

车载软件设计基础

课后实践3



**2022至2023学年第 1 学期**

|  |  |
| --- | --- |
| 学号 | 姓名 |
| E2020196 | 侯加喜 |
| 任课教师 | 陈恒鑫 |
| 成 绩 |  |

|  |  |
| --- | --- |
| 任务书 | |
| 任务内容 | 请针对课后实践2的内容，补充多线激光雷达和底盘模块，使用订阅者模式，实现以下功能。   1. 多线激光雷达作为主题方，发布主题，并接受订阅 2. 底盘模块中增加观察者，向雷达模块进行订阅注册，并接收订阅通知。 3. 当雷达更新障碍物状态时，底盘模块通过订阅者模式接收通知并执行对应操作。 4. 障碍物状态为“前方”，则底盘执行“后退”执行 5. 障碍物状态为“右前方”，则底盘执行“左转”执行 6. 障碍物状态为“左前方”，则底盘执行“右转”执行   注意：  1、底盘执行指令，只需要在屏幕输出类似“左转。。。。”文字即可。  2、障碍物状态可通过命令行方式输入。比如1代表前方障碍，2代表左前方障碍，3代表右前方障碍。 |
| 程序规范 | （1）所有程序代码采用C++编写，使用git进行源代码管理；  （2）类名、变量名、函数名应符合C++的命名规范，并在代码中前后保持一致；  （3）涉及面向对象的程序，例如自定义的类，应符合面向对象的设计原则；  （4）正确使用头文件和源文件，自定义的头文件应符合头文件的编写原则，例如用条件宏定义确保头文件不被多次引用、不在头文件中进行类和函数的实现（模板除外）； |
| 报告要求 | （1）报告至少应该包括程序设计、程序效果展示、总结分析3个部分；  （2）程序设计描述组成程序的模块、类、函数以及他们之间的相互关系，若有算法，可以描述算法流程；  （3）程序效果展示除了程序运行效果截图之外，应该有必要的文字说明；  （4）总结分析可以分析实现的效果与理想情况的差异，分析导致这些差异的原因，切忌不要写成心得体会；  （5）报告应该格式规范、排版整洁、少语病和错误。 |
| 作业提交 | （1）含有git仓库（有.git目录）的完整源代码；  （2）任务报告。 |
| 评分标准 | 按照五级制打分，分为优秀、良好、中等、及格、不及格，各评分项占总成绩的比例为：  （1）任务完成情况占评分的60%；  （2）报告占评分的40%。  评分老师根据各部分的完成情况，直接给出总成绩。 |

1. **程序设计**

**1.1数据结构**

由作业2已知,智能小车由底盘、Agx套件、轮胎、摄像头等部件组成，并且每位学生分配一辆智能小车。故设计了10个class用于表示这些组件及学生，数值通过程序输入。

为实现功能要求,在**CmultiLineLidar**类中设计**void setObstacle(int \_directionOfObstacle)**和 **void publisher()**方法,分别实现障碍物的认为设置和障碍物的检测发布。

多线激光雷达类数据结构:



图1.1-1 CmultiLineLidar

在**CChassis**类中设计**void subscriber(int publishSign)** 和 **void turn(int turnDirection)**方法分别实现对多线激光雷达型号的订阅和转向动作的实施。

底盘类数据结构:



图 1.1-2 CChassis

**CintelligentCar**类中**CmultiLineLidar**和**Cchassis**均为私有变量,故在**CintelligentCar**设计

void setMultiLineLidarObstacle(int directionOfObstacle);

int multiLineLidarPublisher();

void chassisSubscriber(int publishSign);

三个public方法实现对**CmultiLineLidar**和**Cchassis**中方法的调用**。**



图 1.1-3 CIntelligentCar

**1.2关键函数**

(1).人为设置障碍物

void CMultiLineLidar::setObstacle(int \_directionOfObstacle)

{

this->directionOfObstacle = \_directionOfObstacle;

return;

}

(2).发布障碍物位置信息

int CMultiLineLidar::publisher()

{

int dir = this->directionOfObstacle;

switch (dir)

{

case 1:

return 1;

case 2:

return 2;

case 3:

return 3;

default:

break;

}

return -1;

}

(3).订阅发布者发布的信息

void CChassis::subscriber(int publishSign)

{

switch (publishSign)

{

case 1:

this->turn(1);

break;

case 2:

this->turn(2);

break;

case 3:

this->turn(3);

break;

default:

break;

}

return;

}

(4).转向动作实施

void CChassis::turn(int turnDirection)

{

switch (turnDirection)

{

case 1:

std::cout << turnDirection << ": 后退..." << std::endl;

break;

case 2:

std::cout << turnDirection << ": 右转..." << std::endl;

break;

case 3:

std::cout << turnDirection << ": 左转..." << std::endl;

break;

default:

break;

}

return;

}

(5).在Cintelligent中调用私有对象的方法

void CIntelligentCar::setMultiLineLidarObstacle(int directionOfObstacle)

{

this->multiLineLidar.setObstacle(directionOfObstacle);

return;

}

int CIntelligentCar::multiLineLidarPublisher()

{

int publishSign;

publishSign = this->multiLineLidar.publisher();

return publishSign;

}

void CIntelligentCar::chassisSubscriber(int publishSign)

{

this->chassis.subscriber(publishSign);

return;

}

1. **程序效果展示**

**2.1编译源代码**

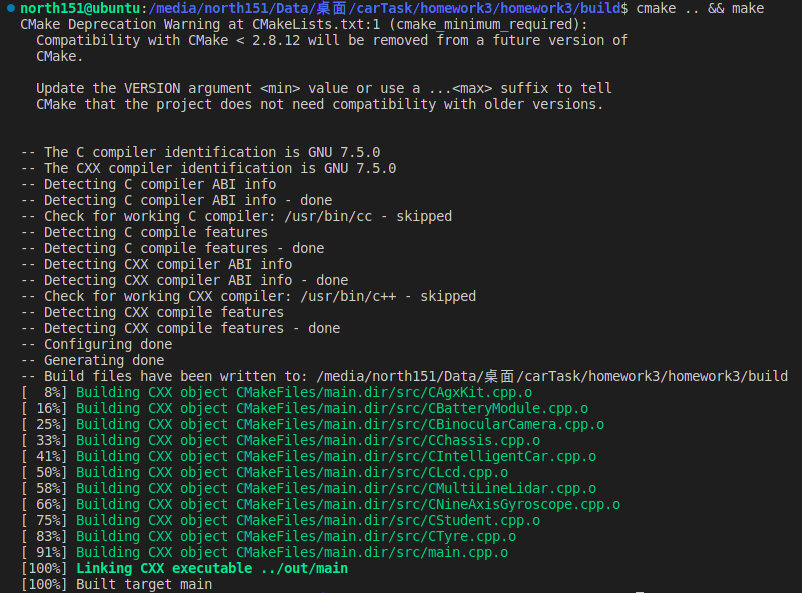


图 2.1-1 源代码编译

**2.2录入结果展示**

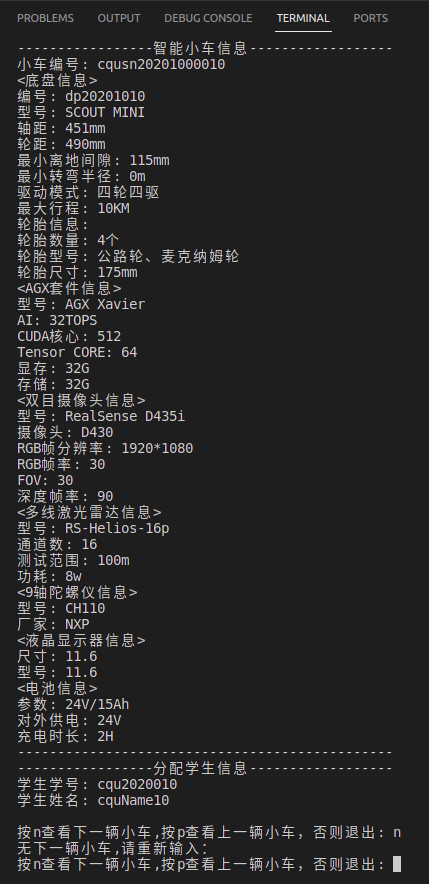


图 3.2-1 录入效果展示

**2.3测试小车**

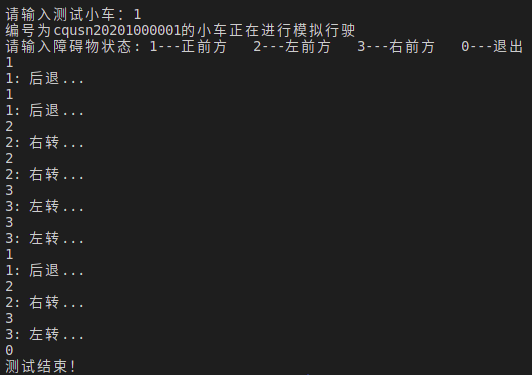


图2.3-1 小车测试效果展示

**三、总结和分析**

[Github](https://github.com/north151/carTask): https://github.com/north151/carTask

[Gitee](https://gitee.com/north151/homework3): https://gitee.com/north151/homework3

此次实验基本达到目标，小车的激光雷达在探测到前方障碍物后能够向订阅它的底盘组件发出障碍物状态信息，然后进行后退、转弯操作。最核心的是要理解观察者模式的特点，观察者模式又叫发布-订阅模式，其中的订阅表示这些观察者对象需要向目标对象进行注册，这样目标对象才知道有哪些对象在观察它。发布指的是当目标对象的状态改变时，它就向它所有的观察者对象发布状态更改的消息，以让这些观察者对象知晓。这里我们清晰地划分出各个功能模块,通过void setObstacle(int \_directionOfObstacle)方法来人为协助设置障碍物,void turn(int turnDirection)方法来实现订阅者收到订阅消息后的执行动作。主要的发布-订阅发生在int CMultiLineLidar::publisher()和void CChassis::subscriber(int publishSign)之间。

**四、参考文献**

[1]详细解读设计模式：观察者模式.http://t.csdn.cn/XUqVc,2020-07-22.

[2]C/C++编写命名规范.http://t.csdn.cn/VzFcz,2022-05-21.

[3]C++ 保存数据的四种方式.http://t.csdn.cn/6g3xK,2021-03-20.

[4]C++面向对象设计原则详解.http://t.csdn.cn/KkfqZ,2020-03-29.

[5][美]Stephen Prata.C++ Primer plus[M].张海龙 袁国忠.北京:人民邮电出版社,2015:96-183.

[6]设计模式之观察者模式（C++）.www.cnblogs.com/carsonzhu/p/5770253.html,2019-07-14.