# 测试总结

## 从设计单位角度出发的测试策略

从设计单位的角度出发，对系统中的各个接口、函数进行单元测试，测试内容如表格所示

|  |  |
| --- | --- |
| **被测试的接口、函数名** | **概述** |
| getinstance() | 返回一个唯一的tcp服务线程实例 |
| connectSend(String str) | 新建一个socket，与服务器端建立连接，并发送字符串str，启动收听线程 |
| startServerReplyListener() | 新建一个线程，接收服务器发送的消息 |
| run() | 启动tcp服务线程，从UI进程接收消息并作出相应处理 |
| str2shell2(str) | 将输入的字符串转换为机器人的指令并执行 |
| tcplink() | 启动服务器，监听消息，并将收到的消息发送至移动端，表示确认 |
| minimal.launch | 启动相关节点，为控制移动做准备 |
| hector\_mapping.launch | 开始建图 |
| map\_saver | 保存构建完毕的地图文件 |
| go\_forward | 控制机器人以0.1m/s的速度前进1s |
| go\_backward | 控制机器人以0.1m/s的速度后退1s |
| go\_left | 控制机器人以0.1m/s的速度向左行走1s |
| go\_right | 控制机器人以0.1m/s的速度向右行走1s |
| turn\_left | 控制机器人向左转动 |
| turn\_right | 控制机器人向右转动 |
| script\_add1.launch | 机器人前往指定地点，收到语音命令后，抓取物体，并返回起点 |
| script\_add2.launch | 机器人受用户控制行走至目的地点，记录地点，重复两次，随后返回起点。然后前往地点一，地点二，返回起点 |
| shopping.launch | 启动跟随模式 |

首先针对各个函数、接口进行单元测试是十分必要的。如果没有单元测试而直接进行集成测试，一旦出现问题后，由于系统的复杂度比较高，定位问题比较困难，也不利于快速解决问题。进行单元测试，测试比较简单直接，能够发现各个函数实现的是否正确，便于快速确定出现问题的位置。在进行集成测试之前，确保每一个功能部件基本正确，为后续测试打好基础。

## 从需求覆盖角度出发的测试策略

从需求覆盖的角度，对系统进行测试，主要的测试内容如表格所示

|  |  |
| --- | --- |
| **测试用例序号** | **测试对应的需求** |
| T108TC01 | 生成房间的地图 |
| T108TC02 | 完成机器人的移动功能，包括前进、后退、左移、右移、左转、右转 |
| T108TC03 | 机器人抓取物体，并带着物体回到出发点，将物体松开 |
| T108TC04-1 | 机器人根据设定的导航点进行定点巡逻 |
| T108TC04-2 | 机器人遇到障碍物后停止运动，并在移开障碍物后继续运动 |
| T108TC05 | 机器人拿到指定的物体并返回 |

针对提出的需求进行测试，将系统的各个部分整合起来，确保系统能够正常完成提出的各种需求。仅仅进行单元测试是不够的，将各个部分组合起来后可能出现一些问题，从需求覆盖的角度进行测试，能够模拟用户使用的过程，降低用户使用的过程中出现问题的概率。

所有测试综合起来，我们的安卓应用端代码覆盖率98%，分支覆盖率92%；服务器端代码覆盖率达到96%。

## 测试中发现的问题

在测试过程中，发现的问题主要存在于物体抓取部分、运动控制部分以及跟随部分。

在物体抓取过程中，如果桌面上物体较多，机器人可能无法正确识别被抓取物体，导致抓取失败。此外，经测试发现机器人的物体识别算法在识别桌面的平面时，仅能识别前方带挡板的桌面，面对其他种类的桌子时无法正确识别桌面。

在运动控制的过程中，机器人在执行向右移动时会向右后方偏移，可能是机器人的轮轴出现磨损造成的。此外，机器人在执行前往一个点并返回的过程中，机器人返回的位置与实际的起点位置存在一个小的偏移，直线距离在0~4cm之间。在项目开发初期，可能出现机器人前进后无法停止的情况，在改进运动控制算法后，这个问题得到了解决。

在跟随模式下，如果用户行走的过快，机器人可能丢失跟随的目标而原地旋转。