TurtleBot3 移动机器人实验报告

1. 实验目的 （通过需要的实验目的去设计实验）

已知起始位置和已知的最终位置，在已知地图的周围规划一条路径，不能接触地图中的任何障碍物。

1）基于图的规划方法(Graph-based Plan Methods)

在这个模块中，我们将介绍经过网格的路径规划问题，在该规划方法中机器人只能出现在离散位置上。我们可以将这些情境建模为图，其中的节点对应于网格位置，边对应于相邻网格单元之间的路径。教材中的算法原理，可以用于规划起始节点和目标节点之间的路径，一些经典算法包括 Voronoi 图法等。

2）形位空间(Configuration Space, C-Space)的计算

这个模块，在于理解掌握形位空间的概念。它是一个数学工具，用于考虑机器人能够达到的位置集合。相应地，形位空间障碍的概念，指形位空间中由于存在障碍或其他实验装置，机器人无法到达的区域。这个公式有利于促进对路径规划问题的思考，允许通过形位空间点构造轨迹。通过离散连续的形位空间为图，就可以应用基于图的工具来解决我们的运动规划问题。

3）基于采样的路径规划法

在这个模块中，通过实验理解掌握基于样本的路径规划技术的概念。这涉及到在形位空间中随机采样点，然后在相邻的采样点之间形成无碰撞的边缘，形成一个捕获机器人形位空间结构的图。探索概率路线图和快速随机搜索树（Randomly Exploring Rapid Trees, RRTS）及其运动规划问题中的应用。

2. 实验原理

（待补充）

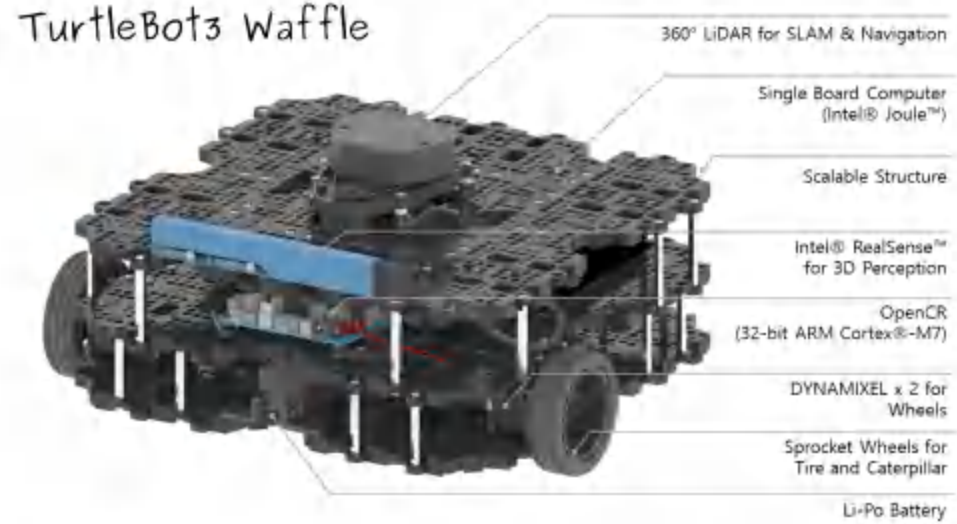
3. 实验环境

3.1. 机器人设备：

3.1.1. Turtlebot3 介绍：

TurtleBot3 是一款模块化，紧凑且可定制的新一代移动机器人。TurtleBot3 的目标是在不牺牲功能，功能和质量的情况下大幅缩小平台的尺寸并降低价格。诸如底盘，计算机和传感器等可选部件都可用。TurtleBot3 可以通过各种方式进行定制。 TurtleBot3 愿意通过应用 SBC（单板计算机）的最新技术进步，深度传感器和 3D 打印技术，成为制造商运动的中心。

3.1.2. Turtlebot 外形



3.1.3. Turtlebot3 特性：

1）流行的开发平台

Turlebot 是最流行的教育和研究用的开源机器人。新一代的 Turtlebot3 兼顾小型、廉价、全可编程的基于 ROS 的移动机器人。它是为教育、研究、业余爱好和产品原型开发的新一代产品。

2）可承受的价格

Turtlebot 是为了迎合学校、实验室和企业的预算限制的需求开发的产品，Turtlebot3 是目前市面上装备有 360 度雷达距离传感器、能够使用 SLAM 功能的价格最实惠的移动机器人

3）小尺寸携带便捷

TurtleBot3 Burger 的尺寸仅为 138mm x 178mm x 192mm（长 x 宽 x 高），它的大小约为前辈的四分之一。想象一下，将 TurtleBot3 放在背包里，开发你的程序并在任何地方进行测试。

4）ROS 标准

TurtleBot 品牌由 Open Robotics 管理，Open Robotics 负责开发和维护 ROS。如今，ROS 已经成为全球所有机器人专家的首选平台。TurtleBot 可以与现有的基于 ROS 的机器人组件集成，但 TurtleBot3 可以成为一个负担得起的平台，为他们开始学习 ROS。

5）可扩展性

TurtleBot3 鼓励用户使用一些替代选项来定制其机械结构：开源嵌入式电路板（作为控制板），计算机和传感器。TurtleBot3 Burger 是一款两轮差速驱动式平台，但它可以通过多种方式进行结构和机械定制：汽车、自行车、拖车等。通过可扩展结构上的各种 SBC、传感器和电机，将您的想法超越想象。

6）移动机器人的模块化执行器

TurtleBot3 能够通过在车轮关节中使用 2 个 DYNAMIXEL 获得精确的空间数据。DYNAMIXEL XM 系列可通过 6 种操作模式（XL 系列：4 种操作模式）中的一种操作：轮子的速度控制模式、转矩控制模式或关节的位置控制模式等。DYNAMIXEL 甚至可用于制造移动机械手。轻，但可以通过速度、扭矩和位置控制进行精确控制。DYNAMIXEL 是使 TurtleBot3 完美的核心组件，组装、维护、更换和重新配置非常简单。

7）为 ROS 开放控制板

控制板采用硬件方式开源，并以 ROS 通信的软件方式进行开源。开源控制板 OpenCR1.0 功能强大，不仅可以控制 DYNAMIXEL，还可以控制常用于基本识别任务的 ROBOTIS 传感器，而且成本效益更高。各种传感器，如触摸传感器、红外传感器、彩色传感器和一些更多可用。OpenCR1.0 在电路板内部有一个 IMU 传感器，因此它可以增强对无数应用的精确控制。该电路板具有 3.3V、5V、12V 电源，以加强可用的计算机设备阵容。

8）强大的传感器阵容

TurtleBot3 Burger 使用增强型 360°LiDAR，9 轴惯性测量单元和精密编码器来进行研究和开发。TurtleBot3 Waffle 配备了完全相同的 360°LiDAR，另外还提供了一个强大的带有识别 SDK 的英特尔®Realsense™，这将是制造移动机器人的最佳硬件解决方案。

9）开源

TurtleBot3 的硬件，固件和软件都是开源的，这意味着欢迎用户下载，修改和共享源代码。TurtleBot3 的所有组件均采用注塑成型塑料制造以实现低成本，但 3D CAD 数据也可用于 3D 打印。3D CAD 数据通过全云 3D CAD 编辑器 Onshape 发布。用户可以通过桌面 PC，笔记本电脑甚至便携式设备上的网络浏览器进行访问。Onshape 允许绘制 3D 模型并将其与同事组装。此外，对于想要自行开发 OpenCR1.0 开发板的用户来说，OpenCR1.0 开发板的所有细节，如原理图，PCB Gerber 文件，BOM 和固件源代码都是在用户开放源代码许可证和 ROS 社区。您可以修改下载的源代码和硬件以与朋友分享。

3.2. 软件环境

3.2.1.ROS 操作系统来源

ROS 是为机器人设计的开源、基于元操作的系统，它确是着你对操作系统服务的期望，包括硬件抽象，低等级驱动控制，常用功能的实现，进程之间的消息传递和包管理，它还提供用于在多台计算机上获取，生成，写入和运行代码的工具和库。

ROS 是一个有许多祖先和贡献者的大型项目，机器人研究界的许多人都认为需要一个开放式的协作框架，并且已经为此目标创建了许多项目。

斯坦福大学在二十世纪二十年代中期进行了各种努力，涉及 STanford 人工智能机器人（STAIR）和个人机器人（PR）计划等集成的，体现的人工智能，为机器人应用创建了灵活的动态软件系统的内部原型。2007 年，附近有远见的机器人孵化器 Willow Garage 提供了大量资源来进一步扩展这些概念并创建经过充分测试的实现。无数的研究人员为核心 ROS 理念及其基本软件包贡献了他们的时间和专业知识，推动了这一努力。在整个过程中，该软件都是使用宽松的 BSD 开源许可进行公开开发的，并逐渐成为机器人研究界广泛使用的平台。

从一开始，ROS 就是在多个机构和多个机器人开发的，其中包括许多从 Willow Garage 收到 PR2 机器人的机构。尽管所有贡献者将代码放置在同一台服务器上要简单得多，但多年来，“联合”模型已成为 ROS 生态系统的一大优势。任何团队都可以在自己的服务器上启动他们自己的 ROS 代码库，并且他们可以维护它的完全所有权和控制权。他们不需要任何人的许可。如果他们选择公开提供他们的存储库，他们可以获得应有的成就和荣誉，并从特定的技术反馈和改进中受益，例如所有开源软件项目。

3.2.2.ROS 系统特性

ROS 生态系统现在由全球数以万计的用户组成，致力于从桌面业余爱好项目到大型工业自动化系统等领域。

机器人操作系统（ROS）是编写机器人软件的灵活框架。它是一组工具、库和约定，旨在简化在各种机器人平台上创建复杂而强大的机器人行为的任务。

为什么？因为创建真正强大的通用机器人软件非常困难。从机器人的角度来看，对于人类来说微不足道的问题往往在任务和环境的实例之间大相径庭。处理这些变化非常困难，以至于没有任何一个人、实验室或机构可以希望自己做。

因此，ROS 从根本上建立起来，鼓励协作机器人软件开发。例如，一个实验室可能拥有测绘室内环境的专家，并可以为制作地图提供一世界级的系统。另一组可能有专家在使用地图进行导航，而另一组可能已经发现了一种计算机视觉方法，该方法对于识别杂物中的小物体非常有效。ROS 是专门为像这样的团体设计的，以便在本网站中描述的彼此协作和共同工作。

3.3. 网络环境

ROS 是一种分布式计算环境。一个运行的 ROS 系统可以包括许多，甚至数以百计的节点，分布在多台机器上。根据系统的配置方式，任何节点在任何时候都可能需要与任何其他节点通信。

因此，ROS 对网络配置有一定的要求：

- 必须有完整的，在所有端口上的所有对机器之间的双向连接。
- 每个机器必须用所有其他机器可以识别的名称来做广播。

则在实验环境中较为便捷的网络配置方案为：给每一个机器人配置不同的节点名称，将计算主机和机器人通过同一无线网络路由器进行链接。

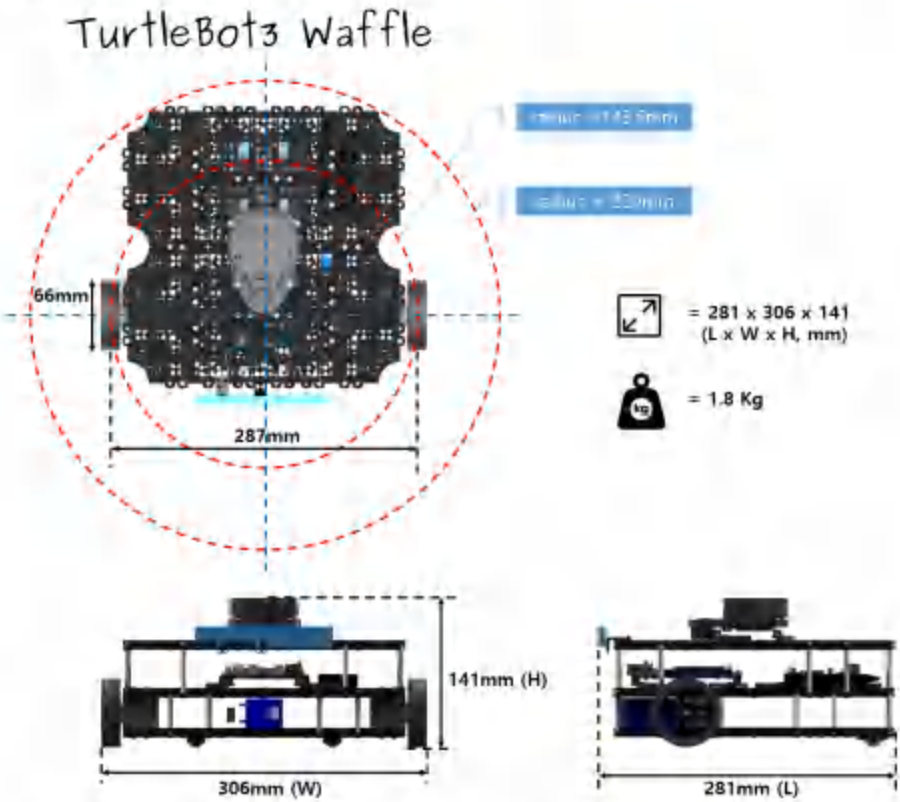
4. 实验配置方法

4.1. TurtleBot3 移动机器人装配（安装尺寸及运动性能参数、外形）

Turtlebot3 新产品需要自行组装，根据随机器提供的组装说明书进行组装。

电子版组装教程见链接：

http://emanual.robotis.com/docs/en/platform/turtlebot3/hardware_setup/#hardware-setup



4.2. 硬件组成及系统逻辑结构图 （硬件分布 各个模块的功能 目前实验控制的架构软件控制模块的功能介绍等）

4.2.1.核心板 Intel® Joule™ SBC（single board computer）

Intel 对开发套件市场来说并不陌生，Joule 作为新的单板计算机产品优点为外形紧凑，处理性能强大，其微型模块上系统 (SOM) 只有 24mm x 48mm 大小，但却囊括了众多功能，如支持 Intel® RealSense™ 技术的 4K 视频捕获和显示功能，该产品允许设备捕获景深 (DOF) 信息，让应用对环境及其中的任何对象都有一个空间感，我们很容易就能想象出

它在以下领域的作用多么强大：自动驾驶车辆、无人机和机器人、汽车行业的驾驶员辅助应用、增强现实和虚拟现实领域、工业物联网...以及实现对当地环境的三维空间感认识会非常有用的许多其他应用。

Intel® Joule™ 570X 模块的功能：

- 高性能 64 位 1.7 GHz 四核 Intel® Atom™ T5700 处理器，突发频率可增强至 2.4 GHz
- 4GB LPDDR4 RAM 和 16GB eMMC 内存
- Intel® 高清显卡，支持 4K 视频捕获和显示
- 支持 MIMO 和蓝牙 4.1 的 802.11ac Wi-Fi
- USB 3.0、MPI* CSI 和 DSI 接口，以及多个 GPIO、I2C 和 UART 接口
- 基于 Linux 的操作系统，专为物联网和智能设备而定制
- 增强对 Intel® RealSense 摄像头和数据库的支持

4.2.2.控制板 OpenCR

OpenCR1.0 是 TurtleBot3 的主控制器板， OpenCR1.0;针对 ROS 的开源控制模块，是针对 ROS 嵌入式系统开发的，提供完全开源的硬件和软件、一切关于控制板、原理图，PCB 、BOM 和 TurtleBot3 的固件源代码等资料可以根据用户和 ROS 社区的开源许可证免费获取。

STM32F7 系列是 OpenCR1.0 开发板中的主要芯片，该开发板基于功能强大的带有浮点单元的 ARM Cortex-M7， OpenCR1.0 的开发环境从 Arduino IDE 和 Scratch 开放，适合年轻学生使用专家的固件开发。

OpenCR1.0 提供数字和模拟输入/输出引脚，可以连接扩展板或各种传感器，此外，OpenCR1.0 还具有各种通信接口：用于连接 PC，UART，SPI，I2C 和 USB 的 USB，用于其他嵌入式设备。

使用 SBC 时，OpenCR1.0 可以提供最佳的解决方案。它支持 SBC 和传感器的 12V，5V，3.3V 电源输出。它还支持电池和 SMPS 之间的热插拔电源输入。

4.3. 远程电脑的软件环境配置

4.3.1.操作系统 Ubuntu16.04LTS

ROS 在 Linux 环境下运行，目前使用的版本号为 Ubuntu16.04LTS。通过官方网站下载系统安装镜像之后根据指导手册进行安装。

指导手册链接：https://www.ubuntu.com/download/desktop/install-ubuntu-desktop

4.3.2.在 Ubuntu 中安装 ROS。

声明：使用的 Ubuntu 版本为 16.04(Xenia), ROS 使用的版本号为 Kinetic。

- 设定 sources.list

将电脑设置为能够接受 ROS 包的状态。首先打开 linux 的命令窗口界面（ctrl+alt+T）键入：

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

- 设置 ROS 密钥

```
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
```

- 安装 ROS

首先确定系统组件为最新版本

```
Sudo apt-get update
```

在 ROS 中有很多不同的库和工具，在开始阶段有几种不用的默认安装设置可以选择，并可以分开安装 ROS 的不同组件，我们在这个阶段可以选择全组件完整安装（推荐）。在全组件安装中，软件包包含了 ROS、rqt（机器人调试界面）、rviz（机器人可视化操作界面）、机器人通用库、2D/3D 模拟器、导航和 2D/3D 感知组件。

```
sudo apt-get install ros-kinetic-desktop-full
```

安装过程中可能会遇到中断，一般是由于网络问题导致，只需要重新键入命令即可。

在全组件包安装完成之后，用户可以根据自身需求单独安装其他的组件，搜索可用组件，可以使用搜索命令，根据搜索得到的包名称进行安装：

```
apt-cache search ros-kinetic

sudo apt-get install ros-kinetic-PACKAGE
```

e.g.

```
sudo apt-get install ros-kinetic-slam-gmapping
```

- 初始化 rosdep

在开始使用 ROS 之前，需要初始化运行 rosdep，可以让用户轻松地为要变异的源安装系统依赖项，并且在 ROS 某些核心组件中也需要该程序。

```
sudo rosdep init

rosdep update
```

- 建立 ROS 运行环境

建立运行环境的目的在于使得每一次运行新的程序时，ROS 环境自动地从已经建立好的位置开始运行。设置的方法：

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc   \\将位置写入到 ROS 位置表中

source ~/.bashrc                                        \\确认位置表
```

- 构建包的依赖关系

截至目前已经安装好运行核心 ROS 包的各种组件，为了能够自己创建和管理 ROS 工作空间，还可以另外分别安装其他你需要的各种工具。比如，rosinstall 就是在你需要能快捷安装各种 ROS 包时候使用的命令行工具。

为了安装这些工具和构建 ROS 包的依赖关系，运行

```
sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

至此便完成了 ROS 的安装过程，通过一步步的操作已经在电脑中建立了完备的 ROS 工具组，在这之后便可以自己安装需要的 ROS 开源库进行更高级的操作了。

4.3.3. 安装 Turtlebot3 特定开源库

目前我们已经有了完整的 ROS 工具，在这之上 turtlebot 是已经组装好的产品化的机器人，我们需要调用由开发者提供的开源库进行控制。开源库的作用便是建立了 ROS 工具和机器人的特定映射关系，下面安装 Turtlebot3 控制使用的依赖包（dependent packages）

```
sudo apt-get install ros-kinetic-joy ros-kinetic-teleop-twist-joy ros-kinetic-teleop-twist-keyboard ros-kinetic-laser-proc ros-kinetic-rgbd-launch ros-kinetic-depthimage-to-laserscan ros-kinetic-rosserial-arduino ros-kinetic-rosserial-python ros-kinetic-rosserial-server ros-kinetic-rosserial-client ros-kinetic-rosserial-msgs ros-kinetic-amcl ros-kinetic-map-server ros-kinetic-move-base ros-kinetic-urdf ros-kinetic-xacro ros-kinetic-compressed-image-transport ros-kinetic-rqt-image-view ros-kinetic-gmapping ros-kinetic-navigation ros-kinetic-interactive-markers
```

在系统安装完所有依赖包之后，创建一个工作空间

```
$ cd ~/catkin_ws/src/                    \\创建文件夹
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git  \\从 github 上加载软件包
$ cd ~/catkin_ws && catkin_make      \\进入文件夹 使用 make 命令
```

如果运行 catkin_make 之后，没有报错则说明 Turtlebot3 的运行环境已经设置好了。

4.3.4. 网络设置

ROS 系统的正常运行需要远程主机和机器人可以进行双向通信，在前面的网络环境介绍中已提到，我们需要用 IP 地址来进行节点识别。

首先将远程主机和机器人接入同一无线局域网内，此时可以通过以下命令查看分配到的网络 IP 地址

```
ifconfig
```

此时得到

```
wlp2s0    Link encap:Ethernet  HWaddr ac:2b:6e:6d:08:ee
          inet addr:192.168.0.100  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::77a:7d5c:9ca8:bd9c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:468 errors:0 dropped:0 overruns:0 frame:0
          TX packets:630 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:107986 (107.9 KB)  TX bytes:89522 (89.5 KB)
```

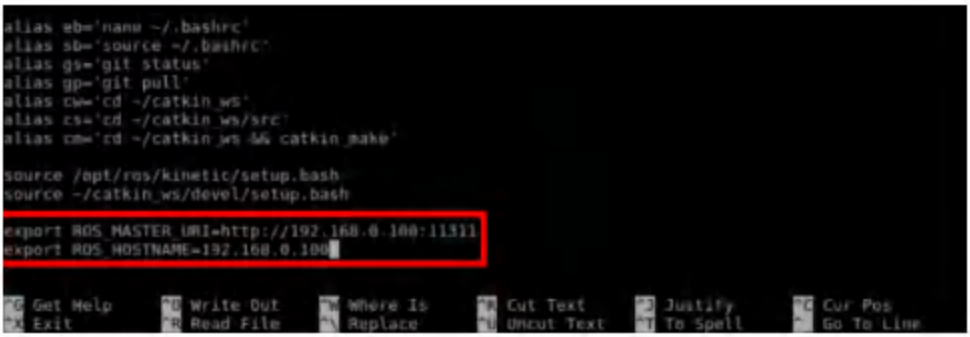
此时键入以下命令：

```
nano ~./bashrc
```

然后按‘ alt+/ ‘ 滑到列表最下面，对本地主机（localhost）地址进行编辑，将上一步得到的 IP 地址写入主机地址行，界面如下图所示，其中，

```
export ROS_MASTER_URI=http:// 192.168.0.11 : 11311  \\11311 为 ROS 环境默认主机代号
```

```
export ROS_HOSTNAME = 192.168.0.11
```



关闭窗口之后，在终端输入命令以确认设置

```
source ~./bashrc
```

4.4. Turtlebot3 的软件配置方法

实验前已经为机器人配置了 linux 系统环境，其系统版本为 Alternative Ubuntu 16.04 for Intel® Joule™，安装系统后，在 Ubuntu 下安装 ROS，具体流程与在电脑上安装 ROS 一样，此处不作介绍。

4.4.1. 安装 Turtlebot3 依赖包及 USB 设置

与远程电脑的区别在于移动机器人上安装了不同的依赖包，用来控制不同的底层组件，其代码如下：

```
cd ~/catkin_ws/src

git clone https://github.com/ROBOTIS-GIT/hls_lfcd_lds_driver.git

git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git

git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
```

然后删除了几个在 PC 端使用的依赖包

```
cd ~/catkin_ws/src/turtlebot3

sudo rm -r turtlebot3_description/ turtlebot3_teleop/ turtlebot3_navigation/ turtlebot3_slam/ turtl
ebot3_example/
```

安装 Turtlebot 用依赖包

```
sudo apt-get install ros-kinetic-rosserial-python ros-kinetic-tf
```

然后重启 Joule 之后构建包

```
cd ~/catkin_ws && catkin_make
```

USB 设置使得 OpenCR1.0 不需要根权限也能使用 USB 端口

```
rosrun turtlebot3_bringup create_udev_rules
```

注：至此的操作只需要在第一次配置机器人时需要，完成操作后不需要更改依赖包等设置。

4.4.2. 网络设置

同样使用以下命令行得到机器人分配到的网络 IP 地址

```
ifconfig
```

根据得到的网络地址，同样是打开 bashrc 表单

```
nano ~/.bashrc
```

在最后两行中 MASTER_URI 输入远程主机的 IP 地址，在 HOSTNAME 中输入机器人的 IP 地址，这样便在机器人中设置好了网络名称。

同样，在完成修改之后需呀确定修改表单

```
source ~/.bashrc
```

注：每一次进行试验或接入均需要检查

5. 实验准备

5.1. 预备知识

在实验中，电脑的操作环境为 LINUX 系统，Linux 是一套免费使用和自由传播的类 Unix 操作系统，是一个基于 POSIX 和 UNIX 的多用户、多任务、支持多线程和多 CPU 的操作系统。Linux 同时具有字符界面和图形界面，在字符界面用户可以通过键盘输入相应的指令来进行操作，在图形界面用户也可以使用鼠标进行操作。实验之前，同学需要自行熟悉 Linux 的终端操作方法和基本命令的读识。

5.2. 硬件知识

实验之前了解 SBC 和控制板以及驱动轮的信息，其规格见附录 2。

实验步骤详解

实验一：机器人连接实验

实验目的

当准备工作做好之后，我们便可以开始着手于对机器人的控制了，首先需要做的是建立远程主机和 Tultlebot3 的网络连接，最终达到能双向通讯的效果。

实验步骤参考：

1.准备好实验环境

- 网络：实验环境需要在联网的情况下进行,且需要连接相同的无线网络。这样做的目的在于 ROS 系统在使用过程中移动机器人需要访问并传递信息到远程电脑建立的程序核心服务器(roscore),信息不经过互联网所以只能在局域网中进行连接。
- 控制：Turtlebot 开始运行前需要连接显示屏，(roslaunch)启动模块控件。
- 空间：无随机障碍环境。
- 电源：调试过程请使用 12V 电源转换器连接，使用 SLAM 实验使用电池。同时使用时拔掉转换器会引起故障，建议调试完成后只接电池重启运行。

2.建立连接前准备

首先在远程电脑上进行测试，在开始连接之前需要确定：

- 本代码在远程电脑上运行，如果在 Turtlebot 上输入代码请不要运行 roscore 程序
- 确保远程主机和机器人 IP 网络地址设置正确
- 当电池电量低于 11V 时，电池会发出持续的警报声，而且部分控件无法工作，此时应该将电池充电。

在准备工作做好之后，建议将 bashrc 表单中设定机器人型号名称，正确的设定在后续可视化操作中才能调用正确的三维模型。

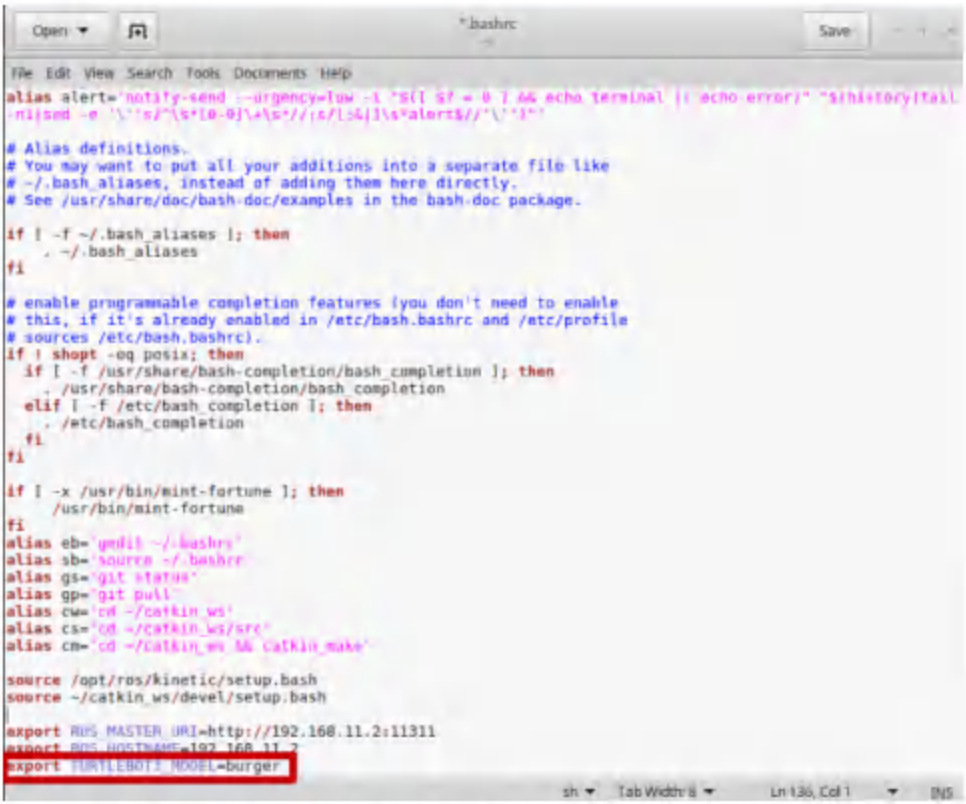
调用更改 表单命令行：

```
gedit ~/.bashrc
```

这样能使用 gedit 工具修改表单，如果无法正确调用表单需要在命令之前加上 sudo 以提供根权限。

修改为：

```
export TURTLEBOT3_MODEL=waffle
```



同样，在每一次修改 bashrc 表单之后都需要重复操作确认表单命令已完成更新。

```
source ~/.bashrc
```

Bringup_主机设置 [PC]

在远程主机终端启动程序，使用 ROS 的第一步就是运行：

```
roscore
```

此时你会看到以下代码：

```
... logging to ~/.ros/log/9cf88ce4-b14d-11df-8a75-00251148e8cf/roslaunch-machine_name-1303
9.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
started roslaunch server http://machine_name:33919/
ros_comm version 1.4.7


SUMMARY
======


PARAMETERS
 * /rosversion
 * /rosdistro
NODES
auto-starting new master
process[master]: started with pid [13054]
ROS_MASTER_URI=http://machine_name:11311/
setting /run_id to 9cf88ce4-b14d-11df-8a75-00251148e8cf
process[rosout-1]: started with pid [13067]
started core service [/rosout]
```

以上代码表述了从加载程序文件然后计算磁盘和内存空间，建立了一个尾号为 11311 的主服务器地址，并开始服务。

如果 roscore 没有成功初始化，说明网络设置有问题，可以回到网络设置说明检查主机地址设置是否正确。

Bringup_Waffle 设置 [Waffle]

在机器人桌面终端调用 Turtlebot3 应用程序

```
roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

注，此处出现常见问题见本节错误提示[Error-2]

如果想单独调用激光传感器（lide sensor），Intel® RealSense™ R200 和核心驱动程序，可以分别打开三个终端运行以下命令：

```
$ roslaunch turtlebot3_bringup turtlebot3_lidar.launch

$ roslaunch turtlebot3_bringup turtlebot3_realsense.launch

$ roslaunch turtlebot3_bringup turtlebot3_core.launch
```

注：在使用 Realsense 相机之前请阅读相关说明 Intel® RealSense™

注：如果终端提示 lost sync with device 说明传感器没有连接良好。

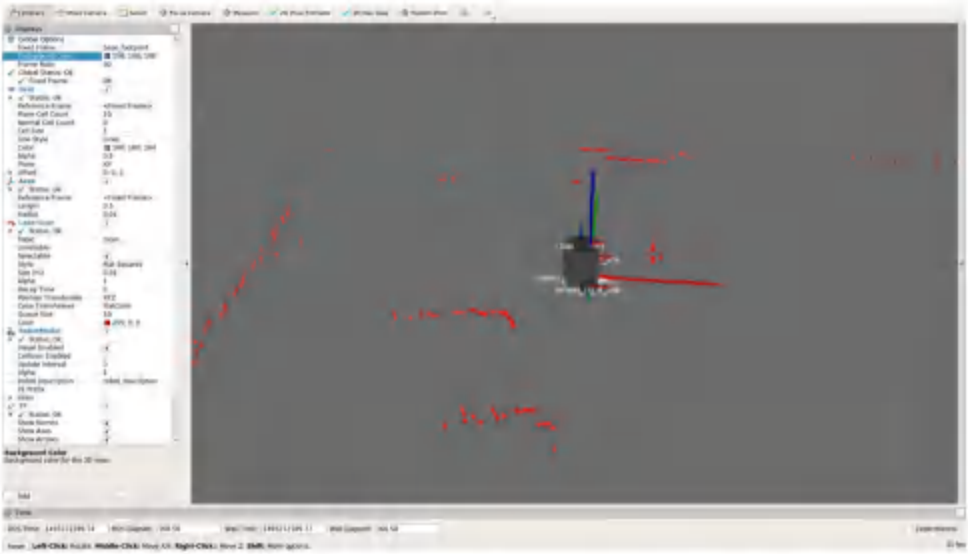
在命令运行良好的情况下，可以观察到终端信息不断刷新，传感器信号被成功传输到主服务器上，此时便完成了远程主机和机器人的成功连接。

运行可视化软件 RViz [PC]

Rviz 是 ROS 环境下的 3D 可视化工具，通过这个工具可以在电脑中模拟机器人在世界中的位置，并通过读取传感器数据获得虚拟环境信息。

首先声明调用的模型为 waffle，然后启动 Turtlebot 远程应用，启动 Rviz。

```
$ export TURTLEBOT3_MODEL=waffle

$ roslaunch turtlebot3_bringup turtlebot3_remote.launch

$ rosrun rviz rviz -d `rospack find turtlebot3_description`/rviz/model.rviz
```



此时可以在程序窗口观察到调用的 3D 模型，窗口内出现的红点即为激光传感器的探测信息，描述了周围环境的光学反射点，通过对应的观察可以识别出椅子墙壁等物件。

使用遥控控制机器人 [PC]

Turtlebot 提供了多种遥控方案，如 Keyboard / Andriod /Xbox /Ps 手柄.....在完成连接的基础上，我们可以通过远程主机的键盘对小机器人进行运动控制。在开始运行之前请保证机器人在宽敞的平面上运行，如果在桌面上运行请确保机器人不会从桌面上掉落。

这里我们首先在 PC 上运行无线操作依赖包中的键盘驱动程序。

```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

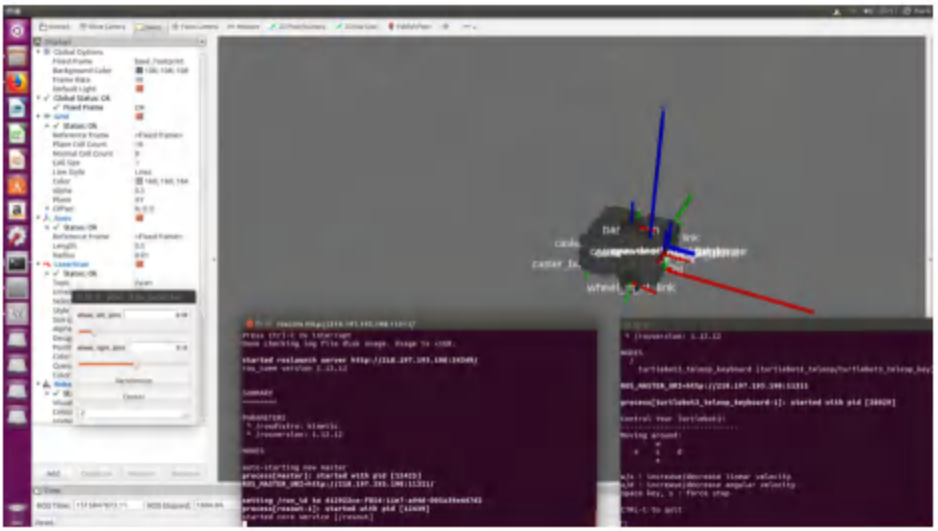
如果程序成功运行，终端界面会如下图所示：

```
Control Your Turtlebot3!
---------------------------
Moving around:
        w
   a    s    d
        x

w/x : increase/decrease linear velocity
a/d : increase/decrease angular velocity
space key, s : force stop

CTRL-C to quit
```

注:本命令常见问题请查阅[Error-2]

注意，在键盘控制过程中可以发现，键盘设定的为两个驱动轮的角速度，所以在实际控制当中，改变方向之后小车并不会往这个方向直行而是继续在路径切线上左偏或右偏，这样的运行效果是小车采用绕圈的行动方式。

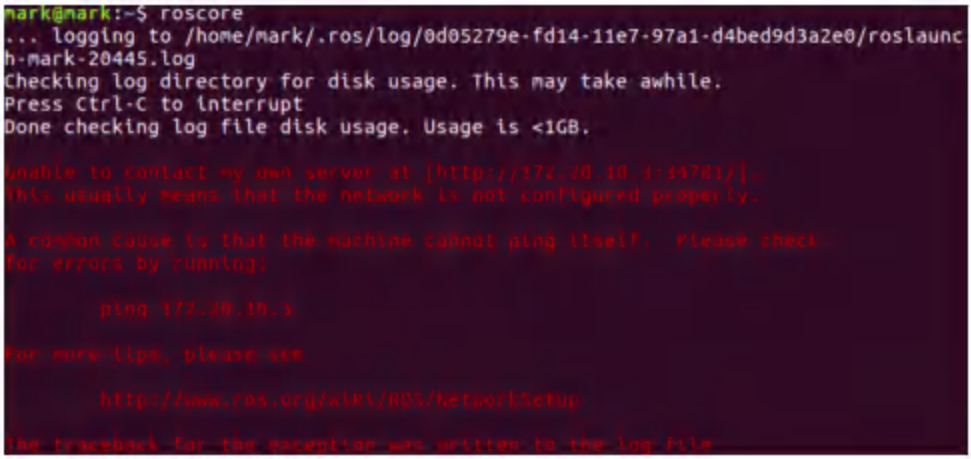
在小车运行过程中，上一小节中运行的激光距离传感器也同时在采集数据，运行时产生的点云扫描结果如下 图所示。



实验常见问题：

[ERROR-1]

```
~$ roscore
```



提示无法建立服务器连接;

解决办法:

第一次建立网络连接请确定主机和移动机器人服务器地址设定正确，设定网络连接命令:

```
~$ sudo gedit ~/.bashrc
```

注: 更改索引文件 在设定主机 HOST 地址填入正确的网络地址

(ifconfig 查看远程电脑网络地址 具体操作请重做 设定ROS Network 步骤)

```
~$ source ~/.bashrc
```

%%重设索引文件

[ERROR-2]

```
~$ roslaunch teleop_twist_joy teleop.launch
```

OR

```
~$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

出现 rospack 无法找到的情况，首先检查拼写。拼写无误情况下出现错误提示请运行:

```
~$ source ~/catkin_ws/devel/setup.bash
```

重新加载 catkin_ws 工作空间设置文件，并查找 ROS 依赖包，比如 bringup 包

```
~$ rospack find turtlebot3_bringup
```

终端会返回 rospack 地址，表示已经找到依赖包。

这个时候再运行

```
~$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

命令才可以正确启动。

[Error 待补充]

实验二 远程操作和 SLAM

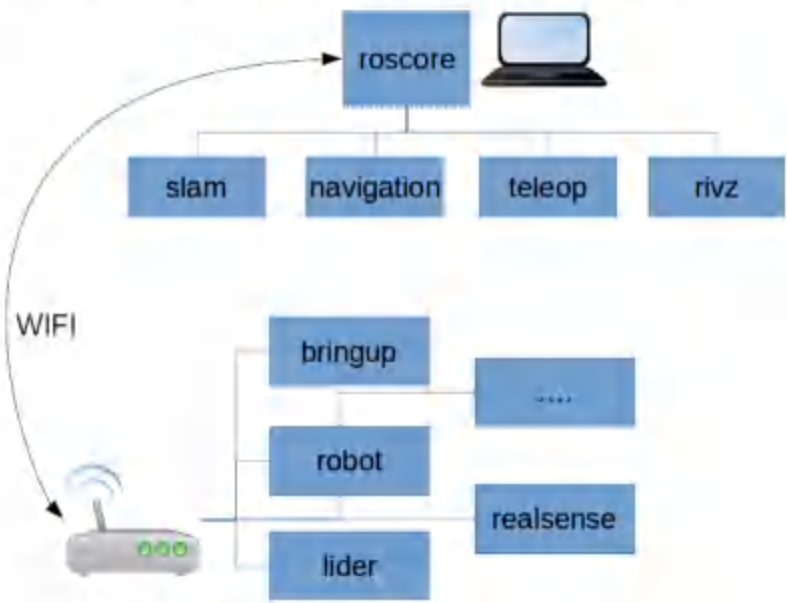
实验准备:

同实验一，并将远程主机和机器人运行至正常连接并启动传感器。

实验步骤

SLAM (simultaneous localization and mapping),也称为 CML (Concurrent Mapping and Localization), 即时定位与地图构建，或并发建图与定位。问题可以描述为：将一个机器人放入未知环境中的未知位置，是否有办法让机器人一边逐步描绘出此环境完全的地图，所谓完全的地图（a consistent map）是指不受障碍行进到房间可进入的每个角落。SLAM 最早由 Smith、Self 和 Cheeseman 于 1988 年提出。由于其重要的理论与应用价值，被很多学者认为是实现真正全自主移动机器人的关键。

Turtlebot3 相对于前代机器人的最知名的功能就是加入了 SLAM 模块，本实验旨在使用提供的 SLAM 程序了解和使用 Waffle 机器人的地图实时构建功能。



上图表示了在 SLAM 和 Navigation 实验中所需要的主要程序模块及结构，其中 roscore 创建连接服务器，Turtlebot 建立连接用以交换数据。建立连接完成之后 turtlebot 内启动激光雷达等部件，视使用需求使用 robot 总装模块和 lider core 等独立模块。

SLAM 实验[PC]

建立连接之后进入 SLAM 程序

首先声明使用的模型为 waffle 机器人，并启动 SLAM 例程。

```
$ export TURTLEBOT3_MODEL=waffle

$ roslaunch turtlebot3_slam turtlebot3_slam.launch
```

此时可以观察到终端窗口能够实时刷新传感器发送过来的数据。此时可以在 RViz 中打开模拟界面。

```
$ rosrun rviz rviz -d `rospack find turtlebot3_slam`/rviz/turtlebot3_slam.rviz
```

并启动键盘控制（或手柄控制，如果条件允许）

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

保证移动机器人有充足控件运行的前提下，通过键盘控制机器人运动，此时可以从 RViz 程序界面观察到由激光传感器发送过来的环境信息正在随着机器人位置的移动逐步建立了环境的外形，并通过多个位置的测量使环境信息更加丰富准确。

完成地形测量之后可以得到扫描地形图一个，打开新的终端，输入保存地图命令：

```
$ rosrun map_server map_saver -f ~/map
```

此时在主文件夹中 /home/<username> 会出现保存到的地图文件 map.pgm 和 map.yaml

下图为实例测量 C210 办公室的地形图，可以观察到比较明确的办公室布置。

注：扫描过程中需要注意不要使机器人与环境产生接触，不要人为在机器人附近活动，不要将机器人速度提到太快，太快的移动速递会导致环境探测信号量减少。按照规范操作，以便得到正确的地形数据。

自动导航 Navigation

完成地形探测之后可以调用自动导航功能。导航功能的原理是利用已经测量好的实验地形，将机器人与原初始点相对位置和姿态确定好之后，将自然条件下的实时探测行进问题转变为在虚拟空间中对目标位置进行轨迹规划的问题。

在开始导航实验之前确认已经运行过 Bringup 任务，同样在新的终端中声明使用的机型 waffle

```
$ export TURTLEBOT3_MODEL=waffle

$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```

以上程序便完成了读取在上一节中测量好的地图数据 map.yaml

接下来启动 RViz 可视化程序执行导航任务。

```
$ rosrun rviz rviz -d `rospack find turtlebot3_navigation`/rviz/turtlebot3_nav.rviz
```

在开始导航之前，RViz 需要确定机器人的初始位置。初始位置的正确建立即产生了正确的从实体空间到地图空间的映射，校准方法如下：

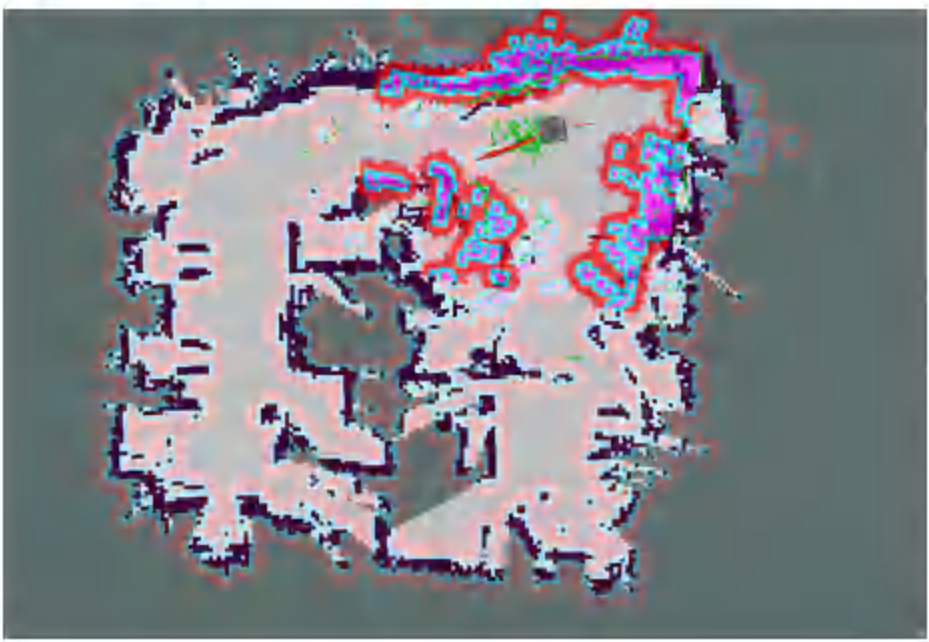
- 点击工具栏 2D Pose Estimate 按键。
- 在地图上点击机器人相对于环境的正确位置并拖动箭头以确定机器人的姿态，可以通过多次调整确定最合适的初始位置设定。

完成设定后的绿色箭头便表示了机器人相对于地图所处的位置和初始姿态，机器人上的激光传感器同时会测量周围环境，可以通过实时测量的环境信息对初始位置进行校对（红点），如果发现位置不正确可以重复定位操作。

完成定位工作之后，需要设定目标位置和目标姿态，设定方法如下：

- 点击工具栏 2D Nav Goal 按键
- 点击一个确定的点并拖动箭头以设定目标位置和机器人最后停止的方向。

实验过程中要求操作者确定好初始位置和目的位置及姿态之后，机器人便会自行计算运行路径并移动至目标点。



提问：根据观察推断机器人路径规划的大致计算方法？

实验常见问题

本次实验建立在完整的连接和已有的控制命令中，不需要操作者参与计算，操作较为简单，可以在熟悉操作的情况下没有障碍的完成时实验。

主要会出现的情况有：如果没有确定好初始位置开始进行扫描测量地形，其地图初始点未知，在导航实验中如果不能从原出发点开始运行，会产生地图定位不准的问题，所以在一定程度上也会影响到内部程序对轨迹规划的运算，因为在导航功能中小车的轨迹规划是同时计算原测量地图和实时雷达扫描障碍物的最优解。同时，在实验中杂乱的周边环境对导航功能也会产生较大影响。所以总的来说，小车的运行原理和计算能力对实验场地提出了一定要求。

实验三 模拟控制和 GAZEBO 虚拟世界构建

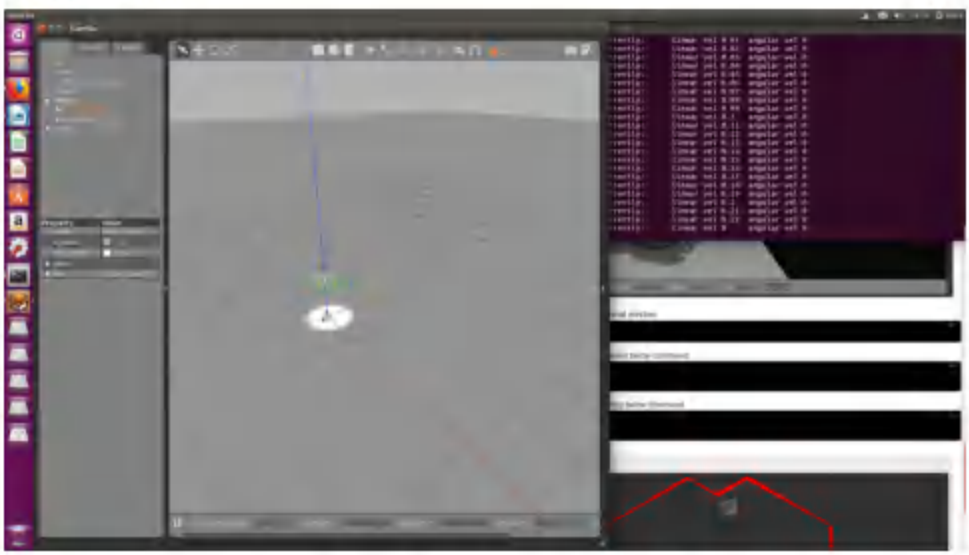
实验准备

本实验基于 ubuntu 系统的 Gazebo 模拟软件，在电脑上运行模拟程序。只需要远程电脑一台和正确的网络连接环境。

Simulation 步骤参考：

http://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#gazebo-3d

Gazebo 软件介绍：Ubuntu 环境下的模拟软件，可以建立空间和实体，使用 turtlebot 提供的调用程序可以使用键盘控制模拟机器人在模拟空间中的运行和探测。其软件界面和操作效果如下图所示：



首先需要做的事下载克隆模拟依赖包，一下命令便描述了安装包应该下载的位置和来源，最后经过构建程序（catkin_make）完成安装。

```
$ cd ~/catkin_ws/src/

$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git

$ cd ~/catkin_ws && catkin_make
```

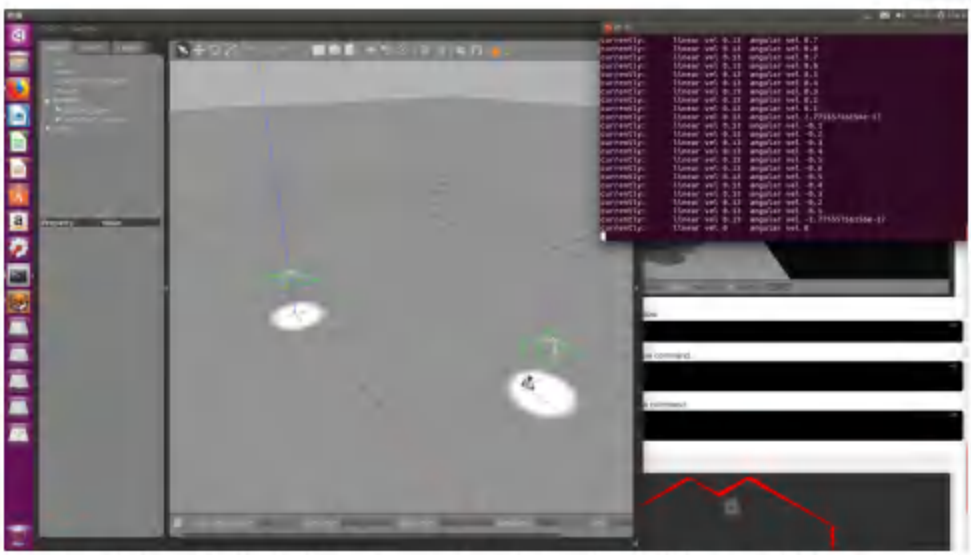
这里使用的方法是在 ROS 中建立一个虚假节点 TurtleBot3 fake node ，不需要实体机器人也可以进行通信和控制，你甚至可以通过无线控制节点在 RViz 中对虚拟机器人进行控制.

同样在运行之前先声明需要加载的 3D 模型

```
$ export TURTLEBOT3_MODEL=burger

$ roslaunch turtlebot3_fake turtlebot3_fake.launch
```

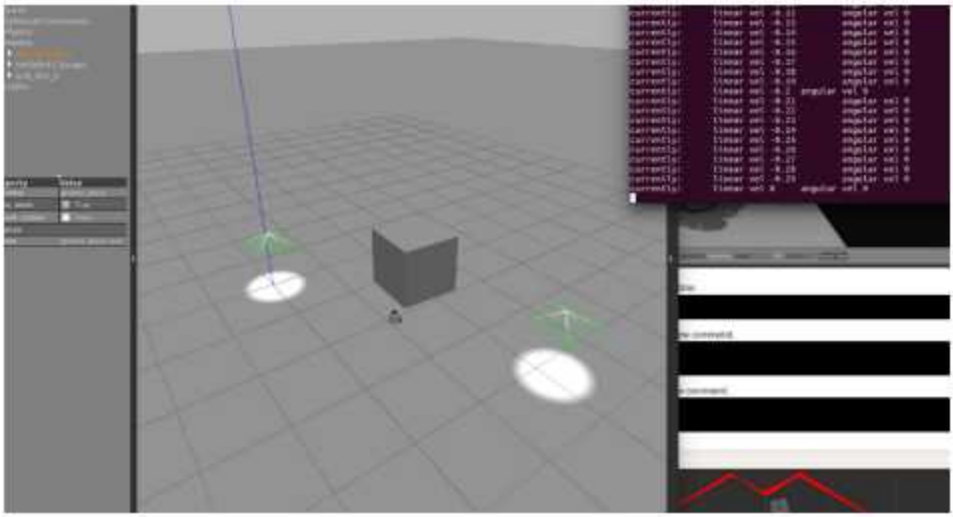
然后建立无线控制节点

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```



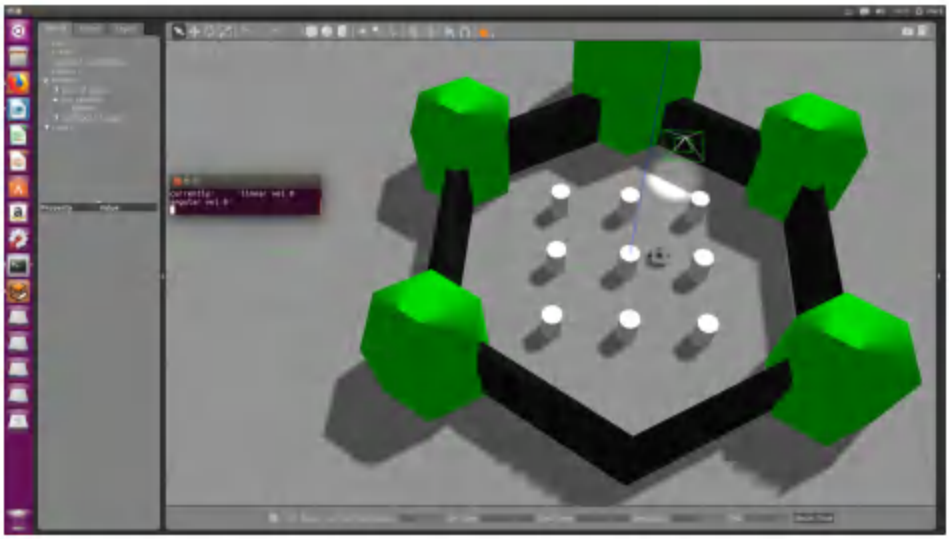
如上图所示，使用键盘控制程序可以操作小车在已建立好的世界平面内进行运动。模拟效果相当直观简洁，控制操作同操作实体一样，而且在速度和运动空间上有更大自由，能达到较高的速度，同时还可以自己建立虚拟实体（如下图所示），且可以设定实体大小位置质

量固定性等参数，模拟小车与实体接触的不同情况。



如下图所示，在软件中可以建立空白空间或者调用已经建立好的运动空间：

```
export TURTLEBOT3_MODEL=waffle

roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch

或者

roslaunch turtlebot3_gazebo turtlebot3_world.launch
```



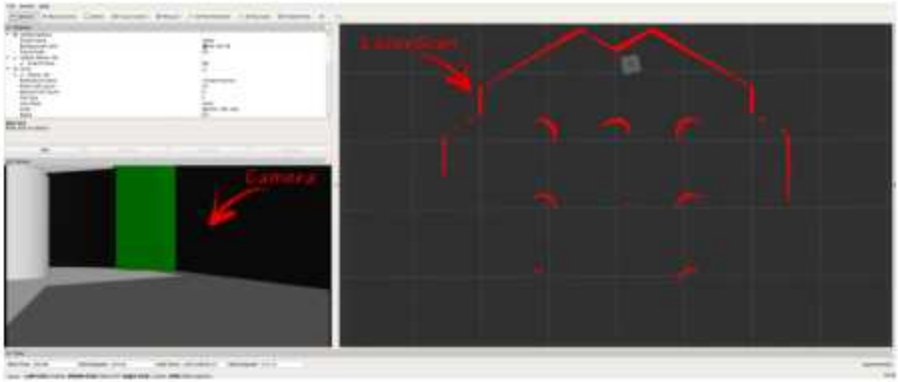
为了能够控制机器人同样需要在新的终端中创建使用键盘控制的远程控制节点

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

在建立好的控件里面，turtlebot 已经提供了软件接口，可以只接调用键盘控制程序控制 Gazebo 软件下的模拟器，对模拟环境下的使用键盘进行控制。

还可以调用已经提供的命令接入模拟的激光传感器进行探测环境的点云可视化。模拟环境下的激光雷达探测环境界面如下图所示

```
$ export TURTLEBOT3_MODEL=waffle

$ roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch
```

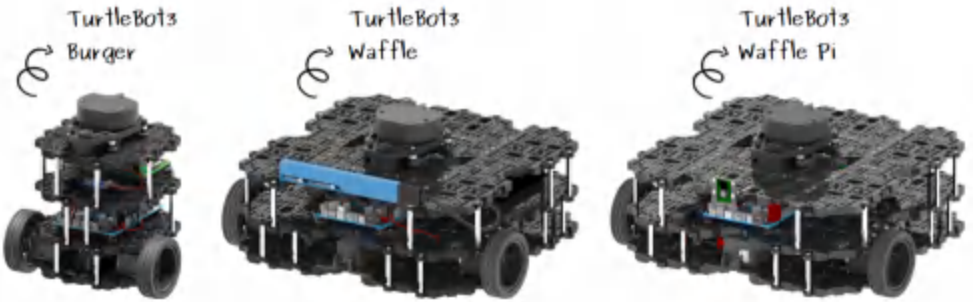


同时如果需要在虚拟环境中实现导航功能可以在新的终端中输入命令：

```
$ export TURTLEBOT3_MODEL=waffle

$ roslaunch turtlebot3_gazebo turtlebot3_simulation.launch
```

模拟器具有良好的开放性，可以自己建立运行环境对不同环境进行模拟，可以很好的扩展机器人操作的视野，对解决真实环境下的机器人探测导航等问题提供生动具体的认识。

附录 1



Items	Burger	Waffle	Waffle Pi
Maximum translational velocity	0.22 m/s	0.26 m/s	0.26 m/s
Maximum rotational velocity	2.84 rad/s (162.72 deg/s)	1.82 rad/s (104.27 deg/s)	1.82 rad/s (104.27 deg/s)
Maximum payload	15kg	30kg	30kg
Size (L x W x H)	138mm x 178mm x 192mm	281mm x 306mm x 141mm	281mm x 306mm x 141mm
Weight (+ SBC + Battery +	1kg	1.8kg	1.8kg

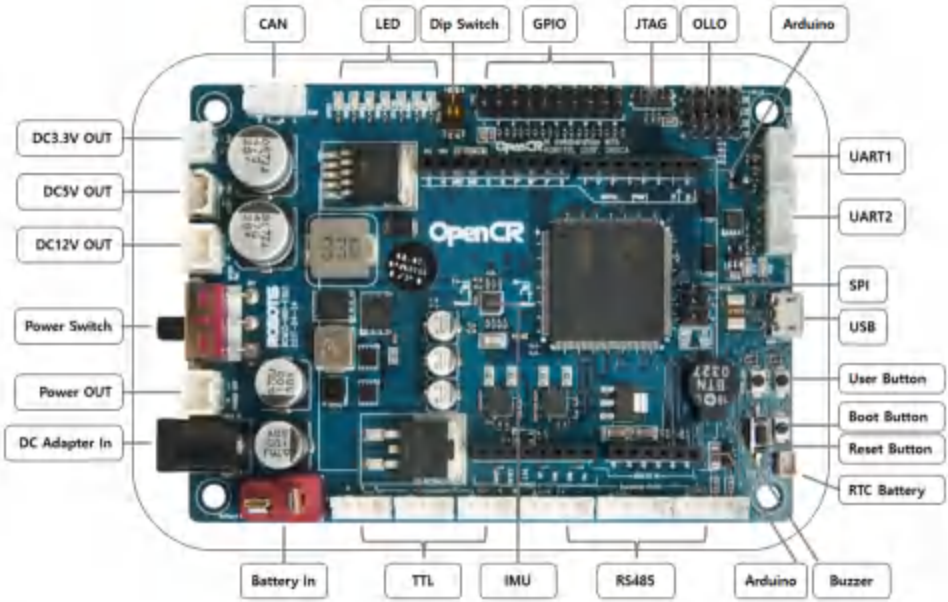
Items	Burger	Waffle	Waffle Pi
Sensors)			
Threshold of climbing	10 mm or lower	10 mm or lower	10 mm or lower
Expected operating time	2h 30m	2h	2h
Expected charging time	2h 30m	2h 30m	2h 30m
SBC (Single Board Computers)	Raspberry Pi 3 Model B	Intel® Joule™	Raspberry Pi 3 Model B
MCU	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)
Remote Controller	-	-	RC-100B + BT-410 Set (Bluetooth 4, BLE)

Items	Burger	Waffle	Waffle Pi
Actuator	Dynamixel XL430-W250	Dynamixel XM430-W210	Dynamixel XM430-W210
LDS(Laser Distance Sensor)	360 Laser Distance Sensor LDS-01	360 Laser Distance Sensor LDS-01	360 Laser Distance Sensor LDS-01
Camera	-	Intel® Realsense™ R200	Raspberry Pi Camera Module v2.1
IMU	Gyroscope 3 Axis Accelerometer 3 Axis Magnetometer 3 Axis	Gyroscope 3 Axis Accelerometer 3 Axis Magnetometer 3 Axis	Gyroscope 3 Axis Accelerometer 3 Axis Magnetometer 3 Axis
Power connectors	3.3V / 800mA 5V / 4A 12V / 1A	3.3V / 800mA 5V / 4A 12V / 1A	3.3V / 800mA 5V / 4A 12V / 1A

Items	Burger	Waffle	Waffle Pi
Expansion pins	GPIO 18 pins Arduino 32 pin	GPIO 18 pins Arduino 32 pin	GPIO 18 pins Arduino 32 pin
Peripheral	UART x3, CAN x1, SPI x1, I2C x1, ADC x5, 5pin OLLO x4	UART x3, CAN x1, SPI x1, I2C x1, ADC x5, 5pin OLLO x4	UART x3, CAN x1, SPI x1, I2C x1, ADC x5, 5pin OLLO x4
Dynamixel ports	RS485 x 3, TTL x 3	RS485 x 3, TTL x 3	RS485 x 3, TTL x 3
Audio	Several programmable beep sequences	Several programmable beep sequences	Several programmable beep sequences
Programmable LEDs	User LED x 4	User LED x 4	User LED x 4
Status LEDs	Board status LED x 1 Arduino LED x 1 Power LED x 1	Board status LED x 1 Arduino LED x 1 Power LED x 1	Board status LED x 1 Arduino LED x 1 Power LED x 1
Buttons and	Push buttons	Push buttons	Push buttons

Items	Burger	Waffle	Waffle Pi
Switches	x 2, Reset button x 1, Dip switch x 2	x 2, Reset button x 1, Dip switch x 2	x 2, Reset button x 1, Dip switch x 2
Battery	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C
PC connection	USB	USB	USB
Firmware upgrade	via USB / via JTAG	via USB / via JTAG	via USB / via JTAG
Power adapter (SMPS)	Input : 100–240V, AC 50/60Hz, 1.5A @max Output : 12V DC, 5A	Input : 100–240V, AC 50/60Hz, 1.5A @max Output : 12V DC, 5A	Input : 100–240V, AC 50/60Hz, 1.5A @max Output : 12V DC, 5A

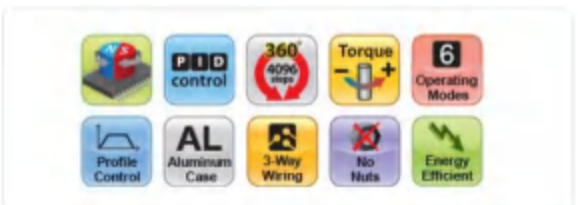
附录 2



Items	Specifications
Microcontroller	STM32F746ZGT6 / 32-bit ARM Cortex®-M7 with FPU (216MHz, 462DMIPS)
Sensors	Gyroscope 3Axis, Accelerometer 3Axis, Magnetometer 3Axis (MPU9250)
Programmer	ARM Cortex 10pin JTAG/SWD connector USB Device Firmware Upgrade (DFU) Serial
Extension pins	32 pins (L 14, R 18) *Arduino connectivity

Items	Specifications
	Sensor module x 4 pins Extension connector x 18 pins
Communication circuits	USB (Micro-B USB connector/USB 2.0/Host/Peripheral/OTG) TTL (B3B-EH-A / Dynamixel) RS485 (B4B-EH-A / Dynamixel) UART x 2 (20010WS-04) CAN (20010WS-04)
LEDs and buttons	LD2 (red/green) : USB communication User LED x 4 : LD3 (red), LD4 (green), LD5 (blue) User button x 2
Powers	External input source 5 V (USB VBUS), 7-24 V (Battery or SMPS) Default battery : LI-PO 11.1V 1,800mAh 19.98Wh Default SMPS: 12V 5A External output source 12V@1A(SMW250-02), 5V@4A(5267-02A), 3.3V@800mA(20010WS-02) External battery Port for RTC (Real Time Clock)

Items	Specifications
	(Molex 53047-0210) Power LED: LD1 (red, 3.3 V power on) Reset button x 1 (for power reset of board) Power on/off switch x 1
Dimensions	105(W) X 75(D) mm
Mass	60g



Items	XL430-W250 (for Burger)	XM430-W210 (for Waffle)
Microcontroller <td colspan=2> ST CORTEX-M3		

Items	XL430-W250 (for Burger)	XM430-W210 (for Waffle)
(STM32F103C8 @ 72Mhz, 32bit) </td>		
Position Sensor <td colspan=2> Contactless Absolute Encoder (12bit, 360°) </td>		
Motor	Cored Motor	Coreless Motor
Baud Rate	9600 bps ~ 4.5 Mbps	
Control Modes	Velocity, Position, Extended Position, PWM	Velocity, Position, Extended Position, PWM, Current, Current-base Position
Gear Ratio	258.5 : 1	212.6 : 1
Stall Torque	1.0 N.m (@ 9V, 1A)	2.7 N.m (@ 11.1V, 2.1A)

Items	XL430-W250 (for Burger)	XM430-W210 (for Waffle)
	1.4 N.m (@ 11.1V, 1.3A)	3.0 N.m (@ 12V, 2.3A)
	1.5 N.m (@ 12V, 1.4A)	3.7 N.m (@ 14.8V, 2.7A)
No Load Speed	47rpm (@ 9V)	70rpm (@ 11.1V)
	57rpm (@ 11.1V)	77rpm (@ 12V)
	61rpm (@ 12V)	95rpm (@ 14.8V)
Communication	TTL Level Multi Drop Bus	TTL Level / RS485 Multi Drop Bus
Material	Engineering Plastic	Full Metal Gear, Metal Body, Engineering Plastic
Standby Current	52mA	40mA

- 360 Laser Distance Sensor LDS-01 is a 2D laser scanner capable of sensing 360 degrees that collects a set of data around the robot to use for SLAM (Simultaneous Localization and Mapping) and Navigation.
- The LDS-01 is used for TurtleBot3 Burger, Waffle and Waffle Pi models.
- It supports USB interface(USB2LDS) and is easy to install on a PC.
- It supports UART interface for embedded baord.

• **Items**	**Specifications**
Operating supply voltage	5V DC ±5%
Light source	Semiconductor Laser Diode(λ=785nm)
LASER safety	IEC60825-1 Class 1
Current consumption	400mA or less (Rush current 1A)
Detection distance	120mm ~ 3,500mm
Interface	3.3V USART (230,400 bps) 42bytes per 6

• Items	Specifications
	degrees, Full Duplex option
Ambient Light Resistance	10,000 lux or less
Sampling Rate	1.8kHz
Dimensions	69.5(W) X 95.5(D) X 39.5(H)mm
Mass	Under 125g

Measurement Performance Specifications

Items	Specifications
Distance Range	120 ~ 3,500mm
Distance Accuracy (120mm ~ 499mm)	±15mm
Distance Accuracy(500mm ~ 3,500mm)	±5.0%
Distance Precision(120mm ~ 499mm)	±10mm
Distance Precision(500mm ~ 3,500mm)	±3.5%
Scan Rate	300±10 rpm
Angular Range	360°

Items	Specifications
Angular Resolution	1°