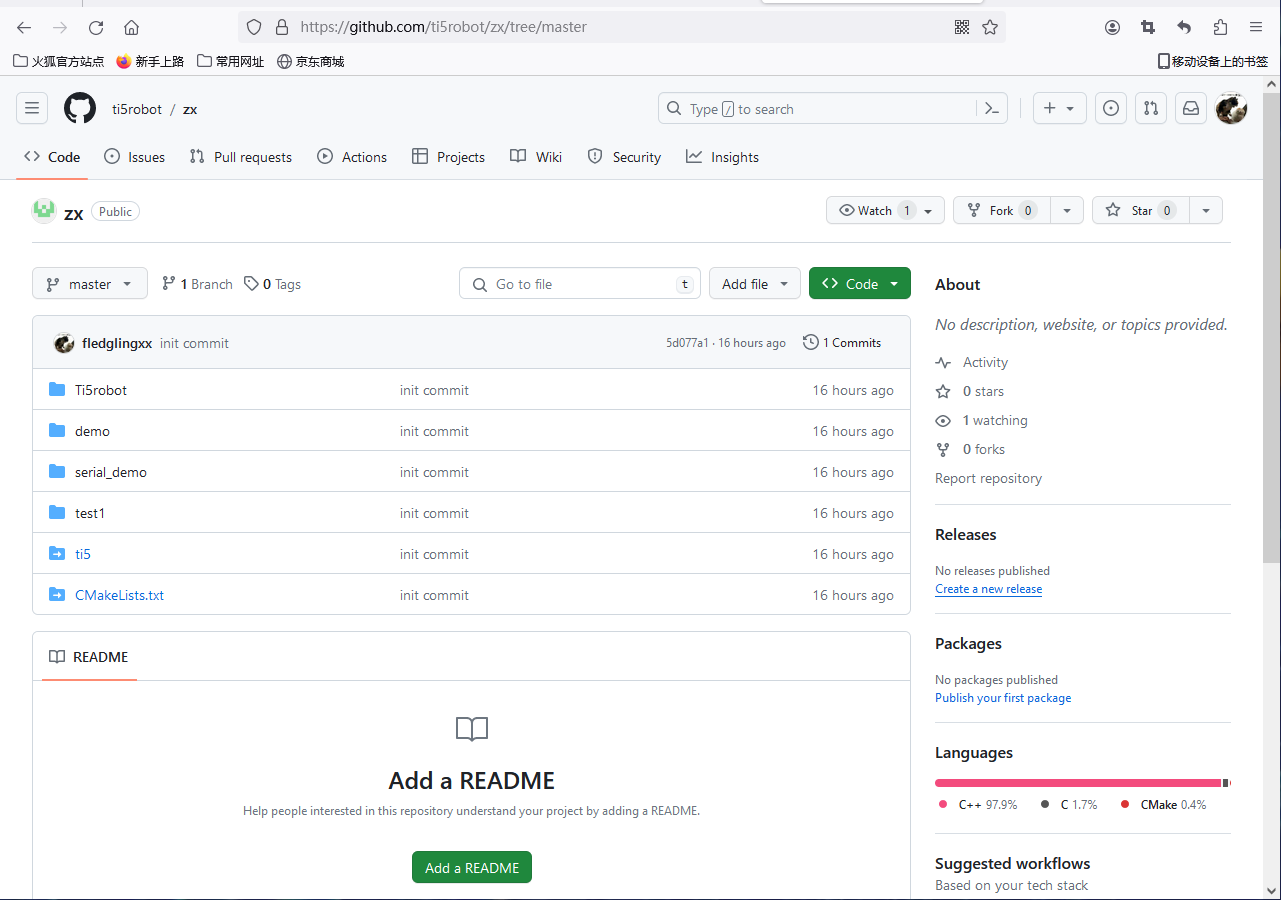
ROS接口使用说明

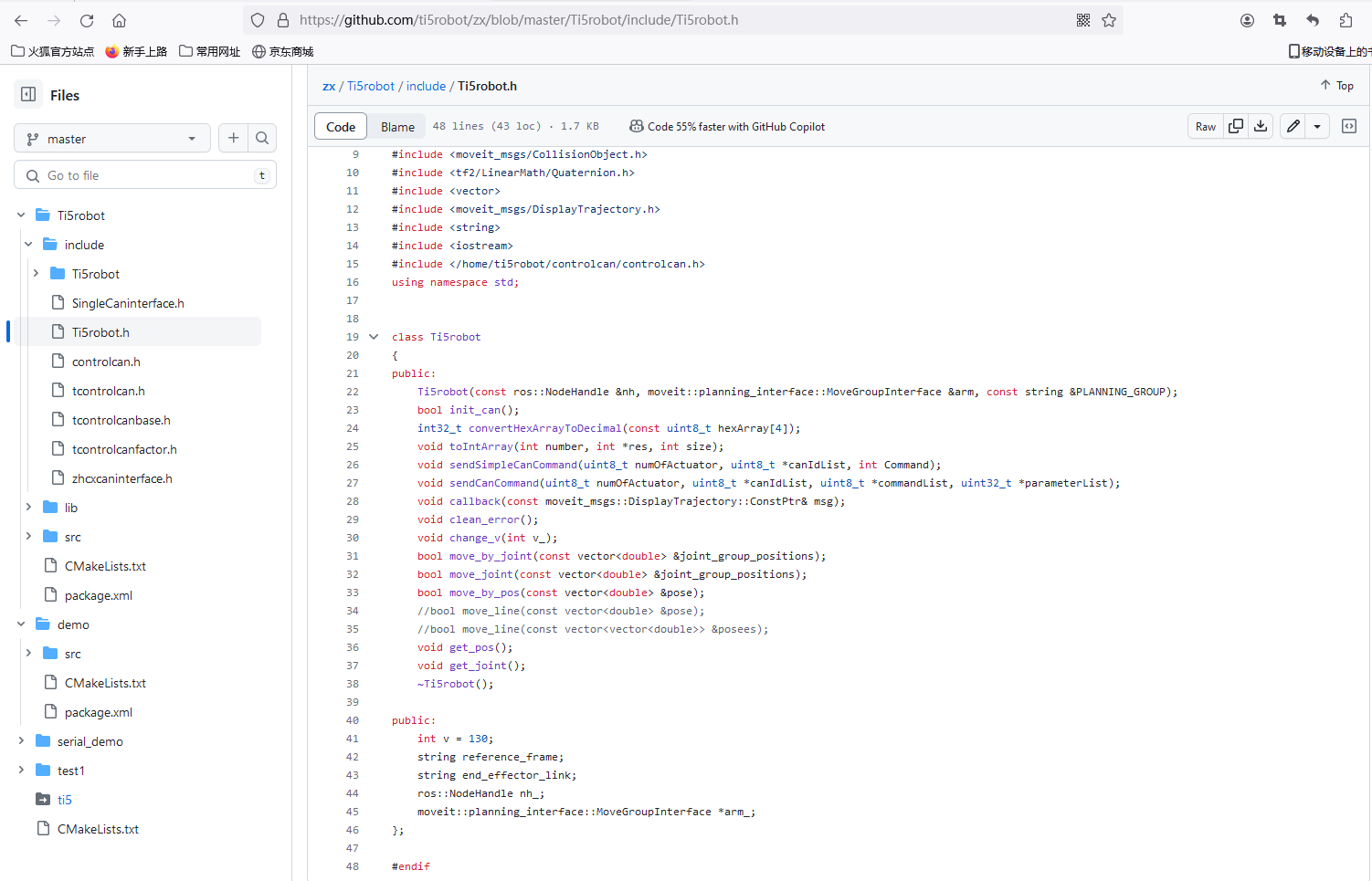
代码github地址： <https://github.com/ti5robot/zx/tree/master>



关于Ti5robot功能包介绍

Ti5robot功能包中主要实现了Ti5robot类，功能包中集成了函数具体实现的Ti5robot.cpp和相应头文件Ti5robot.h以及相应的串口库的头文件。

Ti5robot.h



相应函数功能和参数说明

Ti5robot(const ros::NodeHandle &nh, moveit::planning\_interface::MoveGroupInterface &arm, const string &PLANNING\_GROUP);

该函数为TI5robot类的构造函数，ros::NodeHandle &nh 参数是一个ROS节点句柄的应用。ROS中的节点句柄是用于与ROS系统进行通信的主要方式之一，允许节点发布和订阅主题、

调用服务等。moveit::planning\_interface::MoveGroupInterface &arm 为一个moveit库中的MoveGroupInterface对象的引用，moveit是用于机器人运动规划和控制的库，MoveGroupInterface用来管理机器人的运动规划和执行。const string &PLANNING\_GROUP用于指定机器人规划组的名称，在moveit中，机器人的运动规划是根据器所属的规划组进行的。

bool move\_by\_joint(const vector<double> &joint\_group\_positions);

该函数实现了机械臂的关节运动，const vector<double> &joint\_group\_positions为一个类型为vector的变长数组，里面定义的数值为机械臂每个轴需要达到的角度数值，机械臂关节角度的范围为-3.14到3.14。

bool move\_joint(const vector<double> &joint\_group\_positions);

该函数实现了机械臂的关节运动，const vector<double> &joint\_group\_positions为一个类型为vector的变长数组。该函数与move\_by\_joint(const vector<double> &joint\_group\_positions)的区别为：move\_by\_joint函数实现了机械臂各关节运动到的位置为joint\_group\_positions中数据的位置，move\_joint实现了机械臂各关节运动到的位置为当前的角度加上传入的joint\_group\_positions中数据的位置。

例如当前机械臂六关节的位置分别为0.2、0.3、0.4、0.5、0.6、0.7,joint\_group\_positions中的数据为1、1、1、1、1、1，如果运行move\_by\_joint函数，则机械臂各关节角度为1、1、1、1、1、1，如果运行move\_joint函数，则机械臂各关节角度为1.2、1.3、1.4、1.5、1.6、1.7.

bool move\_by\_pos(const vector<double> &pose);

该函数实现了机械臂的位姿运动，const vector<double> &pose 为一个类型为vector的变长数组，里面定义了机械臂末端位姿的x、y、z、roll、pitch、yaw的具体数值。x、y、z分别是绩溪北末端在基坐标系下的位置坐标，描述了机械臂末端在三个空间方向上的位置，roll、pitch、yaw分别代表绕x、y、z轴的旋转角度，描述了机械臂末端的姿态或方向。

在某些位置，由于机械臂结构的限制或特定的姿态导致给出x、y、z、roll、pitch、yaw并不能解算出对应的机械臂关节角度，如果不能解算出相应的关节角度，该函数则会给出move\_p:FAILED的提示。

void get\_joint();

该函数实现了得到当前机械臂各关节的角度并显示在终端窗口中。

void get\_pos();

该函数实现了得到当前机械臂末端位姿x、y、z、roll、pitch、yaw并显示在终端窗口中。

void change\_v(int v\_);

该函数实现了改变机械臂运动速度，程序中设置的机械臂默认速度为3651转速。

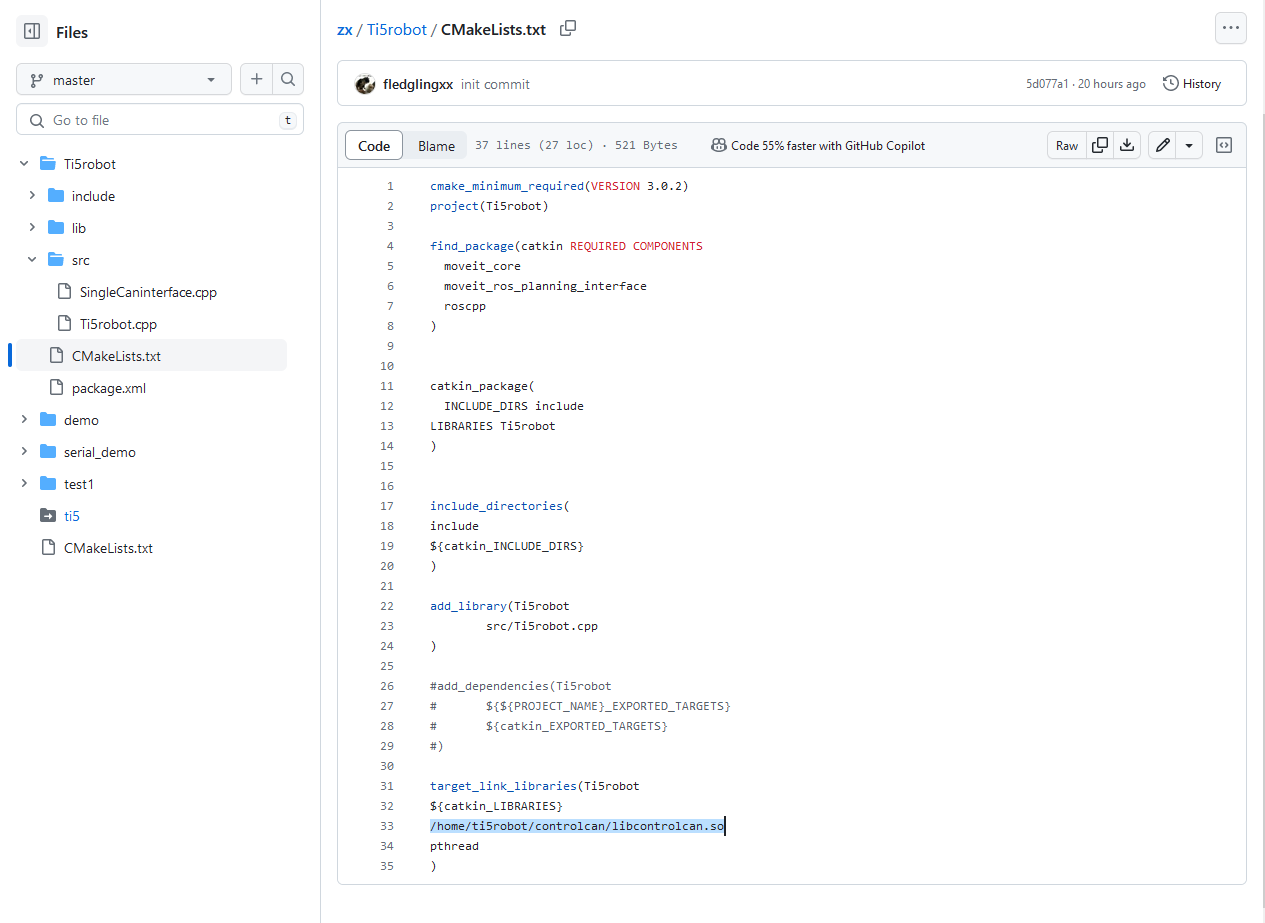
void clean\_error();

该函数实现了清除电机错误。电机有时会出现宕机不运动情况，该函数可以清除电机错误，作用类似于重启电机。

具体使用步骤说明：

1. Ti5robot功能包配置

由于与机械臂通信需要一个控制CAN的libcontrolcan.so库，请先确保Ti5robot功能包中的CMakeList.txt中的libcontrolcan.so库的路径是正确的，即根据下图中第33行的路径是否能够找到libcontrolcan.so。



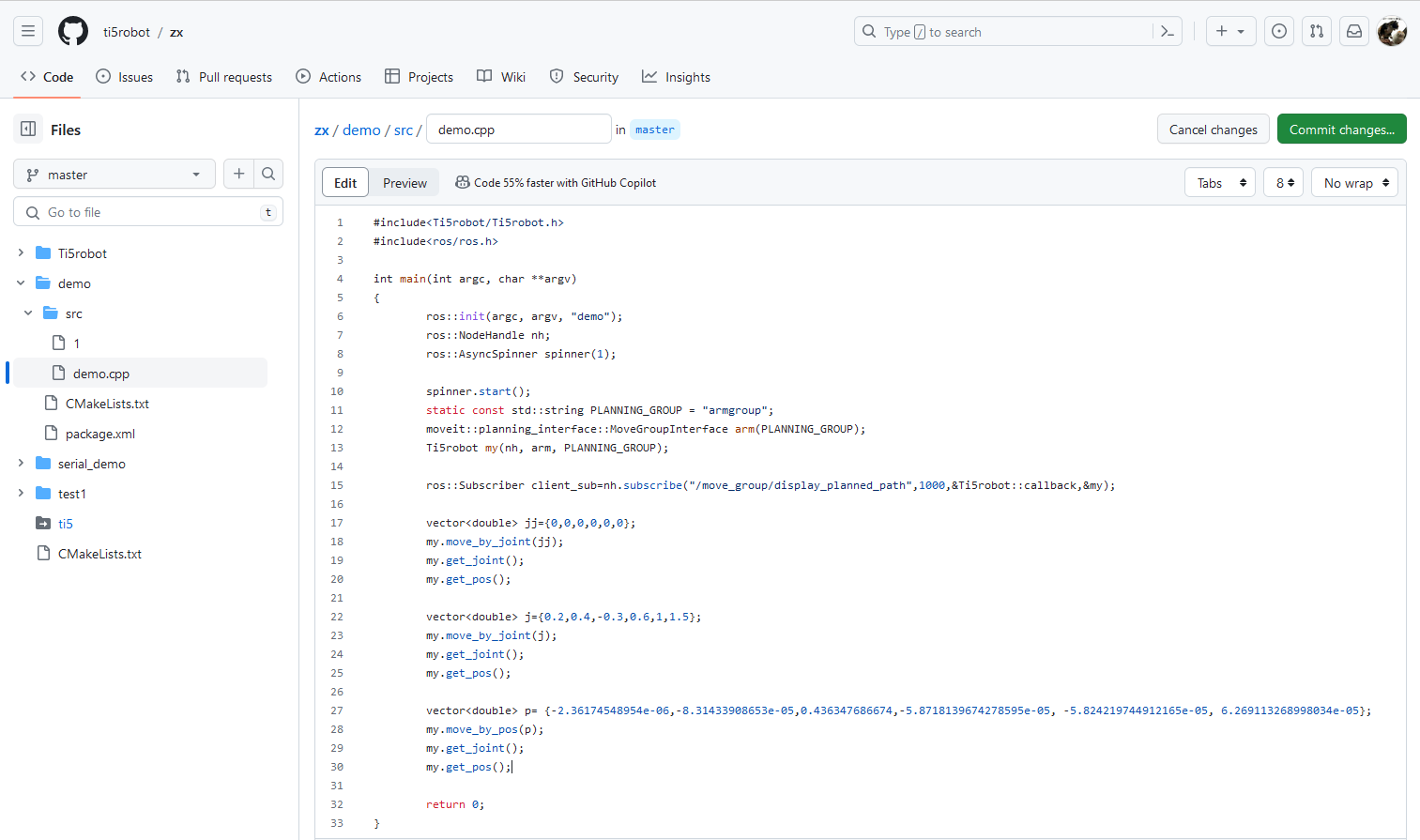
1. 创建新的功能包

完成第一步之后就可以创建新的功能包来使用Ti5robot功能包中的函数。

首先执行catkin\_create\_pkg demo roscpp rospy Ti5robot std\_msgs

该命令会创建一个demo功能包并生成CMakeList.txt、package.xml和src文件夹，在src文件夹中新建demo.cpp文件。

如下图所示



头文件中通过#include<Ti5robot/Ti5robot.h>来调用Ti5robot类的函数。

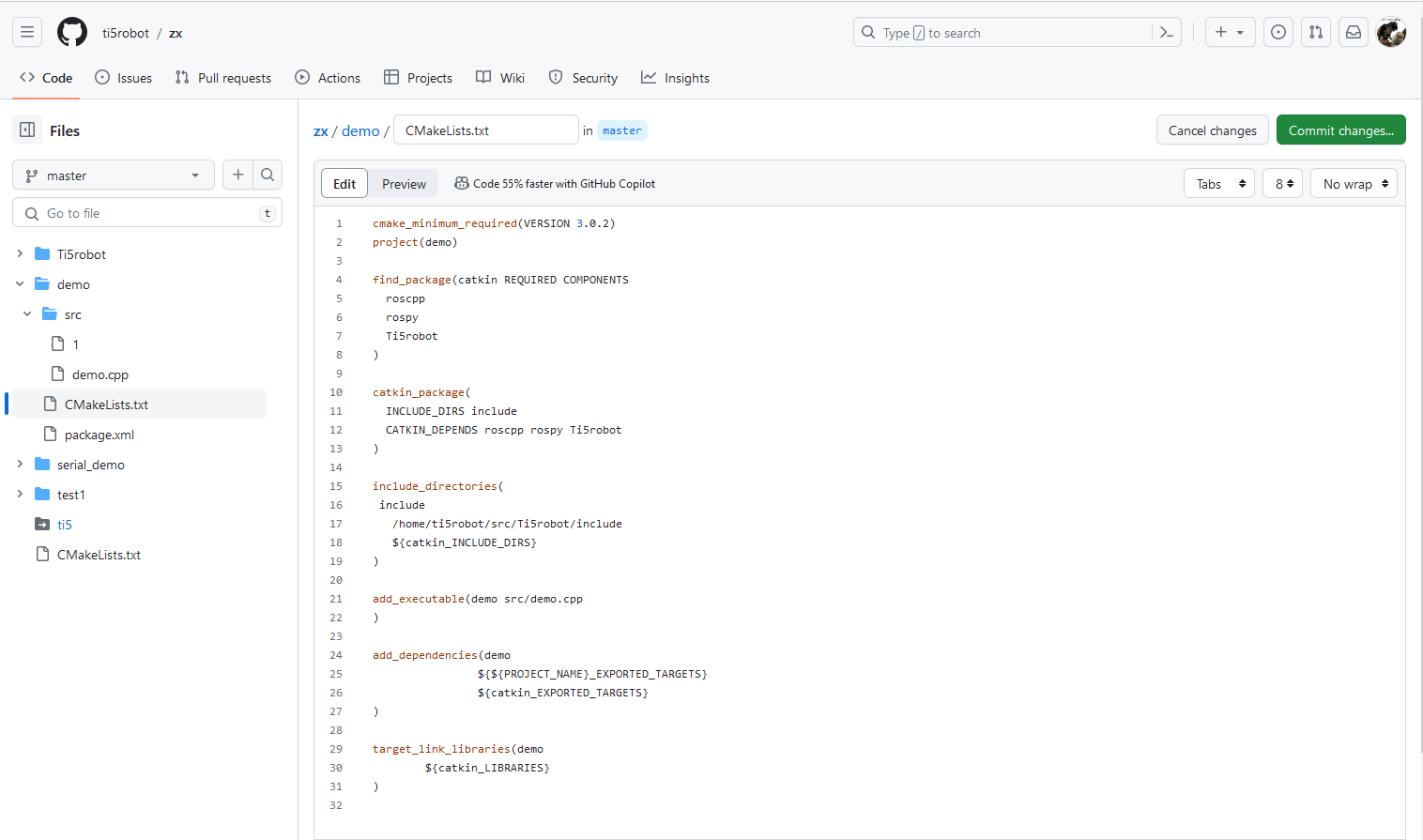
上图中的第6至15行实现了定义ros句柄，规划组，MoveGroupInterface对象并用来初始化一个叫my的Ti5robot类，并且实现订阅/move\_group/display\_planned\_path。

初始化之后可以通过调用my.move\_by\_pos(p)这种方式来调用Ti5robot类中实现好了的函数。

配置demo功能包的CMakeList.txt和package.xml

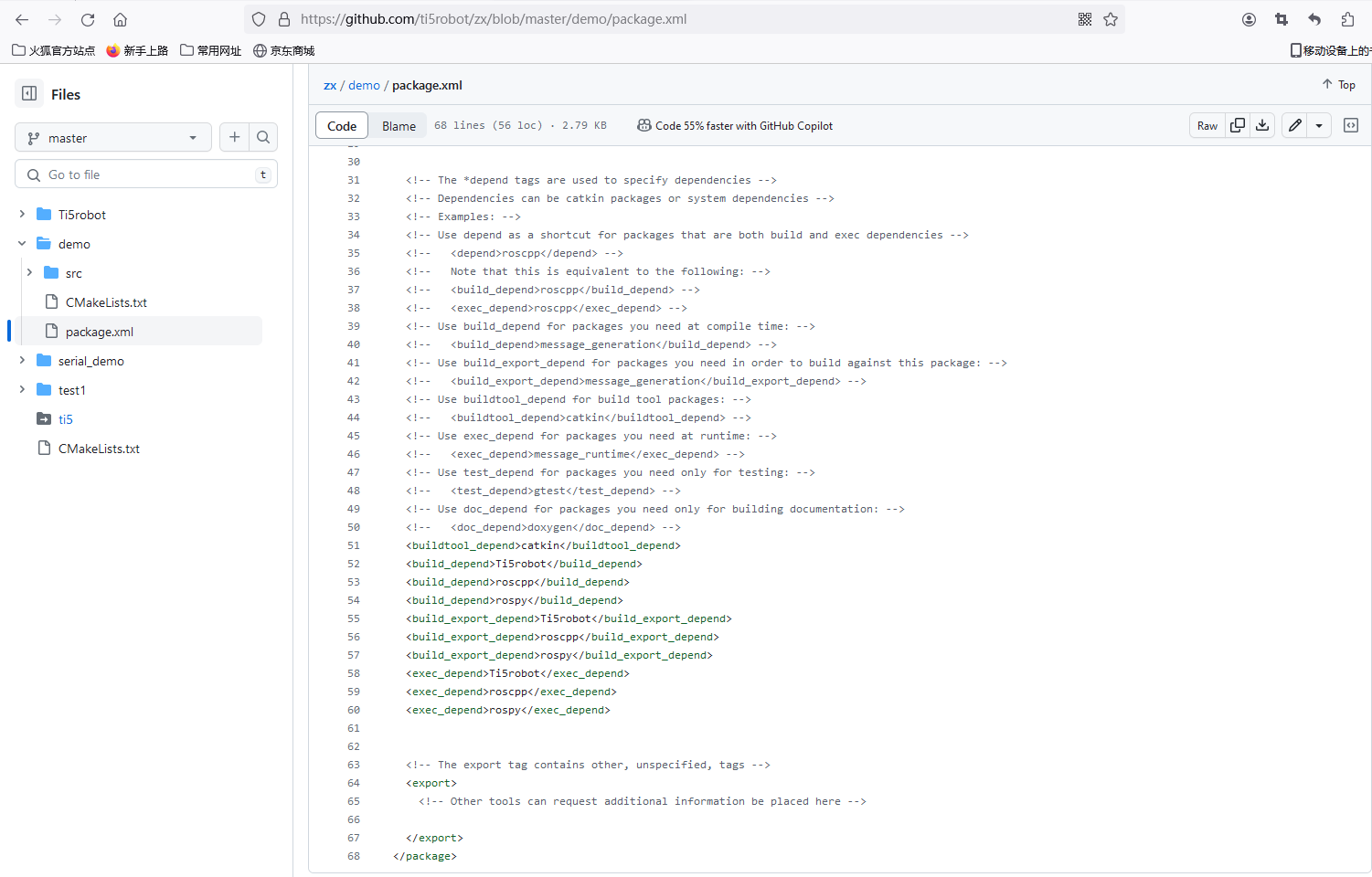
上述的demo.cpp编写完成之后，就可以配置demo功能包的CMakeList.txt和package.xml

CMakeList.txt



CMakeList.txt文件中find\_package中要加上Ti5robot，catkin\_package中CATKIN\_DEPENDS中也要加上Ti5robot，一般来说，如果创建功能包catkin\_create\_pkg时就加上了Ti5robot的话CMakeList.txt中就不需要再手动添加，然后在CMakeList.txt文件后面完善add\_dependencies和target\_link\_libraries相关即可。

package.xml



在package.xml文件中需要添加

<build\_depend>Ti5robot</build\_depend>

<build\_export\_depend>Ti5robot</build\_export\_depend>

<exec\_depend>Ti5robot</exec\_depend>

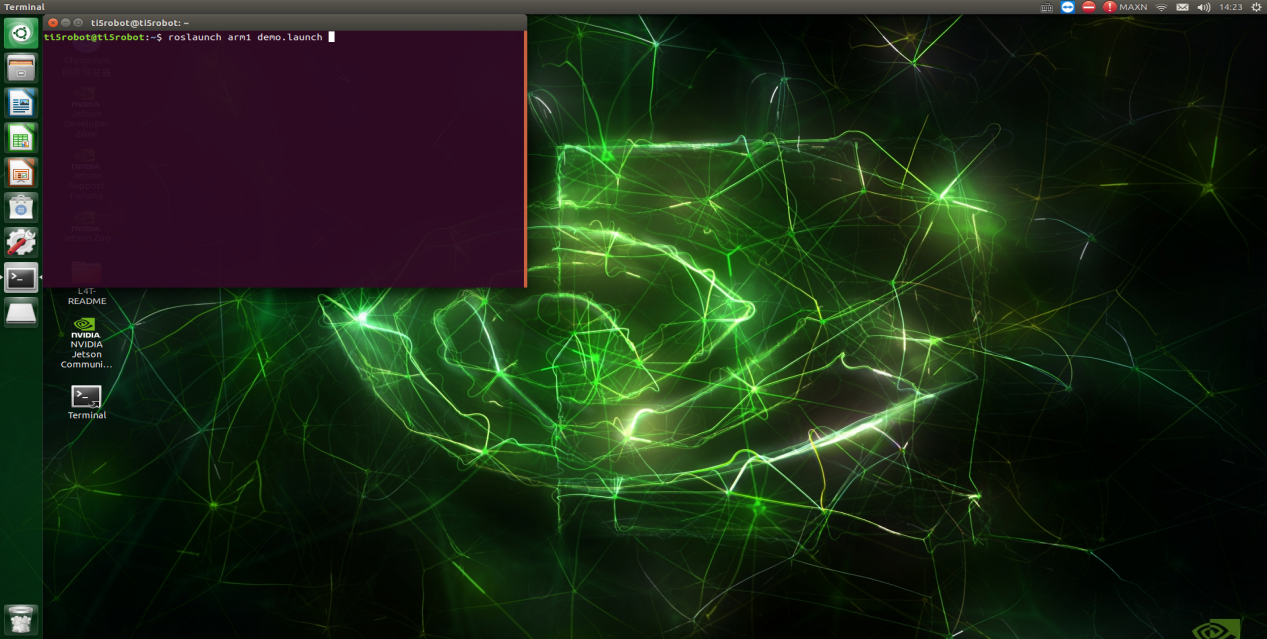
一般来说，如果创建功能包catkin\_create\_pkg时就加上了Ti5robot的话CMakeList.txt中就不需要再手动添加。

1. 编译运行

完成上述工作之后，在ros的工作空间下运行catkin\_make进行编译。

编译成功之后，记得运行source devel/setup.bash来使更改生效。

完成之后打开一个新的终端输入rosrun arm1 demo.launch （此处只以arm1为例）



然后在打开第二个终端，输入sudo chmod -R 777 /dev/

然后输入rosrun demo demo便可以看到机械臂的运动

