系统设计说明书

**项目名称：**

**“Tooth Fairy”睡眠助手系统**

**开发团队： “Tooth Fairy”小组**

**成　　员： 　 杜圣哲 王必聪 道发发**

**柳博 严旭东 高烁琳 李雪**

**哈尔滨工业大学**

**软件学院**

二零一六年十二月

# 系统概要设计

## 系统的功能设计

### 1.1.1“自然醒”闹钟

* 模拟日出过程的渐变式“日光”效果
* 播放鸟叫、流水声等自然环境声音
* 通过手机端应用设置闹钟

### 1.1.2个性化贴士

* 用户起床后播放音乐
* 用户起床后播报当日天气、新闻、出行交通情况等信息

### 1.1.3睡眠环境监测与调节

* 实时监测室内的温度、湿度、气压等信息数据
* 根据数据与预判条件值，做出相应的调节措施

### 1.1.4安全防护

* 通过监测门窗是否被恶意打开进行防盗
* 通过收集声音和图像/影像信息对防盗判断结果进行辅助判定（待加功能）
* 对厨房附近可燃气体浓度进行监测，预防燃气泄漏带来的危险
* 对常用加热型电器或涉水电器进行温度等监测，防火、防电（待加功能）
* 通过手机交互实现“一键开关”常用电器
* 系统判断遇到危险后进行主动报警并叫醒用户

### 1.1.5 Android App交互

* 支持用户以语音的形式与手机交互，控制系统中某些功能、获取“贴士”信息等

### 1.1.6其他

* 支持RFID标签标识特定用户，实现更换睡眠地点时的快速设置（待加功能）

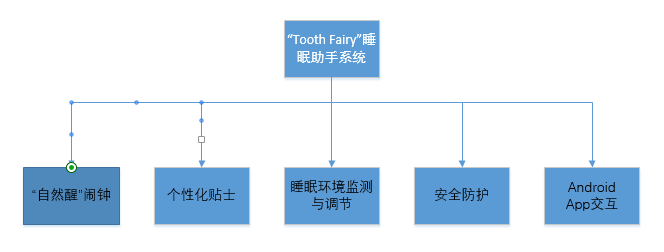


图1-1 功能模块图

## 系统技术架构设计

基于现有的设备和需求分析中的要求，我们整个系统分为感知层，网络层，应用层，应用层为C/S架构。

### 1.2.1感知层设计

本系统的感知层由8种传感器组成，分别为

* 0：温度
* 1：湿度
* 2：气压
* 3：烟雾
* 4：门磁
* 5：红外渐进光
* 6：周围环境光
* 7：继电器
* 8：加速度

其中各个设备的型号分别为

|  |  |
| --- | --- |
| 设备 | 型号 |
| 继电器 | G6K-2F |
| 光传感器 | ISL29028A |
| 加速度传感器 | MMA8452Q |
| 门磁传感器 | A1104 |
| 传感器 | MPL3115A2 |
| 温湿度传感器 | SHT1x\_SHT7x |
| 烟雾传感器 | MQ-2 |

门磁传感器和继电器采用系统主动控制的方式，其他传感器的数据均为五秒采集一次。

### 1.2.2网络层设计

网络层负责接收并转发各个传感器的数据。感知层各传感器与本地网关板的通信协议为ZigeBee，当启动传感器模块的时候，ARM网关会主动连接到传感器并进行通信。ARM网关与上位机的通信协议为UDP协议。协议的详细格式见下文

### 1.2.3应用层设计

应用层主要进行逻辑分析，得出分析结果后进行相应的处理。

* 判断家里的安全状况：燃气是否泄漏、门是否打开等，出现异常及时通知用户；
* 判断晚上睡眠环境是否是最优睡眠环境，是否需要进行调节；
* 早上充当自然闹钟的角色，唤醒用户，并播报当天的天气、交通等相关信息。

#### 1.2.3.1客户端设计

本系统中用户直接操作的客户端为Android App，本应用需要实现接收服务器定时推送、“语音助手”交互功能、设置“日光闹钟”时间、控制外围智能硬件状态等控制功能。

Android端应用架构如下：

* activity包：包含应用各个界面的Activity以及应用的Activity基类
* receiver包：包含应用的各个注册的receiver（第三方API平台用到的receiver和自己构建的receiver）
* util包：包含应用中能进行封装的工具类，如语言API JSON格式文本结果处理、网络访问工具等

Android端用到的第三方SDK有：

* 极光推送平台
* 讯飞语言助手SDK（在线语言合成、在线语音识别等）

#### 1.2.3.2服务端设计

主要实现感知层数据的可视化以及命令的通知，数据包含室内环境的信息，用户设定的闹铃信息以及其他一些个性化设置。在发生火灾，煤气泄露，门窗被恶意打开等等突发情况时进行警报。

服务器端基于Python flask开发，按照如下架构进行开发：

* app为项目主文件夹
* apiv1为Android端提供API。
* utils提供推送、控制、计时器工具类。
* \_\_init\_\_.py初始化app。
* models.py建立了数据库中应有的表。
* scripts为外部脚本。
* test测试外部API。
* config.py提供各个配置对象。
* run.py为主模板。

## 系统的存储设计

本系统使用关系型数据库存储用户数据和传感器数据，共有用户信息表、闹钟时间表、气压传感器表、温湿度传感器表、门磁传感器表、烟雾传感器表、加速度传感器表、环境信息控制表，每个表信息和数据格式如下：

表1-1 用户信息表

|  |  |
| --- | --- |
| UserID | Int(10) |
| UserName | Char(10) |

表1-2 闹钟信息表

|  |  |
| --- | --- |
| AlarmID | Int(10) |
| UserID | Int(10) |
| time | Datetime |
| flag | Int(1) |

表1-3 温湿度传感器表

|  |  |
| --- | --- |
| TempSensorID | Int(10) |
| Time | Datetime |
| Tempreture | Double(6,4) |
| Wet | Double(6,4) |

表1-4 门磁传感器表

|  |  |
| --- | --- |
| HallSensorID | Int(10) |
| Time | Datetime |
| Level | Int |

表1-5 加速度传感器表

|  |  |
| --- | --- |
| UserID | Int(10) |
| Time | Datetime |
| gravity | Double(6,4) |

表1-6 气压传感器表

|  |  |
| --- | --- |
| PreSensorID | Int(10) |
| time | Char(10) |
| Pressure | Double(7,4) |
| Temperature | Double(64,) |
| Altitude | Double(10,4) |

表1-7 烟雾传感器表

|  |  |
| --- | --- |
| UserID | Int(10) |
| Time | Datetime |
| concentration | Int |

# 系统详细设计



## 系统感知层详细设计

设备选型：ARM网关、CC2531开发板、ZIGEBEE通信模块、传感器模块，其中传感器模块有温度、湿度、气压、周围环境光、光照、电磁、继电器、加速度等八个传感器。

### 2.1.1 ARM网关板详细信息



图 2- 网关板外观图

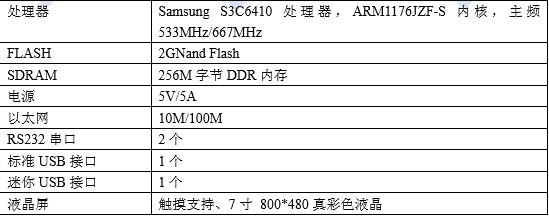




图 2- 网关板详细参数列表

### 2.1.2 ZigBee 传感节点详细信息

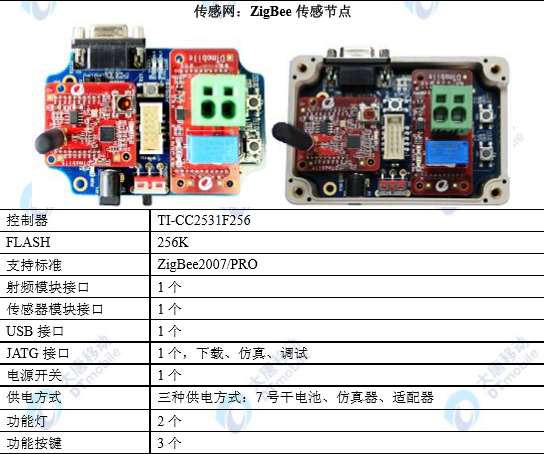


图 2- ZigBee传感节点参数信息

## 系统网络层详细设计

### 2.2.1组网

本系统采用的组网方式为星型网络，传感器与ARM网关之间通过ZigBee协议进行通信，ARM网关与上位机之间通过UDP协议进行通信。以下为传感器与ARM网关的网络拓扑图。

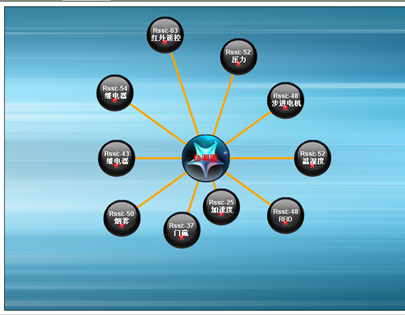


图 2- 传感器与网关板网络拓扑

上位机、ARM网关、传感器节点之间的网络图如下所示。其中ZigBee节点在工作时，与ARM网关相连，由网关上客户端程序通过Socket API将数据转发。同时还有其他传感设备通过ARM网关转发至服务器，服务器对上来的数据进行解析显示。

为了统一消息解析流程及平台的可扩展性，对通信协议进行了约定，各模块按照下面的约定进行消息交互。



图 2- 信息交互示意图

### 2.2.2数据通信协议

**1. 上位机与ARM网关SOCKET通信消息定义：**

表2-1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 消息头 | | | | | | | | 消息体 |
| 消息长度 | 固定  标记位 | 网关  标识 | 设备类型标识 | 保留 | 设备节点地址 | 消息  ID | 流水号 | 数据结构自定义，注意4字节对齐 |
| 2Bytes | 2Bytes  (0x5473) | 2Bytes | 1Byte | 1Bytes  (0xFF) | 2Bytes | 2Bytes | 4Bytes |

说明：

1. 消息长度是指消息头 + 消息体的长度，即消息的总长度。
2. 固定标记位标记实验箱设备，物联网实验箱固定标记位统一为0x5473。
3. 网关标识，区分不同ARM网关。在ARM网管请求登陆服务器时由PC侧服务器动态分配，。
4. 设备类型标识区分接入ARM网关的设备：

ZigBee设备： 0x02；

1. 设备节点地址为每个接入ARM网关设备在网络中的唯一标识；
2. 消息头中的消息ID为PC和ARM网关之间消息交互的标识，和消息内容中串口的消息ID无关。
3. 消息头的消息ID为0x1234，标识ARM网关登录PC服务器请求消息，此时，网关标识、消息内容都可为空；
4. 消息头的消息ID为0x4321，标识ARM网关登录PC服务器响应消息，PC服务器动态分配网关标识，供ARM网关使用；
5. 消息头的消息ID为0x0010，标识ZigBee等设备向PC发送正常业务消息；
6. ARM网关与PC有其他类型的通信消息，再加以补充；
7. 流水号：收到一条请求消息，要以同样的流水号回响应消息。
8. 消息体内容：ZigBee及RFID的消息体定义参考第二部分串口通信数据格式；
9. 消息体定义

串口设备消息定义

表2-2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 标识位  1Byte  (固定0xAB) | 标识位  1Byte  (固定0xCD) | 保留位  2Bytes  （0x0000） | D\_LEN  1Byte | 设备类型  1Byte | 消息ID  2Bytes | 流水号  4bytes | DATA | CHECK  1byte |

说明：

1. 标识位：标识位固定为0xCDAB，长度为2Bytes；
2. 保留位：2Bytes的保留位，默认填写0x0000;
3. D\_LEN：该字段长度为1字节，表示DATA部分的长度。
4. 设备类型：区分ZigBee、RFID等不同设备；

RFID：0x01；ZigBee：0x02；

1. 消息ID，区分不同设备的消息类型，不同设备类型的ID可以重复；
2. **ARM与串口设备之间公共消息定义：**

ARM对串口接入设备，发送设备类型请求消息ID：0x0001，DATA部分不进行填充，同时ARM会对接入的设备定时发送此请求消息，作为心跳消息；

串口设备回应设备类型请求消息ID：0x0010，同时填充自己的设备类型返回消息；

**B、ZigBee消息标识（Cluster\_ID）定义如下:**

0x0030 //温度；

0x0031 //湿度

0x0032 //光照

0x0033 //红外对管

0x0034 //烟雾

0x0036 //加速度

0x0037 //压力

0x0043 //继电器控制

0x0050 //长短地址匹配

0x0030 //温度

0x0031 //湿度

0x0032 //红外渐近光

0x0038 //周围环境光

0x0034 //烟雾

0x0033 //红外对管

0x0036 //加速度

0x0039 //控制继电器开

0x0040 //控制继电器关

0x0037 //压力

0x0011 //接收功率RSSI

0x0051 //PANID

0x0053 //信道号查寻

1. 流水号：收到一条请求消息，要以同样的流水号回响应消息。
2. DATA为数据部分。
3. **ZigBee部分数据定义格式如下表2所示，上下行数据格式相同：**

表2-3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ENDP  1Byte | LO\_ADDR  1Byte | HI\_ADDR  1Byte | EP  1Byte | LO\_ID  1Byte | HI\_ID  1Byte | LEN  1Byte | DAT |

1. ENDP：节点标号，源节点，在星行网络里面，固定为0xCB（协调器）。
2. LO\_ADDR-HI\_ADDR：短地址，EP节点的网络地址
3. EP：终端节点标号
   1. 终端节点：0xD3
   2. 协调器： 0xCB
   3. 路由节点： 0xD2
4. LO\_ID-HI\_ID：Profile ID标号（消息类型，定义与头部zigbee消息ID相同）
5. LEN：数据负荷长度。
6. DAT：数据负荷。
7. CHECK：校验和，该字段长度为1字节，确保数据包的完整性。

校验和计算方法：整条消息逐字节进行异或计算，然后累加，具体算法如下

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

 \* @fn      SPIMgr\_CalcFCS

 \*   Calculate the FCS of a message buffer by XOR'ing each byte.

 \*   Remember to NOT include SOP and FCS fields, so start at the CMD

 \*   field.

 \* @param   byte \*msg\_ptr - message pointer

 \* @param   byte len - length (in bytes) of message

 \* \* @return  result byte

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

byte SPIMgr\_CalcFCS( uint8 \*msg\_ptr, uint8 len )

{

  byte x;

  byte xorResult;

  xorResult = 0;

  for ( x = 0; x < len; x++, msg\_ptr++ )

    xorResult = xorResult ^ \*msg\_ptr;

  return ( xorResult );

}

## 系统应用层详细设计

### 2.3.1 总体逻辑设计

感知层采集到的数据通过网络层上传到服务器中，应用层对各项数据综合进行处理，判断各项指标是否正常。对于烟雾传感器上传的可燃气体浓度，应用层后台逻辑判断其是否达到预设的报警浓度，如果未达到，不进行任何处理，继续检测；若达到预设的报警浓度值则触发报警以及紧急通知用户事件。在晚上睡眠时刻，通过应用层逻辑比较环境指标——温度、湿度和大气压是否在适宜范围内，若在，继续检测；不在则触发相应事件，启动加湿器，或者空调进行调节，使各项指标达到适宜范围。在早上，用户预设的闹钟成为早起事件的触发器，到达该时点后，模拟日光开始工作、音乐响起、播放天气预告以及交通路况等，直到用户起床告知系统已经起床再停止。系统部分功能操作的活动图如下所示。

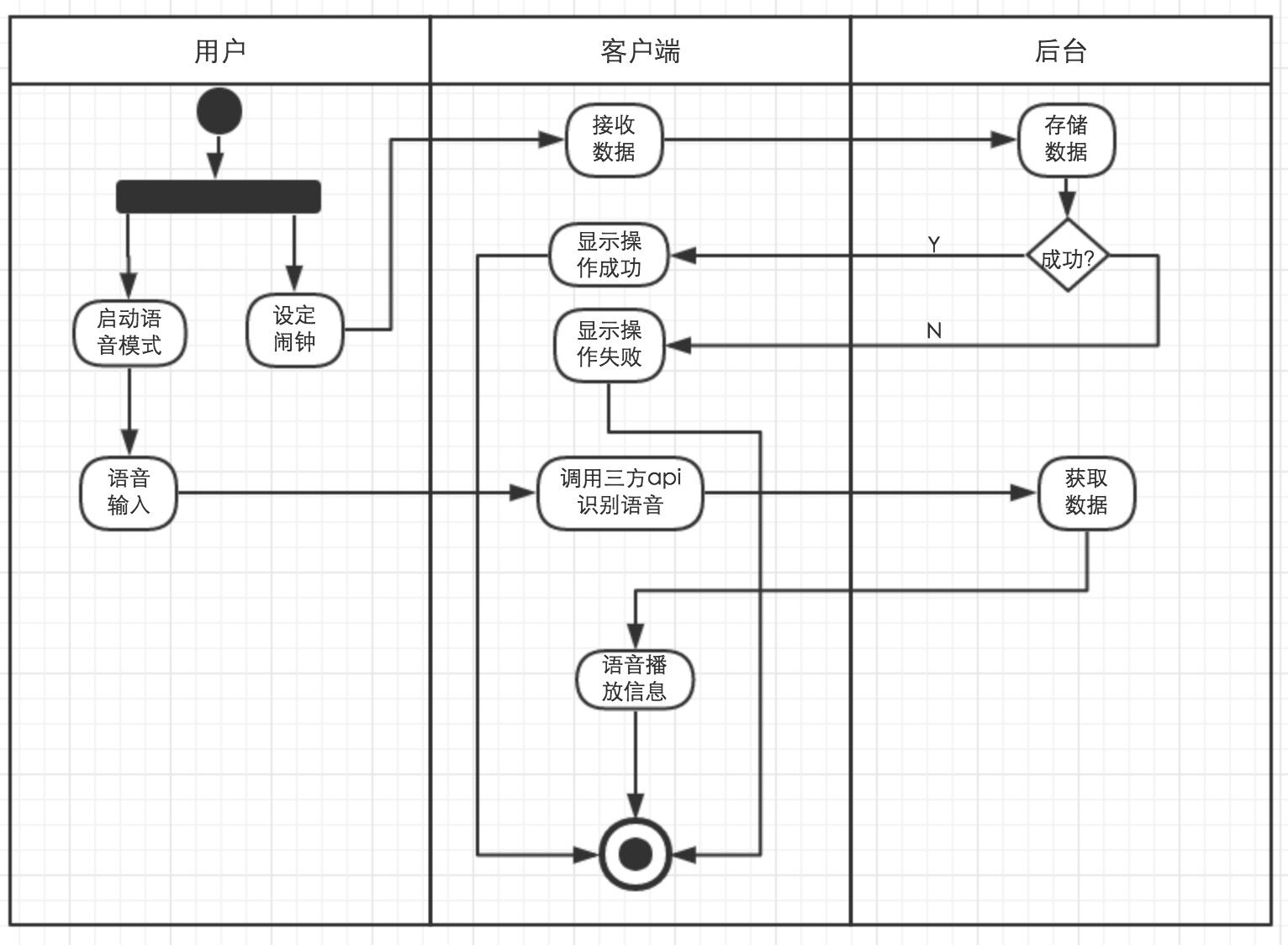


图 2- 用户客户端操作活动图

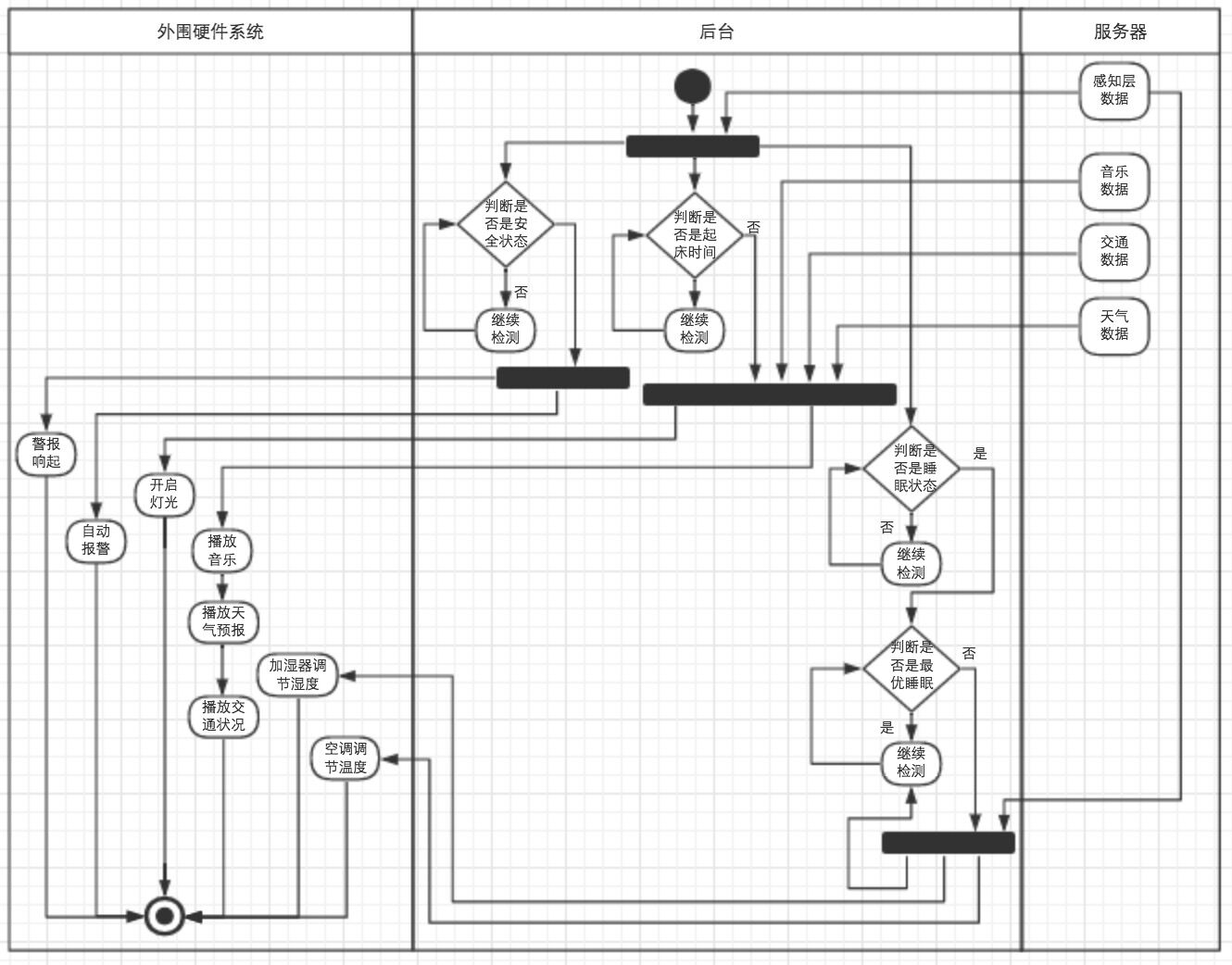


图 2- 硬件系统活动图

### 2.3.2服务器端详细设计

#### 2.3.2.1 MVC模式

本项目的业务逻辑将模型、视图、控制器分离。三部分之间代码耦合度很低。

#### 2.3.2.2 API设计

本项目使用到的外部API有心知天气、知乎日报、图灵机器人、极光推送。

服务器端访问外部API，对返回的Json文件进行预处理，将预处理的结果返回或者主动推送给Android端。

API均按照HTTP Restful接口进行设计，使用Get、Post方法进行数据的获取与存储。

#### 2.3.2.3工具类设计

本项目使用到的工具类有推送、控制与计时器。

计时器在后端程序初始化时开启，当后台获取闹钟信息时，设置定时事件，到时间后出触发特定事件。

推送基于极光推送提供的Restful API，推送模板内将推送功能封装为私有方法，针对知乎、天气等不同API的推送调用主推送方法编写特定接口，实现代码的高复用。

控制工具主要控制Arduino的灯亮、报警功能和音乐播放功能，控制工具类内的方法均在多线程中开启，以免产生阻塞。

#### 2.3.2.4配置的继承

本项目将配置分为多种配置模式，如开发模式和生产模式。

多种配置模式均继承自一个配置基类，共有的配置在配置基类中声明，各自的配置在子类中声明。

### 2.3.3硬件系统设计

硬件部分设计如图5所示。

涉及的设备有：树莓派3代、Arduino UNO、面包板、杜邦线若干、橘色LED若干、红色LED若干、绿色LED若干、电阻若干、无源蜂鸣器、串口转换线等。

其中使用树莓派充当本地服务器，存储感知层上传的数据，并触发相应的事件；用一排黄色LED小灯模拟日光渐亮的过程，实现“日光闹钟”功能，考察其是否是渐变式亮起；用红色的LED小灯以及蜂鸣器标识非安全状态（煤气泄漏、火灾风险、盗窃风险等），并提醒用户；用绿色的LED小灯标识“一键开关电器”等操作的成功执行状态。

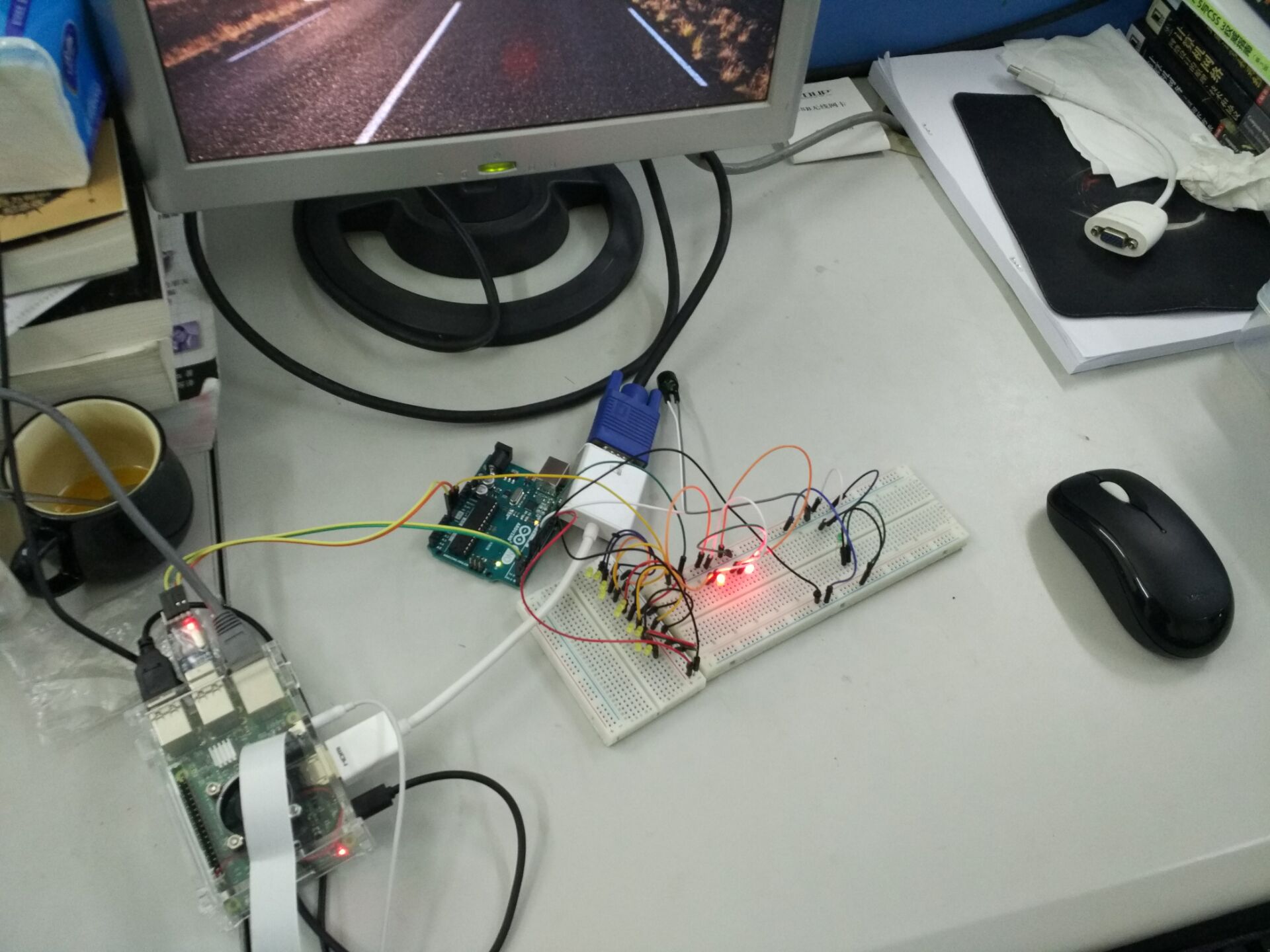


图5

### 2.3.4 UI设计

#### 2.3.4.1 客户端界面设计

首先界面整体设计如图1所示，上方三个图标分别与应用监控家中情况、设置闹钟、语音助手相对应；底部个功能键用于睡眠时间的统计，左部表示入睡，右部表示起床。系统分别记录2个时间节点即可计算出睡眠时间。



图2-8 客户端主界面

设置闹钟，选择闹钟的界面如图所示。

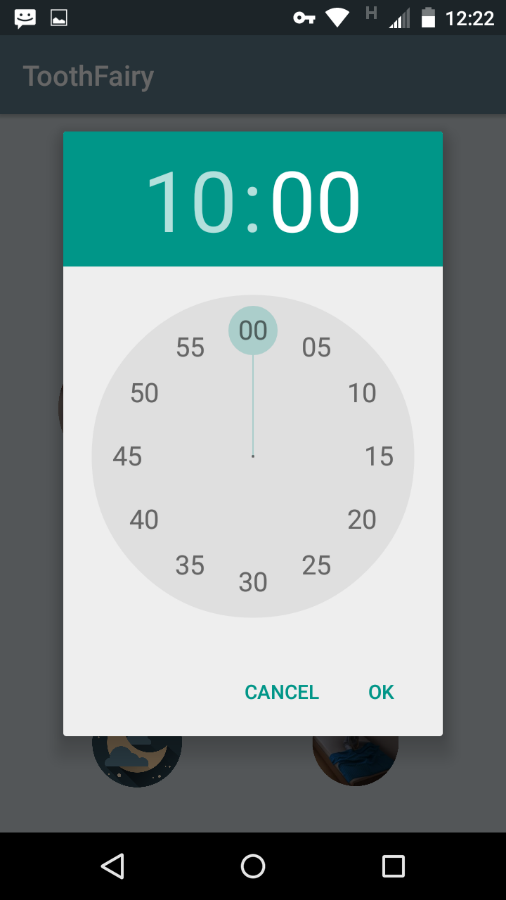


图2-9 添加闹钟界面

添加闹钟成功后应给用户进行反馈，如图3所示。

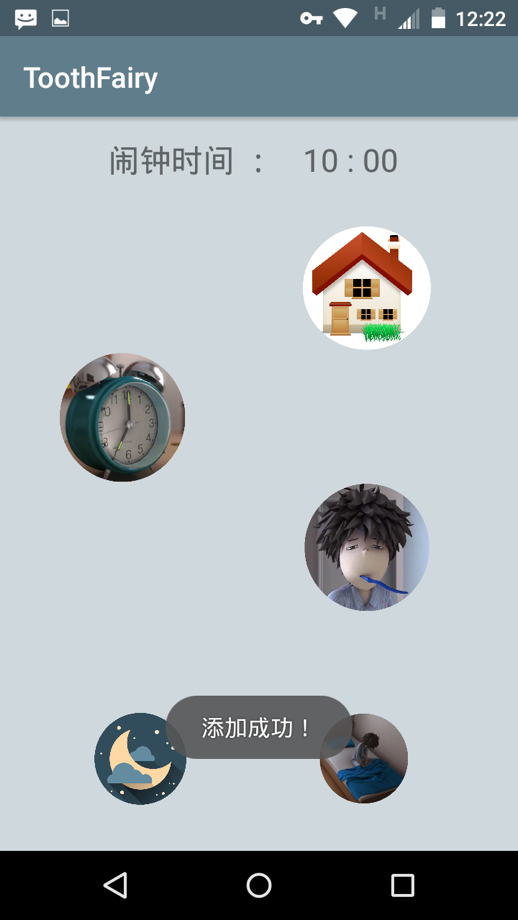


图 2-10 添加成功界面

“语音助手”的用户交互界面如图所示。

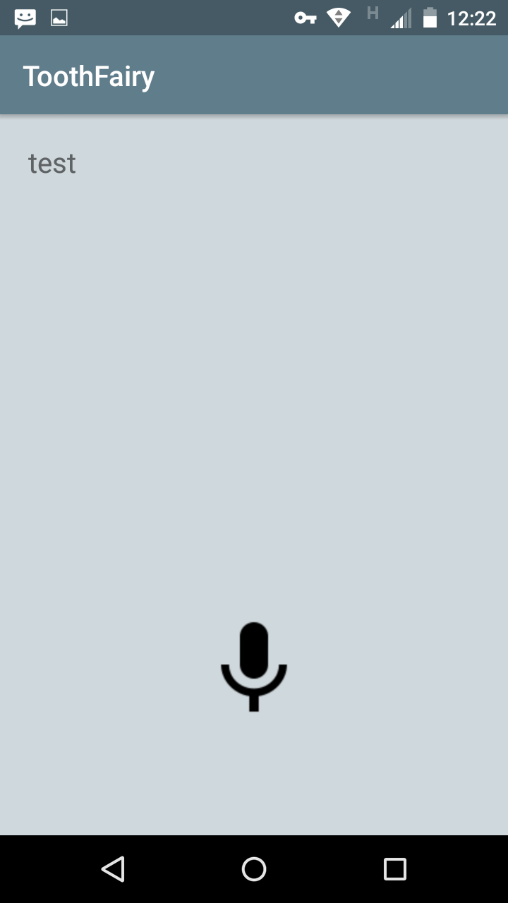


图2-11 “语音助手”界面