****

《嵌入式系统》

课程设计报告

智能家居控制系统搭建指南（基于STM32F429）

指导老师： 贺建飚

学 院： 计算机学院

专 业： 物联网工程

班 级： 物联网1801 1802

姓 名： 刘润 王云鹏

目录

[1 问题分析 3](#_Toc76804141)

[1.1 课程设计题目 3](#_Toc76804142)

[1.2 需求分析 3](#_Toc76804143)

[1.3 设计流程 3](#_Toc76804144)

[2 配置开发环境 3](#_Toc76804145)

[3 交互界面与显示 4](#_Toc76804146)

[3.1 设计交互界面 4](#_Toc76804147)

[3.2 图片显示 5](#_Toc76804148)

[3.3 文字显示 6](#_Toc76804149)

[4 触控反馈 6](#_Toc76804150)

[4.1触控基本原理 6](#_Toc76804151)

[4.2 计算触控区域 7](#_Toc76804152)

[5 撤防设防逻辑 8](#_Toc76804153)

[5.1 播放警报声音 8](#_Toc76804154)

[5.2 逻辑设计 9](#_Toc76804155)

[6 中断设计 9](#_Toc76804156)

[6 课程设计结果 14](#_Toc76804157)

[7 课设体会 14](#_Toc76804158)

## 问题分析

### 1.1 课程设计题目

**智能家居控制系统：**完成一个智能家居控制系统设计，要求采集室内多种信号（包括但不限于门磁开关、窗磁开关、煤气泄漏开关等），可在液晶屏上实时显示各开关状态，当系统处于设防状态时，一旦门被打开则报警，其他信号也做相应报警，撤防后不报警。同时可以通过相应按钮手动打开窗帘、电灯等。

### 1.2 需求分析

该嵌入式系统应包括功能为：对各开关（数字量）的监测、将检测信息显示在触控屏幕上、能够在发生异常情况时通过扬声器报警以及通过用户按动按键（数字量）控制外设。由于缺少相应外设，于是我们令开发板上的LED0和LED1分别代替窗帘和电灯。当用户通过按钮手动打开窗帘或电灯时，对应的LED会亮起，表示对应外设已启动。

### 1.3 设计流程

该课程设计的设计流程如下：

1. 在主机上配置好开发板的开发环境；
2. 设计人机交互界面，将界面显示在开发板的触摸屏上；
3. 设计触控反馈逻辑，将反馈信号显示在对应的LED上；
4. 设计设防撤防逻辑，并配置扬声器使其能够发出警报；

## 配置开发环境

下载软件Keil uVision5，并在官网安装STM32F429的MDK5 Software Packs，如下图1所示。下载完毕后，安装Keil uVision5并安装STM32F429的MDK5 Software Packs后，就可以连接好开发板的ST-LINK，准备灌入编写好的程序。由于没有许可证的Keil uVision5不能够编写较大的程序，所以需要我们通过一些手段获取到Keil uVision5的许可证，在这里就不详细说明如何获取许可证，可以自行百度查找破解方法。

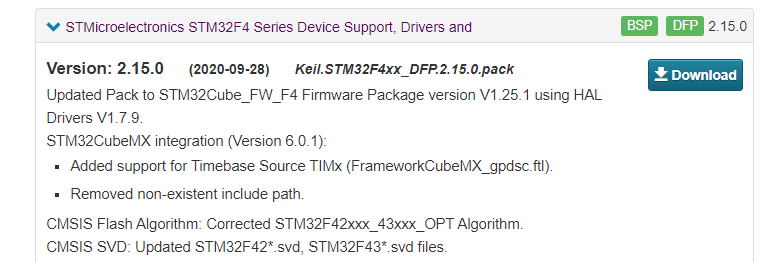


图1 在https://www.keil.com/dd2/Pack下载对应芯片软件包

## 交互界面与显示

### 3.1 设计交互界面

该嵌入式系统的触摸屏为人机交互设备，用户可以在触摸屏上实时查看各设备状态，此外还可以通过点击屏幕上设置的按键，远程控制各设备的状态。由于代码没有图形用户接口，所以我们人工绘制图片界面，通过监听用户触控的位置，进而判断用户的点击意图，从而实现远程控制各设备的状态功能。

实验所提供的触控屏的分辨率为480×800。在进行一系列测试后，发现该触控屏能够很好地显示出19.2厘米×36厘米的图片（无黑边）。于是根据题目要求，我们设计了如图2所示的人机交互界面。



图2 人机交互界面

|  |
| --- |
| * 注意   显示的图片的边长要尽可能满足9.6:18这样的比例，其它比例可能会在触控屏上产生黑边，图片的大小可以在满足比例的条件下主观调整。 |

### 3.2 图片显示

图片显示需要“ff.h”头文件。图片显示的过程如下：打开图片文件夹→得到总的有效文件数→打开目录显示图片。

|  |
| --- |
| * 注意   需要准备一张SD卡，并将画好的人机交互界面放到SD卡的PICTURE文件夹中（PICTURE文件夹需要自己创建），然后将SD卡插入到开发板的SD卡槽中。这张SD卡在开发板中会被识别为O:盘。 |

对应代码如下所示：

|  |
| --- |
| u16 pic\_get\_tnum(u8 \*path)  {  u8 res;  u16 rval=0;  DIR tdir;  FILINFO \*tfileinfo;  tfileinfo=(FILINFO\*)mymalloc(SRAMIN,sizeof(FILINFO));  res=f\_opendir(&tdir,(const TCHAR\*)path);  if(res==FR\_OK&&tfileinfo)  {  while(1)  {  res=f\_readdir(&tdir,tfileinfo);  if(res!=FR\_OK||tfileinfo->fname[0]==0)break;  res=f\_typetell((u8\*)tfileinfo->fname);  if((res&0XF0)==0X50)  {  rval++;  }  }  }  myfree(SRAMIN,tfileinfo);  return rval;  }  Int main(void){  ……  while(f\_opendir(&picdir,"0:/PICTURE"))//打开图片文件夹  {  Show\_Str(30,170,240,16,"PICTURE文件夹错误",16,0);  delay\_ms(200);  LCD\_Fill(30,170,240,186,WHITE);  delay\_ms(200);  }  totpicnum=pic\_get\_tnum("0:/PICTURE");//得到总有效文件数  while(totpicnum==NULL)//如果文件数为0  {  Show\_Str(30,170,240,16,"没有图片文件",16,0);  delay\_ms(200);  LCD\_Fill(30,170,240,186,WHITE);  delay\_ms(200);  }  res=f\_opendir(&picdir,(const TCHAR\*)"0:/PICTURE");//打开目录  while(res==FR\_OK){  ……  }  ……  } |

### 3.3 文字显示

文字显示需要“fontupd.h”头文件，将该头文件添加到工程文件后，就可以调用函数在LCD屏幕上显示文字了。首先要检查字库，字体无异常后才能够使用字库。对应代码如下所示：

|  |
| --- |
| while(**font\_init()**){ //检查字库函数  LCD\_ShowString(30,50,200,16,16,"Font Error!");  delay\_ms(200);  LCD\_Fill(30,50,240,66,WHITE);  delay\_ms(200);  } |

当没有提示“Font Error”的时候，就说明字库正常，能够使用字库中的文字。

## 4 触控反馈

### 4.1触控基本原理

实现触控需要添加“touch.h”头文件。里面有一个TP\_Scan函数，说明了触摸按键扫描过程：

|  |
| --- |
| u8 TP\_Scan(u8 tp)  {  if(PEN==0)  {  if(tp)TP\_Read\_XY2(&tp\_dev.x[0],&tp\_dev.y[0]);  else if(TP\_Read\_XY2(&tp\_dev.x[0],&tp\_dev.y[0]))  {  tp\_dev.x[0]=tp\_dev.xfac\*tp\_dev.x[0]+tp\_dev.xoff;  tp\_dev.y[0]=tp\_dev.yfac\*tp\_dev.y[0]+tp\_dev.yoff;  }  if((tp\_dev.sta&TP\_PRES\_DOWN)==0)  {  tp\_dev.sta=TP\_PRES\_DOWN|TP\_CATH\_PRES;  tp\_dev.x[4]=tp\_dev.x[0];  tp\_dev.y[4]=tp\_dev.y[0];  }  }else  {  if(tp\_dev.sta&TP\_PRES\_DOWN)  {  tp\_dev.sta&=~(1<<7);  }else  {  tp\_dev.x[4]=0;  tp\_dev.y[4]=0;  tp\_dev.x[0]=0xffff;  tp\_dev.y[0]=0xffff;  }  }  return tp\_dev.sta&TP\_PRES\_DOWN;  } |

当调用tp\_dev.scan(0)时，LCD会扫描当前屏幕的触控状态，当发现用户用手触碰屏幕时，会将触碰的x和y位置存储于tp\_dev.x[0]和tp\_dev.y[0]中。

### 4.2 计算触控区域

由3.1节我们设计了触控屏的界面，我们需要计算按键的触控区域，为后续的触控逻辑做好准备。我们使用的屏幕的分辨率是480×800，我们需要用直尺量出各个按键区域相对于屏幕左上角的位置，我们记某点相对于屏幕左上角的位置为，单位cm，则对应的tp\_dev.x[0]和tp\_dev.y[0]数值分别为



其中分别代表屏幕的x方向上的分辨率和y方向上的分辨率，分别代表屏幕的x方向上的分辨率和y方向上的长度。通过上式，就可以计算出各个需要设计触控反馈的触控区域了。相关代码如下：

|  |
| --- |
| if(tp\_dev.x[0]>56&&tp\_dev.x[0]<223&&tp\_dev.y[0]>62&&tp\_dev.y[0]<125){  LED0=0;  Show\_Str(155,470,240,24,"窗帘状态：打开",24,0);  }  if(tp\_dev.x[0]>251&&tp\_dev.x[0]<418&&tp\_dev.y[0]>62&&tp\_dev.y[0]<125){  LED0=1;  Show\_Str(155,470,240,24," 窗帘状态：关闭",24,0);  }  if(tp\_dev.x[0]>56&&tp\_dev.x[0]<223&&tp\_dev.y[0]>170&&tp\_dev.y[0]<230){  LED1=0;  Show\_Str(155,495,240,24,"电灯状态：打开",24,0);  }  if(tp\_dev.x[0]>251&&tp\_dev.x[0]<418&&tp\_dev.y[0]>170&&tp\_dev.y[0]<230){  LED1=1;  Show\_Str(155,495,240,24," 电灯状态：关闭",24,0);  }  if(tp\_dev.x[0]>56&&tp\_dev.x[0]<223&&tp\_dev.y[0]>275&&tp\_dev.y[0]<335){  Show\_Str(155,570,240,24,"设防状态：设防",24,0);  defence=1;  }  if(tp\_dev.x[0]>251&&tp\_dev.x[0]<418&&tp\_dev.y[0]>275&&tp\_dev.y[0]<335){  Show\_Str(155,570,240,24, "设防状态：关闭",24,0);  defence=0;  } |

## 5 撤防设防逻辑

### 5.1 播放警报声音

播放警报声音需要“WM8978.h”头文件，以及相关的拓扑依赖文件。

|  |
| --- |
| * 注意   将准备好的警报音频放到SD卡的MUSIC文件夹中（MUSIC文件夹需要自己创建），然后将SD卡插入到开发板的SD卡槽中。这张SD卡在开发板中会被识别为O:盘。另外要注意，**音频文件必须要是.wav类型的文件！.mp3的文件不识别。** |

首先要初始化WM8978，执行WM8978\_Init()，然后设置扬声器音量WM8978\_SPKvol\_Set(40)，在必要时，执行audio\_play()函数，则会听到扬声器发出声音。

### 5.2 逻辑设计

本系统有两个状态，分别是设防和撤防。顾名思义，设防就是保护房屋安全。所以在设防状态下，煤气危险会响起警报，窗户打开也会响起警报。而撤防状态下就不会响起警报。代码如下。

|  |
| --- |
| if( (defence==0&&co2Flag==1) || (defence==0&&curtainFlag==1) ){//警戒状态且煤气危险，或窗帘打开时，报警；当按键破坏条件后，停止播放警戒  audio\_play();  } |

另外，煤气危险时，系统会控制窗帘自动打开，以通风保证安全。代码如下。

|  |
| --- |
| if(co2Flag==0){  Show\_Str(155,520,240,24,"煤气安全：安全",24,0);  }else{  Show\_Str(155,520,240,24,"煤气安全：危险",24,0);  if(curtainFlag==0)curtainFlag=!curtainFlag;//关闭就打开  Show\_Str(155,470,240,24,"窗帘状态：打开",24,0);  } |

## 6 中断设计

这个部分是本系统最为复杂和困难的部分，大家做好准备。

原理：在主函数进入死循环时，当外部中断到来，程序会进入到中断服务程序执行，执行完之后再返回退出的位置接着执行。

本系统有两种中断，分别是按键中断和触摸屏中断，两者皆采用外部中断的方式实现。

按键中断的代码可参考“实验4 外部中断实验”，这个代码实现了按键控制LED灯的闪烁。我们修改了代码，使按键控制家具的状态显示。之所以这么做，是因为我们没有窗磁开关，也没有煤气传感器，所以就用按键来代替不同的外部状态。这个部分的实现较为简单。

另外一个中断是触摸屏中断，示例代码给出的触摸屏实验是轮询的形式（即不断循环判断是否有触摸）。轮询的代码可参考“实验30 触摸屏实验”，其中实现了一个在触摸屏上画画的功能，可最多同时支持五根手指。我们在实现本系统过程中，发现采用轮询的形式会导致系统反应太慢。然后我们又尝试了定时器中断，但是老师表示定时扫描屏幕太浪费系统资源，让我们再努力努力，于是我们开心的[doge]接着努力实现触摸屏中断。通过查阅网络，结合附带的资料，我们实现了触摸屏中断。但是尚无法做到一次触摸就使系统产生反应，需要连续点击两次，第一次进入中断，第二次使屏幕扫描到触摸坐标。这个部分的实现较为困难。

对于触摸屏中断的实现原理与实现代码，大家可以参考一个博客，经我们验证，其中部分代码是可用的，链接：<https://blog.csdn.net/weixin_43898067/article/details/108376124>

还有你要知道，这个LCD电容触摸屏分为LCD显示和触摸屏两个部分，这块屏幕由一块驱动芯片支持，经询问与查找，这个芯片的型号为GT911，其说明书链接：<https://wenku.baidu.com/view/4e91782303020740be1e650e52ea551810a6c984.html?re=view&qq-pf-to=pcqq.c2c>

关于触摸屏的驱动芯片，这个芯片给出了四根线连接到外面。包括I2C的SCL和SDL，所以触摸屏（准确的说是驱动芯片）和CPU通信是通过I2C协议的。还包括有一根中断int线，正是通过这根线发出的中断信号。经查阅资料，你如果跑我们的代码但是进不了中断，那么有可能是板子上一块叫做C32的电容把中断信号滤掉了，那么你只需要把那个电容焊下来就可以进入中断了。

另外，我们使用的正点原子的设备有一个官方论坛，其中由很多帖子，如果由无法解决的问题可尝试在其中搜索，链接: <http://www.openedv.com/forum.php>

最终这份代码可以做到连续点击屏幕两次做到成功触摸，第一次进入中断，然后程序会进入中断服务程序的读取触摸坐标循环中，第二次点击就可以使程序读到坐标并退出循环。当然我们很希望实现只需一次点击就能读取坐标，但经过努力后未能成功，我们希望后来者能做到只点击一次屏幕即实现坐标的读取。

为了实现这一点，你可能需要知道：I2C总线通讯协议并读懂I2C通讯部分的代码、GT911驱动ic的各个寄存器以及它们的作用（这个在说明书中有）、读懂相关芯片的硬件原理图（下载的包里有，请仔细寻找）等知识。当然这些知识应该并不足够，笔者做完了这些一样搞不懂中断的触摸服务程序怎么写，示例程序只有轮询的（那个程序也只能做到轮询）。所以我们鼓励做这个实验的同学多多查询资料和网上的资源，做到这一点。

接下来让我们结合代码讲一下实现过程。首先中断有个初始换函数，这个初始化函数会在主函数中执行一次，以在中断发生时通知CPU。代码如下，我们会在注释中详细解释原理与来源。

|  |
| --- |
| //外部中断初始化  void EXTI\_Init(void)  {  GPIO\_InitTypeDef GPIO\_Initure;    //时钟不开启，中断就不起作用，推测是用来同步总线上的传输脉冲  \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE(); //开启GPIOA时钟  \_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE(); //开启GPIOC时钟  \_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE(); //开启GPIOH时钟    //触摸屏中断初始化，这个部分的配置来源于上面提到的博客  //这个ph7是触摸屏驱动芯片的中断int针脚，可在芯片说明书中查到  GPIO\_Initure.Pin=GPIO\_PIN\_7; //PH7  GPIO\_Initure.Mode=GPIO\_MODE\_IT\_RISING; //上升沿触发  //中断来到时，int线上传来一个脉冲  GPIO\_Initure.Pull=GPIO\_PULLUP; //上拉电平  GPIO\_Initure.Speed=GPIO\_SPEED\_HIGH; //高速模式，I2C的几个模式之一  HAL\_GPIO\_Init(GPIOH,&GPIO\_Initure); //初始化中断  HAL\_NVIC\_SetPriority(EXTI9\_5\_IRQn,2,0); //设置优先级  HAL\_NVIC\_EnableIRQ(EXTI9\_5\_IRQn); //使能中断  //按键中断初始化，这里四个pin对应了四个按钮，就是板子上黄色的四个  GPIO\_Initure.Pin=GPIO\_PIN\_0; //PA0  GPIO\_Initure.Mode=GPIO\_MODE\_IT\_RISING; //上升沿触发  GPIO\_Initure.Pull=GPIO\_PULLDOWN;  HAL\_GPIO\_Init(GPIOA,&GPIO\_Initure);    GPIO\_Initure.Pin=GPIO\_PIN\_13; //PC13  GPIO\_Initure.Mode=GPIO\_MODE\_IT\_FALLING; //下降沿触发  GPIO\_Initure.Pull=GPIO\_PULLUP;  HAL\_GPIO\_Init(GPIOC,&GPIO\_Initure);    GPIO\_Initure.Pin=GPIO\_PIN\_2|GPIO\_PIN\_3; //PH2,3  HAL\_GPIO\_Init(GPIOH,&GPIO\_Initure);    //中断线0-PA0  HAL\_NVIC\_SetPriority(EXTI0\_IRQn,2,0); //抢占优先级为2，子优先级为0  HAL\_NVIC\_EnableIRQ(EXTI0\_IRQn); //使能中断线0    //中断线2-PH2  HAL\_NVIC\_SetPriority(EXTI2\_IRQn,2,1); //抢占优先级为2，子优先级为1  HAL\_NVIC\_EnableIRQ(EXTI2\_IRQn); //使能中断线2    //中断线3-PH3  HAL\_NVIC\_SetPriority(EXTI3\_IRQn,2,2); //抢占优先级为2，子优先级为2  HAL\_NVIC\_EnableIRQ(EXTI3\_IRQn); //使能中断线2    //中断线13-PC13  HAL\_NVIC\_SetPriority(EXTI15\_10\_IRQn,2,3); //抢占优先级为2，子优先级为3  HAL\_NVIC\_EnableIRQ(EXTI15\_10\_IRQn); //使能中断线13  } |

中断到来时，程序会调用中断服务程序，先进handler，再在handler中调用底下的callback函数。一般来说，不同的中断有不同的中断号，但也有多个中断共享中断号的现象。所以不同的中断一般有不同的handler。最后调用的callback是这个库里所有中断都会调用的一个服务函数，在其中利用switch来判断到底是什么中断。

|  |
| --- |
| //触摸屏中断服务函数  void EXTI9\_5\_IRQHandler(void)//中断号5-9  {  HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_7);//??????????  }  //按键中断服务函数  void EXTI0\_IRQHandler(void)//中断号0  {  HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_0);//调用中断处理公用函数  }  void EXTI2\_IRQHandler(void)//中断号2  {  HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_2);//调用中断处理公用函数  }  void EXTI3\_IRQHandler(void)//中断号3  {  HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_3);//调用中断处理公用函数  }  void EXTI15\_10\_IRQHandler(void)//中断号10-15  {  HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_13);//调用中断处理公用函数  }  //中断服务程序中需要做的事情  //在HAL库中所有的外部中断服务函数都会调用此函数  //GPIO\_Pin:中断引脚号  void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin)  {  char string[16] = {0};//这个是用来打印数据的，用于查看输出以debug  // delay\_ms(100); //消抖  switch(GPIO\_Pin)  {  //按键中断处理  case GPIO\_PIN\_0:  if(WK\_UP==1)  {  }  break;    case GPIO\_PIN\_2:  if(KEY1==0) //key1按下，代表灯的开关状态改变  {  lightFlag=!lightFlag;  }  break;    case GPIO\_PIN\_3:  if(KEY0==0) //窗帘状态改变  {  curtainFlag=!curtainFlag;  }  break;  case GPIO\_PIN\_13:  if(KEY2==0) //煤气安全状态改变  {  co2Flag=!co2Flag;  }  break;    //触摸屏中断处理  case GPIO\_PIN\_7:  if(KEY\_LCD==1)  {  //失败的中断读取，我们尝试一次触摸即读取，可惜失败了  //lcd\_touch(GPIO\_PIN\_7);    //双击  tp\_dev.scan(0);//这个scan是轮询的形式，做不到中断读取  //所以在底下while循环读取屏幕触摸坐标，读到了才退出  while(tp\_dev.x[0]==65535&&tp\_dev.y[0]==65535)tp\_dev.scan(0);  // FT5206\_Scan(0);//效果和上面的代码是一样的  // while(tp\_dev.x[0]==65535&&tp\_dev.y[0]==65535)FT5206\_Scan(0);    // //输出点击的坐标  // Int2String(tp\_dev.x[0],string);  // Show\_Str(30,50,200,16,string,16,0);  // Int2String(tp\_dev.y[0],string);  // Show\_Str(30,75,200,16,string,16,0);  }  break;  }  } |

## 6 课程设计结果



图3 初始界面显示

当用户使用该设备时，设备的显示状态为图3所示。然后当用户按动开发板上的按键时，会改变设备的显示状态，如图4、5所示。

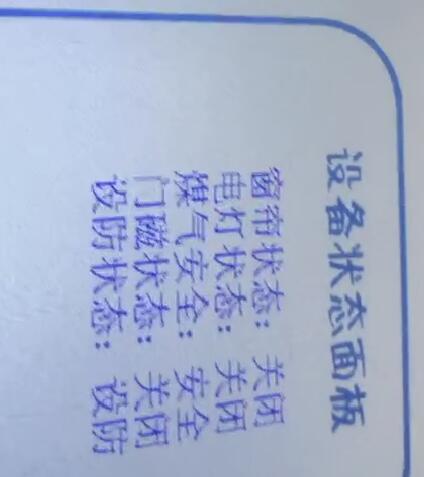
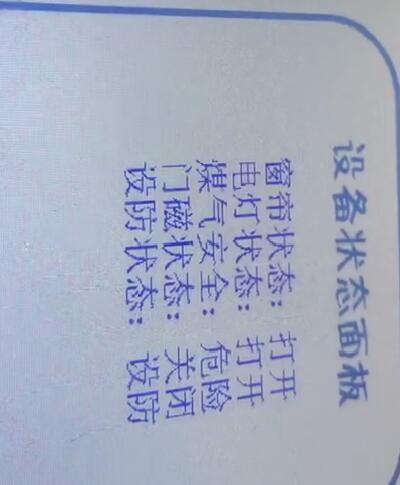
 

图4 窗帘关闭电灯关闭煤气状态安全 图5 窗帘打开电灯打开煤气状态危险

当用户触摸界面上的按钮时，对应设备的状态也会改变，如图6所示。

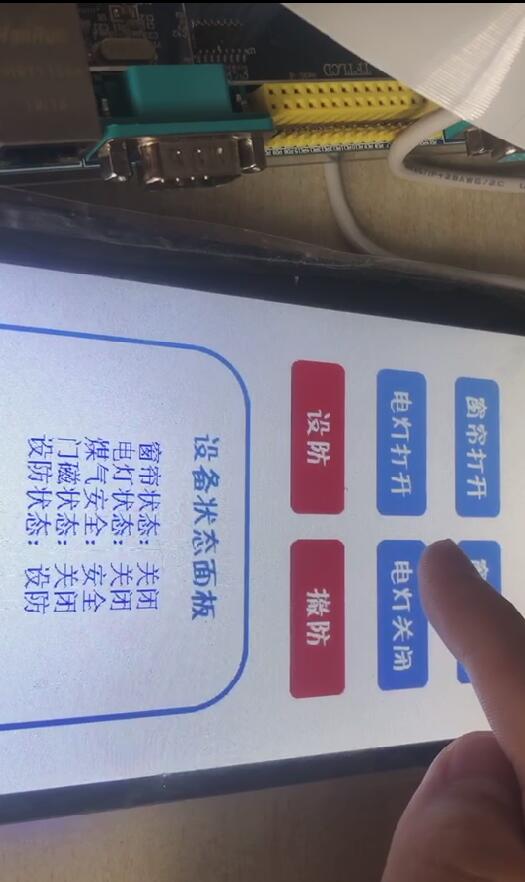


图6 用户触摸触控屏时设备状态改变

当用户点击“电灯关闭”按钮时，设备状态面板中的“电灯状态”就变化为“关闭”状态。

## 7 课设体会

嵌入式系统这门课应该是物联网工程最核心最重要的一门课。与理论课配套的实验课——嵌入式系统课程设计既检验了专业知识也检验了动手能力的，是十分重要的实践类课程。和以往的课程设计不同，这次的课程设计的收获颇丰，从阅读芯片原理图、芯片时序图到软件调试、硬件调试，每一步都走得很艰辛，不过功夫不负有心人，最后的课程设计近乎达到了我们想要的结果，虽然我们在一些细节上还并不满意，我们可以在后续的时间里完善它。

本次嵌入式系统课程设计，我们基于STM32F429芯片制作了一个智能家居控制系统，用户能够通过触碰触控屏控制对应的家居设备，也能通过开发板上的按键控制设备，当设备状态发生改变时，屏幕也能实时显示对应设备的状态变化。此外，在设防状态下，当环境出现危险或异常时，开发板上的扬声器会发生警报。在整个系统中，开发了屏幕显示、字体显示、声音播放、中断响应等功能，使系统更加完备易用。

最难忘的就是在开发系统的过程中出现的各种各样的BUG。嵌入式系统既包括硬件也包括软件，这就需要开发人员拥有硬件调试和软件调试能力。我们在硬件和软件都遇到过很多困难，有些困难甚至花了十几个小时都没有解决，印象最深刻的就是LCD触控屏中断的BUG了。

关于这次实验使用到的中断，我们以前在各种课程上都上过相关内容，但从来没读过真正的代码，也没写过。这次是我们第一次使用中断来处理问题，来更好的解决软件的需求。我们遇到了各种困难，折腾良久终于解决。

首先是中断配置的问题。我们从博客上找到了相关配置，比如说上升沿触发、I2C总线协议和触摸屏中断对应的中断号。经过配置这些东西我才感觉到自己和底层的硬件以及协议栈有了直接的联系，I2C总线的协议原来是一行一行实现的。而触摸屏中断号的寻找让我明白了中断工作的机制，ARM架构的统一编址，硬件厂商提供的说明书的阅读方法。

然后是硬件的问题。我们在配置完成后迟迟无法进入中断，尝试各种方法无果。直到我遍历各个文档，才发现有提到说一个叫做C32的电容会将中断的脉冲信号过滤掉。我们在老师的帮助下拿到了设备并将电容焊了下来，这才最终成功的进入中断。

最后是中断服务程序的问题。我们使用的触摸屏中断服务程序来自于示例代码，而示例代码采用的是轮询的形式，这就导致即使是照原样抄下来再修改也无法成功的在一次触摸的情况下就成功采集到触摸坐标。这个问题我们想了很久最终也没能成功解决，只是采用了连续两次点击屏幕的方法来采集触摸坐标。第一次进入中断，第二次采集坐标。希望之后在这点上能有更好的解决方案。