2013年全国大学生电子设计竞赛

**简易旋转倒立摆及控制装置（C题）**

**【本科组】**



**2013年9月6日**

**摘 要**

本系统采用XS128单片机最小系统作为控制中心、与电机驱动模块，减速直流电机、旋转臂、摆杆、反馈装置一起构成摆杆角度和电机角速度的双闭环调速系统。由于直流电机属于一阶惯性，经分析在控制周期远小于电机时间常数的情况下建立电压与加速度的近似比例关系模型。选择减速直流电机带动旋转臂旋转，采用增量式旋转编码器测量摆杆的角度，单片机输出占空比可变的PWM波控制电机角加速度，从而控制电机的加速度和给摆杆固定轴心的扭矩，实现控制摆杆的摆动幅度并达到和保持平衡状态。控制方式采用PID控制，比例环节进行快速响应，积分环节实现无静差，微分环节减小超调，加快动态响应。从而使该系统具有良好的性能，能很好的实现起振、圆周运动、倒立、360度倒立旋转，同时具有很好的抗扰动性能。

**关键词：**XS128 直流电机 加速度 倒立平衡

**目 录**

[1系统方案 1](#_Toc366347492)

[1.1 角度传感器模块的论证与选择 1](#_Toc366347493)

[1.2电机的论证与选择 1](#_Toc366347494)

[1.3 电机驱动模块的论证与选择 1](#_Toc366347495)

[2理论分析与计算 2](#_Toc366347496)

[2.1 机械结构设计的分析 2](#_Toc366347497)

[2.1.1 摆杆 2](#_Toc366347498)

[2.1.2 底座 2](#_Toc366347499)

[2.2电机物理模型的分析与计算 2](#_Toc366347500)

[2.2.1 直流电机的机械特性 2](#_Toc366347501)

[2.2.2 直流电机的动态特性分析 2](#_Toc366347502)

[2.2.3 直流电机加速度分析与计算 3](#_Toc366347503)

[2.3摆杆运动状态的分析及控制思路 4](#_Toc366347504)

[2.3.1摆杆起振 4](#_Toc366347505)

[2.3.2摆杆做圆周运动 4](#_Toc366347506)

[2.3.3摆杆达到倒立平衡状态 4](#_Toc366347507)

[2.4 摆杆控制量的分析与计算 4](#_Toc366347508)

[2.4.1 摆杆起振状态分析 4](#_Toc366347509)

[2.4.2 摆杆进入开始调整倒立平衡的角度范围 5](#_Toc366347510)

[3电路与程序设计 6](#_Toc366347511)

[3.1电路的设计 6](#_Toc366347512)

[3.1.1系统总体框图 6](#_Toc366347513)

[3.1.2 系统电路原理图 6](#_Toc366347514)

[3.1.3 电机驱动模块的电路设计 6](#_Toc366347515)

[3.2程序的设计 7](#_Toc366347516)

[3.2.1程序功能描述与设计思路 7](#_Toc366347517)

[3.2.2程序流程图 7](#_Toc366347518)

[4测试方案与测试结果 11](#_Toc366347519)

[4.1测试方案 11](#_Toc366347520)

[4.2 测试条件与仪器 11](#_Toc366347521)

[4.3 测试结果及分析 11](#_Toc366347522)

[4.3.1测试结果(数据) 11](#_Toc366347523)

[4.3.2测试分析与结论 12](#_Toc366347524)

[5总结 12](#_Toc366347525)

[附件1：系统电路原理图 14](#_Toc366347526)

[附件2：电机控制源程序 15](#_Toc366347527)

**简易旋转倒立摆及控制装置（C题）**

**【本科组】**

# 1系统方案

本系统主要由主控模块、角度传感器模块、机械结构模块、电源模块等几部分组成，下面分别论证这几个模块的选择。

## 1.1 角度传感器模块的论证与选择

方案一：采用绝对值式旋转编码器。绝对值式编码器的每一个位置对应一个确定的数字码，因此它的示值只与测量的起始和终止位置有关，而与测量的中间过程无关。但是绝对值式编码器体积和重量不太理想，安装后可能会对电机有较大的干扰，而且绝对值编码器相对昂贵，性价比不高。

方案二：采用增量式编码器。增量式编码器是将位移转换成周期性的电信号，再把这个电信号转变成计数脉冲，用脉冲的个数表示位移的大小。增量式编码器不能定位，但是O位的确定可以通过软件解决，而且增量式编码器价格低廉易得，反馈的信号可以满足题目的要求。

综合考虑以上两种方案，选择方案二，采用增量式编码器。

## 1.2电机的论证与选择

方案一：采用步进电机。步进电机将电脉冲信号转变为角位移，电机的转速，停止的位置只取决于脉冲信号的频率和脉冲数，因此可以通过控制脉冲频率来控制电机转动的速度和加速度。但是步进电机在高速运转下，开环会产生丢步的情况，开环控制情况不尽如人意，不是首选的电机。

方案二：采用360度旋转舵机。舵机为随动系统，运动时可以外接较大的转动负载，输出扭矩大，而且抗抖动性很好。360度旋转舵机内置驱动电路，用单片机输出的信号可以直接驱动，接线方便，但是舵机的转速普遍很低，加速度不能达到理想的要求。

方案三：采用带速度闭环的直流电机。我们对于直流电机的数学模型较为熟悉，而且能够比较精确地控制位置，速度，和加速度，只要在选择直流电机型号的时候注意选择额定参数较大的电机，能够输出足够的转矩带动负载就可以达到要求。而且带速度闭环的直流电机可以

综合考虑采用方案三。

## 1.3 电机驱动模块的论证与选择

方案一：采用购买的L298N集成芯片，该集成电路带散热器和光电耦合器，可以保护单片机不受电机的影响，最大程度的保护单片机。但是经过我们连线调试之后，发现该种芯片一个致命的缺点，在PWM输入达到KHZ级别的时候，输出端波形严重畸变甚至直接输出高电平，导致电路不能运行在所给的频率下，经分析可能是光耦本身截止频率就在KHZ级别，一端输出大电压之后，带宽降低，不能满足要求。

方案二：采用L298N芯片，自己制作L298N驱动模块，因为L298N驱动电路较简单，外围元器件低廉易得。在将自己制作的L298N电路连线调试之后，我们发现完全可以在程序要求的频率下工作，虽然死区电压有些不近人意，低速是电机会有抖动现象，但是总体来说，情况比较乐观。

综合考虑采用方案二。

主控模块采用XS128最小系统板，因为XS128作为飞思卡尔常用芯片，其PWM波有8个输出通道，

# 2理论分析与计算

## 2.1 机械结构设计的分析

### 2.1.1 摆杆

由于直流电机的扭矩输出并不是很大，因此要求负载很轻甚至能够忽略为空载，可以为计算和控制带来很大的简便，因此选择了力学性能优异的新材料——碳纤维，它的比重不到钢的1/4，但强度是钢的7~9倍，抗拉[弹性模量](http://baike.baidu.com/view/30660.htm" \t "_blank)为230~430Gpa亦高于钢。而且中空碳纤维摆杆，是比较理想的摆杆模型。

### 2.1.2 底座

由于自己加工制作的连接不能精确的达到完全对称，导致电机不在底座的中央，因此我们选择质量较大的钢材来做底座，并焊接一个港版来固定底座，防止倾倒。然后用两个固定夹固定在桌子边缘。经过测试，可以承受电机高速旋转带来额离心力的作用。

## 2.2电机物理模型的分析与计算

### 2.2.1 直流电机的机械特性

直流电机在稳态运行下，有下列方程式：

 公式1

 公式2

 公式3

### 2.2.2 直流电机的动态特性分析

电机传动系统的运动方程式为：

 公式4

机械特性方程式为：

 公式5

由公式1和公式2可得出转速随时间变化的规律为：

公式6

由上述分析可以得出结论：直流电机的物理模型为一阶惯性系统。

### 2.2.3 直流电机加速度分析与计算

由公式4求导可以得到：

 公式7

上式中加速度为a，公式4经变换可得到加速度的表达式：

 公式8

再结合直流电机机械特性的三个方程式可以得到电机电压与加速度的关系如下式所示：

 公式9

## 2.3摆杆运动状态的分析及控制思路

### 2.3.1摆杆起振

摆杆从静止到振荡是一个需要摆动幅度逐渐增加的过程。而对于电机的控制，是通过检测到反馈到单片机的摆杆摆动的角度，然后单片机输出PWM控制一个小的加速度给电机，这个加速度摆杆向反方向摆动更大的角度。

基于以上的程序设计，结合分析了直流电机的物理模型之后，经过了一系列简化计算，我们认为电机的加速度几乎正比于控制电机的电压，电压大小可以通过单片机输出的PWM波的占空比的改变而改变的。因此我们根据反馈的摆动角度，算出响应的加速度控制电机加速转动，这样摆动角度越大，摆杆收到电机所给的加速度越大，形成一个正反馈环节，使摆杆逐步摆动幅度增大，实现起振。

特意要说明的是，由于摆杆在自然下垂状态时检测到的角度为0，因此按照以上推论，此时单片机输出的加速度为0，这样摆杆永远无法起振。为了解决这个问题，我们在摆杆处于自然下垂状态摆杆处于0位时，我们在程序中在此刻增加一个很小的角度，使得即使在0位，也可以通过控制算法输出一个小的控制量，来实现最初的起振。

### 2.3.2摆杆做圆周运动

当摆杆起振后由于正反馈环的存在，电机通过旋转臂给摆杆做功，摆杆不断获取能量，摆杆摆杆摆动幅度越来越大，直到超过180度，能够完成一系列圆周运动。

### 2.3.3摆杆达到倒立平衡状态

当摆杆能够摆到接近于180度的位置时（正负大约15度），希望摆杆能够趋向平衡位置（180度位置），并动态保持平衡。

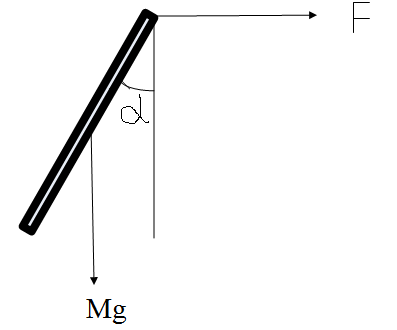
要实现上述功能，我们首先只开启摆杆的角度反馈，不给电机控制电压，然后用手转动摆杆，找到第一个Z相信号，并将之定在180的平衡位置上。然后每经过一个Z相信号，都会将角度标准重新标定一下（标定0度），之后我们发现这个标定很有必要，因为摆杆在最低点的往复摆中，由于方向时刻改变，而测速脉冲（编码器A相信号和B相信号异或之后的信号，相当于二分频）在方向不同是会有一个相移，导致测速脉冲有一个累积误差。经过实验，随着摆动次数增加，累积误差可高达10几度，导致零位严重偏移。而Z相信号的标定可以弥补这个误差。

在做完标定之后，我们将程序设计为一开始就通过PID调节给电机一个最大的加速度，使摆杆尽量能在一个摆动周期内摆至180度平衡位置附近，然后结束使摆杆幅度加大的正反馈环节，开始进入到所加的加速度随着摆杆与180度平衡位置夹角的大小成正比，但加速度的方向与夹角变化方向成反比的程序中。

## 2.4 摆杆控制量的分析与计算

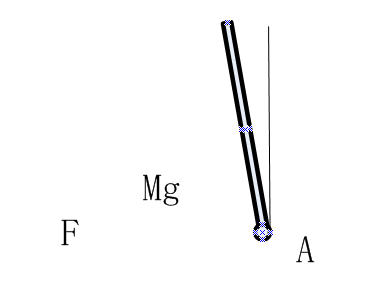
### 2.4.1 摆杆起振状态分析

假设某一时刻，摆杆向左摆到了α度，此时摆杆的受力状态如图X所示：



A点为摆杆与旋转臂连接处，设此时摆杆收到旋转臂的力为F，方向如图所示，此时力F的作用是使摆杆继续向左摆起，即增大摆杆的摆动复读。因此可以根据此受力分析给电机一致方向的角加速度。

### 2.4.2 摆杆进入开始调整倒立平衡的角度范围



假设图中角度为β，则想要使摆角趋向180度，要给摆杆一个图示方向的力F使得下式成立：

 公式10

由于考虑的连轴的摩擦力，转动时收到的空气阻力，外部接线对摆杆的干扰力等等，增加一个阻尼力，使得上式变为：

 公式11

在算出了角速度的控制量范围之后，需要确定单片机提供的相应控制量。为了比较接近实际的物理情况，我们应用2.2中分析和计算的直流电机物理模型，以及直流电机控制电压与加速度的关系，即公式9,。再结合我们手头可以得到的关于直流电机的各种参数，我们很难精确地算出时间常数，只能近似，而且再加上程序的控制周期远小于时间常数，综合考虑之后，我们选择了忽略公式9中等号右边的后两项变量，也就是近似认为直流电机的控制电压与加速度成正比关系，这样便于单片机给出控制信号。但是经过实际的实验之后，我们发现，用PD调节来控制电压信号，可以使摆杆处于倒立平衡状态，但是相当不稳定，还经常出现电机高速旋转的情况。

因此，我们在此基础上加入了旋转臂的速度反馈。之前我们在模型的建立上忽略了公式9等号右边的后两项，造成了比较大的误差。现在加入了旋转臂的速度反馈之后，就可以计算出了，可以只忽略最后一项，大大减小了误差。同时经过实验证明，双环控制的效果能达到大部分的性能指标。

# 3电路与程序设计

## 3.1电路的设计

### 3.1.1系统总体框图

整体系统由机械结构，XS128单片机最小系统板，电机驱动模块，电源模块，异或门，计数器和D触发器，以及带速度反馈的直流电机和增量式编码器组成。

系统总体框图如图1所示，



图1 系统总体框图

### 3.1.2 系统电路原理图

见附件2

### 3.1.3 电机驱动模块的电路设计

如图2所示

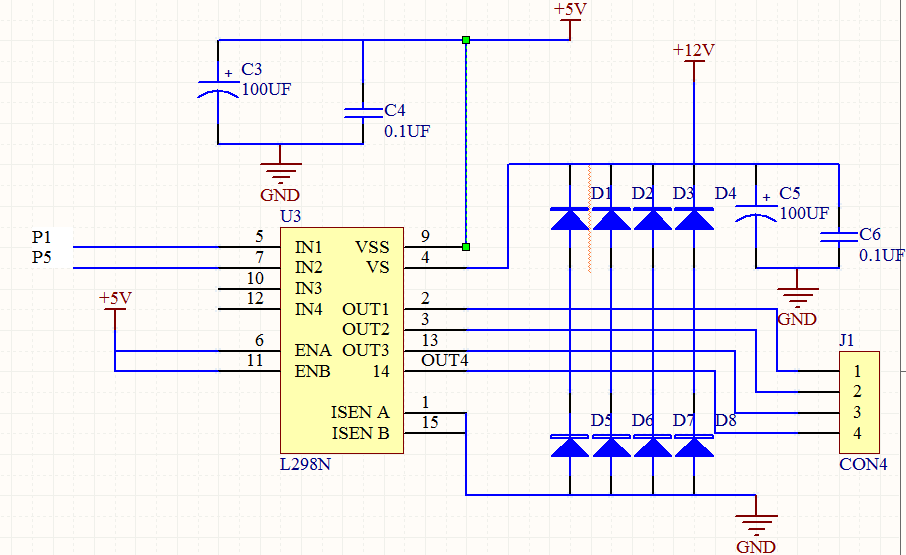


图2 电机驱动模块的电路

## 3.2程序的设计

### 3.2.1程序功能描述与设计思路

1、程序功能描述

根据题目要求，本系统程序的任务如下：对检测摆杆位置的增量编码器1的输入信号进行计数和方向识别，根据摆杆的摆角发出PWM波，对电机进行控制，同时接受外部计数器对旋转臂速度检测反馈回来的信息，综合两个反馈，根据基于物理模型得到的电机控制电压计算公式，控制电机使给摆杆一个加速度，不同的控制算法能够实现相应的功能，如起振，圆周运动，倒立平衡等。

1. 电机控制程序内容：

见附件3

### 3.2.2程序流程图

1、主程序与子程序框图

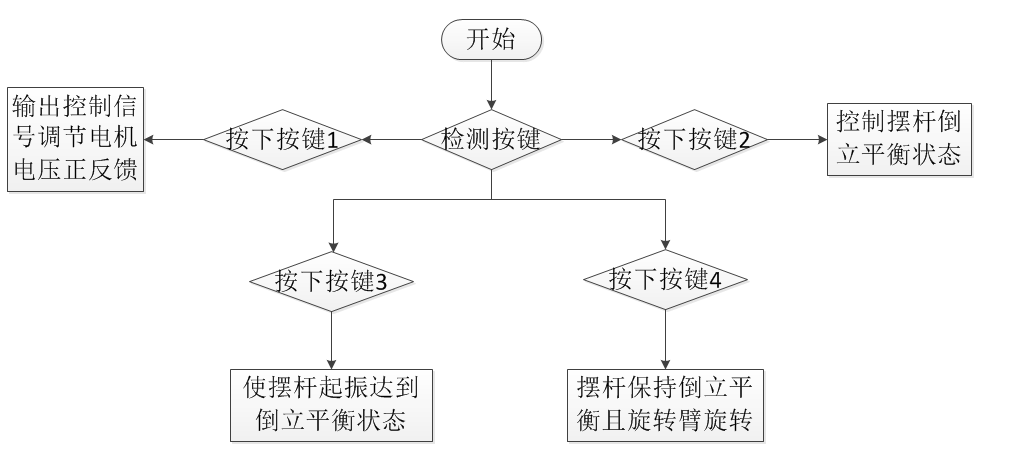


图3 主程序与子程序框图

2、“起摆做圆周运动”子程序流程图



图4 “起摆做圆周运动”子程序流程图

3、“起摆并尽快达到稳态”子程序流程图



图5 “起摆并达到倒立平衡状态”子程序流程图

# 4测试方案与测试结果

## 4.1测试方案

硬件软件联调：调试时，下载线连接单片机，以便程序执行后实时观测控制算法中多个变量，如摆杆角度，方向，电机速度以及加速度等等。

通过拨码开关来决定运行不同的程序，以实现不同的功能。相互功能之间没有干扰。实现另一个功能需要单片机复位后重新运行。记录下每次实现不同功能的性能指标。

## 4.2 测试条件与仪器

测试条件：支架水平放置于桌面上，保证摆杆处于自然下垂状态，除电机外无其他外力对摆杆造成干扰（空气阻力，摩擦力等不算在内）。

测试仪器：直流稳压电源，数字示波器，数字万用表，秒表，量角器.

## 4.3 测试结果及分析

### 4.3.1测试结果(数据)

测试功能一：摆杆起振

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试次数 | 1 | 2 | 3 | 4 | 5 | 6 |
| 摆动幅度  （度） | >60 | >60 | >60 | >60 | >60 | >60 |
| 摆动时间（s） | 2 | 1 | 2 | 3 | 5 | 4 |

测试功能二：摆杆完成圆周运动

说明：以完整的完成一个圆周运动的时间为计时标准。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 测试次数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 完成时间  （s） | <2 | <2 | <2 | <2 | <1 | <2 | <1 | >1 |

测试功能三：摆杆尽快达到倒立平衡状态并保持5s以上，且旋转臂旋转的转动角度不大于90度。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 测试次数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 保持时间  （s） | >5 | >5 | <5 | >5 | >5 | <5 | >5 | >5 |
| 旋转臂最大转动角 （度） | <90 | <90 | >90 | <90 | >90 | >90 | <90 | >90 |

测试功能四：从摆杆处于自然下垂状态开始，控制旋转臂做往复旋转运动，尽快使摆杆摆起倒立，保持倒立时间不小于10s

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 测试次数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 摆起时间  （s） | 2 | 3 | -- | 1 | 2.5 | 2．5 | 1 | 4 |
| 保持时间（s） | >10 | >10 | 0 | >10 | >10 | >10 | >10 | >10 |

测试功能五：在摆杆保持倒立状态下，施加干扰后摆杆能继续保持倒立或2s内恢复倒立状态。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 测试次数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 能否保持 | 能 | 能 | 能 | 能 | 能 | 能 | 能 | 否 |
| 恢复时间（s） | 2 | 3 | 4 | 3 | 3 | 4 | 5 | 4 |

测试功能六：在摆杆保持倒立状态的前提下，旋转臂做圆周运动，并尽快使单方向转过角度达到或超过360度。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 测试次数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 旋转臂转过360度时间（s） | >5 | >5 | >5 | >5 | >5 | >5 | >5 | >5 |

### 4.3.2测试分析与结论

根据上述测试数据，该系统基本上能够达到基本部分和发挥部分(1)(3)的性能指标，由此可以得出以下结论：

1、电机的性能不是很理想，联轴器有些松动，长时间运行两者有些脱节，起振达到平衡很困难，

2、双环控制需要时间上的配合，加入的时间和参数很重要。

3、好的硬件电路时成功的奠基石。

综上所述，本设计基本达到设计要求。

# 5总结

本次比赛中我们组遇到的最大的困难是硬件问题，包括合适的电动机的选型，先后换了三种型号的电动机，电动机与底座和旋转臂的连接，以及能够经受住长时间离心力作用的联轴器和钢珠等支承结构。四天三夜中很多时间在组建和检修硬件部分，导致浪费了很多调试程序的时间，东奔西跑地寻找合适的材料也浪费了很多体力。在断断续续的将硬件做好之后，在调试软件时我们一直采用的程序设计思路在我们攻克倒立平衡的稳定性时，遇到了很大的挑战，我们在这个程序思路上下了很多功夫，但是调整结果很不满意。在第四天的下午，离比赛结束还有几个小时时，我们放弃了一直坚持的程序主线——将加速度环作为内环，速度环作为外环进行控制，转而采用了将速度环作为内环，而加速度环作为外环，消除了速度的采样滞后，才取得了比较可喜的进步，但是由于时间实在太有限，我们没有机会将所有部分都完美的移植在新的程序思路上。遗憾是有的，但是收获也是巨大的，四天三夜，体验到了很多东西，团队的力量，沟通的重要性，最重要的是遇到困难能够不被得失的情绪所干扰，而是一心一意去解决问题，这是我们以后对待任何困难的一种成功的心态。

在这比赛的最后，要感谢学校和学院老师的给力的后勤工作，使我们能够在不被烦扰的情况下，用最充足的时间完成比赛。感谢全国大学生电子设计大赛的组委会，提供这样优秀的平台和机会给我们充足的锻炼。

# 附件1：系统电路原理图

P0

P0

P1

P1

P2

P2

P3

P3

J0

J0

J1

J1

K0

K0

K1

K1

K2

K2

K3

K3

T0

T0

T1

T1

T2

T2

T3

T3

T4

T4

T5

T5

T6

T6

T7

T7

K4

K4

K5

K5

B0

B0

B1

B1

B2

B2

B3

B3

B4

B4

B5

B5

B6

B6

B7

B7

P7

P7

P6

P6

P5

P5

P4

P4

K7

K7

NC

NC

M7

M7

M6

M6

M5

M5

M4

M4

M3

M3

M2

M2

M1

M1

M0

M0

J7

J7

J6

J6

S7

S7

S6

S6

S5

S5

S4

S4

S3

S3

S2

S2

S1

S1

S0

S0

+5

+5

+5

+5

GND

GND

GND

GND

AD15

AD15

AD14

AD14

AD13

AD13

AD12

AD12

AD11

AD11

AD10

AD10

AD9

AD9

AD8

AD8

AD7

AD7

AD6

AD6

AD5

AD5

AD4

AD4

AD3

AD3

AD2

AD2

AD1

AD1

AD0

AD0

A7

A7

A6

A6

A5

A5

A4

A4

A3

A3

A2

A2

A1

A1

A0

A0

VRH

VRH

GND

GND

GND

GND

+5

+5

H7

H7

H6

H6

H5

H5

H4

H4

E7

E7

E6

E6

E5

E5

E4

E4

H3

H3

H2

H2

H1

H1

H0

H0

E3

E3

E2

E2

E1

E1

E0

E0

9S12XS/E-P112

1

2

3

4

5

6

7

8

S1

²¦Âë¿ª¹Ø

X0

12

X1

14

X2

15

X3

11

Y0

1

Y1

5

Y2

2

Y3

4

INH

6

A

10

B

9

VEE

7

X

13

Y

3

U2

CD4052

R1

20k

R2

20k

R3

20k

R4

20k

+5

GND

1

VIN+

5

ON/OFF

3

GND

4

FEEDBACK

2

OUTPUT

LM2596¿ª¹Øµç

Ñ¹µ÷½ÚÄ£¿é

U2

+

C1

680UF

D1

IN5824

L

33u

+

C2

220UF

+12VÖ±Á÷ÊäÈë

-12VÖ±Á÷ÊäÈë

+5VÊä³ö

GND

VR?

VR

IN1

5

IN2

7

IN3

10

IN4

12

ENA

6

ENB

11

VSS

9

VS

4

OUT1

2

OUT2

3

OUT3

13

14

OUT4

ISEN A

1

ISEN B

15

U3

L298N

D1

D2

D3

D4

D5

D6

D7

D8

+12V

GND

GND

GND

+

C3

100UF

C4

0.1UF

C6

0.1UF

+

C5

100UF

+5V

+5V

1

2

3

4

J1

CON4

1

IRd/

2

1D

3

1CP

4

ISd

5

1Q

6

1Q/

7

GND

14

VSS

13

2Rd/

12

2D

11

2CP

10

2Sd

9

2Q

8

2Q/

\*

Component\_1

P1

P5

1

NC

2

VCC

3

GND

4

NC

5

A/

6

A

7

B/

8

B

9

I/

10

I

1

189

2

+5v

3

GND

4

192

5

193

6

P7

7

195

8

P6

9

197

10

P4

U5

2

1B

3

1Y

4

2A

5

2B

6

2Y

7

GND

1

1A

14

VCC

13

4B

12

4A

11

4Y

10

3B

9

3A

8

3Y

U6

74LS86Òì»òÃÅ

# 附件2：电机控制源程序

子函数：

#include "carsub.h"

#include <hidef.h>

#include "derivative.h"

void GetMotorPulse(void)

{

unsigned int nLeftPulse,nRightPulse;

unsigned char leftdir,rightdir;

//unsigned char ii;

leftdir = PORTB & 0x01; //读取转向

rightdir = PORTB & 0x04;

nLeftPulse = PACNT; //left为内部计数器，right为外部计数器

nRightPulse = PORTA; //读取脉冲数

PACNT = 0;

PTJ=PTJ|0XFF;

Delay(15);

PTJ=PTJ&0X00; //计数器清零

if(leftdir == 0) {

LeftMotorPulse -= (int)nLeftPulse;

LeftMotorPulseSigma -= (int)nLeftPulse;

}

else {

LeftMotorPulse += (int)nLeftPulse;

LeftMotorPulseSigma += (int)nLeftPulse;

}

if(rightdir == 0)

RightMotorSpeed = -(int)nRightPulse;

else

RightMotorSpeed = (int)nRightPulse;

//PACNT = 0;

//if(leftdir == 0)

g\_lPosition = LeftMotorPulse;

}

void SetMotorVoltage(int nLeftVal)

{

if(nLeftVal>MOTOR\_OUT\_MAX)

nLeftVal=MOTOR\_OUT\_MAX;

if(nLeftVal<MOTOR\_OUT\_MIN)

nLeftVal=MOTOR\_OUT\_MIN;

if (nLeftVal > 0)

{

PWMDTY45 = 0;

PWMDTY01 = nLeftVal + 120;

}

else

{

PWMDTY45 = -nLeftVal + 120;

PWMDTY01 = 0;

}

}

void MotorOutput(void)

{

//计算最终给定的PWM占空比

//int LeftMotorOutput;

//float K,P;

//if (PORTB\_PB3 == 1)

// K = 350;

//else K = 700;

LeftMotorOutput = (int)K\*PositionControlOut + SpeedControlOut;

if (PORTB\_PB4 ==0) {

if ((LeftMotorSpeed <= 0)&&(LeftMotorSpeedOld > 0))

{

g\_ucRight = 1;

g\_ucLeft = 0;

}

if ((g\_ucRight == 1)&&((GravityAngle > 0)&&(GravityAngle < 60)))

{

LeftMotorOutput = 0 ;

//SPEED\_CONTROL\_P = 0;

//SPEED\_CONTROL\_D = 0;

}

if ((LeftMotorSpeed >= 0)&&(LeftMotorSpeedOld < 0))

{

g\_ucLeft = 1;

g\_ucRight = 0;

}

if ((g\_ucLeft == 1)&&((GravityAngle < 0)&&(GravityAngle > -60))) {

LeftMotorOutput = 0;

}

}

if (PORTB\_PB3 == 0) {

if ((LeftMotorSpeed <= 0)&&(LeftMotorSpeedOld > 0))

{

g\_ucRight = 1;

g\_ucLeft = 0;

}

if ((g\_ucRight == 1)&&((GravityAngle >=0)&&(GravityAngle <= 180)))

LeftMotorOutput = -LeftMotorOutput;

if ((LeftMotorSpeed >= 0)&&(LeftMotorSpeedOld < 0))

{

g\_ucLeft = 1;

g\_ucRight = 0;

}

if ((g\_ucLeft == 1)&&((GravityAngle <= 0)&&(GravityAngle >= -180)))

LeftMotorOutput = -LeftMotorOutput;

}

SetMotorVoltage(LeftMotorOutput);

}

void PositionControl(void)

{

float fP,fD;

POSITION\_CONTROL\_P = 12;

POSITION\_CONTROL\_D = 0.002;

if(GravityAngle < 0) {

//if (GravityAngle > -45 ) fP = -15;

//if(GravityAngle < -180) fP = -POSITION\_CONTROL\_P\*tancode[-GravityAngle - 180];

//else

fP = -POSITION\_CONTROL\_P\*tancode[-GravityAngle];

}

//else if(GravityAngle > 180) {

//if(GravityAngle > 315)

// fP = -15;

//else

//fP = POSITION\_CONTROL\_P\*tancode[GravityAngle - 180];

// }

else {

//if(GravityAngle < 45)

// fP =15;

//else

fP = POSITION\_CONTROL\_P\*tancode[GravityAngle];

}

fD = POSITION\_CONTROL\_D\*AngleSpeed;

//fI1 += POSITION\_CONTROL\_I\*PositionErr;

PositionControlOld = PositionControlNew;

PositionControlNew = fP + fD;//ＰＩＤ得到的是角度控制量，加减速由车身的前倾角度决定

}

void SpeedControl(void)

{

float fP,fI,fD;

//RightMotorSpeedGoal = 0;

//RightMotorSpeed = RightMotorPulseSigma;

//RightMotorPulseSigma = 0;

RightMotorSpeedErr0 = RightMotorSpeedErr1;

RightMotorSpeedErr1 = RightMotorSpeedErr2;

RightMotorSpeedErr2 = RightMotorSpeedGoal - RightMotorSpeed;//计算速度误差

if((PORTB\_PB4 == 0) && (g\_ucBegin == 0))

{

SPEED\_CONTROL\_P = 0;

SPEED\_CONTROL\_I = 0;

}

else

{

SPEED\_CONTROL\_P = 120;

SPEED\_CONTROL\_I = 1;

}

//if(g\_ucBegin == 0) {

//SPEED\_CONTROL\_P = 0;

//SPEED\_CONTROL\_I = 0;

//SPEED\_CONTROL\_D = 0;

//}

fP = SPEED\_CONTROL\_P\*(RightMotorSpeedErr2 - RightMotorSpeedErr1);

fI = SPEED\_CONTROL\_I\*RightMotorSpeedErr2;

fD = SPEED\_CONTROL\_D\*(RightMotorSpeedErr2 - 2\*RightMotorSpeedErr1 + RightMotorSpeedErr0);

//SpeedControlOld = SpeedControlNew;

SpeedControlOut += fP + fI + fD;

}

void AngleCalculate(void)

{

LeftMotorSpeedOld = LeftMotorSpeed;

LeftMotorSpeed = LeftMotorPulseSigma;

LeftMotorPulseSigma = 0;

GravityAngle = ((int)(GRAVITY\_ANGLE\_RATIO\*g\_lPosition))%360;

if(GravityAngle > 180)

GravityAngle = GravityAngle - 360;

AngleSpeed = ANGLE\_SPEED\_RATIO\*LeftMotorSpeed;

}

void Delay(int num)

{

int delay\_c;

for(delay\_c=0;delay\_c<num;delay\_c++);

}

void PositionSetOut(void)

{

float fvalue;

fvalue = PositionControlNew - PositionControlOld;

PositionControlOut = fvalue\*(PositionControlCount + 1)/POSITION\_CONTROL\_COUNT + PositionControlOld;

}

主函数：

#include <hidef.h>

#include "derivative.h"

#include "init.h"

#include "carsub.h"

const float tancode[181]=

{

0.010,0.0175,0.0349,0.0524,0.0699,0.0875,0.1051,0.1228,0.1405,0.1584,0.1763,0.1944,0.2126,

0.2309,0.2493,0.2679,0.2867,0.3057,0.3249,0.3443,0.3640,0.3839,0.4040,0.4245,0.4452,0.4663,

0.4877,0.5095,0.5317,0.5543,0.5774,0.6009,0.6249,0.6494,0.6745,0.7002,0.7265,0.7536,0.7813,

0.8098,0.8391,0.8693,0.9004,0.9325,0.9657,1.0000,1.0355,1.0724,1.1106,1.1504,1.1918,1.2349,

1.2799,1.3270,1.3764,1.4281,1.4826,1.5399,1.6003,1.6643,1.7321,1.8040,1.8807,1.9626,2.0503,

2.1445,2.2460,2.3559,2.4751,2.6051,2.7475,2.9042,3.0777,3.2709,3.4874,3.7321,4.0108,4.3315,

4.7046,5.1446,5.6713,6.3138,7.1154,8.1443,9.5144,10.0000,10.0000,10.0000,10.0000,10.0000,

10.0000,-10.0000,-10.0000,-10.0000,-10.0000,-10.0000,-9.5144,-8.1443,-7.1154,-6.3138,-5.6713,

-5.1446,-4.7046,-4.3315,-4.0108,-3.7321,-3.4874,-3.2709,-3.0777,-2.9042,-2.7475,-2.6051,-2.4751,

-2.3559,-2.2460,-2.1445,-2.0503,-1.9626,-1.8807,-1.8040,-1.7321,-1.6643,-1.6003,-1.5399,-1.4826,

-1.4281,-1.3764,-1.3270,-1.2799,-1.2349,-1.1918,-1.1504,-1.1106,-1.0724,-1.0355,-1.0000,-0.9657,

-0.9325,-0.9004,-0.8693,-0.8391,-0.8098,-0.7813,-0.7536,-0.7265,-0.7002,-0.6745,-0.6494,-0.6249,

-0.6009,-0.5774,-0.5543,-0.5317,-0.5095,-0.4877,-0.4663,-0.4452,-0.4245,-0.4040,-0.3839,-0.3640,

-0.3443,-0.3249,-0.3057,-0.2867,-0.2679,-0.2493,-0.2309,-0.2126,-0.1944,-0.1763,-0.1584,-0.1405,

-0.1228,-0.1051,-0.0875,-0.0699,-0.0524,-0.0349,-0.0175,0.0000

};

//const float

float POSITION\_CONTROL\_P,POSITION\_CONTROL\_D,POSITION\_CONTROL\_I;

float SPEED\_CONTROL\_P,SPEED\_CONTROL\_I,SPEED\_CONTROL\_D;

float GRAVITY\_ANGLE\_RATIO = 0.5,ANGLE\_SPEED\_RATIO = 100;

float K=650;

unsigned char g\_ucBegin,g\_ucRight,g\_ucLeft;

unsigned char PositionControlCount;

int LeftMotorPulse;

//int g\_iZOffset;

int g\_lPosition = 0,g\_lPositionBase = 0;

//long g\_lPositionGoal;

//float PositionControlNew,PositionControlOld;

float PositionControlOut,PositionControlNew,PositionControlOld,SpeedControlOut;

//int PositionErr = 5500,PositionErrOld;

float LeftMotorSpeed,LeftMotorSpeedOld;

int RightMotorPulseSigma,LeftMotorPulseSigma,RightMotorSpeed,RightMotorSpeedGoal,RightMotorSpeedGoal1;

float RightMotorSpeedErr0,RightMotorSpeedErr1,RightMotorSpeedErr2;

int LeftMotorOutput;

//float fI1 = 0;

int GravityAngle;

float AngleSpeed;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*主程序\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{ //总线频率设置为80Mhz

PLL\_init();

PIT\_init(); //定时器中断初始化

PWM\_init();

TIM\_init();

GPIO\_init();

XINT\_init();

g\_ucBegin = 0; //初始化Z信号尚未到来

EnableInterrupts;

for(;;)

{

//if(PORTB\_PB1 == 1)

// EnableInterrupts;

}

}

#pragma CODE\_SEG \_\_NEAR\_SEG NON\_BANKED //中断函数置于非分页区内

void interrupt VectorNumber\_Vpit0 PIT0Interrupt(void)

{

EnableInterrupts;

GetMotorPulse(); //获取编码器脉冲

PositionControlCount ++;

if(PositionControlCount == POSITION\_CONTROL\_COUNT)

{

if(RightMotorSpeedGoal <RightMotorSpeedGoal1)

RightMotorSpeedGoal += 0.1;

AngleCalculate();

PositionControl();

PositionControlCount = 0;

}

//SpeedSetOut();

PositionSetOut();

//PositionControl();

SpeedControl();

if((PORTB\_PB1 == 0) || (PORTB\_PB3 == 0) || (PORTB\_PB4 == 0))

MotorOutput();

PITTF\_PTF0=1;

EnableInterrupts;

}

void interrupt VectorNumber\_Vtimch0 TIM0Interrupt(void)

{

//EnableInterrupts;

LeftMotorPulse = 360;

if(PORTB\_PB4 == 0) {

if ((LeftMotorSpeed <2) &&(LeftMotorSpeed > -2) ) {

SPEED\_CONTROL\_P = 30;

SPEED\_CONTROL\_I = 0.3;

g\_ucBegin = 1;

RightMotorSpeedGoal1 = 20;

}

g\_ucRight= 0;

g\_ucLeft = 0;

}

TFLG1\_C0F = 1;

}

//#pragma CODE\_SEG DEFAULT //后续代码置于默认区域内