项目名称：云上家居系统

本项目由 MeJinzejun 主导完成，GitHub： MeJinzejun (github.com)

微信小程序端参考 b 站用户 【【挽救小白第一季】STM32+8266+小程序智能家居毕设实战】 https://www.bilibili.com/video/BV1ae411W7yD?vd_source=42bb0cca1151b5ec11813aae5e75a6ee

非常感谢。

目前控制了一系列 TMF 开发板连接的设备如风扇台灯舵机电机等。另外还有智能晾衣架项目没有接入云平台。项目可为设备接入阿里云提供一定参考价值。

目　录

第 1 章 需求分析

1.1 作品创意来源与产生背景

随着社会经济水平与科技水平的飞速发展，人们的生活质量不断提高，人们对于生活品质的要求也越来越高。然而目前的智能家居系统不同厂商生产的家电型号不同导致不能统一进行智能化的控制，同时对于家居系统智能化的需求日益强烈，智能家居系统对于人们生活质量具有重要的意义，所以设计一个智能化、安全性高、实用性强的家居系统是一个重要的研究方向。

1.2 作品的用户群体

本作品是一个智能化的云上家居系统。针对家庭用户对于智能化家居设备的需求我们提供基于阿里云和微信小程序的云上家居控制系统进行智能联网采集环境数据并进行智能化控制家电设备，同时针对学校、工厂、养殖厂等需要大批量进行智慧控制的单位提供私人化定制服务进行大量远程的智能化控制。

1.3 作品的主要功能与特色

本作品是一个基于 TMF6200 开发板、阿里云平台和微信小程序的智能化云上家居系统。主要光照、温湿度、人体红外、雨滴等环境信息的采集功能，LCD 屏幕实时显示功能，数据上传阿里云平台存储功能，微信小程序实时显示数据功能，系统智能控制运行功能，按钮触摸设置系统，微信小程序远程控制。信息采集功能主要通过温湿度传感器、光敏传感器、人体红外传感器、雨滴传感器完成数据的采集，LCD 屏幕显示功能将采集的数据实时显示在 LCD 屏幕上，数据上传功能通过 ESP82600 模块连接网络，然后通过消息转发接收数据，微信小程序功能实现系统的远程控制实时的显示环境的数据信息，通过下发指令控制云上家居系统的运行。系统设计运行模式可自动感应环境的变化情况，进行智能化的自动控制，同时系统设置手动控制按钮。

1.4 作品的应用价值与推广前景

2022 年 8 月 8 日，工业和信息化部、住房和城乡建设部、商务部、市场监管总局近日联合发布《推进家居产业高质量发展行动方案》[1]，到 2025 年，在家用电器、照明电器等行业培育制造业创新中心、数字化转型促进中心等创新平台，重点行业两化融合水平达到 65%，培育一批 5G 全连接工厂、智能制造示范工厂和优秀应用场景。反向定制、全屋定制、场景化集成定制等个性化定制比例稳步提高，绿色、智能、健康产品供给明显增加，智能家居等新业态加快发展。由此可见，云上家居项目的研究工作在现实中具有重要的现实意义。以住宅为平台，融合建筑、网络通信、智能家居设备、服务平台，集系统、服务、管理为一体，其目的是为用户提供高效、舒适、安全、便利的居住环境。智能家居作为物联网技术在家庭环境中的典型应用，是将物联网技术融入传统家居系统的成果。

1.5 作品的性能指标

作品功能包含光照、温湿度、人体红外、雨滴等环境信息的采集功能，LCD 屏幕实时显示功能，数据上传阿里云平台存储功能，微信小程序实时显示数据功能，系统智能控制运行功能，按钮触摸设置系统，微信小程序远程控制。主要从系统启动时间、电机控制、照明灯控制、舵机控制、电风扇控制、微信小程序数据显示、远程控制、触摸按钮控制、云平台数据存储等方面分析系统的性能指标。

1.6 作品的创新性

微信小程序设计，本作品设计简介实用的微信小程序便于远程控制，以及设备联网等。硬件终端设计，本作品以 TMF6200 开发板作为核心控制主板结合温湿度传感器、光敏传感器、人体红外传感器、雨滴传感器完成数据的采集，并且通过 ESP8266 实现联网将数据传输至阿里云服务器端，进而转发至微信小程序。

云上家居系统，设计自动运行模式实现智能化的控制。当天黑时，自动关闭窗帘，自动打开照明灯。当天亮时，自动打开窗帘，自动关闭照明灯。当湿度值高于阈值时自动打开电风扇。通过舵机舵机实现开门和关门控制。

第 2 章 技术方案

2.1 相关技术原理

2.1.1 微信小程序

微信小程序[2]是一种不需要下载安装就能使用的小程序，用户通过搜索或扫 描二维码即可打开应用。从字面上来看“微信小程序”这个词，“微信”标注了其运 行环境，“小程序”说明它是具备轻便特征的程序。除此之外，小程序的使用具有 以下特色，一、可以直接在微信进行搜索，不用下载安装；二、使用完成之后 可以快速退出，不需要对应的卸载处理。 微信小程序使用 JSON（JavaScript Object Notation） 技术来表现应用的配 置信息。一般来说需要进行页面配置和路由，还有应用的基本信息等等。 微信小程序通过定制 WXML+WCSS 技术来实现页面视图。其中包括按钮、文本列表、文字处理框、条件选择框等形象化元素都是通过 WXML 语言来表达。按照 WXML 的语法格式来编写代码；如果页面具有相同的风格，则可以使用 WXSS 进行设置。微信小程序使用 JavaScript 语言来实现逻辑层结构，通过在视图层和逻辑 层之间提供数据和事件传输功能，可以方便的处理包括用户操作和系统 API 调 用等事件。一般来说，使用微信小程序的应用功能都比较单一，所以对架构的 要求也比较低[3]。 除此之外，微信小程序使用 JSON 来进行数据交换，JSON 数据格式简明易懂，适合用户阅读和编写，也方便机器解析和生成，能够有效地提升网络传输效率。

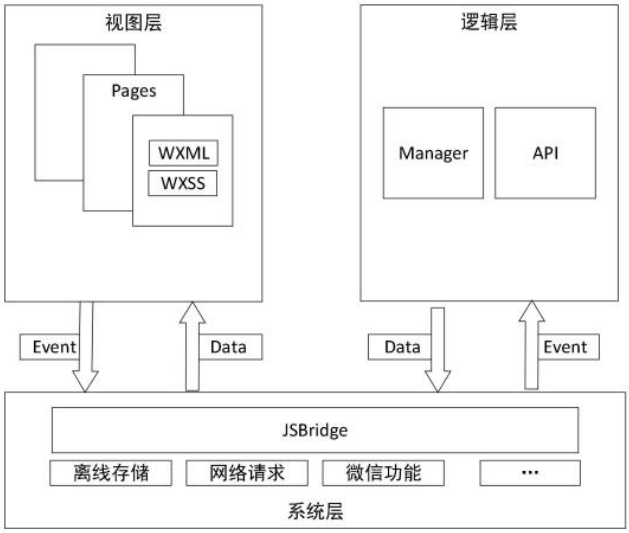


图 1 微信小程序框架架构图

2.1.2 阿里云服务器

2018 年第四季度的中国云市场份额排名前 5 的云平台分别为阿里云、 腾讯云、AWS、中国电信和中国联通。阿里云凭借其独到的眼光，抢先于国内其 他巨头公司进入云计算领域，并且凭借其自主品牌的身份，占据国内云计算市场 的大量份额。同样，阿里云[4]在性能和服务上也有很大的优势：（1）稳定。在国内拥有 2300 个服务器节点，海外拥有 500 多个节点，保障 服务器网络负载均衡。此外，阿里云提供“异地双活”、“两地三中心”的灾备 解决方案，当服务器宕机或系统因意外停止工作时，系统、应用、数据等可自动 迁移，继续对外提供服务。(2)安全。阿里具有世界顶尖的防卫系统，同时阿里云还配备防 DDoS 系统、 安全组规则保护等，能够绝对保护用户服务器安全。（3）配置灵活。阿里云弹性 ECS 可按需垂直调整实例配置、水平调整实例 数量，轻松做到灵活、高效、弹性、节约，提供了 10 大类、前后 4 代共近 200 种 计算实例类型，4 类 8 种存储类型供选择，全面满足用户在功能、成本与性能的 均衡考虑。（4）服务好。继承了万网的服务能力，提供免费的备案服务和网站搬迁服务， 同时提供 24 小时专人专线服务，能够很好的帮助用户解决问题。（5）性价比高。相较于传统 IDC（Internet Data Center），阿里云可节省 80% 成本投入，同时维护费用极低，一次性投入最低可至几百块钱即可享受高效的服务。

图 2 阿里云物联网平台架构图

2.1.3 TMF6200 开发板

Tai-Action Phoenix SDK 是珠海泰为电子有限公司[5]提供的一套性能强大的开发工具和嵌入式软件模块，能够让开发人员在 TMF6200 平台上快速、轻松地开发应用。TMF6200 是一款集成度高、性能卓越的工业实时控制处理器。TMF6200 内置一个通用的 32 位 Cortex-M3 核，实现了出色的计算性能和对异常能力的快速响应。单芯片即可满足电力电子相关领域控制系统的应用需求。内置增强型可重构配置单元（Enhanced Reconfigurable Processing Unit）、多通道的高分辨率 ADC、预处理模拟信号前端电路并且支持多种 PWM 模式。TMF6200 器件采用 32 位多层总线结构，该结构使得系统中的多个主机和从机之间的并行通信成为可能。多层总线结构包括一个 AHB 互联矩阵、两个 AHB 总线和两个 APB 总线。

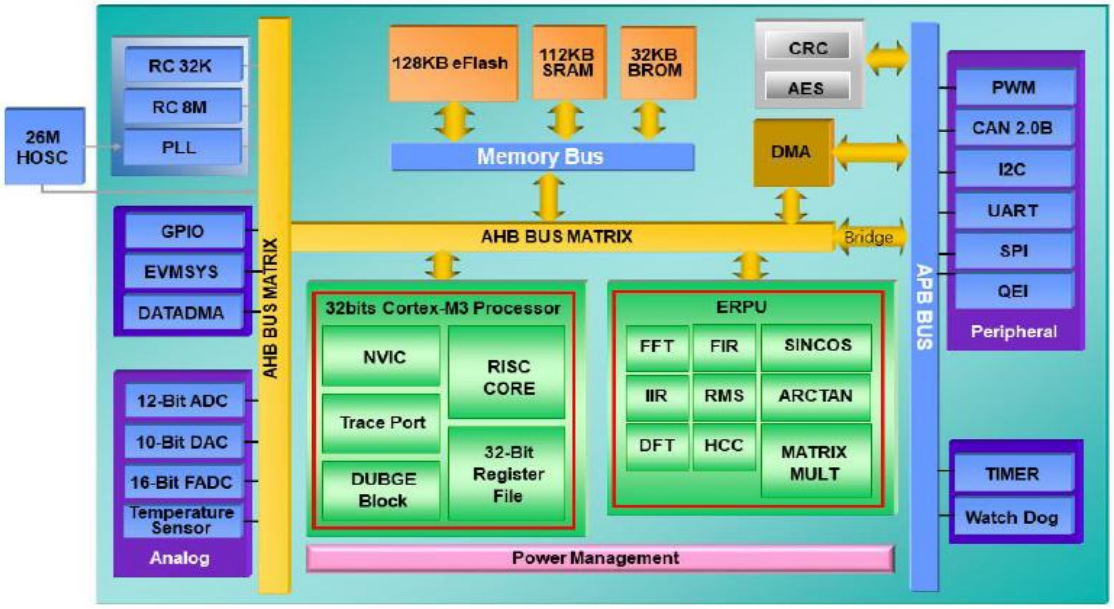


图 3 TF6200 系统架构图

在 51 单片机的开发中多数采用的开发方式是直接操作寄存器，比如要控制 P0 口的状态，直接操作寄存器：P0=0x11；而在 TMF6200 的开发中，同样可以操作寄存器：GPIOB->BSRS=00000001；但这种方法的劣势是需要去掌握每个寄存器的语法，方能正确使用 TMF6200，而对于类似 TMF6200 这种内核为 Cortex 级别的 MCU，数百个寄存器记起来非常空难。于是珠海泰为电子有限公司推出了 Tai-Action Phoenix SDK，Tai-Action Phoenix SDK 将这些寄存器底层操作都封装起来，提供一整套接口(API)供开发者调用，大多数场合下，不需要去知道操作的是哪个寄存器，只需要知道调用哪些函数即可。

2.2 系统架构设计

2.2.1 系统总体架构

基于阿里云的云上家居物联网系统整体上包括数据采集模块、屏幕显示模块、系统控制模块、网络传输模块、云服务器模块、客服端微信小程序模块组成。数据采集模块由温湿度传感器、光敏传感器、人体红外传感器、雨滴传感器组成实现对光照、温湿度、人体红外、雨滴等环境信息的采集功能。屏幕显示模块由OV7725LCD 显示屏完成数据显示功能。系统控制模块由触摸开关、继电器、42步进电机、舵机、电风扇和台灯组成实现云上家居系统的智能控制功能。网络传输模块由 ESP8266 芯片完成连接 WIFI 进行数据传输的功能。云服务器模块负责接受设备传输过来的数据信息并且将数据存储在云端。客服端微信小程序模块通过使用微信小程序开发者工具制作完成，用户通过微信小程序可以远程控制云上家居系统，并且实时接受云上家居系统采集的数据，显示于屏幕。云上家居系统架构图如图 4 所示。

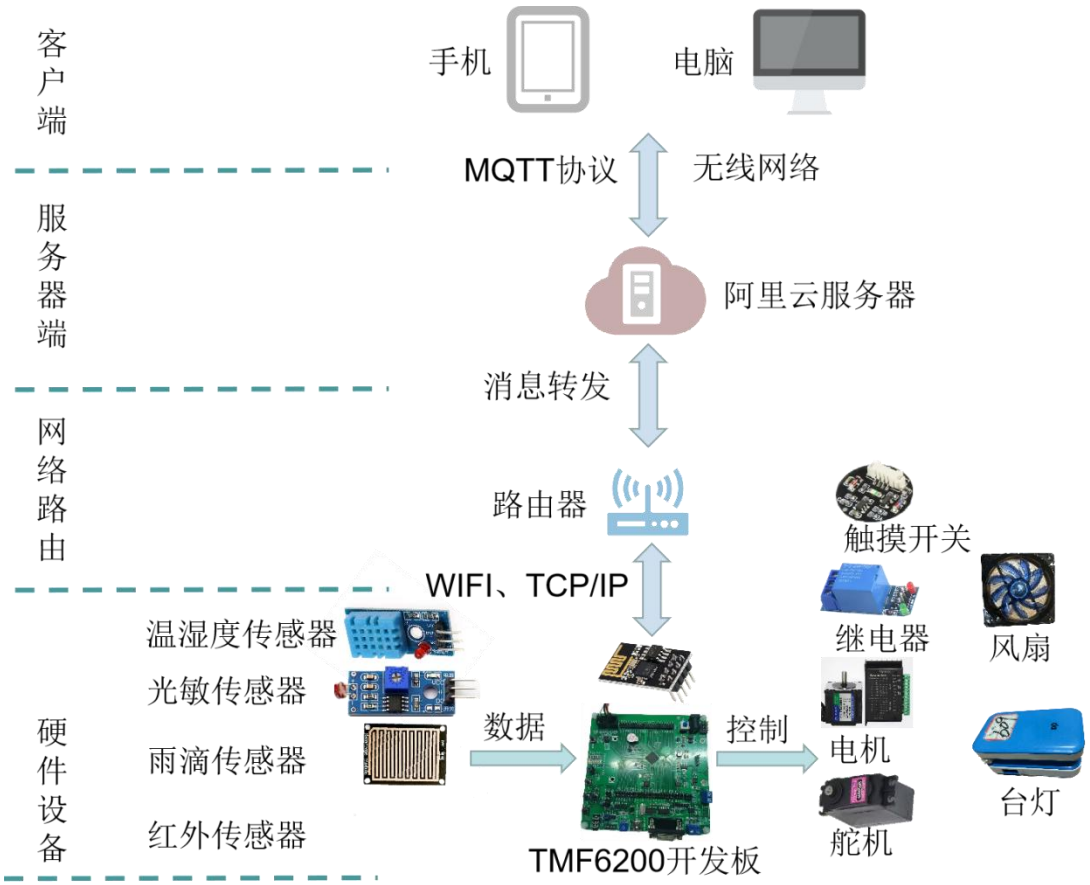


图 4 云上家居系统架构图

2.2.2 系统功能设计

云上家居系统有光照、温湿度、人体红外、雨滴等环境信息的采集功能，LCD 屏幕实时显示功能，数据上传阿里云平台存储功能，微信小程序实时显示数据功能，系统智能控制运行功能，按钮触摸设置系统，微信小程序远程控制。如图 5 所示云上家居系统功能设计图。

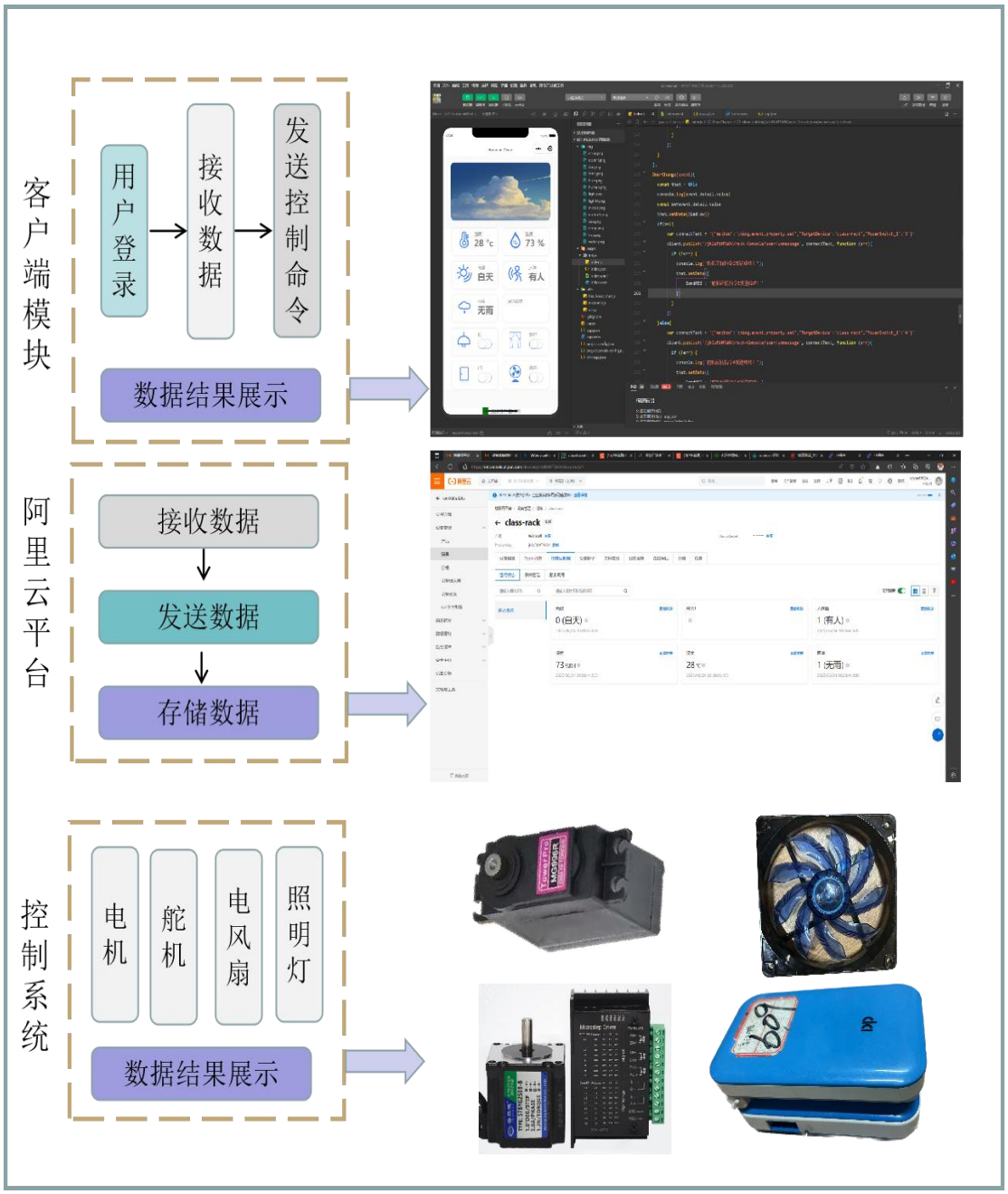


图 5 云上家居系统功能设计图

2.3 数据采集模块

数据采集模块由温湿度传感器、光敏传感器、人体红外传感器、雨滴传感器组成实现对光照、温湿度、人体红外、雨滴等环境信息的采集功能。传感器如图5所示。

2.3.1 DHT11 温湿度传感器

DHT11 温湿度模块它应用专用的数字模块采集技术和温湿度传感技术，确保产品具有可靠的稳定性，响应快，抗干扰能力强。

2.3.2 人体红外传感器

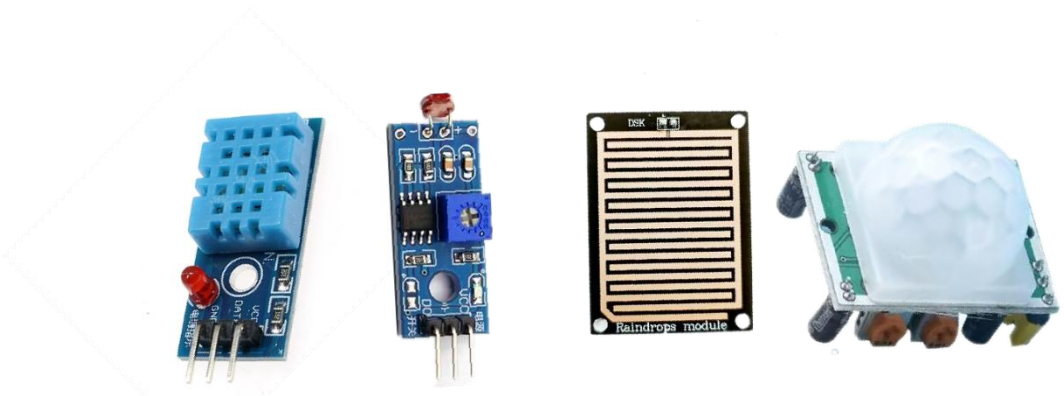
HC-SR501 人体红外传感器模块，它主要依靠特定温度（36-38℃）的物体运动来判断是否是人体。它有两个调节旋钮，一个调节最远探测距离，一个调节延迟时间。当人走过或停留在感应范围中，模块通过 D0 发送高电平信号。

2.3.3 雨滴传感器

雨滴传感器用于检测环境是否下雨，感应板上没有水滴时，DO 输出为高电平，感应板上有水滴时，DO 输出为低电平。

2.3.4 光敏传感器

光敏传感器是利用光敏元件将光信号转换为电信号的传感器，它的敏感波长在可见光波长附近，包括红外线波长和紫外线波长。



温湿度传感器　光敏传感器　雨滴传感器　人体红外传感器

图 5 数据采集传感器

2.4 屏幕显示模块

2.4.1 TFT 显示屏

TFT(Thin Film Transistor) 即薄膜场效应晶体管，它可以“主动地”对屏幕上的各个独立的像素进行控制，这样可以大大提高反应时间。一般 TFT 的反应时间比较快，约 80 毫秒，而且可视角度大，一般可达到 130 度左右，主要运用在高端产品。从而可以做到高速度、高亮度、高对比度显示屏幕信息。TFT 属于有源矩阵液晶显示器，在技术上采用了“主动式矩阵”的方式来驱动，方法是利用薄膜技术所作成的电晶体电极，利用扫描的方法“主动拉”控制任意一个显示点的开与关，借助液晶分子传导光线，通过遮光和透光来达到显示的目的。TFT 型的液晶显示器主要的构成包括：萤光管、导光板、偏光板、滤光板、玻璃基板、液晶材料、薄模式晶体管等等。TFT 显示屏硬件原理图如图 6 所示。

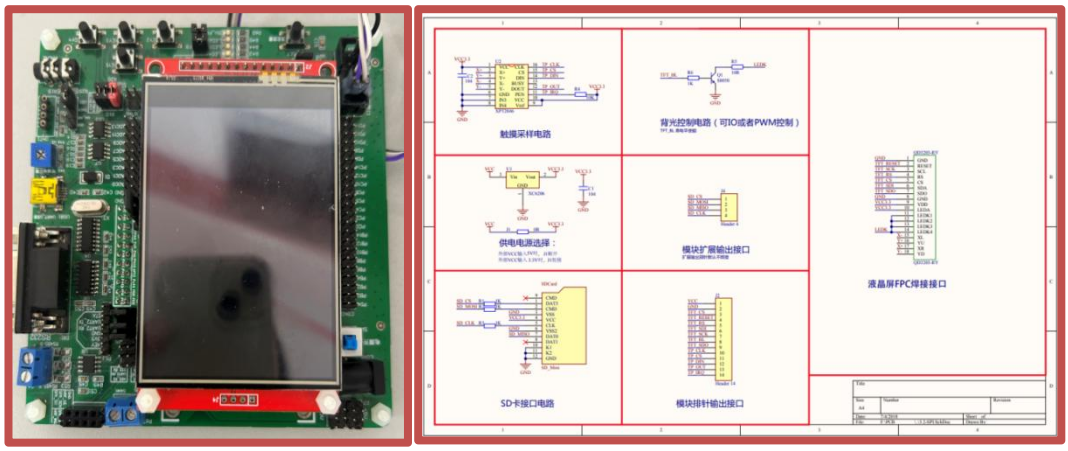


图 6 TFT 显示屏硬件原理图

2.4.2 屏幕显示核心程序

屏幕的文字显示定义于 lcd_show_shuju(void)函数，包括在页面的顶部显示系统的名字，在页面底部显示我们系统设计人员的名字。页面的中间部分是用于数据显示和指令显示的区域，首先设计温度显示于页面的（10, 80）位置，湿度显示于（10, 96），人体值显示于（10, 112），光敏值显示于（10, 128），雨滴值显示于（10, 144）。TFT 显示数据的时候对于字符类型使用 LCD_ShowChar()函数，如果是字符串类型使用 LCD_ShowString()，如果是数字使用 LCD_ShowNum()函数，显示中文字符使用中文字符函数 GUI_DrawFont16()。

屏幕显示核心程序

```
void lcd_show_shuju(void){
    //顶部显示
    LCD_ShowString(0, 0, 12, "================================", 0);
    GUI_DrawFont16(84, 0, POINT_COLOR, WHITE, "飞",0);
    GUI_DrawFont16(100, 0, POINT_COLOR, WHITE, "天",0);
    GUI_DrawFont16(116, 0, POINT_COLOR, WHITE, "系",0);
    GUI_DrawFont16(132, 0, POINT_COLOR, WHITE, "统",0);
    //底部显示
    LCD_ShowString(0, 264, 12, "================================", 0);
    LCD_ShowString(0, 280, 12, "================================", 0);
    GUI_DrawFont16(84, 280, POINT_COLOR, WHITE, "金",0);
    GUI_DrawFont16(100, 280, POINT_COLOR, WHITE, "鹏",0);
    GUI_DrawFont16(116, 280, POINT_COLOR, WHITE, "公",0);
    GUI_DrawFont16(132, 280, POINT_COLOR, WHITE, "司",0);
    //数据显示
    LCD_ShowString(0, 64, 12, "++++++++++++++++++++++++++++++++++++++++", 0);
    LCD_ShowString(54, 64, 12, "数据展示区域", 0);
    GUI_DrawFont16(10, 80, POINT_COLOR, BACK_COLOR, "温",1);
    GUI_DrawFont16(26, 80, POINT_COLOR, BACK_COLOR, "度",1);
    LCD_ShowString(42, 80, 12, ":", 0);
    GUI_DrawFont16(90, 80, POINT_COLOR, BACK_COLOR, "℃",1);
    GUI_DrawFont16(10, 96, POINT_COLOR, BACK_COLOR, "湿",1);
    GUI_DrawFont16(26, 96, POINT_COLOR, BACK_COLOR, "度",1);
    LCD_ShowString(42, 96, 12, ":", 0);
    GUI_DrawFont16(90, 96, POINT_COLOR, BACK_COLOR, "℃",1);
    GUI_DrawFont16(0, 112, POINT_COLOR, BACK_COLOR, "人",1);
    GUI_DrawFont16(26, 112, POINT_COLOR, BACK_COLOR, "体",1);
    GUI_DrawFont16(42, 112, POINT_COLOR, BACK_COLOR, "值",1);
    LCD_ShowString(58, 112, 12, ":", 0);
    GUI_DrawFont16(0, 128, POINT_COLOR, BACK_COLOR, "光",1);
    GUI_DrawFont16(26, 128, POINT_COLOR, BACK_COLOR, "敏",1);
    GUI_DrawFont16(42, 128, POINT_COLOR, BACK_COLOR, "值",1);
    LCD_ShowString(58, 128, 12, ":", 0);
    GUI_DrawFont16(0, 144, POINT_COLOR, BACK_COLOR, "雨",1);
    GUI_DrawFont16(26, 144, POINT_COLOR, BACK_COLOR, "滴",1);
    GUI_DrawFont16(42, 144, POINT_COLOR, BACK_COLOR, "值",1);
    LCD_ShowString(58, 144, 12, ":", 0);
    //控制命令显示
    LCD_ShowString(0, 160, 12, "++++++++++++++++++++++++++++++++++++++++", 0);
    LCD_ShowString(54, 160, 12, "控制命令区域", 0);
}
```

2.4.3 显示中文字符程序

屏幕显示中文核心程序

```
void GUI_DrawFont16(u16 x, u16 y, u16 fc, u16 bc, u8 *s,u8 mode)
{
   u8 i,j;
   u16 k;
   u16 HZnum;
   u16 x0=x;
   HZnum=sizeof(tfont16)/sizeof(typFNT_GB16); //自动统计汉字数目
   for (k=0;k<HZnum;k++) {
       if((tfont16[k].Index[0]==*(s))&&(tfont16[k].Index[1]==*(s+1))){
           LCD_SetWindows(x,y,x+16-1,y+16-1);
       for(i=0;i<16*2;i++){
        for(j=0;j<8;j++){
             if(!mode){//非叠加模式：字体带有背景色，显示时会将原来
              if(tfont16[k].Msk[i]&(0x80>>j)){
                              Lcd_WriteData_16Bit(fc);
                         }else {
                              Lcd_WriteData_16Bit(bc);
              }
             }else{//叠加模式：字体不带背景色，直接叠加显示到原来显示的内容上
              POINT_COLOR=fc;
              if(tfont16[k].Msk[i]&(0x80>>j)) {
                    LCD_DrawPoint(x,y);//画一个点
              }
              x++;
              if((x-x0)==16){
                   x=x0;
                   y++;
                   break;
              }
             }
            }
        }
     }
     continue;  //找到对应点阵字库立即退出，防止多个汉字重复取模带来影响
    }
    LCD_SetWindows(0,0,lcddev.width-1,lcddev.height-1);//恢复窗口为全屏
}
```

中文数组结构体--节选部分

```
typedef struct
{
       unsigned char Index[2];   //存放汉字 GBK 码
       char Msk[32]; //存放汉字取模数据
}typFNT_GB16;  //结构体名称可以自己定义
const typFNT_GB16 tfont16[]=
{
"温",0x00,0x00,0x23,0xF8,0x12,0x08,0x12,0x08,0x83,0xF8,0x42,0x08,0x42,0x08,0x13,0xF8,
0x10,0x00,0x27,0xFC,0xE4,0xA4,0x24,0xA4,0x24,0xA4,0x24,0xA4,0x2F,0xFE,0x00,0x00,
"度",0x01,0x00,0x00,0x80,0x3F,0xFE,0x22,0x20,0x22,0x20,0x3F,0xFC,0x22,0x20,0x22,0x20,
0x23,0xE0,0x20,0x00,0x2F,0xF0,0x24,0x10,0x42,0x20,0x41,0xC0,0x86,0x30,0x38,0x0E,
"湿",0x00,0x00,0x27,0xF8,0x14,0x08,0x14,0x08,0x87,0xF8,0x44,0x08,0x44,0x08,0x17,0xF8,
0x11,0x20,0x21,0x20,0xE9,0x24,0x25,0x28,0x23,0x30,0x21,0x20,0x2F,0xFE,0x00,0x00,
"人",0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x02,0x80,0x02,0x80,
0x04,0x40,0x04,0x40,0x08,0x20,0x08,0x20,0x10,0x10,0x20,0x08,0x40,0x04,0x80,0x02,
"体",0x08,0x40,0x08,0x40,0x08,0x40,0x10,0x40,0x17,0xFC,0x30,0x40,0x30,0xE0,0x50,0xE0,
0x91,0x50,0x11,0x50,0x12,0x48,0x15,0xF4,0x18,0x42,0x10,0x40,0x10,0x40,0x10,0x40,
"值",0x08,0x40,0x08,0x40,0x0F,0xFC,0x10,0x40,0x10,0x40,0x33,0xF8,0x32,0x08,0x53,0xF8,
0x92,0x08,0x13,0xF8,0x12,0x08,0x13,0xF8,0x12,0x08,0x12,0x08,0x1F,0xFE,0x10,0x00,
"紫",0x08,0x80,0x28,0x88,0x2E,0xF0,0x28,0x84,0x2E,0x84,0xF0,0x7C,0x02,0x00,0x04,0x20,
0x1F,0xC0,0x01,0x80,0x06,0x10,0x3F,0xF8,0x01,0x08,0x11,0x20,0x25,0x10,0x42,0x08,
"外",0x10,0x40,0x10,0x40,0x10,0x40,0x10,0x40,0x3E,0x40,0x22,0x60,0x42,0x50,0x42,0x48,
0xA4,0x44,0x14,0x44,0x08,0x40,0x08,0x40,0x10,0x40,0x20,0x40,0x40,0x40,0x80,0x40,
"线",0x10,0x50,0x10,0x48,0x20,0x40,0x24,0x5C,0x45,0xE0,0xF8,0x40,0x10,0x5E,0x23,0xE0,
0x40,0x44,0xFC,0x48,0x40,0x30,0x00,0x22,0x1C,0x52,0xE0,0x8A,0x43,0x06,0x00,0x02,
"金",0x01,0x00,0x01,0x00,0x02,0x80,0x04,0x40,0x08,0x20,0x10,0x10,0x2F,0xE8,0xC1,0x06,
0x01,0x00,0x3F,0xF8,0x01,0x00,0x11,0x10,0x09,0x10,0x09,0x20,0xFF,0xFE,0x00,0x00,
"鹏",0x00,0x10,0x77,0x20,0x55,0x7C,0x55,0x44,0x55,0x64,0x77,0x54,0x55,0x44,0x55,0x4C,
0x55,0x40,0x77,0x7E,0x55,0x02,0x55,0x02,0x55,0x7A,0xB5,0x02,0x89,0x0A,0x13,0x04,
"统",0x10,0x40,0x10,0x20,0x20,0x20,0x23,0xFE,0x48,0x40,0xF8,0x88,0x11,0x04,0x23,0xFE,
0x40,0x92,0xF8,0x90,0x40,0x90,0x00,0x90,0x19,0x12,0xE1,0x12,0x42,0x0E,0x04,0x00,
"光",0x01,0x00,0x21,0x08,0x11,0x08,0x09,0x10,0x09,0x20,0x01,0x00,0xFF,0xFE,0x04,0x40,
0x04,0x40,0x04,0x40,0x04,0x40,0x08,0x42,0x08,0x42,0x10,0x42,0x20,0x3E,0xC0,0x00,
"敏",0x40,0x20,0x40,0x20,0x7F,0x20,0x80,0x3E,0x7E,0x44,0x42,0x44,0x52,0x44,0x4A,0xA4,
0xFF,0xA8,0x42,0x28,0x92,0x10,0x8A,0x10,0xFF,0x28,0x02,0x48,0x14,0x84,0x09,0x02,
"雨",0x00,0x00,0xFF,0xFE,0x01,0x00,0x01,0x00,0x01,0x00,0x7F,0xFC,0x41,0x04,0x41,0x04,
0x49,0x44,0x45,0x24,0x41,0x04,0x49,0x44,0x45,0x24,0x41,0x04,0x41,0x14,0x40,0x08,
"滴",0x00,0x80,0x20,0x40,0x17,0xFC,0x11,0x10,0x80,0xA0,0x47,0xFC,0x44,0x44,0x15,0xF4,
0x14,0x44,0x25,0xF4,0xE5,0x14,0x25,0x14,0x25,0xF4,0x24,0x04,0x24,0x14,0x04,0x08,
};
```

2.5 电机、舵机模块

2.5.1 电机控制原理

步进电机是一种用电脉冲进行控制，将电脉冲(数字信号)转化为角位移的执行机构。在非超载的情况下，电机的转速、停止的位置只取决于脉冲信号的频率和脉冲数，而不受负载变化的影响，当步进驱动器接收到一个脉冲信号，它就驱动步进电机按设定的方向转动一个固定的角度，称为“步距角”，它的旋转是以固定的角度一步一步运行的。可以通过控制脉冲个数来控制角位移量，从而达到准确定位的目的；同时可以通过控制脉冲频率来控制电机转动的速度和加速度，从而达到调速的目的。每输入一个脉冲信号，步进电机前进一步。步进电机旋转的步距角，是在电机结构的基础上等比例控制产生的，如果控制电路的细分控制不变，那么步进旋转的步距角在理论上是一个固定的角度。在本次项目中我们使用 TB6200 驱动板驱动 42 步进电机进行工作。TB6200 驱动板和 42 步进电机如图 7 所示。

图 7 TB6200 驱动板和 42 步进电机

42 步进电机为两相四线电机，相数是指线圈内部的线圈组数。控制信号有六根脚，总共 3 组信号。ENA 代表 Enable（使能），控制步进电机的通电与否（步进电机通电会锁死，断电可以自由旋转）；DIR 代表 Direction（方向），控制步进电机的旋转方向。PUL 代表 Pulse（脉冲，有些驱动器会写 PLS），控制步进电机的旋转步数。步距角： 输入一个电脉冲信号，步进电动机转子相应的角位移。它与控制绕组的相数、转子齿数和通电方式有关。步距角越小，运转的平稳性越好。此电机步矩角：1.8°，故转 1 圈需要 脉冲数 ＝360° / 1.8° ＝200 个。细分倍数： 把电机步距角微分，减小每个脉冲走相对应得角位移，即减小脉冲当量，使得电机运行更平稳，在脉冲频率不变情况下，速度也就变慢了。比如当细分倍数为 2 时，转 1 圈需要 脉冲数 ＝360° /（1.8°/2）＝400 个。 脉冲当量 ＝1.8°/2＝0.9°

拨码开关的 S1、S2、S3 是用来设置细分数的，可以设置细分数 200、400、800、1600 等，设置为 200 的话，给一个脉冲是转 1.8°，设置为 400 的话，给一个脉冲是转 0.9°，为了演示方便，这里选择 800 细分，给一个脉冲是转 0.45 度，因此同样频率的脉冲信号可以使得步进电机的转速更低。拨码开关的 S4、S5、S6 是用来调电流大小的，我用的步进电机的额定电流为 1.5A，因此设置 S4-ON、S5-ON、S6-OFF。

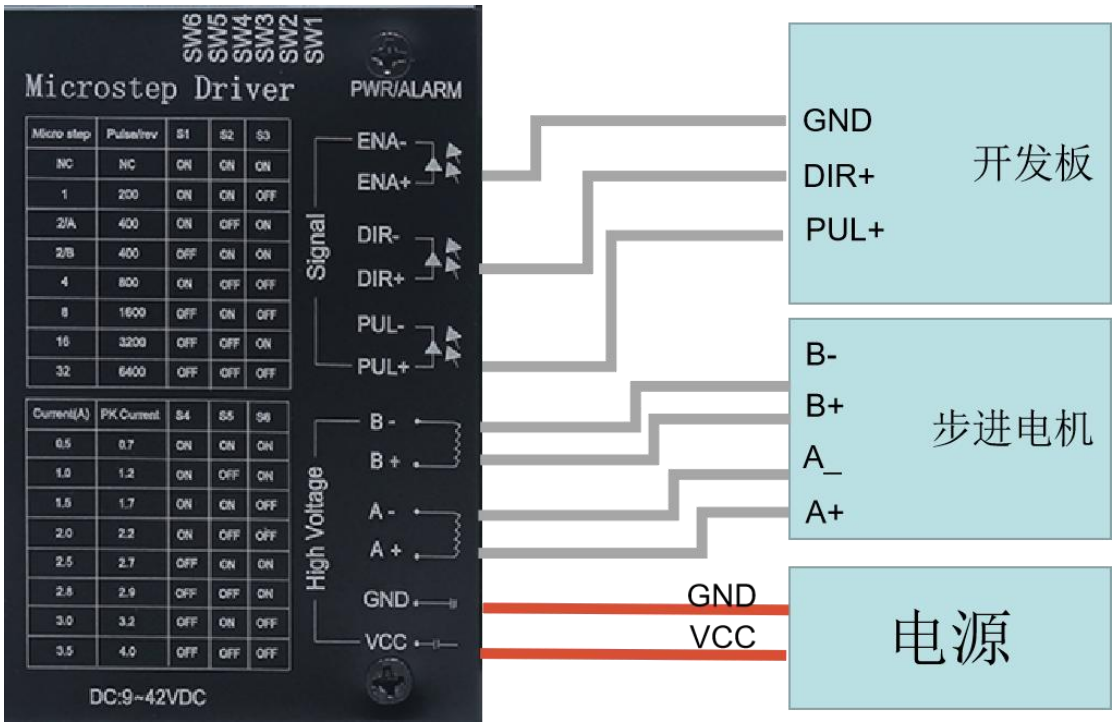


图 8 TB6200 和 42 步进电机接线图

2.5.2 电机控制核心程序

电机控制核心程序

```
void dianjizheng(int quanshu){
    for(int i=0;i<(quanshu*6400);i++){
        ll_gpio_bit_set(GPIOC, BIT(0));
        ll_gpio_bit_set(GPIOC, BIT(2));
        delay_us(10);
        ll_gpio_bit_reset(GPIOC, BIT(0));
        delay_us(10);
    }
}
void dianjifan(int quanshu){
    for(int i=0;i<(quanshu*6400);i++){
        ll_gpio_bit_set(GPIOC, BIT(0));
        ll_gpio_bit_reset(GPIOC, BIT(2));
        delay_us(10);
        ll_gpio_bit_reset(GPIOC, BIT(0));
        delay_us(10);
    }
}
```

2.5.3 舵机控制原理

舵机的输入有三根线，一般的中间的红色线为电源正极，咖啡色线的为电源负极，黄色色线为控制线号线。舵机的控制通常采用 PWM 信号，例如需要一个周期为 20ms 的脉冲宽度调制（PWM），脉冲宽度部分一般为 0.5ms-2.5ms 范围内的角度控制脉冲部分，总间隔为 2ms。当脉冲宽度为 1.5ms 时，舵机旋转至中间角度，大于 1.5ms 时舵机旋转角度增大，小于 1.5ms 时舵机旋转角度减小。舵机分 90°、180°、270°和 360°舵机。

模拟舵机内部的控制驱动电路板从外界接收控制信号，经过处理后变为一个直流偏置电压，在控制板内部有一个基准电压，这个基准电压由电位器产生,将外部获得的直流偏置电压与电位器的电压进行比较获得电压差，并输出到电机驱动芯片驱动电机，电压差的正负决定电机的正反转，大小决定旋转的角度，电压差为 0 时，电机停止转动。控制电路向电机传送动力脉冲，电机通过级联减速齿轮带动电位器旋转，当电压差为 0 时，电机停止转动。舵机的输出功率与它所需要转动的距离成正比。如果输出轴需要转动很长的距离，控制电路就会发送最

大宽度的动力脉冲使马达全速运转；如果它只需要短距离转动，马达就会以较慢的速度运行（速度比例控制），这意味着如果有一个比较小的控制动作，舵机就会发送很小的初始动力脉冲到马达，马达的反应会非常迟钝或者根本就没有反应（模拟电机缺陷）。当到达指定的位置，控制电路发送的动力脉冲就会减小脉冲宽度。

本次使用 MG996R 舵机，它的性能强于其他舵机，适用于大多数情况，但是体积也增大许多。转动角度有 180°/360°。常用性能参数：尺寸:40.8* 20* 38mm；重量:55g；无负载速度:4.8V@0.20sec/60°（4.8V）or 0.19sec/60°（6V）；扭矩:13kg/cm（4.8V） or 15kg/cm（6V）；电压:4.8V-7.2V；响应脉宽时间： ≤ 5usec；角度偏差：回中误差 0 度，左右各 45°误差≤3°；接口规格：JR/FUTABA 通用；连接线长度： 300mm。

2.5.4 舵机控制核心程序

舵机控制核心程序

```
void Control_SG90(uint32_t us)
{
//500us,0 度
//1000us,45 度
//1500us,90 度
//2000us,145 度
//2500us,180 度
    int i=0;
    for(i=0;i<10;i++)
    {
        if(us<=20000)
        {
            ll_gpio_bit_set(GPIOB, BIT(6));
            delay_us(us);
            ll_gpio_bit_reset(GPIOB, BIT(6));
            delay_us(20000-us);
        }
    }
}
```

2.6 照明、风扇模块

2.6.1 照明灯

本作品使用台灯作为云上智能家居系统的照明灯，模拟真实环境下的系统运行情况，照明灯如图 9 所示。

图 9 照明灯

照明灯核心程序

```
void Control_SG90(uint32_t us)
{
//500us,0 度
//1000us,45 度
    int i=0;
    for(i=0;i<10;i++)
    {
        if(us<=20000)
        {
            ll_gpio_bit_set(GPIOB, BIT(6));
            delay_us(us);
            ll_gpio_bit_reset(GPIOB, BIT(6));
            delay_us(20000-us);
        }
    }
}
```

2.6.2 电风扇

本作品使用电脑散热电风扇作为云上智能家居系统的通风系统，模拟真实环境下的系统运行情况，同风扇如图 10 所示。

图 10 电风扇

电风扇核心程序

```
void Control_SG90(uint32_t us)
{
//500us,0 度
//1000us,45 度
//1500us,90 度
//2000us,145 度
//2500us,180 度
    int i=0;
    for(i=0;i<10;i++)
    {
        if(us<=20000)
        {
            ll_gpio_bit_set(GPIOB, BIT(6));
            delay_us(us);
            ll_gpio_bit_reset(GPIOB, BIT(6));
            delay_us(20000-us);
        }
    }
}
```

2.7 阿里云服务器

本产品使用阿里云企业版为产品服务器，最大实例数达 10000 个，支持消息转发、物理模型显示、可视化界面搭建消息转发等功能。是一款十分强大且实用的服务器。主要通过以下步骤与服务器进行通信。

开发前准备：
添加产品与设备，定义控制端和被控端，获取设备的三元组（产品密钥：{ProductKey}设备名字：{DeviceName}设备密钥：{DeviceSecret}）以便使用

图 11

再创建两个设备分别为小程序控制端和开发板端。

图 12创建设备

TXF6200 开发板+ESP01s 连接云服务器进行通信

2.7.1 连接云服务器：

利用 ESP01s(已经烧录含 MQTT 的 AT 固件)，连接 TXFF6200 开发板，其中 ESP01s 的 RX 与 TX 反接。连接好开发板后，连接串口，调试 AT 指令，确认 A T 指令可用后，取下串口在 T XF6200 实验例程中修改代码 wifimode 代码

连接云服务器所需 A T 指令如下(用 defin 定义)：
#define JOIN_WIFI_INFO "AT+CWJAP=\"Hwaial\",\"qqqqwwwwqq\"\r\n"
#define CFG"AT+MQTTUSERCFG=0,1,\"NULL\",\"class-rack&DeviceName\",\"3bbfde7d33ca4065242048dd72d0977a8a17e3dbc85ea7f6a87d061c10560fb7\",0,0,\"\"\r\n"
#define CLIENTID "AT+MQTTCLIENTID=0,\"DeviceName.productKEY|securemode=2\\,signmethod=hmacsha256\\,timestamp=1685613072669|\"\r\n"
#define CONN "AT+MQTTCONN=0,\"iot-060a32fu.mqtt.iothub.aliyuncs.com\",1883,1\r\n"
//订阅 topic，以利用 topic 来与云设备相互通信

```
#define SUB "AT+MQTTSUB=0,\"/jk1zFbWTW6V/class-rack/user/get\",1\r\n"
```

2.7.2 调用函数如 Drv_WIFI_SendCmdAndWaitRequest(CLIENTID, "OK", 3000) != 0)发送指令

正确发送ＡＴ连接指令后，设备于云服务器将会显示为在线

2.7.3 定义云服务器物理模型

由于项目需要上传的传感器信息有：温湿度、光敏、人体感应、雨滴。故需要定义四个物理模型，用于在云平台上显示。阿里云的物理模型支持历史数据查询并绘制历史数据为图表。

功能类型	功能名称（全部）	标识符	数据类型	数据定义	操作
属性	光敏 自定义	guangmin	bool (布尔型)	布尔值： 0 - 白天 1 - 黑夜	查看
属性	人体值 自定义	renti	bool (布尔型)	布尔值： 0 - 无人 1 - 有人	查看
属性	雨滴 自定义	yudi	bool (布尔型)	布尔值： 0 - 有雨 1 - 无雨	查看
属性	开关1 可选	PowerSwitch_1	bool (布尔型)	布尔值： 0 - 关闭 1 - 开启	查看
属性	湿度 自定义	hum	int32 (整数型)	取值范围：0 ~ 500	查看
属性	温度 自定义	temp	int32 (整数型)	取值范围：0 ~ 100	查看

图 13创建物理模型

2.7.4 上传信息到云服务器物理模型的 Topic

需注意：利用 AT 指令上传信息。由于消息内有双引号，此处需特别注意转义字符”\”的使用。保证 AT 指令的正确上传。

（1）获取信息，此处以温湿度数据为例。使用 dht11_read(&temp,&hum); 函数，读出温湿度值 Temp。由于读出的温度值是一个整型数据，发送数据函数 HW_WIFI_SendData()中的参数需要为字符串，因此，需要定义一个字符串数组用于存储温湿度过来的整型数据。

（2）处理信息，利用 Sprintf 函数的组合字符串以及格式化整型数据为字符类型的方法，将读出的 temp 值存入 Temp_str 字符数组内。代码示例如下

格式化数据存入字符数组
sprintf(temp_str, "AT+MQTTPUB=0,\"/sys/jk1XXXXXXX/class-rack/thing/event/property/post\",\"{\\\"method\\\":\\\"thing.event.property.post\\\"\\,\\\"id\\\":\\\"1234\\\"\\,\\\"params\\\":{\\\"temp\\\":%d}\\,\\\"version\\\":\\\"1.0.0\\\"}\",1,0\r\n", temp);

（3）发送信息，调用 HW_WIFI_SendData(temp_str, strlen(temp_str));函数通过串口 USART2 发送数据,为保证数据正确发送，设定延时 delay_ms(100);

（4）查看信息，通过已定义的物理模型就可以看到发送过来的数据

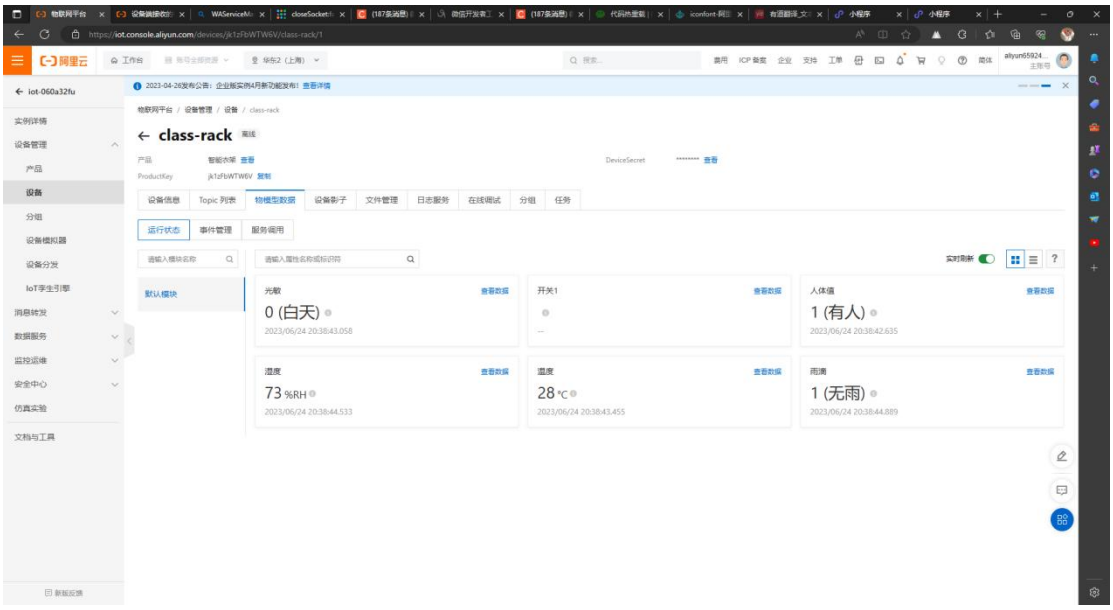


图 14物理模型显示的数据

2.7.5 云服务器控制设备

实现云服务消息上传的同时也需要获取云服务器的指令以控制开发板的设备。在订阅 topic 后，就可以通过此 topic 向开发板发送指令。但云端发下的消息需要处理。由于消息在小程序，云服务器，之间是通过 JSON 格式流转。为方便通信，设备下发指令为 json 数据。具体流程如下：

（1）云端通过 topic 发送命令

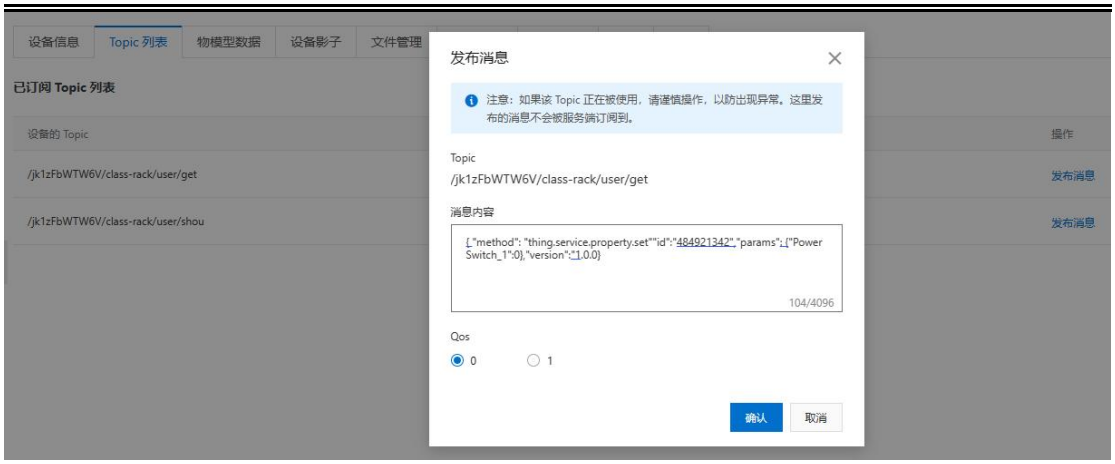


图 15 topic消息下发

图 16串口助手收到的数据

（2）接受云端命令

串口收到了数据，需要通过函数处理，首先需要将数据从串口读出。阅读串口中断函数代码，发现有串口接收函数：

```
if(LL_USART_CHECK_RX_DATA_AVAIL_INTERRUPT_ENABLE(USART2)) {
    /* read USARTx->RBR to clear the interrupt of RX_DATA_AVAIL,
     * But need to be implemented in the A function, you can call
     * LL_USART_READ_DATA() to clear.
     */

    /* Handling something */
    while(LL_USART_GET_LINE_DATA_READY(USART2) != 0) {
        LL_USART_READ_DATA(USART2, usart2_buf[usart2_buf_idx]);
        usart2_buf_idx++;
        if(usart2_buf_idx >= USART2_RBUF_LEN){
            usart2_buf_idx = 0;
        }
    }
}
break;
```

图 17串口接收函数

此函数在串口接收到数据后自动进入中断，并读取串口的数据存放到中断寄存器内。我们需要将中断寄存器内的数据取出并分析。故在 usart2_buf_idx++;上方有以下代码添加：

读取串口数据并存放到 RECS

```
RECS[i++]=usart2_buf[usart2_buf_idx];
                                        if((RECS[i-2]=='\r')|(RECS[i-1]=='\n
'))
                                        {
                                            RECS[i-2]='\0';
                                            i = 0;
                                            cmdAnalyse();//数据处理函数
                                        }
```

当有数据到达时，将数据存储在 usart2buf 数组中，并检查是否有回车或换行符。如果有，将存储在 usart2_buf 数组中的数据复制到 RECS 数组中，并将 i 重置为 0。然后调用 cmdAnalyse()函数对接收到的命令进行分析处理。

（3）分析云端命令

通过图 6 可知，串口收到来自云端的数据的前 13 个字节为”+MQTTSUBRECV:”。图六发送的数据为一个属性为 PowerSwitch_1 的值（此处的开关仅作消息测试，并非云服务器发送的消息）故可以通过此特征判断串口收到的数据。定义 cmdAnalyse()函数，对数据进行分析代码如下：

cmdAnalyse()数据分析函数

```
void cmdAnalyse(void){
    if (strncmp(RECS,"+MQTTSUBRECV:",13)==0){
        uint8_t i=0;
        //uint8_t pos=0; //添加一个变量来记录当前比较的位置
        while(RECS[i++] != '\0') {
            if(strncmp((RECS+i),func4,8)==0) { //从 RECS 数组的当前位置开始比较
                            while(RECS[i++]!=':');
                            Switch2[0]=RECS[i+1];
            }

        }
    }
}
```

首先通过 Strncmp 函数（strncmp(str1，str2 ，num)首先将 str1 第 num 个字符值减去 str2 第一=num 个字符值。若 str1 str2 前 n 个字符相同，则返回 0）检查字符串 RECS 是否以"+MQTTSUBRECV:"开头，如果是，则进入 while 循环。在 while

循环中，它从第一个字符按照 8 个一组逐个检查字符串 RECS，直到遇到空字符为止。在每个字符上，它检查是否有一个名为 fun4（fun4 是定义的属性值，此时为 PowerSwitch_1）的子字符串。如果找到了这个子字符串，则将其值存入 Switch[]字符数组内。

（4）判断指令并控制

在收到数据后，通过对数据判断可执行不同的功能，这里以调用 LCD_ShowString()函数显示数据为例进行控制

判断指令并控制示例

```
                    LCD_ShowString(0, 260, 12, Switch2, 0);
                    if(Switch2[0]=='0')
                    {
                        if(Switch2[1]=='0')
                    {
                        ;
                    LCD_ShowString(64, 260, 12, "00", 0);
                    }else if(Switch2[1]=='1'){
                    LCD_ShowString(64, 260, 12, "01", 0);
                    }else{
                    delay_ms(1);
                    }
                    }else if(Switch2[0]=='1'){
                        if(Switch2[1]=='0')
                    {
                    LCD_ShowString(64, 260, 12, "10", 0);
                    }else if(Switch2[1]=='1'){
                    LCD_ShowString(64, 260, 12, "11", 0);
                    }else{
                    delay_ms(1);
                    }
                    }
    }
```

2.7.6 云服务器消息转发

项目需要将数据显示到微信小程序中，MQTT 协议规定同一 ClientID 只能有一个连接，如果是用同一个 ClientID 在不同地方登录，会先把最先登录的

ClientID 踢下线，所以不同设备之间不能通过订阅同一个直接通信。需要通过云服务转发消息功能将开发板的信息转到微信小程序中显示。同样，从微信小程序发送控制消息到开发板也需要消息转发。此处以开发板使用云服务器转发消息到微信小程序控制端为例介绍。

图 18消息转发

（1）添加数据源

单击数据源页签，执行以下步骤，添加数据源

单击创建数据源。

在弹出的创建数据源对话框，输入数据源名称，例如：AppData。

单击确定。

图 19创建数据源结果（1）

在 AppData 页面，单击添加 Topic：/sys/jk1zFbWTW6V/${deviceName}/thing/event/property/post。

在添加 Topic 对话框，选择需要处理的消息 Topic，然后单击确定。

图 20选择数据源的topic结果（2）

（2）配置数据目的

返回云产品流转页面，单击数据目的页签。

单击创建数据目的。

在弹出的对话框中，输入数据目的的名称

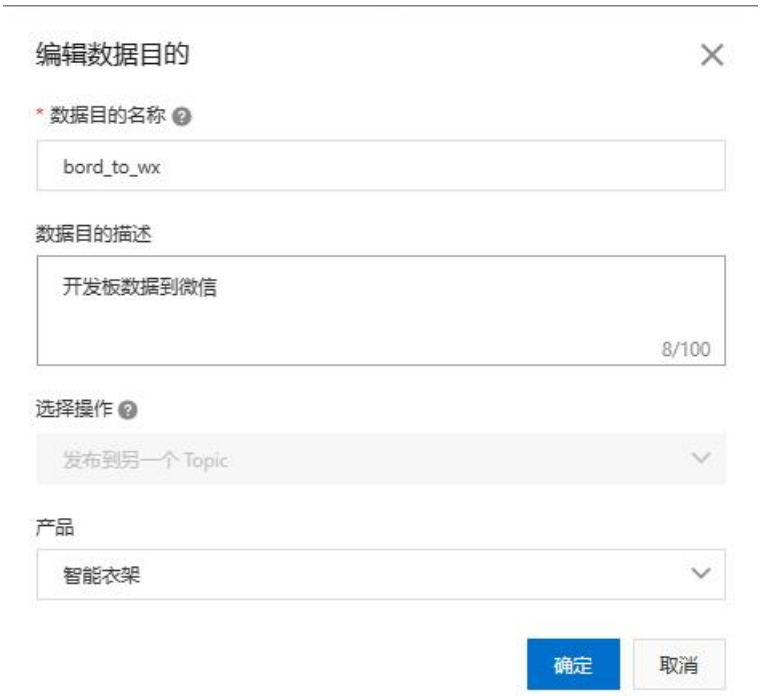


图 21创建数据目的

（3）配置并启动解析器

单击解析器页签，执行以下操作，添加解析器。

单击创建解析器。

在弹出的对话框中，输入解析器名称，例如：DataParser。

单击确定。

在 DataParser 页面，关联数据源 bord_to_wx:

/sys/jk1zFbWTW6V/${deviceName}/thing/event/property/post

在配置向导的数据源下，单击关联数据源此处的数据源是来自开发板物理模型内的数据，获取的数据在弹出的对话框中，单击数据源下拉列表，选择已创建的数据源 bord_to_wx。

单击确定。

在 DataParser 页面，单击配置向导的数据目的，关联数据目的。

单击数据目的列表右上方的关联数据目的。

在弹出的对话框中，单击数据目的下拉列表，选择已创建的数据目的 OtherTopic。

单击确定。

在数据目的列表，查看并保存数据目的 ID，例如为 1000。

后续解析脚本中，需使用此处的数据目的 ID。

在 DataParser 页面，单击配置向导的解析器脚本，完成脚本配置。

图 22解析脚本

在脚本输入框，输入解析脚本，将智能灯设备具有订阅权限的 Topic（微信端订阅的 topic）作为接收开发板指令的 Topic。

//通过 payload 函数，获取设备上报的消息内容，并按照 JSON 格式转换。

单击调试，在右侧面板，选择产品和设备，输入 Topic 和 Payload 数据（JSON 格式），验证脚本可执行。

可执行后，项目即可转发消息。

2.7.7 微信小程序显示数据（经过数据处理后显示，详情在 2.8 内微信小程序一节解释）

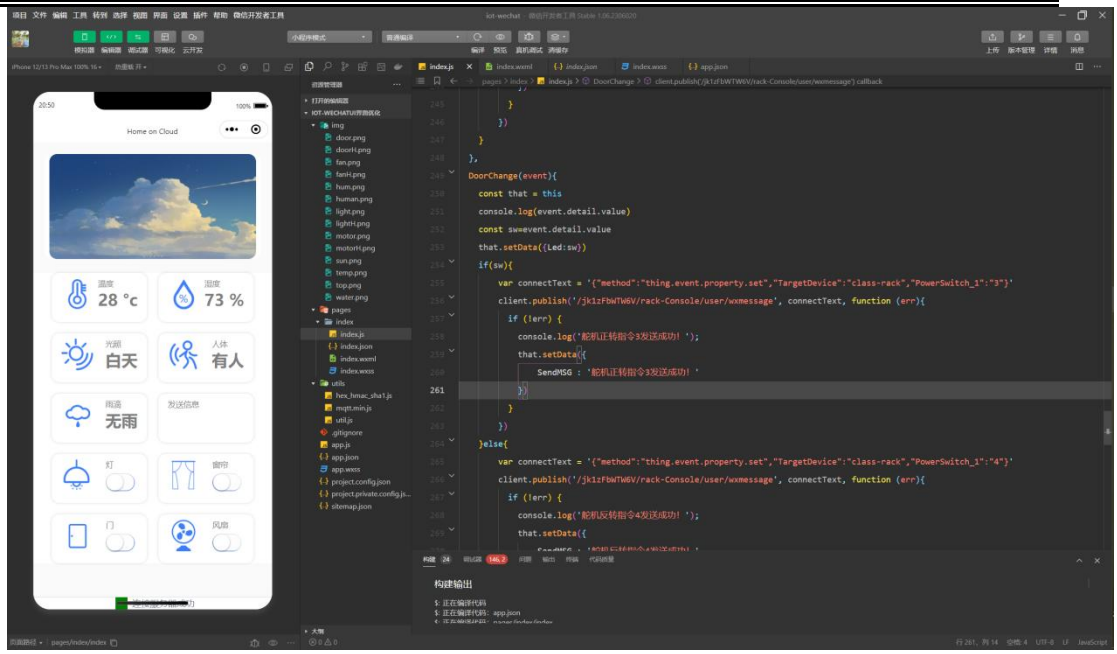


图 23显示结果

2.8 微信小程序

为了实现真正的物联网功能，使用户在外也能查看室内的环境数据和控制家中的设备，我们设计了微信小程序用于用户查看和控制设备。使用微信小程序的目的主要是为了使用简便。下面介绍微信小程序的设计过程：

2.8.1、功能设计

（1）MQTT 连接阿里云

1、首先下载两个必要的文件:MQTT.js 和 hmac-sha1 算法库 hex_hmac_sha1.js。这两个文件是微信小程序连接到阿里云的必要文件，放在项目的 UNTILS:文件夹内。

2、由于微信小程序需要连接服务器，故需要配置与服务器连接的 socket 域名。在微信小程序网页，登录后选择</>开发，在开发管理内配置服务器域名：

图 24配置域名

3、通过代码连接阿里云并订阅 topic。

微信小程序连接阿里云

```
                    doConnect() {
    var that = this;
   // 自己的阿里云三元组
    const deviceConfig = {
      productKey: "",
      deviceName: "",
      deviceSecret: "",
      regionId: "cn-shanghai"
    };
    const options = this.initMqttOptions(deviceConfig);
    console.log(options)

    client                                                                    =
mqtt.connect('wxs://jk1zFbWTW6V.iot-as-mqtt.cn-shanghai.aliyuncs.com', options)
    client.on('connect', function () {
      that.setData({
        connect_text: "连接服务器成功",
        iot_connect: true
      })
///jk1zFbWTW6V/rack-Console/user/get
      //订阅主题
       client.subscribe('/productKey/deviceName/user/wxmessage', function (err)
{
         if (!err) {
           console.log('订阅成功！');
         }
       })
    })
```

（2）登录功能

由于家中设备与环境是私密信息，故设计登录界面以保护用户隐私。

具体实现代码如下

密码验证

```
confirmPwd:function(){
    var pwd = this.data.password_input;
    var that = this
    if(pwd != this.data.key){
      wx.showToast({
        title: '密码错误',
        icon: 'none',
        duration: 2000
      })
    }else{
      wx.showToast({
        title: '验证通过',
        icon: 'success',
        duration: 2000
      })
      wx.setStorage({
        key: "password",
        data: pwd,
      })
      this.login()
    }
  },

  login:function(){
    var that = this
    wx.getStorage({
      key: 'password',
      success(res) {
        console.log(res)
        var pwd = res.data
        if (pwd == that.data.key) {
          that.setData({
            login: true
          })
          that.doConnect()
        }
      }
```

```
    })
  },

  onLoad: function () {
    this.login()
  },
```

当用户输入密码并单击确认按钮时，将调用 confirmPwd 函数。它将输入的密码与存储的密钥进行比较，如果密码匹配，则显示成功消息。如果不匹配，则返回错误提示。如果密码匹配，则将输入的密码存储在设备的存储器中，并调用登录函数。

login 函数从设备的存储中检索存储的密码，并将其与存储的密钥进行比较。如果密码匹配，则将登录变量设置为 true，并调用 doConnect 函数。

加载页面时调用 onLoad 函数，并调用 login 函数检查用户是否已经通过身份验证。

（3）数据显示

在 doConnect 函数内，函数 client.on('message', function (topic, message)可以获取来自 topic 的消息。通过将消息转换为 JSON 格式后，即可使用 date.xxx.xxx 阅读 json 格式消息内容。并解析。下面是示例代码

微信收到数据后解析

```
//接收消息监听
    client.on('message', function (topic, message) {
      // message is Buffer
      console.log('收到消息：' + message.toString())
      //关闭连接 client.end()
      var data = JSON.parse(message.toString())// 将消息转换为 JSON 对象，其消息一定得是 JSON 格式
      //console.log(data);
      if(data.items.temp){
        var tempstr = data.items.temp.value;
        //console.log(tempstr);
        that.setData({
            temperature: tempstr
        })
      }
      if(data.items.hum){
        var humstr = data.items.hum.value;
```

```
        //console.log(tempstr);
        that.setData({
            humidity: humstr
        })
      }
      if(data.items.guangmin){
        var guangminstr = data.items.guangmin.value;
        //白天=0,黑夜=1
        if(guangminstr){
            that.setData({
                GmStatusMSG : '黑夜'
            })
        }else{
            that.setData({
                GmStatusMSG : '白天'
            })
        }
      }
```

此代码为读取光敏传感器发来的数据并解析的示例。

如果消息包含一个名为“temp”的键，则使用该键的值更新页面的“temperature”数据字段。类似地，如果消息包含一个名为“hum”的键，则使用该键的值更新页面的“湿度”数据字段。最后，如果消息包含一个名为“guangmin”的键，则使用该键的值更新页面的“GmStatusMSG”数据字段。如果“guangmin”的值为 1，则“GmStatusMSG”设置为“夜晚”，否则设置为“白天”。

（4）控制指令传送

使用 SWITCH 开关发送控制指令<switch checked="" bindchange="时间处理函数" color="#3d7ef9"/>

此处以灯的时间处理函数 onLedChange 为例介绍控制指令发送，具体内容如下：

灯开关的事件处理函数 OnLedChange（）

```
onLedChange(event){
    const that = this
    console.log(event.detail.value)
    const sw=event.detail.value
    that.setData({Led:sw})
    if(sw){
        var                                connectText                                =
'{"method":"thing.event.property.set","TargetDevice":"class-rack","PowerSwitch_1":"
```

```
7"}'
        client.publish('/xxxxxxxx/xxxxxxxxx/user/wxmessage', connectText, function
(err){
            if (!err) {
              console.log('台灯开指令 7 发送成功！');
              that.setData({
                SendMSG : '台灯开指令 7 发送成功！'
            })
            }
          })
    }else{
        var                                connectText                                =
'{"method":"thing.event.property.set","TargetDevice":"class-rack","PowerSwitch_1":"
8"}'
        client.publish('/xxxxxxxxx/xxxxxxxxx/user/wxmessage', connectText, function
(err){
            if (!err) {
              console.log('台灯关指令 8 发送成功！');
              that.setData({
                SendMSG : '台灯关指令 8 发送成功！'
            })

            }
          })
    }
  },
```

如果开关的新状态为“on”(即:“sw”为 true)，该函数构造一个 JSON 消息，其中包含打开 LED 灯的命令。然后使用“client.publish()”函数将此消息发布到 MQTT 的 topic。如果消息成功发布，该函数将一条消息记录到控制台，并更新页面的“SendMSG”数据字段，以表明命令已成功发送。

如果开关的新状态为“off”(即:“sw”为假)，该函数构造一个不同的 JSON 消息，其中包含关闭 LED 灯的命令。然后使用“client.publish()”函数将此消息发布到 MQTT 代理上的相同主题。如果消息成功发布，该函数将一条消息记录到控制台，并更新页面的“SendMSG”数据字段，以表明命令已成功发送。

2.8.2、UI 界面设计

图 25UI界面实机图

为了项目美观，在项目内使用了来自阿里图标素材库的素材，并同一颜色为#3d7ef9 定义了五个环境数据显示框，一个信息发送框以及四个功能框和头部图片显示框。UI 代码如下

1、主界面

UI 界面 index.wxml

```
<!-- 头部部分 -->
<view class="page-container">
    <view class="head">
        <image class="headIMG" src="/img/top.png"></image>
    </view>
</view>
  <!-- 数据部分 -->
<view class="data-contaniner">
<!-- 温度 -->
    <view class= "data-card">
    <image class="data-card_icon" src="/img/temp.png"/>
    <view>
        <view class="data-card_title">
        温度
        </view>
        <view class="data-card_value">
            {{ temperature}} ° c
        </view>
    </view>
    </view>
    <!-- 湿度 -->
    <view class= "data-card">
    <image class="data-card_icon" src="/img/hum.png"/>
    <view>
        <view class="data-card_title">
        湿度
        </view>
        <view class="data-card_value">
            {{humidity}} %
        </view>
    </view>
    </view>
    <!-- 光照度 -->
    <view class= "data-card">
    <image class="data-card_icon" src="/img/sun.png"/>
    <view>
        <view class="data-card_title">
        光照
        </view>
        <view class="data-card_value">
        {{GmStatusMSG}}
```

```
        </view>
    </view>
    </view>
    <!-- 人体 -->
    <view class= "data-card">
    <image class="data-card_icon" src="/img/human.png"/>
    <view>
        <view class="data-card_title">
        人体
        </view>
        <view class="data-card_value">
          {{RtStatusMSG}}
        </view>
    </view>
    </view>
    <!-- 雨滴 -->
    <view class= "data-card">
    <image class="data-card_icon" src="/img/water.png"/>
    <view>
        <view class="data-card_title">
        雨滴
        </view>
        <view class="data-card_value">
        {{YdStatusMSG}}
        </view>
    </view>
    </view>
        <!-- MSG -->
        <view class= "data-card">
    <!-- <image class="data-card_icon" src="/img/fanH.png"/> -->
    <view>
        <view class="data-card_title">
        发送信息
        </view>
        <view class="data-card_msg">
        <!-- <switch checked="" bindchange="FanChange" color="#3d7ef9"/> -->
        {{SendMSG}}
        </view>
    </view>
    </view>
     <!-- 灯 -->
     <view class= "data-card">
    <image class="data-card_icon" src="/img/lightH.png"/>
    <view>
```

```
        <view class="data-card_title">
        灯
        </view>
        <view class="data-card_value">
        <switch checked="" bindchange="onLedChange" color="#3d7ef9"/>
        </view>
    </view>
    </view>
     <!-- 窗帘 -->
    <view class= "data-card">
    <image class="data-card_icon" src="/img/motorH.png"/>
    <view>
        <view class="data-card_title">
        窗帘
        </view>
        <view class="data-card_value">
        <switch checked="" bindchange="MotorChange" color="#3d7ef9"/>
        </view>
    </view>
    </view>
     <!-- 开关门 -->
    <view class= "data-card">
    <image class="data-card_icon" src="/img/doorH.png"/>
    <view>
        <view class="data-card_title">
        门
        </view>
        <view class="data-card_value">
        <switch checked="" bindchange="DoorChange" color="#3d7ef9"/>
        </view>
    </view>
    </view>
    <!-- 风扇 -->
     <view class= "data-card">
    <image class="data-card_icon" src="/img/fanH.png"/>
    <view>
        <view class="data-card_title">
        风扇
        </view>
        <view class="data-card_value">
        <switch checked="" bindchange="FanChange" color="#3d7ef9"/>
        </view>
    </view>
    </view>
```

```
</view>

  <view class='status'>
    <view                                  style='width:5vw;height:5vw;margin:auto
2vw;background-color:{{iot_connect?"green":"red"}}'></view>
    <view>{{ connect_text }}</view>
  </view>
```

2、样式表

UI 界面 index.wxml

```
.page-container{
    padding: 36rpx;
}
page{
  background-color: rgb(250, 250, 250);
}

.login_block{
  position: absolute;
  left: 0;right: 0;top: 0;bottom: 0;
  display:flex;
  justify-content: center;
  flex-direction: column;

  width: 80vw;
  height: 30vh;
  background-color: white;
  margin: auto;
  padding: 0 3vh;
  border: 1px solid rgb(202, 201, 201);
  border-radius: 20px;
}



.button-container{
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
  }
```

```
.button{
    background-color: white;
    border: 1px solid gray;
    border-radius: 100%;
    width: 20vw;
    height: 20vw;
    text-align: center;
    margin: 10px;
    display:flex;
    justify-content: center;
    align-items: center;
    box-shadow: 0 0 30px rgb(170, 170, 170);
  }

.button_hover{
  border: 2px solid rgb(0, 183, 255);
}



.button_img{
  width: 100px;
  height: 100px;
  margin: 0 auto;
}

.status{
  position: absolute;
  display: flex;
  justify-content: center;
  flex-direction: inherit;
  border-top: 1px solid gainsboro;
  background-color: white;
  width: 100%;
  text-align: center;
  bottom: 0;
  color: slategray;
}

.showdata{
    position: fixed;
    display: flex;
    justify-content: center;
    flex-direction: inherit;
```

```
    border-top: 1px solid gainsboro;
    background-color: white;
    width: 100%;
    text-align: center;
    bottom: auto;
    top: 50%;
    transform: translateY(-50%);
    color: slategray;
  }
  .head{
    display: flex;
    justify-content: space-between;
    border-radius:25rpx;
    justify-content: center;
    background-color: rgb(255, 255, 255);

}
.head .headIMG{
    display: flex;
    justify-content: space-between;
    border-radius:25rpx;
    justify-content: center;
    height: 330rpx;
    width: 650rpx;

}
.data-containiner{
    background-color: rgb(255, 255, 255);
    margin-top: 10rpx;
    display: grid;
    justify-content: center;
    grid-template-columns: repeat(auto-fill,300rpx);
    grid-gap: 36rpx;
}

.data-containiner .data-card{
    background-color: #ffffff;
    height:70px;
    box-shadow: #d6d6d6 0 0 8rpx;
    border-radius:25rpx;
    display: flex;
    justify-content:space-between ;
    padding: 16rpx 30rpx;
}
```

```
.data-containiner .data-card .data-card_icon{
    height: 100rpx;
    width: 100rpx;
}
.data-containiner .data-card .data-card_value{
    font-size: 52rpx;
    font-weight:bold ;
    color: #7f7f7f;
}
.data-containiner .data-card .data-card_msg{
    font-size: 30rpx;
    font-weight:lighter ;
    color: #7f7f7f;
}
.data-containiner .data-card .data-card_title{
    font-size: 25rpx;
    color: #7f7f7f;


}
```

2.9 系统效果

2.9.1 UI 显示数据

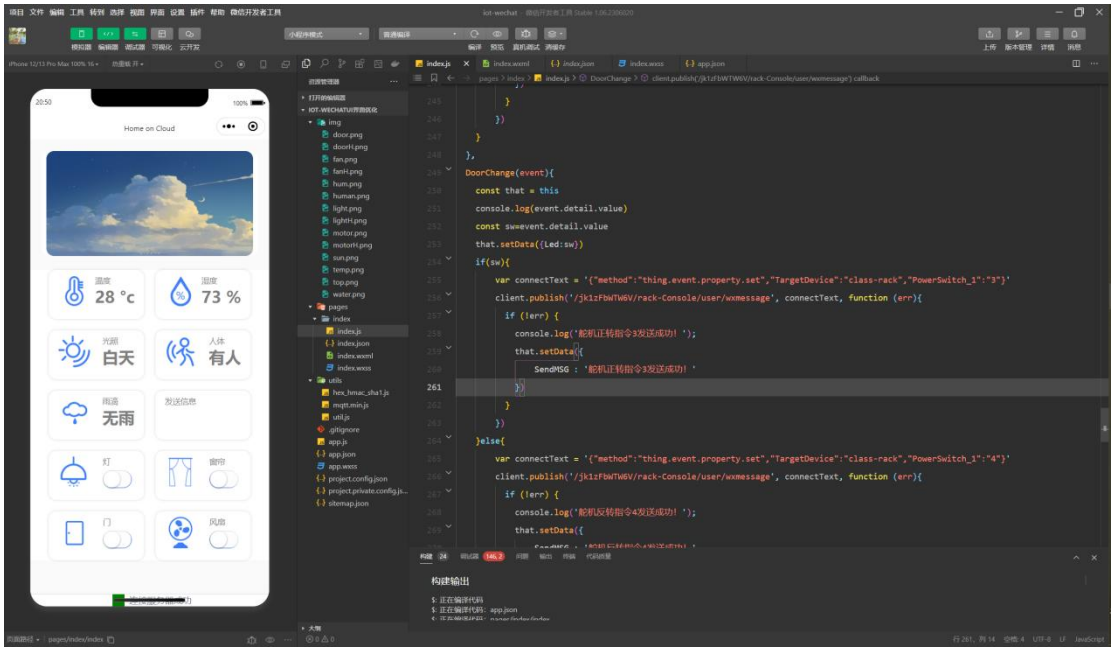


图 26 系统效果

2.9.2 阿里云云平台

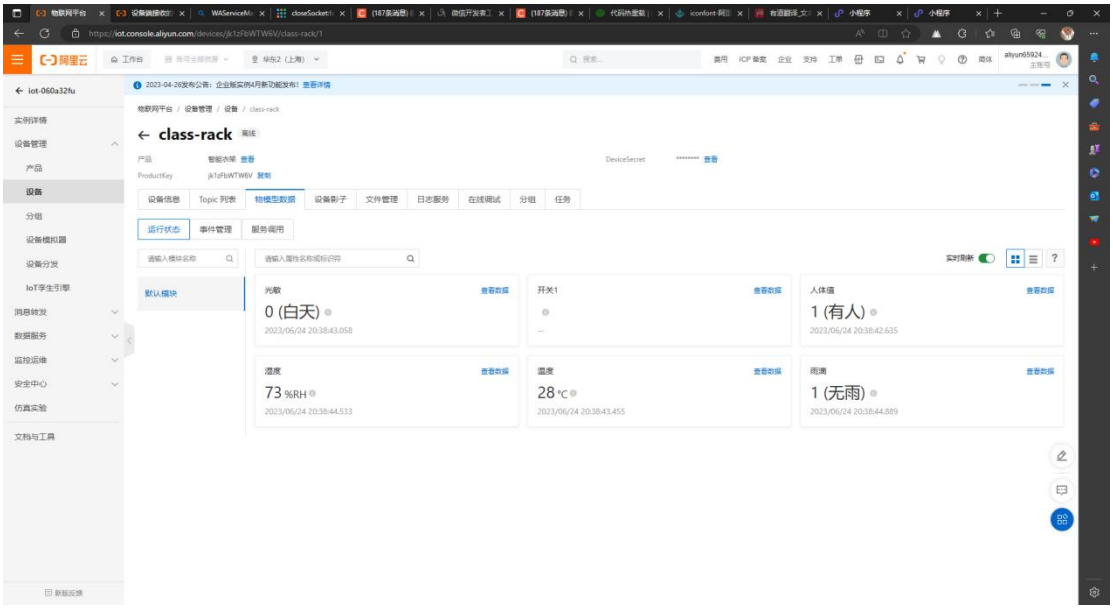


图 26 系统效果

2.9.3 温度云平台历史数据

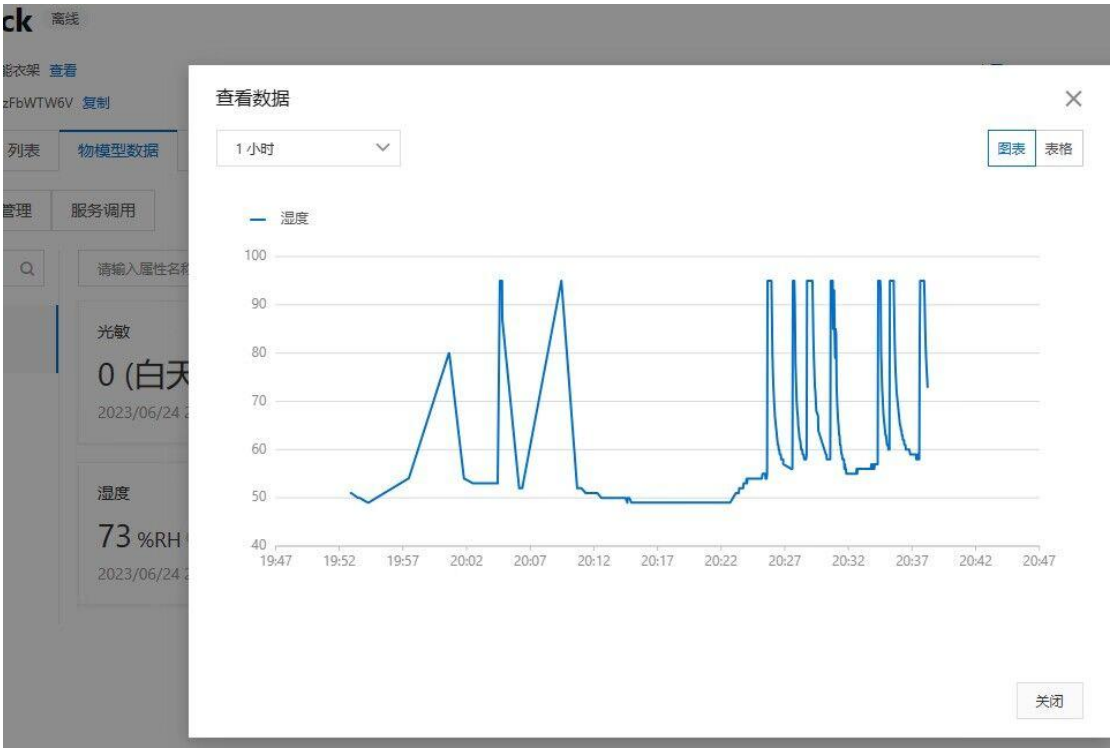


图 26 系统效果

2.9.4 传感器数据

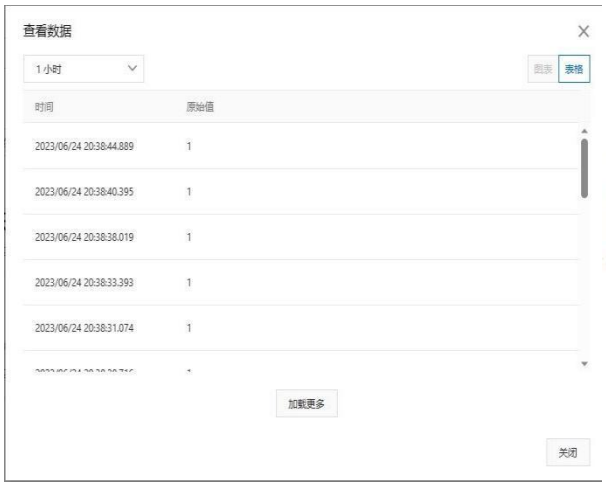


图 26 电风扇

图 26 电风扇

第 3 章 测试报告

作品设计完成后，项目成员综合考虑使用人员因素，使用场景因素，使用方法因素做出相关的性能分析。充分考虑测试云上家居系统数据采集功能，LCD 屏幕实时显示功能，数据上传阿里云平台存储功能，微信小程序实时显示数据功能，系统智能控制运行功能，按钮触摸设置系统，微信小程序远程控制。充分 koala 用户的使用体验。项目测试结果如下：

3.1 数据采集模块测试

云上家居项目的数据采集模块由温湿度传感器、光敏传感器、人体红外传感器、雨滴传感器组成实现对光照、温湿度、人体红外、雨滴等环境信息的采集功能。项目成员从用户角度考虑进行了数据采集模块采集数据的时间和采集数

据的效果进行测试，随机测试了 480 次系统的数据采集。数据采集测试结果如表 2 所示，数据采集效果测试结果图 10 所示。

表1 数据采集测试

传感器	温湿度	人体红外	雨滴	光敏
时间	1-2s	3-4s	0-2s	1-2s

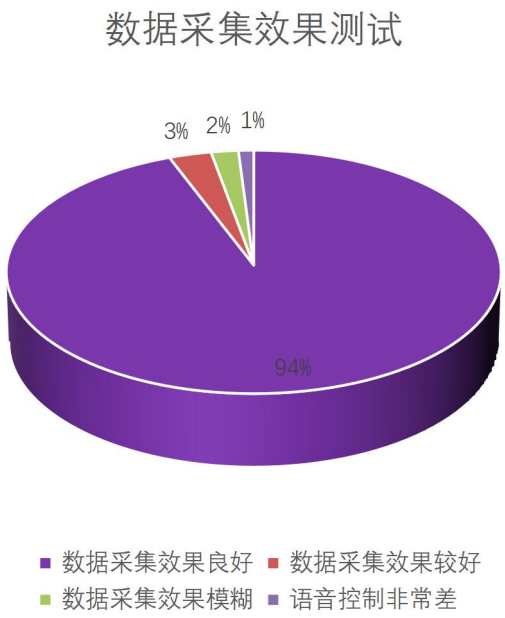


图 27 数据采集效果测试

3.2 屏幕显示模块测试

云上家居系统 LCD 屏幕显示功能将采集的数据实时显示在 LCD 屏幕，项目成员从用户角度考虑进行了屏幕灵敏度和屏幕显示字符测试，随机测试了 500 次系统的屏幕测试。屏幕灵敏度测试结果如 2 所示，屏幕显示字符测试结果表 3 所示。

表2 屏幕灵敏度测试

屏幕灵敏度	快速反应	延迟反应	无反应
百分比（次数）	93%	5%	2%

表3 屏幕显示字符测试

显示类型	字符	字符串	数字	汉字
清晰度	较清晰	较清晰	较清晰	较清晰

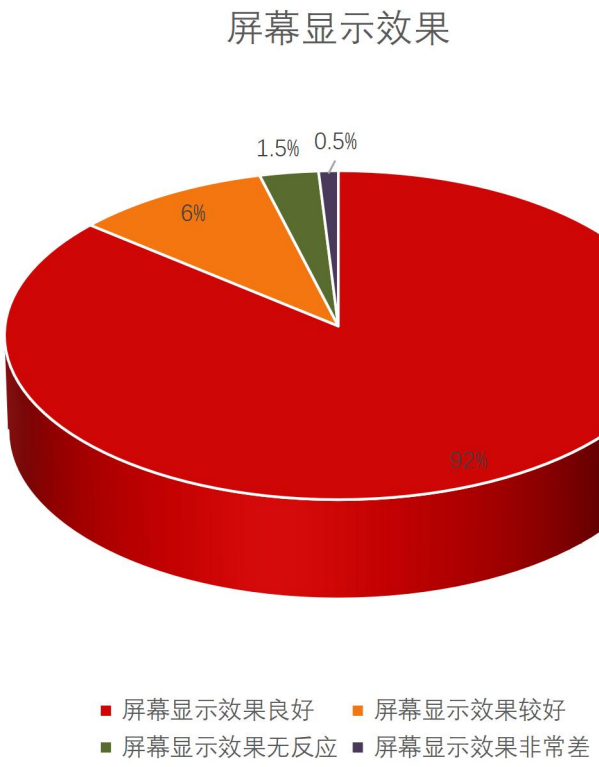


图 28 屏幕显示测试

3.3 电机、舵机模块测试

云上家居系统使用电机模拟实现对窗帘的控制，同时使用舵机实现对于家庭门的智能开锁功能，项目成员从用户角度考虑进行了电机运行测试和舵机运行测试，随机测试了 500 次系统的电机和舵机。电机运行测试结果如表 4 所示，舵机运行测试结果表 5 所示。

表4 电机运行测试

运行状态	正常运行	有延时	不工作
百分比	90%	8%	2%

表5 舵机运行测试

运行状态	正常运行	有延时	不工作
百分比	70%	18%	12%

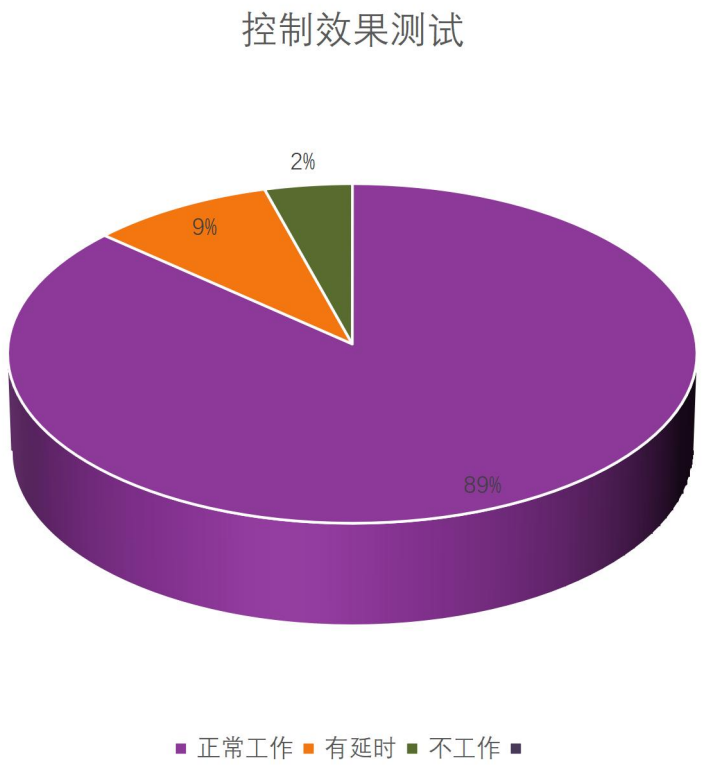


图 29 控制效果测试

3.4 照明、风扇模块测试

云上家居系统使用台灯模拟家庭照明情况，同时使用电风扇实现对于家庭室内气体的排放通风，项目成员从用户角度考虑进行了台灯运行测试和电风扇运行测试，随机测试了 500 次系统的台灯和电风扇。台灯运行测试结果如表 6 所示，电风扇运行测试结果表 7 所示。

表6 台灯运行测试

运行状态	正常运行	有延时	不工作
百分比	86%	10%	4%

表7 电风扇运行测试

运行状态	正常运行	有延时	不工作
百分比	80%	15%	5%

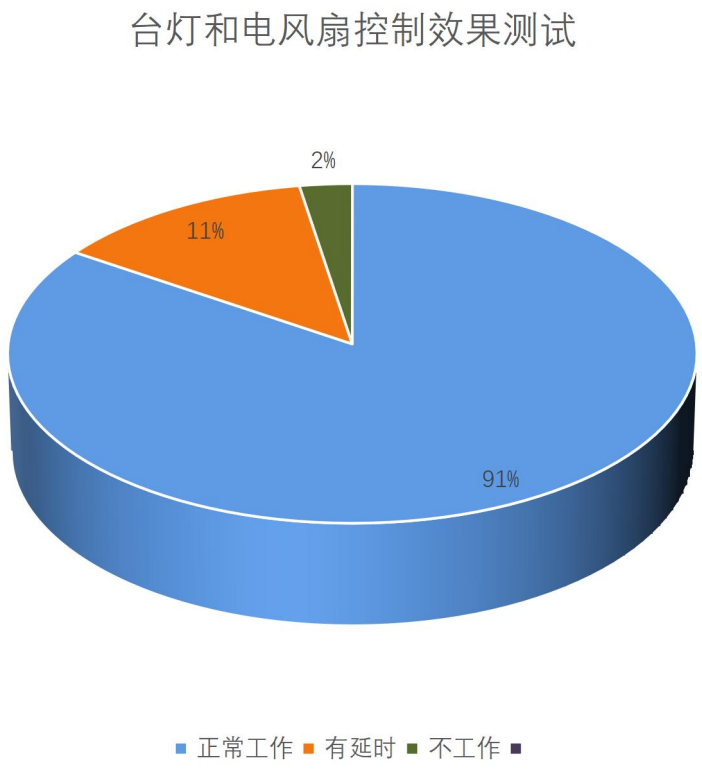


图 30 台灯和电风扇控制效果测试

3.5 阿里云服务器模块测试

云上家居智能控制系统通过阿里云服务器实现对家居系统采集的数据上传到云端，并且存储。项目成员从用户角度考虑进行了阿里云服务器的接收数据测试，随机测试了 500 次系统的阿里云服务器接收数据。阿里云服务器接收数据测试结果表 7 所示。

表7 阿里云服务器接收数据测试

阿里云服务器	数据接受成功	数据接收失败
百分比	90%	10%

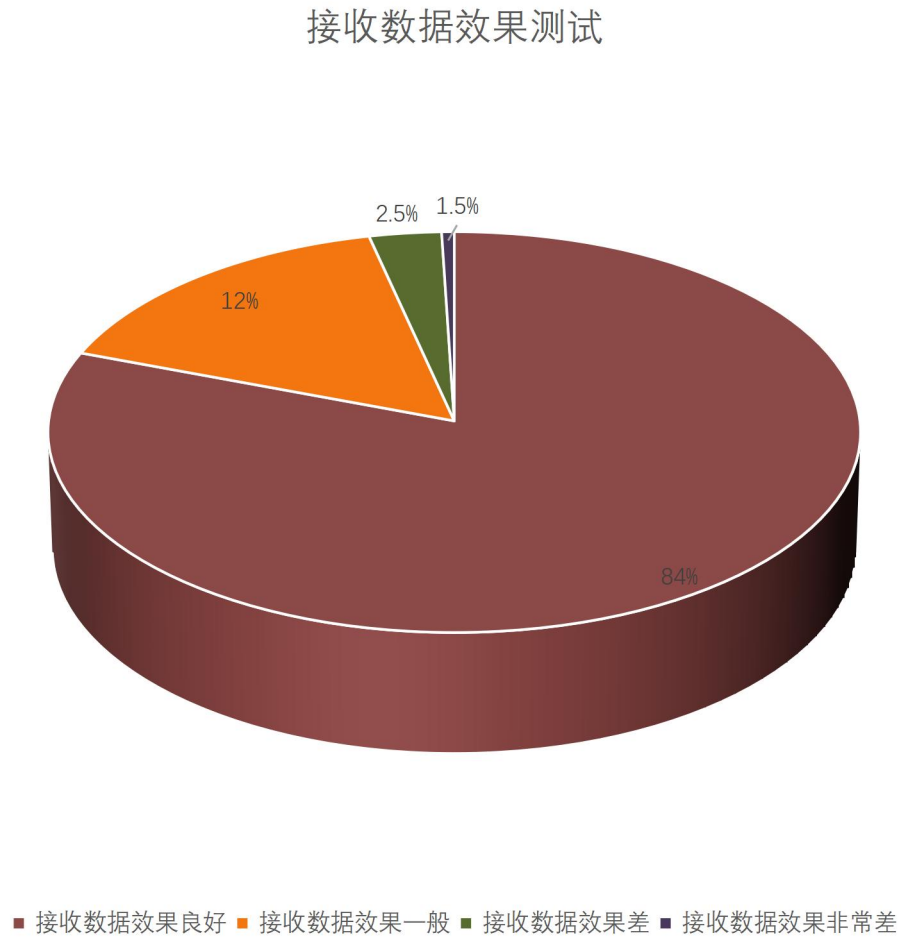


图 31 接收数据效果测试

3.6 微信小程序模块测试

云上家居智能控制系统通过微信小程序实现对家居系统远程控制。项目成员从用户角度考虑进行了微信小程序数据显示测试和微信小程序控制测试，随机测试了 500 次系统的微信小程序数据显示测试和微信小程序控制测试。微信小程序数据显示测试结果如表 8 所示，微信小程序控制测试结果如表 9 所示。

表8 微信小程序数据显示测试

显示结果	正常	错误
百分比	80%	20%

表9 微信小程序数据控制测试

控制结果	正常	错误
百分比	99%	1%

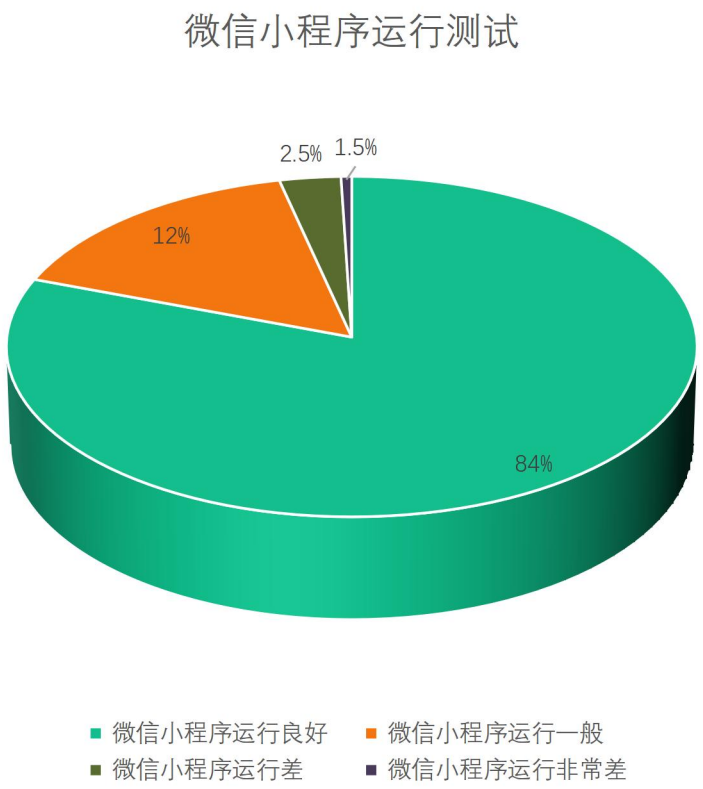


图 32 微信小程序运行测试

第 4 章 应用报告

4.1 市场前景

随着科技的发展，物联网技术已经逐渐渗透到我们生活的各个方面，其中，智能家居是一个重要的应用领域。根据《推进家居产业高质量发展行动方案》[1]，

我国政府已经明确提出要推动家居产业的高质量发展，这为智能家居的发展提供了良好的政策环境。

我们的云上家居智能控制系统，正是基于物联网技术开发的。它可以实现对家居环境的实时监控和远程控制，提高用户的生活便利性和舒适度。此外，我们的产品还可以实现数据的实时采集和上传，这将为用户提供更加精准和个性化的服务。

中国智能家居发展始于 1999 年，2005 年前处于萌芽期，行业内主要参与者为欧美智能化定制品牌;2005-2008 年中国企业开始探索发展路径，智能窗帘、智能灯控等细分领域抢先发展;2008-2019 年，家电、电器类龙头企业开始参与到市场中，行业内企业开始生态圈建设，智能门锁、智能音箱等新领域爆发，行业逐渐进入快速发展期;2020 年以来，随着物联网、5G 等技术的逐渐成熟，智能家居行业迎来了全面爆发期。

在这个市场中，我们的产品有着明显的优势。首先，我们的产品可以实现对家居环境的实时监控和远程控制，这是普通家居产品无法比拟的。其次，我们的产品可以实现数据的实时采集和上传，这将为用户提供更多信息以便用户了解家中的环境情况，并在家中有老人和孩子等需要帮助的人群时，用微信小程序协助其控制，为他们的生活提供更多便利。

4.2 应用推广

为了推广我们的产品，我们可以采取以下几种策略：

1. 与各大家居品牌合作，将我们的产品集成到他们的产品中，以此来扩大我们的市场份额。根据《推进家居产业高质量发展行动方案》[1]，我国政府已经明确提出要推动家居产业的高质量发展，这为我们与家居品牌的合作提供了良好的政策环境。

2. 利用社交媒体和网络广告进行产品宣传，提高我们的品牌知名度。如抖音小红书等社交平台进行推广是一个非常有效的宣传渠道。

3. 举办产品体验活动，让用户亲自体验我们的产品，了解我们的产品优势。根据一项研究，消费者在购买产品时，体验是影响他们决策的重要因素。

4. 提供优质的售后服务，建立良好的口碑，吸引更多的用户。根据一项研究，消费者在购买产品时，售后服务是影响他们决策的重要因素。

4.3 作品展望

在未来，我们将继续改进和完善我们的产品，以满足用户的需求和市场的变化。

首先，我们将进一步提高我们的产品的智能化程度。我们将引入更多的传感器和控制模块，以及引入人工智能算法，以实现对更多家居设备的控制和监控。例如，我们可以引入智能门锁、智能窗帘等设备，以实现对家居安全和隐私的保护。此外，我们还将引入人工智能技术，如对监控摄像头的视频数据进行识别，判断出如家中老人孩子有特殊行为时（如跌倒，靠近窗户，哭喊）以实现对用户的安全进行保障，提供更加个性化的服务。

其次，我们将进一步提高我们的产品的可靠性和安全性。我们将加强对产品的测试和质量控制，以确保产品的稳定性和可靠性。我们还将加强对用户数据的保护和隐私保护，以确保用户数据的安全性和保密性。我们将引入更加先进的加密技术和安全措施，以防止数据泄露和黑客攻击。

第三，我们将进一步提高我们的产品的用户体验。我们将优化产品的界面和交互设计，以提高用户的使用便利性和舒适度。我们还将引入更加智能化的语音交互和手势识别技术，以实现更加自然和便捷的操作方式。此外，我们还将加强对用户反馈的收集和分析，以不断改进和完善我们的产品。

第四，我们将进一步扩大我们的市场份额。我们将加强与各大家居品牌的合作，将我们的产品集成到他们的产品中，以扩大我们的市场份额。我们还将加强对产品的宣传和推广，利用社交媒体和网络广告进行产品宣传，提高我们的品牌知名度。我们还将举办产品体验活动，让用户亲自体验我们的产品，了解我们的

产品优势。此外，我们还将提供优质的售后服务，建立良好的口碑。我们将建立一个专业的售后服务团队，为用户提供全方位的技术支持和服务。我们将建立一个完善的售后服务体系，包括电话咨询、上门服务、远程技术支持等多种服务方式，以满足用户的不同需求。我们还将建立一个用户反馈平台，及时收集用户的反馈和意见，并及时处理和解决用户的问题和困难。我们相信，通过优质的售后服务，我们将赢得用户的信任和支持，建立良好的口碑，进一步扩大我们的市场份额。

最后，云上家居智能控制系统是一款具有广阔市场前景的智能家居产品。我们将不断改进和完善我们的产品，提高产品的智能化程度、可靠性和安全性，优化用户体验，扩大市场份额，提供优质的售后服务，建立良好的口碑。我们相信，通过不断努力和创新，我们的产品将成为智能家居领域的领先品牌。

参考文献

[1] 工业和信息化部、住房和城乡建设部、商务部、市场监管总局近日联合发布《推进家居产业高质量发展行动方案》[EB/OL]. [2023-06-22.]https://www.gov.cn/xinwen/2022-08/08/content_5704634.html

[2] 腾讯. 微信官方文档-小程序[EB/OL]. [2021-04-24]. https://developers.weixin.qq.com/miniprogram/dev/framework/

[3] 易伟.微信小程序快速开发[M].北京:人民邮电出版社,2017.

[4] Sudhir Kumar Sharma, Bharat Bhushan, Raghvendra Kumar, Aditya Khamparia,Narayan C. Debnath. Integration of WSNs into Internet of Things:A Security Perspective[M].CRC Press:2021-04-15.

[5] 珠海泰为电子有限公司[EB/OL].[2023-06-22.]http://www.tai-action.com/