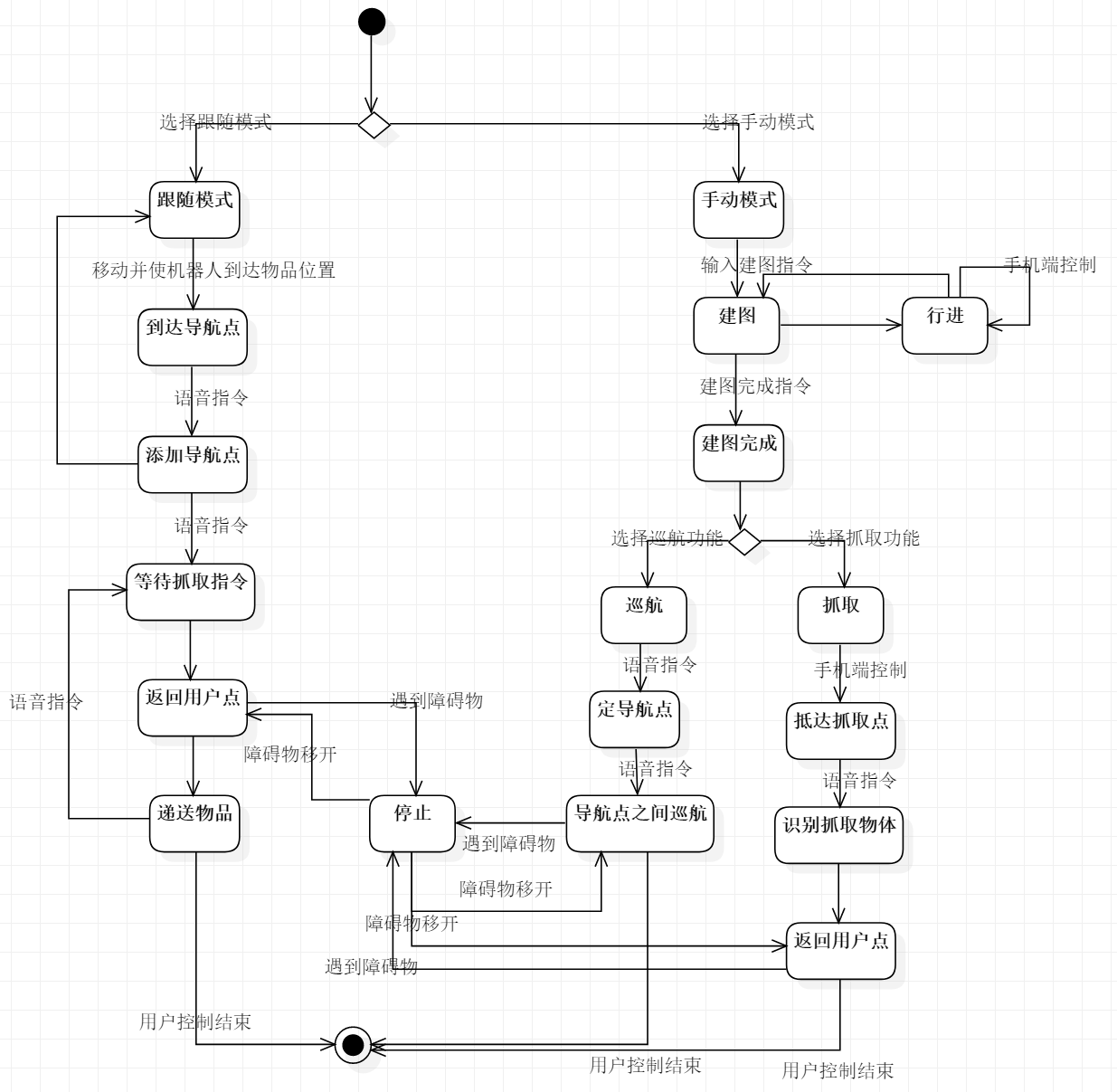
**设计阶段总结**

在机器人端所有程序的设计中，我们分析了机器人指导书上给出的相关例程，通过分析例程的方式来了解机器人运转的方式，综合我们在需求分析阶段提出的，让机器人拥有抓取、导航和建图三项功能，最后确定了如图所示的整体业务流程系统架构。



在确定了业务流程的基础上，我们逐步开始完成整个系统的架构。我们采取了增量开发的方式，逐渐向机器人中添加功能。完成的顺序如下表所示：

|  |  |
| --- | --- |
| 完成模块 | 对应需求阶段功能 |
| 建图模块 | 生成房间的地图 |
| 基础运动模块 | 实现用户操控的前后左右、转动操作 |
| 物体识别模块 | 识别摆放物体的桌面和桌面之上的物体 |
| 物体抓取模块 | 将识别出来的物体抓取 |
| 添加导航点模块 | 巡航时判断抵达的位置 |
| 巡航模块 | 根据导航点逐个生成路径并进行遍历 |
| 避障模块 | 遇到障碍物之后紧急停止 |
| 物体递送模块 | 松开机械爪把物体松开 |
| 语音识别模块 | 使机器人听到指令 |
| 任务脚本系统模块 | 串联以上模块使机器人能产生一定的逻辑行为 |

在设计阶段我们也遇到了非常多的挑战，但是通过大家的不懈努力我们想了很多办法来完成最终的工作。

在设计之初面临的巨大挑战就是我们的设计与机器人之间有较大的差别。我们设想中的机器人是什么都没有的一个框架，甚至需要我们自行去写底层的驱动。而我们看到的机器人是一个已经具备了一定功能的比较完备的机器人。这一定程度上减轻了我们编写程序的困难，但同时增加的困难是我们不得不花时间去了解现有的一些模块的调用方式以及现有功能的实现程度。ROS系统过于庞大，直至现在，完成了功能需求的我们，对一些其他我们没有用到的系统部分仍然是一知半解。

对系统了解程度不够这个弊病一直存在在整个开发周期中。例如我们一开始设想的物体识别是利用Kinect的摄像头去获取物体的照片，然后根据照片实现基于机器学习的物体识别，没想到底层只提供了照相的接口而并没有实现照相的算法。于是我们被迫放弃了这个方案。

在发现了这个问题之后我们转换思维，从现有的例程中去学习ROS系统，从现有的例程中去改造例程使之成为我们需要的功能。我们后半程的开发过程中，我们逐渐将ROS视为一个黑盒子，试图直接在我们写的程序和机器人的表现中建立直接的联系。这就大大加大了我们调试的难度。

具体而言，我们实现的第一个自己的功能就是在未建图的前提下，让机器人能够如同遥控小车一般实现遥控。我们出发的样例程序是指导书提供的速度控制的程序。而这个程序仅仅实现了让机器人动起来，却没办法让它停下来。在很长时间的调试后，我们终于明白了让它停下来需要再次发送一次0速度，让机器人静止下来。但又出现了新的问题：有时当机器人运行很多的节点时，小车又没法停下来了。在很多次不同条件下测试之后，我们终于找到了，有可能是应为这边速度程序发出的控制命令时间太短，当这边被挂起的时候速度控制的信号丢掉了。于是我们再行改进，将程序改成阻塞式的程序，最终终于完成了目标。

整个开发过程看下来，很希望课程组能够生成一套ROS的简易教程以及如何在ROS上编程（编写Cmake文件）的教程。这样能大量减少后续的上手的工作量，以及可能导致的误删系统文件的问题。

**测试总结**

在机器人端的测试工作中，我们主要分为单元测试和用例测试两部分进行展开。

其中单元测试是在每次编码之后进行的测试，主要也分为两个部分，一是以lauch的方式调用我们写的程序来判断这个程序的行为是否正确。二是在关键节点的地方加入ROSINFO的方式来观测我们传入其他节点的数据是否正确。（在这里我们假设如果我们传入了正确的参数，相关节点就能返回正确的行为）

用例测试中，我们根据我们的需求，编写了相关的用例，并对通过这些用例提出了一定的条件。

|  |  |
| --- | --- |
| 序号 | 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 | 抓取物体，机器人抓取物体，并带着物体回到出发点，把物体松开。 |
| T108TC04-1 | 机器人根据设定的导航点进行定点巡逻。 |
| T108TC04-2 | 机器人停止运动，并在移开障碍物后继续运动。 |
| T108TC05 | 机器人能够拿到我们指定的物体并返回。 |

具体可以参考我们的测试文档。