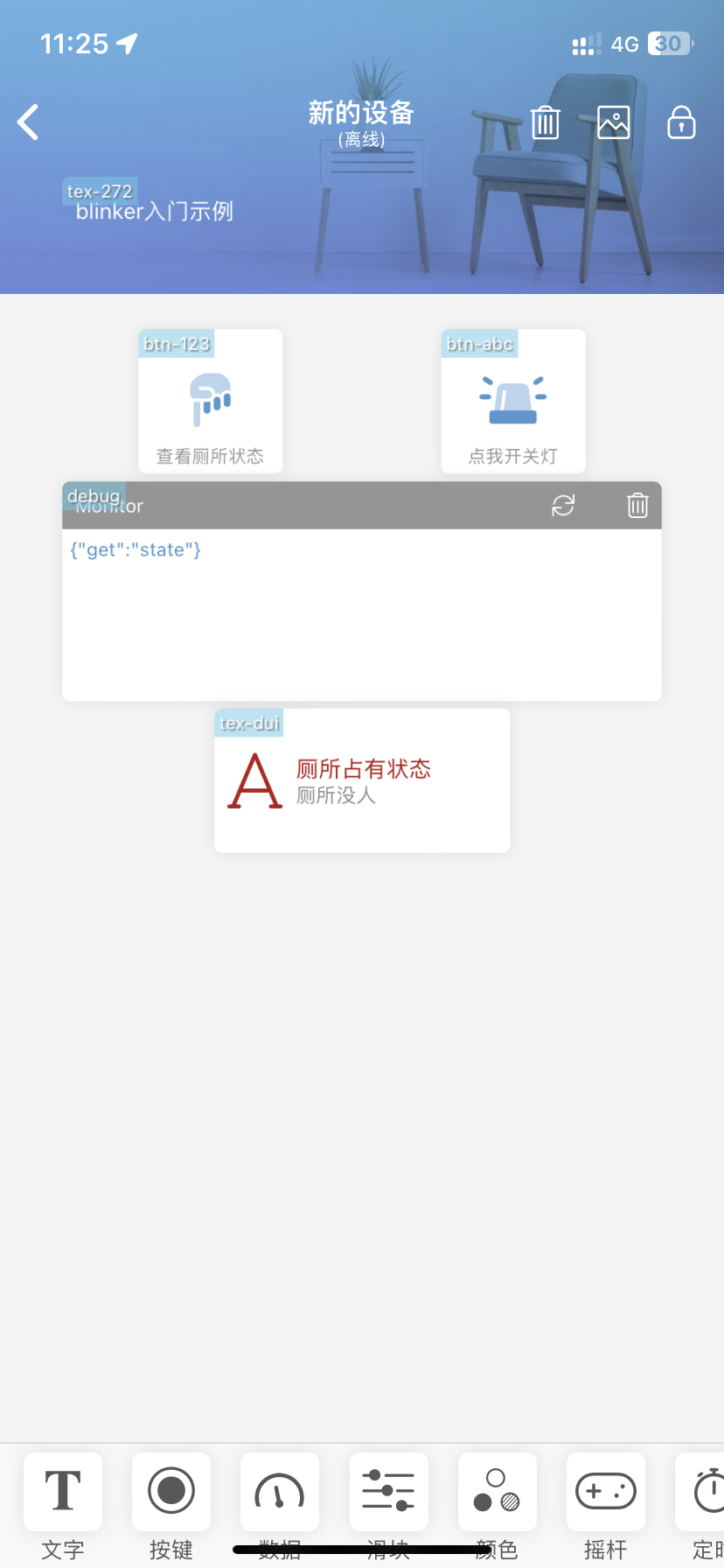
#### esp8266 WIFI模块的烧录代码（连接宿舍wifi、Blinker服务器）

手机APP对应的UI控件

BlinkerButton Button1("btn-abc");//开关的UI按钮

BlinkerNumber Number1("num-abc");//查看厕所状态的按钮

BlinkerText Text1("tex-dui");//厕所状态的文字框



//Arduino

#define BLINKER\_PRINT Serial

#define BLINKER\_WIFI

#include <Blinker.h>

char auth[] = "86f3b7d0d871";//Blinker服务器中的设备识别码

char ssid[] = "9721\_2.4G";//宿舍wifi与密码

char pswd[] = "97219721";

// 新建组件对象

BlinkerButton Button1("btn-abc");//开关的UI按钮

BlinkerNumber Number1("num-abc");//查看厕所状态的按钮

BlinkerText Text1("tex-dui");//厕所状态的文字框

int counter = 0;

// 按下按键即会执行该函数

void button1\_callback(const String & state) {

BLINKER\_LOG("get button state: ", state);

digitalWrite(LED\_BUILTIN, !digitalRead(LED\_BUILTIN));

}

// 如果未绑定的组件被触发，则会执行其中内容

void dataRead(const String & data)

{

BLINKER\_LOG("Blinker readString: ", data);

counter++;

Number1.print(counter);

Blinker.vibrate();

uint32\_t BlinkerTime = millis();

Blinker.print("millis", BlinkerTime);

if(char(Serial.read())=='|')//读取串口信息。如果读到｜则说明厕所有人

Text1.print("厕所状态", "有人");

else if(char(Serial.read())=='/')//若读取到 / 则表示，厕所门开着表示厕所没人

Text1.print("厕所状态","没人");

}

void setup() {

// 初始化串口

Serial.begin(115200);

#if defined(BLINKER\_PRINT)

BLINKER\_DEBUG.stream(BLINKER\_PRINT);

#endif

// 初始化有LED的IO

pinMode(LED\_BUILTIN, OUTPUT);

// 初始化blinker

Blinker.begin(auth, ssid, pswd);//登陆blinker服务器

Blinker.attachData(dataRead);

Button1.attach(button1\_callback);

}

void loop() {

Blinker.run();

}

#### 1.2 arduino开发板、串口通信代码

#include <ArduinoJson.h>

byte sensorPin = 3;//人体红外传感器

byte indicator = 13;//人体红外传感器模块 自带的测试小灯 亮-有人 暗-没人

int red=8;//门外警示灯

int white=7;//照明灯

int fan=9;//风扇

int crushPin = 2; // 碰撞传感器

int DoorStatus=0; //门的状态 0 ->off 1->on

int wifiState=0; //wifi的开关灯按钮 是要控制灯开还是关 1 on $$ 0 off

int count=0;//远程UI 开灯按的次数tap

void WIFImes(){

String inString="";

String com1="";

while(Serial.available()>0){//读取串口信息 即读取esp8266发送的信息

inString += char(Serial.read());

delay(10); // 延时函数用于等待字符完全进入缓冲区，可以尝试没有延时，输出结果会是什么

}

// 检查是否接收到数据，如果接收到数据，则输出该数据

if(inString!=""){

com1=inString.substring(inString.length()-5,inString.length()-2);//字符串切割，得到手机button的点击信息

Serial.print("commond:");

Serial.println(inString);

}

if(com1=="tap")

{

count++;

if(count==1)//tap按一次

wifiState=1;//开灯

if(count>1)//超过1次即2次

{

wifiState=0;//就关灯

count=0;

}

}

}

void setup()

{

pinMode(sensorPin,INPUT);//设置引脚模式

pinMode(indicator,OUTPUT);

Serial.begin(115200);

pinMode(red,OUTPUT);

pinMode(white,OUTPUT);

pinMode(fan,OUTPUT);

pinMode(crushPin, INPUT);

}

void loop()

{

byte humanState = digitalRead(sensorPin);

byte doorState = digitalRead(crushPin);

digitalWrite(indicator,humanState);

WIFImes();

// 单独 人体检测

if(humanState == 1)

{

// Serial.println("Somebody is in this area!");

digitalWrite(red, HIGH); //将管脚设置为高电平, 则LED灯亮

digitalWrite(white, HIGH); //将管脚设置为高电平, 则LED灯亮

digitalWrite(fan, HIGH); //将管脚设置为高电平, 则LED灯亮

Serial.print("|");// 有人 给esp8266发送的串口通信信息

}

if(humanState == 0&&doorState==0)

{

// Serial.println("No one!");

digitalWrite(red, LOW); //将管脚设置为高电平, 则LED灯亮

digitalWrite(white, LOW); //将管脚设置为高电平, 则LED灯亮

digitalWrite(fan, LOW); //将管脚设置为高电平, 则LED灯亮

Serial.print("/");//没人 给esp8266发送的串口通信信息

}

//单独 手机远程控制

if(wifiState == 1)

{

digitalWrite(red, HIGH); //将管脚设置为高电平, 则LED灯亮

digitalWrite(white, HIGH); //将管脚设置为高电平, 则LED灯亮

digitalWrite(fan, HIGH); //将管脚设置为高电平, 则LED灯亮

Serial.print("|");// 有人 给esp8266发送的串口通信信息

}

else

{

digitalWrite(red, LOW); //将管脚设置为低电平, 则LED灯不亮

digitalWrite(white, LOW); //将管脚设置为低电平, 则LED灯不亮

digitalWrite(fan, LOW); //将管脚设置为低电平, 则LED灯不亮

Serial.print("/");//没人 给esp8266发送的串口通信信息

}

//人体红外检测和远程操控结合。远程点击或者检测到人体——开灯，监测到厕所无人且门被开关过——关灯

if(wifiState == 1)

{

digitalWrite(red, HIGH); //将管脚设置为高电平, 则LED灯亮

digitalWrite(white, HIGH); //将管脚设置为高电平, 则LED灯亮

digitalWrite(fan, HIGH); //将管脚设置为高电平, 则LED灯亮

Serial.print("|");// 有人 给esp8266发送的串口通信信息

}

else if(humanState == 1)

{

// Serial.println("Somebody is in this area!");

digitalWrite(red, HIGH); //将管脚设置为高电平, 则LED灯亮

digitalWrite(white, HIGH); //将管脚设置为高电平, 则LED灯亮

digitalWrite(fan, HIGH); //将管脚设置为高电平, 则LED灯亮

Serial.print("|");// 有人 给esp8266发送的串口通信信息

}

else if(humanState == 0&&doorState==0)

{

// Serial.println("No one!");

digitalWrite(red, LOW); //将管脚设置为低电平, 则LED灯不亮

digitalWrite(white, LOW); //将管脚设置为低电平, 则LED灯不亮

digitalWrite(fan, LOW); //将管脚设置为低电平, 则LED灯不亮

Serial.print("/");//没人 给esp8266发送的串口通信信息

}

else{

Serial.print("/");//没人 给esp8266发送的串口通信信息

}

delay(5);

}

## 二、树莓派代码

#### 2.1 AS608指纹模块头文件（as608.h）

**//具体实现请查看as608.c文件**

#ifndef \_\_AS608\_H\_\_

#define \_\_AS608\_H\_\_

#ifndef \_\_cplusplus

#include <stdbool.h>

#endif

typedef unsigned char uchar;

typedef unsigned int uint;

typedef struct AS608\_Module\_Info {

uint status; // 状态寄存器 0

uint model; // 传感器类型 0-15

uint capacity; // 指纹容量，300

uint secure\_level; // 安全等级 1/2/3/4/5，默认为3

uint packet\_size; // 数据包大小 32/64/128/256 bytes，默认为128

uint baud\_rate; // 波特率系数

uint chip\_addr; // 设备(芯片)地址

uint password; // 通信密码

char product\_sn[12]; // 产品型号

char software\_version[12]; // 软件版本号

char manufacture[12]; // 厂家名称

char sensor\_name[12]; // 传感器名称

uint detect\_pin; // AS608的WAK引脚连接的树莓派GPIO引脚号

uint has\_password; // 是否有密码

} AS608;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BEGIN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 全局变量

\*/

extern AS608 g\_as608;

extern int g\_fd; // 文件描述符，即open()函数打开串口的返回值

extern int g\_verbose; // 输出信息的详细程度

extern char g\_error\_desc[128]; // 错误代码的含义

extern uchar g\_error\_code; // 模块返回的确认码，如果函数返回值不为true，读取此变量

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifdef \_\_cplusplus

extern "C" {

#endif

extern bool PS\_Setup(uint chipAddr, uint password); // 0x00000000 ~ 0xffffffff

extern bool PS\_GetImage();

extern bool PS\_GenChar(uchar bufferID);

extern bool PS\_Match(int\* pScore);

extern bool PS\_Search(uchar bufferID, int startPageID, int count, int\* pPageID, int\* pScore);

extern bool PS\_RegModel();

extern bool PS\_StoreChar(uchar bufferID, int pageID);

extern bool PS\_LoadChar(uchar bufferID, int pageID);

extern bool PS\_UpChar(uchar bufferID, const char\* filename);

extern bool PS\_DownChar(uchar bufferID, const char\* filename);

extern bool PS\_UpImage(const char\* filename);

extern bool PS\_DownImage(const char\* filename);

extern bool PS\_DeleteChar(int startpageID, int count);

extern bool PS\_Empty();

extern bool PS\_WriteReg(int regID, int value);

extern bool PS\_ReadSysPara();

extern bool PS\_Enroll(int\* pPageID);

extern bool PS\_Identify(int\* pPageID, int\* pScore);

extern bool PS\_SetPwd(uint passwd); // 4字节无符号整数

extern bool PS\_VfyPwd(uint passwd); // 4字节无符号整数

extern bool PS\_GetRandomCode(uint\* pRandom);

extern bool PS\_SetChipAddr(uint newAddr);

extern bool PS\_ReadINFpage(uchar\* pInfo, int size/\*>=512\*/);

extern bool PS\_WriteNotepad(int notePageID, uchar\* pContent, int contentSize);

extern bool PS\_ReadNotepad(int notePageID, uchar\* pContent, int contentSize);

extern bool PS\_HighSpeedSearch(uchar bufferID, int startPageID, int count, int\* pPageID, int\* pScore);

extern bool PS\_ValidTempleteNum(int\* pValidN);

extern bool PS\_ReadIndexTable(int\* indexList, int size);

// 封装函数

extern bool PS\_DetectFinger();

extern bool PS\_SetBaudRate(int value);

extern bool PS\_SetSecureLevel(int level);

extern bool PS\_SetPacketSize(int size);

extern bool PS\_GetAllInfo();

extern bool PS\_Flush();

// 获得错误代码g\_error\_code的含义，并赋值给g\_error\_desc

extern char\* PS\_GetErrorDesc();

#ifdef \_\_cplusplus

}

#endif

#endif // \_\_AS608\_H\_\_

#### 2.2 AS608指纹识别主程序(main.c)

#include "../as608.h"

#include "./utils.h"

#include <wiringPi.h>

#include <wiringSerial.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

extern AS608 g\_as608;

extern int g\_fd;

extern int g\_verbose;

extern char g\_error\_desc[128];

extern uchar g\_error\_code;

int g\_argc = 0; // 参数个数，g\_argc = argc - g\_option\_count

int g\_option\_count = 0; // 选项个数-v、-h等

char g\_command[16] = { 0 }; // 即argv[1]

Config g\_config; // 配置文件 结构体，定义在"./utils.h"头文件中

void printConfig();

void printUsage();

bool readConfig(); // 读取文件到 g\_config

bool writeConfig(); // 将 g\_config 写入文件

void asyncConfig();

void priorAnalyseArgv(int argc, char\* argv[]);

void analyseArgv(int argc, char\* argv[]);

bool waitUntilDetectFinger(int wait\_time); // 阻塞至检测到手指，最长阻塞wait\_time毫秒

bool waitUntilNotDetectFinger(int wait\_time);

void checkFinger();

// 因为as608.h内的函数执行失败而退出程序

bool PS\_Exit() {

printf("ERROR! code=%02X, desc=%s\n", g\_error\_code, PS\_GetErrorDesc());

exit(2);

return true;

}

// 程序退出时执行的工作，关闭串口等

void atExitFunc() {

if (g\_verbose == 1)

printf("Exit\n");

if (g\_fd > 0)

serialClose(g\_fd);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* main()

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(int argc, char \*argv[]) //int serialOpen (const char \*device, const int baud)

{

// 1.读取配置文件，获得芯片地址和通信密码

if (!readConfig())

exit(1);

// 2.优先解析的内容，如参数选项、配置本地文件等

priorAnalyseArgv(argc, argv);

if (g\_verbose == 1)

printConfig();

// 3.初始化wiringPi库

if (-1 == wiringPiSetup()) {

printf("wiringPi setup failed!\n");

return 1;

}

// 4.检测是否有手指放上的GPIO端口，设为输入模式

pinMode(g\_config.detect\_pin, INPUT);

// 5.打开串口

if((g\_fd = serialOpen(g\_config.serial, g\_config.baudrate)) < 0) {

fprintf(stderr,"Unable to open serial device: %s\n", strerror(errno));

return 1;

}

// 6.注册退出函数(打印一些信息、关闭串口等)

atexit(atExitFunc);

// 7.初始化 AS608 模块

// 地址 密码

PS\_Setup(g\_config.address, g\_config.password) || PS\_Exit();

// 8.主处理函数，解析普通命令(argv[1])，

analyseArgv(argc, argv);

//checkFinger();

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* 函数定义

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*void checkFinger(){

while(1){

//checkArgc(2);

printf("Please put your finger on the module.\n");

delay(2000);

PS\_GetImage();

PS\_GenChar(1);

int pageID = 0, score = 0;

if (!PS\_Search(1, 0, 300, &pageID, &score))

continue;

else{

int PWM\_pin = 1;

int intensity ;

if (wiringPiSetup () == -1)

exit (1) ;

pinMode (PWM\_pin, PWM\_OUTPUT) ;

int time = 1;

while (time--)

{

for (intensity = 0 ; intensity < 1024 ; ++intensity)

{

pwmWrite (PWM\_pin, intensity) ;

delay (1) ;

}

delay(1);

for (intensity = 1023 ; intensity >= 0 ; --intensity)

{

pwmWrite (PWM\_pin, intensity) ;

delay (1) ;

}

delay(1);

printf("Matched! pageID=%d score=%d\n", pageID, score);

}

break;

}

}

}\*/

// 检查参数数量是否正确是否

bool checkArgc(int argcNum) {

if (argcNum == g\_argc)

return true;

else if (argcNum == 2)

printf("ERROR! \"%s\" accept no parameter\n", g\_command);

else if (argcNum == 3)

printf("ERROR! \"%s\" accept 1 parameter\n", g\_command);

else if (argcNum > 3)

printf("ERROR! \"%s\" accept %d parameters\n", g\_command, argcNum);

exit(1);

}

// 匹配argv[1], 即g\_command

bool match(const char\* str) {

return strcmp(str, g\_command) == 0;

}

// 需要优先解析的命令，如配置文件的修改、选项解析等

// 不需要与模块通信

void priorAnalyseArgv(int argc, char\* argv[]) {

if (argc < 2) {

printUsage();

exit(1);

}

// 检查选项 -v -h

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

if (strcmp(argv[i], "-h") == 0) {

printUsage();

g\_option\_count++;

exit(0);

}

else if (strcmp(argv[i], "-v") == 0) {

g\_verbose = 1;

g\_option\_count++;

}

}

g\_argc = argc - g\_option\_count;

strcpy(g\_command, argv[1]);

if (match("cfg")) {

printConfig();

exit(0);

}

// 配置通信地址

else if (match("cfgaddr")) {

checkArgc(3);

g\_config.address = toUInt(argv[2]);

writeConfig();

exit(0);

}

// 配置通信密码

else if (match("cfgpwd")) {

checkArgc(3);

g\_config.password = toUInt(argv[2]);

g\_config.has\_password = 1;

writeConfig();

exit(0);

}

// 配置串口号

else if (match("cfgserial")) {

checkArgc(3);

strcpy(g\_config.serial, argv[2]);

writeConfig();

exit(0);

}

else if (match("cfgbaud")) {

checkArgc(3);

g\_config.baudrate = toInt(argv[2]);

writeConfig();

exit(0);

}

else if (match("cfgpin")) {

checkArgc(3);

g\_config.detect\_pin = toInt(argv[2]);

writeConfig();

exit(0);

}

}

// 阻塞至检测到手指，最长阻塞wait\_time毫秒

bool waitUntilDetectFinger(int wait\_time) {

while (true) {

if (PS\_DetectFinger()) {

return true;

}

else {

delay(100);

wait\_time -= 100;

if (wait\_time < 0) {

return false;

}

}

}

}

bool waitUntilNotDetectFinger(int wait\_time) {

while (true) {

if (!PS\_DetectFinger()) {

return true;

}

else {

delay(100);

wait\_time -= 100;

if (wait\_time < 0) {

return false;

}

}

}

}

// 主处理函数，解析命令

void analyseArgv(int argc, char\* argv[]) {

if (match("add")) {

checkArgc(3);

printf("Please put your finger on the module.\n");

if (waitUntilDetectFinger(5000)) {

delay(500);

PS\_GetImage() || PS\_Exit();

PS\_GenChar(1) || PS\_Exit();

}

else {

printf("Error: Didn't detect finger!\n");

exit(1);

}

// 判断用户是否抬起了手指，

printf("Ok.\nPlease raise your finger!\n");

if (waitUntilNotDetectFinger(5000)) {

delay(100);

printf("Ok.\nPlease put your finger again!\n");

// 第二次录入指纹

if (waitUntilDetectFinger(5000)) {

delay(500);

PS\_GetImage() || PS\_Exit();

PS\_GenChar(2) || PS\_Exit();

}

else {

printf("Error: Didn't detect finger!\n");

exit(1);

}

}

else {

printf("Error! Didn't raise your finger\n");

exit(1);

}

int score = 0;

if (PS\_Match(&score)) {

printf("Matched! score=%d\n", score);

}

else {

printf("Not matched, raise your finger and put it on again.\n");

exit(1);

}

if (g\_error\_code != 0x00)

PS\_Exit();

// 合并特征文件

PS\_RegModel() || PS\_Exit();

PS\_StoreChar(2, toInt(argv[2])) || PS\_Exit();

printf("OK! New fingerprint saved to pageID=%d\n", toInt(argv[2]));

}

else if (match("delete")) {

int startPageID = 0;

int count = 0;

// 判断参数个数

if (g\_argc == 3) {

startPageID = toInt(argv[2]);

count = 1;

printf("Confirm to delete fingerprint %d: (Y/n)? ", startPageID);

}

else if (argc == 4) {

startPageID = toInt(argv[2]);

count = toInt(argv[3]);

printf("Confirm to delete fingerprint %d-%d: (Y/n)? ", startPageID, startPageID+count-1);

}

else {

printf("Command \"delete\" accept 1 or 2 parameter\n");

printf(" Usage: fp delete startPageID [count]\n");

exit(1);

}

// 询问是否继续

fflush(stdout);

char cmd = getchar();

if (cmd == 'n' || cmd == 'N') {

printf("Canceled!\n");

exit(0);

}

PS\_DeleteChar(startPageID, count) || PS\_Exit();

printf("OK!\n");

}

else if (match("search")) {

//checkArgc(2);

pinMode(27,OUTPUT);

pinMode(26,OUTPUT);

while(1){

printf("Please put your finger on the module.\n");

delay(2000);

PS\_GetImage();

PS\_GenChar(1);

int pageID = 0, score = 0;

if (!PS\_Search(1, 0, 300, &pageID, &score)){

//PS\_Exit();

digitalWrite(27,1);

continue;

}

else{

int PWM\_PIN = 1;

wiringPiSetup();

digitalWrite(27,0);

digitalWrite(26,1);

sleep(1);

digitalWrite(26,0);

pinMode(PWM\_PIN, PWM\_OUTPUT); // 设置PWM输出

pwmSetMode(PWM\_MODE\_MS); // 设置传统模式

pwmSetClock(192); // 设置分频

pwmSetRange(2000); // 设置周期分为2000步

pwmWrite(PWM\_PIN, 200);

delay(900);

pwmWrite(PWM\_PIN, 150);

delay(3000);

pwmWrite(PWM\_PIN, 100);

delay(900);

pwmWrite(PWM\_PIN, 150);

printf("Matched! pageID=%d score=%d\n", pageID, score);

}

}

}

// 列出指纹列表

else if (match("list")) {

checkArgc(2);

int indexList[512] = { 0 };

PS\_ReadIndexTable(indexList, 512) || PS\_Exit();

int i = 0;

for (i = 0; i < 300; ++i) {

if (indexList[i] == -1)

break;

printf("%d\n", indexList[i]);

}

if (i == 0) {

printf("The database is empty!\n");

}

}

else {

printf("Unknown parameter \"%s\"\n", argv[1]);

exit(1);

}

} // end analyseArgv

// 打印配置文件内容到屏幕上

void printConfig() {

printf("address=%08x\n", g\_config.address);

if (g\_config.has\_password)

printf("password=%08x\n", g\_config.password);

else

printf("password=none(no password)\n");

printf("serial\_file=%s\n", g\_config.serial);

printf("baudrate=%d\n", g\_config.baudrate);

printf("detect\_pin=%d\n", g\_config.detect\_pin);

}

// 同步g\_config变量内容和其他变量内容

void asyncConfig() {

g\_as608.detect\_pin = g\_config.detect\_pin;

g\_as608.has\_password = g\_config.has\_password;

g\_as608.password = g\_config.password;

g\_as608.chip\_addr = g\_config.address;

g\_as608.baud\_rate = g\_config.baudrate;

}

// 读取配置文件

bool readConfig() {

FILE\* fp;

// 获取用户主目录

char filename[256] = { 0 };

sprintf(filename, "%s/.fpconfig", getenv("HOME"));

// 主目录下的配置文件

if (access(filename, F\_OK) == 0) {

trimSpaceInFile(filename);

fp = fopen(filename, "r");

}

else {

// 如果配置文件不存在，就在主目录下创建配置文件，并写入默认配置

// 设置默认值

g\_config.address = 0xffffffff;

g\_config.password= 0x00000000;

g\_config.has\_password = 0;

g\_config.baudrate = 9600;

g\_config.detect\_pin = 1;

strcpy(g\_config.serial, "/dev/ttyAMA0");

writeConfig();

printf("Please config the address and password in \"~/.fpconfig\"\n");

printf(" fp cfgaddr 0x[address]\n");

printf(" fp cfgpwd 0x[password]\n");

printf(" fp cfgserial [serialFile]\n");

printf(" fp cfgbaud [rate]\n");

printf(" fp cfgpin [GPIO\_pin]\n");

return false;

}

char key[16] = { 0 };

char value[16] = { 0 };

char line[32] = { 0 };

char \*tmp;

while (!feof(fp)) {

fgets(line, 32, fp);

// 分割字符串，得到key和value

if (tmp = strtok(line, "="))

trim(tmp, key);

else

continue;

if (tmp = strtok(NULL, "="))

trim(tmp, value);

else

continue;

while (!tmp)

tmp = strtok(NULL, "=");

// 如果数值以 0x 开头

int offset = 0;

if (value[0] == '0' && (value[1] == 'x' || value[1] == 'X'))

offset = 2;

if (strcmp(key, "address") == 0) {

g\_config.address = toUInt(value+offset);

}

else if (strcmp(key, "password") == 0) {

if (strcmp(value, "none") == 0 || strcmp(value, "false") == 0) {

g\_config.has\_password = 0; // 无密码

}

else {

g\_config.has\_password = 1; // 有密码

g\_config.password = toUInt(value+offset);

}

}

else if (strcmp(key, "serial") == 0) {

int len = strlen(value);

if (value[len-1] == '\n')

value[len-1] = 0;

strcpy(g\_config.serial, value);

}

else if (strcmp(key, "baudrate") == 0) {

g\_config.baudrate = toInt(value);

}

else if (strcmp(key, "detect\_pin") == 0) {

g\_config.detect\_pin = toInt(value);

}

else {

printf("Unknown key:%s\n", key);

fclose(fp);

return false;

}

} // end while(!feof(fp))

asyncConfig();

fclose(fp);

return true;

}

/\*

\* 写配置文件

\*/

bool writeConfig() {

// 获取用户主目录

char filename[256] = { 0 };

sprintf(filename, "%s/.fpconfig", getenv("HOME"));

FILE\* fp = fp = fopen(filename, "w+");

if (!fp) {

printf("Write config file error!\n");

exit(0);

}

fprintf(fp, "address=0x%08x\n", g\_config.address);

if (g\_config.has\_password)

fprintf(fp, "password=0x%08x\n", g\_config.password);

else

fprintf(fp, "password=none\n");

fprintf(fp, "baudrate=%d\n", g\_config.baudrate);

fprintf(fp, "detect\_pin=%d\n", g\_config.detect\_pin);

fprintf(fp, "serial=%s\n", g\_config.serial);

fclose(fp);

}

// 打印程序使用说明

void printUsage() {

printf("A command line program to interact with AS608 module.\n\n");

printf("Usage:\n ./fp [command] [param] [option]\n");

printf("\nAvailable Commands:\n");

printf("-------------------------------------------------------------------------\n");

printf(" command | param | description\n");

printf("-------------------------------------------------------------------------\n");

printf(" cfgaddr [addr] Config address in local config file\n");

printf(" cfgpwd [pwd] Config password in local config file\n");

printf(" cfgserial [serialFile] Config serial port in local config file. Default:/dev/ttyAMA0\n");

printf(" cfgbaud [rate] Config baud rate in local config file\n");

printf(" cfgpin [GPIO\_pin] Config GPIO pin to detect finger in local confilg file\n\n");

printf(" add [pID] Add a new fingerprint to database. (Read twice) \n");

printf(" enroll [] Add a new fingerprint to database. (Read only once)\n");

printf(" search [] Collect fingerprint and search in database.\n");

printf(" list [] Show the registered fingerprints list.\n");

printf("\nAvaiable options:\n");

printf(" -h Show help\n");

printf(" -v Shwo details while excute the order\n");

printf("\nUsage:\n ./fp [command] [param] [option]\n\n");

}

#### Blinker连接智能门锁（PiConnectBlinker.py）

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

\_\_author\_\_ = "stao"

from blinker import Device, ButtonWidget, NumberWidget

import wiringpi

device = Device("3f07e35c3649")//服务器设备识别码

INPUT = 0

OUTPUT = 1

PWM\_OUTPUT = 2

GPIO\_CLOCK = 3

SOFT\_PWM\_OUTPUT = 4

SOFT\_TONE\_OUTPUT= 5

PWM\_TONE\_OUTPUT = 6

LOW = 0

HIGH = 1

PWM\_MODE\_MS = 0

PWM\_MODE\_BAL = 1

button1 = device.addWidget(ButtonWidget('btn-123'))

button2 = device.addWidget(ButtonWidget('btn-abc'))

number1 = device.addWidget(NumberWidget('num-abc'))

num = 0

async def button1\_callback(msg):

global num

num += 1

await number1.text("num").value(num).update()

async def button2\_callback(msg):

print("Button2: {0}".format(msg))

PWM\_PIN = 1

wiringpi.wiringPiSetup() # WiringPi初始化

wiringpi.pinMode(PWM\_PIN, PWM\_OUTPUT) # 设置引脚为PWM模式

wiringpi.pwmSetMode(PWM\_MODE\_MS) # 修改PWM为传统模式

wiringpi.pwmSetClock(192) # 设置分频系数

wiringpi.pwmSetRange(2000) # 设置周期步长(占空比范围)

wiringpi.pwmWrite(PWM\_PIN, 200)#设置舵机逆时针旋转速度

wiringpi.delay(800)#设置旋转持续时间

wiringpi.pwmWrite(PWM\_PIN, 150) #设置舵机速度为0

wiringpi.delay(3000)

wiringpi.pwmWrite(PWM\_PIN, 100) #设置舵机顺时针旋转速度

wiringpi.delay(730)

wiringpi.pwmWrite(PWM\_PIN, 150) #设置舵机速度为0

async def heartbeat\_func(msg):

print("Heartbeat func received: {0}".format(msg))

# 文本组件

async def ready\_func():

# 获取设备配置信息

print(vars(device.config))

button1.func = button1\_callback

button2.func = button2\_callback

device.heartbeat\_callable = heartbeat\_func

device.ready\_callable = ready\_func

if \_\_name\_\_ == '\_\_main\_\_':

device.run()