**概要设计报告（软件）**

# 概述

* 1. 目的：

设计一套后台处理软件，实现基于Ubuntu系统的ROS开发框架下，接收（地面站）上位机发送的控制指令（topic），对相关作业文件进行分段解析，根据指令内容需求分段下发对应作业文件内容，订阅并解析外接传感器数据，实现路点可变，两种工作模式控制机械臂及其他执行器完成带电作业流程中的每个作业步骤，并且反馈当前进行的步序状态，自动作业过程中可切换手动作业模式。

* 1. 应用范围：

配电网带电作业机器人系统主控后台管理

# 总体设计

## 系统架构

2.1.1 系统组成：

主控系统主要由消息监听器，调度器，文件解析器，文本提取，变量解析与赋值，命令执行几部分组成。

2.1.2 相互联系：

主控启动同时会创建并启动调度器，等待调度段到来；接收到地面站消息会调用解析器，同时根据消息内容进入调度器调用指定作业文件并解析，解析器需要调用文本提取模块分段逐行解析，解析结果会将文件分成不同调度段，每段包含不同命令块，每块包含不同命令行；开始依次调度段中的命令行并且执行命令行，每执行一行内容会进入反馈等待，收到反馈之后再次执行下一条命令行。

2.1.3 外部接口：

（1）通过统一话题与消息格式的订阅与发布；（2）服务请求与回应。

2.1.4 组成部分功能

（1）消息监听：监听地面站、双机械臂，雷达滑台消息

（2)调度器：调度需要执行的作业内容

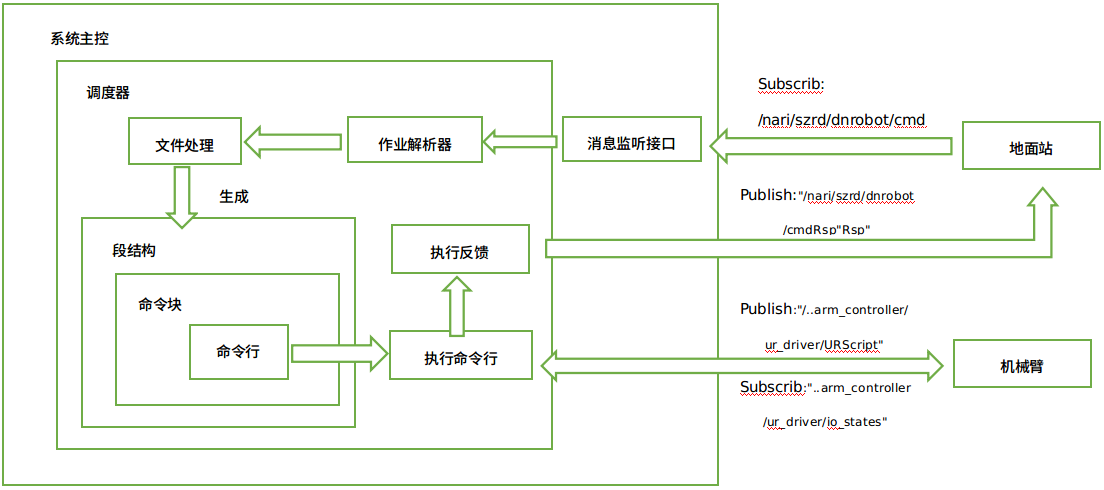
（3）文本解析器：实现分段解析作业文件、话题消息

（4）文本内容提取：文件读取与逐行数据读取

（5)命令执行器：命令发布与服务呼叫

（6）变量操作：变量生成、释放、输出等系列操作

2.1.5 系统架构图



2.1.6 系统元素组成一览表

详见：系统元素一览表.xlsx

## 2.2 系统功能逻辑流程

系统功能启动---创建并启动调度器---等待调度段到来---接收cmd\_msg并逐行解析---根据解析内容解析指定作业文件生成segment---根据关键信息匹配需要执行的段与方式（顺序，并行，计算）---发送控制指令---等待反馈---收到反馈通知---接着往下执行......

系统流程图如下：

## 2.3开发及运行环境

### 2.3.1 硬件环境

工构机、笔记本电脑、台式机均适用

### 2.3.2软件环境

操作系统：Linux、ROS

相关库：rapidjson、Eigen、c++

程序设计语言：Linux C++

开发环境：visual studio code

# 3 模块设计要求

## 3.1 模块1（回调监听）

### 3.1.1 功能和性能

功能：

（1）地面站命令消息监听接口

（2）左、右臂执行结果监听接口

（3）通用数字量信息监听接口

（4）通用模拟量信息监听接口

（5）控制命令监听后处理

### 3.1.2 输入

1 输入参数:详细说明--->数据类型与结构

（1）监听地面站命令消息接口：

header:

msgID: 0

srcNodeType: "RemoteControl"

srcNodeName: "RemoteControl"

dstNodeType: "MainControl"

dstNodeName: "MainControl"

msgType: ''

msgTime:

secs: 0

nsecs: 0

object: 200

type: 0

content: "{\"cmdArray\": [{\"seq\":1,\"type\":\"uint8\_t[]\",

\"value\":[3,1,1,1,4,1,56,0,0,0],\"\name\":\"cmd\"}]}"

（2）监听左、右臂执行结果接口消息格式：

data: "def function():\n set\_digital\_out(0,False)\n

set\_tcp(p[0.011,0.0,0.02,0.0,0.0,0.0])\n\\ set\_payload(5.0)\n movel

(p[-.130608531530,-.431269092087,.125809174659,-.001220983028,3.116276481241,.038891916168],a=0.25,v=1.2)\n\ \ set\_digital\_out(0,True)\nend\n"

（3）监听通用数字量信息、通用模拟量信息（水平，竖直滑台位置）接口消息格式：

header:

msgID: 0

srcNodeType: ''

srcNodeName: ''

dstNodeType: ''

dstNodeName: ''

msgType: ''

msgTime:

secs: 0

nsecs: 0

object: 200

type: 0

content: "{\"cmdArray\": [{\"seq\":8,\"type\":\"uint8\",\"value\":1,

\"name\":\"cmd\"}]}"

3.1.3 参数入口数：5

如下：（1）地面站消息接收接口

ros::Subscriber cmd\_msg\_sub = nh.subscribe(COMMAND\_MESSAGE\_LISTEN\_TOPIC, 1000, cmd\_call\_back); // 命令消息

void cmd\_call\_back(const controller\_pkg::cmd\_msg& msg)

（2）左、右臂执行反馈消息接口

例：左臂执行反馈消息订阅

ros::Subscriber cmd\_rsp\_left\_arm\_sub = nh.subscribe(COMMAND\_RESPONSE\_LEFT\_

ARM\_LISTEN\_TOPIC, 1000, cmd\_rsp\_left\_arm\_call\_back); // 命令回复消息

void cmd\_rsp\_left\_arm\_call\_back(const controller\_pkg::IOStates& msg)

（3）通用数字量、模拟量监听接口

### 例：数字量监听

### void status\_digital\_call\_back(const controller\_pkg::status\_digital\_msg& msg)

### 3.1.4 输出

1 模块出口数：2

1. 输出json文本内容，字符串格式如下：

例："{"cmdArray": [{"seq":0,"type":"uint8\_t[]","value":[1,2,3,4,5,6,7,8,9,10,11],"name":"cmd"}]}"

1）输出值:json格式文件；2）取值类型：字符串格式数组。

（2）输出需调度的文件名称以及需要调度的段号：

例：./doc/job\_file/C0x0xox.....0x.csv

### 1）格式：需要指定文件路径以及文件名称，

### 实现方式

自行开发，可随时根据需求调整代码逻辑，添加对应处理函数与功能

3.2 模块2（段调度器）

### 3.2.1功能和性能

功能：

1. 根据解析topic内容的序号组调度段
2. 调度器运行
3. 命令行运行（命令行回复类型）
4. 按照输入序号执行调度段，顺序块、并行块、计算块
5. 发送当前段执行状态
6. 等待左右臂到位信号触发下一条命令执行
7. 等待手工操作确认信号，滑台位置确认、到位、归零等信号
8. 收到左右臂执行反馈通知
9. 段到来通知
10. 绑定命令函数

### 3.2.2输入

1：输入参数:详细说明--->数据类型与结构

1. 输入文件路径、段号

例：./doc/job\_file/C0x0xox.....0x.csv

1. 输入组段编号

例：[3,1,1,1,5,0,7,8,0,0,0]

2：参数入口数：1

获取作业文件路径和执行序号组

bool SegmentScheduler::schedule(const std::string& path, const std::vector<int>&

segment\_idxs)

3：输入校验：

（1）判断路径是否为空

（2）判断执行解析状态

（3）判断段号idxs是否合法

### 3.2.3输出

### 1 模块出口数：4

1. 输出解析文件路径；

例：JobParser(./doc/job\_file/C0x0xox.....0x.csv)

1. 输出命令行内容；

例：#SEGMENT 1 ARM\_CONTROL BEGIN

1. 绑定命令行函数；

例：func\_map\_[JOB\_LINE\_TYPE\_MOVEJ] = movej\_line\_func

### 获取命令行类型

例：get\_job\_line\_type(line)

### 3.2.4实现方式

自行开发，可随时根据需求调整代码逻辑，添加对应处理函数与功能

3.3 模块3（解析器）

### 3.3.1功能和性能

（1）模块功能

1）：获取并返回作业命令行类型

2）：获取作业文件并解析

3）：实现解析功能，返回文件解析状态

4）：变量解析，返回文件状态

5）：格式化输出文件状态

### 3.3.2输入

1：输入参数:详细说明--->数据类型与结构

（1）获取某一命令行类型

例:get\_job\_line\_type(#MOVEJ,1,0,0,0,0,0,0,-0.64,-1.9,-1.5,-1.0,1.5,-0.1,0,0,0,0,0,0.2,0.3)

（2）获取作业文件路径

例："doc/job.txt"

（3）获取含变量的行

例：#MOVEJ1,1,0,0,0,0,0,0,-0.64,-1.9,-1.5,-1.0,1.5,-0.1,var\_TCP\_vartcp,0,0.2,0.3

2：参数入口数：3

分别如下：（1）JobLineType get\_job\_line\_type(const std::string& line)

（2）JobParser::JobParser(const std::string& job\_path)

（3）JobParser::Status JobParser::parseVariables(std::string line)

### 3.3.3输出

1 模块出口数：2

1. 输出需要打开的文件路径：

例：read\_file(job\_path\_.c\_str())

1. ：输出需要解析的行

例：getline(content, line)

### 3.3.4实现方式

自行开发，由于随时有需求调整，代码逻辑发送变动，需要添加对应处理函数

## 3.4模块4（文件处理）

### 3.4.1功能和性能

1：内部实现格式化生成string

2：读取指定路径文本文件

3：从字符串中读取行

4：从字符串中读取行，将读取的所有行存入容器并返回它

5：解析回复命令消息错误码

6：给地面站发送指令消息

### 3.4.2输入

1：输入参数:详细说明--->数据类型与结构

（1）读取指定文件：

例：read\_file("doc/job.txt")

（2）输入文本内容

例：#MOVEL2, 2, var\_PP\_varpp, var\_TCP\_vartcp, 5.0, 1.2, 0.25

#MOVEJ,1,0,0,0,0,0,0,-0.64,-1.9,-1.5,-1.0,

#MOVEL,2,-.13,-.43,.12,0.00,3.11,.03,0.0

........

2：参数入口数：3

（1）输入文本路径

std::string read\_file(const char\* path, const char\* delim)

（2）输入文本内容

int getline(std::string& content, std::string& line, const char delim)

### 3.4.3输出

无输出，仅有反馈值

### 3.4.4实现方式

自行开发，可随时根据需求优化处理逻辑，优化代码结构

## 3.5模块5（命令行处理）

### 3.5.1功能和性能

1：发送滑台运动指令

2：生成move脚本，并返回脚本

3：根据命令函数类型与机械臂ID号自动匹配控制

4：呼叫滑台写服务，根据seq、value、name参数值获得滑台参数及状态

5：发送允许手动操作信号给地面站

### 3.5.2输入

1：输入参数:详细说明--->数据类型与结构

（1）输入seq（int）、value（double）、name（string）

call\_slider\_write\_srv(scheduler.getWriteServiceClient(), SERVICE\_SEQ\_HORIZONTAL\_SLIDER\_REACH, pos, "HorSliderGo");

（2）输入机械臂相关控制信息：

argname\_to\_value("TCPX") is\_move\_j ? "j" : "l", "p",

（3）执行调度器与命令行

CmdLineFuncRsp movel2\_line\_func(SegmentScheduler& scheduler,const std::string&

line)

2：参数入口数：12

例如：

CmdLineFuncRsp movel2\_line\_func(SegmentScheduler& scheduler, const std::string& line); CmdLineFuncRsp waittime\_line\_func(SegmentScheduler&, const std::string& line);

CmdLineFuncRsp var\_assignment\_line\_func(SegmentScheduler& scheduler, const

std::string& line);

### 3.5.3输出

1 模块出口数：9

（1）输出要执行的脚本，字符串格式如下：

script = format(output\_template,

argname\_to\_value("TCPX"), argname\_to\_value("TCPY"), argname\_to\_value("TCPZ"),

argname\_to\_value("TCPRX"),.........

（2）给地面站发送手动操作确认信号

例：scheduler.sendManuallyOperationSignal(0);

1. ：输出话题消息，字符串格式：

例：{"cmdRspArray": [{"seq": 6, "type": "uint8\_t", "value": 2, "statusCode": 4}]}

### 3.5.4实现方式

自行开发，应设计需求随时变动，需修改处理逻辑，优化代码结构

# 4 系统全局数据结构设计

本系统无复杂数据结构设计，系统包含1个全局变量，在回调中使用

std::shared\_ptr<SegmentScheduler> scheduler;

# 5 CBB策略

5.1【鼓励项目使用已有可共享技术和模块，如果有请说明】

可共享模块有：解析器，文件处理，命令行处理等模块均可实现代码复用

## 5.2 拟共享技术需求分析

【本项目对拟采用的公司货架产品（CBB）的技术需求分析】

（1）解析：针对本系统需求，虽然作业文件不唯一，指令内容更不完全一样，所以解析条件不同，但是会有很多固定格式，如果不设计通用模块，会导致代码量很大，也不符合程序设计要求，用起来也很不方便，由于处理方法大类似，故而设计一套通用文件解析器，不仅能大大减少不必要的代码量，给人视觉美感，还能逻辑清晰，通俗易懂；

（2）文件处理：无论哪个作业文件，都只需要读取全部内容，逐行输出，对内容没有要求，因此只需要一个文件处理模块即可；

（3）指令处理：观察所有move指令发现，格式有很相似的地方，只是参数之间差异，所以处理过程类似，这将很有必要设计公用函数处理模块，能减少很多重复性操作代码，实现代码优化。

## 5.3 拟采用CBB适应性分析

【对本项目拟采用的CBB进行适应性分析，确定拟采用的CBB的功能特性适合本项目使用】

（1）解析器：主要实现作业文件内容解析，输出需要调用的命令行，因此对作业文件本身没有要求，只是需要增加对作业文件命令行类型判断即可，只要能正确匹配到作业类型，找到对应作业文件就可以成功解析文件内容，故而适应性很强；

（2）文件处理：次部分只需要对对应作业路径下的文件进行逐行读取，对内容无要求，所以适应性没有任何限制；

（3）命令执行函数：这里主要针对move函数做优化处理，由于move指令无论是否有变量，格式都一样，所以只是参数不同而已，适应性没问题。

## 5.4 项目成果可重用建议（CBB贡献）

【对项目产生的CBB提出可重用建议】

暂无

## 5.5项目CBB的开发策略

【提出CBB开发策略，比如：完全新开发、部分开发等】

（1）解析器：自行开发，设计一个解析器，满足所有move、超时、以及末端工具等指令的解析；

（2）文件处理：自行开发，设计一个模块，实现所选定作业文件的内容读取，并得到想要的数据；

（3）指令处理：自行开发，所有move指令下发前针对同一类型不同参数、变量做对应处理然后下发对应消息。

## 5.6 CBB需求及开发评估

【汇总形成CBB统计表】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **层次** | **CBB名称** | **CBB来源** | **应用说明** | **开发说明** |
|  | 解析器 | 自行开发 | 实现作业指令类型判断、逐行解析作业文件 | 使用枚举类型对命令行进行分类处理，根据作业类型逐行解析文件内容 |
|  | 文件处理 | 自行开发 | 逐行读取作业文件 | 解析器调用文件处理模块时，逐行区分并读取指定的文件 |
|  | 命令行处理 | 自行开发 | 按照输入参数不同执行不同作业指令 | 调度器调度命令行的时候会区分命令行类型，根据不同命令行函数做相应处理之后，将参数统一传到同一个函数处理、赋值发布 |

# 

# 6 可靠性设计

6.1 硬件故障：

已设置系统运行时实时跟踪运行情况，实时反馈执行的作业步骤与做业状态，每一步都会设有详细出错信息提示，可根据错误提示找出存在的问题然后直接解决问题（如网络通信问题）；

另外，硬件故障发生时，主控未正常检测到相关信号会自动切换成手动模式，提示地面站可手动接管完成相关作业流程

6.2 误操作或非法操作：

当用户已选择要执行某个作业流程，而且已经开始执行某个命令行的时候，用户将无法手动操作，只有当该命令行执行完成，主控给用户发送可手动操作信号之后，用户才能手动操作，这样即避免误操作，而且主控会有异常自动处理机制，只有消息指令等均无异常才开始执行。

6.3 关键程序和数据

对应关键程序和数据均会与其他元素分开，关键程序会作为一个独立的模块，实现代码复用，高效开发易检测易维护；关键数据则会与其他数据分割独立存取，易检测、维护与处理。

6.4 出错信息以及解决办法

详见表：错误及处理.xlsx

6.5 系统维护设计

针对每个可能会出错的点都会设计有校验提示，遇到意外错误发生时，都会有针对性的提示信息，若遇到问题可根据提示解决；对于一些模块的检测，会有对应的测试脚本，可检测出错的原因，针对出错内容做维护。

为了维护方便，函数命名已遵循程序语言规范，类似功能的函数已经进行了归类，不同功能的函数分别独立，彼此互不影响。

6.6 可测试性设计

6.6.1 说明本软件系统进行模块测试、子系统测试、系统测试等测试时，设置测试点的技术。

（1）模块测试：采取唯一变量，编写对应模块测试的作业文件，减少其他因素影响；

（2）子系统测试：主要针对执行逻辑做对应测试；

（3）系统测试：会模拟整个工作站执行工作，测试主控与地面站、机械臂等部分的交互逻辑。

6.6.2 说明模块中测试设计原理，测试接口，测试内容和范围。

系统测试时，会针对地面站给主控发送的消息格式以及内容写对应的测试脚本，测试时执行脚本模拟地面站发包，可检测执行逻辑、对应功能是否能实现，性能，是否能出现bug等多种情况；

6.6.3 说明子系统中测试设计原理，测试接口，测试内容和范围。

子系统测试时，针对不同内容写对应测试脚本，例如：某个机械臂执行完成某一条指令后，需要等待执行反馈，这时候需要针对反馈写测试脚本，测试时执行脚本即可。

6.6.4 说明系统中测试设计原理，测试接口，测试内容和范围。】

由于该软件系统是由模块化层层递进关系设计，后续模块对之前的模块有依赖关系，所以每当完成一个模块时都会反复测试，确保没有问题，例如解析模块，会针对不同的作业文件以及作业内容进行反复测试，确保能百分百正确解析出作业内容，主要测试对象还是针对模块不同的输入，输出值作为校验值进行检测，根据输入参数的不同检测执行逻辑与结果是否符合设计需要。

7 接口设计

7.1 用户接口

这里会给用户提供一个主控系统启动脚本，直接执行脚本即可，操作简单；主控系统启动命令：catkin\_ws$ ./script/run\_controller.sh；启动成功系统会提示：

"============ controller started ============"

7.2 外部接口

【说明本系统同外界的所有接口的安排，包括软件与硬件之间的接口、本系统与各支持软件之间的接口关系。】

7.2.1 软件与硬件之间的接口：无

7.2.2 本系统与外部软件的接口关系：

本系统软件基于Linux Ubuntu 16.04或者18.04操作系统下的ros开发平台设计，与外部设备是根据ssh协议在同一局域网下建立连接通信;

（1）系统与雷达驱动：通过topic"/sys\_X/server\_position"获取雷达数据；

（2）系统与水平、竖直滑台驱动：通过server"/pkg\_slider/srv\_slider\_control"通信

（3）系统与左机械臂驱动：通过topic"/r\_arm\_controller/ur\_driver/URScript"控制；

（4）系统与右机械臂驱动：通过topic"/l\_arm\_controller/ur\_driver/URScript"控制；

（5）系统与地面站：通过topic "/nari/szrd/dnrobot/cmd"通信；

## 7.2.3内部接口

（1）系统元素包括：调度器、解析器、消息接口，文件处理，命令行处理；

（2）元素之间接口安排：

1）消息接口与调度器：消息接口接收到消息之后会将作业文件路径与作业文件名一同传给调度器，或者接收消息之后进入调度等待确认信号；

2）调度器与解析器：调度器会将作业文件路径和作业文件名传给解析器，解析结果再传给调度器统一调度命令函数；

3）解析器与文件处理：解析器将文件名传给文件处理模块，实现文件逐行读取，之后将读取的内容返回给解析器做处理；

4）调度器与命令处理模块：调度器会根据调度类型绑定对应函数，再将命令行传给命令行处理模块；返回执行结果。

# 8 关键技术问题说明

【说明系统在设计和开发过程中，需掌握哪些关键技术（包括关键组件的获得等），有哪些技术难点和不确定的技术问题，说明对这些关键技术的解决方案，是否有专利限制，并分析技术上的风险（系统设计、软件质量、项目进度方面的风险）等。对于纯软件项目还需说明获得技术专利的费用。】

8.1 系统涉及关键技术：

（1）文件解析

（2）智能调度

（3）雷达坐标转换

（4）机械臂左右手互换

8.2 技术难点：

（1）雷达坐标转换算法实现

（2）机械臂左右手互换算法实现

8.3 不确定技术：

（1）雷达坐标装换

8.4 针对关键技术解决方案：

（1）文件解析：采用c++文件I\O操作和JSON Document进行程序设计，共同实现作业文件读取和分段解析；

（2）智能调度：根据系统逻辑框架进行有效分析和详细设计，所有执行步骤均由调度器全局调度处理，开启多线程并行处理；

（3）雷达坐标转换：首先了解雷达坐标系与其他坐标系空间位置关系，根据坐标系关系进行算法设计，之后算法移植，再进行实物验证；

（4）机械臂左右手互换：首先进行数学分析，了解不同坐标系之间的关系，建立空间位置关系，根据需求先进行仿真设计，之后再运用在实物机器人上进行理论验证，通过验证结果进行算法设计，设计完成之后进行算法移植、仿真验证，实物验证。

8.5 技术风险：

1. 系统设计

由于开发时间有限，而且提供的技术文档不详细，整个系统设计仓促，完全由模块拼接而成，并且需求不固定，随时增添新需求，模块内容随时修改，导致系统随时需要优化逻辑，匆忙设计导致代码没能复用，只能逐渐优化代码结构。

1. 软件质量

经过模块反复测试，系统多次模拟测试，和系统联调，系统各模块功能无异常，能正常处理，系统测试结果无异常，逻辑清晰，消息接收、文件管理、消息发布无异常。

1. 项目进度方面的风险

由于项目时间有限，且需求不断增加，所以项目进度随时需要调整，原本计划均须提前完成，任务量大，进度安排欠合理，导致在系统设计上只能先考虑功能实现，而无精力从优设计。

**共 21 页 编制：何小勇 日期：20190730-20190810**

# 审核： 日期： 批准： 日期：