**AAA**

**需求规格说明书**

**SRS-107**

**V0.1**

分工说明

|  |  |  |
| --- | --- | --- |
| 小组名称 | AAA | |
| 学号 | 姓名 | 本文档中主要承担的工作内容 |
| 16061073 | 刘禹廷 | 范围、功能需求的一部分 |
| 16061042 | 杨枫 | 业务需求 |
| 16061079 | 柴林政 | 数据需求、运行与开发环境 |
| 16061049 | 王政 | 功能需求的一部分 |
| 1506 | 赵振宁 | 非功能需求 |

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 审核人 | 版本说明 |
| V0.1 | 2019.3.29 | 刘禹廷 | 杨枫 | 文档初稿 |
| V0.2 | 2019.4.1 | 刘禹廷 | 杨枫 | 更改业务需求和功能需求的内容 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1. 范围 1](#_Toc4762737)

[1.1 项目概述 1](#_Toc4762738)

[1.2 文档概述 1](#_Toc4762739)

[1.3 术语和缩略词 1](#_Toc4762740)

[1.4 引用文档 2](#_Toc4762741)

[2. 业务需求 2](#_Toc4762742)

[2.1 用户画像 2](#_Toc4762743)

[2.1.1 普通用户 2](#_Toc4762744)

[2.1.2 系统管理员 3](#_Toc4762745)

[2.2 用例 3](#_Toc4762746)

[3. 功能需求 4](#_Toc4762747)

[3.1 用例模型： 4](#_Toc4762748)

[3.2 用户详细说明 5](#_Toc4762749)

[3.3 其他用例 6](#_Toc4762750)

[3.3.1 向系统中添加智能设备与房间定制 6](#_Toc4762751)

[3.3.2 宠物监视 6](#_Toc4762752)

[3.3.3 灯光控制 7](#_Toc4762753)

[3.3.4 窗帘控制 7](#_Toc4762754)

[3.3.5 设备故障 7](#_Toc4762755)

[4. 数据需求 7](#_Toc4762756)

[5. 非功能需求 8](#_Toc4762757)

[5.1 安全性需求 8](#_Toc4762758)

[5.2 可靠性需求 8](#_Toc4762759)

[5.3 兼容性需求 9](#_Toc4762760)

[5.4 性能需求 9](#_Toc4762761)

[5.5 易用性需求 9](#_Toc4762762)

[5.6 保密需求 9](#_Toc4762763)

[6. 运行与开发环境 9](#_Toc4762764)

[6.1 运行环境 9](#_Toc4762765)

[6.2 软件环境 10](#_Toc4762766)

[6.3 用户界面需求 11](#_Toc4762767)

# 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，和使用摄像头、动作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、亮度等等信息。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。此外对于家中有宠物的用户来说，可以使用手机端的适配APP获取宠物的状态。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 |
| 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 术语和缩略词

## 引用文档

1. 《AAA项目开发计划(SPD-107)》 v0.2 2019.3.28

# 业务需求

使用者：

• 普通用户

• 系统管理员

## 用户画像

### 普通用户

普通用户是本软件的最终使用者。普通用户没有任何 EE&CS 专业知识背景, 仅仅能够与手机和计算机进行基本的交互 (例如填写并提交表单, 通过 GUI 访问程序页面等)。普通用户主要关注：

1. 系统易于使用，易于配置
2. 系统出错率低（或是系统出错后可以在不被察觉的情况下修复）

普通用户会和系统的终端产生交互。系统的全部在用户眼中只是手机或电脑上的 GUI 程序。由于用户的特点，普通用户仅能够使用模糊的自然语言描述理想中的系统该如何工作。我们需要通过访谈或是问卷的方式，获取普通用户的需求数据，并对这些数据进行整理与建模，然后记录为标准化用例。

### 系统管理员

系统管理员拥有专业知识背景，且具有完全理解系统工作原理的能力。系统管理员：

1. 测试代码是否完善
2. 系统的架构是否合理
3. API 文档是否完善
4. 代码质量

系统管理员的需求将由开发团队本身，在进行项目开发的过程种进行整理与记录。

# 功能需求

## 用例模型：

使用者：房主，系统管理员，传感器

使用功能描述：

1. 用户可通过控制面板安装和解除系统
2. 用户可通过因特网访问系统
3. 用户/传感器可响应报警事件
4. 用户/传感器可关联错误条件
5. 系统管理员重新配置传感器以及相关的系统特征
6. 传感器可以控制相应的设备进行相应的工作
7. 系统需要存储相应的账号密码，以及用户输入的参数到数据库
8. 系统需要把自身内部状态的变化通知给执行者，比如空调是否打开，报警器是否启动，大灯是否打开
9. 执行者需要获取当前所有参数，如果当前没有参数则需要创建一个，每一个用户有他所对应的参数，存储于数据库中。如果想要更新参数，则必须把原来的参数删除。
10. 系统必须知道传感器的参数，在没有人为干预的情况下自动调整
11. 用户可以更新报警音乐，音量大小，
12. 用户可以更新空调温度
13. 用户可以更新大灯开关的时间

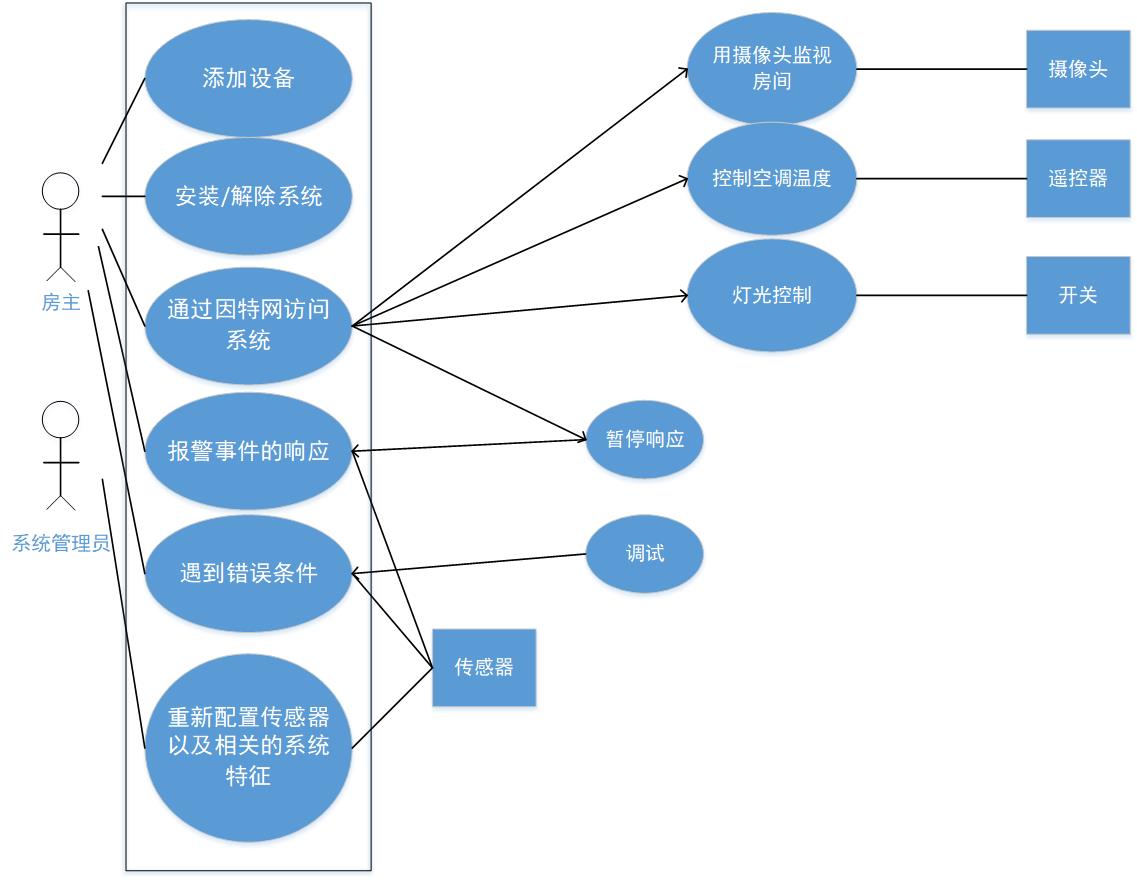


Figure 3 模型用例图

## 用户详细说明

在上面的具体入侵报警场景中，情景如下：

1. 用户自身不在家；
2. 检测到门锁已经上锁或遭到破坏：
   1. 检测到门锁已经被破坏则直接对用户进行报警；
   2. 门锁完整且已经上锁则检测人体；
   3. 检测到人体；
3. 对用户手机发送推送通知并播放声音进行报警。

上面列举了一个入侵检测报警的用例。用户作为本项目智能家居系统的主要服务者，受益于智能家居系统带来的更加舒适与便利的生活。在完成配置整套系统之后，设备会根据用户自定义的场景触发相应的动作。

用户的主要工作包括购买设备、安装设备、添加设备、配置相应场景。

对添加设备，配置场景将在3.3节中详细说明。

## 其他用例详细说明

### 使用摄像头监控房间

主要参与者：普通用户

目标：从任何远程地点通过互联网查看遍布房间的摄像头输出；

前置条件：必须完整配置系，用户具有合法的账号和密码；

启动：房主决定查看房屋内部状况；

场景：

1. 房主登录SmartHome软件；
2. 房主输入账号和密码；
3. 系统显示所有的主要功能按钮；
4. 房主从主要功能按钮中选择“监视”；
5. 房主选择“选取摄像头”；
6. 系统显示房间的平面设计图；
7. 房主从房屋的平面设计图选择某个摄像头的图标；
8. 房主选择“视图”按钮；
9. 系统显示一个由摄像头编号确定的视图窗口；
10. 系统在视图窗口中显示前段时间屋内的转态图片；

异常情况：

账号或者密码不正确——出现点：场景中的2，参看用例“确认账号和密码”；

优先级：中；

何时可用：第三个增量

使用频率：中等

次要参与者：通过无线网络连接的摄像头

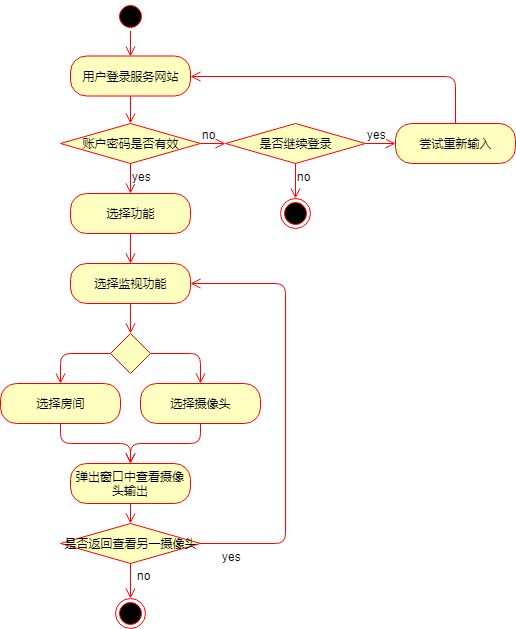


Figure 4 选择摄像头进行查看

### 向系统中添加智能设备与房间定制

在用户购入新的智能家居设备之后，需要将新设备添加到系统中进行管理，首先在系统中进行登录，登录完成后新设备将会出现在系统的设备列表中，并有新设备传感器数据的显示。用户可以根据自定义的房间配置选择新设备所在的房间和位置，对新设备传感器数据设置阈值与相应的自动化动作，可能的场景和动作如后文所述。

场景：

1. 房主打开新设备；
2. 房主使用手机连接到网关进行新设备的搜索；
3. 在搜索到设备之后房主选择设备所在房间；
4. 房主选择创建场景
5. 根据用户设定的触发条件

异常情况：

系统识别不到要添加的设备——出现点：场景中的(2)，参看用例“是否搜索到新设备”；

优先级：中；

何时可用：第四个增量

使用频率：低等

次要参与者：新添智能设备

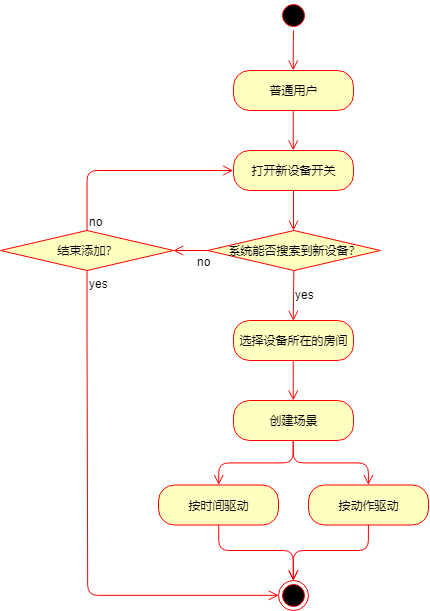


Figure 5 添加新设备

### 灯光控制

灯光控制作为智能家居的主要功能之一，能够显著提升用户家居生活的幸福程度。用户在系统中登录灯具后，可以直接在手机APP上控制灯具的亮度、颜色、色温等属性。在更加智能化的场景当中，设备将于用户手机GPS协同合作，在用户不在家但将要到家的情况下，自动开启家中玄关灯光客厅灯光等。

场景：

1. 用户添加灯光设备
2. 用户选择添加场景
3. 用户设定触发条件
4. 用户选择触发的动作，包括打开灯光，改变色温等。

异常情况：

系统识别不到要添加的设备——出现点：场景中的(1)，参看用例“添加新设备”中搜索不到新设备；

优先级：中；

何时可用：第五个增量

使用频率：高等

次要参与者：智能灯光设备

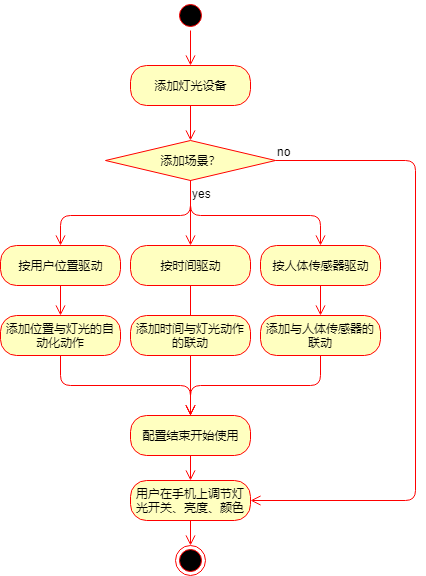


Figure 6 灯光控制

# 数据需求



Figure 7 数据需求ER图

如图4所示，我们将主要建立如下几个实体:

1. 用户：用户用来登录管理系统的实体，包括账号，密码以及一些用户配置信息。
2. 房间：一个用户可以拥有多个房间，房间的一些简单配置。
3. 设备：设备是一个父类，拥有了所有设备都具备的属性，子类样例如图5所示。拥有了一些设备特有的信息。

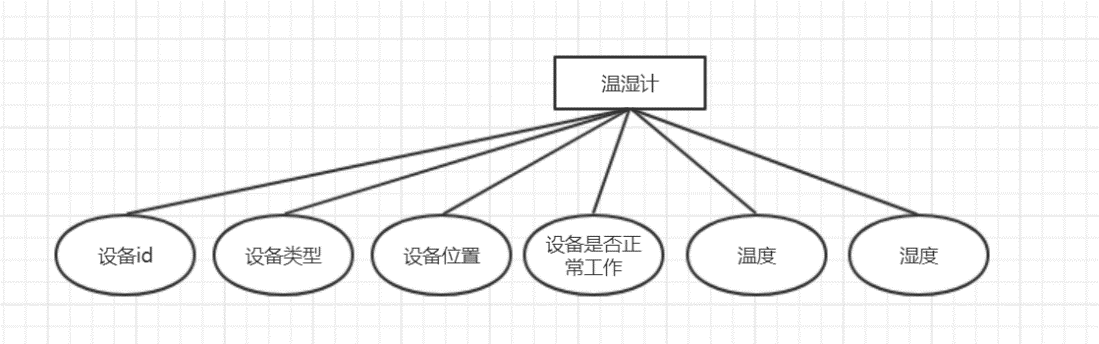


Figure 8 子类样例

1. 场景：一个房间可以配置多个场景， 例如防盗报警场景，防火场景，感应灯场景，启动这些场景，这些场景中的相应设备就会开始工作，并可以通过场景中的配置来统一配置设备的一些阈值和参数。

# 非功能需求

## 安全性需求

对用户进行严格的权限管理，只能访问其权限范围内的数据，只能进行其权限范围内的操作。对用户身份进行身份认证，保护数据不被非法/越权访问和篡改，要确保数据的机密性和完整性。

系统应具备一定的抵抗恶意攻击的能力，如病毒、黑客。

可以提供运行日志管理功能，方便追踪系统使用历史。

## 可靠性需求

系统应具有健壮性，能处理系统运行过程中出现的各种异常情况，如：人为操作错误、输入非法数据、硬件设备失败等，系统应该能正确的处理，恰当的回避。

本系统涉及到家庭安全，应保证超长时间持续运行并且故障率极低。

## 兼容性需求

本系统支持IOS、Android操作系统。

## 性能需求

温度、湿度、亮度数据每隔5s采集一次，温度精确到0.1℃，湿度精确到±2%RH。

能实时响应对照明的控制，实时查看视频监控。

网络通畅、设备正常时，响应时间最多应不超过2s。

## 易用性需求

普通用户可以在1小时内完成初始注册、设备配置、所有基础功能的检验；在一周后能熟练使用系统控制代替手动控制，能够自定义自动化方案。

## 保密需求

网络传递数据应经过加密。需要保证数据在采集、传输和处理过程中不被偷窥、窃取、篡改。

# 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 | Android >= 7.0 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 9 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） |
|  | 安卓app的开发 | Android Studio （>=3.3.2 ） |
|  | 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 10 软件环境

## 用户界面需求



Figure 11 Web端总览

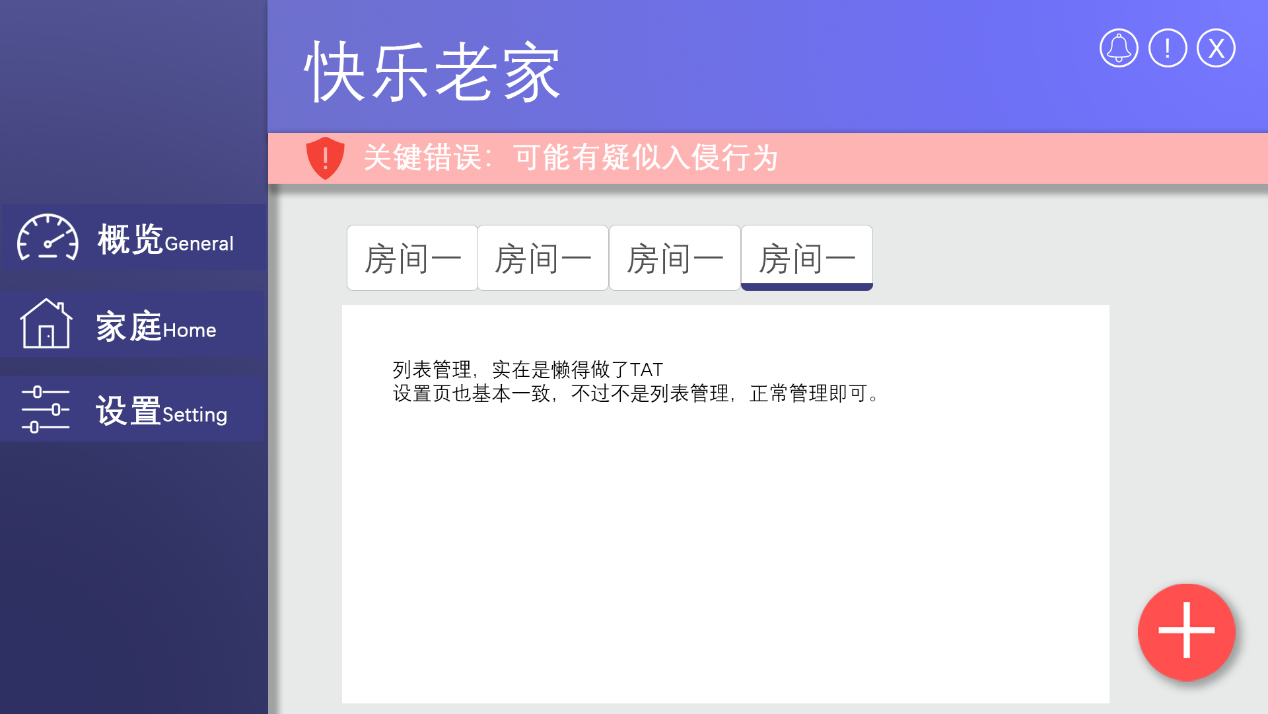


Figure 12 Web房间管理界面