Driver开发流程

本工程中仅添加几个常用的SDK，ROS的封装，后续可以根据用户需要添加自定义的接口，本指南旨在描述将SDK对ROS接口方法。

可以将封装过程分为以下几个部分：

1. 根据SDK和用户需求在elite_msgs下建立消息文件（服务或话题）。
2. 在elite_driver中添加对应接口的文件，建立服务。
3. 编写单元测试完成接口测试（可使用FakeEc）。

1.消息文件建立方法

首先根据数据类型确定使用话题还是服务。

一般数据发布采用话题方式，而控制采用服务方式实现。

关于ROS如何自定义接口方式请参考教程：自定义消息

示例1：IO控制

```
uint32 address
float32 value
---
bool result
```

示例2：JointMove

```
float32[] target_joint #(list): 目标关节数据,为8个,6个会报错
float32 speed # (float): 关节速度百分比
float32 acc #(int, optional): 加速度,不写默认为0. Defaults to 0.
float32 dec #(int, optional): 减速度,不写默认为0. Defaults to 0.
bool is_blocking # 是否阻塞运行，默认阻塞
---
bool result
```

在`elite_robot/src/elite_msgs/srv`建立文件，接着
在`/root/elite/elite_robot/src/elite_msgs/CMakeLists.txt`中添加对应的文件即可。

2.驱动层编写

代码编写

在`elite_robot/src/elite_driver/src/elite_driver`中建立对应的文件，比如`elite_robot/src/elite_driver/src/elite_driver/set_analog_io.py`,接着编写文件内容

- 导入定义好的接口
- 声明服务与回调函数
- 在回调函数里调用SDK完成操作
- 将操作结果组装成Response返回给客户端

具体代码实现如下

```python
import rospy
from elite_msgs.srv import SetIO, SetIOResponse


class SetAnalogIOService():
    """设置模拟输出服务"""

    def __init__(self) -> None:
        # print("SetAnalogIO service is started...")
        self.set_io_server = rospy.Service(
            "set_analog_io", SetIO, self.handle_set_analog_io_)
        self.res = SetIOResponse()

    def handle_set_analog_io_(self, req):
        """处理设置模拟输出"""
        result = self.elite_robot.set_analog_io(  # pylint: disable=E1101
            req.address, req.value)
        self.res.result = result
        print(f"result:{result}")
        return self.res
```

3.测试编写

接着我们可以编写一个测试用pytest测试该服务是否可以正常的调用和return

新建`/root/elite/elite_robot/src/elite_driver/tests/test_set_digital_io.py`

编写代码如下

需要注意的时FakeEc目前并没有所有的EC对应接口实现，若需要测试相应接口需要手动添加FakeEc代码。

```python
import pytest
import rospy
from elite_msgs.srv import SetIO, SetIORequest
from elite_driver.set_analog_io import SetAnalogIOService
from elite_driver.fake_ec import FakeEc


@pytest.fixture
def ec_fake() -> FakeEc:
    """
    生产虚拟机械臂，用于测试使用
    """
    fake_ec_ = FakeEc("123")
    return fake_ec_


def test_set_analog_io_server(ec_fake):  # pylint: disable=W0621
    """测试设置模拟IO输出"""
    rospy.init_node("test_set_analog_io_server")
    set_analog_io_server = SetAnalogIOService()
    set_analog_io_server.elite_robot = ec_fake
    rospy.wait_for_service("set_analog_io_server")
    set_analog_io_client = rospy.ServiceProxy("set_analog_io_server", SetIO)
    req = SetIORequest()
```

```
    req.address = 10
    req.value = 1
    res = set_analog_io_client(req)
    assert res.result is True
    print("res", res)


if __name__ == "__main__":
    test_set_analog_io_server(FakeEc("123"))
```

接着我们可以使用python进行测试

```
pytest test_set_digital_io.py
```

以上即ROS接口封装过程。