## 系统总体结构

App ----- **web-server** ------- **hardware-server** ---------- devices

|\_\_\_\_\_\_\_ **database** \_\_\_\_\_\_\_|

|

**pay\_program**

其中，颜色标识出的部分，是需要此项目需要开发的部分。特别的：

web-server：为修改（前任开发者提供的代码基础上）；

hardware-server：重新开发

pay\_program： 重新开发

## 工作流程说明

web-server与hardware-server联合工作；

pay\_program独立工作**（另附文档说明）**。

web-server处理所有前端请求，所有设备控制信息则与hardware-server联合处理。

本节仅对设备控制信息流程进行说明，其余工作已前任开发者已经完成。

**web-server功能说明**

当app端的设备控制指令调用web-server后，web-server进行以下处理：

1. 对设备控制指令及控制人进行验证；通过验证后进入步骤2，否则退出；（已完成）
2. 读取devices 数据表，获取设备当前状态；
3. 根据设备状态及指令，进行指令处理。即如果设备已被其他人占用，直接回复拒绝操作，结束操作；如果设备空闲，则设置devices相关字段，占用设备，并且生成硬件控制指令，发往hardware-server；**（后文会有详细说明）**
4. hardware-server收到指令，立即回复指令收到，web-server结束工作退出；
5. 否则web-server进行三次重发，如果仍未收到回复，恢复devices空闲状态，然后结束退出；**（后文会有详细说明）**

**hardware-server功能说明**

服务器传至hardware-server的所有控制指令，默认已经经过合法性验证，hardware-server不再进行任何控制指令合法性验证工作。其主要工作为：**接收来自服务器端的指令、设备连接、设备心跳；定时查询devices数据表，进行超时重发、设备状态更改工作**。具体工作流程为：

**服务器端控制指令处理**

1. 如果收到服务器端控制指令，则转发至硬件设备；
2. 如果硬件设备回复操作成功，或收到心跳包，表明操作成功，则设置devices表中设备开启时间字段，完成所有操作；

如果设备没有连接，或返回操作失败，则恢复devices表至设备未占用状态；

如果收到设备心跳包，并表明设备没有开启成功，则读取devices表中对应信息，根据指令接收时间，决定重发指令，或回复设备至未占用状态；

**硬件心跳包处理**

1. 对收到的心跳包进行处理，获取硬件当前状态，根据devices表中的设备当前状态，更新设备记录，并进行计费等工作；**（后文有详细说明）**

**硬件连接处理**

1. 程序中记录硬件连接，并根据心跳判断硬件是否断线；

**devices表定时轮询**

1. 根据设备指令、指令接收时间、当前时间、设备当前状态，进行指令重发、恢复设备至未占用状态；
2. 如设备开启时间超出规定时间，则关闭设备，并更新设备至未占用状态；**（后文有详细说明）**

综上所述，web-server和 hardware-server主要通过devices表进行硬件控制事务的处理；web-server主要负责提出设备控制需求，而hardware-server负责执行及结果反馈。

## web-server数据表操作逻辑

web-server仅根据设备控制指令，对devices进行操作。下文中student\_no、order等所有devices中的字段，均表示当前记录数值；发送指令的学生学号为csid。

**（注：请严格按照指定的action顺序执行）**

**指令(csid发送) student\_no action**

OPEN -1（未占用） set student\_no=csid；

order=OPEN；

order\_recv\_t；

open\_t；

clear close\_t；

返回控制成功（往APP）；

student\_no== csid 返回操作无效（往APP）；

order==OPEN

(同一个操作者连续开设备)

student\_ no == csid 返回操作无效（往APP）；

order==CLOSE

（此情况仅理论上可能出现）

student\_no！=csid 返回操作失败（往APP）；

CLOSE -1（未占用） 返回操作无效（往APP）；

student\_no== csid set order=CLOSE；

order==OPEN order\_recv\_t；

返回接收成功（往APP）；

发送设备关闭（往hareware-server）

student\_no== csid 返回接收成功（往APP）；

order==CLOSE

(同一个操作者连续关设备)

student\_no！=csid 返回操作失败（往APP）；

## 五、hardware-server数据表操作逻辑

此程序循环运行步骤为：

While 1 {

select（3秒）

处理接收到的指令

每n秒，轮询数据库一次

硬件心跳超时轮询处理

}

以下是hardware-server对硬件反馈、心跳包中包含的硬件真实状态进行处理。

**设备默认状态为CLOSE**。同时在sokect管理链表内，**设置此socket最后活动时间**。

**设备真实状态 student\_no order dev\_state action**

0(关闭) -1 0 无动作；

-1 1 set dev\_state=0；

!=-1 OPEN 0 根据order\_recv，决定是否重发；

超时，恢复设备至未占用状态；

!=-1 OPEN 1 生成计费信息；

set dev\_state=0；

open\_t=0；

close\_t=0；

重发open控制指令；

!=-1 CLOSE 0 恢复设备至未占用模式；

（此状态仅理论存在）

!=-1 CLOSE 1 set dev\_state=0；

生成计费信息；

恢复设备未占用状态；

1(开启) -1 0 set dev\_state=1；

**设备异常报警；（？？？）**

-1 1 **设备异常报警；（？？？）**

!=-1 OPEN 0 set dev\_state=1；

open\_t；

!=-1 OPEN 1 无动作；

!=-1 CLOSE 0 set dev\_state=1；

恢复设备至未占用模式；

**设备异常报警；（？？？）**

（此状态仅理论存在）

!=-1 CLOSE 1 根据order\_recv，决定是否重发；

超时，生成计费信息；

恢复设备至未占用模式；

**注： 恢复设备至未占用模式，不涉及dev\_state操作。具体为：**

**set student\_no = -1**

**order = NONE**

**order\_recv\_t = 0**

**open\_t = 0**

**close\_t = 0**

**下同。**

以下是硬件设备心跳包发送超时需要进行的处理。等效于设备状态为0。发生时间写入log文件。同时令此硬件sokcet=-1。

**student\_no order dev\_state action**

-1 0 无动作；

-1 1 无动作；

！=-1 OPEN 0 恢复至设备未占用状态；

！=-1 OPEN 1 set dev\_state=0；

生成计费记录，需要减去超时时间；

恢复至设备未占用状态；

！=-1 CLOSE 0 恢复至设备未占用状态；

！=-1 CLOSE 1 set dev\_state=0；

生成计费记录，需要减去超时时间；

恢复至设备未占用状态；

以下是对数据库轮询处理逻辑

**student\_no order dev\_state action**

-1 0 无动作；

-1 1 set student\_no=0；

order=CLOSE；

order\_recv\_t；

发送关闭指令；

！=-1 OPEN 0 根据order\_recv\_t，发送OPEN指令；

如超时，恢复设备至未占用模式；

！=-1 OPEN 1 无动作；

！=-1 CLOSE 0 无动作；

！=-1 CLOSE 1 根据order\_recv\_t，发送CLOSE指令；

如超时，产生计费（补偿超时）恢复设备至未占用模式；

**注：本文中，不对student\_no==0的用户计费；**