## Ubuntu14.04系统下安装ros\_barrett\_package教程

北京利曼科技有限公司

[Ubuntu14.04系统下安装ros\_barrett\_package教程 1](#_Toc30332)

[1. 下载及解压各种包 3](#_Toc30772)

[2. 创建工作空间（先执行第三步） 4](#_Toc2466)

[3. peak驱动安装 4](#_Toc8968)

[5. 通过launch文件登陆各节点 7](#_Toc31094)

[6. 启动服务/运行方式： 8](#_Toc11062)

[8 . 启动rqt 8](#_Toc24590)

[9 . Pyhand API 10](#_Toc26547)

[10. 灵巧手关节的定义 11](#_Toc4333)

此版本确认了PCAN的安装方法和常规错误的解决办法

此教程应用的版本为indigo 64为系统，可从以下网站下载镜像文件并安装：

ROS package to control the Barrett Hand. This package is based on the pyHand library for Linux. 以下安装步骤最好切换为root权限来安装，root权限提升及切换可通过以下代码来实现。

先安装驱动，最后再编译工作空间。

### 下载及解压各种包

<http://wiki.ros.org/barrett_hand>

<http://wiki.ros.org/libpcan>

<https://github.com/ipa320/cob_extern>

<https://github.com/RobotnikAutomation/pcan_python>

<http://www.peak-system.com/fileadmin/media/linux/index.htm>

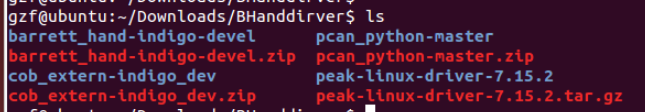
（linux 驱动下载 **7.x**） 本次使用7.15.2

从上述网页采用download到/home/robot/Download/

或者使用git clone <网址>

注意先确认下default branch是否为indigo

下载之后，分别解压



unzip ...zip

tar xzvf ....tar.gz

**建议先装驱动，再装包（从驱动看起）**

**预先装两个依赖项**

**$ sudo apt-get install swig**

**$ sudo apt-get install libpopt-dev**

### 创建工作空间（先执行第三步）

按照常规方法来安装ros package，需要先创建一个工作空间，然后用ros提供的catkin\_make命令来安装package，以下为安装步骤：

创建catkin工作空间（已有工作空间的话用原来即可）：

**$ mkdir -p /home/robot/bhand\_ws/src**

**$ cd ~/bhand\_ws**

**$ catkin\_make**

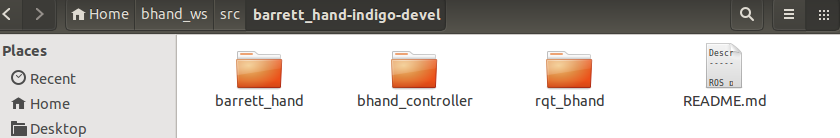
**$ source ~/bhand\_ws/devel/setup.bash**

**echo “source ~/bhand\_ws/devel/setup.bash”>>~/.bashrc**

**$ cp -r /home/robot/Downloads/barrett\_hand-indigo-devel /home/robot/** **bhand\_ws /src/**

**“或者从图形化文件管理器说，使用copy to/move to”**

**如果你下载的文件夹名字稍有区别，是因为下载方式不同，如git 命令行，或者网页选择分支造成的**



**$ cd ~/bhand\_ws**

**$ catkin\_make**

**$ catkin\_make Install （可选）只是安装空间devel还是install的区别**

（注：重新创建工作空间后需要用source命令临时设置环境变量，不然会提示找不到相应的相应节点或者launch文件）

$ source ~/bhand\_ws/devel/setup.bash

### peak驱动安装

解压cob\_extern\_indigo\_dev文件，并运行：

$cd ~/**Downloads**/

$unzip cob\_extern-indigo\_dev

**$ cd ~/****Downloads/cob\_extern-indigo\_dev/libpcan/**

**//编译libpcan 文件夹下文件**

**$ cmake CMakeLists.txt**

**$ make**

**$ sudo make install**

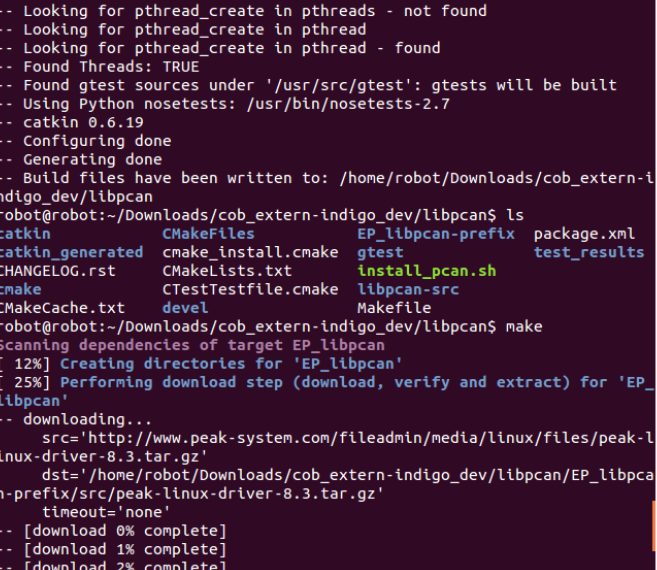
**//接下来安装peak-linux-driver-7.15.2(在libpcan文件夹下)**

**$ mkdir build**

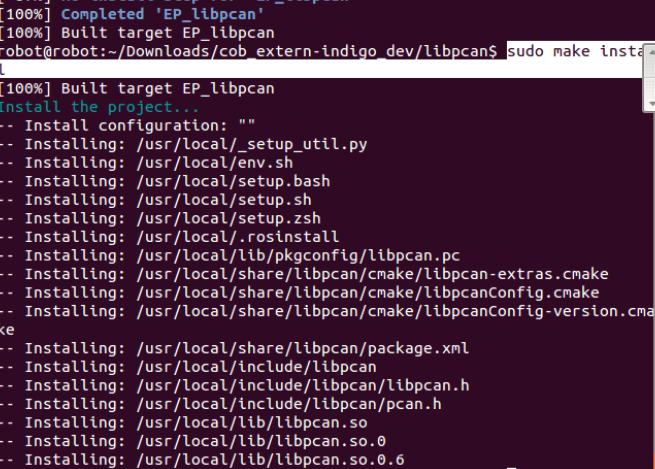
**然后peak-linux-driver-7.15.2 文件夹复制到libpcan/build文件夹下**

**$ cd build/peak-linux-driver-7.15.2 ~/Downloads/cob\_extern-indigo\_dev/libpcan/build**/**peak-linux-driver-7.15.2**

**$ make**



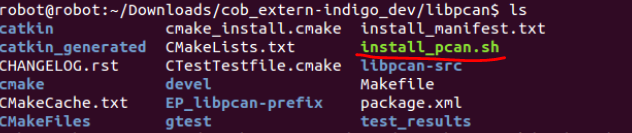
**$ sudo make install**



**$ cd ../..**

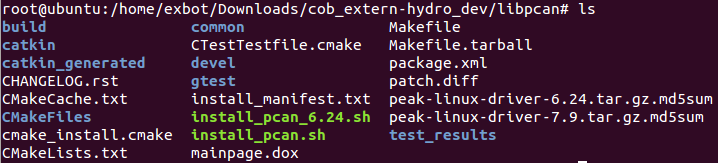
//使用文本编辑器打开install\_pcan.sh

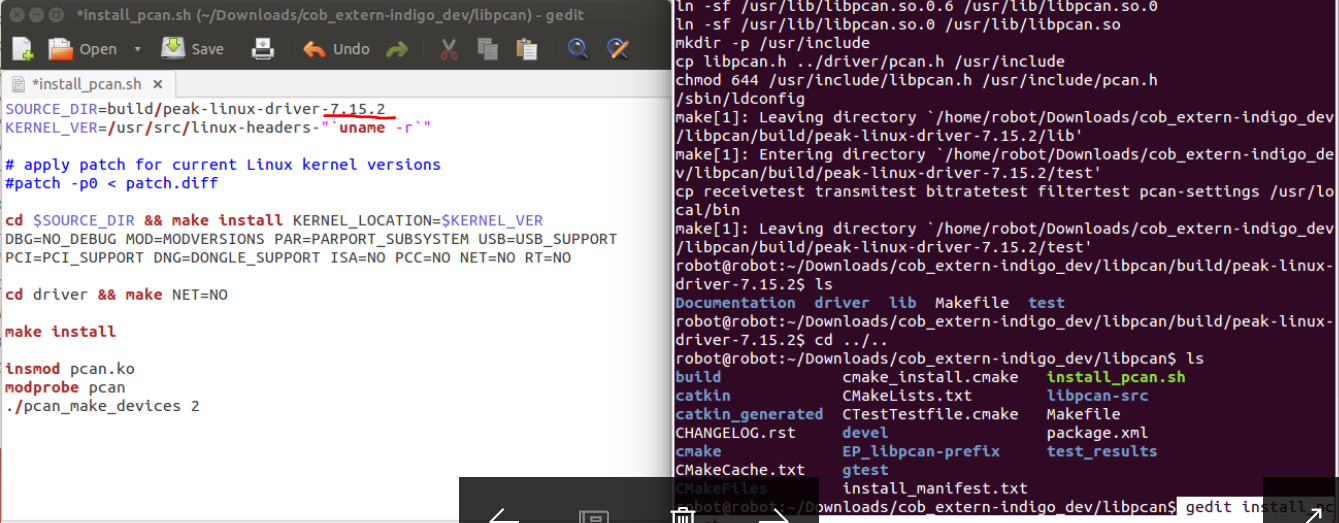
**$ gedit install\_pcan.sh**



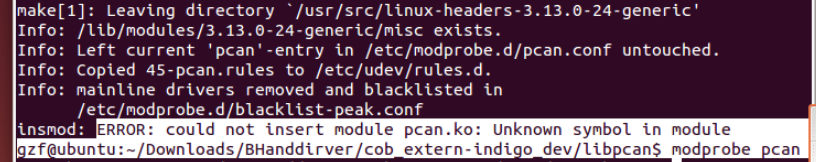
**第一行中将peak-linux-driver版本改成7.15.2（默认为8.3）,保存**

**$sudo ./install\_pcan.sh**





此步最后insmod报错的话，接着手动加载驱动：

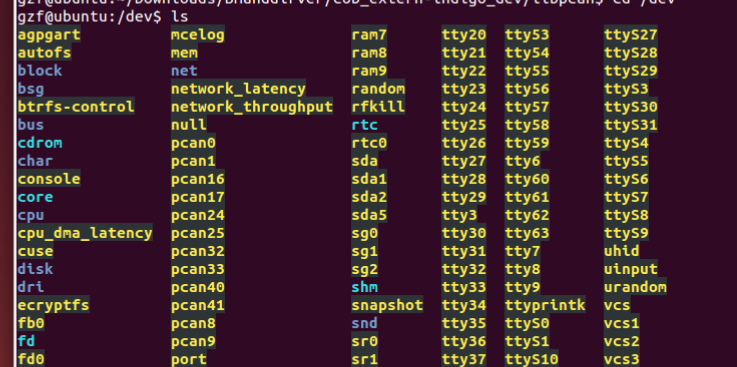


**$ modprobe pcan**

这时候应该是没有错误产生的

安装完成后在/dev/ 目录下可以看到CAN口设备 pcanusb0；

**$ ll –l /dev/pcanusb0**



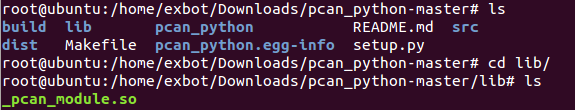
1. 编译 \_pcan\_module.so模块(该模块最终安装在/usr/lib下，下面export 声明不可少)

进入到pcan\_python-master文件夹下，然后：

**$ cd ~/Downloads/pcan\_python-master**

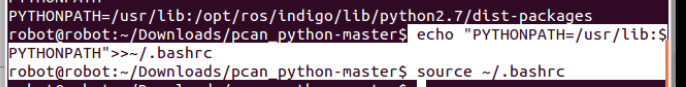
**$ make**

**$ sudo make install**



详见：<https://github.com/RobotnikAutomation/pcan_python>

**$ export PYTHONPATH=/usr/lib:$PYTHONPATH**



确认

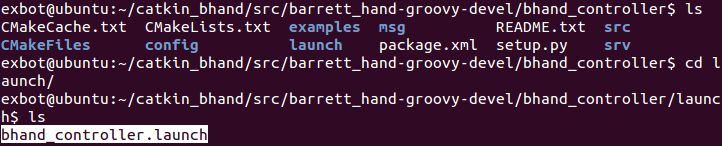
$ printenv |grep PY\*

========================驱动安装完成======================

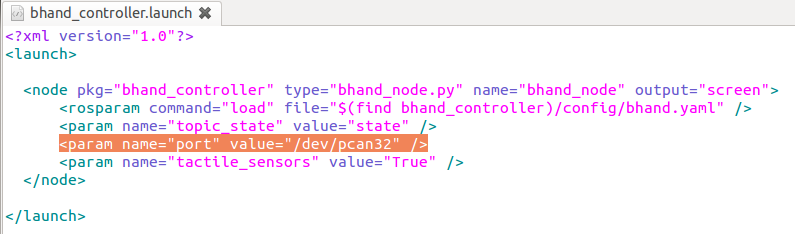
### 通过launch文件登陆各节点

例，Bhand\_controller launch 文件

**$source ~/bhand\_ws/devel/setup.bash**



如果launch文件无法启动，则打开launch文件中包含的can口设备信息：



有时候运行结点时候，提示初始化失败，此处value需要更改

### 启动服务/运行方式：

**$ cd /home/robot/bhand\_ws/devel**

**$ source setup.bash**

**$ roslaunch bhand\_controller bhand\_controller.launch**

新开一个命令行：

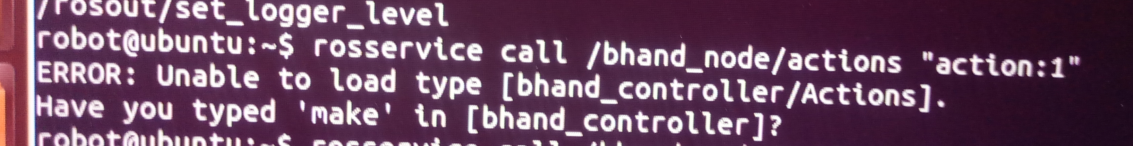
**$ cd /home/robot/bhand\_ws/devel**

**$ source setup.bash**

**$ rosservice call /bhand\_node/actions "action: 1"**

此时node便会连接机器人完成初始化，然后将灵巧手指移动到初始位置(张开)。

如果服务调用出错



此处文件需要额外编译

cd ~/bhand\_ws/src/ barrett\_hand-indigo-devel/bhand\_controller/

cmake .（注意.前有空格）

make

. ~/bhand\_ws/devel/setup.bash

Action1 : 初始化，在进行下面调用action时，这是必须步骤

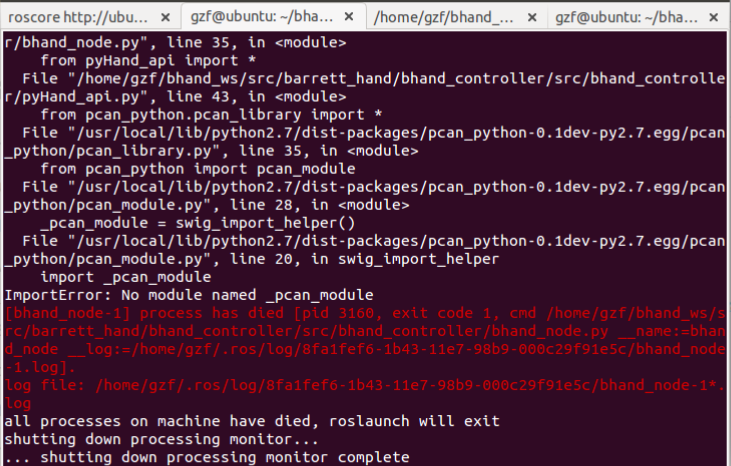
Action2: 手指关闭

Action3: 手指张开

### 8 . 启动rqt

启动workspace/src/rqt/launch 下的rqt界面程序，类似pyhand

如果启动节点错误，发现如下错误：



请将行面两个PYTHONPATH环境变量添加到/home/robot/.bashrc最后面，以便在每次启动shell 时自动更新

****$ export PYTHONPATH=/usr/lib:$PYTHONPATH****

$cd ~/bhand\_ws/src/barrett\_hand-indigo-devel/rqt\_bhand/scripts&&./rqt\_band

或者

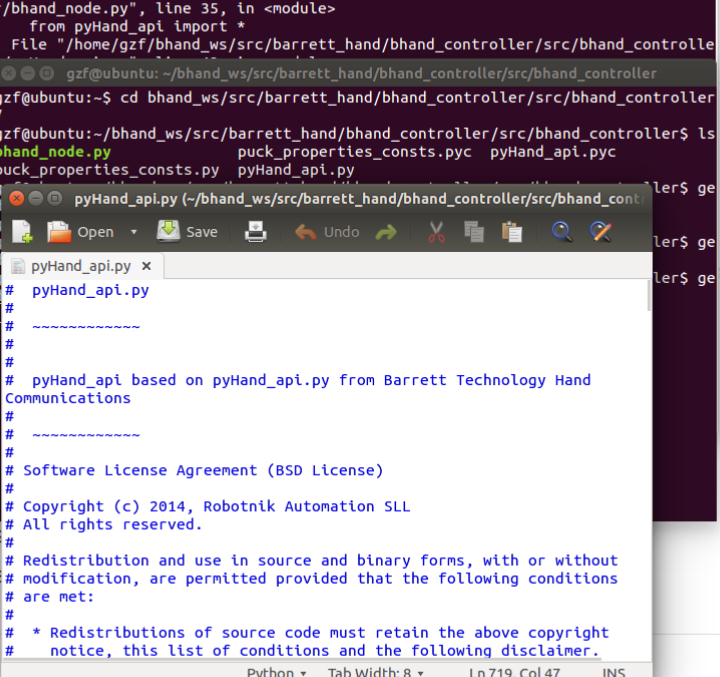
$rosrun rqt\_bhand rqt\_bhand

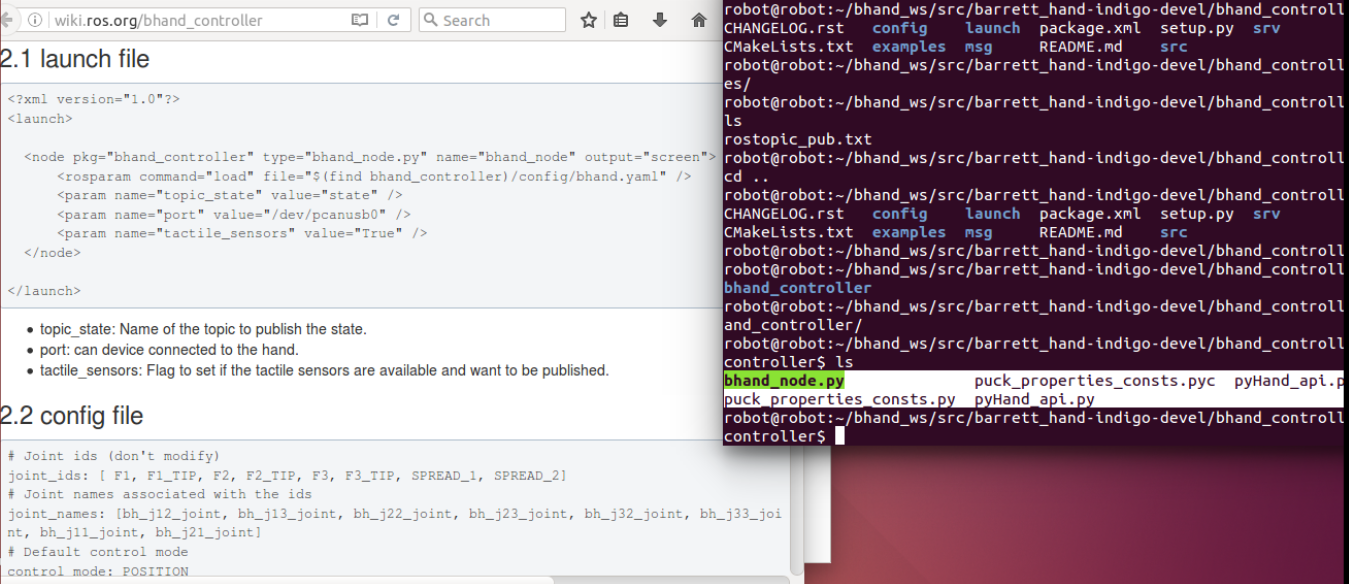
打开后点击初始化

初始化下面的滑动条，注意选择位置和速度模式，以及下面的enable motor 打勾。

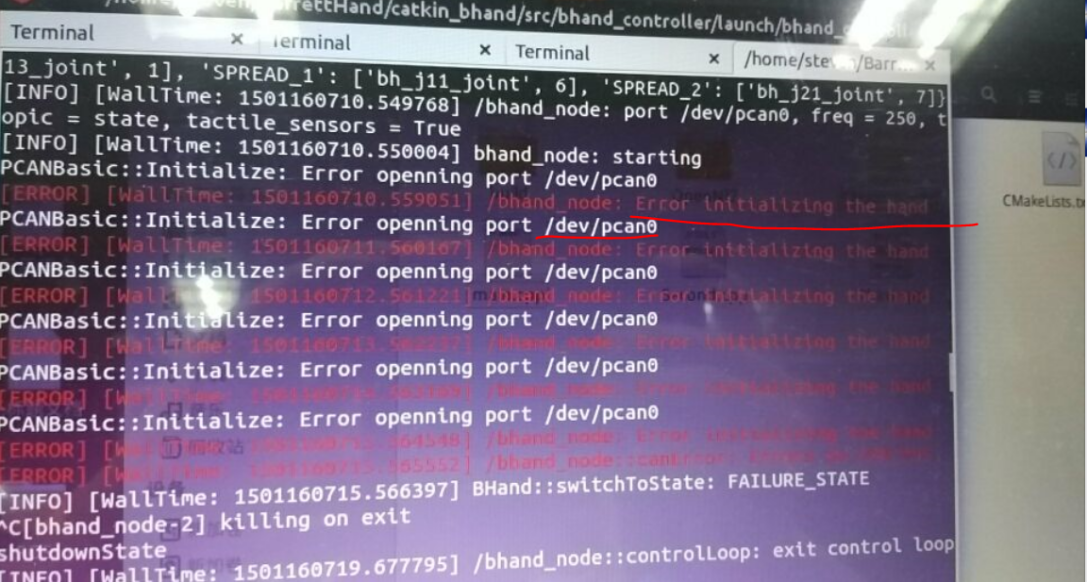
Mode1 和mode2 表示spread自由度的两个极限位置

### 9 . Pyhand API





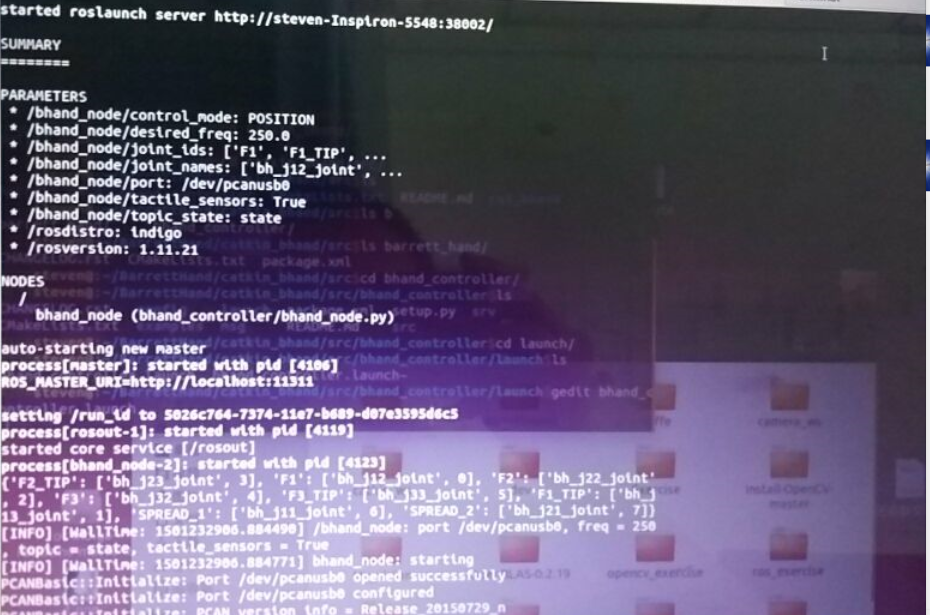
如果遇到了错误如下：



**(1)请将launch文件的pcan0改成pcanusb0**

**(2)确认插上can转usb到电脑usb上，查看/dev/是否生成pcanusb0文件？？！！！**

**如果没有，则驱动没有正确安装。**

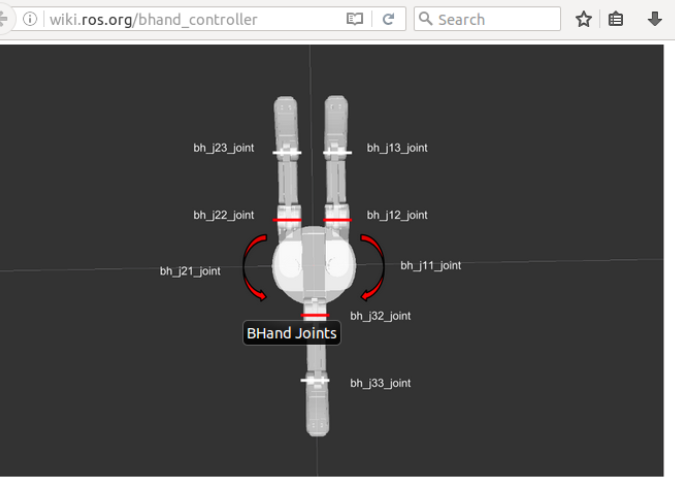


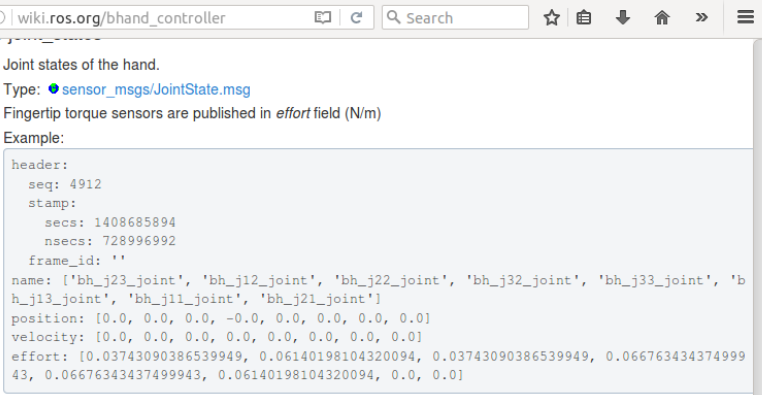
### 灵巧手关节的定义

（图下为home位置）

（此时关节F1F2F3编码器数值为0，闭合时最大数值CT为190000，DS=25600)

（此时spread编码器数值为0，闭合式最大数值CT为35950，DP=17975）





Spread : bh\_j11\_joint bh\_j21\_joint

F1: bh\_j12\_joint(F1) bh\_j13\_joint(F1\_TIP)

F2: bh\_j22\_joint(F2) bh\_j23\_joint（F2\_TIP）

F3: bh\_j32\_joint(F3) bh\_j33\_joint(F3\_TIP)

Action1 为初始化

Aciton2 为close

Action3？

4 为打开

Action5 三指并拢 张开

Action6 三指并拢 半闭

Action7 三指并拢 闭合

截自Bhand\_controller/bhand\_node.py

# Predefined actions

if len(self.actions\_list) > 0:

action = self.actions\_list[0]

# Removes action from list

self.actions\_list.remove(action)

# Actions performed in CONTROL POSITION

if self.control\_mode == CONTROL\_MODE\_VELOCITY:

self.setControlMode(CONTROL\_MODE\_POSITION)

if action == Service.CLOSE\_GRASP:

self.closeFingers(3.14)

if action == Service.CLOSE\_HALF\_GRASP:

self.closeFingers(1.57)

elif action == Service.OPEN\_GRASP:

self.openFingers()

elif action == Service.SET\_GRASP\_1:

if self.joint\_state.position[f1\_index] > 0.1 or self.joint\_state.position[f2\_index] > 0.1 or self.joint\_state.position[f3\_index] > 0.1:

#rospy.logerr('BHand::ReadyState: Service SET\_GRASP 1 cannot be performed. Rest of fingers have to be on zero position')

self.openFingers()

time.sleep(2.0)

self.desired\_joints\_position['SPREAD\_1'] = 0.0

self.desired\_joints\_position['SPREAD\_2'] = 0.0

self.hand.move\_to(SPREAD, self.hand.rad\_to\_enc(self.desired\_joints\_position['SPREAD\_1'], BASE\_TYPE), False)

self.grasp\_mode = action

elif action == Service.SET\_GRASP\_2:

if self.joint\_state.position[f1\_index] > 0.1 or self.joint\_state.position[f2\_index] > 0.1 or self.joint\_state.position[f3\_index] > 0.1:

#rospy.logerr('BHand::ReadyState: Service SET\_GRASP 2 cannot be performed. Rest of fingers have to be on zero position')

self.openFingers()

time.sleep(2.0)

#else:

self.desired\_joints\_position['SPREAD\_1'] = 3.14

self.desired\_joints\_position['SPREAD\_2'] = 3.14

self.hand.move\_to(SPREAD, self.hand.rad\_to\_enc(self.desired\_joints\_position['SPREAD\_1'], BASE\_TYPE), False)

self.grasp\_mode = action

# NO pre-defined actions

else:

if self.control\_mode == CONTROL\_MODE\_POSITION:

# Moves joints to desired pos

if self.new\_command:

try:

f1\_joint = self.joint\_names['F1'][0]

if self.desired\_joints\_position[f1\_joint] != self.joint\_state.position[f1\_index]:

#print 'moving joint %s to position %f'%(self.joint\_state.name[0], self.desired\_joints\_position['j12\_joint'])

self.hand.move\_to(FINGER1, self.hand.rad\_to\_enc(self.desired\_joints\_position[f1\_joint], BASE\_TYPE), False)

f2\_joint = self.joint\_names['F2'][0]

if self.desired\_joints\_position[f2\_joint] != self.joint\_state.position[f2\_index]:

#print 'moving joint %s to position %f'%(self.joint\_state.name[2], self.desired\_joints\_position['j22\_joint'])

self.hand.move\_to(FINGER2, self.hand.rad\_to\_enc(self.desired\_joints\_position[f2\_joint], BASE\_TYPE), False)

f3\_joint = self.joint\_names['F3'][0]

if self.desired\_joints\_position[f3\_joint] != self.joint\_state.position[f3\_index]:

#print 'moving joint %s to position %f'%(self.joint\_state.name[4], self.desired\_joints\_position['j32\_joint'])

self.hand.move\_to(FINGER3, self.hand.rad\_to\_enc(self.desired\_joints\_position[f3\_joint], BASE\_TYPE), False)

spread1\_joint = self.joint\_names['SPREAD\_1'][0]

spread2\_joint = self.joint\_names['SPREAD\_2'][0]

if self.desired\_joints\_position[spread1\_joint] != self.joint\_state.position[spread1\_index]:

#print 'moving joint %s to position %f'%(self.joint\_state.name[6], self.desired\_joints\_position['jspread\_joint'])

self.hand.move\_to(SPREAD, self.hand.rad\_to\_enc(self.desired\_joints\_position[spread1\_joint], SPREAD\_TYPE), False)

elif self.desired\_joints\_position[spread2\_joint] != self.joint\_state.position[spread2\_index]:

#print 'moving joint %s to position %f'%(self.joint\_state.name[6], self.desired\_joints\_position['jspread\_joint'])

self.hand.move\_to(SPREAD, self.hand.rad\_to\_enc(self.desired\_joints\_position[spread2\_joint], SPREAD\_TYPE), False)

except Exception, e:

rospy.logerr('%s::readyState: error sending command: %s'%(rospy.get\_name(), e))

errors = errors + 1

self.new\_command = False

else:

# VELOCITY CONTROL

if ((time.time() - self.timer\_command) >= self.watchdog\_command):

try:

#rospy.loginfo('BHand::readyState: Watchdog velocity')

self.desired\_joints\_velocity[self.joint\_names['F1'][0]] = 0.0

self.desired\_joints\_velocity[self.joint\_names['F2'][0]] = 0.0

self.desired\_joints\_velocity[self.joint\_names['F3'][0]] = 0.0

self.desired\_joints\_velocity[self.joint\_names['SPREAD\_1'][0]] = 0.0

self.desired\_joints\_velocity[self.joint\_names['SPREAD\_2'][0]] = 0.0

self.setJointVelocity('F1')

self.setJointVelocity('F2')

self.setJointVelocity('F3')

self.setJointVelocity('SPREAD\_1')

except Exception, e:

rospy.logerr('%s::readyState: error sending command: %s'%(rospy.get\_name(), e))

errors = errors + 1

# Moves joints to desired pos

if self.new\_command:

try:

self.setJointVelocity('F1')

self.setJointVelocity('F2')

self.setJointVelocity('F3')

self.setJointVelocity('SPREAD\_1')

except Exception, e:

rospy.logerr('%s::readyState: error sending command: %s'%(rospy.get\_name(), e))

errors = errors + 1

self.new\_command = False

time.sleep(0.002)