目录

一、 机器人向调度服务器返送状态信息

1. 通讯端口

调度服务器接收机器人状态信息的端口号为 20202，连接方式为 UDP。这部分代码可以在 wpr_warehousing_monitor/src/wpr_server_node.cpp 中找到。

2. 消息结构

机器人返送给调度服务器的状态信息结构是以 Struct 的形式定义在头文件 Struct.h 中，这个文件存在于调度服务器和机器人程序的 include 文件夹中。后续开发中，若协议结构和字段定义发生改变，只需要将修改后的 Struct.h 同时更新到调度服务器和机器人程序中即可。机器人状态信息的消息结构定义如下：

```
typedef struct stRobotInfoMsg
{
    unsigned char header[2];
    unsigned char msg_type;
    unsigned char dev_type;
    unsigned char id;
    unsigned char battery;
    float map_x;
    float map_y;
    float map_yaw;
    int cmd_recv;
    int state;
    float joint_pos[10];
    int box_color;
}stRobotInfoMsg;
```

- header[2]
  消息的包头，这里定义为 0x55 0xAA 两个字节，用于解析时判断数据包起始位置。
- msg_tyep
  消息包的类型，目前类型定义如下：

MSG_T_ROBOT_STATE	机器人状态消息
MSG_T_PATH	路径消息
MSG_T_SERVER_CMD	服务器发给机器人的指令

  机器人返送给调度服务器的状态信息为 **MSG_T_ROBOT_STATE**。
- dev_type
  机器人设备类型，目前类型定义如下：

DEV_WPR_1	启明 1 服务机器人
DEV_WPB_HOME	启智 ROS 机器人
DEV_WPV_3	启程 3 移动机器人
DEV_EXT_ARM	固定机械臂

- id
  机器人的 ID 号，从数字 1 开始。
- battery
  机器人的电池电压，单位为“伏”。
- map_x、map_y、map_yaw
  机器人在地图上的（x，y）坐标，以及机器人的朝向角。
- cmd_recv
  机器人接收到的调度服务器指令，主要用于判断机器人是否成功收到服务器指令。
- state
  机器人的状态，目前已经定义的状态如下：

RBT_ST_STOP	闲置待命状态
RBT_ST_GOTO	导航移动状态
RBT_ST_ARRIVED	到达导航目标
RBT_ST_DOCK_FACETO	对准充电坞
RBT_ST_DOCK_DONE	进入完成
RBT_ST_DOCK_LEAVE	离开充电坞
RBT_ST_LEAVE_DONE	离开完成
RBT_ST_BOX_DETECT	检测物料盒子
RBT_ST_BOX_F_MOVE	靠近物料盒子
RBT_ST_BOX_F_ROT	对准物料盒子
RBT_ST_BOX_GRAB	抓取物料盒子
RBT_ST_BOX_DONE	拿到物料盒子
RBT_ST_MOBILE_DETECT	检测移动货架
RBT_ST_MOBILE_F_MOVE	靠近移动货架
RBT_ST_MOBILE_F_ROT	对准移动货架
RBT_ST_MOBILE_PLACE	放置料盒到移动货架
RBT_ST_MOBILE_DONE	放置完毕
RBT_ST_PALLET_DETECT	检测固定货架
RBT_ST_PALLET_F_MOVE	靠近固定货架
RBT_ST_PALLET_F_ROT	对准固定货架
RBT_ST_PALLET_MEASURE	定位放置空位
RBT_ST_PALLET_PLACE	放置料盒到固定货架
RBT_ST_PALLET_DONE	放置完毕
RBT_ST_CONVEY_BOX	传送带传送料盒（预留给下料口用的）
RBT_ST_EXIT_BOX	料盒到达转移口（预留给下料口用的）
RBT_ST_GM_DETECT	检测移动货架
RBT_ST_GM_F_MOVE	靠近移动货架
RBT_ST_GM_F_ROT	对准移动货架
RBT_ST_GM_OBJECT	搜索移动货架上的物品
RBT_ST_GM_GRAB	抓取移动货架上的料盒
RBT_ST_GM_DONE	抓取完毕

- joint_pos[10]
  机器人的关节数值，主要用于同步调度服务器 Rviz 里的模型姿态显示。

- box_color;
  机器人检测到的料盒颜色，用于控制调度策略的分支选择。

二、 机器人向调度服务器返送导航路径

1. 通讯端口

机器人向调度服务器发送导航路径消息的端口号为 20203，连接方式为 UDP。这部分代码可以在 wpr_warehousing_monitor/src/wpr_server_node.cpp 中找到。

2. 消息结构

机器人向调度服务器发送导航路径消息结构是以 Struct 的形式定义在头文件 Struct.h 中，这个文件存在于调度服务器和机器人程序的 include 文件夹中。后续开发中，若协议结构和字段定义发生改变，只需要将修改后的 Struct.h 同时更新到调度服务器和机器人程序中即可。机器人向调度服务器发送导航路径消息结构定义如下：

```
typedef struct stPathMsg
{
    unsigned char header[2];
    unsigned char msg_type;
    unsigned char dev_type;
    unsigned char id;
    unsigned char len;
    float path_x[100];
    float path_y[100];
}stPathMsg;
```

- header[2]
  消息的包头，这里定义为 0x55 0xAA 两个字节，用于解析时判断数据包起始位置。
- msg_tyep
  消息包的类型，定义见机器人状态信息部分。
- dev_type
  机器人设备类型，定义见机器人状态信息部分。
- id
  机器人的 ID 号，从数字 1 开始。
- len
  路径点的数量。
- path_x[100]
  路径点的 x 坐标值数组。
- path_y[100]
  路径点的 y 坐标值数组。

三、 调度服务器向机器人发送控制指令

1. 通讯端口

调度服务器向机器人发送控制指令信息的端口号为 20201，连接方式为 UDP。这部分代码可以在 wpr_warehousing_monitor/src/server/DataCenter.cpp 中找到。

2. 消息结构

调度服务器向机器人发送控制指令信息结构是以 Struct 的形式定义在头文件 Struct.h 中，这个文件存在于调度服务器和机器人程序的 include 文件夹中。后续开发中，若协议结构和字段定义发生改变，只需要将修改后的 Struct.h 同时更新到调度服务器和机器人程序中即可。机器人状态信息的消息结构定义如下：

```
typedef struct stCommandMsg
{
    unsigned char header[2];
    unsigned char msg_type;
    unsigned char id;
    int command;
    float map_x;
    float map_y;
    float map_yaw;
    float data[10];
}stCommandMsg;
```

- header[2]
  消息的包头，这里定义为 0x55 0xAA 两个字节，用于解析时判断数据包起始位置。
- msg_tyep
  消息包的类型，定义见机器人状态信息部分。
- dev_type
  机器人设备类型，定义见机器人状态信息部分。
- id
  机器人的 ID 号，让机器人确认此消息是给自己的。
- command
  控制指令，定义如下：

CMD_STOP	停止移动
CMD_ROBOT_GOTO	移动到指定坐标
CMD_GRAB_BOX	抓取物料盒子
CMD_PLACE_MOBILE	将物料盒子放置到启程 3 移动货架上
CMD_PLACE_PALLET	将物料盒子放置到固定货架上
CMD_DOCKING	进入充电坞
CMD_CHARGING	充电
CMD_LEAVE_DOCK	离开充电坞

CMD_CONVEY_BOX	下料口传送料盒（预留给下料口使用）
CMD_GRAB_MOBILE	抓取移动货架的物品

- map_x、map_y、map_yaw
  部分指令会携带（x，y）坐标，以及朝向角。
- data[10]
  预留的用于扩展功能的数组字段，目前没有使用。

四、 为调度系统扩展新的设备的步骤

1. 确定通讯端口。如果消息结构可以复用上述结构体，建议端口也复用，这样解析类可以使用同一个。
2. 定义好设备的有限状态机，将状态定义添加到结构体文件 Struct.h 中，并将该文件拷贝覆盖到 wpr_warehousing_monitor 中。
3. 为新的设备添加新的设备类型和 ID 号，添加到结构体文件 Struct.h 中，并将该文件拷贝覆盖到 wpr_warehousing_monitor 中。
4. 在设备端编写发送代码，将 stRobotInfoMsg 结构的数据发送给调度服务器。
5. 在 wpr_warehousing_monitor 的 src\server\DataCenter.cpp 里，找到解析函数 void CDataCenter::RecvNewPackage(stRobotInfoMsg* inInfo)。使用 ROS_INFO 或者 ROS_WARN 将参数 inInfo 中的关键字段数据显示在终端程序里，观察数值是否与设备端发送的一致。
6. 为 wpr_warehousing_monitor 添加 Rviz 三维模型的显示（这部分内容比较多，可以双方配合完成）。
7. 为设备定义新的控制指令（CMD_xxxxx），将新的控制指令添加到结构体文件 Struct.h 中，并将该文件拷贝覆盖到 wpr_warehousing_monitor 中。
8. 在调度服务程序 wpr_warehousing_monitor\src\server\MissionManager.cpp 中添加新的指令发送对象和机器人状态结构体。（这部分内容比较多，可以双方配合完成）。
9. 修改调度服务程序 wpr_warehousing_monitor\src\test_command.cpp 节点的代码，让调度系统向新设备发送新定义的指令。
10. 从调度服务器和新设备的实际运行状态来判断新添加的状态信息和控制指令是否奏效。