2013年全国大学生电子设计竞赛

设计（论文）报告

简易旋转倒立摆及控制装置

****

2013年9月7日星期六

**摘 要：**本简易旋转倒立摆及控制装置采用由Freescale半导体公司生产的MC9S12XS128MAA单片机作为主控系统，采用BTS7960作为电机驱动。通过MPU6050加速度计陀螺仪模块采集到摆杆角度和角加速度，主控芯片处理采集到的信号并给定PWM控制直流电机旋转速度和角度，同时对使用光电编码器采集电机转动状态，对电机实现闭环控制，从而使得摆杆处于不同位置，实现简易旋转倒立摆控制。该系统硬件部分主要包括MC9S12XS128MAA单片机，BTS7960电机驱动电路，MPU6050加速度陀螺仪模块，电源模块等。本设计主要运用MC9S12XS128MAA单片机控制直流电机，根据摆杆的状态，通过程序控制算法，使直流电机带动旋转臂来控制摆杆摆动并保持一定平衡。

**关键词：**MC9S12XS128MAA单片机；电机驱动；加速度陀螺仪；电机控制

[1 设计任务与要求 4](#_Toc366348115)

[1.1 **设计任务** 4](#_Toc366348116)

[1.2 **设计要求** 4](#_Toc366348117)

[2 系统方案比较与论证 5](#_Toc366348118)

[2.1 **系统主控芯片的选择** 5](#_Toc366348119)

[2.2 **系统结构分析** 6](#_Toc366348120)

[3 理论分析与计算 7](#_Toc366348121)

[3.1 **电动机选型** 7](#_Toc366348122)

[3.2 **摆杆状态检测** 7](#_Toc366348123)

[3.3 **电动机驱动方案的比较与选择** 7](#_Toc366348124)

[3.4 **控制算法** 8](#_Toc366348125)

[4 系统电路设计 9](#_Toc366348126)

[4.1 **总体电路框图** 9](#_Toc366348127)

[4.2 **单元电路设计** 9](#_Toc366348128)

[4.3 **全部电路图** 12](#_Toc366348129)

[5 系统软件设计 13](#_Toc366348130)

[5.1 **系统软件设计流程图** 13](#_Toc366348131)

[5.2 **各模块软件设计** 14](#_Toc366348132)

[6 系统调试与测试 19](#_Toc366348133)

[6.1 **系统调试** 19](#_Toc366348134)

[6.2 **测试仪器** 19](#_Toc366348135)

[6.3 **测试指标** 20](#_Toc366348136)

[6.4 **测试结果分析** 20](#_Toc366348137)

[7 设计总结 21](#_Toc366348138)

[参考文献 21](#_Toc366348139)

**设计题目**

# 设计任务与要求

**设计任务**

1、设计并制作一套简易旋转倒立摆及其控制装置，旋转倒立摆的结构如图1-1所示。

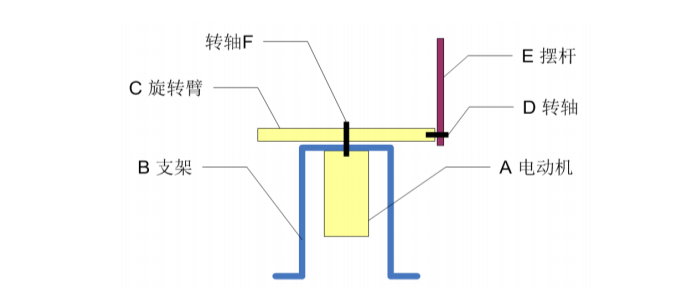


图1-1 旋转倒立摆结构示意图

**设计要求**

1. 基本要求
2. 摆杆从处于自然下垂状态（摆角0°）开始，驱动电机带动旋转臂做往复旋转使摆杆摆动，并尽快使摆角达到或超过-60°~+60°；
3. 从摆杆处于自然下垂状态开始，尽快增大摆杆的摆动幅度，直至完成圆周运动；
4. 在摆杆处于自然下垂状态下，外力拉起摆杆至165°位置，外力撤除同时，启动控制旋转臂使摆杆保持倒立状态时间不少于5s；期间旋转臂的转动角度不大于90°。
5. 发挥部分
6. 从摆杆处于自然下垂状态开始，控制旋转臂往复旋转运动，尽快使摆杆摆起倒立，保持倒立状态时间不少于10s；
7. 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；
8. 在摆杆保持倒立状态的前提下，旋转臂作圆周运动，并尽快使单方向转过角度达到或超过360°；
9. 其他

# 系统方案比较与论证

**系统主控芯片的选择**

主控芯片的选择范围非常广泛，目前技术开发过程中，主要使用DSP、FPGA、单片机以及ARM作为主控芯片，其中单片机使用相对较多，因此本系统主控芯片的选择主要是单片机系列，如以下三种方案：

方案一：采用STC89C52单片机作为主控芯片，STC89C52单片机是STC89C51单片机的增强板，是STC公司生产的一种低功耗、高性能8位微控制器，具有8k可编程Flash存储器，总共有32 位I/O 口线， 3个16 位定时器/计数器，4个外部中断，全双工串行口等资源。但是主频低，编程繁琐，外设资源少。

方案二：采用STM32F103ZET6作为主控芯片，STM32系列单片机是基于专为要求高性能、低成本、低功耗的嵌入式应用专门设计的ARM Cortex-M3内核。它具有512K 片内FLASH,64K片内RAM ,片内FLASH 支持在线编程(IAP)，多达80个IO(大部分兼容5V逻辑),4个通用定时器,2个高级定时器,2个基本定时器,3路SPI接口,2路I2S 接口,2路I2C接口,5路USART等丰富的资源，但是该芯片开发周期长，编程复杂，对底层库函数使用和外设要求高。

方案三：采用MC9S12XS128单片机作为主控芯片，MC9S12XS128是由飞思卡尔半导体公司生产的16位单片机，其工作频率为40MHZ,该单片机具有128KB程序Flash、8KB RAM和8KB数据Flash(D-lash),主要功能模块包括：内部存储器、PLL锁相环模块、2个异步串口通信、一个串行外设接口、1个8通道I/O比较定时器模块、中断定时器模块、16通道A/D转换模块、1个8通道PWM模块、多个I/O模块等

方案比较与论证：基于上述对每种主控芯片的简介，方案一中STC89C52单片机具有工作频率低，可用资源少等缺点，因此本系统不予以采用。方案二采用STM32F103ZET6作为主控芯片,其资源和工作频率远远超过该系统要求，存在资源浪费，且编程复杂，开发周期长，因此本系统不采用该芯片作为主控芯片。方案三，使用MC9S12XS128单片机作为主控芯片，该芯片相对于方案一的STC89C52单片机，其资源和工作频率丰富，满足系统要求。该芯片相对于方案二的STM32F103ZET6单片机，其具有自动生成代码功能，开发周期短，编程简单，不会存在资源浪费。基于上述分析考虑，本系统采用方案三。

**系统结构分析**

通过上述分析可知，本系统采用MC9S12XS128单片机作为主控芯片。考虑到简易旋转倒立摆装置的控制需求，该系统主要由单片机最小系统板、电机驱动模块、摆杆状态监测模块、蓝牙通信模块、上位机显示组成。系统方框图如图2-1所示。通过系统方框图可以看出，该系统具有了检测、控制、驱动、显示、调试等功能，说明该系统完善可靠。

MC9S12XS128最小系统主控板

DC

电

机

摆杆状态检测模块

上位机

显示

蓝牙通信模块

OLED显示

电机驱动

编码器

摆杆

图2-1 系统方框图

# 理论分析与计算

**电动机选型**

方案一：采用步进电动机。

方案二：采用直流电动机。

方案论证：方案一，采用步进电动机带动旋转臂，其驱动电路与控制方式简单，但是存在丢步及连续性不好等问题；方案二，采用直流电动机带动旋转臂，其连续性明显比步进电机好，但是其驱动电路与控制方式复杂。

基于以上理论分析论证，经过多次试验后，本装置最终采用大功率带转速比的直流电机，DC-16V,转速700r/min。

**摆杆状态检测**

方案一：采用光电编码器。

方案二：采用WDD35D5角度传感器。

方案三：采用MPU-6050加速度陀螺仪。

方案论证：方案一，采用光电编码器检测可以直接测出摆杆角度，但是其体积大、精度低、仅能测出摆杆角度等问题。方案二，采用WW35D5角度传感器检测，其原理相当于可调电位器，根据摆杆旋转角度产生相应的阻值，通过AD转换处理后从而算出相应的角度，但是该方案需要AD转换、不易放置、仅能直接得到角度等问题。方案三，采用MPU-6050加速度陀螺仪模块，该模块整合了3轴陀螺仪和3轴加速度器，输出六轴旋转矩阵、四元数、Euler角的融合演算数据，通过处理后可检测到摆杆的旋转角度及其加速度，以便于对摆杆的控制。

基于以上三种方案的理论分析论证，由于方案一和方案二无法检测到摆杆的角加速度，因此本装置选用MPU-8050加速度陀螺仪模块。

**电动机驱动方案的比较与选择**

通过对电动机选型方案的论证，本装置采用直流电动机，通常直流电动机驱动方案有如下几种：

方案一：采用继电器对电动机的开或关进行控制，通过开关的切换对电机的速度进行调整。该方案电路比较简单，但是继电器的响应时间慢、机械结构易损坏、寿命较短可靠性不高。

方案二：采用MOSFET等功率管组成全桥型电机驱动电路，并利用PWM波来实现对输出电压的有效值大小和极性进行控制。该方案具有易于调速、过载能力大，能承受频繁的负载冲击，能耗小，但是硬件电路复杂，不易于控制。

方案三：采用L298专用芯片进行驱动。L298芯片的工作原理和方案二一致，但是其工作时较方案二稳定，且编程较为简单，便于调试。另外L298内部集成了两个H桥，能同时驱动两个电机，硬件实现较方案二简单。

方案四：采用BTS7960专业集成电机驱动芯片，该方案原理与方案三一样为专业集成驱动芯片，编程简单。但是相对L298芯片，该集成芯片具有导通电阻小，耐压高等优势。同时，本简易旋转倒立摆装置仅需要驱动一个电机，而L298内部集成两个H桥，存在资源浪费。

基于上述理论分析，综合考虑后，本装置电动机驱动采用方案四。

**控制算法**

根据简易旋转倒立摆装置控制要求，该系统采用经典的PID 控制算法，通过对摆杆状态的检测可以得到一个角度和角速度，根据摆杆的角度值和角速度值加入到PID算法过程中后，通过PWM以此来控制电机旋转方向与速度。其控制算法图如图3-1所示。

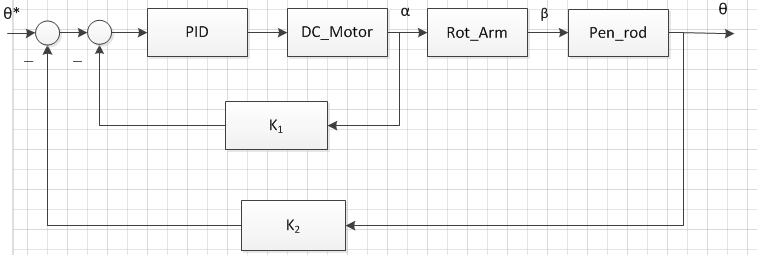


图3-1 控制算法框图

其中PID为数字式PID控制器，Rot\_Arm为旋转臂，Pen\_rod为摆杆。K1为电机转速角度反馈比例，K2为摆杆角度反馈比例。

# 系统电路设计

**总体电路框图**

根据该系统需求，本系统电路设计主要包括电源、蓝牙通信模块、拨码开关电路、最小系统板、电机驱动模块、编码器模块以及加速度计陀螺仪模块等。系统电路设计总体电路框图如图4-1所示。

电 源

最小系统板

驱动

模块

DC

电机

加速度计陀螺仪

编码器

蓝牙通信模块

拨码开关

图4-1 系统框图

**单元电路设计**

1. MC9S12XS128MAA最小系统板

最小系统板原理图如图4-2所示。

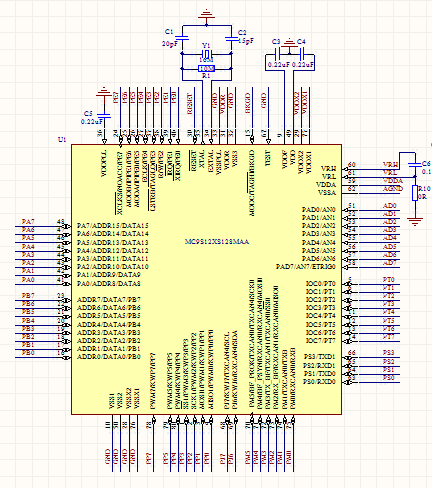


图4-2 MC9S12XS128MAA最小系统板

1. 电源模块

由于单片机等其他外围器件需要+5V电压，本系统采用LM2940CS-5.0稳压芯片，该芯片稳压性能好，输出电流大，基本无需考虑散热等问题。电路原理图如图4-3所示。

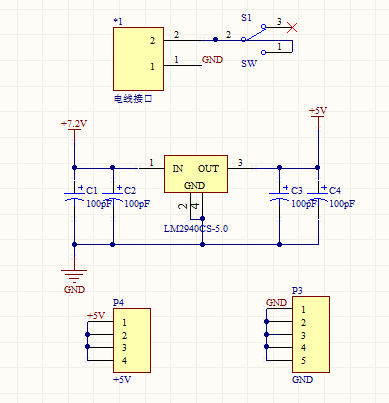


图4-3 电源模块

1. 电机驱动模块

由于一片BIS7960内部含有一个半桥驱动模块，因此电机驱动模块采用两片BTS7960搭建H桥， H桥可以实现电机无级变速与正反转，从而达到该系统的要求。电路原理图如图4-4所示。

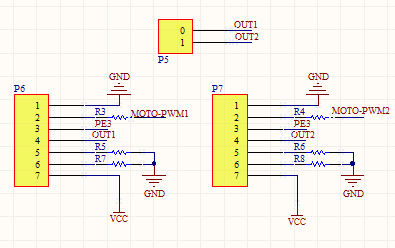


图4-4 电机驱动模块

1. 拨码开关模块

为了便于测试，本系统在硬件电路中增加了拨码开关电路，实现不同算法切换功能。本系统中使用八位拨码开关，根据拨码开关的原理，拨码开关一段连接系统板PE端口，一段全部接地。拨码开关模块原理图如图4-5所示。

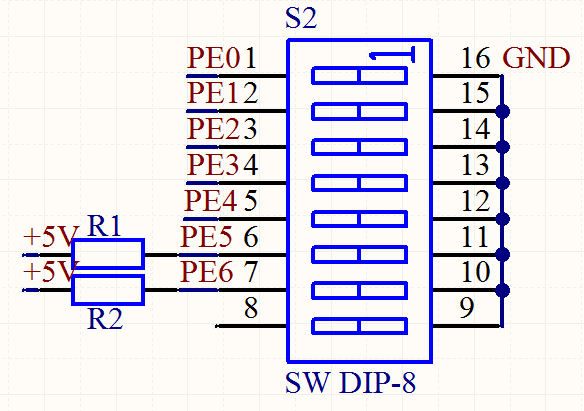


图4-5 拨码开关模块

1. 蓝牙通信模块

为了方便检验加速度计陀螺仪所采集信号准确性，查看相应波形，便于在线调试，本系统增加了一个蓝牙通信模块，蓝牙通信模块，其原理与串口通信一样，使用上位机查看相应数据与波形。蓝牙通信模块原理图如图4-6所示。

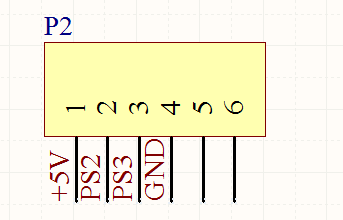


图4-6 蓝牙通信模块

1. 光电编码器模块

为了测得电机转速与方向，实现电机闭环控制，本系统采用360线双向光电编码器对电机状态进行检测。电路原理图如图4-7所示。

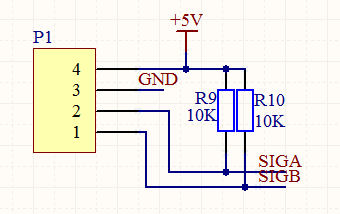


图4-7 光电编码器模块

1. 加速度计陀螺仪模块

加速度计陀螺仪模块是用于检测摆杆的状态，该模块与最小系统板I/O接口连接，其原理图如图4-8所示。

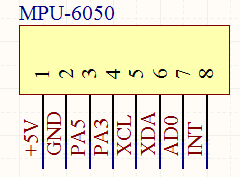


图 4-8 加速度计陀螺仪模块

**全部电路图**

最小系统板电路原理图见附录A。

整体电路图见附录B。

# 系统软件设计

**系统软件设计流程图**

1、系统软件设计流程图如图5-1所示。



图5-1 系统软件框图

**各模块软件设计**

1、电机初始化函数

#include "MOTO\_init.h"

#include "PE\_Types.h"

#include "PE\_Error.h"

#include "PE\_Const.h"

#include "IO\_Map.h"

#include "Cpu.h"

#pragma DATA\_SEG MOTO\_init\_DATA /\* Select data segment "MOTO\_init\_DATA" \*/

#pragma CODE\_SEG MOTO\_init\_CODE

#pragma CONST\_SEG MOTO\_init\_CONST /\* Constant section for this module \*/

bool MOTO\_init\_GetVal(void)

void MOTO\_init\_PutVal(bool Val)

{

if (Val) {

setReg8Bits(PTT, 0x02U); /\* PTT1=0x01U \*/

} else { /\* !Val \*/

clrReg8Bits(PTT, 0x02U); /\* PTT1=0x00U \*/

} /\* !Val \*/

}

void MOTO\_init\_ClrVal(void)

void MOTO\_init\_SetVal(void)

/\* END MOTO\_init. \*/

2、加速度陀螺仪程序

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

卡尔曼融合程序。将实测角度和积分得到的陀螺仪角度融合，

X(k|k-1)=A X(k-1|k-1)+B U(k); //由前一次最优值得到下一次估计值

P(k|k-1)=A P(k-1|k-1) A'+Q ; //由前一次最优协方差得到下次协方差估计值

//根据实测值Z(k)做卡尔曼融合得到最优值

Kg(k)=P(k|k-1)H'/[H P(k|k-1)H'+R]

X(k|k)=X(k|k-1)+Kg(k)[Z(k)-H X(k|k-1)]

P(k|k)=[I-Kg(k)H]\*P(k|k-1)

得到当前最优估计角度，并计算下次估计值，

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//输入实测值，保留估计值

//以倾角计得到角度做为实测值，

//以陀螺仪积分得到的角度融合上一次估计值得到最优值

#include "Cpu.h"

int g\_nAngleControlOut=0;

float Angle\_Adjust\_P=0;

float Angle\_Adjust\_D=0;

int Angle\_Set=0;

int Angle\_Dot\_Set=0;

//速度往前为正，角度、角速度 向后为正

int Complement\_Filter(int newAngle,int newRate);

void CalculateAngle(void);

int Complement2\_Filter(int newAngle,int newRate);

int KalmanCalculate(int newAngle,int newRate) ;

extern int Accele[3];

extern int Accele0[3];

extern int ENC03[3];

extern int ENC03\_0[3];

extern int glyro[2];

float dtc=0.006;//0.03;

int Angle\_dot=0;

int Angle\_accle=0;

int Angle=0;

void CalculateAngle()

{

// static int nPD\_Pre;

int PD;

static int Angle\_g=0;

Angle\_accle=(Accele[1]-Accele0[1])\*0.06;

Angle\_dot=(ENC03[2]-ENC03\_0[2]);

Angle\_g+=Angle\_dot\*dtc;

Angle=Complement\_Filter(Angle\_accle,Angle\_dot);

PD=(Angle-Angle\_Set)\*(Angle\_Adjust\_P)+(Angle\_dot-Angle\_Dot\_Set)\*(Angle\_Adjust\_D);

//Complement\_Filter(Angle\_accle\_x,Angle\_dot\_x)

xPD=(Angle\_x-Car\_Angle\_Set\_x)\*(Angle\_Adjust\_P\_x)+(Angle\_dot\_x-Car\_Angle\_Dot\_Set\_x)\*(Angle\_Adjust\_D\_x);

yPD=(Angle\_y-Car\_Angle\_Set\_y)\*(Angle\_Adjust\_P\_y)+(Angle\_dot\_y-Car\_Angle\_Dot\_Set\_y)\*(Angle\_Adjust\_D\_y);

zD=(Angle\_dot\_z-Car\_Angle\_Dot\_Set\_z)\*(Angle\_Adjust\_D\_z);

if(xPD>ANGLE\_CONTROLOUT\_MAX)

g\_nAngleControlOut\_x=ANGLE\_CONTROLOUT\_MAX;

else if(xPD<ANGLE\_CONTROLOUT\_MIN)

g\_nAngleControlOut\_x=ANGLE\_CONTROLOUT\_MIN;

else

g\_nAngleControlOut\_x =xPD;

if(yPD>ANGLE\_CONTROLOUT\_MAX)

g\_nAngleControlOut\_y=ANGLE\_CONTROLOUT\_MAX;

else if(yPD<ANGLE\_CONTROLOUT\_MIN)

g\_nAngleControlOut\_y=ANGLE\_CONTROLOUT\_MIN;

else

g\_nAngleControlOut\_y =yPD;

if(zD>ANGLE\_CONTROLOUT\_MAX)

g\_nAngleControlOut\_z=ANGLE\_CONTROLOUT\_MAX;

else if(zD<ANGLE\_CONTROLOUT\_MIN)

g\_nAngleControlOut\_z=ANGLE\_CONTROLOUT\_MIN;

else

g\_nAngleControlOut\_z =zD;

if(PD>ANGLE\_CONTROLOUT\_MAX)

g\_nAngleControlOut\_x=ANGLE\_CONTROLOUT\_MAX;

else if(PD<ANGLE\_CONTROLOUT\_MIN)

g\_nAngleControlOut\_x=ANGLE\_CONTROLOUT\_MIN;

else\*/

AS0\_SendChar('1');

AS0\_SendChar((unsigned char)(Accele[1]>>8));

AS0\_SendChar((unsigned char)(Accele[1]));

AS0\_SendChar('2');

AS0\_SendChar((unsigned char)(ENC03[2]>>8));

AS0\_SendChar((unsigned char)(ENC03[2]));

AS0\_SendChar('3');

AS0\_SendChar((unsigned char)(Angle>>8));

AS0\_SendChar((unsigned char)(Angle));

AS0\_SendChar('4');

AS0\_SendChar((unsigned char)(Accele[1]>>8));

AS0\_SendChar((unsigned char)(Accele[1])); /\* \*/

if(Angle\_dot<=0&&dot0>=0&&Angle>=0){

SetSpeed(-100);

n=10;

} else if(Angle\_dot>=0&&dot0<=0&&Angle<=0) {

SetSpeed(100);

n=10;

}

if((Angle0>0&&Angle<0)||(Angle0<0&&Angle>0)) {

SetSpeed(0);

}

if(n<=0){

SetSpeed(0);

} else{

n--;

}

dot0= Angle\_dot;

Angle0= Angle;

}

# 系统调试与测试

**系统调试**

为了便于系统调试，本系统使用CodeWarrior IDE进行源程序的编写、编译和链接，并最终生成可执行文件。该编译器具有高度可视化界面操作机自动恢复高度可视化操作及自动创建复杂嵌入式系统应用的功 能，包括集成开发环境 IDE、处理器专家库、全芯片仿真、可视化参数显 示工具、项目工程管理器、C 交叉编译器、汇编器、链接器以及调试器，为使用 Freescale 嵌入式产品开发提供了便利。如图6-1为该编译器调试界面

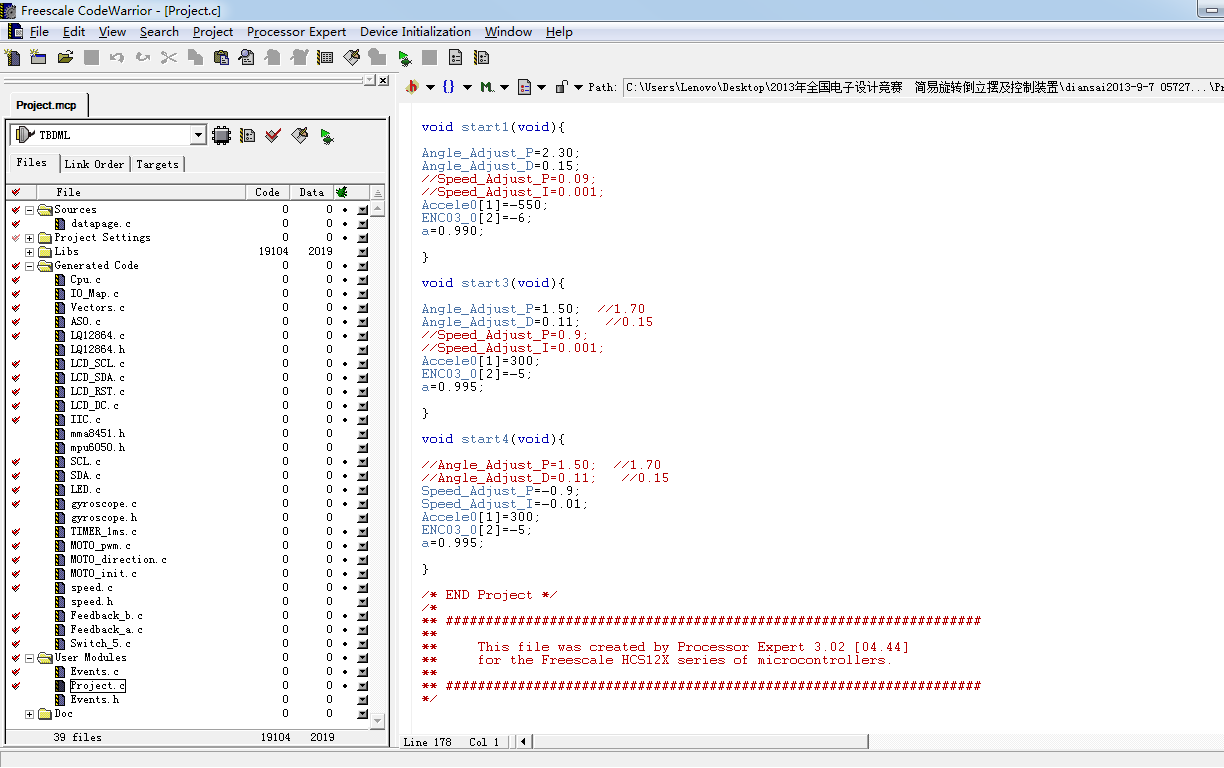
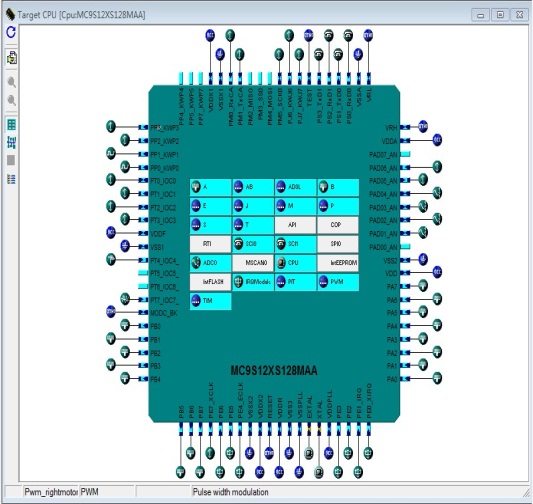
 

图6-1 编译器调试界面

同时，为了查看加速度陀螺仪波形，本系统使用了串口猎人上位机来实时的显示波形，下位机通过蓝牙模块与上位机串口通信完成。如图6-2为串口猎人上位机调试界面。

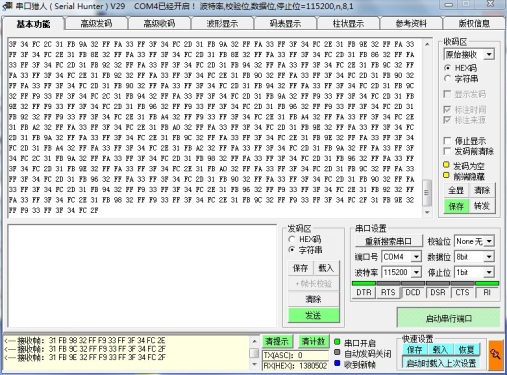
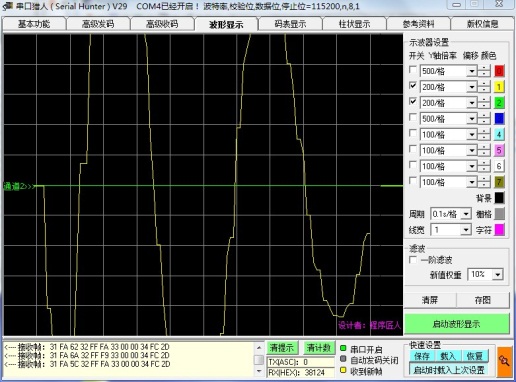
 

图6-2 上位机调试界面

**测试仪器**

1. 秒表
2. 量角器

**测试指标**

1、摆杆摆动角度

2、摆杆完成圆周运动情况

3、摆杆保持倒立时间

表格6-1为基本部分测试结果

|  |  |  |  |
| --- | --- | --- | --- |
|  | 测试项目 | 要求 | 测试结果 |
| 基本部分 | 摆杆摆动角度 | θ大于-60°~ +60° | 完成 |
| 摆杆旋转角度 | 圆周运动 | 完成，可摆动两个圆周运动以上 |
| 摆杆接近165°  时保持倒立时间 | 保持倒立时间不少于5s，旋转臂角度不大于90° | 基本完成，保持到时时间3.5s左右 |

表格6-2为发挥部分测试结果

|  |  |  |  |
| --- | --- | --- | --- |
|  | 测试项目 | 要求 | 测试结果 |
| 基本部分 | 摆杆往复旋转运动，摆杆倒立 | 摆杆保持倒立状态时间不少于10s | 基本完成 |
| 施加干扰，摆杆保持倒立状态 | 施加干扰后摆杆能保持倒立或2s内恢复 | 施加干扰后  保持倒立3s左右 |
| 摆杆保持倒立，旋转臂运动 | 旋转臂单方向运动，转过角超过360° | 完成，可以单方向运动超过360° |

**测试结果分析**

测试结果是基于多次测试后得到的数据，经过多次测试后，基本完成所有任务。有些目标尚未完全完成，测试环境对测试结果也有一定的影响，同时由于摆杆状态的不同得到结果也不同。总而言之，本次测试结果据实可靠。

# 设计总结

经过四天三夜的奋力苦战，由本队设计的简易旋转倒立摆及其控制装置基本完成题目的所有要求。但由于时间紧，任务重，系统还存在许多可以改进的地方，比如说，在机械结构方面，可以更多的尝试其他机械结构，通过多次实验可以找到更好的机械材料，构造出更好的机械结构，这方面的改进对后项控制算法和系统的设计非常有利。在电路设计方面，本队采用在赛前准备阶段制作好的通用电路板进行改造而来，如果时间允许的话，本队完成可以自行制作针对该系统的电路板，这也有助于系统性能的提升。在程序控制方面，系统算法控制和程序设计还可以进一步优化。经过在机械结构、电路设计方面和优化程序控制算法后，系统性能还有进一步提升的空间。

# 参考文献

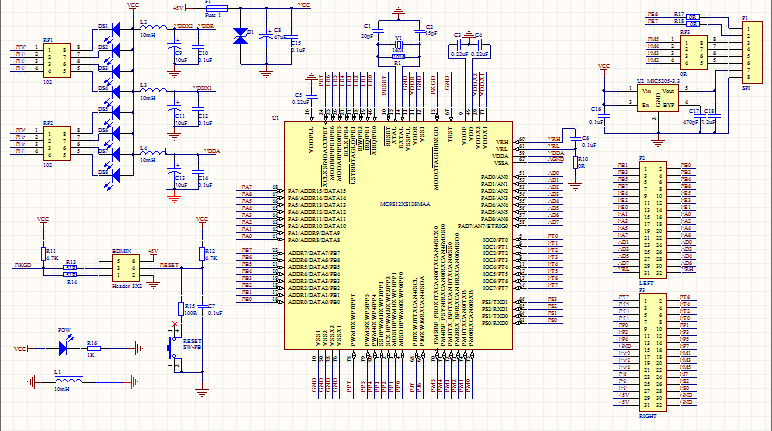
[1] 滕青芳，范多旺，董海鹰，路小娟.自动控制原理.北京:人民邮电出版社,2008.210-230

[2] Katsuhiko Ogata.现代控制工程.北京:电子工业出版社,2003

[3] 陈杰.MATLAB 宝典.北京:电子工业出版社,2007

[4] 胡寿松.自动控制原理.北京:科学出版社,2001

附录A



附录B

