**概述**

在之前ROSJava配置的时候，我就提到了，我们需要建立ROS与Android设备之间的交互，通过网络建立节点间通信，才能够使手机端可以成功访问ROS中发布的话题和消息。从已知的方案中，我们可以使用ROSBridge直接进行网络连接进行底层的信息交互，这种交互传递的数据一定程度上是与ROS本身无关的，但是从零开始使用ROSBridge自行构建通信比较复杂。于是我想通过使用已有的交互架构进行编程，那就是Android Remocon。

**机器人端软件部署**

使用如下命令安装软件：

**sudo apt-get install ros-indigo-rocon-interactions ros-indigo-rocon-master-info ros-indigo-zeroconf-avahi-suite**

该软件主要用于配合android\_interaction包进行交互。

其次要安装rosbridge包，我们之前提到ROSbridge的作用，在rocon软件中使用了rosbridge作为连接的工具，该包通过如下命令安装：

**sudo apt-get install ros-indigo-rosbridge-server**

至此机器人端也就是计算节点的软件部署就完成了。

**Android端软件部署**

首先需要使用源码安装android\_interaction包并使用catkin工具编译，该软件包是在rosjava和android\_core两个包已部署的基础上进行交互的，该软件包可通过如下命令进行：

**mkdir -p ~/android\_interactions**

**wstool init -j4 ~/android\_interactions/src https://raw.github.com/rosjava/rosjava/indigo/android\_interactions.rosinstall**

**source /opt/ros/indigo/setup.bash**

**source ~/rosjava/devel/setup.bash**

**source ~/android\_core/devel/setup.bash**

**cd ~/android\_interactions**

**catkin\_make**

编译成功之后使用命令**source ~/android\_interactions/devel/setup.bash**，读取相应的环境变量。

我们也可以使用echo ‘…’ > ~/.bashrc把如下几条source指令统一添加到~/.bashrc中：
**source /opt/ros/indigo/setup.bash**

**source ~/rosjava/devel/setup.bash**

**source ~/android\_core/devel/setup.bash**

**source ~/android\_interactions/devel/setup.bash**

然后从Ubuntu虚拟机上打开Android Studio，导入项目，我们这里有两个项目可以导入，一个是android\_core，另一个是android\_remocons，首先使用Gradle进行同步，这里要注意修改android\_core中buildscript.gradle和android\_remocons中build.gradle的内容，把classpath中的版本从1.0.+改成3.2.0。



图1 Plugin版本修改前



图2 Plugin版本修改后

这个问题之前我在讨论区中进行了描述，可以从中看到该问题全部的解决过程。

每个版本的Plugin对应的Gradle版本如下：
 

图3 官网上的版本对应

之后编译这两个项目，并选择其中的子软件安装到手机上。

**ROS通信网络建立**

ROS本身是一个分布式的计算系统。在运行过程中，一个ROS系统可能会由成百上千个分布在不同物理设备上的节点共同构成。这个由ROS节点构成的网络如果要能够正常运行，必须要保证以下两点：

* 任意两节点之间双向连通；
* 每个节点具有唯一且可识别的名称。

而目前我们需要用到ROS节点交互的地方有如下两处：

* 通过ssh远程连接到计算节点的网络；
* 通过移动APP与计算节点网络进行交互。

我主要通过移动APP与计算节点网络进行交互来展示和ROS节点进行沟通的方式：
首先要修改Ubuntu虚拟机的配置：



图4 修改虚拟机网络连接设置

要把虚拟机的网卡（网络适配器）属性进行修改，从NAT模式转换修改成桥接模式。原因是如果使用网络地址转换的话，从移动客户端无法使用分配给虚拟机的IP地址直接访问到虚拟机（应该也可以设置静态的IP，要保证在同一个子网下，这个我还没有尝试），而如果使用桥接模式的话，虚拟机可直接连接到物理网络。

之后在虚拟机的虚拟网络编辑器中，把使用的网卡改成本机的无线网卡（通过手机热点进行组网）。



图5 修改虚拟机使用的网卡

之后在命令解释器中使用ifconfig命令查看虚拟机的IP地址，这里我的IP地址为172.20.10.13：



图6 查看虚拟机的IP地址

然后再~/.bashrc中添加如下命令：

**export ROS\_IP=172.20.10.13**
将如下命令：

**export ROS\_HOSTNAME=localhost**

**export ROS\_MASTER\_URI =http:// localhost:11311**

修改为：

**export ROS\_HOSTNAME=172.20.10.13**

**export ROS\_MASTER\_URI =http://172.20.10.13:11311**

之后获取到移动客户端的IP地址（172.20.10.14,，可以通过手机的“设置”-“关于手机”-“状态设置”中查看），互相ping一下，如果虚拟机打开了防火墙的话，可以通过如下命令进行查看和修改：

**sudo ufw status**

**sudo ufw disable**

最后如果可以相互ping通的话，就可以暂时告一段落了，下一节将进行几个Demo的展示。