# 《软件工程》课程设计总结报告

## 项目名称

课程设计的项目名称是 Shattuckite 家庭报警装置

## 项目概述

### 项目目标

shattuckite 项目首先作为2019年《软件工程》课程的课程设计, 用于帮助开发团队获取这门课程的学分。

本项目旨在面向家庭及小型民用建筑物, 提供一个可以通过手机及计算机远程访问环境数据及控制机电 设备的计算机系统.

本系统将允许用户将若干设备连接至一个嵌入式终端，并通过手机或计算机获取信息或控制设备行为。

嵌入式终端是一个拥有嵌入式CPU的硬件实体。它拥有一定的计算能力，并可以通过以太网和云端服务器进行数据交互。

典型的设备包括物理量传感器和执行器。物理量传感器包括温度/湿度/空气质量等能够产生离散时间信号的设备，执行器包括继电器/线性导轨等能改变物理世界状态的设备。

设备到嵌入式终端的连接指的是设备通过有线或无线信道,通过一定的中继装置,建立与嵌入式终端的数据交互通路。有线信道包括运行USART协议的RS232总线，无线信道包括 LoRa , Wifi。中继装置包括 Lora网关或Wifi网卡。

信息至少包括

1. 传感器产生的数据
2. 执行器的状态
3. 当前系统事件

控制设备指改变配置或改变状态。改变配置指改变传感器或执行器的行为，例如数据持久化的策略/数据更新的速率/事件产生的条件等。改变状态特指改变执行器的状态。

### 功能和非功能需求

请参见shattuckite 《需求文档》

（在线访问地址<http://buaase.plan.cnworkshop.xyz:2333/SRS-html/>）

### 最终完成的功能

本项目最终实现以下功能

1. 安装温度传感器/压力传感器/报警器/继电器/报警触发装置
2. 实时监控温度传感器和压力传感器的数据
3. 查看传感器历史数据
4. 查看历史事件
5. 处理历史事件
6. 浏览系统中的传感器
7. 配置传感器报警逻辑
8. 实时推送报警事件
9. 浏览系统中的执行器
10. 操作系统中的执行器

### 达到的效果

本项目最终达到的效果是，可以让用户

1. 安装温度传感器/压力传感器/报警器/继电器/报警触发装置
2. 实时监控温度传感器和压力传感器的数据
3. 查看传感器历史数据
4. 查看历史事件
5. 处理历史事件
6. 浏览系统中的传感器
7. 配置传感器报警逻辑
8. 实时推送报警事件
9. 浏览系统中的执行器
10. 操作系统中的执行器

## 需求分析阶段总结

我们在需求分析阶段面临的主要困难在于以下几点

### 问题描述

#### 需求不明确

团队在需求建模过程中面临的最严重的问题，是需求不明确的问题。此处的“需求不明确”不仅仅指无法很好的确认项目的需求本身，同时也包括了“无法明确如何明确需求”这一更高阶的问题。

问题在第一次需求文档评审时暴露。具体表现为开发者理解的项目和老师要求的项目之间存在较大的偏差。

#### 需求文档难以修改

很难确保需求文档在迭代的过程中，不会对其他引用了需求文档内容的文档产生影响。事实上，后续其他开发文档的编写均严重依赖需求文档的内容。难以追踪需求变化让我们在第一个需求文档的版本发布后，修改需求变的寸步难行。

#### UML建模不规范

文档的UML图在规范性上有所欠缺。在面向用户的需求建模一节中使用的用例图并没有很好的表达出我们的设计思想。而在阐述具体用例的时候，活动图的绘制与UML规范不符。

### 原因分析

#### 需求不明确原因分析

经过我们的分析，需求不明确的原因可能如下。

1. 需求提出者和开发者间缺少充分的交流。

由于本项目特殊的性质（课程设计），需求提出者（老师）和开发者（学生）之间的交流并不充分。在项目进行需求分析的过程中，并没有（也不可能）像真实世界中，要约方和被要约方频繁的讨论需求以期达成共识。本次项目的开发过程中，老师首先提出需求，然后完全由学生自己对于PPT上给出的碎片化的词语与模糊的信息（预期目标模糊+能够提供的硬件资源模糊）进行解读，撰写需求文档，之后仅在评审时进行一次正式的反馈。这种低密度的交流不利于明确需求

1. 用于需求建模的时间较为紧张

开发者并非全职进行这个项目，而需求提出者对文档规范性，例如文档的格式和UML图表是否100%符合Object Management Group 提出的标准这一类问题，要求非常严格，所以开发者在花费大量的精力提高文档的规范性后，留给思考如何进行需求建模以及实践需求建模的时间并不多。

#### 文档难以修改原因分析

1. 需求文档高扇入低扇出的特性

需求文档是一个项目的灵魂。它不会引用太多其他文档（低扇出），但是会被很多其他文档引用（高扇入）。

1. 难以追踪引用

我们在撰写需求文档的过程中，发现很难追踪对于需求文档的引用。即无法得知在什么地方引用了需求文档中的哪些内容。因此当对修改需求文档中的内容时，极有可能在其他引用了此处内容的文档处引入不可预期的错误。

### 应对方法

#### 开发文档撰写工具

针对上述问题，我们尝试使用“唯一数据源”的方式。即将需求文档的信息抽象后集中存储在某一区域，然后使用工具程序生成需求文档；在别的文档需要引用需求文档时，从唯一数据源中进行引用；需要修改需求文档的内容时，在唯一数据源中进行修改。如此保证了引用需求文档的文档能够跟踪需求文档的修改。一定程度上解决了这个问题。

#### 重新修改部分需求定义

针对开发者和需求提出者之间未能在项目需求上达成共识这一问题，我们再需求文档评审结束后，又根据反馈信息对需求文档进行了修改，泛化了若干过于抽象的用例，并重新绘制了不符合规范的用例图。

#### 重新绘制了UML图

在需求文档评审结束后，我们根据老师提出的建议，对UML进行了重绘。我们对UML官方规范进行了研究，重绘了系统用例图和所有的活动图。但是由于UML规范中并未显示声明ER图规范，因此我们保留了原有的ER图。

## 设计阶段总结

### 系统架构概述

shattuckite 软件分为三个部分:

1. 嵌入式终端软件
2. 服务端软件
3. 用户端软件

此处是根据运行时环境对软件进行分类的。 嵌入式终端软件运行于嵌入式处理器上; 服务端软件运行于服务器上; 用户终端软件运行于 个人计算设备上(例如智能手机和个人电脑)。

嵌入式终端软件负责驱动边缘网络硬件，从传感器中收集数据，并作为一个发布者, 将收集到的数据发布到服务器broker; 同时作为一个订阅者向broker订阅控制信息,实现用户远程控制设备.

服务端软件提供

1. Broker Pattern 中的 Broker 组件, 用于接收其他发布者的数据并向订阅者推送;
2. 基于 Pipe-filter Pattern 设计的构件,用于订阅 Broker 中的特定 Topic 作为 Source , 数据通过一个用于处理数据持久化逻辑的 Flow, 最终将数据推至由关系型数据库构成的``Sink``。
3. Client-Server 模型中的 Server , 并通过``RESTful `` 接口, 将关系型数据库中的数据暴露给用户端软件, 同时处理用户与嵌入式终端的交互。

用户端软件提供用户与系统交互的人机接口.上述讨论的三个部分，每一部分都由若干更小的构件组成。系统整体的构件图如下图所示。

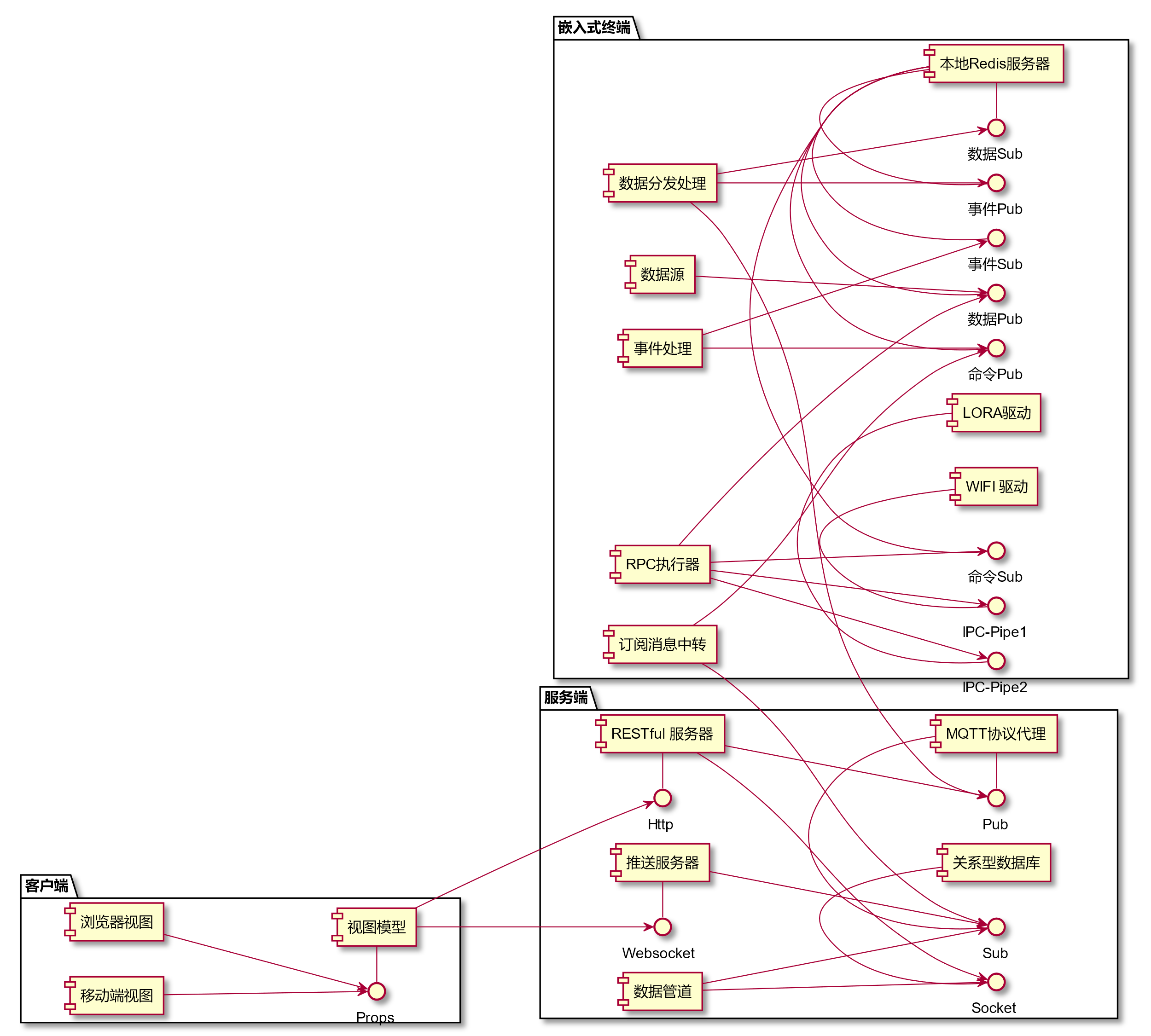


图 1 系统整体构件图

### 设计阶段遇到的问题

#### 问题描述

##### 确定设计文档的边界

在我们撰写设计文档时，遇到的难题之一在于如何确定设计文档的边界。即对于设计文档中的“详细设计“一节，究竟该写到多详细？

##### 保持文档的一致性

如何确保文档中的新定义的概念/术语在全文中的含义都是一致的。

##### 追踪项目需求

撰写设计文档的目的在于实现需求文档中定义的需求。但是我们在完成需求可追踪性工作上遇到了一些困难。

##### 追踪具体实现代码

如何确定项目代码和设计文档中的内容保持一致，以保证设计文档是有意义的，而且不会对阅读文档的开发者产生误导。

#### 问题原因分析

##### 边界难以界定的原因

在撰写组件详细设计的过程中，我们可以选择精确记录一个组件中的所有类，每一个类的方法，每一个方法的输入，输出及异常处理方式；也可以选择仅仅简要描述组件的作用以及和其他组件的关系。记录的越详细，文档提供的信息量越大，但是相应的占用开发者的时间就会越多；反之，简略的描述提供的信息量小，但是工作量小，可能性价比会更高。对于一个真实项目，“人时“是一定的，在某项工作上花费更多时间必然意味着在其他工作上花费更少的时间。我们需要一个系统方法论来帮助我们决定到底该如何撰写文档。

##### 保持文档一致性

设计文档是本次项目撰写的若干文档中，和具体技术联系最紧密的文档。为了简化表达以及精确的表达软件的设计思想，我们在文档中引用了许多术语，并对现有的术语/概念进行组合，定义出新的术语。随着文档长度的增加，保持文档内部的一致性难度也在逐渐增加。如何确保一个名词在不同的上下文中都能无歧义的精确表达一个概念是一个需要解决的问题。

##### 追踪项目需求

设计文档的目的之一是为如何实现需求文档中的需求提供指导性的信息，因此和需求文档时刻保持一致非常重要。

尽管设计文档中有“需求概述“一节，而且我们通过需求分析阶段总结一节中提到的应对方法解决了一部分设计文档和需求文档同步的问题。但这还远远不够，我们只是解决了表面上直接通过文字引用的内容的一致性，但并没有真正实现追踪项目需求。

可以模拟一个非常简单的情景，假如我们在需求文档中删去用户操作执行器的用例，理想情况下，设计文档中有关RPC（远程过程调用）组件的设计需要做相应的删改。但事实上除了“需求概述”一节的需求列表会在文档下次Building时追踪到需求文档的变化，设计文档的其他部分并不会发生任何变化。

问题的根本原因在于缺乏一种由系统组件向需求用例的映射方案。系统组件是面向程序开发者进行设计的，需求用例是面向软件使用者设计的。二者之间不存在线性的对应关系，导致设计文档追踪项目需求变的异常困难。

总而言之，设计文档追踪项目需求是一个非常困难且非常有意义的问题。

##### 追踪具体实现代码

设计文档中的内容最终都要体现在具体的代码上。能否有效的追踪具体实现代码决定了设计文档中的内容是否有意义。毕竟，如果设计文档中描述的信息和实际代码不一致的话，设计文档也没有意义了。

相比追踪需求，让设计文档追踪具体实现代码要简单许多。因为设计文档直接为具体实现提供指导，因此我们可以建立起设计文档的不同章节和不同的代码之间的对应关系。

本次项目中遇到的问题是缺乏称手的工具来管理与维护文档与代码间的关系。

#### 问题解决方案

##### 确定边界

撰写需求文档时最终决定简化文档的撰写，重点在描述组件对外的行为以及组件间的关系上。

##### 保持文档一致性

开发了Grossular工具帮助自动完成子系统组建图绘制，接口枚举及术语引用等工作。一定程度上保持了文档中术语的一致性。

##### 追踪项目需求

未解决

#### 追踪具体代码

根据组件建立不同的Git仓库，手动维护代码和设计文档的一致性。

## 迭代一总结

### 任务分配

第一次迭代在看板上分配了8项任务，概要信息如下表所示。

|  |  |  |
| --- | --- | --- |
| 任务名称 | 任务类别 | 负责人 |
| 嵌入式端测试-模拟数据源编写 | 测试,测试工具开发,第一次迭代 | CNLHC,baixusata |
| 嵌入式端测试-qemu测试环境搭建 | 测试,测试工具开发,第一次迭代 | CNLHC,Dicky35 |
| 前端开发-移动端UI设计-第二期 | 开发,核心功能,第一次迭代 | mqlKKK |
| 前端开发-移动端UI原型设计 | 开发,核心功能,第一次迭代 | mqlKKK |
| 非x86 平台测试环境搭建 | 测试,测试工具开发,第一次迭代 | Dicky35 |
| arm交叉编译环境搭建 | 核心功能,测试,第一次迭代 | Dicky35 |
| 服务端测试-数据库Setup工具的编写 | 测试,测试工具开发,第一次迭代 | CNLHC,Dicky35,baixusata |
| 服务端开发-提供若干Restful接口 | 开发,核心功能,第一次迭代 | CNLHC,mqlKKK |

表 1 第一次迭代任务分配

其中有3项开发任务及5项测试任务。

### 代码提交

在第一次迭代中，我们为3项开发任务及1项测试工具开发任务提交了代码。具体提交情况请前往看板查看。

### 测试

第一次迭代主要在编写测试工具。

### 问题

第一次迭代评审时，暴露的主要问题有

1. 开发进度较慢，应加快速度，尽快实现核心功能。
2. 管理流程不够完善

针对上述问题，我们在之后的一周

1. 加快了开发速度，在第二次迭代期间安排了嵌入式端和服务端大部分核心功能的开发任务。
2. 在第二次迭代期间，引入Github的Issue进行任务分配追踪。

## 迭代二总结

### 任务分配

第二次迭代在看板上分配了8项任务，概要信息如下表所示。

|  |  |  |
| --- | --- | --- |
| 任务名称 | 任务类别 | 负责人 |
| 嵌入式端驱动开发-完成Lora驱动并对外提供接口 | 开发,核心功能,第二次迭代 | CNLHC |
| 嵌入式端开发-具体RPC实例开发 | 开发,核心功能,第二次迭代 | CNLHC |
| 嵌入式端测试-传感器数据上行测试 | 测试,测试用例,第二次迭代 | CNLHC,Dicky35,baixusata |
| 嵌入式端开发-完成基于消息队列的RPC | 开发,核心功能,第二次迭代 | CNLHC,Dicky35 |
| 事件接口单元测试 | 测试,测试用例,第二次迭代 | CNLHC,Dicky35 |
| 传感器数据接口单元测试 | 测试,测试用例,第二次迭代 | CNLHC,baixusata |
| 传感器接口单元测试 | 测试,测试用例,第二次迭代 | CNLHC,Dicky35 |
| 服务端测试-Restful接口单元测试 | 测试,测试用例,第二次迭代 | CNLHC,Dicky35,baixusata |

表 2 第二次迭代任务分配

### 代码提交

在第二次迭代中，我们为3项开发任务及5项测试任务提交了代码。具体提交情况请前往看板看。

### 测试

本次迭代的测试主要围绕着第一次迭代完成的RESTful接口进行。分别对事件接口，传感器数据接口和传感器接口编写了单元测试脚本。

此外第三次迭代完成了传感器节点向服务器上传数据的测试。由于测试流程较为复杂，所以没有编写自动测试脚本。

### 问题

第二次迭代评审时，暴露的主要问题有

1. 测试用例过少。
2. 管理流程仍然不够完善。

针对上述问题，我们在之后的一周

1. 进一步针对已有代码进行单元测试
2. 在第二次迭代期间，引入Github的Project 看板功能进行任务分配与追踪。

## 迭代三总结

### 任务分配

第三次迭代在看板上分配了24项任务，概要信息如下表所示。

|  |  |  |
| --- | --- | --- |
| 任务名称 | 任务类别 | 负责人 |
| 提高设备的响应速度 | 开发,第三次迭代,附加功能开发 | CNLHC |
| 实现线程安全的单点发送 | 开发,第三次迭代,附加功能开发 | CNLHC |
| 嵌入式端驱动测试-Lora模块通信协议测试 | 测试,测试用例,第三次迭代 | CNLHC,Dicky35,baixusata |
| RPC执行测试 | 测试,测试用例,第三次迭代 | CNLHC,Dicky35,baixusata |
| RPC实例测试 | 测试,测试用例,第三次迭代 | CNLHC,Dicky35 |
| 嵌入式端开发-校验RPC请求格式 | 开发,第三次迭代,附加功能开发 | CNLHC,baixusata |
| 嵌入式端开发-RPC协议超时扩展 | 开发,第三次迭代,附加功能开发 | CNLHC,Dicky35 |
| 嵌入式端测试-模拟RPC调用者编写 | 测试,测试工具开发,第三次迭代 | CNLHC,Dicky35,baixusata |
| 嵌入式端测试-模拟执行器编写 | 测试,测试工具开发,第三次迭代 | CNLHC,baixusata |
| 完成基本的lora-c8t6节点 | 开发,核心功能,第三次迭代 | CNLHC |
| 前端开发-完成当前报警的接口调用 | 开发,核心功能,第三次迭代 | mqlKKK |
| 前端开发-完成历史事件接口的调用 | 开发,核心功能,第三次迭代 | mqlKKK |
| 前端开发-完成获取实时数据的接口的调用 | 开发,核心功能,第三次迭代 | mqlKKK |
| 关于系统前后端交互的讨论 | 开发,提议,第三次迭代 | CNLHC,mqlKKK |
| 前端开发-完成设备信息接口的调用 | 开发,核心功能,第三次迭代 | mqlKKK |
| 前端开发-完成历史数据接口的调用 | 开发,核心功能,第三次迭代 | mqlKKK |
| 修改项目Readme | 提议,第三次迭代 | mqlKKK |
| GUI开发-为视图模型添加Typescript类型支持 | 开发,第三次迭代,附加功能开发 | CNLHC,mqlKKK |
| 测试服务器全局Mqtt对象可用性 | 测试,测试用例,第三次迭代 | CNLHC,baixusata |
| 为数据管道添加单元测试 | 测试,测试用例,第三次迭代 | CNLHC,Dicky35 |
| 添加传感器实时数据接口获取的单元测试 | 测试,测试用例,第三次迭代 | CNLHC,baixusata |
| 传感器数据-自动增加时间字段 | 开发,第三次迭代,附加功能开发 | CNLHC,Dicky35 |
| 服务端开发-完善数据管道 | 开发,核心功能,第三次迭代 | CNLHC |
| 服务端开发-提供Mqtt远程过程调用接口 | 开发,核心功能,第三次迭代 | CNLHC,mqlKKK |

第三次迭代中，共有开发任务15项，测试任务8项及1项提议。

### 代码提交

第三次迭代中，我们为12项核心开发功能提交了代码，为6项测试工作提交了代码。

截止项目结束时，有三项附加功能仍未着手进行开发；1个测试用例尚未完成测试；1项测试工具开发任务尚未完成。

### 测试

在第三次迭代中，我们继续为除了RESTful接口的其他部分引入了单元测试。除此之外，我们对系统中的一些组件，包括RPC执行器和嵌入式端的数据处理器进行了手动测试，由于没有系统的方法指导，这些测试并没有被很好的记录下来。

### 问题

在第三次结束后，我们已经实现了需求文档中预设的大部分功能。此外还有几个附加功能尚未实现。

同时，我们的测试工作仅局限于对于代码的单元测试。没有跟踪需求文档中列出的用例进行测试。

考虑到项目即将截止，我们决定放弃未实现的附加功能，并将工作的重点转移到对于系统的集成测试上。

## 测试总结

本次项目的测试主要分为两部分，单元测试和集成测试。

### 单元测试

本项目中的单元测试针对系统中的一个或若干函数组成的子系统进行。 测试时使用单元测试脚本, 仿真若干组正常或异常的输入, 测试系统是否返回期望的输出以及能否正确的处理异常输入。每一个单元测试用例应该保证测试的原子性，在测试代码内部合理的使用断言。 单元测试用例的结果有两种: 通过与不通过。

由于单元测试工作与项目开发工作同步进行，当测试出现问题时会被及时修复，因此，所有单元测试的结果均为通过。

### 集成测试

本项目的集成测试针对整个系统或是系统的一个子系统进行。测试时，以一个或多个用例为基本单位， 测试人员扮演用户的角色与系统交互, 测试系统是否能够提供需求文档用例描述的功能。

关于集成测试的结果，可参见测试报告文档。

### 测试分析

本项目提供的服务端软件,客户端软件和嵌入式端软件中

* 服务端软件的RESTFul服务器组件实现了所有接口的单元测试,接口代码测试覆盖率100%.
* 服务端软件的数据管道组件实现了所有接口的单元测试,接口代码测试覆盖率100%.

其余组件均未编写单元测试。

本项目共提供18个集成测试用例。覆盖了需求文档中列举出的12个用户用例(共16个用户用例)。

## 团队协同总结

按照要求，本节将以个人为单位进行文档撰写。每一位项目组成员撰写的内容占一个小节。

### 刘瀚骋-16231275

我作为组长，负责开发工作的管理。项目开始时，我们决定采用主程序员制小组的方式进行团队的组织，由我来担任主程序员，负责系统架构设计与所有后端项目的编写；孟巧岚同学担任备程序员，负责前端项目的编写；张起铭和邓健同学担任测试程序员，负责进行软件测试与项目运维；许文广同学担任秘书，负责文档的整理与编撰。

本次开发任务的开发管理，可大致划分为三个阶段：有序期，失控期，回暖期。

#### 有序期

“井然有序期”出现在项目开始之后，需求文档评审之前。这一阶段的开发管理井然有序。我会在每周一为每一个人布置本周任务。这一阶段里，主要由我和备程序员撰写需求文档内容，秘书进行文档中UML图片建模，两位测试工程师学习CI/CD及容器工具，为之后的软件测试做准备。大家都能及时完成任务。

#### 失控期

“失控期“出现在需求文档评审之后，第三次迭代评审之前。这一阶段的开发管理失控，开发工作开始不按我的预期进行。

这一阶段主要由我来完成所有的开发/测试/项目运维的工作。此外，这一阶段中备程序员始终在进行前端页面的开发；开发秘书始终在进行文档的维护。但是由于缺乏和我的沟通，他们的进度比较慢。两位开发程序员处于失联状态。

失控的表现主要体现为组员已经不能按时完成我通过Issue布置的任务。于是我不得不自己完成这些任务，逐渐开发工作就演变成只有我一人在参与。

失控的原因是多方面的，大概可以总结为以下两点。

1. 我太忙了。由于我所在的专业本学期仍有30学分的课程，繁重的学业导致我并不能花足够多的时间在这个项目的维护上。为了能够尽快完成任务，当我发现沟通成本过高时，我会直接放弃沟通，自己进行实现，久而久之，会导致其他成员跟不上进度。
2. 设计过于复杂。在一些对于本项目而言不是特别必要的地方引入了过多的复杂性，例如引入sphinx来撰写文档；使用脚本驱动文档内容的生成；使用CI/CD工具等。在由专业工程师组成的队伍里使用这些小trick能够提高工作效率，但是在本项目中引入它们带来了许多本可以避免的复杂性。

#### 回暖期

“回暖期“出现在第三次迭代评审之后。在老师的”挂科警告“下，测试程序员开始主动联系我来参与到项目开发中。但是由于此时项目已接近尾声，再加上临近期末，大家都非常忙碌，我确实没有额外的精力帮助他们跟上项目的进度，只能在github上发布任务时@他们，但是如果他们太久没有响应Issue，我会自己完成任务。

这一阶段，开发已经结束。我和备程序员分别完成了系统的单元测试和集成测试工作。秘书基本掌握了文档构建工具的用法，并帮助修复了一些文档中的遗留问题。两位测试程序员则分别选择了一个我发布在看板上的任务进行实现，虽然他们的代码并没有被应用到实际项目中，但是看得出他们为跟上开发进度付出了一番努力。

### 孟巧岚-16061053

在本项目中，按照计划作为备程序员，协同主程序员完成软件开发任务，与开发秘书完成文档撰写。

在实际开发过程中，主要参与了需求文档的编写、软件前端开发、系统集成测试和测试文档的编写等工作，并通过与团队其他人员的合作与交流，取得了较好的效果。

### 邓健-16061136

在本次项目中，主要是负责编写测试大纲、linux服务器上搭建qemu仿真测试环境以及完成RPC协议超时扩展的开发等工作。

### 许文广-16061069

本次课程这一个学期过来离不开团队的作用，每个人都被分配了不同的角色，并且在我看来大家都各司其职，各尽其用，组长作为领导者和团队的核心人物，带领大家协同完成课程的任务，让我提前感受到有领导模式下的团队作业，收获颇丰。

### 张起铭-16061044

我们团队的分工模式是成员各司其职，由组长将任务具体化为不同的标签发送给小组成员的分配模式。在实际任务成过程中，我主要进行了文档自动化部署和附加功能开发的一部分工作。我认为团队这种任务分配的协同效果还是很不错的，任务具体，任务完成对象明确，成员们可以实时看到各任务完成的情况，这样简明的分配方式有助于提高任务的完成效率

## 感受与建议

按照要求，本节将以个人为单位进行文档撰写。每一位项目组成员撰写的内容占一个小节。

### 刘瀚骋-16231275

#### 感受

1. 好玩。作为电子信息工程学院的学生，专业课学习过程中难免会被各种张牙舞爪的数学公式所包围。像是软件工程这样的实践课程是我难得的消遣
2. 有用。借助本次课程设计，我对之前的一些的想法付诸了实践。比如基于消息队列的RPC框架，基于流的数据处等等，这次课程给了我一个机会去验证他们的正确性。当然最有用的当属我为撰写文档而做的工具Grossular，除了用在本项目中的需求文档和设计文档的撰写工作中，还应用到了另外两门课程的课程设计文档和一个商业项目的需求文档撰写工作中。

#### 建议

希望教师能在这门课程中讲授更多第一手的软件开发经验。

目前这们课程讲授模式在我这留下的印象是：老师会讲授若干理论知识，并告诉你要在课程设计中达到什么目标，但是并不给出具体的解决方案。

例如在各次审阅中，一直强调要求“需求可追踪”。需求可追踪确实非常重要，但是我认为老师去具体的分析若干真实世界中成功的软件开发案例，并结合案例讲解，使用何种工具，践行何种方法论可以实现需求可追踪这一目标可能会对学生更有帮助。

### 孟巧岚-16061053

本学期的软件技术课程将要落下帷幕，回望数月的开发历程，这门课程让我收获颇丰。

关于技术：在接触这门课程前，我对于前端相关技术一无所知，本着‘边做边学’的心态，在其他成员的帮助下，了解学习到了react、redux、css、typescript等前端技术，理解了前端开发相关的基本技术流程，并且从零开始写出了前端的界面。但对其使用仍然不够熟练，对前端界面的真正意义理解不够深刻，前端的开发应该面向用户，让用户能够接受认可并愿意使用。我对这些相关的认知的缺少也直接导致了后期团队对前端的界面否定。其次，了解测试的基本流程，懂得了测试的重要性。测试是一个工程项目的最重要的环节，在一环节中，通过严格对比相关标准，直接决定了项目是否能够投入使用。最后，了解了文档的自动化发布，学会了文档标准化的编写。

关于理论：这门课程让我明白了一个工程项目从提出到实现再到发布的过程和在项目开发过程团队合作的重要性。在此之前，我一直认为这是单纯的代码工作，但事实上代码工作在一个工程项目中只是冰山一角，更重要的是如何制定开发计划、如何让这个工程项目能够顺利按照计划推进、如何去满足客户的需求和如何保证工程的可靠性等。总之，软件工程的开发是一个有着严格流程、严格计划、严格标准的工作，每一个环节都是必要且不可缺少的，所以其需要的也不仅仅是我们的写代码的能力。

建议：1，提供更多的可以选择的项目。2，能够提供更多的技术指导。3，更早的开始迭代。

### 邓健-16061136

这一课程主要是考察团队的协作能力，以及硬件方面的设计与处理。由于之前很少接触这种贴近硬件方面的设计，在项目过程中确实遇到了许多困难，但所幸都在队长的提示下顺利完成了。在这一过程中，也积累了以前比较陌生的硬件方面的一些知识，而不是绝大部分知识都停留于软件层面的状态。

建议：课程可以提供往届的优秀项目展示，让学生能更加清晰的明白项目的管理方法、途径以及一些文档的编写规范。

### 许文广-16061069

本次课程分组是属于我自己找了一个比较不匹配自己的组，所以完成的工作是属于相对简单的文档管理工作。但是在之后开始参与到课程的流程中的时候，文档却是属于最先需要确定下来的方针。我是头一次参与一个比较完整的项目的开发工作，整个学期下来，有紧张的时候，有困难的时候，在逐个解决掉这些问题之后，不得不说软工让我这一个学期的生活变得很充实，而且在看到自己参与的项目渐渐有成果的时候心中充满了成就感。非常感谢组长肯带我这个吊车尾的，还耐心教会了我很多东西，同时也非常感谢软工课程组的所有老师以及助教，一路走来，您们辛苦了，感谢！

对课程的建议：

1、可以鼓励学生创造性的使用或者构建自己的项目体系、工具、整体架构。

2、课程组需要有多元化的选择，可以迎合学生的兴趣来增添项目（执行起来比较困难）

3、保持课程中的流程，这一点我觉得非常重要，时间线管理可以最大程度推进任务进展。

### 张起铭-16061044

一学期的嵌入式软件工程课上下来，感触颇深。课程中我学习了软件工程理论的一些基本方法，之前我一直觉得工程项目就是单纯的写代码，但深入了解软工理论后，才知道工程化方法和团队管理对工程的重要性。课程组选择嵌入式工程作为课程项目，嵌入式系统同样是我知识上的盲点，而课程提供了这样一个让我参与到软硬件结合工程的机会，也让我初步了解了嵌入式系统的工作方法。同时也想在这里为课程改进提三点建议：一是希望课程可以讲解一些课程项目的主流实现方法和系统框架设计，让学生可以更好的设计完成自己的项目。二是希望课程可以优化各阶段的小组汇报过程，各阶段的小组汇报过程冗长乏味，一般参与者只有该小组和老师，希望能以一种让各小组对比不同实现方法的方式，增加汇报过程的参与感，同时减少汇报过程的时间。三是希望课程可以提供更多选择，比如课程项目的选择，工作模式的选择等。