浙江大学实验报告

专业：电子信息工程
姓名：邢毅诚
学号：3190105197
日期：2020-10-26
地点：东三-406

课程名称：电路与电子技术实验　指导老师：祁才君　成绩：
实验名称：智能小车导航控制　实验类型：验证实验　同组学生姓名：郑冰阳

一、 实验目的

(1) 了解智能小车的电路设计

(2) 掌握并实现智能小车的简单控制策略

(3) 使用 QUARTUS 软件实现小车的智能导航功能

二、 实验基本内容

1. 实验内容

(1) 设计并安装实现跟随控制策略的导航功能，记录测试结果，并分析此策略的优缺点

(2) 设计并安装实现纠偏控制策略的导航功能，记录测试结果，并分析此策略的优缺点

(3) 设计并安装实现寻迹控制策略的导航功能，记录测试结果，并分析此策略的优缺点

(4) 设计并安装实现改进跟随控制策略的导航功能，记录测试结果，并分析此策略的优缺点

(5) 设计并安装实现复杂轨道运行的导航功能

2. 实验原理

(1) 轨道状态检测

轨道检测装置的电路图如下图所示，其中“TCRT5000”为红外传感器，由光电二极管以及光敏电阻组成。在本实验中，轨道的颜色为黑色，而其他路面为白色，当光电管位于轨道上时，检测输出逻辑 1，灯灭；光电管偏离轨道时，检测输出逻辑 0，灯亮。通过输出逻辑，我们可以使用 DE10 开发板控制电机的转动，从而实现相应的控制策略。

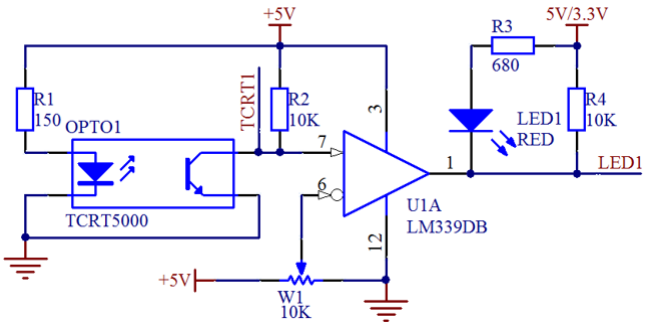

图 1: 轨道检测装置电路图

(2) 设计并安装实现跟随控制策略的导航功能，记录测试结果，并分析此策略的优缺点

跟随控制策略的导航功能如下图所示：

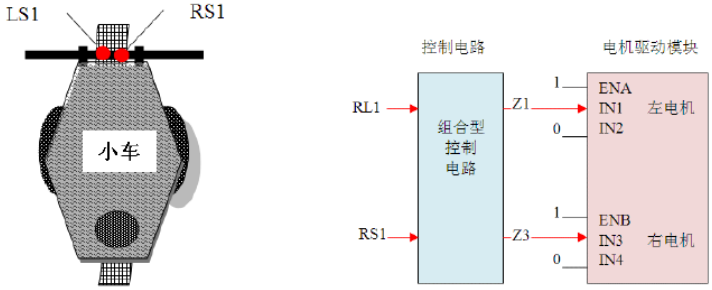

图 2: 跟随控制策略原理

跟随控制策略要求只使用两对光电传感器，并实现控制左右电机前行和停止 2 种状态，以及前进、左转、右转三种运动模式。当两对光电管位于轨道正上方时，小车前进，当左光管或右光管出轨道之后，小车应另一侧偏转，则该侧电机旋转而另一侧电机停止，若两灯管均偏离轨道，则停止旋转。

跟随控制策略的动作要求如下图所示：

左光电管（LS1）	右光电管（RS1）	控制左电机（ENA)	控制右电机（ENB)
偏离轨道	偏离轨道	停止	停止
偏离轨道	位于轨道	前进	停止
位于轨道	偏离轨道	停止	前进
位于轨道	位于轨道	前进	前进

表 1: 跟随控制策略动作要求

跟随控制策略的动作要求如下图所示：

左光电管（LS1）	右光电管（RS1）	控制左电机（ENA)	控制右电机（ENB)
0	0	1	1
0	1	1	0
1	0	0	1
1	1	0	0

表 2: 跟随控制策略逻辑要求

(3) 设计并安装实现纠偏控制策略的导航功能，记录测试结果，并分析此策略的优缺点

纠偏控制策略的原理图如下图所示：

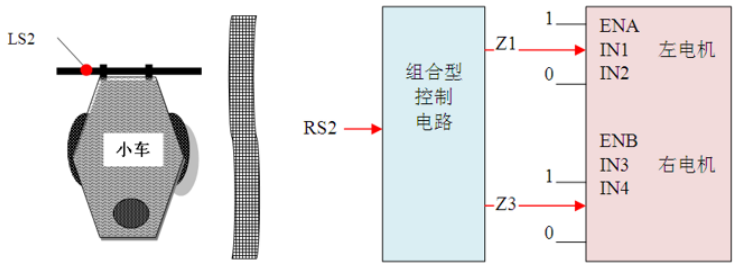

图 3: 纠偏控制策略跟随原理

纠偏控制策略要求只使用两对光电传感器，并实现控制左右电机前行和停止 2 种状态，以及前进、左转、右转三种运动模式。当两对光电管位于轨道外边时，小车前进，当左光管或右光管出轨道时，小车应此侧偏转，及该侧电机停止而另一侧电机旋转。

纠偏控制策略的动作要求如下图所示：

左光电管（LS1）	右光电管（RS1）	控制左电机 (ENA)	控制右电机（ENB）
偏离轨道	偏离轨道	前进	前进
偏离轨道	位于轨道	前进	停止
位于轨道	偏离轨道	停止	前进
位于轨道	位于轨道	停止	停止

表 3: 纠偏控制策略动作要求

纠偏控制策略的动作要求如下图所示：

左光电管（LS1）	右光电管（RS1）	控制左电机（ENA）	控制右电机 (ENB)
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

表 4: 纠偏控制策略逻辑要求

(4) 设计并安装实现寻迹控制策略的导航功能，记录测试结果，并分析此策略的优缺点

控制策略的原理图如下图所示：

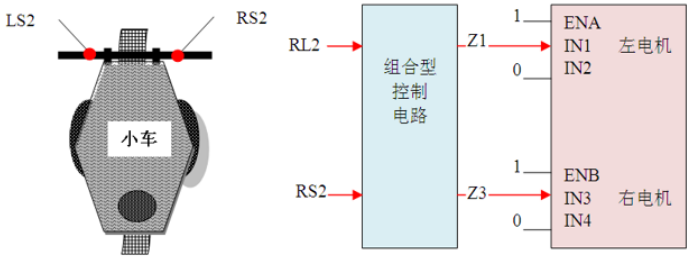

图 4: 寻迹控制策略原理

寻迹控制策略要求只使用一对光电传感器，并实现控制左右电机前行和停止 2 种状态，以及左转、右转两种运动模式。当光电传感器设置于左轮附近，若光电传感器位于轨道之上，则控制小车左转，否则，小车右转。

寻迹控制策略的动作要求如下图所示：

左光电管（LS1）	控制左电机（ENA）	控制右电机（ENB）
偏离轨道	前进	停止
位于轨道	停止	前进

表 5: 寻迹控制策略动作要求

寻迹控制策略的动作要求如下图所示：

左光电管（LS1）	控制左电机（ENA）	控制右电机 (ENB)
0	1	1
1	0	1

表 6: 寻迹控制策略逻辑要求

(5) 设计并安装实现改进跟随控制策略的导航功能，记录测试结果，并分析此策略的优缺点

跟随控制策略的导航功能如下图所示：

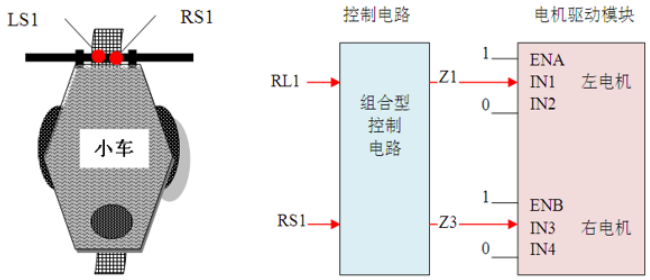

图 5: 跟随控制策略原理

改进跟随控制策略要求只使用两对光电传感器，并实现控制左右电机前行和停止 2 种状态，以及前进、左转、右转三种运动模式。其控制方法与跟随策略大致相同，但为使智能小车在旋转时更加方便，因此我们选择在旋转时将原来停止的电机改为向后倒转。

跟随控制策略的动作要求如下图所示：

左光电管（LS1）	右光电管（RS1）	控制左电机（Z1/Z2）	控制右电机（Z3/Z4）
位于轨道	位于轨道	前进	前进
偏离轨道	位于轨道	前进	后退
位于轨道	偏离轨道	后退	前进
偏离轨道	偏离轨道	停止	停止

表 7: 改进跟随控制策略动作要求

改进跟随控制策略的动作要求如下图所示：

左光电管（LS1）	右光电管（RS1）	IN1	IN2	IN3	IN4
0	0	0	0	0	0
0	1	0	1	1	0
1	0	1	0	0	1
1	1	0	1	0	1

表 8: 改进跟随策略逻辑要求

(6) 设计并安装实现复杂轨道运行的导航功能

在本实验中，我们采取基于跟随控制的复杂导航控制，在实际情况中，我们往往会遇到如下图所示的岔路以及停止等复杂的情况，因此便需要设计可以实现复杂轨道运行的导航功能。

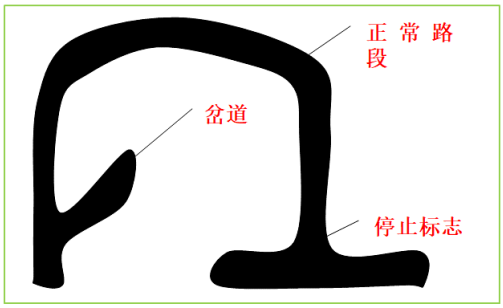

图 6: 跟随控制策略原理

为实现鉴别岔路以及停止等功能，我们选取四对光电传感器，在正常运行时，两对位于轨道之上，一对位于轨道的左边，另一对位于轨道的右边。为使小车遇到岔路时，沿主路行驶，遇到“T 字形”道路时主动停下，我们选取如下的动作要求：

左光电管 LS1	左光电管 LS2	右光电管 RS1	右光电管 RS2	ENA	ENB
0	0	0	0	1	1
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	1	1
0	1	1	0	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	1	1/0	0/1
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	0	0

表 9: 改进控制策略逻辑功能

其中，当 LS1 为 1，LS2 为 0，RS1 为 0，RS2 为 1 时，代表着当遇到左右岔路时，需要根据题目设定的要求，设定转动方向，由于本次实验中，并没有出现这种情况，因此我们设定当出现这种情况时，小车直行。

三、 主要仪器设备

(1) DE10 开发板

(2) FPGA 开发软件 Quartus(Quartus Prime Standard 17.1)

(3) 智能小车

四、 操作方法和实验步骤

(1) 检查实验仪器是否可以正常使用，有无缺失，将 DE10 开发板组装到小车之上

(2) 打开 Quartus 17.1，新建一个空项目，设置项目名为 **EX__1**, 设置硬件主体名为 **qxor**，选择开发设备为 **10M50DAF484C7G**，选择模拟环境为 ModelSim—Altera 并设置语言为 VHDL 语言。

(3) 在 Quartus 中分别编写各个实验的运行代码，具体代码如下图所示：

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.std_logic_unsigned.all;
4  USE IEEE.numeric_std.ALL;
5  USE IEEE.std_logic_arith.all;
6  entity qxor is
7      Port ( a,b: in  STD_LOGIC;
8             z1,z2: out  STD_LOGIC);
9  end qxor ;
10
11 architecture Behavioral of qxor is
12 begin
13 pro1:process(a,b)
14    variable ab:STD_LOGIC_vector(1 downto 0);
15     begin
16     ab:=a&b;
17     case ab is
18       when "00"=>z1<='0';z2<='0';
19       when "01"=>z1<='1';z2<='0';
20      when "10"=>z1<='0';z2<='1';
21       when others=>z1<='1';z2<='1';
22     end case;
23   end process;
24  end Behavioral;
```

图 7: 跟随控制策略代码

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.std_logic_unsigned.all;
4  USE IEEE.numeric_std.ALL;
5  USE IEEE.std_logic_arith.all;
6  entity qxor is
7      Port ( a,b: in  STD_LOGIC;
8             z1,z2: out  STD_LOGIC);
9  end qxor ;
10
11 architecture Behavioral of qxor is
12 begin
13 pro1:process(a,b)
14    variable ab:STD_LOGIC_vector(1 downto 0);
15     begin
16     ab:=a&b;
17     case ab is
18       when "00"=>z1<='1';z2<='1';
19       when "01"=>z1<='0';z2<='1';
20      when "10"=>z1<='1';z2<='0';
21       when others=>z1<='0';z2<='0';
22     end case;
23   end process;
24  end Behavioral;
```

图 8: 纠偏控制策略代码

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
USE IEEE.numeric_std.ALL;
USE IEEE.std_logic_arith.all;
entity qxor is
    Port ( a: in  STD_LOGIC;
           z1,z2,z3,z4: out  STD_LOGIC);
end qxor ;

architecture Behavioral of qxor is
begin
pro1:process(a)
    begin
    case a is
      when '0'=>z1<='1';z2<='0';
      when others=>z1<='0';z2<='1';
    end case;
  end process;
 end Behavioral;
```

图 9: 寻迹控制策略代码

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
USE IEEE.numeric_std.ALL;
USE IEEE.std_logic_arith.all;
entity qxor is
    Port ( a,b: in  STD_LOGIC;
           z1,z2,z3,z4: out  STD_LOGIC);
end qxor ;

architecture Behavioral of qxor01 is
begin
pro1:process(a,b)
   variable ab:STD_LOGIC_vector(1 downto 0);
    begin
    ab:=a&b;
    case ab is
      when "00"=>z1<='1';z2<='0'; z3<='1';z4<='0';
      when "10"=>z1<='0';z2<='1'; z3<='1';z4<='0';
     when "01"=>z1<='1';z2<='0'; z3<='0';z4<='1';
      when others=>z1<='0';z2<='0'; z3<='0';z4<='0';
    end case;
  end process;
 end Behavioral;
```

图 10: 改进跟随控制策略代码

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
USE IEEE.numeric_std.ALL;
USE IEEE.std_logic_arith.all;
entity qxor is
    Port ( a,b,c,d: in  STD_LOGIC;
           z1,z2: out  STD_LOGIC);
end qxor ;

architecture Behavioral of qxor01 is
begin
pro1:process(a,b,c,d)
    variable abcd:STD_LOGIC_vector(3 downto 0);
     begin
     abcd:=a&b&c&d;
     case abcd is
       when "1010"=>z1<='1';z2<='1';
       when "0101"=>z1<='1';z2<='1';
      when "1111"=>z1<='0';z2<='0';
when "0010"=>z1<='1';z2<='0';
       when "0000"=>z1<='1';z2<='1';
      when "0001"=>z1<='1';z2<='0';
when "0011"=>z1<='1';z2<='0';
       when "0100"=>z1<='0';z2<='1';
      when "0110"=>z1<='1';z2<='1';
when "0111"=>z1<='1';z2<='1';
       when "1001"=>z1<='1';z2<='1';
      when "1011"=>z1<='1';z2<='1';
when "1100"=>z1<='0';z2<='1';
       when "1101"=>z1<='1';z2<='1';
      when others=>z1<='1';z2<='1';
    end case;
  end process;
 end Behavioral;
```

图 11: 复杂轨道的控制策略代码

(4) 按照徐娅，设置管脚，并将管脚与对应接口连接

(5) 将代码编译，并导入 DE10 开发板之中

(6) 将小车放到赛道上进行测试，并观察小车在运行的时候的现象

五、 数据处理与分析

(1) 跟随控制策略

在小车使用跟随控制策略进行控制时，在正常情况下，小车可以完整的在轨道上面运行，但经过我们反复实验与思考，仍发现了一定的问题——在面对一些弧度较大的转弯时，小车并不能完美的处理，原因在于小车在弧度较大的弯道上面运行时，常常会出现两个光电传感器全部离开轨道的情况，这时小车便会停止；同时，此类小车面临着并不能完美的判断岔道以及在其他情况下正常运行的问题，原因在于此类小车只使用了两个光电传感器，这也就导致了小车只能判断转弯和停止两种情况，面对其他较为复杂的情况时，便失去了应对的方法。

(2) 纠偏控制策略

在小车使用纠偏控制策略进行控制时，在正常情况下，小车也可以完整的在轨道上面运行，但此类方面仍然面临着无法完美的判断岔道以及在其他情况下无法正常运行的问题，原因同样是此类小车只是使用了两个光电传感器；但此类小车在处理弧度较大的转弯时，会有较好的效果。

(3) 寻迹控制策略

在小车使用寻迹控制策略进行控制时，在正常情况下，小车也可以完整的在轨道运行，但是小车在运行时，会出现左右摇摆的现象，原因在于小车只采用了一对光电传感器进行控制，因此在小车运行的过程中，一旦出现失误，小车便会左右摇摆，因此此方法虽然使用的光电传感器较少，但并不能做到稳定运行，同时此类小车在面临弯道稍大的轨道时也不能完整的运行。

(4) 改进的跟随控制策略功能况在小车使用改进过后的跟随控制策略时，我们测验后发现，小车的运动状态基本与跟随控制策略一致，同时，小车在拐弯的过程中，小车会出现卡顿的现象，这对小车在转弯的速率影响较大，但同时，这种方法可以完美应对小车拐弯弧度较大的情况。

(5) 设计并安装实现复杂轨道运行的导航功能在小车使用改进过后的跟随控制策略时，我们测验后发现，小车基本可以完美的应对任何路线——无论是岔路还是停止等等，但是，当轨道与轨道之间的宽度较窄的路线时，小车会出现将另一轨道识别为岔道的情况，在我们进行测试后，小车运动的时间大致为 19 秒。

六、 探究题

查阅并简单描述无人驾驶的硬件框架，软件框架和系统构成

(1) 硬件框架

- 摄像头：主要用于车道线，交通标示牌，红绿灯以及车辆，行人检测等
- 激光雷达：主要用于高精地图制作，障碍物识别，跟踪和自身定位
- 毫米波雷达：主要用于交通车辆和行人的检测
- 定位系统：用于确定车辆自身位置
- 超声波传感器：主要用于近距离和低矮障碍物探测，避免车辆周围近距离感知盲区

(2) 软件框架

软件框架主要分为实时操作系统，运行时框架和各应用算法模块：

- 实时操作系统：一般是针对自动驾驶定制化的高实时、高并发、低时延的 Linux 操作系统。
- 运行时框架：基于操作系统层的各算法模块调度框架，主要负责各模块之间的消息通信、资源分配和运行调度等。目前主要的框架有开源的 ROS（Robot Operating System），以及百度自研的 Cybertron 框架。
- 高精地图：提供车道线拓扑结构、红绿灯位置、交通标志位置和类型、道路限速等信息服务，供感知、决策规划、定位等模块查询使用。
- 定位模块：为各算法模块提供厘米级的高精度定位信息，包括车辆的世界坐标、车辆姿态和朝向等信息。
- 感知模块：主要功能为检测车道线标志，识别红绿灯状态，检测跟踪识别车辆、行人，交通标识牌识别等。
- 规划模块：主要为基于定位信息、感知信息，结合行驶目的地信息，实时对行驶路线做出规划，为自动驾驶车辆提供行驶轨迹点。
- 控制模块：基于规划路径，对车辆行驶下发控制命令，主要为转向、油门、制动、灯光、喇叭、车内空调等的控制。

- 端到端：主要为数据采集、车辆状态和软件监控、在线升级客户端等为云端服务的各软件模块集合。
- 人机交互接口：主要为乘客或远程控制员，提供与车辆交互的功能，包括规划行驶路径，打开车载娱乐系统，查看车辆行驶状态等。

(3) 系统构成无人驾驶框架由硬件框架，软件框架，云端服务层构成。其中云端服务层主要运行在分布式的云端，为无人车提供各种服务，包括软件升级、数据更新、信息安全控制、仿真训练和车辆数据收集等，其具体构成如下：

- 高精地图：提供高精地图数据，包括静态地图，反射值地图等，共车端高精地图服务引擎查询或更新使用
- 仿真：提供评测或训练各算法模块的服务平台，可以基于实车采集的数据，不断丰富更新各种测试场景或训练数据，以提高自动驾驶系统的智能和适应性。
- 数据平台：实时存储从无人车上传来的各种数据，包括感知、车辆状态、行驶轨迹、各软件模块状态、关键 log 文件等，供云端平台做离线统计、分析、定位问题、训练模型使用。
- 安全：主要是对车端各种外部请求的认证、鉴权等，保证车端和云端的信息安全，避免遭受黑客攻击。
- 在线升级：提供对无人车的各软件模块的升级、数据更新、证书更新等功能，使得无人车随时保持最新的高精地图数据、配置信息和软件模块。

七、 心得与体会

1. 实验感想

在本次实验中，我们学习了智能小车的控制方法；在初次进行实验时候，我们发现我们组的实验小车有着一定的问题——相比于其他组，我们组的小车在运动过程中，会出现拐弯半径过大的情况，这也就导致了小车在转弯时会出现很多问题，在我们不懈的努力下，我们组最终成功运行，但同时，由于小车的问题，我们的总时间比其他组慢了许多，约为 19 秒。但是，在本次实验中，我也收获了很多东西，比如：智能小车的简易运行方法, 智能小车的运行架构等等，总体而言，收获颇多