Application Environment

rabbitmq network communication based on local network

Protocol and Connection mode

Protocol

API contents are using protobuf format,there are two types of API feedback data:

> 1. Send order to visit: all using protobuf format
> 2. Self Responce : all using json format, there is no sequency between same level fields, please using stardard json format to parse.

Connection mode

Connection mode: TCP (rabbitmq local server)

Network setting: TCP client

Server IP Address:

> 192.168.1.108 (This is the static IP address of the host computer,TCP client PC need to connnect to the robot router to build the local network connection with the chassis computer)

Server port:5672

API User Guide

: General interpretation

- General parameter "uuid"

  > If there is no special instruction, in belows commands, uuid must be selected as mandatory parameter, server need to return it as original after carry out the command.

- Parameter Instruction

attribute	Type	Description
uuid	string	task ID，the unique identifier
type	string, include 'action', 'order', 'info'	Task type, movement--"action", command --"order",information--"info"
exec_type	'timing', 'immediate_exec'	type of task executed, scheduled--"timing", temporary--"immediate_exec"
level	CommandLevel	task level
source	string, include 'server', 'app', 'ui'	Task sender
command	string	Command of execute tasks. Details refer to the API of each function
args	json string	Command parameter of execute tasks. Details refer to the API of each function
mate_tasks	list of Task	accompany with task list, not in use.
wait_mate_finish	bool	accompany with task monitor, not in use.
enable_recovery	bool	hang up type of task, not in use.
require_return	bool	Return the status of task, open in default.
immediate_response	bool	Feedback of receiveing taks, not in use.
loop	int	circle of task, default is "1"
start_time	string	Start time of task, empty as default, start immediately.
end_time	string	End time of task, empty as default.

> Ps: loop parameters only apply to multiple markerpoints movement task.

- "level" value setting instruction in parameter

attribute	value	Description
NORMAL	0	Normal task, apply to scheduled movement task
INTERUPPTED	2	Task been interruppted
IMPORTANT	4	Critical task,apply to temporary movement task

> ps: level only apply to movement task，higher level task will interupt lower level task.

Example (for python as example):

1. Clents sent protobuf message:

```
 robot_message = robotTask_pb2.RequestMessage()
robot_task = robot_message.robotTask.add()
robot_task.uuid = 'ROBOT01_UUID0001'
robot_task.type = 0
robot_task.level = 3
robot_task.exec_type = 2
robot_task.source = 0
robot_task.action = 2
robot_task.args = '{"target_marker": "ChargeStation"}'
robot_task.require_return = True
robot_task.loop = 1
robot_task.start_time = ""
robot_task.end_time = ""
```

2. return:

```
{
    uuid: "ROBOT01_UUID0001",
    status: 1,
    topic: TASK,
    target_marker: "ChargeStation",
    msg: "{"code": 200}" # json String
}
```

1.Robot movement function

1.1 Single markerpoint movement task

topic: TASK_PUBLISHER_TOPIC

Parameter instruction:

Name	Insruction	Mandatotry	remark
type	Task type	Yes	"action"
level	Task priority	Yes	"IMPORTANT"
exec_type	Type of task execute	Yes	"immediate_exec"
source	Task creator	Yes	"server"
command	Task execute API	Yes	"move_single"
args	command parameter	Yes	{"target_marker": Markerpoint name(string)}

> ps: target_marker does not support in multiple maps.

API description

1. Make robot move from current position to a preset target point in the map.
2. There is necessary map creation job required before using this API
3. During robot autonomous navigation and avoiding obstacle, it will plan the path and adjust the speed automately,no action required by operator,operator could wait for the task status feedback to know the status of task execute status.
4. Not supoort

Example

1. Send(Example in python, protobuf pack the task message)

```
 robot_message = robotTask_pb2.RequestMessage()
robot_task = robot_message.robotTask.add()
robot_task.uuid = 'ROBOT01_UUID0001'
robot_task.type = 0
robot_task.level = 3
robot_task.exec_type = 2
robot_task.source = 0
robot_task.action = 2
robot_task.args = '{"target_marker": "ChargeStation"}'
robot_task.require_return = True
```

2. Return(protobuf format)

```
{
    uuid: "ROBOT01_UUID0001",
    status: 1,
    topic: TASK,
    target_marker: "ChargeStation",
    msg: "{"code": 200}" # json String
}
```

Below information is for robot movement function feedback status search.

1. status: Represent the robot status of move_target task, the value and instruction is as below:

Value	Status	Description
1	running	Represents the robot is moving to the move_target, other task with same and lower priority level will be queue up
2	succeeded	Represents the movement task is completed
3	failed	Represents the movement task is failed
4	canceled	Represents the movement task has been canceled

2. code:Description of task execution

Value	Description
100	[Task Execution Start]
200	[Task Executed Successfully]
300	[Task Canceled]
301	[The E_stop is Activated, Task Canceled]
302	[A High-level Task is Received, The Current Task is Interrupted]
400	[Task Execution Failed]
402	[The Soft E_stop is Activated, Task Receiving Denied]
406	[The Hard E_stop is Activated, Task Execution Failed]
408	[The Soft E_stop is Activated, Task Execution Failed]
500	[The Task Failed to Send, Wrong Format]

2. Cancellation of movement function

topic: TASK_PUBLISHER_TOPIC

Parameter description:

Name	Description	Mandatotry	Remark
uuid	Task ID	Yes	Format requirement:Add "_ " after the uuid of cancelled task.
type	Task type	Yes	"order"
level	Task priority	Yes	"IMPORTANT"
exec_type	Task execution type	Yes	"immediate_exec"
source	Task creator	Yes	"server"
command	Task execution API	Yes	"cancel_task"
args	command parameter	Yes	{"cancel_uuid": uuid of cancelled task}

> ps: 1. target_marker does not suppport in different maps. 2. uuid format requirement：Add "_ " after the uuid of cancelled task, example:
>
> > 1. if the uuid of cancelled task is：ROBOT01_UUID0001
> > 2. canceled movement function uuid will be: ROBOT01_UUID0001_

API description

1. Use this to make robot terminate the current executing movement task, robot will switch to standby mode after successful cancelation.
2. During the robot is excuting API 1 (robot movement command),if need to terminate the movement status of robot,this API could be used. Robot will stay at current position until received next movement task.

Example

1. Send(Example in python, protobuf pack task messages)

```
 robot_message = robotTask_pb2.RequestMessage()
robot_task = robot_message.robotTask.add()
robot_task.uuid = 'ROBOT01_UUID0001_'
robot_task.type = 1
robot_task.level = 3
robot_task.exec_type = 2
robot_task.source = 0
robot_task.order = 0
robot_task.args = '{"cancel_uuid": "ROBOT01_UUID0001"}'
robot_task.require_return = True
```

2. Return(protobuf format)

```
 {
    uuid: "ROBOT01_UUID0001_",
    status: 1,
    topic: TASK,
    target_marker: "",
    msg: "{"code": 200}" # json string
}
```

3. Robot Update

topic: STATUS_TOPIC

Parameter

N/A

API description

1. Robot global status update

1. Every 2s, robot will send update of global status to all the connected client in json format.
2. Form this API, other information can be monitored like "Changing Status","Emergency_Stop Status","Battery percentage","current position in the map" and so on.
3. Can used to monitor the task status in live,to make logical judegement in the control flow.

2. markerpoint list update

1. Send the update every 10s, or send update immediately if there is any change on the makerpoint
2. Send details of all the preset markerpoint.

3. Current map information update

1. Send update every 10s
2. Includes the height & width information of the map

Example

Return after successful connection: - 1.Robot global status

```
{
    "type": "response",
    "command": "/api/robot_status",
    "uuid": "",
    "status": "OK",
    "error_message": "",
    "time_stamp": int(Time stamp), # timestamp
    "results": {
        "move_target": "",  # name of target point in movement task
        "move_status": "",  # Status of the executing movement task
        "move_retry_times": 0,  # Every increase of the time of retry means the robot has retried one more time on the plath navigat
        "linear_velocity": 0.0,  # linear velocity
        "steering_angle": 0.0,  # steering angle velocity
        "charge_state": "",  # Charging status
        "hard_estop_state": True,  # Hard E stop status, True->Hard E stop is activated, False-> Hard E stop is not activated.
        "soft_estop_state": True,  # Soft E stop, True-> Soft E stop is activated, False-> Soft E stop is not activated.
        "estop_state": True,  # hard_estop_state || soft_estop_state, True-> E stop activated, False-> E stop not activated.
        "hard_estop": True,
        "control_mode": True,  # In remote controller mode
        "park_mode": True,  # In parking mode
        "control_state": "",  # Control state, auto--auto navigation mode, remote--mannual control mode, control--Remote controller m
        "salt": "",  # Remote control client
        "power_percent": 0.0,  # batterry level in %
        "current_pose": {
            "x": 0.0,
            "y": 0.0,
            "theta": 0.0,
        },
        "current_floor": 0, # Current floor,no in use.
        "error_code": 0  # error code in hexadecimal, representing by 8 digit in total, if not showing 0, stands for machine are in
    }
}
```

Below information is for robot movement function feedback status search. - move_target represents the target markpoint of the movement task. - move_status represents the current task status of robot executing task, the value setting is as below:

Value	Description
idle	Represents the robot has never received any movement command since start up the service.
running	Represents the robot is moving to the move_targer, any new move command will be declined.
succeeded	Represents the movement task is successful.
failed	Represents the movement task is failed.
canceled	Represents the movement task has been cancelled.

- 2.marker pint list update

```
{
    "type": "response",
    "command": "/api/markers/query_list",
    "uuid": "",
    "status": "OK",
    "error_message": "",
    "results": {
        "marker1": {
            "floor": 0,
            "pose": {
                "orientation": {
                    "w": -0.91123658553545,
                    "x": 0,
                    "y": 0,
                    "z": -0.45516000000001
                },
                "position": {
                    "x": -6.37999992370605,
                    "y": 21.5900001333581,
                    "z": 0
                }
            },
            "marker_name": "marker1",
            "key": 0
        },
        ... // other marker point information
    }
}
```

- 3.Current Map information

```
{
    "type": "response",
    "command": "/api/map/get_current_map",
    "uuid": "",
    "status": "OK",
    "error_message": "",
    "results":{
        "map_name": "map_name_1",  # Map name
        "floor": "0"
        "info": {
            "resolution": 0.1,  # Resolution(meter/pixel)
            "width": 764,  # width
            "height": 861,  # height
            "origin_x": -37.3,  # Origin coordinate x (Left bottom corner)
            "origin_y": -40.000001   # Origin coodrinate y (Left bottom corner)
        }
    },
}
```

3.Marker Point function API

3.1 Create marker point at specific coordinates.

topic: TASK_PUBLISHER_TOPIC

Parameter instruction:

Name	Description	Mandatotry	Remark
type	Task type	Yes	"order"
level	Task priority	Yes	"IMPORTANT"

Name	Description	Mandatotry	Remark
exec_type	Task execution type	Yes	"immediate_exec"
source	Task creator	Yes	"server"
command	Task execution API	Yes	"add_marker"
args	command parameter	Yes	{"marker_name": name(string), "x": 0.0, "y": 0.0, "theta": 0.0}

API description

1. Add marker point at specific position

Example

1. Send(Example in python,protobuf pack the task message)

```
 robot_message = robotTask_pb2.RequestMessage()
robot_task = robot_message.robotTask.add()
robot_task.uuid = 'ROBOT01_UUID0001'
robot_task.type = 1
robot_task.level = 3
robot_task.exec_type = 2
robot_task.source = 0
robot_task.order = 8
robot_task.args = '{"marker_name": "ChargeStation", "x": 0.0, "y": 0.0, "theta": 0.0}'
robot_task.require_return = True
```

2. Return(protobuf format)

```
{
    uuid: "ROBOT01_UUID0001",
    status: 1,
    topic: TASK,
    target_marker: "",
    msg: "{"code": 200, "command": "add_marker"}" # json String
}
```

3.2 Delete marker point

topic: TASK_PUBLISHER_TOPIC

Parameter instruction:

Name	Description	Mandatotry	Remark
type	Task type	Yes	"order"
level	Task priority	Yes	"IMPORTANT"
exec_type	Task execution type	Yes	"immediate_exec"
source	Task creator	Yes	"server"
command	Task execution API	Yes	"delete_marker"
args	command parameter	Yes	{"marker_name": name(string),}

API Description

1. Delete the preset marker point, if the point does not exit, then returen failed.

Example

1. Send (Example in python, protobuf pack the task message)

```
 robot_message = robotTask_pb2.RequestMessage()
robot_task = robot_message.robotTask.add()
robot_task.uuid = 'ROBOT01_UUID0001'
robot_task.type = 1
robot_task.level = 3
robot_task.exec_type = 2
robot_task.source = 0
robot_task.order = 9
robot_task.args = '{"marker_name": "ChargeStation"}'
robot_task.require_return = True
```

2. return(protobuf format)

```
{
    uuid: "ROBOT01_UUID0001",
    status: 1,
    topic: TASK,
    target_marker: "",
    msg: "{"code": 200, "command": "delete_marker"}" # json string
}
```

4. Video streaming

rtsp address

rtsp://admin:gzwx2021@192.168.10.64:554/Streaming/Channels/2

APT description

Acquire the video stream from the IP camera on the robot.