人工智能课程项目报告二

小组成员　王辉 2015201943

梁晓周 2015201921

吴红薇 2015201928

1.实验过程

本阶段目标：利用蓝牙串口模块，通过 AICAR 与手机的蓝牙连接，实现实时信息交互以及操作控制。小车的超声波传感器接收到的数据将会传输到手机上，而手机将可以控制小车的运行方向、速度，实现对小车的遥感控制。

1.1 解决现存问题

上阶段未解决的问题主要有两个：

一、小车卡住后无法通过后退来解决困局；

二、小车遇到障碍后只会右转，导致了“秦王绕柱”的局面（即小车绕着一根杆子一直绕圈）。

1.1.1 卡住后退

对于小车“卡住”的判断，我们首先想到采取以下办法：在每次 loop 循环都会执行的函数 ultrasonicCar()中加入了对距离变化的判断，在时间间隔 time2judgebackward 前后分别通过 getdistance()函数测量距离 s1、s2，若二者相等则视为小车卡住，执行倒车操作；否则正常执行 ultrasonicCar(s2)函数。

在多次试验后我们发现，小车卡住后并不会执行倒车操作。梁晓周同学提出，这有可能是因为我们对于距离差|s1-s2|的要求过于严格了，而超声波传感器本身的精度是有限的，因此可能会出现小车卡住了但多次测量的距离不同的情形，因此我们应设置阈值 deltadistance2backward，若|s1-s2|<=deltadistance2backward 则视为卡住。这一修改得到了良好的成效。

但新的问题又出现了，小车有时会一直倒车直至车尾卡在障碍物处，这种情形下，由于每次 loop 中测量的|s1-s2|都小于阈值，因此小车会无休止地执行倒车操作。为解决此问题，我们对“卡住”情形的处理增加了新的措施：小车有可能倒车，也有可能前进，这就有效地避免了这一问题。

但实验结果却差强人意，小车有时在停住后甚至停止了工作，马达也不动了。吴红薇同

学在观察代码后发现了问题：我们将 time2judgebackward 声明为了 int 变量，并将其作为 delay()函数的参数，导致其无限长地执行 delay()函数，故而小车停止运转。在将其类型改为 float 后问题得到了解决。

虽然我们保证了小车不会“卡死”，但是与此同时，我们发现小车经常撞到障碍物，这是由于测量距离差的用时 time2judgebackward 过长，而这段时间里是不执行障碍物判断并避开的操作的，因此会撞上障碍物。为此，我们将 time2judgebackward 缩减到 0.3s。

然而，多次试验后我们又发现小车经常做“不必要的”倒车操作，当它并没有卡住甚至前方都没有障碍物时，它有时也会倒车。王辉同学指出这可能是因为 time2judgebackward 的缩短导致了所测量的距离差|s1-s2|也会相应缩短，当小车行驶速度较为缓慢时，距离差就有可能小于阈值而判断为卡住。我们首先想到的是将阈值 deltadistance2backward 设为小车速度 maxspeed 的线性函数，但收效甚微。我们最后将其设为 1cm，发现小车在不同的速度 maxspeed 下都发挥良好。

此外，我们在倒车操作后又执行了函数 randomright()调换方向，保证倒车后不会再次沿着原方向前进而再次卡住。我们还将障碍物判断的阈值从 40cm 降到了 20cm，得到了更好的表现。

1.1.2 “秦王绕柱”

我们将 randomright()改为了 randomleftorright()，当判断距离 s<20cm 时，我们执行右转或左转操作，这就有效地避免了同方向环绕的问题。

在对代码进行了若干次修改后，我们发现小车前进时总是会向右倾斜，我们认为当前控制左右马达相对转速的参数 RLRatio 已不再适合小车的行驶。基于对该参数名字字面的认识，我们想当然地认为 RLRatio 值越高，则右轮相对转速越快，因此我们尝试着将 RLRatio 置为 2.0，结果发现右倾现象不见好转。当我们为这顽固的右倾分子烦恼时，吴红薇同学发现 forward()函数中 run()函数的参数为 BACKWARD，同时 backward()函数中的 run()的参数为 FORWARD，虽然我们对于为什么这样传参不甚理解，但这也引发了我们的猜想：RLRatio 值增大时，右轮 BACKWARD 的程度加剧，因而更加慢于左轮。因此我们又尝试着置该值为 1.7，收效良好。

1.2 新功能加载

1.2.1 蓝牙交互

老师所给的代码中已有下面几项控制选项：前进、倒车、左打转、右打转、向左倾斜、向右倾斜、停车、获得距离值、打开自动行驶、关闭自动行驶。

我们首先是用 iPhone 安装了蓝牙助手但探测到了许多蓝牙设备，我们并不清楚小车蓝牙的名称，在多次连接尝试后都没能找到正确的设备。在参阅使用文献后我们发现该蓝牙只支持与安卓手机连接，因此我们用安卓手机安装了“蓝牙<->串口”程序，并成功找到了名称为“JDY-30”的蓝牙设备与其连接，顺利实现了蓝牙交互。

在多次试验后我们发现蓝牙交互操作的功能和我们设想的有所出入：在执行某个操作如倒车后，我们发现小车转变为了手动驾驶模式。而我们的设想是在操作后小车仍然保持自动驾驶。因此，我们将每个操作下的“control=0”语句删除，这样当小车执行完某个操作后驾驶模式不会发生切换。只有当我们选择打开或关闭自动驾驶时才会切换模式。

但是试验中我们发现小车对于蓝牙发送的操作无动于衷，自顾自地在自动驾驶。我们于是在每个操作下都加入了 delay()语句，从而使得操作得以显现出来。

1.2.2 “倒车请注意”

为了尽力模拟实际车辆的情形，我们考虑加入蜂鸣器作为喇叭以及提示器。我们声明了新的 AF_DCMotor 变量 Sounder 来表示蜂鸣器，并为它分配了 2 号端口。我们设计了 call()和 rest()函数分别控制其发出响声和停止发声。

现实生活中，货车倒车时会发出“倒车请注意”的声响以提醒车后方的车辆或行人，于是我们在 backward()函数中加入了 call()语句，使得小车在倒车时蜂鸣器会长鸣。

但实验中我们发现蜂鸣器一旦发出声响就不会停止，无论小车倒车与否。这是因为虽然 backward()函数执行结束了，但是由于 call()函数设定了蜂鸣器的值，因此蜂鸣器不会随着函数的结束而停止发声。因此我们在所有其他的操作下加入了 rest()语句，以使得小车不会胡乱地报出“倒车请注意”给车尾人群造成不必要的恐慌。

此外，我们还为蓝牙交互设置了按喇叭以及关闭喇叭的操作，实现了对小车喇叭的人为控制。

1.2.3 车灯

说到模拟实际车辆，车灯是必不可少的：转弯打灯、倒车打灯、危险报警闪光灯……因此，出于端口数量限制，为了加上两个 LED 灯，我们不得不将蜂鸣器去除，而“倒车请注意”则用双闪灯来代替。其实这样的效果更好，因为马达的噪音在一定程度上掩盖住了蜂鸣器的

警示声，而光的形式则不受干扰。

与 call()、rest()类似地，我们为两个小灯分别设计了函数 Ltwinkle()、Lstop()和 Rtwinkle()、Rstop()。left()函数中加入了 Ltwinkle()函数，right()函数中加入了 Rtwinkle()函数，而 backward()函数中则加入了 Ltwinkle()和 Rtwinkle()函数。此外，为了避免类似于蜂鸣器长鸣不止的情形发生，我们在相应操作中加入了 Lstop()或 Rstop()函数。

在实验一开始时两个小灯都在发光后瞬间熄灭了，并且在后续的操作中都不再亮起。在直接通电检测后我们发现两个小灯烧了。王辉同学反应过来由于我们将小灯和蜂鸣器同等对待的缘故，小灯被加上的电压等同于马达所加上的电压，因此电压过高就烧了。我们特意为小灯设置了电压水平 lightRate 以保证其正常工作。在新购置的小灯到货后我们顺利完成了这些功能。

此外，我们还为蓝牙交互设置了长亮小灯以及关闭小灯的操作，同时为了防止对小灯的人工操作和小车行驶时小灯的闪灭之间的冲突，我们设置了标志值 twinkle_flag，当它被置为 1 时小灯长亮且不受小车行驶状态的影响。

我们还设计了危险报警闪光灯函数 Blink()：以同样的时间间隔 blink_interval 对两个小灯执行亮灭操作，并将函数放到 loop 中，保证闪光的持续性。

我们还为蓝牙交互设置了打开和关闭危险报警闪光灯、增加和降低闪光频率的操作。

2.蓝牙操作使用手册

0：关闭自动驾驶

1：打开自动驾驶

! : 双灯长亮

? : 关闭长亮

+ : 加快闪烁

- ：减慢闪烁

< : 左倾

> : 右倾

s : 停车

f : 前进

b : 后退

l : 左打转

r：右打转

q：双闪

p：关闭双闪

d：测距

记录人 梁晓周　　记录时间 2017/11/4