**AAA**

**软件设计说明书**

**SDD-107**

**V2.0**

分工说明

|  |  |  |
| --- | --- | --- |
| 小组名称 | AA | |
| 学号 | 姓名 | 本文档中主要承担的工作内容 |
| 16061073 | 刘禹廷 | 范围、体系结构、接口、详细设计 |
| 16061079 | 柴林政 | 接口设计, 数据库设计, 开发环境, 运行环境 |
| 16061042 | 杨枫 | [服务器端体系结构](bookmark://_Toc6770406),[服务器模块](bookmark://_Toc6770447) |
| 16061049 | 王政 | 需求概述、用例图 |
| 15061130 | 赵振宁 | 审核 |

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 审核人 | 版本说明 |
| V1.0 | 2019-4-21 | 刘禹廷、杨枫、王政、柴林政 | 赵振宁 | 文档初版 |
| V1.1 | 2019-4-22 | 刘禹廷 |  | 按照修改意见进行了修改 |
| V2.0 | 2019-6-8 | 刘禹廷 | 柴林政 | 根据项目对文档进行了修改 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1. 范围 1](#_Toc10902121)

[1.1 项目概述 1](#_Toc10902122)

[1.2 文档概述 1](#_Toc10902123)

[1.3 术语和缩略词 1](#_Toc10902124)

[1.4 引用文档 2](#_Toc10902125)

[2. 需求概述 2](#_Toc10902126)

[2.1 用例总图： 2](#_Toc10902127)

[2.2 安装/解除系统用例图： 2](#_Toc10902128)

[2.3 遇到问题用例图： 2](#_Toc10902129)

[2.4 通过因特网访问系统用例图： 2](#_Toc10902130)

[3. 体系结构设计 3](#_Toc10902131)

[3.1 总体结构 3](#_Toc10902132)

[3.1.1 Arduino端体系结构 4](#_Toc10902133)

[3.1.2 树莓派端体系结构 6](#_Toc10902134)

[3.1.3 服务器端体系结构 7](#_Toc10902135)

[3.2 关键问题及解决方案 8](#_Toc10902136)

[3.2.1 Arduino端 8](#_Toc10902137)

[3.2.2 树莓派端 8](#_Toc10902138)

[3.2.3 服务器端 9](#_Toc10902139)

[4. 接口设计 9](#_Toc10902140)

[4.1 系统用户界面接口 9](#_Toc10902141)

[4.1.1 用户注册接口 9](#_Toc10902142)

[4.1.2 用户登录接口 10](#_Toc10902143)

[4.1.3 修改资料接口 10](#_Toc10902144)

[4.1.4 修改房屋设置 10](#_Toc10902145)

[4.1.5 添加房间 11](#_Toc10902146)

[4.1.6 查看房间信息 11](#_Toc10902147)

[4.1.7 查看设备信息 11](#_Toc10902148)

[4.1.8 创建场景 12](#_Toc10902149)

[4.1.9 启用场景 12](#_Toc10902150)

[4.1.10 禁用场景 13](#_Toc10902151)

[4.2 系统的软硬件接口 13](#_Toc10902152)

[4.2.1 Arduino传感器接口 13](#_Toc10902153)

[4.2.2 Arduino数据传输接口 14](#_Toc10902154)

[4.2.3 管理设备 14](#_Toc10902155)

[4.2.4 更新服务器与本地信息 16](#_Toc10902156)

[5. 数据库设计 17](#_Toc10902157)

[5.1 用户(user) 17](#_Toc10902158)

[5.2 房间(room) 17](#_Toc10902159)

[5.3 设备(device) 17](#_Toc10902160)

[5.4 设备参数(deviceArg) 18](#_Toc10902161)

[5.5 场景(scene) 18](#_Toc10902162)

[5.6 触发条件(trigger) 18](#_Toc10902163)

[5.7 触发动作(action) 18](#_Toc10902164)

[5.8 智能设备列表(本地数据库) 19](#_Toc10902165)

[5.9 传感器设备列表(本地数据库) 19](#_Toc10902166)

[5.10 设备信息(本地数据库) 19](#_Toc10902167)

[5.11 数据库之间的关系如图所示 20](#_Toc10902168)

[6. 详细设计 20](#_Toc10902169)

[6.1 报警模块 20](#_Toc10902170)

[6.2 管理模块（树莓派） 22](#_Toc10902171)

[6.2.1 HomeBridge 22](#_Toc10902172)

[6.2.2 传感器管理与本地数据库 22](#_Toc10902173)

[6.2.3 场景算法 22](#_Toc10902174)

[6.2.4 具体用例说明 23](#_Toc10902175)

[6.3 服务器模块 25](#_Toc10902176)

[7. 运行与开发环境 26](#_Toc10902177)

[7.1 运行环境 26](#_Toc10902178)

[7.2 软件环境 26](#_Toc10902179)

[8. 需求可追踪性说明 26](#_Toc10902180)

# 范围

## 项目概述

智能家庭是近几年中越来越火热的一个话题。科技以人为本，本着使人们的家居生活更便捷舒适的目的，项目计划制作一种带有入侵报警功能的家庭智能网关。具体功能包括连接智能设备并加以控制，和使用人体传感器等进行入侵检测报警。

从用户角度出发，设备可以实时获取家中状态，包括温度、湿度等等信息。在用户外出家中无人的情况下，可以进行屋内的入侵检测和火焰报警。设备可以根据用户设定的自动化方案依据传感器数据和阈值更改连接的智能化设备状态，如用户晚上将要到家时打开客厅灯光等。

## 文档概述

文档会在项目开发计划的基础上确定系统的主要功能，非功能性需求和应用场景。

主要功能包括业务需求、功能需求和数据需求。使用用例驱动建立需求模型，进一步确定功能以及实现功能所需要的数据需求。

本项目需求文档会与项目开发计划一同指导项目的开发进程，根据项目开发计划确定的时间表以及里程碑计划映射逐步实现本文档呈现的需求和功能。

## 术语和缩略词

|  |  |  |
| --- | --- | --- |
| 缩写 | 全称 | 说明 |
| HomeBridge | - | Homebridge is a lightweight NodeJS server you can run on your home network that emulates the iOS HomeKit API. It supports Plugins, which are community-contributed modules that provide a basic bridge from HomeKit to various 3rd-party APIs provided by manufacturers of "smart home" devices. |
| Arduino | - | Arduino是一款便捷灵活、方便上手的开源电子原型平台。包含硬件（各种型号的Arduino板）和软件（Arduino IDE)。 |
| Rep | Respberry树莓派 | 一款基于[ARM](https://baike.baidu.com/item/ARM)的微型电脑主板，以SD/MicroSD卡为内存硬盘，卡片主板周围有1/2/4个USB接口和一个10/100 以太网接口（A型没有网口），可连接键盘、鼠标和网线，同时拥有视频模拟信号的电视输出接口和HDMI高清视频输出接口，以上部件全部整合在一张仅比信用卡稍大的主板上，具备所有PC的基本功能只需接通电视机和键盘，就能执行如电子表格、文字处理、玩游戏、播放高清视频等诸多功能。 Raspberry Pi B款只提供电脑板，无内存、电源、键盘、机箱或连线。 |

Figure 1 1.3 术语和缩略词

## 引用文档

1. 《AAA项目开发计划(SPD-107)》 v2.0 2019.6.8
2. 《AAA项目需求规格说明书》 v2.0 2019.6.8

# 需求概述

## 用例总图：



Figure 2 用例总图

这是总体的用例图。首先，系统管理员可以添加传感器，通过Arduino进行连接，可以进行各种参数的修改。用户有一个安装/解除报警的开关，用来控制报警是否开启。用户可以通过web进行房间定制、设备管理、场景管理、解除报警、控制设备等一系列工作。报警信息推送给用户，用户可以定时开关台灯。

## 安装/解除系统用例图：

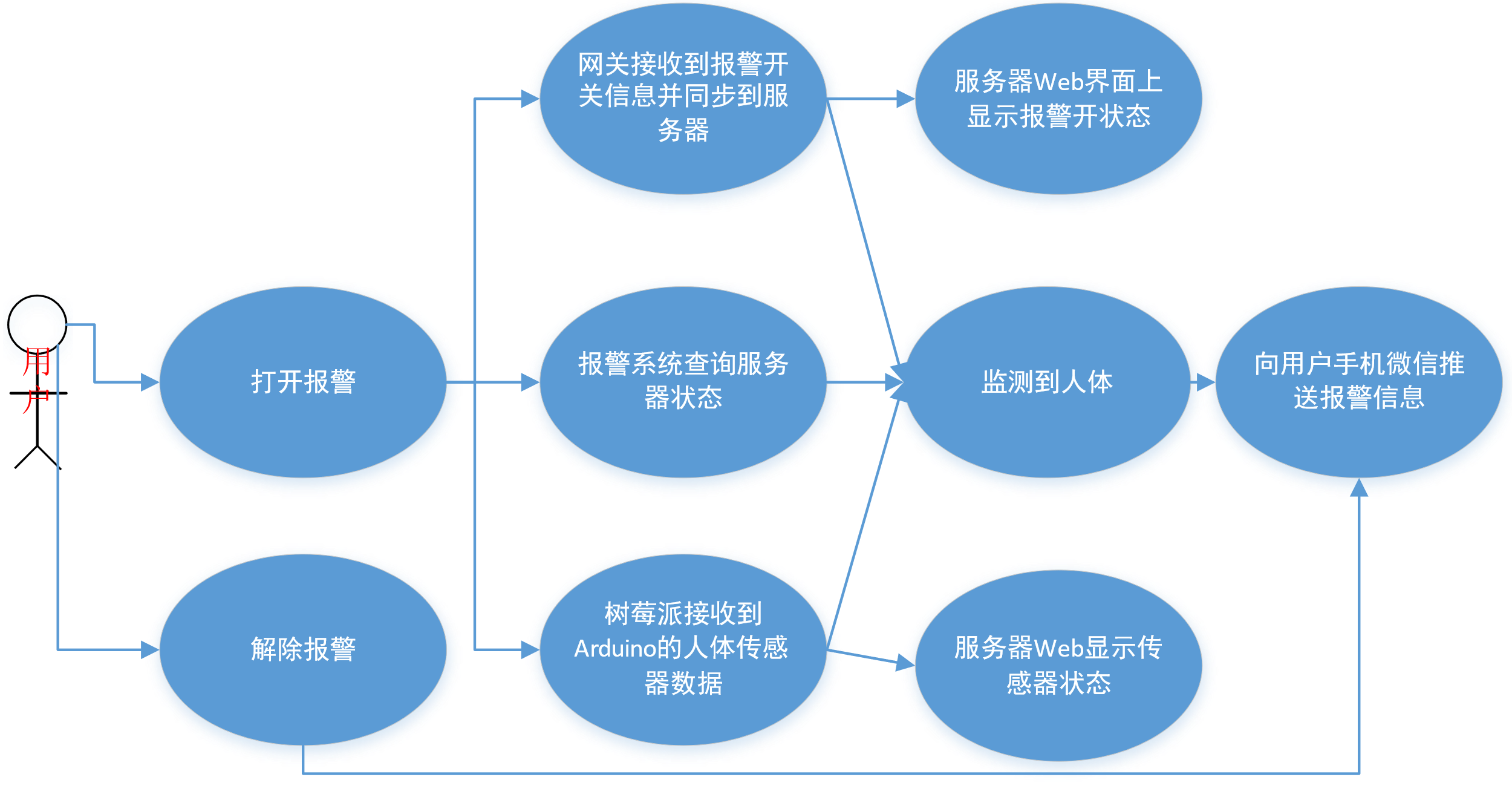


Figure 3 安装/解除系统用例图

用户可以打开或解除报警，在打开报警的状态下，网关接收到报警开关信息，并同步到服务器，服务器web界面上显示报警开状态。同时报警系统查询服务器状态，如果报警开状态，树莓派又接收到Arduino的人体传感器数据，服务器显示传感器监测到人体，就向用户手机微信推送报警信息，用户姐收到信息后可以解除报警来停止推送信息。

## 遇到问题用例图：

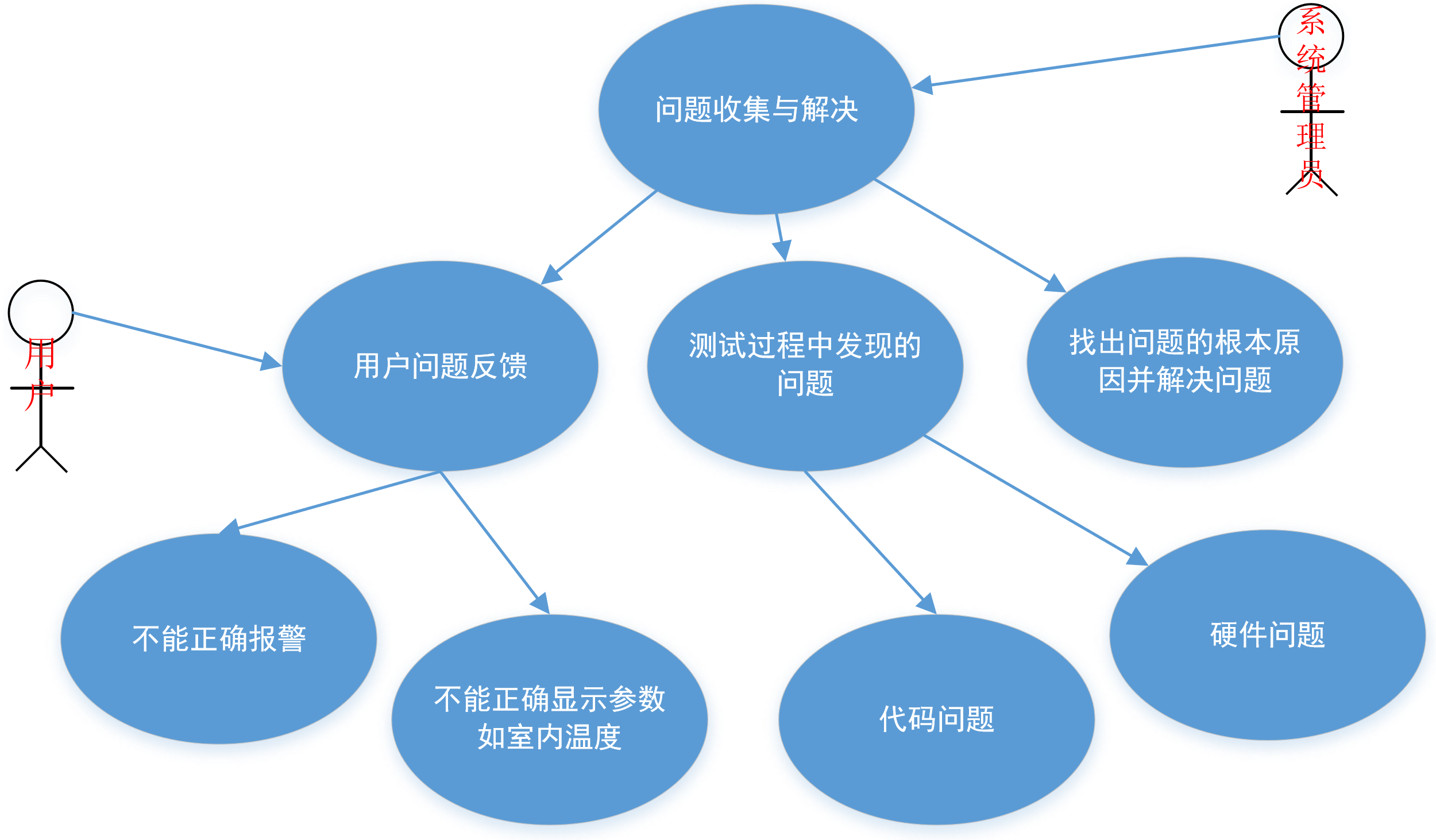


Figure 4遇到问题用例图

用户遇到问题，如发生了火灾，但手机没有发出相应的报警信息；室内温度显示值过大或过小等。用户将这些问题反馈给系统管理员，同时系统管理员会不断地对系统进行测试，在测试中也可能发现问题，如代码问题或硬件问题。系统管理员将这些问题收集起来，找出问题的根本原因并解决问题。

## 通过因特网访问系统用例图：

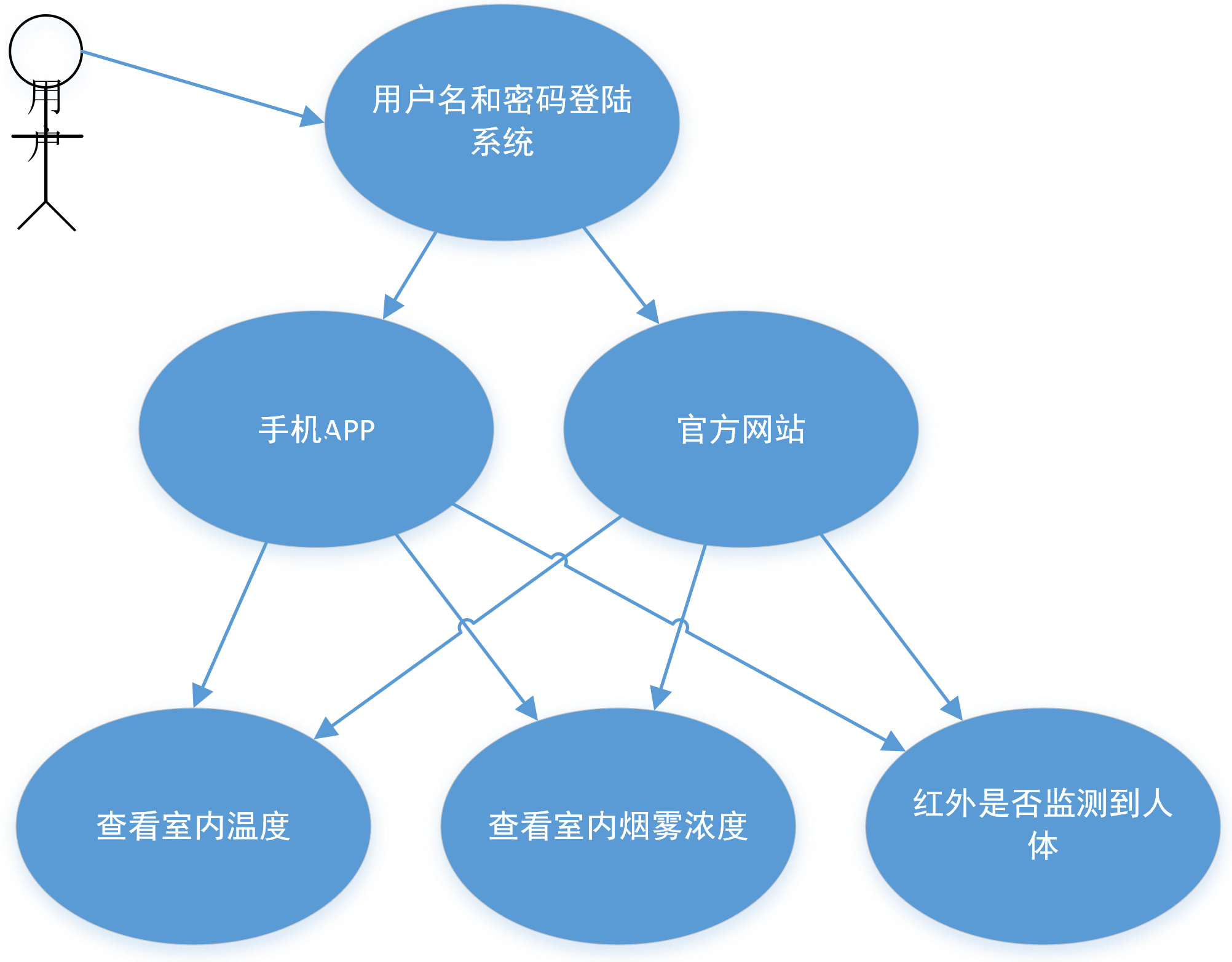


Figure 5 通过因特网访问系统用例图

用户可以用用户名和密码登陆系统，可以在手机APP上登陆，也可以在官方网站上登陆。登陆之后就可以查看各种信息，如室内温度大小，室内烟雾浓度大小，以及红外传感器是否检测到人体。

# 体系结构设计

## 总体结构

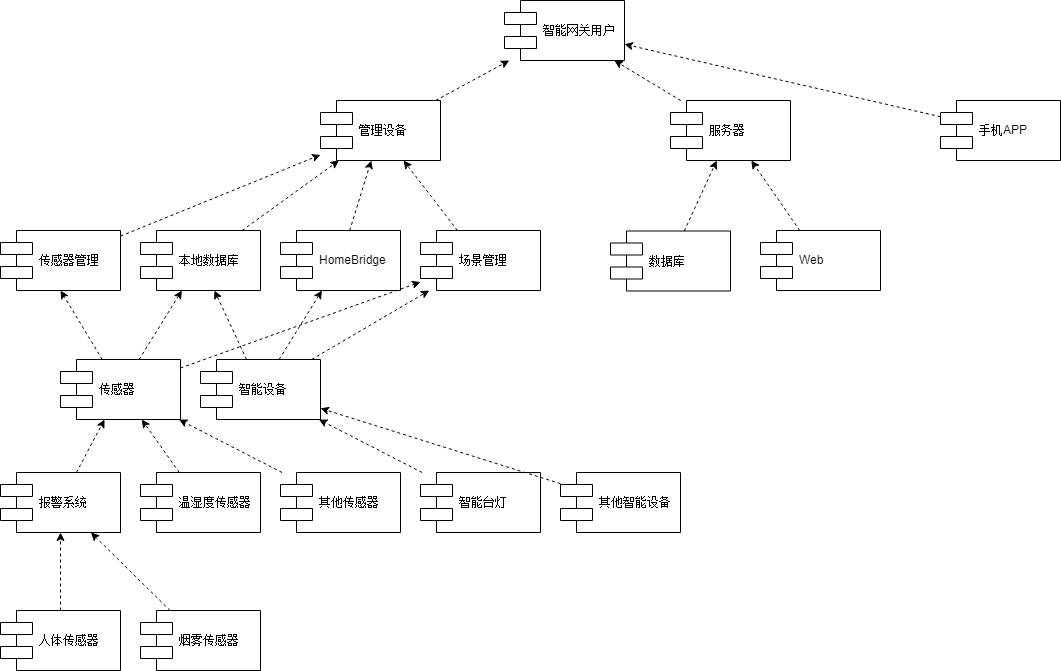


Figure 6 总体构件图

上图中，树莓派是房屋内所有智能硬件的管理设备，Arduino负责屋内的安全系统。报警系统与管理设备之间通过WiFi进行数据交换，通过管理设备设定报警系统的报警阈值与时机。管理设备（树莓派）与服务器进行传感器智能设备和场景设置的数据交换。服务器提供Web界面供用户查看传感器数据、设备状态、场景搭建管理。

### Arduino端体系结构

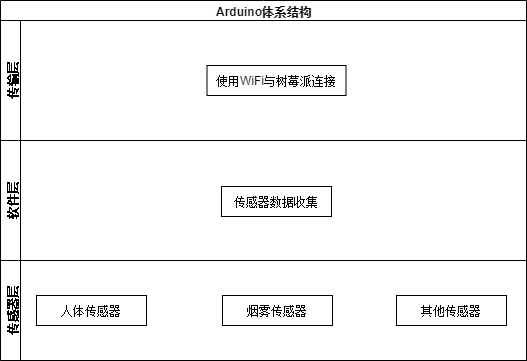


Figure 7 Arduino体系结构

上图为Arduino报警装置的体系结构，传感器部分为硬件，Arduino在一定的时间间隔内通过软件进行传感器数据的刷新收集，并将其打包通过WiFi传递给管理设备（树莓派）。



Figure 8 Arduino状态图

上图表示报警系统在入侵检测时的状态转换。在正常工作时，报警系统每隔一定时间会收集与人体检测有关传感器的数据，将其打包发给管理设备。

### 树莓派端体系结构

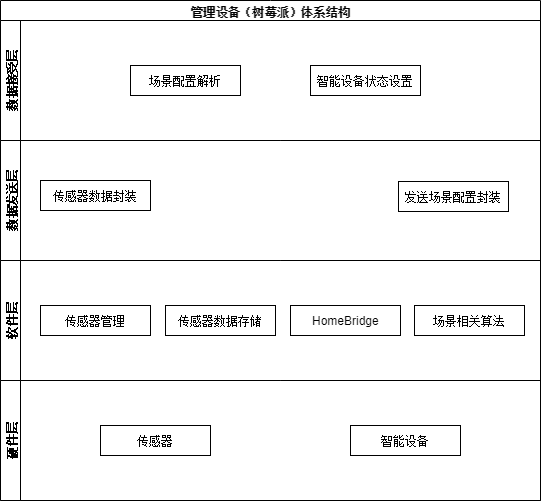


Figure 9 管理设备体系结构

在管理设备的设计中，将传感器硬件与智能设备硬件分成两个部分，分别管理。传感器数据使用数据库存储，智能设备使用HomeBridge管理，场景相关算法会将二者进行统合。在与服务器进行通信时，发送设备列表，传感器数据，接收服务器发来的场景配置并与本地配置进行比较和刷新，接收智能设备状态设定的刷新。具体内容将在章节6进行讲解。

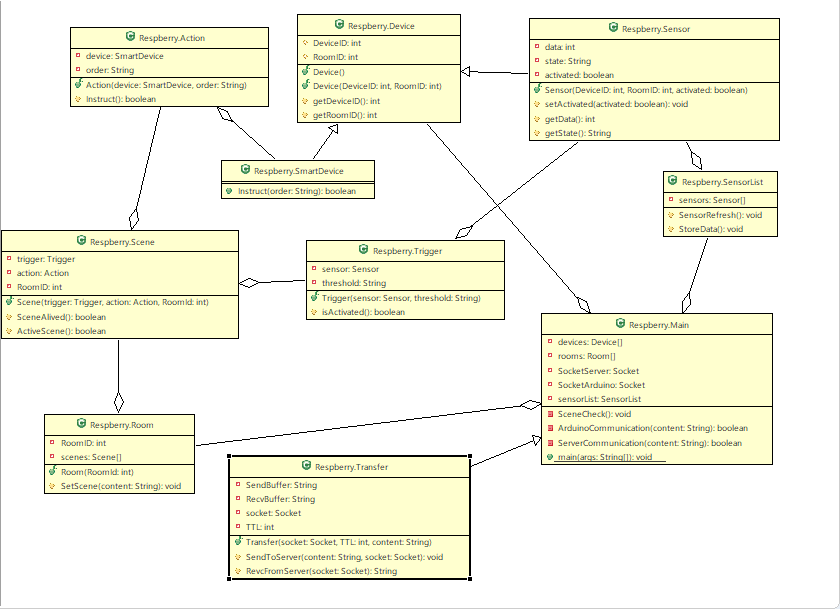


Figure 10 管理设备类图

上图为管理设备（树莓派）的大致类图。主类是Main，用于整体控制。通信类为Transfer，用于与Arduino或服务器进行通信。设备相关的父类为Device，Sensor和SmartDevice为其子类。场景相关的类有Scene，Trigger和Action，用于场景的保存，触发器判断和执行动作。

### 服务器端体系结构

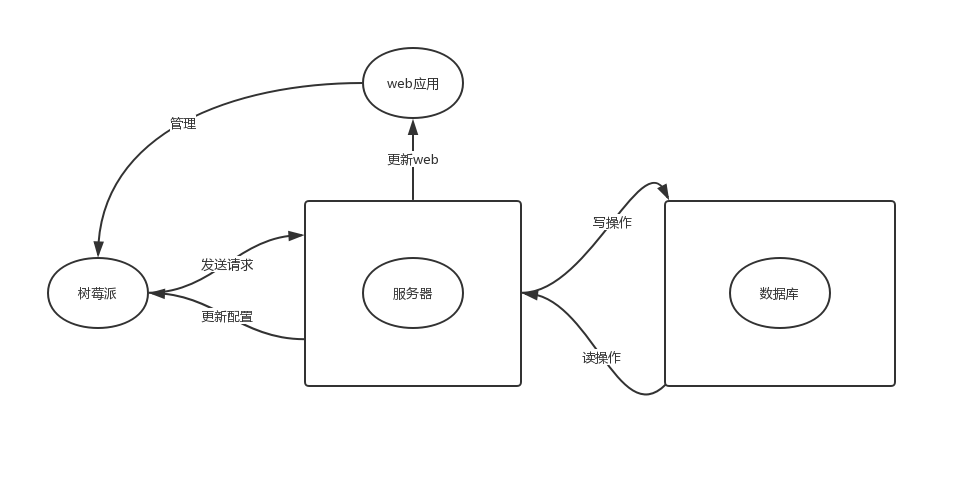


Figure 服务器体系结构

用户通过web应用进行管理树莓派，随后树莓派向服务器发送请求，服务器对树莓派发送来的请求进行解析，向数据库做出具体的操作。之后服务器向树莓派发送更新配置的具体指令，同时对web应用进行重新渲染。

## 关键问题及解决方案

### Arduino端

1. Arduino报警系统传感器连接

使用面包板和条线，系统管理员设置间隔时间进行传感器数据的读取。

1. Arduino报警系统与树莓派的无线连接

使用esp8266芯片进行无线网连接，使用HTTP协议与管理设备之间进行数据传输，Arduino端为客户端，管理设备（树莓派）端为服务端，由Arduino发起连接。具体传输内容将在章节6.1中进行详细说明。

### 树莓派端

1. 树莓派对传感器设备信息的存储

在管理设备上，报警系统作为一个传感器存在。

1. 智能设备相关

智能设备使用开源的HomeBridge工具进行管理。

1. 树莓派与服务器的数据交互

管理设备（树莓派）管理本地数据库，数据库中针对每一个传感器都建有一张表，存储读取到的传感器数据；针对所有智能设备建有一张列表。管理设备将传感器数据，智能设备列表封装为json格式文本发送给服务器，由服务器端传回修改状态后智能设备列表和新场景列表。管理设备将收到的场景列表与本地进行对比更新。

1. 树莓派端场景实现

单独的场景是类Scene的一个实例，存放在数组中，根据一定的时间间隔进行遍历，发现有符合场景触发器的条件，就进行相应动作。

### 服务器端

1. Web端创建智能设备场景

使用类似IFTTT的界面，根据服务器端存储的传感器列表选择触发器，根据服务器端存储的智能设备列表选择动作。将其存储到数据库中，在管理设备发起连接时将其发送给管理设备。

1. 报警时通知

管理设备发送的内容中存在有关于报警系统状态的字段，服务器端会根据这一字段通知用户，用户可以设置字段中的一部分内容暂停报警，用户的自定义设置将会传回管理设备。

# 接口设计

用户界面接口都以http协议方式进行信息传递。

## 系统用户界面接口

### 用户注册接口

1. 描述: 用户使用app或网站注册, 向服务器端发出POST请求
2. 接口名: register()
3. 请求参数说明:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| username | string | 注册的昵称(必填) |
| email | string | 用户邮箱, 用户用来登录的唯一识别标识(必填) |
| phone | string | 手机号码(选填) |
| password | string | 用户登录的密码(必填) |

Figure 请求参数

### 用户登录接口

1. 描述: 进行用户登录, 向服务端发送POST请求
2. 接口名: login()
3. 请求参数说明:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| email | string | 用户登录的账号(必填) |
| password | string | 用户登录的密码(必填) |

Figure 请求参数

### 修改资料接口

1. 描述: 用户进行修改信息的接口, 向服务端发送POST请求
2. 接口名: updateUserInfo()
3. 请求参数说明:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| email | string | 用户登录的账号(选填) |
| username | string | 用户昵称(选填) |
| phone | string | 手机号码(选填) |
| password | string | 用户登录的密码(选填) |

Figure 请求参数

### 修改房屋设置

1. 修改房屋的相关信息
2. 接口名:updateHouseInfo(), 发送POST请求
3. 请求参数说明:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| housename | string | 房屋名称 |
| houseArea | int | 房屋面积 |
| houseType | string | 房屋类型(楼房, 庭院, 别墅) |

Figure 请求参数

### 添加房间

1. 描述: 添加房间, 发送POST请求
2. 接口名: addRoom()
3. 请求参数说明:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| roomName | string | 房间的名称 |

Figure 请求参数

### 查看房间信息

1. 描述: 查看房间的详细信息, 发送GET请求
2. 接口名: queryRoomInfo()
3. 请求参数说明:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| roomId | int | 通过房间id查询房间 |

Figure 请求参数

1. 返回信息:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| roomId | int | 房间id |
| roomName | string | 房间名称 |

Figure 返回信息

### 查看设备信息

1. 描述: 查看设备相应的信息, 发送GET请求
2. 接口名: queryDeviceInfo()
3. 请求参数说明:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| deviceId | int | 设备id |

Figure 请求参数

1. 返回信息:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| deviceId | int | 设备id |
| deviceInfo | int | 设备相关信息 |

Figure 返回信息

### 创建场景

1. 描述: 创建场景, 发送POST请求
2. 接口名: createScene()
3. 请求参数说明:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| sceneName | string | 场景名称 |

Figure 请求参数

### 启用场景

1. 描述: 启用房间中的场景, 向服务端发送POST请求
2. 接口名: enableScene()
3. 请求参数说明:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| roomId | int | 房间id |
| sceneId | int | 场景id |
| deviceId | array int | 场景所需设备id |

Figure 请求参数

### 禁用场景

1. 描述: 禁用房间中的场景, 向服务端发送POST请求
2. 接口名: disableScene()
3. 请求参数说明:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| roomId | int | 房间id |
| sceneId | int | 场景id |

Figure 请求参数

## 系统的软硬件接口

### Arduino传感器接口

1. Arduino连接WiFi

描述：Arduino使用AT指令与esp8266进行设定连接WiFi的函数。

接口名：setwifi()

输入参数：无

返回参数：无

1. 人体传感器数据获取

描述：Arduino从人体传感器读取数据并返回

接口名：pir.active()

输入参数：无

返回参数：

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| hasBody | bool | 如果有探测到人体,返回true, 否则返回false |

Figure 返回参数

1. 烟雾传感器数据获取

描述：Arduino从火焰传感器读取数据并返回

接口名：analogRead()

输入参数：smoke

返回参数：

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| Val | Int | 返回烟雾浓度 |

Figure 返回参数

### 管理设备

1. 与Arduino的数据交换

描述：添加智能设备的功能

接口名：requests.get()

输入参数：

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| url | string | Arduino端的ip和开启的httpserver接口 |
| Timeout | int | 超时时间 |

Figure 输入参数

返回参数：

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| Text | string | Arduino端返回的json格式数据 |

Figure 27 返回参数

1. 与服务器的数据交换

描述：与服务器之间的数据同步

接口名：requests.post()

请求参数说明：

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| url | String | 服务器url |
| Headers | list | 头部 |
| Data | String | 发送给服务器的json格式字符串 |

Figure 请求参数

返回信息：

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| Text | String | 返回设备信息json格式字符串 |

Figure 返回信息

1. 场景遍历

描述：遍历已有场景，使用场景对象的isActivated方法进行动作。

接口名：SceneLoop()

请求参数说明：

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| sceneList | List | 定义的场景列表 |

Figure 请求参数

### 更新服务器与本地信息

1. 描述: 树莓派向服务器发送更新的数据, 并从服务器端得到最新的配置文件信息, 向服务端发送POST请求, 数据以json格式发送
2. 接口名: updateLocal()
3. 请求参数说明:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| email | string | 用户邮箱(客户端标识) |
| update\_info | json | 更新的本地信息 |

Figure 请求参数

1. 返回参数说明:

|  |  |  |
| --- | --- | --- |
| 参数字段 | 类型 | 说明 |
| email | string | 用户邮箱(客户端标识) |
| time | date | 更新时间 |
| update\_config | json | 更新的配置信息 |

Figure 返回参数

# 数据库设计

## 用户(user)

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 约束 | 说明 |
| userId | int | primary key | 用户id |
| username | varchar(20) | not null | 用户昵称 |
| password | varchar(20) | not null | 用户密码 |
| email | varchar(20) | not null | 用户邮箱 |
| phone | varchar(15) |  | 用户手机 |

Figure 33 用户

## 房间(room)

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 约束 | 说明 |
| roomId | int | primary key | 房间id |
| roomname | varchar(20) | not null | 房间名称 |

Figure 34 房间

## 设备(device)

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 约束 | 说明 |
| deviceId | int | primary key | 设备id |
| deviceType | int | not null | 设备类型 |
| roomId | int | not null | 设备所在房间id |
| enableUse | int | not null | 设备能否使用 |
| argId | int | Not null | 设备参数id |

Figure 35 设备

## 设备参数(deviceArg)

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 约束 | 说明 |
| deviceid | int | Primary key | 设备id |
| argId | int | primary key | 设备参数id |
| args | array | Not null | 设备参数 |

Figure 设备参数

## 场景(scene)

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 约束 | 说明 |
| sceneId | int | primary | 场景id |
| sceneName | var(20) | not null | 场景名称 |
| triggerId | int | not null | 触发条件id |
| actionId | int | not null | 触发的动作id |
| enableScene | var(20) | not null | 场景是否生效 |

Figure 37 场景

## 触发条件(trigger)

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 约束 | 说明 |
| triggerId | int | Primary | 触发条件id |
| type | int | Not null | 触发类型 |
| arg | int | Not null | 触发参数 |

Figure 触发条件

## 触发动作(action)

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 约束 | 说明 |
| actionId | int | Primary | 触发动作id |
| type | int | Not null | 动作类型 |
| arg | int | Not null | 触发参数 |

Figure 触发动作

## 智能设备列表(本地数据库)

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 约束 | 说明 |
| deviceId | int | Primary | 设备id |
| type | int | Not null | 设备类型 |

Figure 智能设备列表

## 传感器设备列表(本地数据库)

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 约束 | 说明 |
| deviceId | int | Primary | 设备id |
| type | int | Not null | 设备类型 |

Figure 传感器设备列表

## 设备信息(本地数据库)

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 约束 | 说明 |
| deviceId | int | Primary | 设备id |
| args | array | Not null | 设备参数信息 |
| time | date | Not null | 信息记录时间 |

Figure 设备信息

## 数据库之间的关系如图所示



Figure 43 ER图

# 详细设计

系统总体由报警系统，管理设备，服务器模块和手机APP四部分组成，构件之间的联系由上文的Figure 7进行说明。

## 报警模块

报警系统硬件部分由Arduino组成，软件部分由Arduino，管理设备，服务器端三部分构成。

Arduino设备使用面包板和跳线与传感器进行连接，通过系统管理员设定的时间间隔进行传感器设备读取。

传感器数据收集算法如下：

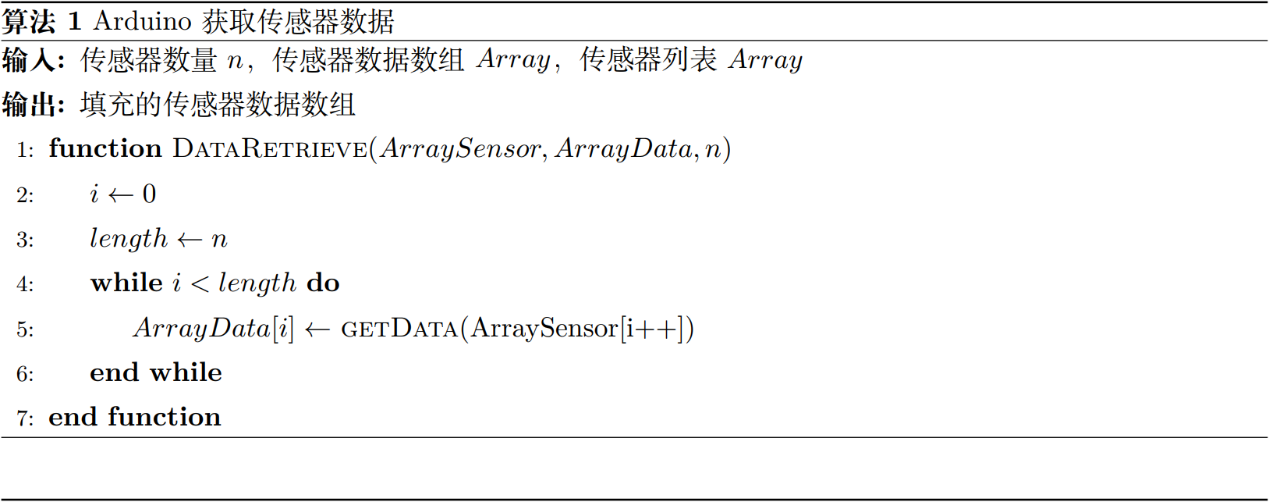


Figure 44 获取传感器数据

收集结束后将信息组成json格式，在管理设备使用http请求时发送给管理设备，管理设备将这些信息进行收集并刷新存储设备信息的数据库，在与服务器进行数据同步时发送给服务器，在服务器端使用上传来的传感器信息和报警开关信息来具体判断是否给用户推送报警信息。

## 管理模块（树莓派）

管理设备的类图已经在章节3.1.2中进行过展示。

管理设备的软件层面主要由四部分构成：HomeBridge、传感器管理、本地数据库和场景算法。这几部分使用和维护几张json表格，这些表格将在后面进行详细说明。

传感器管理部分和场景相关部分是我们要具体编写的代码部分，其将与本地数据库协同处理传感器相关数据。

### HomeBridge

HomeBridge是一款开源的串接服务，使不支持苹果HomeKit的智能设备能够统一使用苹果设备进行管理。其github网页是<https://github.com/nfarina/homebridge>。

### 传感器管理与本地数据库

在管理设备上，每一个传感器在软件层面都是本地存储的json表Devices里sensors数组中的一项，其传感器数据为在表项中进行存储。

本地数据库为MySql在树莓派上的搭建。其作用为存储：①传感器设备列表；②智能设备列表；③一段时间内的所有传感器数据，每个传感器配有一张专用的数据库表；④场景配置列表。

上述列表按照设定，将在一定时间内由管理设备发起的连接中与服务器端的数据库进行同步。管理设备将上述内容和即时传感器数据打包为json格式通过http协议发送给服务器，服务器将传感器启用信息和场景列表发送给管理设备。双方的数据传输将在后文有具体描述。

### 场景算法

场景算法是本产品的核心。与场景有关的部分有：①本地场景存储数据库和json文件Scenes.json；②场景触发器判断；③触发动作实现；④本地场景与服务器端的同步；⑤服务的存储。

在管理设备上，一个具体的场景是Scenes.json表中的一个项，其拥有触发器、动作的属性，其中触发器拥有读取服务，条件和阈值的属性；动作拥有控制服务，下达的指令的属性。

触发器的具体工作方式：SceneLoop()方法会持续读取Scenes.json中存储的额场景列表。根据其中存储的读取服务标识readserviceid，条件以及阈值创建trigger对象，使用动作服务标识controlserviceid和动作创建action对象。此后使用trigger对象的isTriggered()方法判断是否被触发，若触发，则使用action对象的act()方法向设备发送指令。

此外，SceneLoop()方法还负责维护Scenes.json这张表，通过requests库与服务器进行同步，获取远端场景列表后将其存储到Scenes.json中。

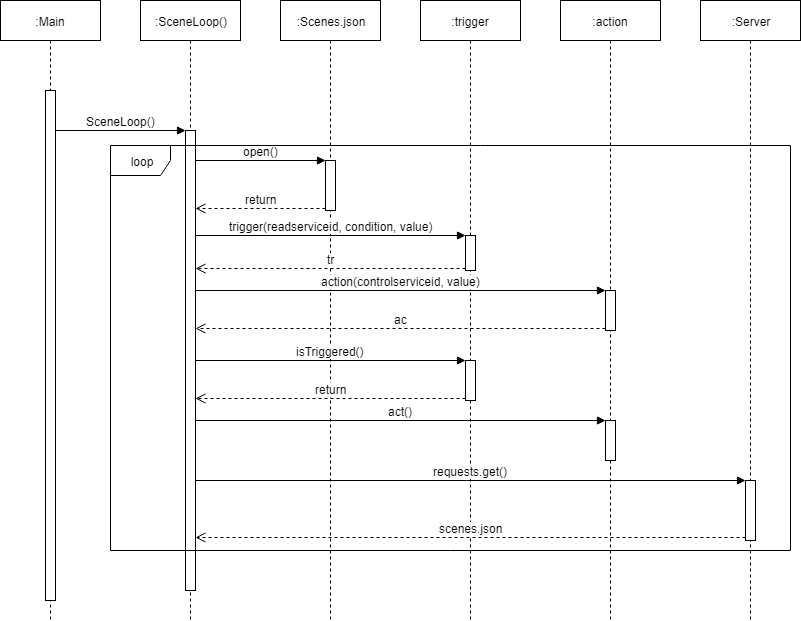


Figure 45 场景遍历

### 表格与数据同步说明

数据同步是报警，Web端信息显示和Web端设备控制的主要实现。系统中主要使用三张json表格：Devices.json，Scenes.json，Services.json

Devices.json中将设备分为传感器sensors和可控制设备（智能设备）accessories分为两类，sensors列表中的项格式如下面所示：

{

"iid": 10,

"name": "\u6e29\u5ea6\u4f20\u611f\u5668",

"aid": 4,

"type": {

"currentvalue": 24,

"valuetype": 0

}

}

其中aid为HomeBridge中对这一设备的具体标识，iid为HomeBridge中对设备字段的标识，在上面的项中，iid=10代表这一传感器具体数值的获取标识是10。name是显示的名字，type为一个字典，包括目前的值和值类型，值类型为0时代表其值为int类型，值类型为1时代表其值为bool类型。

Devices.json中accessories列表中的项格式如下：

{

"name": "\u5c0f\u7c73\u53f0\u706f-1",

"iids": [

{

"iid": 10,

"currentvalue": false,

"valuetype": 1

}

],

"aid": 2

}

其中名字为现实的名字，aid同样是HB中对这一设备的具体标识，iids字段是一个列表，其中包含了HB中关于这一设备的属性字段，在上述项中iid为10时代表这一设备的电源开关，currentvalue存储了其当前值，valuetype与sensors中的valuetype含义相同。

Devices.json在管理设备与服务器之间第一次同步时将所有设备信息全部发送给服务器，而在后续的同步中将传感器信息全部同步，但只将同步间隔中属性有更改的accessories发送给服务器。

Sences.json中存储了场景列表，其具体的项如下所示：

{

"trigger" : {

"readserviceid" : 40010,

"condition" : 1,

"value" : 20

},

"action" : {

"controlserviceid" : 80010,

"value" : true

}

}

其中trigger为触发，action为动作。trigger中，readserviceid是Service.json中特定项的aid\*10000+iid，condition用作条件判断，其值为1时代表条件可以有小于等于和大于三种，其值为0时代表只能由等于一种条件，value的值为阈值。action中存储的controlserviceid与上面的id类似，value字段为动作的指令。

Services.json中存储了所有场景相关的服务，分为readservices和controlservices。在服务中，controlservice作为可以具体控制某一设备的服务存在，readservice作为获取设备某一属性的服务，值得一提的是某一设备的开关等属性既可以作为控制服务，其属性值也可以作为读取服务。controlservices中的一项如下所示：

{

"iid": 10,

"allowed\_value": 0,

"name": "\u5c0f\u7c73\u53f0\u706f-1\u5f00\u5173",

"aid": 2

}

Readservices中的一项如下所示：

{

"iid": 10,

"name": "\u6e29\u5ea6\u4f20\u611f\u5668",

"aid": 4,

"allowed\_condition": 1

}

上述的表格中，除了Scenes.json由SceneLoop()方法与服务器进行同步，另外两个表都是由DataToServer.py进行同步。

在DataToServer.py中，使用requests与HB服务进行通信，使用存储的aid和iid获取HB中具体设备的具体属性，使用新的值刷新Devices.json。在第一次同步时将所有设备，包括传感器与配件全部发送给服务器，但在后续的同步时将传感器数据全部发送给服务器而只将同步周期之间更改过属性的配件信息发送给服务器。

待服务器将新的Devices.json表发还给管理设备，本地会将传回的表与本地表进行对比，具体对比字段就是配件的currentvalue字段，使用新的值以达到使用Web端控制本地设备的目的。

### 具体用例说明

1. 报警功能

报警系统的状态转换图在章节3.1.1中进行过说明，在此讲解正常工作和报警时管理设备和服务器的工作。

在设计中，管理设备将报警系统视作传感器进行处理但拥有对其控制和管理的权限。通过对章节6.1中的控制位进行置位达到控制相应传感器和报警功能开关的目的。

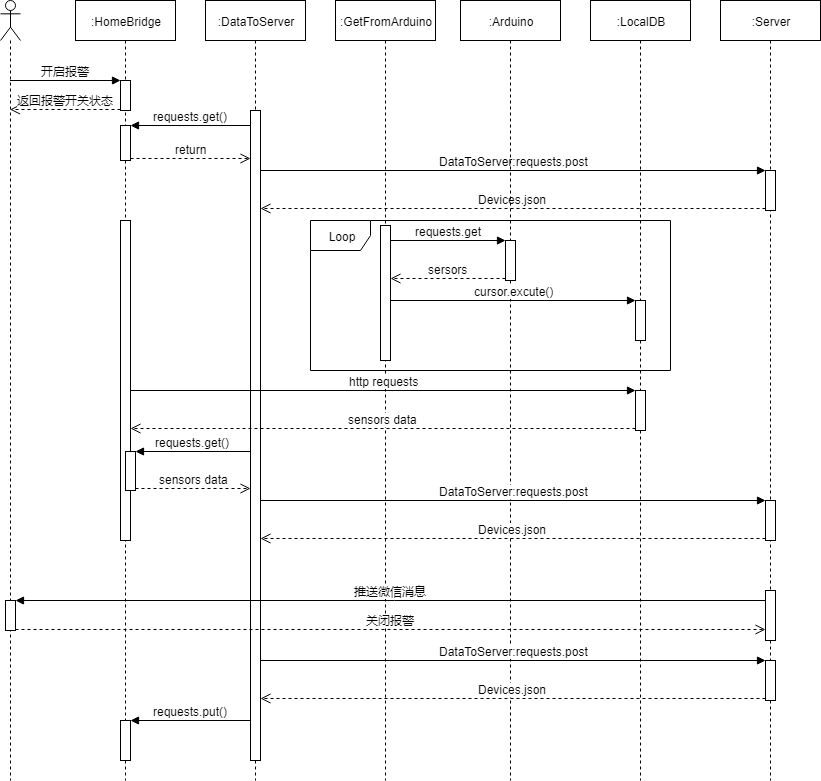


Figure 46 报警顺序图

发生报警时，报警系统将先通知管理设备，再通过管理设备将报警发送给服务器，最终再由服务器将报警信息发送给用户。

用户在Web上关闭报警后在数据同步时将报警关闭状态发还给管理设备，再对HB服务中的报警开关状态进行更改。

1. 场景搭建

场景搭建功能只在服务器Web界面上可用。

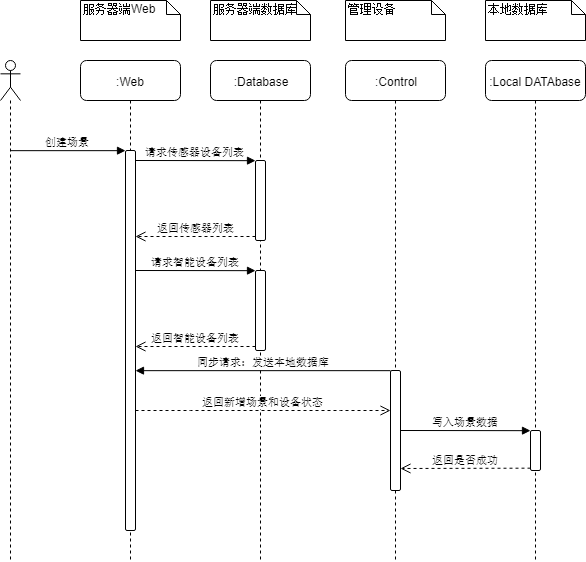


Figure 47 场景搭建

目前场景搭建功能在手机APP上不能实现，用户在服务器Web界面上自定义场景。先选择传感器列表中的传感器，再自定义相应传感器的阈值，定义结束触发器之后，用户将选择相应动作，在智能设备列表中选择想使用的设备，再选择相应智能设备的指令。添加结束后的场景信息会添加到服务器数据库的场景表中，其信息会在由管理设备发起的同步连接中发送给管理设备。

## 服务器模块

功能：此模块的功能是对用户通过web应用管理的树莓派的请求作出响应，更新树莓派的配置，重新渲染web应用。

输入：用户所发布的指令

输出：对树莓派的配置信息和web渲染命令

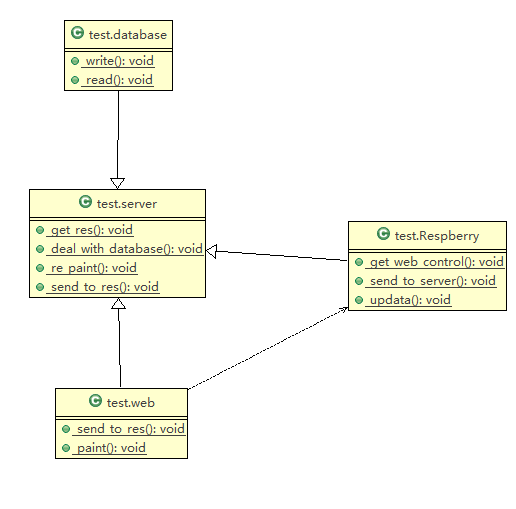


Figure 48 类结构图

该模块的主类是server类，该类可以调用get\_res方法去获取树莓派的请求。树莓派通过get\_web\_control方法获取web应用的控制信号。server通过deal\_with\_database方法与数据库进行交互，通过re\_paint渲染web，同时通过send\_to\_res给树莓派发送更新配置的信息。

# 运行与开发环境

## 运行环境

|  |  |
| --- | --- |
| 名称 | 运行环境 |
| 服务器端 | ubuntu ==18.04(LTS) X64 |
| app客户端 | IOS>=10.0 |
| 报警系统程序 | Arduino开发板 |
| 物联网智能控制程序(homebridge) | RPi开发板 |

Figure 49 运行环境

## 软件环境

|  |  |  |
| --- | --- | --- |
| 类型 | 名称 | 开发环境 |
| 硬件 | 服务器 | Vultr 平台 CPU: 1核 RAM:1GB |
| Arduino单片机 | 型号：UNO R3 |
| 树莓派 | 型号：Raspberry Pi 3b |
| 软件 | Arduino的开发 | ARDUINO IDE （>=1.8.8） |
| 树莓派的开发 | Pycharm （>=2018.3.5） |
| 服务器开发 | **系统：**  ubuntu ==18.04 (LTS)  **部署及web框架：**  docker  nginx >= 1.8.1  uwsgi >= 2.0.15  django >=2.0.13 |

Figure 50 软件环境

# 需求可追踪性说明

|  |  |  |
| --- | --- | --- |
| SRS文档 | SDD文档 | 对应说明 |
| 报警用例 | 章节6.2.4（1） | 达成明确的报警功能。完成用户自定义功能  能够根据报警情况进行报警。 |
| 添加设备 | 章节6.2.4（3） | 能够成功添加设备，并且在手机APP端和服务器Web端显示。  设备添加对应的部分为HomeBridge软件和本地数据库，通过HomeBridge与手机APP的链接和管理设备与服务器的数据同步能够完成上述功能。 |
| 安全性需求 | 章节4 | 加密智能设备与管理设备之间的连接，加密管理设备与手机APP、服务器之间的连接。 |
| 可靠性需求 | 章节5、章节6 | 测试期间长时间（三天）内无系统失效，记录日志文件。 |
| 兼容性需求 | 章节6 | 能够在服务器Web端使用，能够在IOS设备上使用。 |
| 性能需求 | 章节3、章节4、  章节5、章节6 | 报警系统在触发警报的情况下5分钟内通知到用户。  管理设备5秒内响应用户对智能设备的命令。  报警系统与管理设备的数据同步和管理恶化设备与服务器之间的数据同步频率设定能够满足第一条功能。  HomeBridge软件与树莓派的硬件速度能够满足第二条功能。 |
| 易用性需求 | 章节5、章节6 | 服务器Web端的设计达到易用性要求。  Web界面设计与操作逻辑达到上述要求。 |

Figure 51 需求对应