一、介绍

这是一个双足机器人控制和通信的软件接口包。 通过这个接口包，可以方便地与双足机器人底层硬件通信，完成状态信息获取和控制电机动作。

二、安装依赖

1. 首先需要安装ROS。

- 这里建议使用fishros的一键安装，安装桌面版。

```
wget http://fishros.com/install -O fishros && . fishros
```

2. 安装串口通信的相关包。

```
sudo apt-get install libserialport0 libserialport-dev
```

3. 安装python依赖

```
python3 -m pip install empy
```

三、编译

1. 克隆代码到本地

```
git clone https://github.com/HighTorque-Robotics/livelybot_robot.git
```

2. 编译

```
cd livelybot_robot
catkin_make
```

四、测试

1. IMU设备测试

    - 给测试脚本增加执行权限：`chmod +x test_yesense_imu.sh`;
    - 执行测试脚本：`./test_yesense_imu.sh`;
    - 在启动的RVIZ界面，观察IMU的姿态。

2. 电机控制测试1

- 准备主控板和电机若干；
- 根据配置文件`lively_description/robot_param/1dof_STM32H730_model_test_Orin_params.yaml`中的配置信息，在can0上连接1个电机，电机ID为`1`。
- 修改配置文件`livelybot_description.launch`中的参数，将`dof_type`设置为`1`；
- 添加执行权限：`chmod -R 777 test_motor_run.sh`;
- 执行电机测试程序`./test_motor_run.sh`。

3. 电机控制测试2

- 准备主控板和电机若干；
- 根据配置文件`lively_description/robot_param/6dof_STM32H730_model_test_Orin_params.yaml`中的配置信息，在can0上连接3个电机，电机ID分别为`1,2,3`，在can1总线上连接3个电机，电机ID分别为`1,2,3`。
- 修改配置文件`livelybot_description.launch`中的参数，将`dof_type`设置为`6`；
- 执行电机测试程序`./test_motor_run.sh`。

4. 电机数据反馈测试

- 准备主控板和电机若干；
- 根据配置文件`lively_description/robot_param/6dof_STM32H730_model_test_Orin_params.yaml`中的配置信息，在can0上连接3个电机，电机ID分别为`1,2,3`，在can1总线上连接3个电机，电机ID分别为`1,2,3`。
- 修改配置文件`livelybot_description.launch`中的参数，将`dof_type`设置为`6`；
- 执行电机测试程序`./test_motor_feedback.sh`。

五、IMU信息读取

1. 在`livelybot_robot`目录下，执行命令`source devel/setup.bash`;
2. 查看imu设备在系统中的设备名称：`ls /dev`，一般为`ttyACM*`或者`ttyUSB*`;
3. 添加设备权限：`sudo chmod -R 777 /dev/ttyACM*`或`sudo chmod -R 777 /dev/ttyACM*`；
4. 启动IMU设备节点：`roslaunch yesense_imu run_without_rviz.launch`；
5. 查看IMU节点发布的话题消息列表：`rostopic list`；
6. 查看IMU姿态数据：`rostopic echo /yesense/sensor_data`；
7. 如果需要图形化界面，运行：`roslaunch yesense_imu yesense_rviz.launch`，可以通过RVIZ界面观察IMU的姿态。

六、电机控制

1. 连接主控板，查看主控板在系统的设备名称：`ls /dev`，一般为`ttyACM*`或者`ttyUSB*`；
2. 添加设备权限：`sudo chmod -R 777 /dev/ttyACM*`或`sudo chmod -R 777 /dev/ttyACM*`；
3. 编写配置文件，命名和内容参考：`lively_description/robot_param/6dof_STM32H730_model_test_Orin_params.yaml`；
4. 创建机器人对象：`livelybot_serial::robot rb`;
5. `rb`的成员变量`Motors`为存放电机对象的容器，包含配置文件描述的所有电机的对象，电机在容器中的位置顺序为`canport0`的一号、二号、三号电机，依次类推，然后是`canport1`的一号、二号、三号电机，依次类推，依次是`canport3`、`canport4`（如果有点话），依次类推；

6. 获取电机对象:

```
auto it = rb.Motors.begin();
motor *m = &(*it+N);
```

其中，N是电机在容器中的位置，从0开始；

7. 执行控制指令（保存指令到缓存区）：
   - `m->fresh_cmd_int16(pos, vel, tor, kp, ki, kd, acc, voltage, current)`；// 调试、通用模式（需要在配置文件中配置工作模式, 根据工作模式填入对应的参数，其他参数无效）
   - `m->position(pos)`；// 位置模式
   - `m->velocity(vel)`；// 速度模式
   - `m->torque(tor)`；// 力矩模式
   - `m->voltage(volt)`；// 电压模式
   - `m->pos_vel_MAXtqe(pos, vel, tor_upper)`；// 位置、速度模式，带力矩上限
   - `m->pos_vel_tqe_kp_kd(pos, vel, tor, kp, kd)`；// PD位置速度+前馈扭矩模式
   - `m->pos_vel_kp_kd(pos, vel, kp, kd)`；// PD位置速度模式
8. 发送命令（发送到电机控制板）：`rb.motor_send_2()`。

七、电机状态获取

1. 参考上一章电机控制的1、2、3、4、5步骤；
2. 获取电机对象:

```
auto it = rb.Motors.begin();
motor *m = &(*it+N);
```

其中，N是电机在容器中的位置，从0开始；

3. 获取电机状态：`m->get_current_motor_state()`, 返回值类型为`motor_back_t *`，结构体变量如下：

```
typedef struct motor_back_struct
{
    uint8_t ID;     // 电机ID
    float position; // 位置
    float velocity; // 速度
    float torque;   // 力矩
} motor_back_t;
```

八、使用例程

1. 电机控制示例代码：

```
./livelybot_serial/test/test_motor_run.cpp
```

2. 电机数据反馈示例代码：

```
./livelybot_serial/test/test_motor_feedback.cpp
```