成都东软学院

本科毕业设计报告

题目：基于STM32的嵌入式智能家居远程控制系统设计与实现

|  |  |
| --- | --- |
| 系 别： | 计算机科学与工程系 |
| 专 业： | 计算机科学与工程 |
| 班 级： | 计算机15206 |
| 学生姓名： | 张艺可 |
| 学生学号： | 15310120106 |
| 指导教师： | 张超 |

2019年4月

摘 要

5G时代即将到来，在这个具有划时代意义的通信系统普及后，物联网也会随之蓬勃发展。智能家居是在互联网影响之下的物联化体现，且是如今物联网发展的一个重要组成部分，拥有着非常可观的发展前景。智能家居如今处于一个导入期与成长期的临界点，若随着市场推广普及的进一步落实，智能家居市场的消费潜力必然是巨大的。智能家居的使用会变得越来越普遍，它将为人们提供一种更为便利和高效的生活方式。在不久的将来，智能家居几乎可以涵盖所有传统的弱电行业。

本项目是通过TPYBOARD V202开发板设计远程智能控制系统，通过本系统，能够实现在一定范围内通过上位机PC端远程控制LED灯以及获取家中温湿度数据。开发板采用STM32f405内核，外接发光二极管模拟做室内灯光；采用温湿度传感器采集室内温湿度数据值，使得PC端能够接收到室内温湿度数据。通信协议采用MQTT协议，将ONENET智能物联网平台作为中间平台，在Django框架搭建的网页下进行相应的命令下发和数据交互，用于实现远程控制。且引入科大讯飞语音识别API，使得用户可在相应网页上用语音发出命令实现智能控制。

**关键词**：智能家居，MQTT协议，Django框架，STM32，ONENET智能物联网平台，科大讯飞语音识别API

Abstract

The 5G era is coming soon. After the popularity of this epoch-making communication system, the Internet of Things will flourish. Smart home is an embodiment of the Internet of Things under the influence of the Internet, and is an important part of the development of the Internet of Things today, with a very promising development prospects. Smart home is now at a critical point in the introduction period and growth period. If the market is popularized, the consumption potential of the smart home market will inevitably be huge. The use of smart homes will become more and more common, and it will provide people with a more convenient and efficient lifestyle. In the near future, smart homes can cover almost all traditional weak electricity industries.

This project is designed by TPYBOARD V202 development board remote intelligent control system. Through this system, it can realize remote control LED light and obtain home temperature and humidity data through the PC of the host computer within a certain range. The development board adopts STM32f405 core, and the external light-emitting diode simulates indoor lighting; the temperature and humidity sensor is used to collect the indoor temperature and humidity data value, so that the PC end can receive the indoor temperature and humidity data. The communication protocol adopts the MQTT protocol, and the ONENT intelligent IoT platform is used as an intermediate platform to perform corresponding command delivery and data interaction under the Django framework to implement remote control. And the introduction of the IFLYTEK speech recognition API, so that the user can use the voice to issue commands on the corresponding web page to achieve intelligent control.

“Key Words: Smart Home, MQTT Protocol, Django Framework, STM32, ONENET IoT, Baidu ASR API”

目 录

第1章　绪论 1

1.1 论文研究主要内容 1

1.2 国内外现状 1

第2章　关键技术介绍 2

2.1 关键性开发技术的介绍 2

2.1.1 MQTT协议 2

2.1.2 Django框架 2

2.2.2 科大讯飞语音识别API接口 3

2.2 其它相关技术 3

第3章　系统分析 4

3.1 构架概述 4

3.1.1 功能构架 4

3.1.2 模块需求描述 5

3.2 系统开发环境 7

3.3 系统任务的可行性分析 8

3.3.1 技术可行性 8

3.3.2 系统安全性分析 8

第4章　系统设计 9

4.1 设计指导思想和原则 9

4.1.1 指导思想 9

4.2 构架概述 9

4.3 系统的功能结构设计 9

4.4 系统控制流程 10

第5章　系统实现 11

5.1 接口实现 11

5.1.1 控制管理 11

5.1.2 接口服务器 11

5.2 接口设计说明 11

第6章　系统测试 14

6.1 测试方案及测试用例 14

6.1.1 发送数据 14

6.1.2 接收数据 17

6.2测试方法 19

第7章　结　论 20

致 谢 21

参考文献 22

附 录 23

# 第1章　绪论

21世纪物联网技术蓬勃发展。作为物联网发展中重要的一员，智能家居会在不久的将来走进人们的生活。而这将改变人们现有的生活方式，提供一种更舒适的，更便捷的生活模式。

## 1.1 论文研究主要内容

本项目设计了实现智能远程控制家电的系统，利用PC端作为上位机，TPYBOARD V202开发板作为下位机，通过MQTT协议来实现远程控制。使用MQTT协议进行上下位机的通信，借助物联网移动平台ONENET作为上下位机的中间平台进行数据交互与命令收发。上位机用PC端实现，并采用B/S软件架构。用Python编写上位机服务端，并采用Django框架做一个控制界面，使其能够远程操控TPYBOARD V202外接的发光二极管，且能接收到由其外接的温湿度传感器测得的数据并展示在页面中，通过页面中的相应按钮实现命令下发及获取数据。同时使用科大讯飞语音识别API，使得用户可以使用语音远程控制家电。

## 1.2 国内外现状

智能家居不是高科技，而是便利。好的产品并不是单纯降低消费者的体力劳动，同样也要减少消费者的脑力劳动。以目前最火的NEST为例，这款产品的使用只需要消费者简单操作一个星期后便自动记录用户习惯，以后再无需用户操作。设计简单的操作逻辑，给产品赋予智能分析的能力，自动满足用户习惯，再辅以精致的工艺以用户为中心设计产品。

国内目前有些公司在某些独立应用的智能产品上面做得很有特色，由此可以吸收合作建立统一的行业标准。而此外，国内的互联网巨头也正积极的与家电行业合作推出新品。最终经过市场的历练，将会建立起新的智能家居行业标准，形成完整的产品链供用户选择。

而在智能家居的发展历程中，以美国，日本，新加坡为首，早有着广泛的应用，例如智能报警功能，智能对讲功能，智能监控功能，家电控制功能等等。经济较发达国家，提出过“智能住宅”的概念，智能家居是智能住宅的一个缩影，其标准也在不断的修订之中，至今尚未取得一致的认同。

# 第2章　关键技术介绍

## 2.1 关键性开发技术的介绍

### 2.1.1 MQTT协议

MQTT协议，英文全称为Message Queuing Telemetry Transport，中文翻译是遥信消息队列传输。于1999年由IBM公司提出，现在最新版本是3.1.1。MQTT是一个基于TCP的发布订阅协议，设计的初始目的是为了极有限的内存设备和网络带宽很低的网络不可靠的通信，非常适合物联网通信。

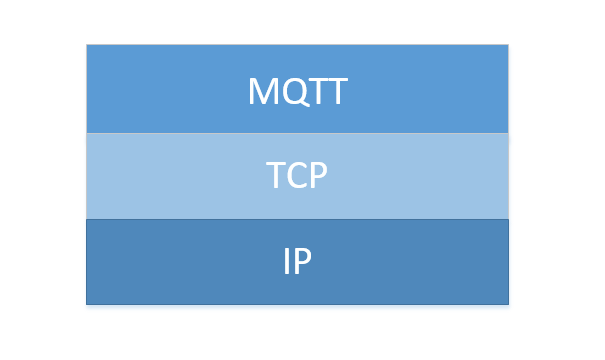


图2.1 MQTT的网络层级

### 2.1.2 Django框架

Django是一个基于MVC构造的框架。但是在Django中，控制器接受用户输入的部分由框架自行处理，所以 Django 里更关注的是模型（Model）、模板(Template)和视图（Views），称为 MTV模式。其中模型，即数据存取层的主要职责是处理与数据相关的所有事务： 如何存取、如何验证有效性、包含哪些行为以及数据之间的关系等。模板，即表现层的主要职责是处理与表现相关的决定： 如何在页面或其他类型文档中进行显示。视图，即业务逻辑层的主要职责是存取模型及调取恰当模板的相关逻辑。模型与模板的桥梁。MVC控制器部分，由Django框架的URLconf来实现。URLconf机制是使用正则表达式匹配URL，然后调用合适的Python函数。

### 2.1.3 科大讯飞语音识别API接口

科大讯飞平台拥有领先的语音识别技术，核心技术达到国际领先水平，语音识别准确率已经超过98%，在业界遥遥领先。其支持多种语种的识别，方便快捷的信息沟通，语音输入速度达180字/分，识别结果响应时间低于200ms——系统运行效率让信息沟通变得无比顺畅。使用其Web语音听写模型时，需要先在Http Request Header中配置相应的授权认证参数。

## 2.2 其它相关技术

本项目中，使用到Socket通信技术。Socket是应用层与TCP/IP协议族通信的中间软件抽象层，它是一组接口。在设计模式中，Socket其实就是一个门面模式，它把复杂的TCP/IP协议族隐藏在Socket接口后面，对用户来说，一组简单的接口就是全部，让Socket去组织数据，以符合指定的协议。

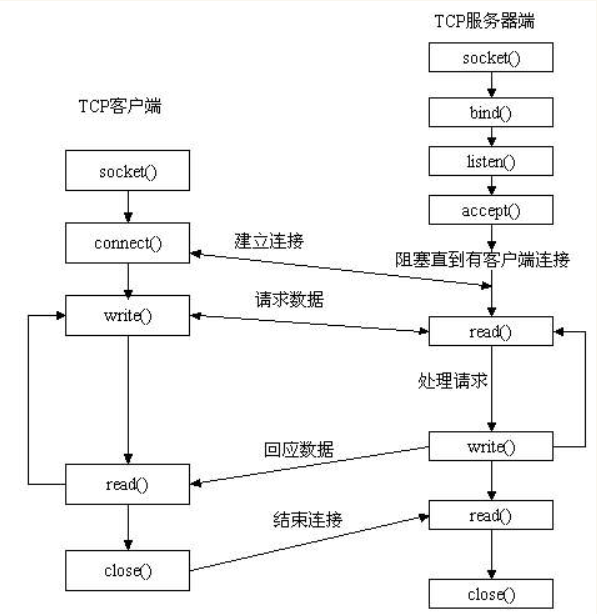


图2.2 Socket工作原理

# 第3章　系统分析

## 3.1 构架概述

本项目为基于STM32的智能家居远程控制系统，主要包括了前端后台及硬件软件的设计。硬件设计主要有发光二极管模块、温湿度检测模块及TPYBOARD V202开发板。软件设计主要有接收温湿度信息模块，控制灯光亮灭模块以及实现智能语音控制硬件模块。

### 3.1.1 功能构架

在Django页面上既可以通过按钮发送指令到ONENET物联网云平台，也可以引入科大讯飞的WEB API，通过识别用户的语音信息来调用控制器中相应的函数，从而能实现命令的发送。当数据传输到ONENET物联网云平台后，通过调用云平台上相应的接口，使用MQTT协议，将指令通过这个平台下发给TPYBOARD V202开发板，然后当开发板接收到指令后，便根据用户的具体操作命令来对开发板外接的发光二极管模块及温湿度传感器模块进行相应的控制。

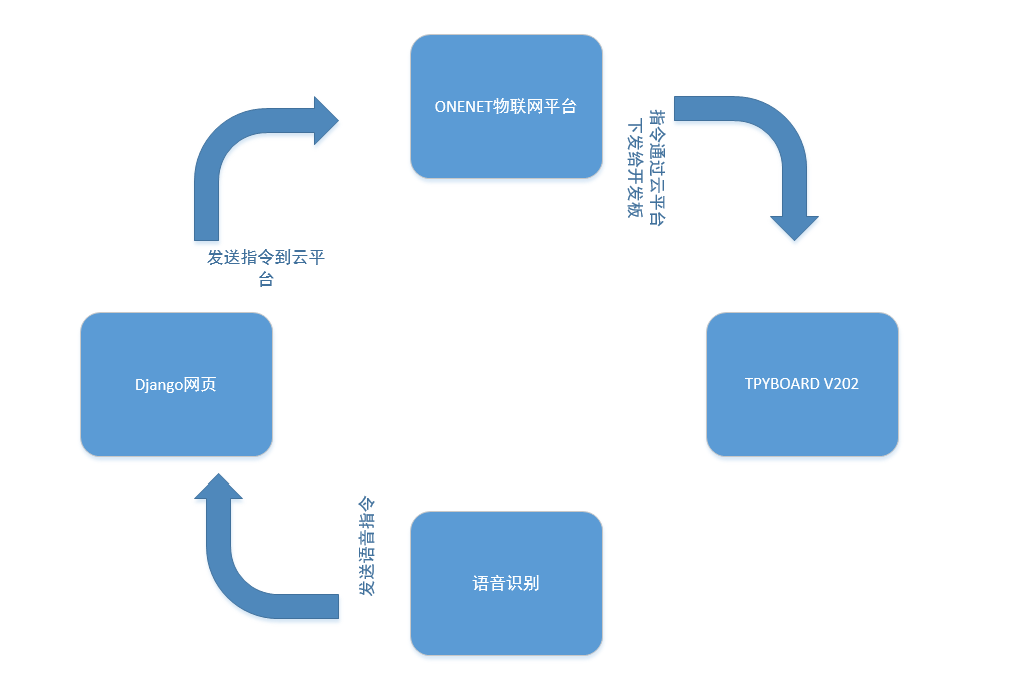


图3.1 基本功能架构图

流程图示意如下：

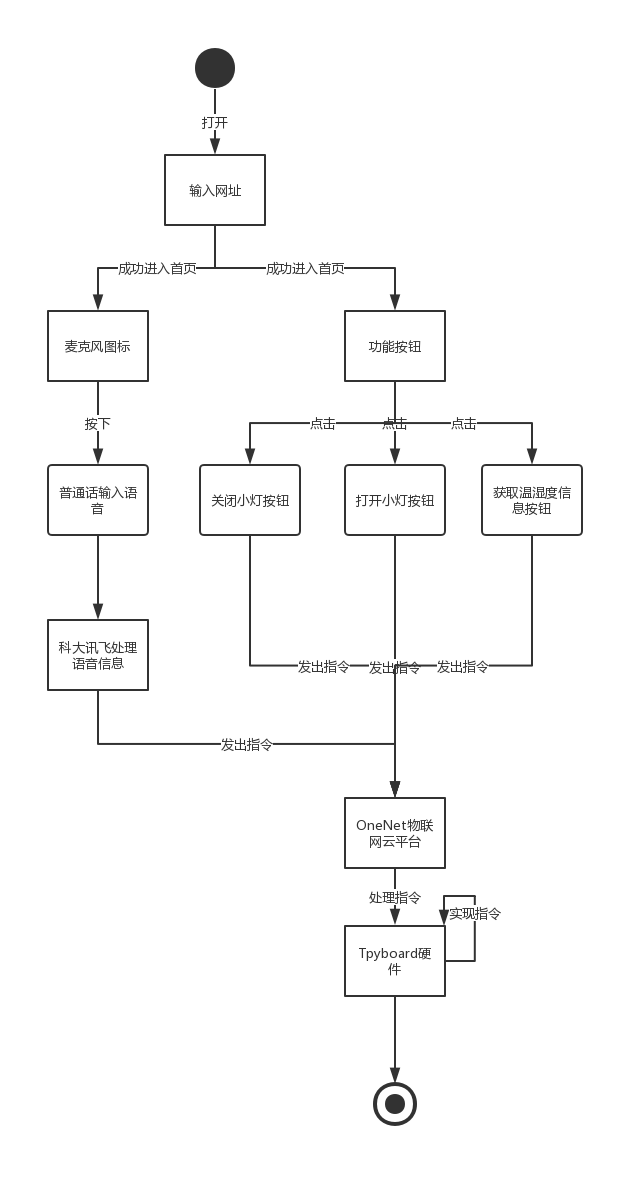


图3.2 智能家居业务流程图

### 3.1.2 模块需求描述

TPYBOARD V202开发板外连两个硬件模块，分别为发光二极管模块以及DHT11温湿度传感器模块。

发光二极管模块：

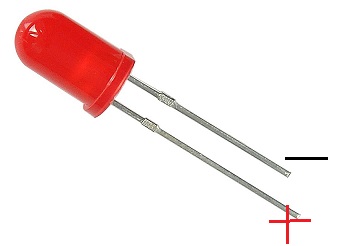


图3.2 二极管模块实物图

TPYBOARD V202与发光二极管模块的具体接线对应：

表3.1 二极管接线对应示意表

|  |  |
| --- | --- |
| TPYBOARD V202 | 发光二极管 |
| G4 | 正极 |
| GND | 负极 |

DHT11温湿度传感器模块：

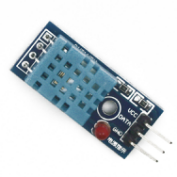


图3.3 温湿度传感器模块实物图

DHT11温湿度共有3个引脚，具体参照如下：

表3.2 DHT11温湿度引脚功能说明

|  |  |
| --- | --- |
| 标注 | 功能 |
| GND | 接地 |
| DATA | 信号输入输出 |
| VCC | 5V |

TPYBoard v202与DHT11温湿度传感器模块的具体接线对应：

表3.3 DHT11接线对应示意表

|  |  |
| --- | --- |
| TPYBOARD V202 | DHT11传感器 |
| +5V | VCC |
| G5 | DATA |
| GND | GND |

连接后效果如下图所示：

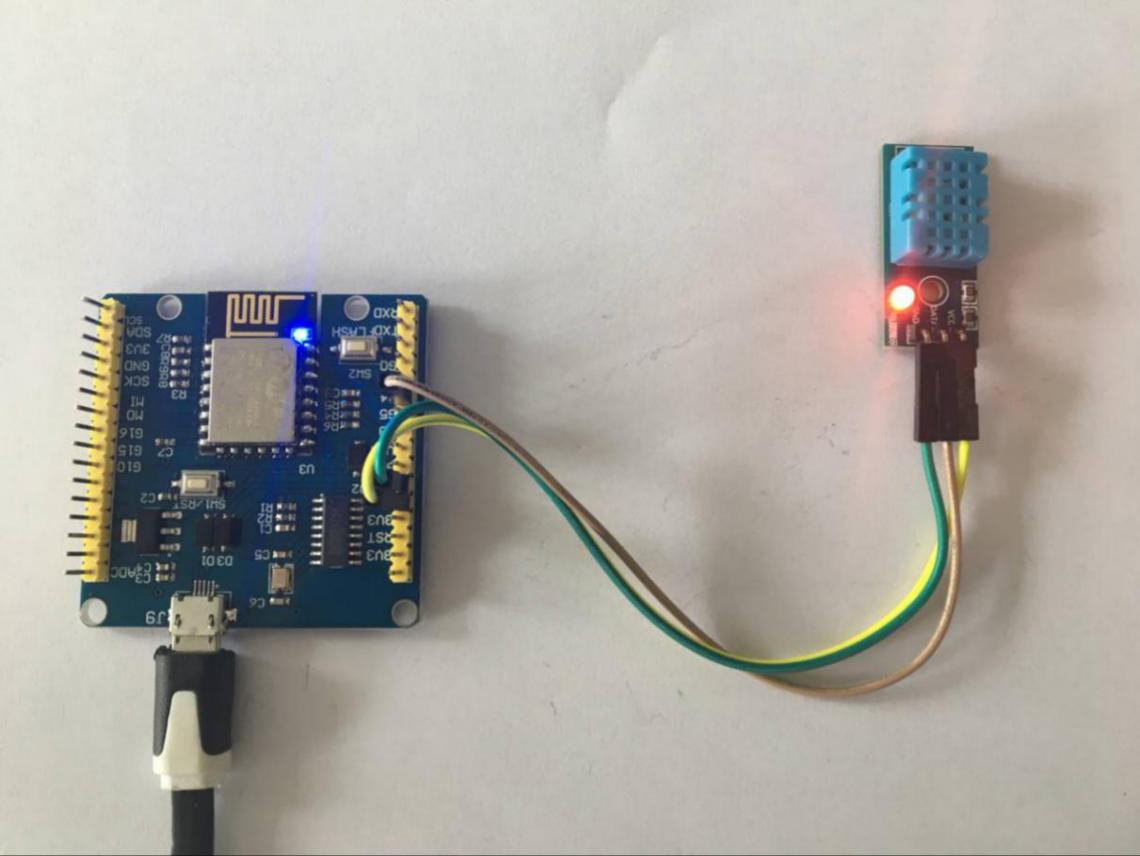


图3.4 连线后效果示意图

## 3.2 系统开发环境

硬件：TPYBOARD V202单片机、DHT11温湿度传感器、发光二极管组成的STM32f405开发板。

软件：MicroPython File Uploader，Visual Studio Code，Google Chrome

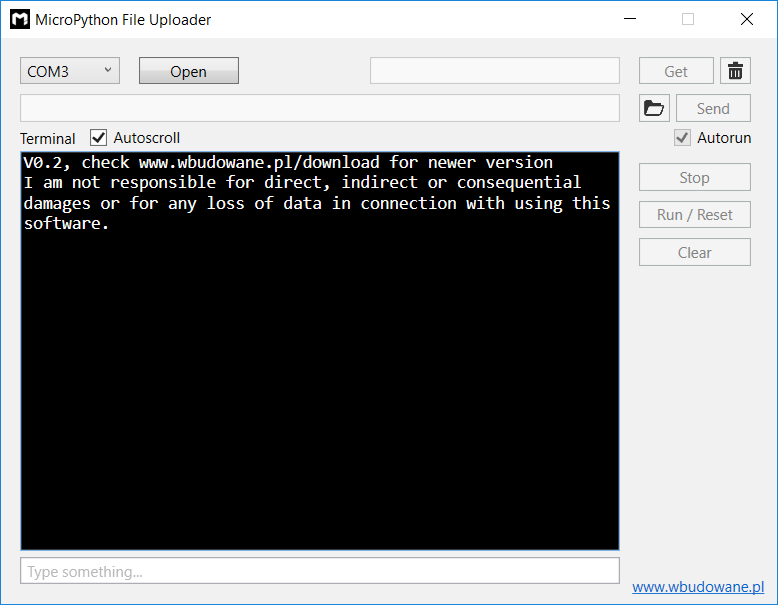


图3.5 MicroPython File Uploader使用截图

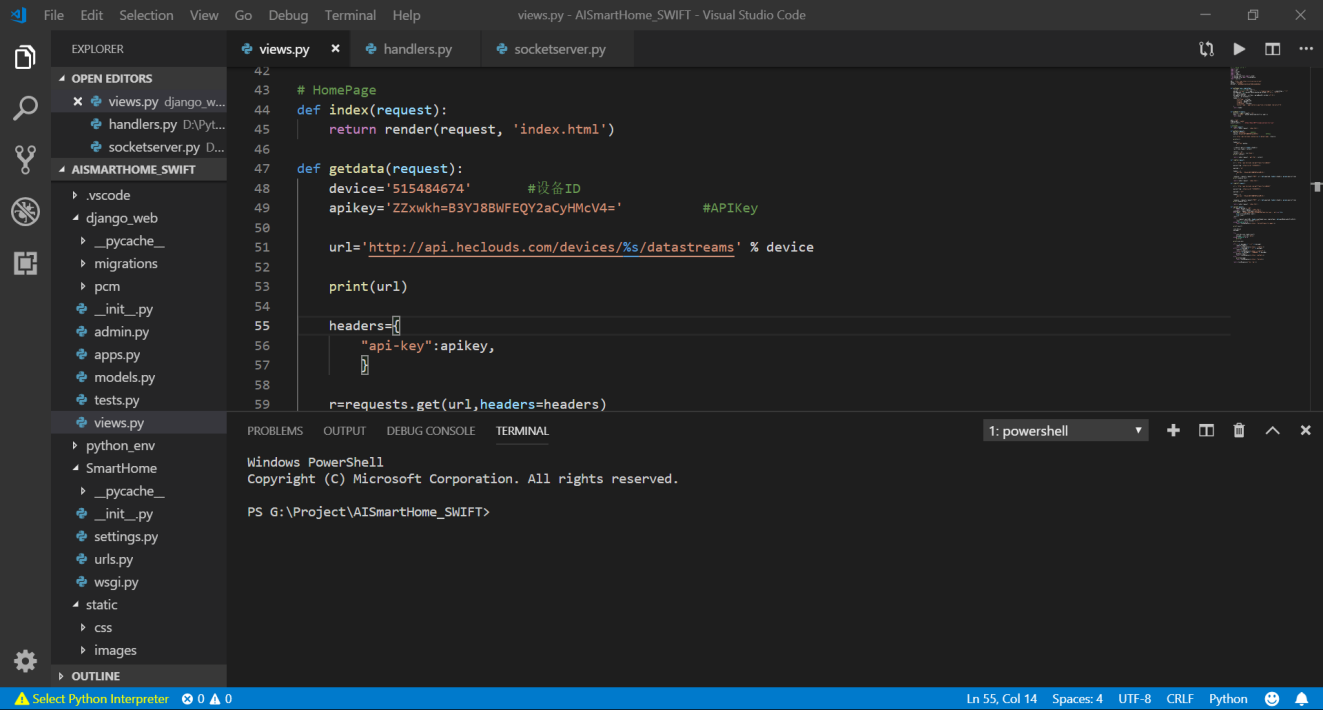


图3.6 Visual Studio Code 使用截图

## 3.3 系统任务的可行性分析

### 3.3.1 技术可行性

本项目用到的开发板是TPYBOARD V202，采用的是STM32f405芯片。f405芯片拥有32位高性能ARM Cortex-M4处理器，支持浮点运算及DSP指令。采用MQTT通信协议，MQTT全称为消息队列遥测传输，是ISO 标准下基于发布/订阅范式的消息协议。它工作在 TCP/IP协议族上，是为硬件性能低下的远程设备以及网络状况糟糕的情况下而设计的发布/订阅型消息协议。项目设计中，拟在下位机用MicroPython编写相关代码。上位机用PC端进行实现，并采用B/S软件架构。用Django框架做一个控制界面，使其能够远程操控TPYBOARD V202外接的发光二极管且能接收到由其温湿度传感器测得的数据。本系统的关键在于使用MQTT协议进行上位机和下位机之间数据的传送。在上位机中使用Django框架做控制界面，下位机中使用MicroPython编写相关代码，同时使用ONENET物联网移动平台作为上下位机通信的中间平台，通过调用ONENET相应的API接口来实现上、下位机之间的命令下发及数据收发，使得用户能在一定距离内实现远程控制。

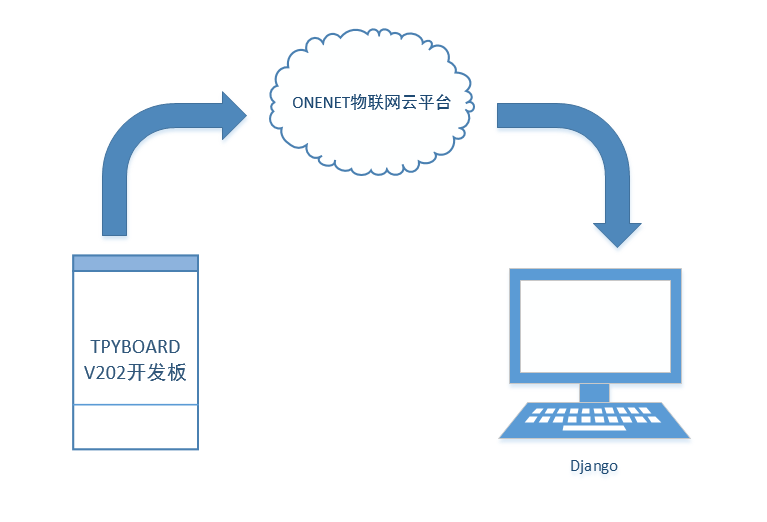


图3.7 可行性分析图

### 3.3.2 系统安全性分析

仅限于处于与TPYBOARD V202开发板相同无线局域网的用户正常使用此系统。

# 第4章　系统设计

## 4.1 设计指导思想和原则

### 4.1.1 指导思想

设计指导思想：精勤博学，学以致用。

## 4.2 构架概述

本项目主要通过STM32f405芯片作为处理器，硬件设计主要有发光二极管、DHT11温湿度传感器及TPYBOARD V202开发板。软件设计主要有接收温湿度数据信息模块，控制灯光亮灭模块以及实现点击界面按钮和智能语音输入控制硬件功能。

## 4.3 系统的功能结构设计

智能家居控制系统由Django控制平台搭建，TPYBOARD V202硬件外接两个模块，分别为发光二极管模块和DHT11温湿度传感器模块。控制系统发出命令至开发板，开发板根据命令内容操作相应外接模块，从而实现远程控制。

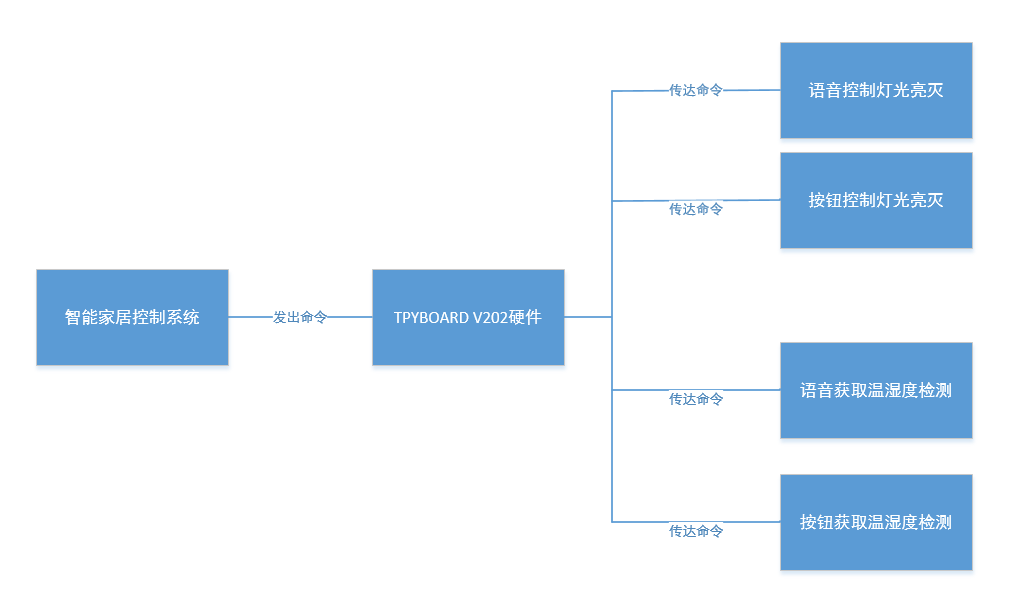


图4.1 功能结构设计图

系统UI界面设计图：

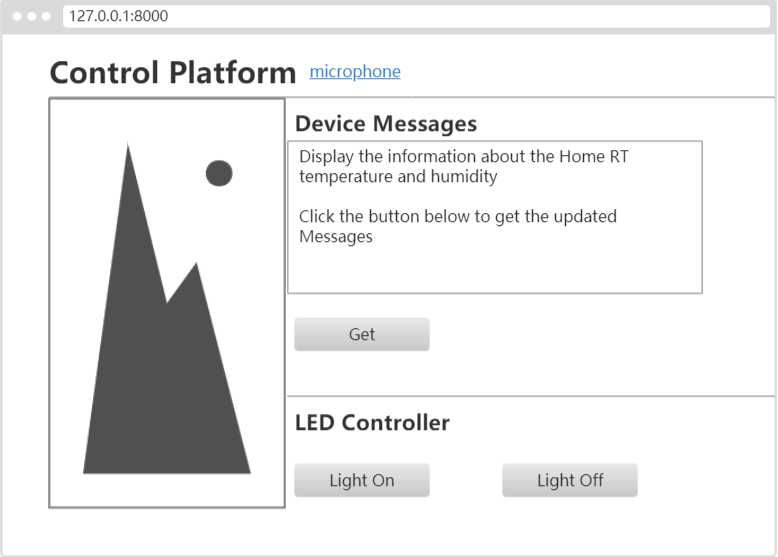


图4.2 Index UI设计图

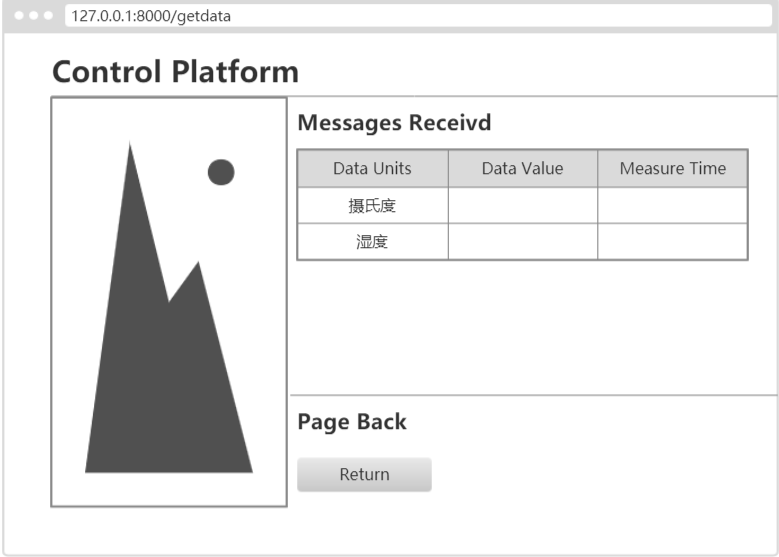


图4.2 Return UI设计图

## 4.4 系统控制流程

本系统用PC做上位机，搭建Django控制平台。用PYBOARD V202开发板做下位机，外接发光二极管模块以及DHT11温湿度感应器模块。上下位机的通信采用MQTT协议，利用ONENET作为中间平台。上位机既可以通过语音发出控制指令，也可以点击Django控制平台上相应的按钮进行命令的下发。

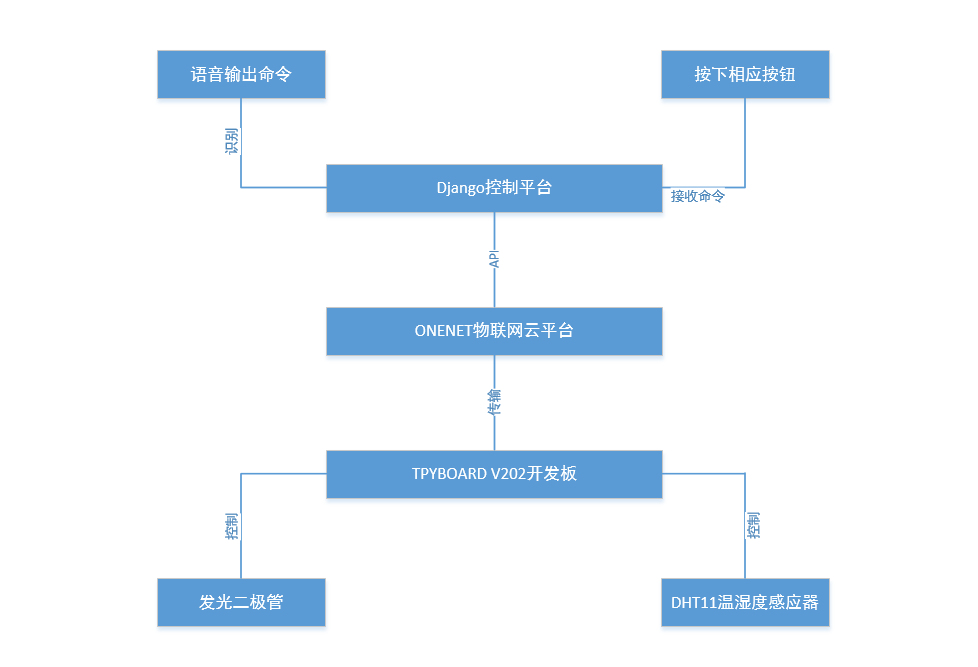


图4.2 智能家居系统控制流程图

# 第5章　系统实现

## 5.1 接口实现

### 5.1.1 控制管理

本项目设计实现远程控制智能家居系统。采用STM32f405内核，TPYBOARD V202开发板上外接发光二极管模块模拟做室内灯光；外接温湿度传感器模块采集室内温湿度数据值，使得PC端能够接收到其传来的温湿度数据值。本系统用PC做上位机，搭建Django控制平台。用TPYBOARD V202开发板做下位机，外接发光二极管模块以及DHT11温湿度感应器模块。上下位机的通信采用MQTT协议，利用ONENET作为中间平台。上位机既可以通过语音发出控制指令，也可以点击Django控制平台上相应的按钮进行命令的下发。

### 5.1.2 接口服务器

ONENET定位为PaaS服务，即在物联网应用和真实设备之间搭建高效、稳定、安全的应用平台：面向设备，适配多种网络环境和常见传输协议，提供各类硬件终端的快速接入方案和设备管理服务；面向应用层，提供丰富的API和数据分发能力以满足各类行业应用系统的开发需求，使物联网企业可以更加专注于自身应用的开发，而不用将工作重心放在设备接入层的环境搭建上，从而缩短物联网系统的形成周期，降低企业研发、运营和运维成本。

## 5.2 接口设计说明

硬件方面：

发光二极管接口设计：

表5.1 发光二极管接口设计表

|  |  |
| --- | --- |
| TPYBOARD V202 | 发光二极管 |
| G4 | 正极 |
| GND | 负极 |

DHT11温湿度传感器接口设计：

表5.2 DHT11温湿度传感器接口设计表

|  |  |
| --- | --- |
| TPYBOARD V202 | DHT11传感器 |
| +5V | VCC |
| G5 | DATA |
| GND | GND |

软件方面：

ONENET物联网云平台接口说明：



图5.1 ONENET API鉴权参数图

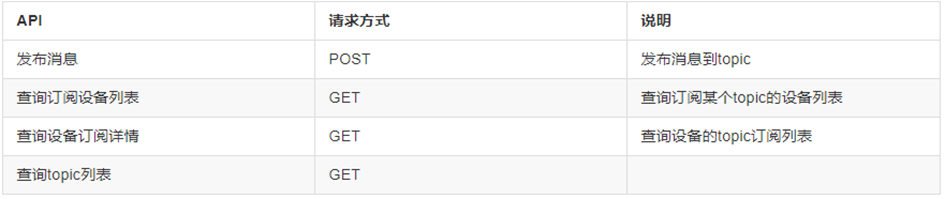


图5.2 设备管理相关API图



图5.3 设备命令相关API图



图5.4 数据管理相关API图

科大讯飞语音听写请求参数：

表5.3 语音听写请求参数表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | 类型 | 必须 | 说明 | 示例 |
| engine\_type | string | 是 | 引擎类型 | sms8k |
| aue | string | 是 | 音频编码 | raw |
| speex\_size | string | 否 | speex音频帧率 | 60 |
| scene | string | 否 | 情景模式 | main |
| vad\_eos | string | 否 | 后端点检测 | 2000 |

科大讯飞语音听写返回值：

表5.4 语音听写返回值参数表

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| code | string | 结果码 |
| data | string | 语音识别后文本结果 |
| desc | string | 描述 |
| sid | string | 会话ID |

# 第6章　系统测试

## 6.1 测试方案及测试用例

测试方法采用黑盒测试。黑盒测试，也可以称为功能测试、数据驱动测试或基于规格说明的测试。测试者不用知道程序的内部情况，不需具有应用程式的程式码、内部结构和程式语言的专门知识。只了解程序的输入、输出和系统的功能，这是从用户的角度针对软件界面、功能及外部结构进行测试，而不用思考程序内部逻辑结构。

测试用例如下表所示：

表6.1 测试用例表

|  |  |  |
| --- | --- | --- |
| 测试用例 | 嵌入式智能家居远程控制系统设计与实现 | |
| 用例描述 | 基于STM32的智能家居远程控制系统 | |
| 测试环境 | 软件环境：Visual Studio Code，MicroPython File Uploader | |
| 硬件环境：TPYBOARD V202 | |
| 网络环境：iPhone手机开启热点，开发板和电脑连接热点 | |
| 设计人 | 张艺可 | |
| 用例编号 | 操作步骤 | 预期结果 |
| 1 | 按下Index界面的Light On按钮 | TPYBOARD V202开发板外接二极管小灯点亮 |
| 2 | 按下Index界面的Light Off按钮 | TPYBOARD V202开发板外接二极管小灯熄灭 |
| 3 | 按下Index界面的Get按钮 | 获取到温湿度信息，并显示在getdata界面上 |
| 4 | 按下Index界面的麦克风按钮，普通话输入语音，完成后松开按钮 | 能识别出普通话中的指令，并实现相应功能 |
| 5 | 按下getdata界面的Return按钮 | 返回到Index界面 |

### 6.1.1 发送数据

通过Django控制平台界面按钮发送命令：

点击Light On按钮来点亮小灯：

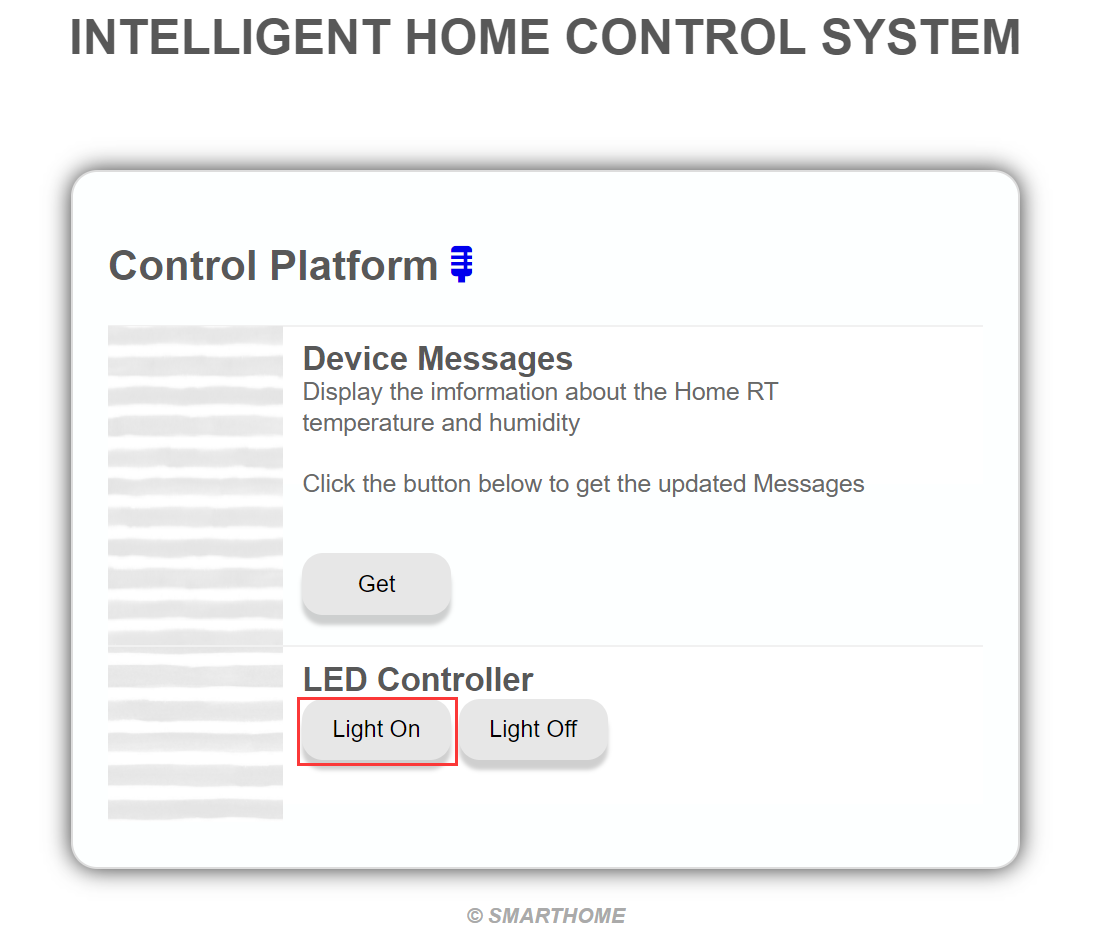


图6.1 Light On按钮点亮小灯

点击Light Off按钮来熄灭小灯：

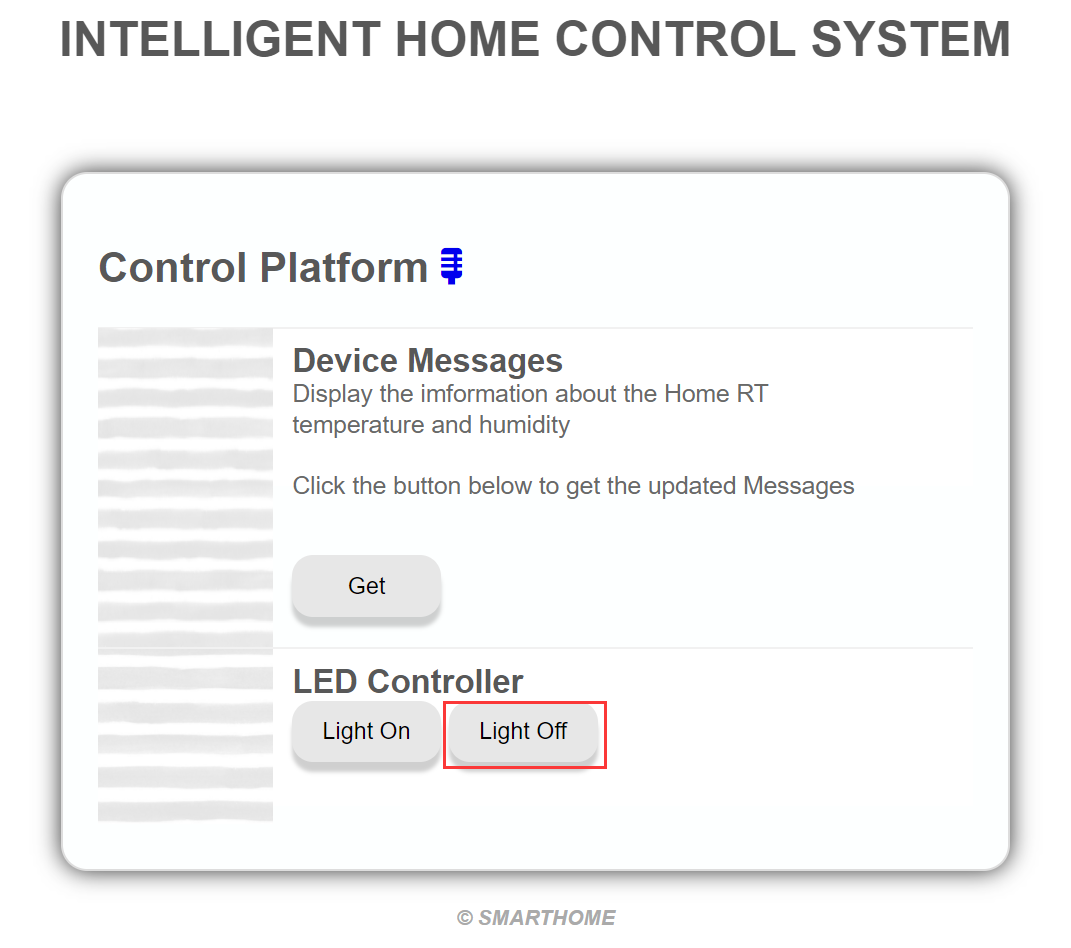


图6.2 Light Off按钮熄灭小灯

点击Get按钮来获取温湿度信息：



图6.3 Get按钮获取温湿度数据

点击“Return”按钮来返回到index界面

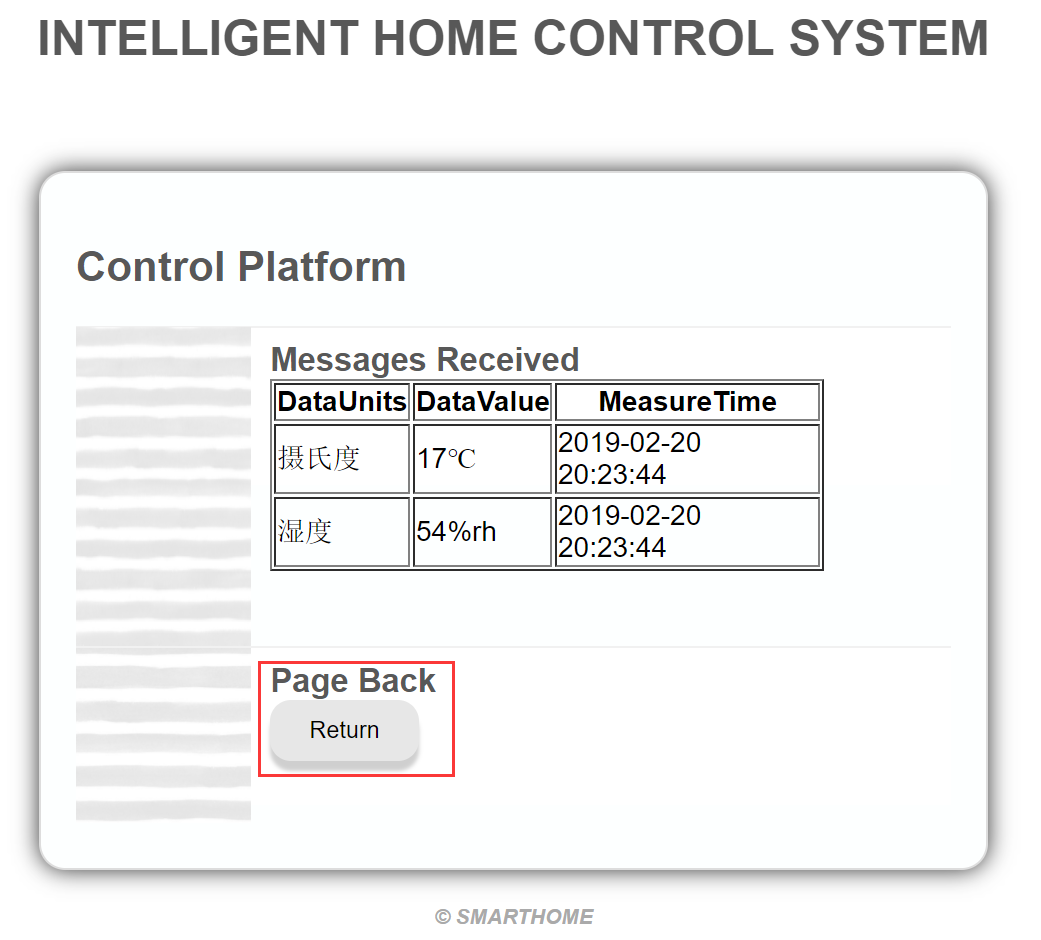


图6.4 Return按钮返回index界面

通过Django控制平台语音输入发送“开灯”/“关灯”/ “获取信息”命令：

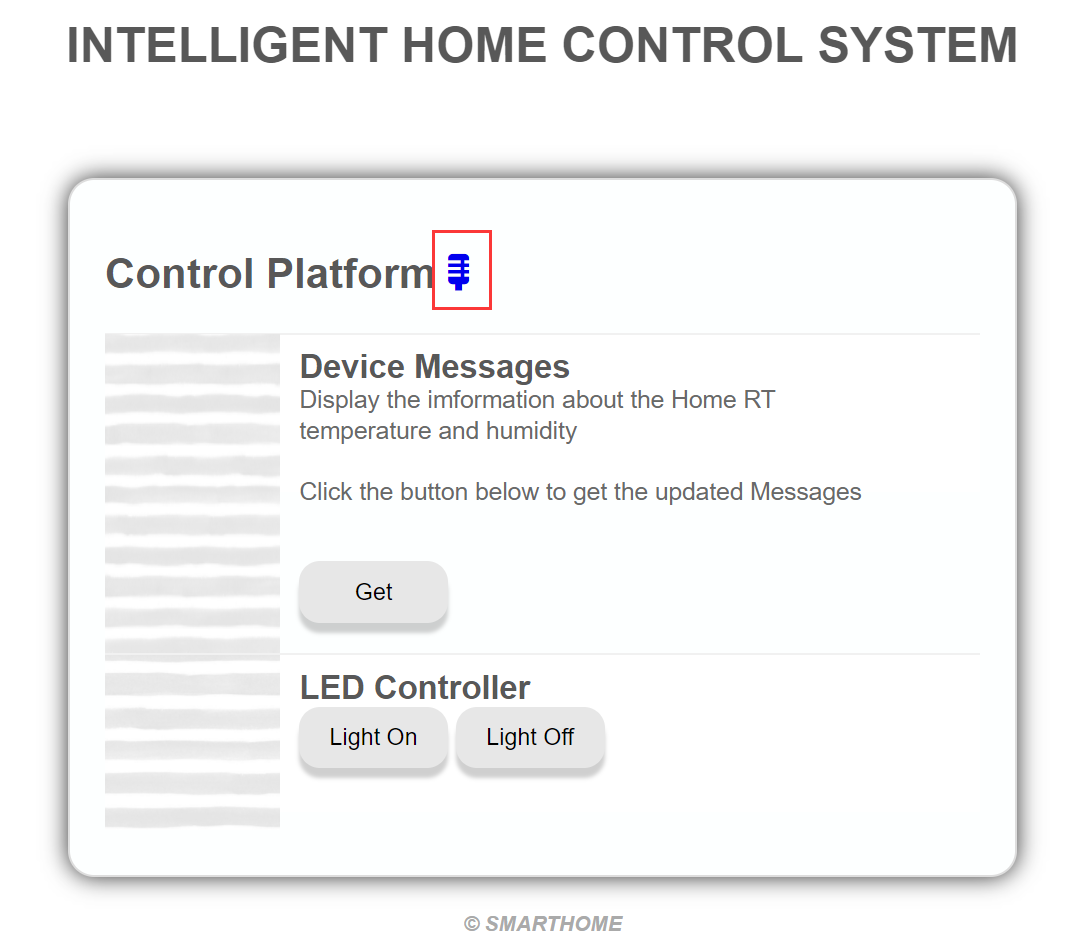


图6.5 通过语音发送命令

### 6.1.2 接收数据

点亮小灯：

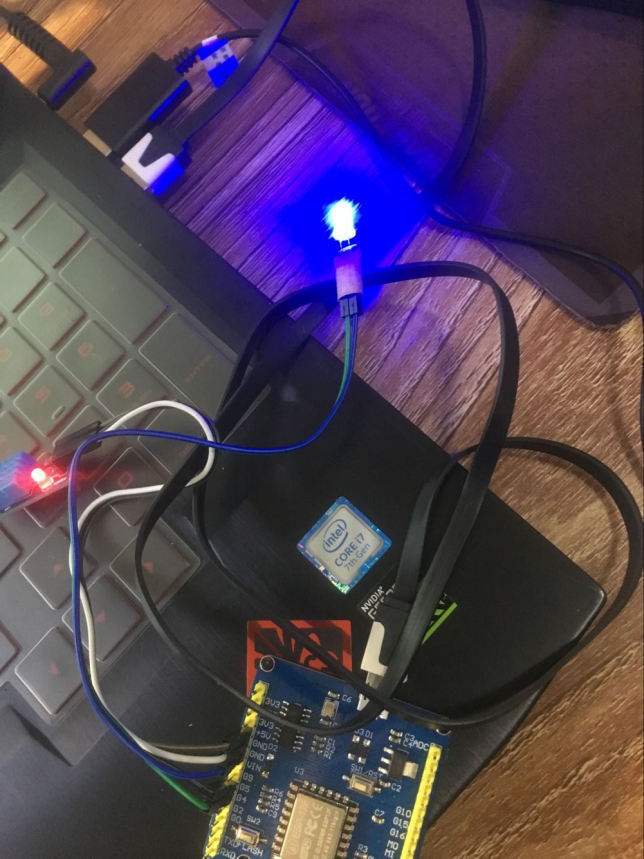


图6.6 点亮小灯

熄灭小灯：



图6.7 熄灭小灯

获取温湿度数据：

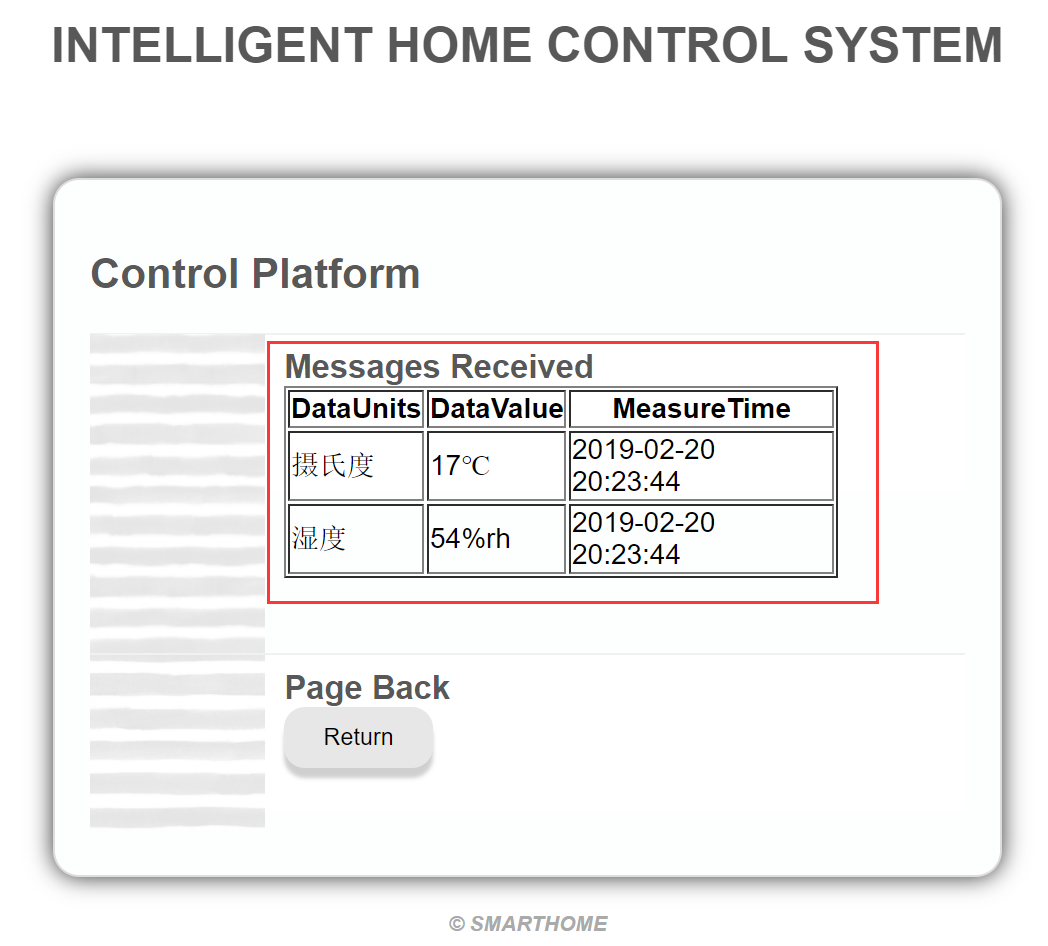


图6.8 获取温湿度数据

返回到index界面：

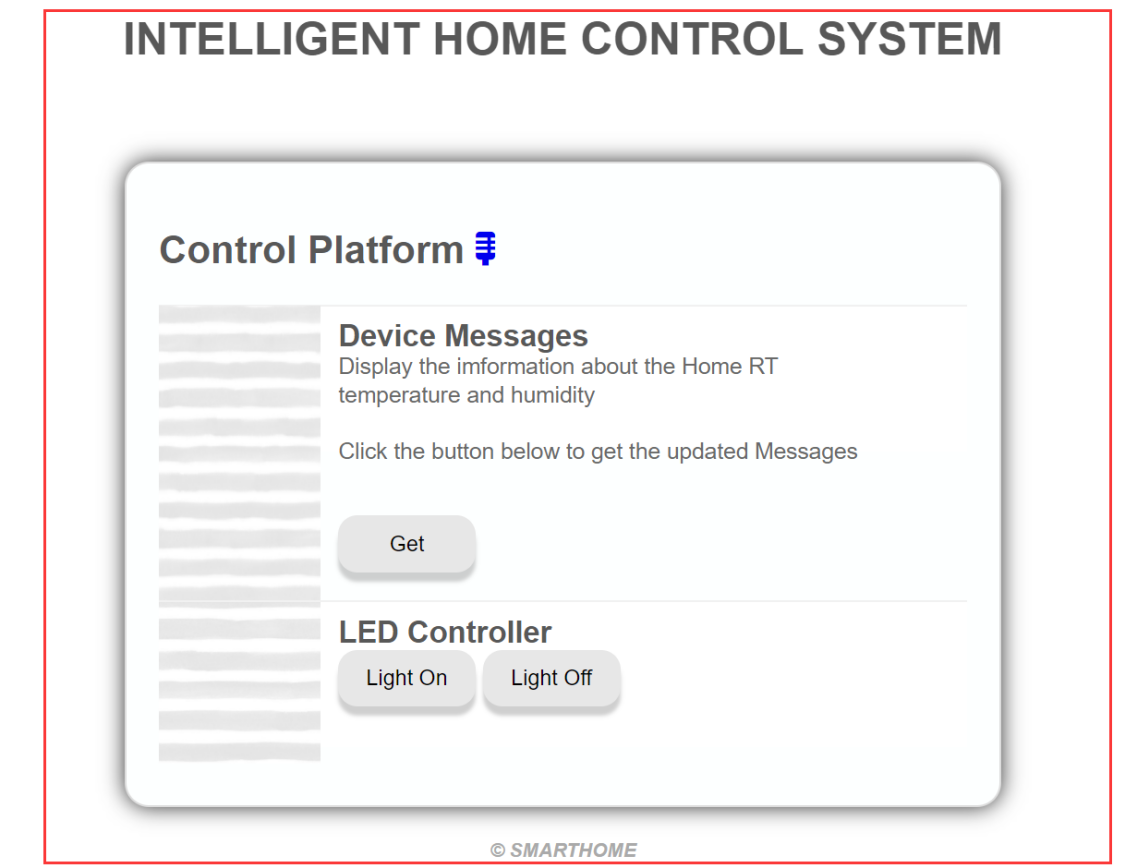


图6.9 返回index界面

## 6.2测试方法

测试方法主要包含两个部分，界面中的按钮测试及语音控制测试。

按钮测试：点击Django界面中的四个按钮，分别为“Light On”、“Light Off”、“Get”、“Return”，测试按钮能够实现相应功能，测试结果表明按钮模块工作正常。

语音测试：按住麦克风样图标，结束录音后松开鼠标，录音内容分别为“开灯”、“关灯”、“获取信息”，测试语音听写能否正确识别并匹配后，调用相应函数实现对应功能，测试结果表明语音识别模块正常。

# 第7章　结　论

本项目采用TPYBOARD V202开发板进行设计，其STM32f405内核能够完美支持MicroPython语言。通过本次毕业设计的实践，完成软硬件部分的相关设计，完成PC端前端后台的相关编写；实现在一定距离内下发命令来远程控制家电，以及接收由温湿度传感器测得的数据的功能，整个项目的设计共分为五个关键步骤来完成。具体步骤如下：

第一步：查阅STM32f405芯片资料，了解其内部基本的电路结构。上网查找TPYBOARD V202开发板学习资料，阅读TPYBOARD V202英文手册。

第二步：查阅MQTT协议的使用。了解MQTT的基本工作原理，重点阅读其关于消息订阅的概念。

第三步：查阅关于在上位机PC端使用Django框架设计界面的相关文献资料，在上位机PC端用Django框架做一个控制界面。整个系统拟采用B/S架构，通过点击界面上的相应功能按钮来对TPYBOARD V202开发板外接二极管及温湿度传感器进行相关操作。

第四步：查阅关于在下位机TPYBOARD V202开发板使用MicroPython语言进行编程的相关文献资料，并实现。将上位机与下位机通过MQTT协议进行通信，并且使用移动物联网平台ONENET作为上下位机交互的中间平台。查阅调取ONENET平台中相应API接口的文档资料。

第五步：上位机通过相应的语音命令或者相应按钮实现控制TPYBOARD V202开发板外接发光二极管模块的亮灭。下位机将温湿度传感器所测得的数据发送到上位机Django界面中的Message模块，使得PC端在接收到相应的语音命令或者按下对应按钮后能够显示测得的室内温湿度数据。

# 致 谢

在整个毕业设计的各个阶段我都得到了张超老师的悉心指导和耐心帮助。因为本人对物联网一直很感兴趣，再加上智能家居占据着物联网的一个举足轻重的地位，所以在设计毕业项目时，优先考虑研究智能家居这一块内容，硬件选取TYPBOARD V202，这块板子采用STM32f405内核，能够完美支持MicroPython语言。在考虑交互式界面的实现时，也使用了Python框架Django，初次接触这个框架，有很多陌生的知识点，但是经过一周左右的熟悉，也能上手使用框架来搭建界面了。

这个系统最初的构想只是通过交互式界面上的按钮功能来实现对硬件上相应部分的控制，例如改变小灯亮灭的状态以及获取温湿度的数据，但在设计整个项目的过程中，我又有了新的想法，尝试用语音来实现其智能语音控制的功能。既然有了新主意，就要找新的思路，在界面按钮功能完成后，我首先搜索了一下关于百度语音的资料，在阅读了相关的文档后，我在整个项目中首次引入了百度语音Python的API包，并在页面的中心设置了有着一个录音功能的按钮，在鼠标按下去时进行录音，松开鼠标后停止录音，然后编写相应代码将其录音文件保存在test.wav中，百度API的作用便是识别录音文件中的语音信息，若其中的语音信息与某个预先设置的关键字匹配，则跳转到相应的界面，实现相应的功能。经过无数次的修改和调试，终于整个代码能够成功运行，可是在实际操作中，我发现百度语音并不能够很好的识别人类语言，我开始以为是中文所以不好识别，可是在换了英文后，识别结果依然不理想。这时，尽管我感到十分困惑，但也不愿就此罢休，在网页上搜索相关技术后，我决定不使用百度API，换用科大讯飞的语音识别，这一次，所有关于百度语音API的代码都不可用，只能删除后重来。换成科大讯飞的语音识别后，识别结果准确率高达百分之九十，我对于这个结果十分满意，也就此攻破了一个技术难点。

经过这次毕业设计的整个流程后，我对于软件的前台后端以及硬件部分都有了更深入的理解。在此，再以此感谢所有帮助我支持我的老师们，我相信在未来的人生路上，我也能不怕辛苦，不畏艰难的走下去，做出令人满意的作品！

# 参考文献

[1] Magnus Lie Hetland. Python基础教程[M]. 保存地：人民邮电出版社, 2018.

[2] 邵子扬. MicroPython入门指南[M]. 保存地：电子工业出版社, 2018.

[3] 卿勇. 智能家居发展及关键技术综述[J]. 软件导刊, 2017.

[4] 周冠方. 基于物联网的智能家居系统研究[D]. 保存地：湖北工业大学, 2017.

[5] 王金帅. 智能家居控制系统的设计[D]. 保存地：郑州大学, 2017.

[6] Kai Cheng, Zhi-Chao Niu, Robin C.Wang, Richard Rakowski, Richard Bateman. Smart Cutting Tools and Smart Machining: Development Approaches, and Their Implementation and Application Perspectives[J]. Chinese Journal of Mechanical Engineering, 2017.

[7] Wan Zhang, Min-Ping Jia, Lin Zhu, Xiao-An Yan. Comprehensive Overview on Computational Intelligence Techniques for Machinery Condition Monitoring and Fault Diagnosis[J]. Chinese Journal of Mechanical Engineering, 2017.

[8] 朱敏玲, 李宁. 智能家居发展现状及未来浅析[J]. 电视技术, 2015.

[9] James F. Kurose. 计算机网络：自顶向下方法[M]. 保存地：机械工业出社社, 2014.

[10] Rim Missaoui, Hussein Joumaa, Stephane Ploix, Seddik Bacha. Managing energy smart homes according to energy prices: Analysis of a building energy management system[J]. Energy Buildings, 2014.

# 附 录

TPYBOARD开发板的主要程序源码如下：

boot.py文件：

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

\* 名称 : do\_connect()

\* 功能 : 连接网络

\* 输入 : 无

\* 输出 : 无

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

def do\_connect():

import network

sta\_if = network.WLAN(network.STA\_IF)

ap\_if = network.WLAN(network.AP\_IF)

if ap\_if.active():

ap\_if.active(False)

if not sta\_if.isconnected():

print('connecting to network...')

sta\_if.active(True)

sta\_if.connect('NicoleiPhone','12345678') #wifi的SSID和密码

while not sta\_if.isconnected():

pass

print('network config:', sta\_if.ifconfig())

do\_connect()

mqtt.py文件：

import network

import socket

import urllib

import time

import dht

from simple import MQTTClient

from machine import Pin

import machine

import micropython

import json

d = dht.DHT11(machine.Pin(5))#声明用到类库中的函数，并设置参数

d.measure()#调用DHT类库中测量数据的函数

temp\_=str(d.temperature())#读取measure()函数中的温度数据

hum\_=str(d.humidity())#读取measure()函数中的湿度数据

#选择G4引脚

g4 = Pin(4, Pin.OUT, value=0)

# MQTT服务器地址域名为：183.230.40.39,不变

SERVER = "183.230.40.39"

#设备ID

CLIENT\_ID = "515484674"

#随便起个名字

TOPIC = b"TurnipRobot"

#产品ID

username='207760'

#产品APIKey:

password='ZZxwkh=B3YJ8BWFEQY2aCyHMcV4='

state = 0

#要上报的数据点

message = {

'datastreams':[

{

'id':'temperature',

'datapoints':[

{

'value':temp\_

}

]

},

{

'id':'humidity',

'datapoints':[

{

'value':hum\_

}

]

}

]

}

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

\* 名称 : pubdata(data)

\* 功能 : 将输入的数据转换成publish格式报文

\* 输入 : 输入数据

\* 输出 : 返回一个字节数组

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

def pubdata(data):

j\_d = json.dumps(data)

j\_l = len(j\_d)

arr = bytearray(j\_l + 3)

arr[0] = 1 #publish数据类型为json

arr[1] = int(j\_l / 256) # json数据长度 高位字节

arr[2] = j\_l % 256 # json数据长度 低位字节

arr[3:] = j\_d.encode('ascii') # json数据

return arr

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

\* 名称 : sub\_cb(topic, msg)

\* 功能 : 控制发光二极管的亮灭

\* 输入 : topic和msg

\* 输出 : 无

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

def sub\_cb(topic, msg):

global state

print((topic, msg))

if msg == b"on":

g4.value(1)

state = 1

print("1")

elif msg == b"off":

g4.value(0)

state = 0

print("0")

elif msg == b"toggle":

state = 1 - state

g4.value(state)

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

\* 名称 : Main()

\* 功能 : 主函数

\* 输入 : 无

\* 输出 : 无

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

def main(server=SERVER):

#端口号为：6002

c = MQTTClient(CLIENT\_ID, server,6002,username,password)

c.set\_callback(sub\_cb)

c.connect()

c.subscribe(TOPIC)

print("Connected to %s, subscribed to %s topic" % (server, TOPIC))

#publish报文上传数据点

c.publish('$dp',pubdata(message))

print('publish message:',message)

try:

while 1:

c.wait\_msg()

finally:

c.disconnect()

Django网页的主要程序源码如下：

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

import json

import sys

import base64

import time

import requests

from django.shortcuts import render

from django.http import JsonResponse

import hashlib

URL = "http://api.xfyun.cn/v1/service/v1/iat"

APPID = "5c613028"

API\_KEY = "a0108de414b143cd27290312b288103e"

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

\* 名称 : getHeader(aue, engineType)

\* 功能 : 科大讯飞语音听写，获取Header

\* 输入 : 音频编码，引擎类型

\* 输出 : 返回Header

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

def getHeader(aue, engineType):

curTime = str(int(time.time()))

param = "{\"aue\":\"" + aue + "\"" + ",\"engine\_type\":\"" + engineType + "\"}"

paramBase64 = str(base64.b64encode(param.encode('utf-8')), 'utf-8')

m2 = hashlib.md5()

m2.update((API\_KEY + curTime + paramBase64).encode('utf-8'))

checkSum = m2.hexdigest()

header = {

'X-CurTime': curTime,

'X-Param': paramBase64,

'X-Appid': APPID,

'X-CheckSum': checkSum,

'Content-Type': 'application/x-www-form-urlencoded; charset=utf-8',

}

return header

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

\* 名称 : getBody(filepath)

\* 功能 : 科大讯飞语音听写，获取内容Body

\* 输入 : 录音文件路径

\* 输出 : 返回Body

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

def getBody(filepath):

binfile = open(filepath, 'rb')

data = {'audio': base64.b64encode(binfile.read())}

return data

aue = "raw"

engineType = "sms8k"

audioFilePath = r"../AISmartHome\_SWIFT/django\_web/pcm/test.wav"

# HomePage

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

\* 名称 : index(request)

\* 功能 : 渲染索引界面

\* 输入 : request请求

\* 输出 : 返回index.html

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

def index(request):

return render(request, 'index.html')

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

\* 名称 : getdata(request)

\* 功能 : 获取温湿度传感器测得的数据

\* 输入 : request请求

\* 输出 : 返回get.html

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

def getdata(request):

device='515484674' #设备ID

apikey='ZZxwkh=B3YJ8BWFEQY2aCyHMcV4=' #APIKey

url='http://api.heclouds.com/devices/%s/datastreams' % device

print(url)

headers={

"api-key":apikey,

}

r=requests.get(url,headers=headers)

res = json.loads(r.content)

context = {}

context['data'] = res['data']

print(r.content)

return render(request, 'get.html', context)

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

\* 名称 : lighton(request)

\* 功能 : 控制发光二极管发光

\* 输入 : request请求

\* 输出 : 返回index.html

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

def lighton(request):

url = "http://api.heclouds.com/mqtt?topic=TurnipRobot"

querystring = {"device\_id":"515484674"}

payload = "on"

headers = {

'api-key': "ZZxwkh=B3YJ8BWFEQY2aCyHMcV4=",

}

response = requests.request("POST", url, data=payload, headers=headers, params=querystring)

print(response.text)

return render(request, 'index.html')

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

\* 名称 : lightoff(request)

\* 功能 : 控制发光二极管熄灭

\* 输入 : request请求

\* 输出 : 返回index.html

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

def lightoff(request):

url = "http://api.heclouds.com/mqtt?topic=TurnipRobot"

querystring = {"device\_id":"515484674"}

payload = "off"

headers = {

'api-key': "ZZxwkh=B3YJ8BWFEQY2aCyHMcV4=",

}

response = requests.request("POST", url, data=payload, headers=headers, params=querystring)

print(response.text)

return render(request, 'index.html')

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

\* 名称 : upload(request)

\* 功能 : 录音音频上传及语音控制相应应用

\* 输入 : request请求

\* 输出 : 返回index.html

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

def upload(request):

if request.method == 'GET':

return render(request, 'index.html')

audioData = request.FILES.get('audioData')

with open('../AISmartHome\_SWIFT/django\_web/pcm/test.wav', 'wb') as file:

file.write(audioData.read())

file.close()

result = ""

try:

r = requests.post(URL, headers=getHeader(aue, engineType), data=getBody(audioFilePath))

result = r.content.decode('utf-8')

except Exception as e:

return JsonResponse({'status':'Failed,Please try again'})

print(result)

json\_obj={}

message=""

try:

json\_obj=json.loads(result)

message = json\_obj['data']

except Exception as e:

print(e)

print(json\_obj)

if("开" in message or "开灯" in message):

lighton(request)

return JsonResponse({'status':'lighton'})

elif("关" in message or "关灯" in message):

lightoff(request)

return JsonResponse({'status':'lightoff'})

elif("数据" in message or "获取信息" in message):

getdata(request)

return JsonResponse({'status':'getdata'})

else:

print(message)

return JsonResponse({'status':'Failed'})

return JsonResponse({'foo':'bar'})

成都东软学院

毕业设计（论文）原创承诺书

1、本人承诺：所提交的毕业设计（论文）是认真学习理解学校的《毕业设计（论文）工作规范》后，在教师的指导下，独立地完成了任务书中规定的内容，不弄虚作假，不抄袭别人的工作内容。

2、本人在毕业设计（论文）中引用他人的观点和研究成果，均在文中加以注释或以参考文献形式列出，对本文的研究工作做出重要贡献的个人和集体均已在文中注明。

3、在毕业设计（论文）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

4、本人完全了解学校关于保存、使用毕业设计（论文）的规定，即：按照学校要求提交论文和相关材料的印刷本和电子版本；同意学校保留毕业设计（论文）的复印件和电子版本，允许被查阅和借阅；学校可以采用影印、缩印或其他复制手段保存毕业设计（论文），可以公布其中的全部或部分内容。

5、本人完全了解《毕业（设计）论文工作规范》关于“学生毕业设计（论文）出现购买、他人代写、或者抄袭、剽窃等作假情形的，取消其学位申请资格；已经获得学位的，依法撤销其学位。取消学位申请资格或者撤销学位者，从处理决定之日起3年内，学校不再接受学生学位申请”的规定内容。

6、本人完全了解《学生手册》中关于在“毕业设计（论文）等环节中被认定抄袭他人成果者”不授予学士学位，并且“毕业学年因违纪受处分影响学位的学生不授予学士学位，并且无学士学位申请资格”的规定内容。

以上承诺的法律结果、不能正常毕业及其他不可预见的后果由学生本人承担！

学生本人签字： 年 月 日