shattuckite 项目计划书

发布 *rc0.0*

CNLHC

2019 年 03 月 05 日

Contents

1 项目背景

`shattuckite` 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.

1.1 开发动机

本项目首先作为 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。

1.2 开发人员

本项目由一个规模为 4 人的小型团队进行开发. 团队成员情况如下

姓名	学号	学院
刘瀚骋	16231275	高等理工学院
孟巧岚	16061053	计算机学院
邓健	16061136	计算机学院
许文广	16061069	计算机学院
张起铭	16061044	计算机学院

1.3 面向用户

本项目面向的用户分为三类:

1. 终端用户

普通的软件使用者。假定这些用户仅有基本的计算机或手机的使用技能，他们对计算机程序的工作原理一无所知。

2. 普通工程人员

系统管理者或是分销商。他们不一定拥有计算机程序设计背景，但是可以付出相对于普通用户更多的成本学习本系统的部署，配置和使用。他们期望使用本项目, 为普通用户提供服务。

3. 开发人员

程序开发者。他们拥有计算机程序设计的工程能力, 具有理解本项目运行原理的能力。他们期望扩展或修改本项目的功能以实现自己的额外的需求。

4. 课程教授

特殊用户, 项目甲方。他们将会根据本项目的完成程度及开发管理组织情况对开发团队成员进行评价。

2 项目约束

2.1 时间约束

本项目甲方要求按照下表列出的时间节点提交相应文件

工作名称	DDL
自由组队	2019 年 3 月 4 日
《项目开发计划书》编写	2019 年 3 月 10 日
《需求分析文档》提交	2019 年 3 月 31 日
需求分析评审	2019 年 4 月 2 日
设计文档提交	2019 年 4 月 21 日
设计文档评审	2019 年 4 月 23
代码评审	2019 年 5 月 14 日
测试文档评审 1	2019 年 6 月 4 日
测试文档评审 2	2019 年 6 月 11 日
课程总结	2019 年 6 月 18 日

2.2 硬件约束

本项目甲方会提供基于 `Power PC` 的嵌入式开发板及基于 `LoRA` 的物联网传感器套件。

2.2.1 嵌入式终端

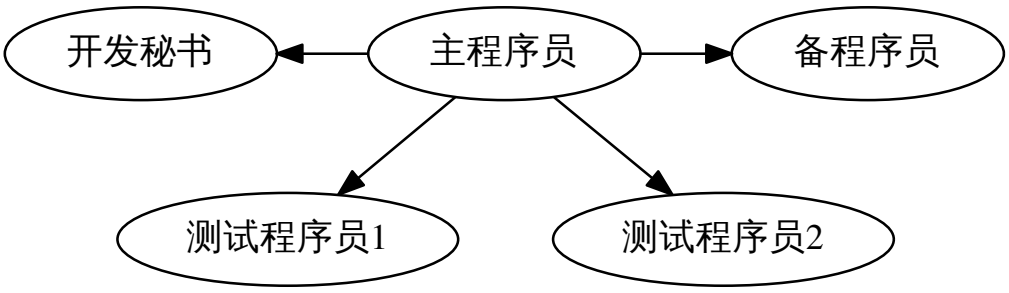
考虑到本项目主要面向民用领域，部署 `Vxworks` 和 `Power PC` 的成本较高。此外，在与甲方负责人交流后得知，最终产品并不需要必须支持上述软硬件件平台。

本项目最终决定使用嵌入式 `Linux` + `ARM` 的组合作为嵌入式终端, 并计划将系统部署到 `Cortex-A8` (基于 Texas Instrument AM3354 Sitara Processor) 和 `Cortex-A9` (基于 Altera Cyclone V SOC FPGA) 两个硬件平台。

3 开发计划

3.1 成员组织

本次开发计划使用类似主程序员小组的方式进行小组组织。组织结构如下图所示。



组织方式示意图

各组员小组角色分配如下

姓名	学号	学院	小组角色
刘瀚骋	16231275	高等理工学院	主程序员
孟巧岚	16061053	计算机学院	备程序员
邓健	16061136	计算机学院	测试程序员
张起铭	16061044	计算机学院	测试程序员
许文广	16061069	计算机学院	开发秘书

3.2 角色职责分配

本节叙述不同小组角色的职责

3.2.1 主程序员

- 管理项目开发进度。
- 设计项目开发路线与体系结构。
- 具体实施软件开发及硬件设计工作。
- 分割开发任务, 与备程序员协同完成开发任务。
- 与测试程序员交流, 完成软件测试工作。
- 与开发秘书共同完成文档的撰写。

3.2.2 备程序员

- 与主程序员交流，具体实施软件开发任务。
- 与开发秘书共同完成文档的撰写。

3.2.3 测试程序员

- 与主程序员交流，设计各个构件的测试方法, 并编写《测试文档》。
- 编写测试代码, 完成软件测试。
- 软件可持续集成与可持续交付的相关工作。

3.2.4 开发秘书

- 与主程序员交流, 完成《需求分析文档》及《设计文档》的撰写
- 管理并发布软件开发过程中产生的各种文档
- 负责与甲方交流, 确保开发团队及甲方始终保持共识。

3.3 流程控制

3.3.1 流程模型

本项目要求一定要在规定的日期前交付至少一个版本, 因此决定采用类似快速原型模型的流程进行开发。

本项目将开发过程中的一次迭代划分成四个环节:

讨论调研

讨论调研时期，开发团队以访谈调查问卷等方式，与甲方以及其他项目面向的用户进行交流，获取用户的最新需求，并确保用户的需求是可行的。

需求分析建模与软件设计

对讨论调研得到的信息进行抽象, 整理; 并撰写或修改《需求分析》文档。

在需求分析建模结束后，根据《需求分析》文档的修改情况，更新《系统说明文档》的内容，对系统实现的设计进行修订。

编码与测试

根据最新版本的《系统说明文档》, 对系统构件进行增删或修改；与此同时，更新相应的测试用例。

发布

完成编码与测试后，根据修改情况，进行版本号控制 (详见版本号控制)。并编写相应的脚本，进行数据库，配置文件等的迁移，将新系统部署到生产服务器。

3.3.2 模型实施

在开发初期我们将按照: 讨论调研-需求分析建模-编码与测试-发布的流程完成第一次迭代。

在第一次迭代完成后, 讨论调研-需求分析建模与编码测试-发布这两个过程将独立异步进行。

为了保证时间节点与甲方的要求相同, 我们计划在 2019 年 3 月 31 次前完成首次迭代过程, 发布一个原型版本。

3.4 版本号控制

3.4.1 构件版本号控制

我们将基于构件进行版本号控制。即每一个构件均拥有自己独立的版本号。

构件的版本号将按照 Semantic Versioning 2.0 标准进行编制。

在版本号变动时，必须在构件仓库目录添加新版本的发布说明，简要的阐述该版本的修改。

构件版本号的主版本号和系统版本号控制 中的系统版本号的主版本号相同。

3.4.2 系统版本号控制

整个系统拥有一个独立的系统版本号。系统版本号同样使用 Semantic Versioning 2.0 标准进行编制。

在开发过程中, 系统版本号的修订号不修改，始终为 0. 完成一个迭代后, 在发布阶段递增系统版本号的次版本号。

系统次版本号递增时，需要同时发布当前版本的系统程序包含的构件版本以及概述本版改动。

3.4.3 文档版本号控制

开发过程中产生的文档应当拥有独立版本号。文档版本号使用 `X.Y` 两级格式。X 和 Y 分别是主版本号和次版本号。

文档的次版本号在每次迭代时进行递增。

文档的主版本号和系统及构件的主版本号一致。

对文档的内容进行小范围的修改 (例如修改错别字，笔误，优化表达方式等) 时, 不为文档分配新的版本号, 在文件名后加入独立的且不会重复的字符串以示区分。

3.5 交付产品

本项目结束时计划交付的内容包括

项目名称	类型	负责人
《项目开发计划书》	文档	主程序员
《需求分析》	文档	主程序员, 备程序员, 开发秘书
《系统设计说明》	文档	主程序员, 备程序员, 开发秘书
《测试文档》	文档	测试程序员, 开发秘书
《系统部署指南》	文档	主程序员, 开发秘书
测试程序	源代码	测试程序员
系统程序	源代码	主程序员, 备程序员

3.5.1 《项目开发计划书》

本文档用于阐述项目的开发计划。主要内容包括项目背景介绍，开发团队的组织管理方式，开发流程控制，计划交付内容及交付时间节点及风险控制等元信息。

3.5.2 《需求分析》

基于场景和数据两方面, 尽可能全面的分析本项目的需求。

基于场景的需求分析将从用户的视角出发, 讨论用户会以何种方式与本系统发生交互。使用自然语言和 UML 活动图, 对所有可能发生的交互场景进行建模。

基于数据的需求分析将结合基于场景的分析结果, 讨论为满足用户使用场景, 系统需要维护的数据和系统需要暴露的控制接口。并使用 DFD(Data flow Diagram) 分别对

1. 数据在系统中的传递与持久化
2. 控制信号的传递

进行建模

3.5.3 《系统设计说明》

系统设计说明将自顶向下的描述系统的具体实现方式。自顶向下的顺序分别是设备级, 构件级, 类/方法级, 实现级四级。

设备定义为物理上独立的实体。设备级设计将规划系统运行所需要的设备以及设备间互联的基本方案。计划使用 UML 部署图来建模这一级的设计。

构件是运行在设备上, 且逻辑功能较为独立的软件。一个设备上可能会运行多个构件。构件级设计将规划每个构件实现的具体功能，使用的具体技术栈以及构件间的互联方式。计划使用 UML 部署图来建模这一级的设计。

类/方法是组织逻辑的最小单元。若干个类/方法相互协作，组合成构件。计划使用 ER 图，UML 类图来建模这一级的设计

实现指类/方法的具体代码。实现级文档不单独撰写，将会根据编码过程中的注释自动生成。

3.5.4 《测试文档》

测试文档计划阐述本项目的测试策略以及运行测试程序的方法。

3.5.5 测试程序

本项目计划使用黑盒测试的方法，编写单元测试。

3.5.6 系统程序

系统程序将以构件为单位进行发布。对于运行在嵌入式平台 (非 x86_64 架构 CPU) 上的程序，计划发布交叉编译后的二进制文件; 对于运行在通用平台 (x86_64 架构 CPU) 上的程序, 计划以 docker 容器的形式和源代码 + 构建脚本两种形式进行发布。

3.6 时间节点

4 风险控制

4.1 风险分析

本项目面临的主要风险是系统无法按期交付导致无法获取成绩。

无法按期交付的原因是开发过程种遇到无法解决的技术难题。

4.2 风险控制

为了规避风险，本项目计划尽早发布第一个原型版本。减少第一个原型的开发时间，以期迫使开发小组不在首次需求设计时引入太多需求而导致遇到无法解决的技术难题；此外，减少第一个原型的设计时间能够留给团队更多审视过去工作，并对开发做出修正的时间。

本项目计划在 2019 年 3 月 31 次前完成第一个版本的迭代。

5 其他

5.1 团队 KPI 管理

为了合理分配甲方提供的开发报酬 (浮动分数), 开发团队将使用 KPI 衡量个人绩效。具体规则如下:

1. 团队中的每个成员有 1000 分基础 KPI 点数
2. 项目经理 (主程序员) 进行任务划分, 为每位成员安排相应的任务, 并根据任务的复杂程度, 为任务规定一个分数。
3. 若成员未能在规定时间内完成任务, 则扣除任务分数的一半。
4. 若成员在延期 1/3 的原定工作时间后仍未完成任务，扣除全部任务分数。
5. 最后根据每个人的剩余 KPI 比例来分配分数