目录

版本号: v0.5

目录

第一章 概述

1.1 文档目的

本文档适用于 MES 系统和叉车调度系统进行交互通信。

1.2 接口类型

本文档提供的接口基于 TCP 协议通讯， 调用接口可参考第三章 示例.
若需要其他通讯协议接口，请参考链接：

https://github.com/RobotWebTools/rosbridge_suite/blob/groovy-devel/ROSBRIDGE_PROTOCOL.md

http://wiki.ros.org/rosbridge_suite

第二章 基础接口：

以下 String 类型的参数必须使用英文字符，但叉货工位和卸货工位可以使用中文字符，以方便用户操作， 编码方式为 UTF-8 编码.

激光叉车服务器 IP: 192.168.10.36
端口号： 7070

2.1 监控调度系统状态：

MES 只需要发送如下数据 1 次，即可定时收到服务器状态， 叉车服务器发布系统状态的频率暂定 1 秒 1 次。

MES 发送数据：

```
{
    "op": "subscribe",
    "topic": "/server_status",
    "type": "scheduling_msgs/ServerStatus"
}
```

服务器收到如上数据后， 将定时向 MES 发送数据：

```
{
    "topic": "/server_status",
    "msg": {
        "text": "叉车",
        "isReady": true
    },
    "op": "publish"
}
```

说明: isReady， bool 类型， 如果 isReady 为 true，server 已经准备好，可以开始调度， 反之为未准备好。

Text 为 string 类型，状态说明， 可选。

程序退出前 MES 发送如下数据， 退订：

```
{
    "op": "unsubscribe",
    "topic": "/server_status"
}
```

2.2 添加或修改任务：

本指令含义： 呼叫一台空叉车去 loading_station 站叉货， 卸货到 unloading_station 站。unloading_station 站未指定时， 叉车只去 loading_station 站叉货， 待 MES 指定 unloading_station 站后再去卸货。

MES 发送数据：

```
{
    "op": "call_service",
    "service": "/add_or_modify_forklift_task",
    "args": {
        "task_id": -1,
        "loading_station": "切割机 1_1",
        "unloading_station": "冲压机 2_2"
    }
}
```

参数说明:

task_id： int32， 任务 ID ，MES 填入-1 为新建一个任务， 服务器接受任务后会返回一个任务 ID 给 MES。MES 如需修改已经存在的任务，需填入要修改的 task_id。

loading_station： 叉货工位名称， 必须指定， 且不可修改

unloading_station：卸货工位名称，可先不指定，不指定时填入字符串"unknown"，MES 可再次调用此接口修改此参数， 修改时需填入服务器返回的 task_id。

叉车服务器返回数据格式如下， 每次调用只向 MES 发送一次：

```
{
    "values": {
        "feedback": 1
    },
    "result": true,
    "service": "add_or_modify_forklift_task",
    "op": "service_response"
}
```

返回数据参数说明：

Feedback，int32， 返回负值为错误码， 返回为 0-int32 最大值为服务器为此任务分配的任务 ID。

错误码： -1：工位错误； -2： 任务 ID 错误； -3：等待任务队列满； -4：存在相同未执行任务； -5： 修改的任务 ID 不存在； -6： 任务已完成， 不可修改； -7： 其它错误

result 为 false 时调用出错， 如服务器未开启 TCP 服务或网络错误。

注： 当 loading_station 参数为"origin"时，task_id 传递叉车 ID(1~5)，可以添加使叉车自动回充电点停车位任务。 比如， task_id 传 1，服务器收到任务后会调度 1 号车回停车位。

2.3 监控任务状态：

每一个任务执行状态改变时会向 MES 报告任务状态， 需要发送如下数据订阅。

发送数据：

```
{
"op": "subscribe",
"topic": "/executing_task_list",
"type": "scheduling_msgs/TaskList2"
}
```

其中的 topic 参数为“executing_task_list”，向 MES 报告正在执行的任务状态；topic 参数为 “pending_task_list”，向 MES 报告等待执行的任务状态；

服务器收到如上数据后， 将不定时向 MES 发送如下数据：

```
{
    "topic": "/executing_task_list",
    "msg": {
        "status": [
            {
                "status": 9,
                "task_id": 0,
                "unloading_station": "冲压机 2_2",
                "text": "EXECUTING",
                "loading_station": "切割机 1_1",
                "agv_id": 0
            }
        ],
        "header": {
            "stamp": {
                "secs": 0,
                "nsecs": 0
            },
            "frame_id": "",
            "seq": 1792
        }
```

```
    },
    "op": "publish"
}
```

参数说明:

Topic: 订阅的主题名称， executing_task_list 是正在执行的任务列表，pending_task_list 是等待执行的任务列表。

Op: 操作类型。
Header: 时间戳。

Msg: 为返回的数据列表， 是一个数组， 为正在执行的所有任务状态， 栏位说明如下：

task_id: 任务 ID
agv_id: 执行此任务的小车 ID(1 到 5)， 为负值时为未分配叉车
loading_station: 叉货工位名称
unloading_station: 卸货工位名称， 为 unknown 时未指定
"status": 任务状态， 0: 等待执行； 1:等待前往叉货工位；2:正在前往叉货工位 3:开始叉货； 4: 叉货完成；5:等待前往卸货工位； 6:正在前往卸货工位； 7:开始卸货； 8: 卸货完成； 9:返回； 10:完成；20:任务取消； -1 任务出错；
"text": 为状态说明， 可选。

程序退出前 MES 发送如下数据， 退订：

```
{
    "op": "unsubscribe",
    "topic": "/executing_task_list"  或者 pending_task_list
}
```

2.4 监控叉车状态和位置：

调度服务器按一定频率（每秒 1 次）自动发送叉车运行信息和位置信息，向 MES 报告叉车状态， MES 需要发送如下数据订阅：（MES 仅需发送 1 次）

发送数据：

```
{
    "op": "subscribe",
    "topic": "/all_agvs_info",
    "type": "scheduling_msgs/AgvList"
}
```

叉车调度服务器将会发送如下数据：

```
{
```

```
"topic": "/all_agvs_info",
"msg": {
    "agvList": [
        {
            "agvName": "/AGV_0",
            "pose": {
                "position": {
                    "y": -1.8828154148007392,
                    "x": 9.291034280400009,
                    "z": 0
                },
                "orientation": {
                    "y": 0,
                    "x": 0,
                    "z": 0.7061322395224714,
                    "w": 0.708079981574807
                }
            },
            "isWorking": true,
            "agvID": 0,
            "working_station_name": "冲压机 2_2",
            "errorInfo": 0,
            "isAgvBoot": true
        },
        {
            "agvName": "/AGV_1",
            "pose": {
                "position": {
                    "y": -3,
                    "x": -2,
                    "z": 0
                },
                "orientation": {
                    "y": 0,
                    "x": 0,
                    "z": -0.7071067811865475,
                    "w": 0.7071067811865476
                }
            },
            "isWorking": false,
            "agvID": 1,
            "working_station_name": "robot_1",
            "errorInfo": 0,
            "isAgvBoot": true
```

```
            }
        ]
    },
    "op": "publish"
}
```

以上数据为返回 2 台叉车信息的例子， 字段说明如下：
agvList ： 返回的叉车信息数组， 应该有 5 个数据， 本处返回 2 个。

agvID ： 叉车 ID，取值 1-5， 没有 0 号叉车。
agvName：叉车名称，此名称是调度服务器所用， MES 显示信息时， 可根据 agvID 灵活使用， 比如显示给用户"1 号激光叉车"， 由 MES 自己定义。
isWorking： 工作状态： true 为工作中， false 为空闲。
isAgvBoot： 开机状态： true 为开机， false 为未开机， 未开机时位置信息未知， MES 可以显示为相应停车位。
errorInfo：错误信息， 暂未启用，扩展用。 初步定义如下：
0：无错误
1：遇到障碍物
2：电量低(叉车使用电量指示灯， 此值对叉车无效)
3：导航错误
4：其它错误
working_station_name：工作工位
pose： 位置姿态， 包含位置坐标和方向。
position： 位置坐标
x： x 坐标， 厂区东西方向为 x 轴，正东方为正方向
y： y 坐标， 厂区南北方向为 y 轴，正北方为正方向
z： 平面坐标， 无用
orientation：方向
四元数表示

注：此处单位为米，(0,0,0)坐标在叉车 5 号停车位正前方， 具体精确位置需现场沟通。

程序退出前 MES 发送如下数据， 退订：

```
{
    "op": "unsubscribe",
    "topic": "/all_agvs_info"
}
```

第三章 示例

如下为 linux TCP 调用示例：

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <arpa/inet.h>
#include <netinet/in.h>


const int port = 7070;
const char* ip = "192.168.0.100";
char buf_read[2048];

int main()
{
        //创建套接字,即创建 socket
        int clt_sock = socket(AF_INET, SOCK_STREAM, 0);
        if(clt_sock < 0)
        {
            //创建失败
            perror("socket");
            return 1;
        }

        //绑定信息，即命名 socket
        struct sockaddr_in addr;
        addr.sin_family = AF_INET;
```

```
        addr.sin_port = htons(port);
        //inet_addr 函数将用点分十进制字符串表示的 IPv4 地址转化为用网络
        //字节序整数表示的 IPv4 地址
        addr.sin_addr.s_addr = inet_addr(ip);

        //不需要监听

        //发起连接
//      struct sockaddr_in peer;
        socklen_t addr_len = sizeof(addr);
        int connect_fd = connect(clt_sock, (struct sockaddr*)&addr, addr_len);
        if(connect_fd < 0)
        {
            perror("connect");
            return 2;
        }

        //订阅服务器状态，只需调用一次，服务器每秒更新 1 次，程序退出前退订
        char buf_1[] = "{\"op\": \"subscribe\", \"topic\": \"/server_status\", \"type\":
\"scheduling_msgs/ServerStatus\"}";
        printf("client: %s\n", buf_1);
        write(clt_sock, buf_1, strlen(buf_1));
        usleep(10);

        //订阅 Task 状态，只需调用一次，服务器不定时更新，程序退出前退订
        char buf_2[] = "{\"op\": \"subscribe\", \"topic\": \"/task_status\", \"type\":
\"scheduling_msgs/TaskStatus\"}";
        printf("client: %s\n", buf_2);
        write(clt_sock, buf_2, strlen(buf_2));
        usleep(10);

        //发布一个新任务，client 根据需要发送任务指令，args 为要填入的参数
        char buf_3[] = "{\"op\": \"call_service\", \"service\": \"add_or_modify_forlift_task\",\
         \"args\": {\"task_id\": -1, \"loading_station\": \"bender\", \"unloading_station\":
\"welder\"}}";
        printf("client: %s\n", buf_3);
        write(clt_sock, buf_3, strlen(buf_3));
        usleep(10);

        while(1)
        {
            ssize_t size;
            size = read(clt_sock, buf_read, sizeof(buf_read));
            if(size > 0)
```

```
            {
                buf_read[size] = '\0';
            }
            else if(size == 0)
            {
                printf("read is done...\n");
                break;
            }
            else
            {
                perror("read");
                return 5;
            }

            printf("server: %s\n", buf_read);
            //处理叉车服务器返回数据
        }

        //退订服务器状态
        char buf_4[] = "{\"op\": \"unsubscribe\", \"topic\": \"/server_status\"}";
        printf("client: %s\n", buf_4);
        write(clt_sock, buf_4, strlen(buf_4));
        usleep(10);

        //退订 Task 状态
        char buf_5[] = "{\"op\": \"unsubscribe\", \"topic\": \"/task_status\"}";
        printf("client: %s\n", buf_5);
        write(clt_sock, buf_5, strlen(buf_5));
        usleep(10);

        close(clt_sock);
        return 0;
}
```

附录- 托盘工位名称

命名规则： **现场多台设备由东向西，依次编号 1,2,3,4**
缓存区由北往南， 依次编号 A, B

充电位_AGV_1: 坐标 X: 5.476 Y: -3.2
充电位_AGV_2: 坐标 X: 4.196 Y: -3.2
充电位_AGV_3: 坐标 X: 2.916 Y: -3.2
充电位_AGV_4: 坐标 X: 1.636 Y: -3.2
充电位_AGV_5: 坐标 X: 0.356 Y: -3.2
(注: 叉车充电位和停车位一致)

切管机
冲压机 1_1 (其中东边为 1 号冲压机，西边 2 号, 1_1 为 1 号冲压机 1 号托盘)
冲压机 1_2
冲压机 2_1
冲压机 2_2

中间缓存区_A_1 (缓存区 2 排，每排 8 个， 北边为 A, 南边为 B, 从东边依次为 1,2,3...)
.....
中间缓存区_A_8

中间缓存区_B _1
.....
中间缓存区_B _8

切割机 1_1
切割机 1_2
切割机 2_1

切割机 2_2

折弯机 1_1
折弯机 1_2
折弯机 2_1
折弯机 2_2
折弯机 3_1
折弯机 3_2
折弯机 4_1
折弯机 4_2

最终缓存区_A_1
……
最终缓存区_A_18

最终缓存区_B_1
……
最终缓存区_B_18

焊接线 1_A_1
……
焊接线 1_A_7

焊接线 1_B_1
……
焊接线 1_B_7

焊接线 1_A_1
……
焊接线 1_A_7

焊接线 1_B_1
……
焊接线 1_B_7

焊接线 2_A_1
……
焊接线 2_A_7

焊接线 2_B_1

.....

焊接线 2_B_7