软硬件协同设计技术报告
——汽车自动巡航系统

小组成员： 李克西 10112510348

张泽曦 10102510126

指导老师: 陈仪香

华东师范大学软件学院

2014 年 6 月 17 日

目　录

一、 摘要

传统的汽车驾驶过程中需要恒定速度时，需要驾驶员根据自身经验进行油量、刹车的控制，匀速的效果往往不尽人意，同时也会造成油量的浪费。采用自动巡航系统不仅方便驾驶员控制车速，同时也能降低汽车耗油量。为此我们对自动巡航系统进行了分析，展开了讨论，并从软硬件协同设计的角度建立控制模型。

系统主要包括状态机和自动巡航控制器两大模块。状态机分为三大状态：停车、手动驾驶、自动驾驶，其中自动驾驶（即巡航状态）又进一步细分为上坡加速、下坡减速及平地匀速三个子状态，从而覆盖了汽车运行过程中所有可能处于的状态。自动巡航系统对传感器采集的坡度值、汽车速度进行一定的处理，通过 PID 闭环控制来决定下一时刻汽车的目标速度，从而实现车速的自适应调整。

通过一系列信号模拟和参数调整，我们建立了一个能够将速度锁定在很小浮动范围的控制方法，并建立了模型。在今后的研究中，还可以引入其他因素，进一步完善控制系统。

二、 项目需求

车辆与运行相关性能要求为：

（1）设定车速120公里/小时

（2）上坡时增加油量以便保持车速

（3）下坡是减少油量以便保持车速

（4）每秒采集一次行车的坡度变化。

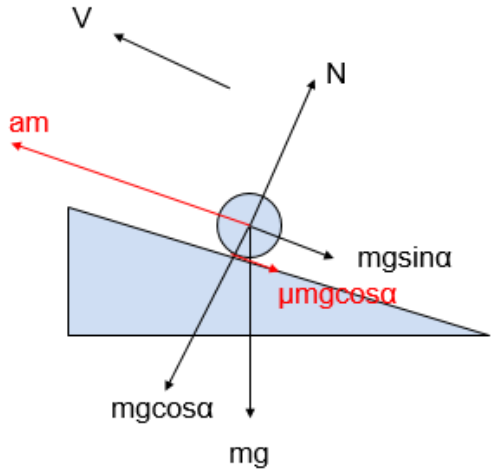
（注：油量与坡度有关，采集频率与车速有关）

三、 初步设计：

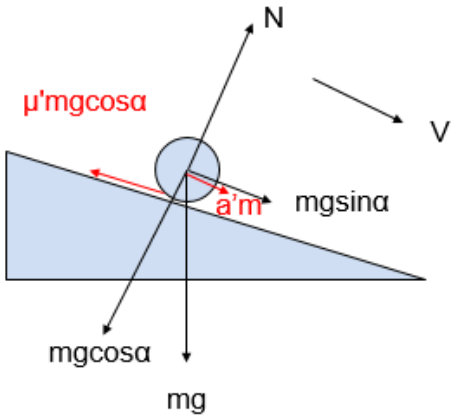
1. 相关物理模型的建立

为了保持恒定速度，当小车上坡时，需要提供一个额外的加速度来保持小车受力平衡，其方法为加大油量；当小车下坡时需要减小外力或者加大滑动摩擦因子 μ 来保持小车受力平衡，其方法为减小油量或者刹车。实际巡航过程中应以油

量为主，尽量避免使用刹车，以免造成速度的剧烈变化引起乘客不适。



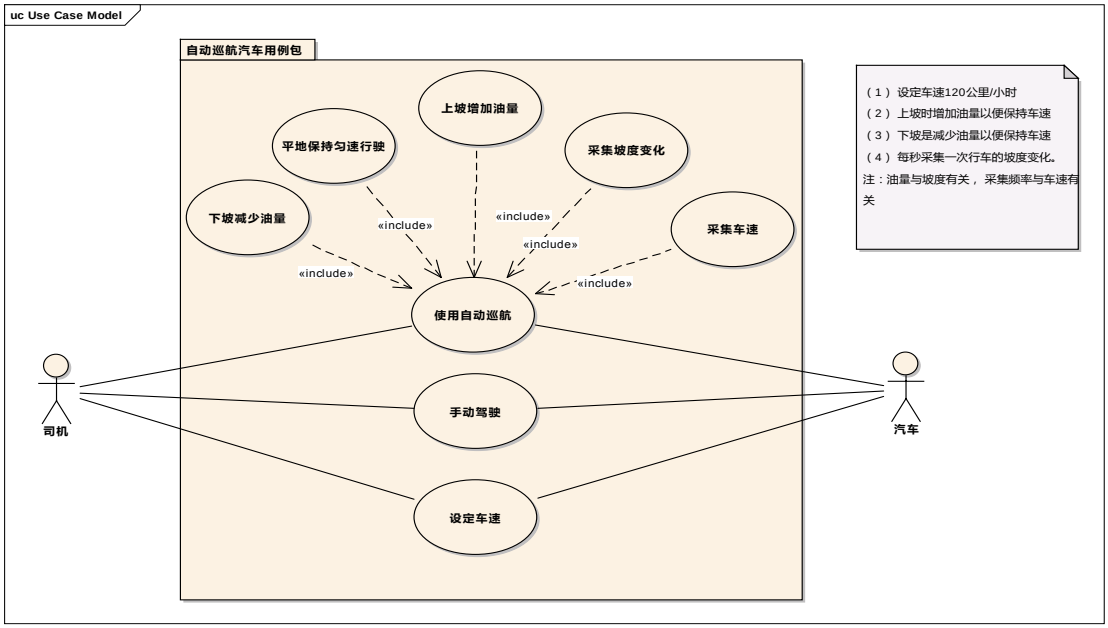
上坡模型



下坡模型

2. UML 设计

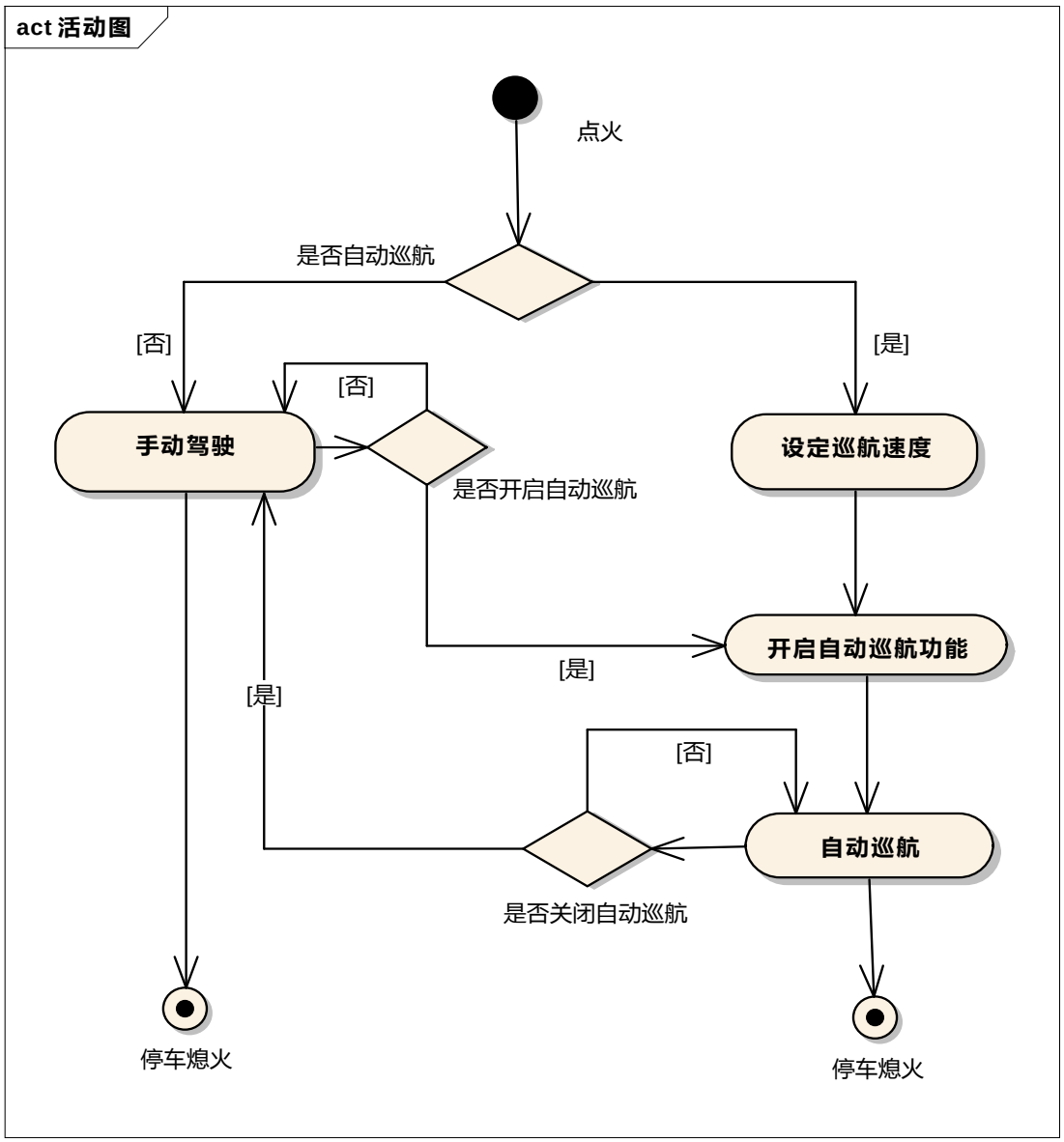
1) 系统用例描述（用例图描述）：



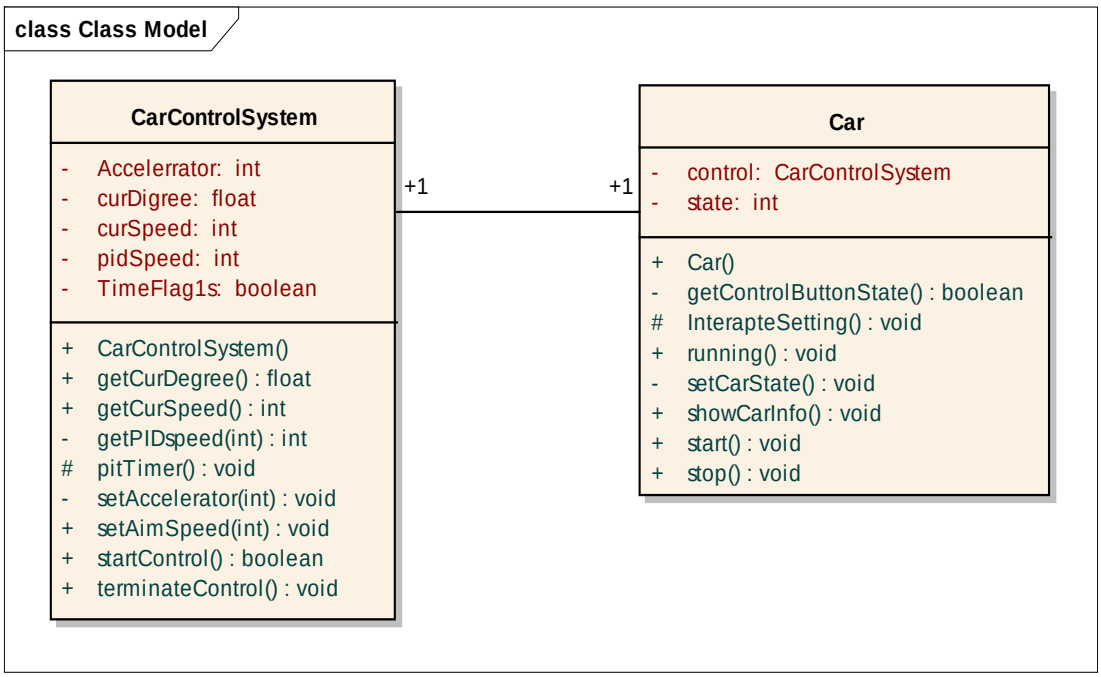
2) 系统活动描述（活动图描述）：

在这里我们只描述汽车手动、自动状态的切换过程，不对汽车巡航具体状态进行活动描

述，这一部分的描述将放到状态描述中。

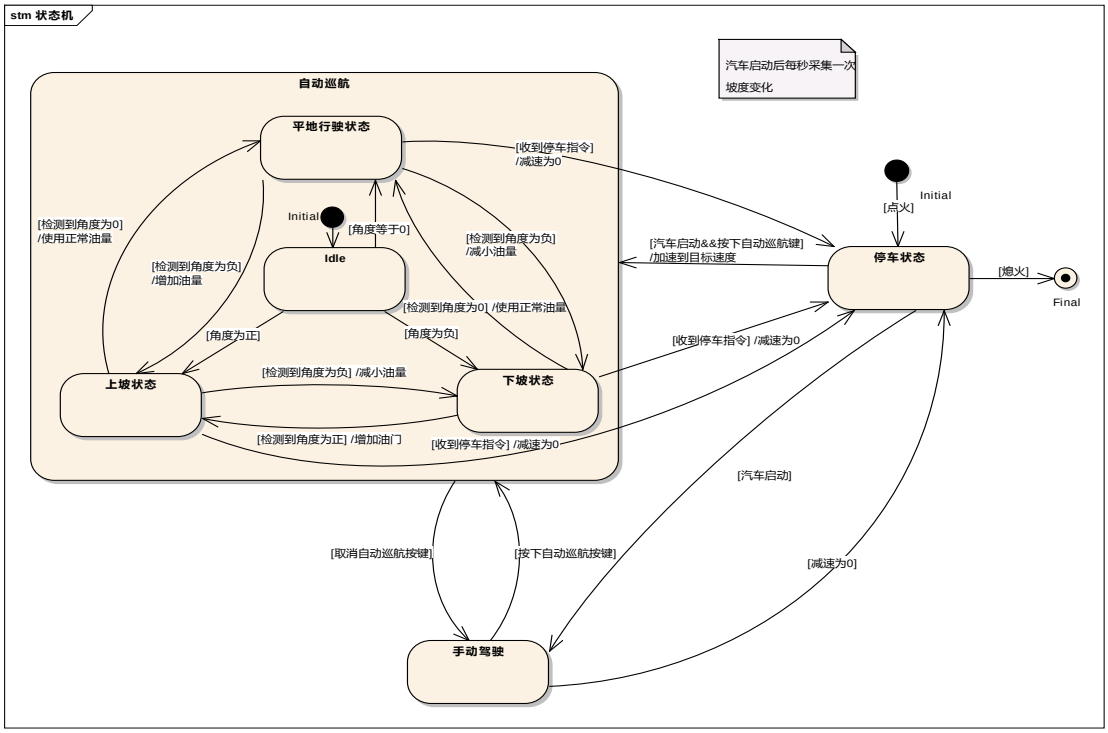


3) 系统方法描述（类图描述）：

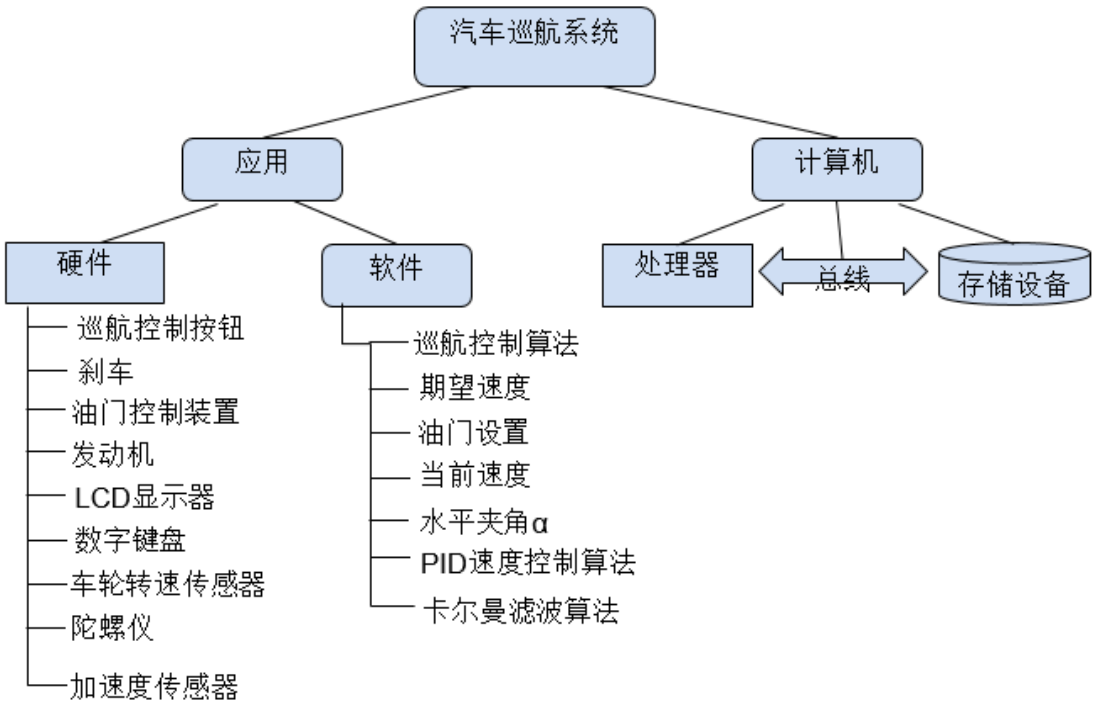


4) 系统状态描述（状态图描述）：

系统由三大状态构成：停车、手动驾驶、自动巡航，其中自动巡航又可以进一步分为一系列子状态：Idle、上坡加速、下坡减速、平地匀速。Idle 状态作为状态切换的临界状态，会迅速转移到其他状态上。



3. 软硬件划分



1) 硬件部分概述：

- **巡航控制按钮：**决定是否启用自动巡航功能；
- **刹车：**驾驶员手动驾驶时使用其减小车速或停车；
- **油门控制装置：**控制油量大小来提供所需动力，以完成加速或者减速操作；
- **发动机：**提供汽车所需动力；
- **LCD 显示器：**显示当前车辆行驶信息，包括：
  - 状态：停车、手动驾驶、自动巡航；
  - 车速：当前实际车速；
  - 油量：当前剩余油量、油的消耗速度；
  - 上下坡角度：当前处于上坡还是下坡状态及具体的角度。
- **数字键盘：**用户命令输入装置；
- **车轮转速传感器：**采集当前车速；
- **陀螺仪：**采集当前倾角；
- **加速度传感器：**采集当前汽车的加速度。

2）软件部分概述：

- **PID控制算法：**采用闭环控制，使车速根据目标速度进行自适应调整。
- **卡尔曼滤波算法：**对传感器采集的信号进行滤波处理，以排除噪声信号

对系统的干扰。

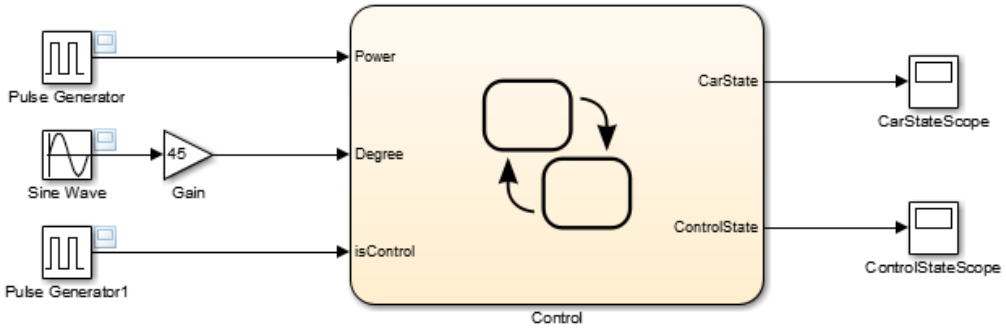
- **AD转换：**将传感器采集电平信号进行模数转换成物理信号。
- **油门控制算法：**根据厂商提供的数据进行函数拟合，得到指定车型的加速度、油量的对应函数。

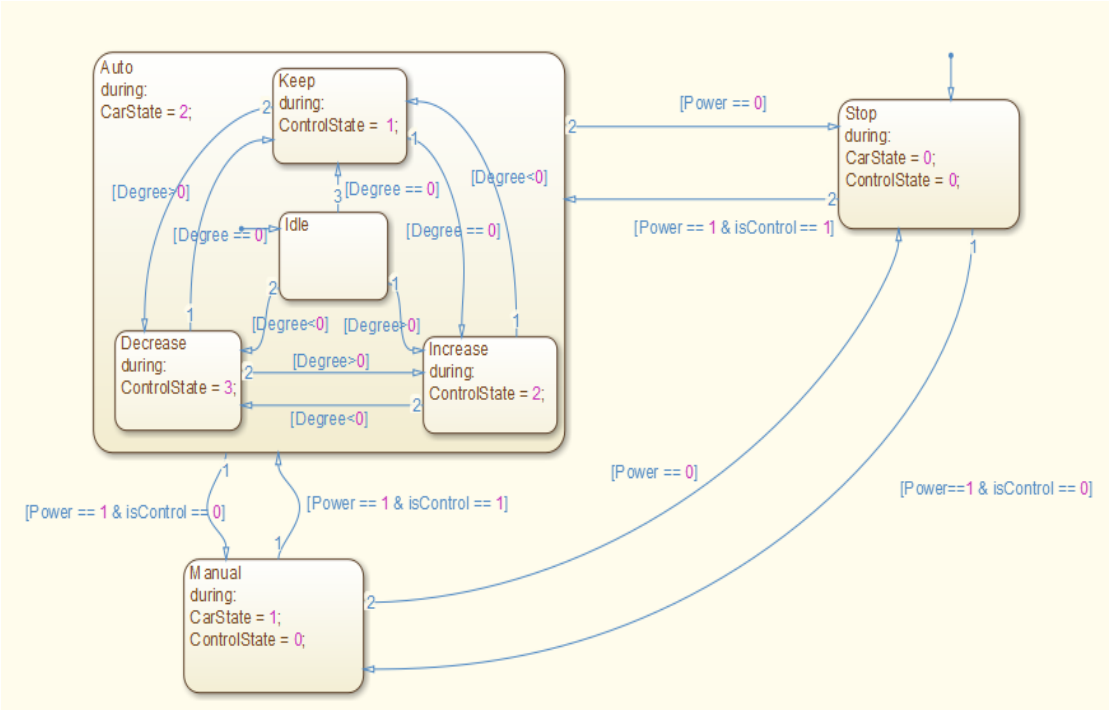
四、 精化设计：

1. 硬件设计：

1) 高速状态机：

根据 UML 状态图的描述，我们可以采用 simulink 中的 stateflow 对状态机进行建模，模型如下：





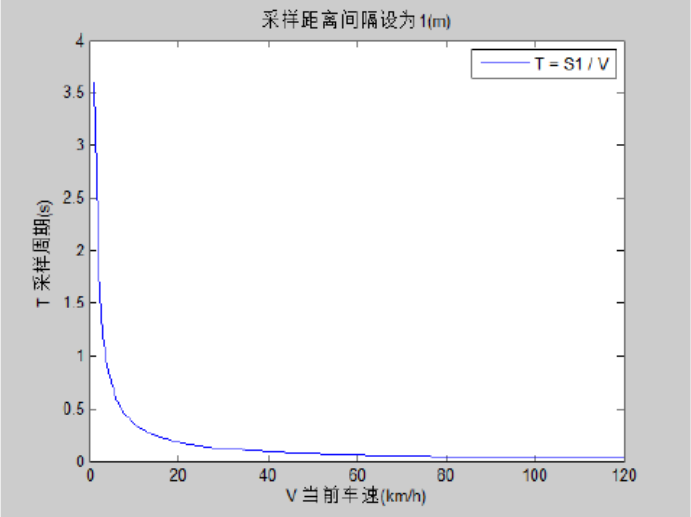
2) 传感器采样周期控制：

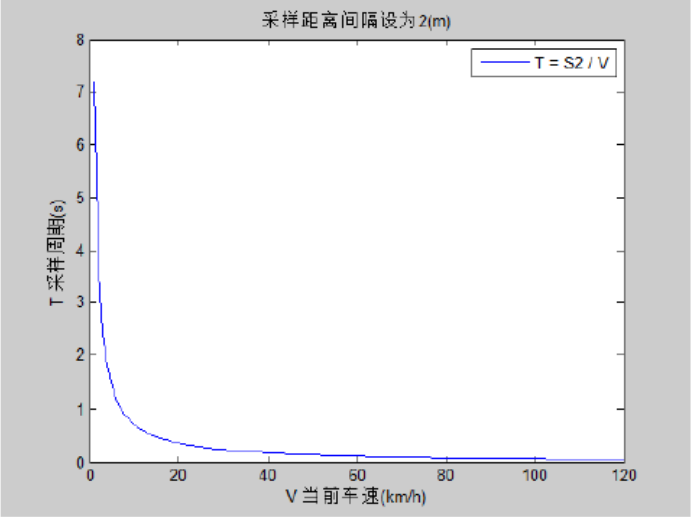
为了保证采集回来的数据的有效性，根据车速不同，我们需要制定对应的采样方案，在车速大的情况下提高采样速率，车速小的时候降低采样速率。如果把采样的路程间隔作为采样频率的基本单位的话，可以得到采样周期为：

$$T = S0/V$$

其中S0是我们设定的采样的路程间隔，V是当前车速，T是计算得到的采样周期。

我们考虑采样距离间隔为 1m 和 2m 的情况，得到对应关系图像如下：





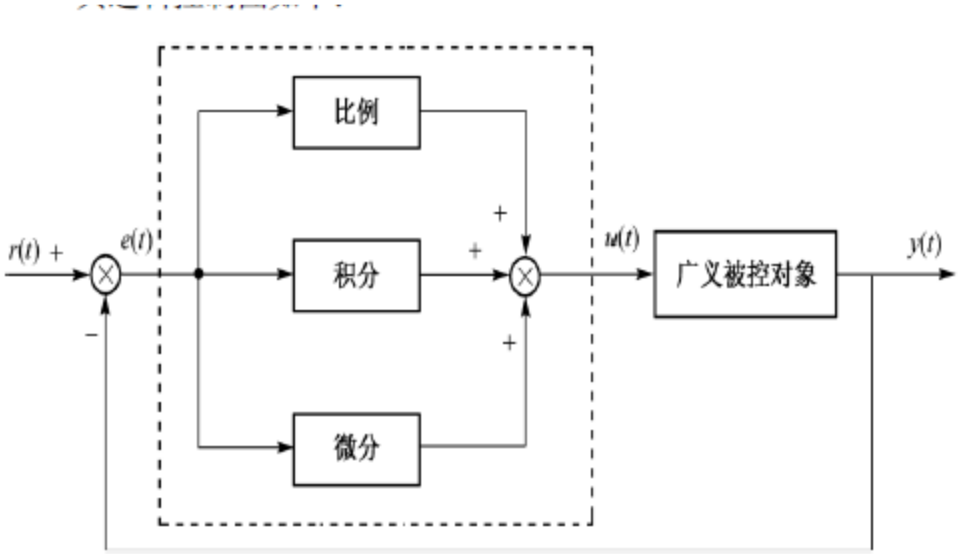
可以看到，在车速较高的情况下，采样间隔为1m或者2m的差别并不大，而在车速较低的情况下，间隔两米的采样周期就偏大了。最终，我们决定将采样间隔设置为1m。

2. 软件设计：

1. PID 控制算法：

比例积分微分控制算法，通过输入当前速度和驾驶员设定的目标速度，计算得出汽车在下一时刻的期望速度，从而改变油量大小对车速进行控制。这是一种闭环控制方法，可以使速度自适应调整，有效减少速度的剧烈波动。

其逻辑控制图如下：



对应的函数关系如下：

$$u(t)=K_{\mathrm{P}}\left[e(t)+\frac{1}{T_{\mathrm{I}}}\int_0^t e(t)\mathrm{d}t+T_{\mathrm{D}}\frac{\mathrm{d}e(t)}{\mathrm{d}t}\right]$$

对连续函数进行离散化以后，我们得到如下关系：

$$u(k)=K_{\mathrm{p}}\left\{e(k)+\frac{T}{T_{\mathrm{i}}}\sum_{j=0}^{k}e(j)+\frac{T_{\mathrm{d}}}{T}[e(k)-e(k-1)]\right\}$$

这便是我们在设计中需要实现的。

2. 卡尔曼滤波算法：

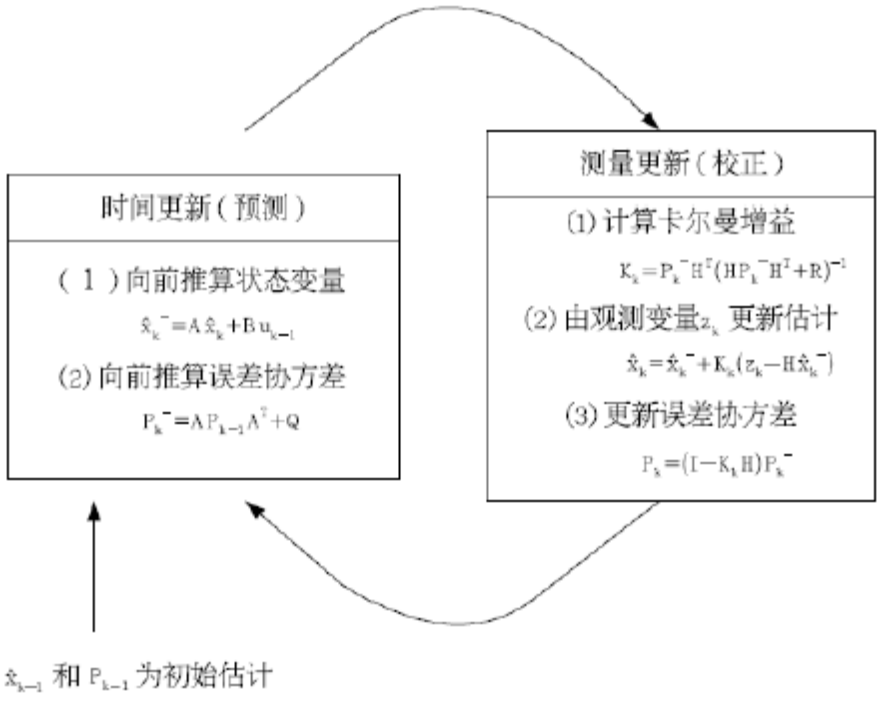
在实时系统中由于环境影响，采集回来的数据可能含有较多噪声，抖动较厉

害，我们实际需要的数据并不是随着一次抖动的发生便得出一个结果，这回干扰我们的控制系统。因此我们需要对采集回来的信号进行滤波处理。在这里我们采用卡尔曼滤波算法实现。

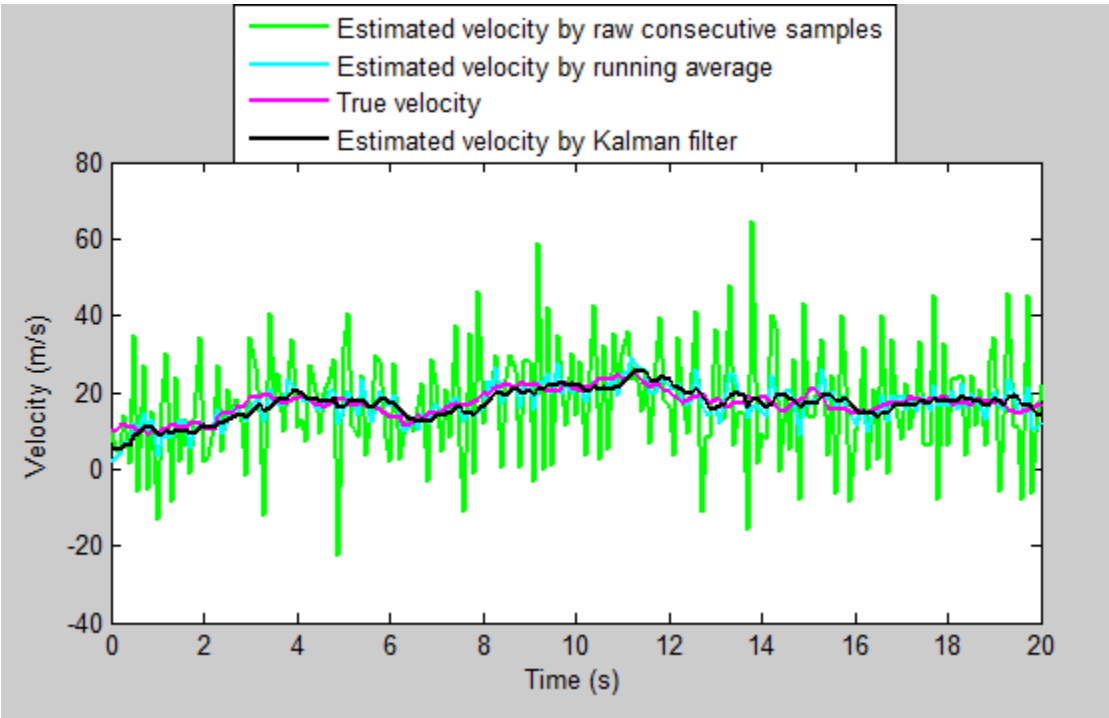
卡尔曼滤波器由一系列递归数学公式描述。它们提供了一种高效可计算的方法来估计过程的状态，并使估计均方误差最小。卡尔曼滤波器应用广泛且功能强大：它可以估计信号的过去和当前状态，甚至能估计将来的状态，即使并不知道模型的确切性质。

卡尔曼滤波器用反馈控制的方法估计过程状态：滤波器估计过程某一时刻的状态，然后以（含噪声的）测量变量的方式获得反馈。因此卡尔曼滤波器可分为两个部分：时间更新方程和测量更新方程。时间更新方程负责及时向前推算当前状态变量和误差协方差估计的值，以便为下一个时间状态构造先验估计。测量更新方程负责反馈——也就是说，它将先验估计和新的测量变量结合以构造改进的后验估计。

其工作原理可采用下图描述：



滤波效果如图所示：



五、 系统仿真

1. 高速状态机的 Verilog 仿真：

我们采用独热码的方式编写高速状态机，并用测试代码验证。

a) 状态机模块(StateMachine.v)：

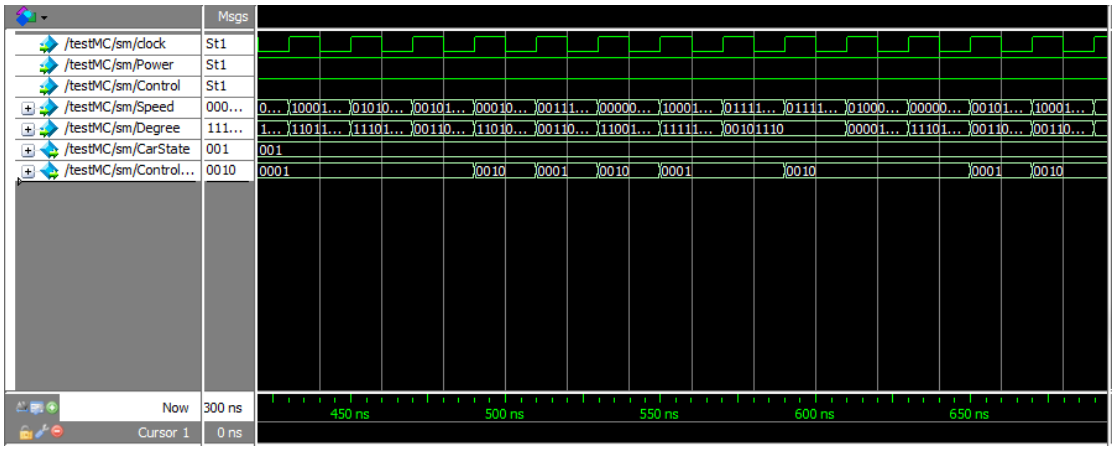
```
module StateMachine(clock,Power,Control,Speed,Degree,CarState,ControlState);
//clock 时钟,Power 汽车使能,Control 自动巡航使能,Speed 汽车速度，Degree 汽车角度
//CarState 汽车状态，ControlState 巡航状态
    input   clock,Power,Control;
    input[7:0] Speed;
    input[7:0] Degree;
    output[2:0] CarState;
    output[3:0] ControlState;
    reg[2:0]   CarState;
    reg[3:0]   ControlState;
    //使用独热码来标记状态
    parameter //CarState
                Stop = 3'b100,
```

```
                Manual = 3'b010,
                Auto = 3'b001,
                //ControlState
                Idle = 4'b1000,
                Keep = 4'b0100,
                Increase = 4'b0010,
                Decrease = 4'b0001;

        always @(posedge clock)
          if(!Power)
            begin
              CarState <= Stop;
              ControlState <= Idle;
            end
          else
            begin
              if(!Control)
                begin
                  CarState = Manual;
                  ControlState = Idle;
                end
              else
                begin
                  CarState = Auto;
                  case(ControlState)
                    //通过判断符号位 Degree[7]来判断角度的正负
                    Idle: if(!Degree[7]&&Degree>0) ControlState<=Increase;
                          else if(Degree[7]) ControlState<=Decrease;
                          else ControlState <= Keep;
                    Keep: if(!Degree[7]&&Degree>0) ControlState<=Increase;
                          else if(Degree[7]) ControlState<=Decrease;
                          else ControlState <= Keep;
                     Increase:    if(!Degree[7]&&Degree>0) ControlState<=Increase;
                                  else if(Degree[7]) ControlState<=Decrease;
                                    else ControlState <= Keep;
                    Decrease:    if(!Degree[7]&&Degree>0) ControlState<=Increase;
                                  else if(Degree[7]) ControlState<=Decrease;
                                  else ControlState <= Keep;
                  endcase
                end
            end
endmodule
```

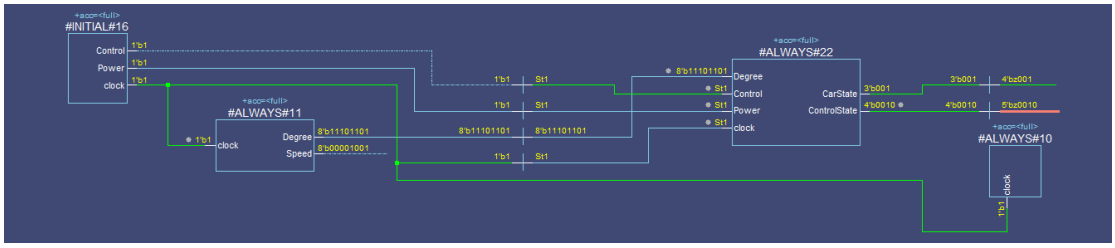
2) 测试模块(testStateMachine):

```
`timescale 1ns/1ns
`include "./StateMachine.v"
module testMC;
    reg clock,Power,Control;
    reg [7:0]Speed;
    reg [7:0]Degree;
    wire[3:0] CarState;
    wire[4:0] ControlState;

    always #10 clock = ~clock;
    always @(posedge clock)
        begin
            Speed = {$random}%150;
            Degree = $random%60;
        end
    initial
        begin
            clock = 0;
            Power = 0;
            Control = 0;
            #100 Power = 1;
            #200 Control = 1;
            #5000 $stop;
        end
        StateMachine sm(clock,Power,Control,Speed,Degree,CarState,ControlState);
endmodule
```

状态机仿真效果如下：



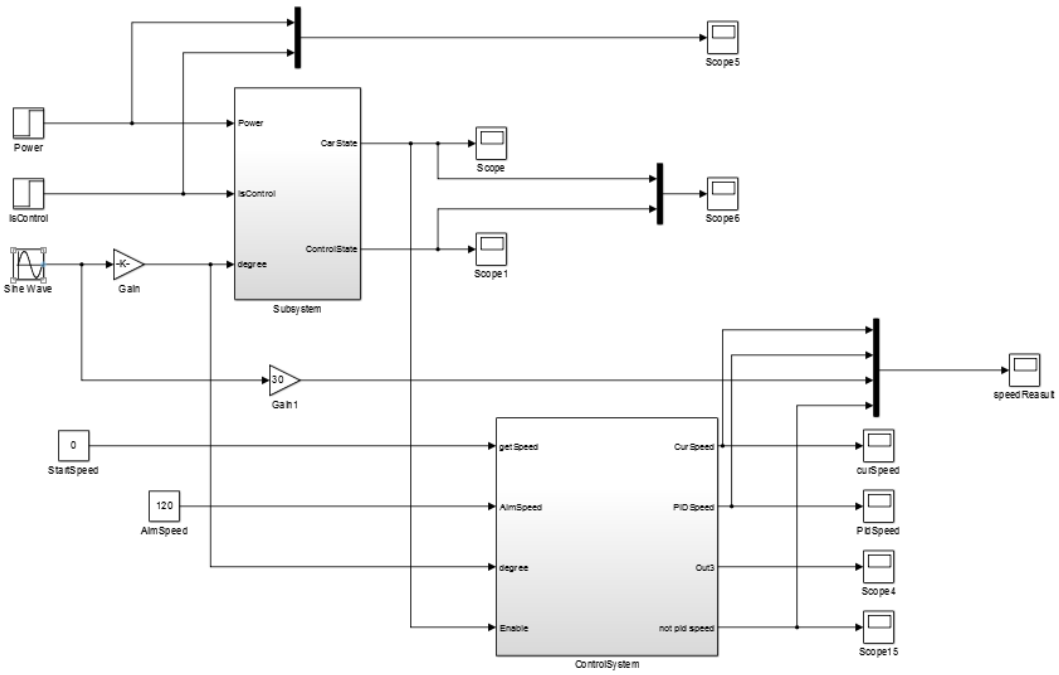
综合电路如下：



2. 系统的 simulink 仿真：

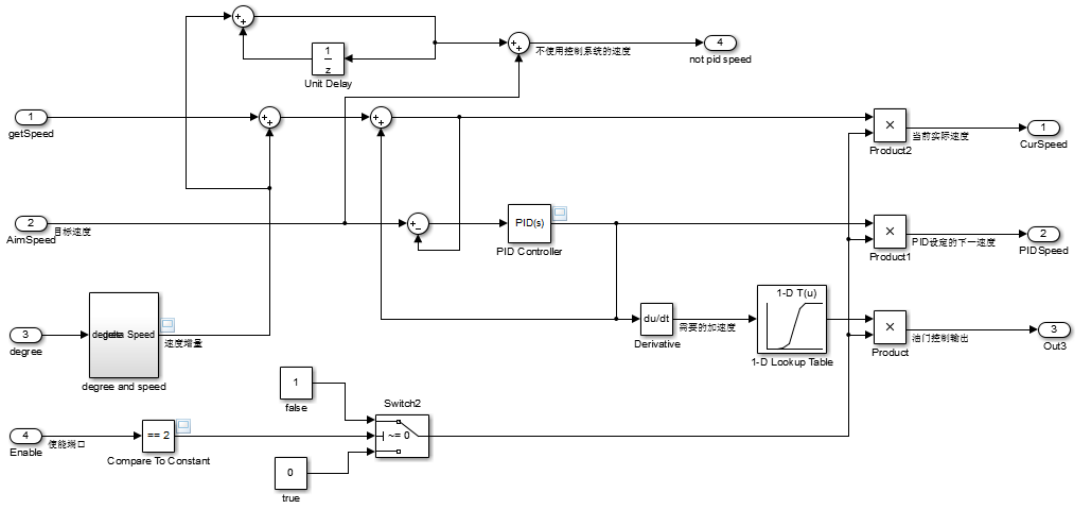
1) 整体结构：

系统整体分为状态机、巡航控制系统两个部分。巡航控制系统需要根据状态机的输出决定是否运行。carState 作为巡航控制的使能信号，只有处于巡航状态时，巡航系统才会开启。



2) 自动巡航控制模块：

自动巡航控制模块的核心是一个 PID 速度控制模块，通过将实际速度与目标速度做差，得到 PID 的输入信号，输出反馈控制速度。输出与实际速度叠加可以得到理论上汽车下一时刻的实际速度值，输出与目标速度叠加可以得到汽车的 PID 整定速度。



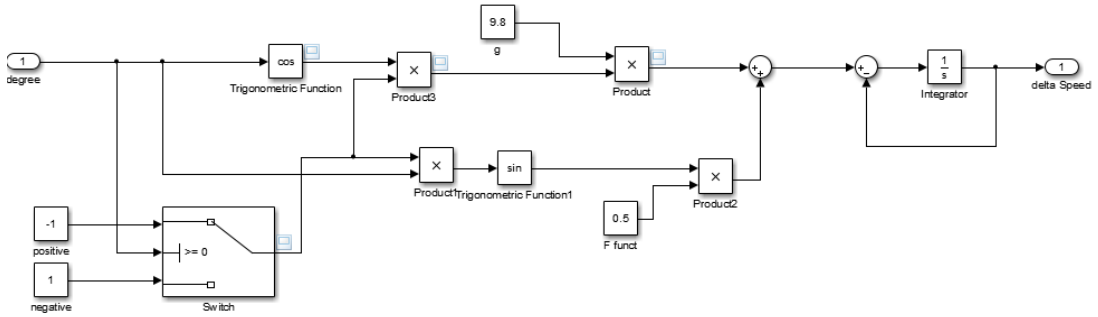
3) 坡度与汽车加速度关系的模拟：

由最初的物理模型的分析我们知道小车在上下坡受到外力的作用情况是不同的，重力分量上坡时是阻力，下坡时是动力。因此可以得到如下关系：

上坡： a = -g cos α -μ g sinα

下坡： a = g cos α -μ g sinα

根据这个关系我们可以模拟出汽车每一时刻外力对它速度的影响情况。

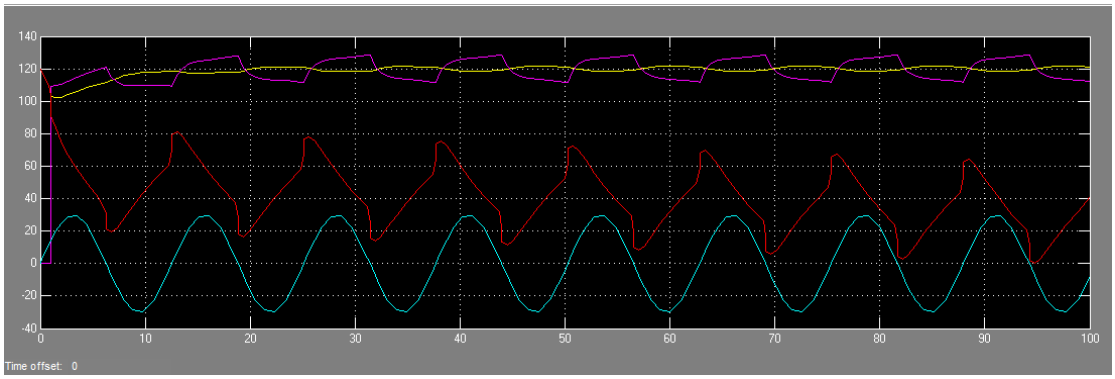


4) 仿真效果

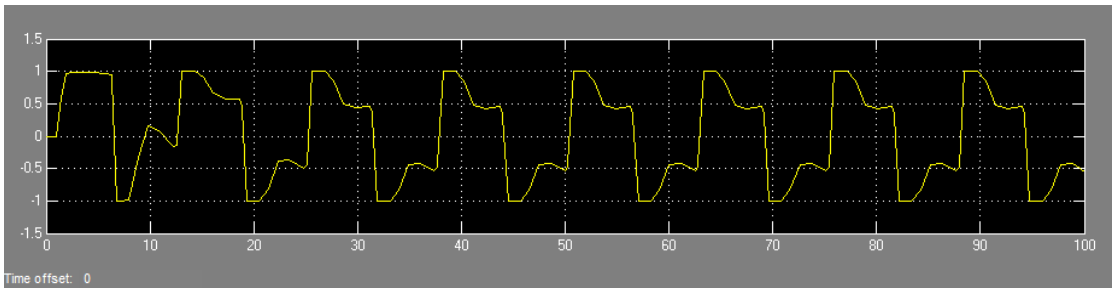
图中不同颜色的曲线有着不同的含义：

- 蓝色：坡度值（-30~30 度）
- 红色：不使用 PID 速度变化效果（初始速度 120，之后不加控制的变化情况）
- 粉色：PID 整定的目标速度
- 黄色：汽车实际运行速度

从图中我们可以发现，使用 PID 控制可以很好地将汽车巡航速度锁定在目标速度很小的浮动范围内，且速度变化十分平滑，效果较好。



下图是根据 look up table 得到的小车每一时刻油量控制的变化情况，与上图时刻相对应。



六、 总结与改进

本文针对汽车自动巡航功能，从基本的物理模型入手，进行了系统的设计、软硬件划分及建模，并给出了设计模型的实际运行效果，满足了需求规约。

在系统设计与模型建立的过程中，我们巩固了软硬件协同设计方法，按照要求一步一步逐步跟进，最终取得了较为满意的成果。

但是，在实际的汽车巡航中，还会有更多的影响因素，比如路况不同会是滑动摩擦因子不同、空气阻力会对速度造成一定影响、汽车使用能源不同导致单位能源提供的动力不同等等，这些因素有待于进一步讨论。

七、附录

小组成员分工情况：

- **李克西** 模型分析、UML 设计、系统划分、Verilog 与 Simulink 仿真、报告整理
- **张泽曦** 系统验证、材料收集、报告初稿